keyestudio WiKi

keyestudio WiKi

Dec 06, 2023

# **KEYESTUDIO DOCS**

1	1. Description	3
2	2. Kit list	5
3	3. Tutorials	13
4	ESP32 Mainboard and ESP32 shield4.11. Keyestudio ESP32 Mainboard4.22. Keyestudio ESP32-IO shield	<b>15</b> 15 19
5	Python tutorial   5.1 1. Preparation for Python(Windows):   5.2 2. Single Sensor/Experiment Projects   5.3 3. Comprehensive Experiments: 2	<b>21</b> 21 57 205
6	Arduino tutorial26.11. Get started with Arduino C:26.22. Basic Projects36.33. Comprehensive Experiments:4	2 <b>83</b> 283 324 481
7	Arduino(Raspberry-Pi) tutorial57.11. Install Raspberry Pi OS System57.22. Preparations for C language:67.33. Linux SystemRaspberry Pi:67.44. How to Add Libraries? :67.55. Basic Projects6	5 <b>71</b> 571 512 517 534 541
	7.6   6. Comprehensive Projects:   7.7	/95



## **1. DESCRIPTION**

The Keyestudio ESP32 42 in 1 sensor kit mainly contains 42 commonly usedsensors/modules, a ESP32 board, a ESP32 expansion board and Dupontwires.

The 42 sensors and modules are fully compatible with the ESP32 ExpansionBoard. You only need to stack the ESP32 mainboard onto the ESP32 Expansion Board, and hook up them with Dupont wires, which is simple and convenient.

To make you master the electronic knowledge, detailed tutorials (Micropython), schematic diagrams, wiring methods and test code are included. Through these projects, you will have a better understanding about programming, logic and electronics.

## TWO

# 2. KIT LIST



(Note: KS5003 kit include ESP32 mainboardKS5004 kit does't include ESP32 mainboard.)

continues on next page

#	Picture	Name	QTY
6		Keyestudio Button Module	1
7	Tilt switch	Keyestudio Tilt Sensor	1
8		Keyestudio PIR Motion Sensor	1
9		Kevestudio Obstacle Avoidance Sensor	1
10		Kayastudio 6812 PCP Modulo	1
11	Temperature	Keyestudio NTC-ME52AT Thermistor	1
11	Photoresistance		1
12		Keyestudio Photoresistor	1 t page

#	Picture		Name	QTY
12	Microphone			
13			Keyestudio Sound Sensor	1
14	Potentiometer	Ì	KeyestudioRotary Potentiometer	1
15	))))   III) IIIII IIIIII		Kevestudio IR Receiver	1
15	Reed Switch		Keyestudio Reed Switch Sensor	1
17	CLK DT SW Rotary ercoder		Keyestudio Rotary Encoder Module	1
18			Keyestudio Joystick Module	1
19			Keyestudio HT16K33 8X8 Dot Matrix Module	1
20			Keyestudio TM1650 4-Digit Tube Display	1
-			continues on next	page

#	Picture	Name	QTY
21		Keyestudio Thin-film Pressure Sensor	1
22		Kevestudio DS1307 Clock Sensor	1
			-
23		Keyestudio SR01 Ultrasonic Sensor	1
24	Touch	Keyestudio Capacitive Sensor	1
26	Photo Interrupter	Kevestudio Photo Interrunter	1
27	Hall maynetic	Kavastudio Hall Sanger	1
21		Regestudio Hall Sensor	1
28		Keyestudio Flame Sensor continues on ne	1 ext page

Table 1 – continued from previous page

#	Picture	Name	QTY
29	Line Tracking	Kevestudio Line Tracking Sensor	1
30		Kevestudio Analog Gas Sensor	1
50		Teyestudio Tilulog Gus Gensor	I
31		Temperature and Humidity Sensor	1
32	Temperature	Keyestudio 18B20 Temperature Sensor	1
33		keyestudio 130 Motor	1
34		Fan	1
35	Laser	Keyestudio Laser Module	1
_		continues on nex	kt page

#	Picture	Name	QTY
36	Steam sensor	Kevestudio Steam Sensor	1
27		Konstalia Dalan Madala	1
37			1
38		Keyestudio KFID Module	1
39	Consult	Keyestudio Collision Sensor	1
40		Keyestudio Alcohol Sensor	1
41		Kustalia I CD 128V22 DOT Madula	1
41		S Charal AD D was Malak	1
42		DXI 345 Acceleration Module	1
44		Kauastudio ESD22 Dausloomoot Doord	1
44	••••••••••••••	continues on per	I t page

Table 1	- continued	from	previous	page
---------	-------------	------	----------	------

#	Picture	Name	OTY
45	ESP-32 keyestudio	Keyestudio ESP32-IO Expansion Board	1
46		Keyestudio IR Remote Control	1
47		USB Cable	1
48		F-F Dupont Wire	1
49		White Card	1
50		ABS RFID Key	1
	1 • • • • • • • • • • • • • • • • • • •		
51		Battery Holder	1

## THREE

## 3. TUTORIALS

- 1.ESP32\_Mainboard\_and\_ESP32\_Expansion\_Board
- 2.Python\_Tutorial(Windows)
- 3.Arduino\_C\_Tutorial(Windows)
- 4.Arduino\_C\_Tutorial(Raspberry-Pi)
- 5.Libraries\_Driver\_Firmware\_and\_APP
- 6.Codes

FOUR

## **ESP32 MAINBOARD AND ESP32 SHIELD**

## 4.1 1. Keyestudio ESP32 Mainboard



## 4.1.1 1.1. Introduction:

Keyestudio ESP32 Core board is a Mini development board based on the ESP-WROOM-32 module. The board has brought out most I/O ports to pin headers of 2.54mm pitch. These provide an easy way of connecting peripherals according to your own needs.

When it comes to developing and debugging with the development board, the both side standard pin headers can make your operation more simple and handy.

The ESP-WROOM-32 module is the industry's leading integrated WiFi + Bluetooth solution with less than 10 external components. It integrates antenna switches, RF balun, power amplifiers, low noise amplifiers, filters as well as power management modules. At the same time, it also integrates TSMC's low-power 40nm technology, power performance and rf performance, making it safe, reliable and easy to expand to a variety of applications.

## 4.1.2 1.2. Specifications:

- Microcontroller: ESP-WROOM-32Module
- USB to Serial Port Chip: CP2102-GMR
- Working Voltage: DC 5V
- Working Current80mAAverage
- Current Supply500mAMinimum
- Working Temperature Range: -40°C ~ +85°C
- WiFi ModeStation/SoftAP/SoftAP+Station/P2P
- WiFi Protocol802.11 b/g/n/e/i802.11nSpeed up to 150 Mbps
- WiFi Frequency Range2.4 GHz ~ 2.5 GHz
- Bluetooth Protocolconform to Bluetooth v4.2 BR/EDR and BLE Standard
- Dimensions55×26×13mm
- Weight9.3g

## 4.1.3 1.3. Pin out:



ESP32 has fewer pins than commonly used processors, but it doesn't have any problems reusing multiple functions on pins.

Warning: The pin voltage level of the ESP32 is 3.3V. If you want to connect the ESP32 to another device with an operating voltage of 5V, you should use a level converter to convert the voltage level.

**Power Pins**: The module has two power pins +5V and 3.3V. You can use these two pins to power other devices and modules.



GND PinsThe module has three grounded pins.

**Enable pin** (**EN**): This pin is used to enable and disable modules. The pin enables module at high level and disables module at low level.

**Input/Output pins (GPIO)**: You can use 32 GPIO pins to communicate with LEDs, switches and other input/output devices. You can also pull these pins up or down internally.

**Note**: GPIO6 to GPIO11 pins (SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD pins) are used for SPI communication for the internal module, which are not recommended.

**ADC**: You can use the 16 ADC pins on this module to convert analog voltages (the output of some sensors) into digital voltages. Some of these converters are connected to internal amplifiers and are capable of measuring small voltages with high accuracy.

DAC: ESP32 module has two A/D converters with 8-bit precision.

**Touch pad**: The ESP32 module has 10 pins that are sensitive to capacitance changes. You can attach these pins to certain pads (pads on a PCB) and use them as touch switches.

**SPI**: There are two SPI interfaces on the module, which can be used to connect the display screen, SD/microSD memory card module as well as external flash memory, etc.

I2C: SDA and SCL pins are used for I2C communication.

**Serial Communication (UART)**: There are two UART serial interfaces on this module, which can be used to transfer up to 5Mbps of information between two devices. The UART0 also has CTS and RTS control functions.

**PWM**: Almost all ESP32 input/output pins can be used for PWM (pulse-width modulation). Using these pins can control the motor, LED lights and colors, etc.

### 4.1.4 1.4. Components:



## 4.2 2. Keyestudio ESP32-IO shield



## 4.2.1 2.1. Overview:

Keyestudio ESP32-IO Expansion Board is designed to be compatible with the Keyestudio ESP32 Core Board (KS0413), which leads all pin connections of the ESP32 Core Board using a row of pins spaced 2.54mm apart. To facilitate the connection of other sensors, it also has two rows of pins with a spacing of 2.54mm rows, which are used to supply 3.3V DC power for external sensors/modules.

A power supply circuit is designed on the control board as it seeks to power the Keyestudio ESP32 Core Board easily. You only need to input DC 6-9V voltage on the black DC head to power it. In addition, it also has a DIP switch to control the power switch.

## 4.2.2 2.2. Specifications:

- Voltage SupplyDC 6-9V
- Operating Current60mA
- Maximum Power0.3W
- Working Temperature-25°C to +65°C
- Dimensions30mm\*20mm
- Environmental Protection AttributesROHS

## 4.2.3 2.3. Pins and Components:



## **PYTHON TUTORIAL**

## 5.1 1. Preparation for Python(Windows):

### 5.1.1 1. Download and Install Thonny

Thonny is a free and open source software platform with small size, simple interface, simple operation and rich functions. It is a Python IDE suitable for beginners. In this tutorial, we use this IDE to develop a ESP32. Thonny supports multiple operating systems including Windows, Mac OS, Linux.

#### 1.1. Download Thonny

- 1). Enter the websitehttps://thonny.org to download the latest version of Thonny.
- 2). Thonny open-source code libraryhttps://github.com/thonny/thonny.



#### 1.2. Install Thonny (Windows System):

1). The downloaded Thonny icon is as follows:



thonny-3.3.13.exe

```
2). Double-click"thonny-3.3.13.exe"and select install mode. You can choose
```

## → Install for me only (recommended)



3). You can also keep selecting Next to finish install.

4	Setup - Thonny	_		$\times$
	📕 D 🏇 💮 🔍 🤇			
		Welcome to using Thonny!		
uess.	py			
1	import Candom	This without will install Thomps 2.2.12 for all users		
2	n = ran randir	This wizard will install moniny 5.5.15 for all users.		
3	U U.U.			
4	guess = int(input			
6	while n l= "quess			
7	if guess < n:			
8	print("gu			
9	guess = i			
10	<pre>elif guess &gt;</pre>			
11	print("gu			
12	guess = i			
13	else:			
14	print("yc			
			0	
		Next	Cance	1



4). If you want to change the route of installing Thonnyjust click"**Browse...**"to select a new route and click **OK**.

🛃 Setup - Thonny	_		×
Select Destination Location Where should Thonny be installed?			Th
Setup will install Thonny into the following folder.			
To continue, click Next. If you would like to select a different folder,	click Bro	wse.	_
C:\Program Files (x86)\Thonny	Bre	owse	
2 At least 99.5 MB of free disk space is required.			
<u>B</u> ack <u>N</u> e	xt	Car	ncel

Setup - Thonny	_		×
Select Start Menu Folder Where should Setup place the program's shortcuts?			Th
To continue, click Next. If you would like to select a different folder, c	rt Menu lick Brov	ı folder. wse.	
Thonny	Bro	owse	
<u>B</u> ack <u>N</u> ext	t	Car	ncel

5). Click **Create desktop icon,** you will view Thonny on your desktop.

🛃 Setup - Thonny			×
Select Additional Tasks Which additional tasks should be performed?			Th
Select the additional tasks you would like Setup to perform while i then click Next.	nstalling Th	ionny,	
Create desktop icon			
	0		
Back	<u>N</u> ext	Ca	ancel

6). Click"Install"



#### 7). Wait for a while but don't click **Cancel**

🛃 Setup - Thonny	_		$\times$
<b>Installing</b> Please wait while Setup installs Thonny on your computer.			Th
Extracting files C:\Program Files (x86)\Thonny\Lib\idlelib\rpc.py			
		Ca	ancel

#### 8). Click "Finish"





#### 1.3. Basic Setting

Double-click Thonny, choose lanuage and initial settings and click Let's go

P		_		$\times$
Th	Language: Englis Initial settings: Standa	h ard Let	'≤ go!	



Click"View"→"File"and"Shell"





## 5.1.2 2. Install the CP2102 driver

Before using the Thonny, we need to install the CP2102 driver in the computer.

#### Windows system

Check if the CP2102 driver has been installed.

1). Interface the ESP32 with your PC with a USB cable.



2). Click "This PC" and right-click "Manage".

Contraction Recycle Bin		
11111	<b>Open</b> Pin to Quick access	
DATE D	Manage Din to Start	
Contr	Map network drive Disconnect network drive	
Pane	Create shortcut Delete	
a h Thonny	Properties	

3). Click "Device Manager", if the CP2102driver has been installed "Silicon Labs CP210x USB to UART Bridge(COMx)" will be shown.



If the CP2102 has not been installed.



Click "CP2102USB to UART Bridge Controller" and "Update driver".
🔝 Computer Management		_		×
File Action View Help				
← → 2 □ □ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0				
Image: Computer Management (Local       ✓       Image: DESKTOP-980K7TG         ✓       Image: System Tools       >       Image: Audio inputs and outputs         >       Image: Computer       >       Image: Computer         >       Image: Event Viewer       >       Image: Disk drives		Actions Device Mor	lanager re Action:	5
<ul> <li>Shared Folders</li> <li>Shared Folders</li> <li>Local Users and Groups</li> <li>Performance</li> <li>Device Manager</li> <li>Device Manager</li> <li>Storage</li> <li>Disk Management</li> <li>Services and Applications</li> <li>Jusplay adapters</li> <li>Display adapters</li> <li>Jusplay adapters</li> <li>J</li></ul>				
CP2102 USB to UART Bridge Controller	are change	:5		
Launches the Update Driver Wizard for the selected device.				

Click "Browse my computer for drivers".



Click "Browse... to" choose "CP2102 Driver File-Windows" and click "Next".



The CP2102 driver will be installed.



### 5.1.3 3. Burn Micropython firmware:

To run a Python program on the ESP32 board, we need to burn the firmware to the ESP32 board first.

Download Micropython firmware microPython websitehttp://micropython.org/

ESP32 firmwarehttps://micropython.org/download/esp32/

## Firmware

Releases

v1.18 (2022-01-17) .bin [.elf] [.map] [Release notes] (latest) v1.17 (2021-09-02) .bin [.elf] [.map] [Release notes] v1.16 (2021-06-23) .bin [.elf] [.map] [Release notes] v1.15 (2021-04-18) .bin [.elf] [.map] [Release notes] v1.14 (2021-02-02) .bin [.elf] [.map] [Release notes] v1.13 (2020-09-02) .bin [.elf] [.map] [Release notes] v1.12 (2019-12-20) .bin [.elf] [.map] [Release notes]

Nightly builds

```
v1.18-121-gd8a7bf83c (2022-02-10) .bin [.elf] [.map]
v1.18-107-gaca40127b (2022-02-09) .bin [.elf] [.map]
v1.18-105-gada836b83 (2022-02-08) .bin [.elf] [.map]
v1.18-103-g6f7d6c567 (2022-02-08) .bin [.elf] [.map]
```

# Firmware (Compiled with IDF 3.x)

### Releases

v1.14 (2021-02-02) .bin [.elf] [.map] [Release notes] (latest) v1.13 (2020-09-02) .bin [.elf] [.map] [Release notes] v1.12 (2019-12-20) .bin [.elf] [.map] [Release notes] v1.11 (2019-05-29) .bin [.elf] [.map] [Release notes] v1.10 (2019-01-25) .bin [.elf] [.map] [Release notes] v1.9.4 (2018-05-11) .bin [.elf] [.map] [Release notes]

The firmware we useesp32-20210902-v1.17.bin

Download firmwarehttps://micropython.org/resources/firmware/esp32-20210902-v1.17.bin

Wo also offer the Firmware"...\Python\_Firmware".

Python\_Firmware

Share View			
K Python_Firmware			~
Name	Date modified	Туре	Size
esp32-20210902-v1.17.bin	2/10/2022 11:23 AM	BIN File	1,492 KB

#### Burn the Micropython firmware

Connect the ESP32 to your PC with a USB cable



Make sure the driver has been installed successfully and the COM port can be identified correctly. Open Device Manager and expand "Ports".



Open Thonnyclick"run"and"Select interpreter..."



Select "Micropython (ESP32)" and "Silicon Labs CP210x USB to UART Bridge(COM3)" and click "Install or update firmware".

🕞 Thonn	y options								×
General Which	Interpreter	Editor	Theme & Font should Thonny u	Run & Debug ise for running v	Terminal	Shell	Assistant	0	
Deta Con Con (loo If yo Con If yo (imp < W Port	ils necting via U nect your de k for your de ou can't find i necting via V our device sup oort webrepl_ ebREPL > be	ISB cable vice to th vice nam t, you ma t, you	: ie computer and ie, "USB Serial" o ay need to instal (EXPERIMENTAL ebREPL, first cor onnect your cor	select correspo r "UART"). I proper USB driv .): .nect via serial, r nputer and devia	nding port ver first. nake sure V ce to same	below VebREP networl	L is enablec k and selec	d t 2	
Silic	on Labs CP2	10x USB t	o UART Bridge (	COM3)			Insta	3 Il or update fi	irmware
								ОК	Cancel

Select"Silicon Labs CP210x USB to UART Bridge(COM3)"click "Browse…"and choose the firmware **esp32-20210902-v1.17.bin.** Check"Erase flash before installing"and"Flash mode"then click"Install".Noteif you fail to install the firmwarepress the Boot button on the ESP32 board and click"Install"

<b>P</b> 50000 C	1 H			~
the ESP32 firm	nware installer			×
This dialog If you need Note that t at micropy alternative	allows installing or updating to set other options, then ple there are many variants of Mic thon.org/download doesn't v s look around in your devic	firmware on ESP32 using the r ease use 'esptool' on the comr croPython for ESP devices. If the vork for your device, then ther e's documentation or at Micro	most common setting mand line. he firmware provided re may exist better oPython forum.	js.
Port	Silicon Labs CP210x USB to U	ART Bridge (COM3)	∼ Reload	
Firmware	rted with Python/Python_Firm	nware/esp32-20220117-v1.18.	bin Browse	
-Flash mo	de			
From in	mage file (keep) 🔿 Quad I/O	(qio)		
O Dual I/	O (dio) 🛛 🔿 Dual Outp	out (dout)		
✓ Erase fl	ash before installing			
			Install Cance	1
The Open				×
← → • ↑	• • Python_Firmware		<b>∨ ບັ</b> 🔎 S	earch Python_Firmware
Organize 🔻 🛛 N	lew folder			::: • 🔟 🕐
lesson on e Drive	^ Name	Date modified	Type Siz	e
💻 This PC	esp32-20210902-v1.17.bin	2/10/2022 11:23 AM	BIN File	1,492 KB
🧊 3D Objects				
📃 Desktop	~			2
	File name: esp32-20210902-v1.17.bin		∽ bin-fil	×
			0	pen Cancel

🖡 ESP32 firr	nware installer	$\times$						
This dialog allows installing or updating firmware on ESP32 using the most common settings. If you need to set other options, then please use 'esptool' on the command line. Note that there are many variants of MicroPython for ESP devices. If the firmware provided at micropython.org/download doesn't work for your device, then there may exist better alternatives look around in your device's documentation or at MicroPython forum.								
Port Firmware	ilicon Labs CP210x USB to UART Bridge (COM3) V Reload							
Flash mo From in Dual I/ Erase fl	de mage file (keep) () Quad I/O (qio) O (dio) () Dual Output (dout) ash before installing							
	Install							

Then click "Close" and "OK".

NoteDuring installation, you can press and hold the Boot button on the ESP32 mainboard. When the upload progress percentage appears, release the button for a while to complete the installation.



🖡 ESP32 fire	mware installer	$\times$
This dialog If you nee Note that at micropy alternative	g allows installing or updating firmware on ESP32 using the most common settings d to set other options, then please use 'esptool' on the command line. there are many variants of MicroPython for ESP devices. If the firmware provided ython.org/download doesn't work for your device, then there may exist better es look around in your device's documentation or at MicroPython forum.	5.
Port	Silicon Labs CP210x USB to UART Bridge (COM3) V Reload	
Firmware	class for Python/Python_Firmware/esp32-20210902-v1.17.bin Browse	
Flash mo From i Dual I/	de mage file (keep) 〇 Quad I/O (qio) 'O (dio)     O Dual Output (dout)	
🗹 Erase f	lash before installing	
	Writing at 0x00031955 (9 %) Install Cancel	

🖡 ESP32 firr	nware installer	$\times$						
This dialog If you need Note that at micropy alternative	g allows installing or updating firmware on ESP32 using the most common settings d to set other options, then please use 'esptool' on the command line. there are many variants of MicroPython for ESP devices. If the firmware provided /thon.org/download doesn't work for your device, then there may exist better is look around in your device's documentation or at MicroPython forum.							
Port	Port Silicon Labs CP210x USB to UART Bridge (COM3)							
Firmware	class for Python/Python_Firmware/esp32-20210902-v1.17.bin Browse							
<ul> <li>Flash mo</li> <li>From i</li> </ul>	de mage file (keep) 🔿 Quad I/O (gio)							
O Dual I/	O (dio) O Dual Output (dout)							
☑ Erase fl	ash before installing							
<u> </u>	Done! Install Close							

<b>Շ</b> ել Մի	onny									_		$\times$
File	ቩ Thonn	y options								$\times$		
File Files This ⊕	General Which Microl Detai Con (lool If yo Con If yo	y options Interpreter interpreter o Python (ESP3 Is necting via U nect your dev k for your dev u can't find it necting via W ur device sup	Editor r device 2) ISB cable vice to th vice nam t, you m VebREPL oports W	Theme & Font should Thonny u secomputer and le, "USB Serial" o ay need to install (EXPERIMENTAL ebREPL, first con	Run & Debug use for running y select correspo r "UART"). I proper USB dri .): nect via serial, r	Terminal your code? nding port ver first. make sure \	Shell A below WebREPL is	s enabled		×		
	(Imp < W Port Silic	ort webrepl ebREPL > bel or WebREPL on Labs CP2	setup), c low	onnect your con	COM3)	ce to same	network a	Install or	update firm	ware Cancel	\pythor	∧ v n.exe
											6	

Turn off all windows and turn to the main page and click <sup>•••</sup> "STOP".

The Thonny	—		$\times$
File Edit View Run Tools Help			
🗋 🗃 📕 💿 🚸 👁 🗈 🖈 🖿 🗰 🛑 STOP Button			
Files ×			
This computer = ^			
🗉 🛁 Windows10 1909 (C:)			
🗉 🚔 Software (D:)			
(Poot directory			
Inter space of ESP32			
used to save files.			
MicroPython device Shell ×			
Spoot.py	with	ESP3	2
Type "help()" for more information.	77 L C11	201.0	-
v >>>			¥
	MicroPy	thon (ES	SP32)

### 5.1.4 4. Test Code:

#### Test the Shell commander

Input print('hello world') in the"Shell"and press "Enter".

The Thonny		_		$\times$
File Edit View Run Tools	Help			
🗋 🚰 📓 🔘 🎋 🔿 🕄 .	n 🕪 🥯			
Files $\times$				
This computer ≡	<b>^</b>			
MicroPython device =				
😓 boot.py	Shell ×			
	lype "help()" for more information.			
	>>> burue( Hello mould )			
	Hello World			
	>>>			~
		MicroP	vthon (ES	P32)

### Run the test code(online):

Connect the ESP32 to your PC. Users can program and debug programs with Thonny.

Open Thonny and click "**Open**".

The Thonny	— [	- ×
File Edit View Run Tools Help		
□ 📴 🔲 🚸 🖘 3		
This computer = ^		
🗉 📾 Software (D:)		
MicroPython device =		
boot.py		
Type "help()" for more information.		^
>>> print('Hello World')		
Hello World		
	MicroPyth	on (ESP32)

When a new window pops up, click "This computer".



Select the file "lesson\_01\_HelloWorld.py".

🏹 Open											$\times$
← → ~ ↑	$\leftarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ ESP32_Python_code(Windows) $\rightarrow$ lesson 1. Hello World $\checkmark$				ū	: م	Search le	sson 1. H	lello Wo	orld	
Organize 🔻 🛛 N	ew folder	r									?
OneDrive	^	Name	^	Date modifie	d Ty	pe		Size			
This PC		Th lesson_01_He	lloWorld	10/15/2021 8:	30 AM Th	onny			1 KB		
3D Objects			- <b>N</b>	1							
E. Desktop	~							12			
	File nar	me: lesson_01_Hel	loWorld			~	Pytho	ontiles			~
							0	Open		Cancel	

Click O, "Hello World" will be printed in the "Shell" monitor.



Note: Press the reset button to reboot

#### Run the test code(offline):

After rebooting the ESP32, run the boot.py file under the root directory first then run your code file.

So, we need to add a guide program to run the code of users.

Move the file "**ESP32\_code\_MicroPython(Windows**)" to the disk(D)the route is "D:/ESP32\_code\_MicroPython(Windows)".



Click lesson 00. Boot and double-click boot.py, then the code under MicroPython device can run offline.



If you want to run the code offline, you nee to upload boot.py and program code to MicroPython device, then press the ESP32's reset button. We will take the lesson 00 and lesson 01 as an example. Select "**boot.py**" and right-click "**Upload to**/".



Thonny - D:\2. ESP32_code_MicroPython\lesson 00. Boot\boot.py @ 30 : 14         -         -         ×					
File Edit View Run To	File Edit View Run Tools Help				
D 🗃 🖩 🛛 🚸 🧔	3 10	TOP			
Files ×		boot.py $\times$			
This computer D: \ ESP32_Python_code(W	≡ ^ Vindows)	1 #!/0 2 <b>imp</b>	opt/bin/lv_micropython ort uos as os	^	
🗆 💫 Iesson 00. Boot		3 <b>imp</b>	ort uerrno as errno		
boot.py     lesson 01. Hello     lesson 02. LED     lesson 03. Traffi	Open in Thonny Open in system d Configure .py file	lefault app s	<pre>&gt; = os.ilistdir() )IR = 0x4000 LEGULAR = 0x8000</pre>		
🗄 💫 lesson 04. Laser 🛛 🛛	Upload to /		.e True:	¥	
Iesson 05. Breat	Move to Recycle New directory	Bin	on v1 17 on 2021 00 02, ESD22 modulo with ESD22		
boot.py	Properties		p()" for more information.		
	~	>>>		~	
			MicroPython (ES	P32)	

Thonny - D:\2. ESP32_code_MicroPythe	n\lesson 00. Boot\boot.py @ 30 : 14	- 🗆	$\times$
File Edit View Run Tools Help			
🗋 😂 📓 🔕 🌸 👁 3e 🕨	3TDP		
Files $\times$	boot.py $ imes$		
This computer = D: \ESP32_Python_code(Windows)	<pre>1 #!/opt/bin/lv_micropython 2 import uos as os</pre>		^
🖃 🔑 lesson 00. Boot 🤯 Uploa	ding ×		
	×		
E Die			
This operation will overv	rite		~
/boot.py' with a larger f	ile (139 B + 768 B) Cancel		
MicroF	odule	e with ESP3	2 ^
🤣 t 🛛 🔿 K	Cancel		
			~
		MicroPython (8	SP32)

Similarly, upload the lesson 01.HelloWorld.py file to the "MicroPython device".

Thonny - D:\2. ESP32_code_MicroPython\lesson 01. Hello World\lesson_01_HelloWorld.py @ 3 : 1 — 🗆 🗙				
File Edit View Run Tools Help	esson 01 HelloWorld.nv ×			
This computer       ■         D: \ESP32_Python_code(Windows)         Image: the second seco	<pre>1 print("Hello World!") 2 print("Welcome Keyestudio") 3</pre>			
Iesson_01_HelloWorld.py     0          ⊕ ↓ Iesson 02. LED       0          ⊕ ↓ Iesson 03. Traffic_Light       0          ⊕ ↓ Iesson 04. Laser sensor       0	pen in Thonny pen in system default app onfigure .py files			
	ove to Recycle Bin			
MicroPython device	operties on 2021-09-02; ESP32 module with ESP32 A			
	→→→→ boot.py has been upload here.			
	MicroPython (ESP32)			

Press the Reset button, you will view code running in the "Shell" monitor.



### 5.1.5 5. Thonny Common Operation:

#### Upload the code to the ESP32

We take the boot.py as an example. If we add a "**boot.py**" in each code directory, reboot the ESP32, the boot.py will run first.



Select "**boot.py**"in the file lesson 02. LED, right-click to select "**Upload to** /". Then the code will be uploaded to the root directory of the ESP32 and click "**OK**".





#### Download the code to your PC

MicroPython device "boot.py", then right-click "Download to...".



#### **Delete files of the ESP32**

For example, click "boot.py" in the MicroPython device and right-click "Delete".



Select "boot.py" in the lesson 02. LED folder, right-click "Move to Recycle Bin" to delete it.

Thonny - D:\2. ES	The Thonny - D:\2. ESP32_code_MicroPython\lesson 02. LED\boot.py @ 6:1 ×			×		
File Edit View Ru	n Iools Help					
🗋 💕 🖩 ၊ 🔕	🔿 3e 🕨	3709				
Files $\times$		boot.py $\times$				
This computer D: \ESP32_Python_co	de(Windows)	1 # T 2 #im	his file is executed on every boot (i port esp	including	wake-ł	^
🗉 🕖 Iesson 00. Boot		3 #es	p.osdebug(None)			
🕀 🕖 lesson 01. Hello	World	4 #im	port webrepl			
🖃 💫 lesson 02. LED		5 #we	brepl.start()			
boot.py	Open in Thonny		1.			
Plesson_2_B	Open in system of	default app	ort os			
🖭 👘 lesson os. man	Configure .py file	es	len(files) >= 2			
MicroPython device	Upload to /		print('The device have %d files'%ler	n(files))		
	Move to Recycle	Bin	<pre>for i in range(len(files)):</pre>			<b>.</b>
	New directory					
	Properties					
		_				^
		MicroPyt	hon v1.17 on 2021-09-02; ESP32 module	e with ESP	32	
		Type "he	lp()" for more information.			
	Ý	>>>		Mine Di	AL	×
				witcropy	thon (ESF	·52)

#### Create and save code

Click "File"  $\rightarrow$  "New" to create and edit code.

🔚 Thonny			-		×
File Edit View Rur	n Tools Help				
🗋 New	Ctrl+N	<b>.</b>			
💕 Open	Ctrl+O	1-			
Recent files	+				
Close	Ctrl+W	= ^			
Close all	Ctrl+Shift+W				
Save	Ctrl+S				
Save All files	Ctrl+Alt+S				
Save as	Ctrl+Shift+S				
Save copy					
Move / rename					
Print	Ctrl+P				
Exit	Alt+F4				
		$\sim$			
MicroPython device		= ^			
			Shell ×		^
			MicroPython v1.17 on 2021-09-02; ESP32 module wit	h ESP3	2
			Type "help()" for more information.		
		$\sim$	>>>		¥

Enter the code in the new file. We take the lesson 02. LED.py as an example.

🏹 Thonny - <untitled> @ 9:28</untitled>	— —	$\times$
File Edit View Run Tools Help		
🗋 💕 🖬 🛛 🐐 🗇 3e 🕨		
Files $\times$	<ul> <li><untilled> * ×</untilled></li> </ul>	
This computer = /	1 from machine import Pin	^
	2 import time	
🕀 🛁 Software (D:)	3	
	4 led = Pin(0, Pin.OUI)# Build an LED object, connect the	
	<pre>6 led.value(1)# turn on led</pre>	
	7 time.sleep(1)# delay 1s	
	<pre>8 led.value(0)# turn off led</pre>	
MicroPython device	9 time.sleep(1)# delay 1s	
A boot my		
- boot.py		~
	Shell ×	
		^
	MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32	
	Type "neip()" for more information.	
	MicroPvthon (ESF	× 32)

Thonny - <untitled> @ 9:28  $\times$ File Edit View Run Tools Help 🔍 🎋 🔿 R. it 🕨 500 H Files <untitled>  $* \times$ This computer = from machine import Pin 1 import time 2 🕀 📄 Windows10 1909 (C:) 🗉 📄 Software (D:) d an LED object, connect the The Where to save to?  $\times$ ed This computer led MicroPython device = boot.py MicroPython device Shel Δ MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>> v MicroPython (ESP32)

Click  $\blacksquare$  to save the code to your PC or the ESP32.

Select MicroPython device and enter main.py in the new page and click "OK".

The Thonny - <untitled> @ 14:1</untitled>		– 🗆 X
File Edit View Run Tools H	elp	
🗋 💕 🖬 🛛 🐐 👁 3 i		
Files $\times$	😱 Save to MicroPython device	×
This computer		^
🗉 🛁 Windows10 1909 (C:)	MicroPython device	= ^
🕀 🛁 Software (D:)	Name	Size (bytes)
	net state and the second secon	139
	1	2 ×
MicroPython device		
🤣 boot.py	File name: main.py	OK Cancel
	man or y more taran or would be	P32
	Type "help()" for more info	rmation.
		MicroPython (ESP32)

Then the code will be uploaded to the ESP32.



Disconnect the USB cable and connect it, you can see the effect of the LED flashing continuously in the circuit on a cycle.



# 5.2 2. Single Sensor/Experiment Projects

When we get the kit, we can see that there are 42 sensors/modules in the kit, which contain the corresponding ESP32 mainboard, ESP32 Expansion Board and wirings. Here, we will connect the 42 sensors individually to the ESP32 mainboard and the ESP32 Expansion Board using a wiring. Then run the corresponding test code to test the function of each sensor separately. Our next lesson is to study the principles of individual modules/sensors from simple to complex as well as some extended applications of sensors to consolidate and deepen our understanding of the kits.

Note : When connecting the module/sensor wirings in the experiment, the wiring method and position must be followed in the document. What's more, do not misconnect the power supply and signal pin, otherwise there may be no experimental results or damage to the modules/sensors.

### 5.2.1 Project 1: Hello World

#### Overview

For ESP32 beginners, we will start with some simple things. In this project, you only need a ESP32 mainboard, a USB cable and a computer to complete the "Hello World!" project, which is a test of communication between the ESP32 mainboard and the computer as well as a primary project.

#### Components



#### Wiring Diagram

In this project, we will use a USB cable to connect the ESP32 to a computer.



#### **Running code online**

To run the ESP32 online, you need to connect the ESP32 to the computer, which allows you to compile or debug programs using Thonny software.

Advantages:

1). You can use the Thonny software to compile or debug programs.

2).Through the "Shell" window, you can view error messages and output results generated during the running of the program as well as query related function information online to help improve the program.

Disadvantages:

1). To run the ESP32 online, you must connect the ESP32 to a computer and run it with the Thonny software.

2). If the ESP32 is disconnected from the computer , when they reconnect, the program won't run again.

Basic Operation:

1). Open Thonny and click"Open...".

Thonny - <untitled> @ 1:1</untitled>		-		$\times$
File Edit View Run Tools Help				
□ 📴 🖬 💿 🔅 👁 3	vunti	led> ×		
This computer =	^ 1			^
<ul> <li></li></ul>	~			
A best my				
ч- ососру				~
	Shell	×		
				^
	Micr	oPython v1.17 on 2021-09-02; ESP32 module with E	5P32	
	Туре	"help()" for more information.		
	>>>			~
		Micro	Python (ESF	P32)

2). Click"This computer"in the new pop-up window.

T	Where to open from?	×
	This computer	
	MicroPython device	

3). In the new dialog boxselect"Project\_01\_HelloWorld.py",click"Open". The code used in this tutorial is saved in the file "...\6.Codes\ESP32\_Python\_code(Windows)". You can move the code to anywhere. for example, we can save the file "ESP32\_Python\_code(Windows)" in the Disk(D), the route is D:\ESP32\_Python\_code(Windows).

📙   🛃 📙 🖵   2. E	ESP32_code_MicroPython			_	□ ×
File Home	Share View				~ ?
← → ~ ↑	> This PC > Software (D:) > ESP32_Pyth	on_code(Windows)	ت		. ESP32_c
💻 This PC 🔷 🕺	Name	Date modified	Туре	Size	^
🗊 3D OŁ	Iesson 00. Boot	4/2/2022 4:39 PM	File folder		
Deskti	📙 lesson 01. Hello World	4/2/2022 4:39 PM	File folder		
🚔 Docur	lesson 02. LED	4/2/2022 5:26 PM	File folder		
	📙 lesson 03. Traffic_Light	4/2/2022 5:26 PM	File folder		
- Down	📙 lesson 04. Laser sensor	4/2/2022 5:26 PM	File folder		
J Music	lesson 05. Breath	4/2/2022 5:27 PM	File folder		
E Pictur	esson 06. RGB	4/2/2022 5:27 PM	File folder		
🚆 Video	🔄 lesson 07. button	4/2/2022 5:27 PM	File folder		~
≝_ Windc ✓ <					>
04 Items					
🕞 🕞 Ra Open					×
← → • ↑ <mark>-</mark> «	Software (D:) > ESP32_Python_code(Windows) >	lesson 1. Hello World	✓ <sup>0</sup> ,0 Se	earch lesson 01.	Hello World
Organize 🔻 New fo	older			·== <b>·</b>	• •
Videos	^ Name	Date modified	Туре	Size	
늘 Windows10 1909	Th lesson_01_HelloWorld	10/15/2021 8:30 AM	Thonny	1 KB	
Software (D:)					
🗙 New folder (\\de		1			
	×			2	
File	e name: lesson_01_HelloWorld		~ Pythe	files	~
			Ot	ben	Cancel

3). Click "Run current script" to execute the program "Hello World!", "Welcome Keyestudio", which will be printed in the "Shell" window.



#### Exit running online

When running online, click "" "Stop /Restart Backend" or press "Ctrl+C" on the Thonny to exit the program.



#### **Test Code**

<pre>print("Hello</pre>	World!")
<pre>print("Welcon</pre>	<pre>Me Keyestudio")</pre>

### 5.2.2 Project 2: Lighting up LED



#### Overview

In this kit, we have a Keyestudio Purple Module, which is very simple to control. If you want to light up the LED, you just need to make a certain voltage across it.

In the project, we will control the high and low level of the signal end S through programming, so as to control the LED on and off.

#### **Working Principle**

The two circuit diagrams are given.

The left one is wrong wiring-up diagram. Why? Theoretically, when the S terminal outputs high levels, the LED will receive the voltage and light up.

Due to limitation of IO ports of ESP32 board, weak current can't make LED brighten.

The right one is correct wiring-up diagram. GND and VCC are powered up. When the S terminal is a high level, the triode Q1 will be connected and LED will light up(note: current passes through LED and R3 to reach GND by VCC not IO ports). Conversely, when the S terminal is a low level, the triode Q1 will be disconnected and LED will go off.



Components



Wiring Diagram



fritzing

Test Code

from machine import Pin
import time

(continues on next page)

(continued from previous page)

```
led = Pin(0, Pin.OUT)# Build an LED object, connect the external LED light to pin 0, and.

→ set pin 0 to output mode

while True:

    led.value(1)# turn on led

    time.sleep(1)# delay 1s

    led.value(0)# turn off led

    time.sleep(1)# delay 1s
```

#### **Code Explanation**

Machine module is indispensable, we use **import machine** or **from machine import...** to program ESP32 with microPython.

time.sleep() function is used to set delayed time, as time.sleep(0.01), which means, the delayed time is 10ms.

**led = Pin(0, Pin.OUT)**created a pin example and we name **led.** 

**0** is indicative of connected pin GP0**Pin.OUT represents output mode** can use **.value**() **to output high levels** (3.3V) **led.value**(1) **or low levels** (0V) **led.value**(0).

#### while True is loop function

It means that sentences under this function will loop unless **True** changes into **False**. For the function **whileled.value(1)**, outputs high levels to the pin 0; then LED lights up. Then the delayed function **time.sleep(1)** will wait for 1s. When **led.value(0)** output low levels to the pin 0, the LED will go offand the function **time.sleep(1)** will wait for 1s, cyclically, and LED will flash.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code

starts executing, we will see that the LED in the circuit will flash alternately. Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

### 5.2.3 Project 3: Traffic Lights Module



#### Overview

In this lesson, we will learn how to control multiple LED lights and simulate the operation of traffic lights.

Traffic lights are signal devices positioned at road intersections, pedestrian crossings, and other locations to control flows of traffic.

In this kit, we will use the traffic light module to simulate the traffic light.

#### Working Principle

In previous lesson, we already know how to control an LED. In this part, we only need to control three separated LEDs. Input high levels to the signal R(3.3V), then the red LED will be on.



Components



Wiring Diagram



fritzing

#### **Test Code**

```
import machine
import time
led_red = machine.Pin(15, machine.Pin.OUT)
led_yellow = machine.Pin(2, machine.Pin.OUT)
led_green = machine.Pin(0, machine.Pin.OUT)
while True:
   led_green.value(1) # green light turn on
   time.sleep(5) # delay 5s
   led_green.value(0) # green light turn off
    for i in range(3): # yellow light blinks 3 times
        led_yellow.value(1)
        time.sleep((0.5))
        led_yellow.value(0)
        time.sleep((0.5))
   led_red.value(1) # red light turn on
   time.sleep(5) # delay 5s
   led_red.value(0) #red light turn off
```

#### **Code Explanation**

Create pins, set pins mode and delayed functions.

We use the **for** loop.

The simplest form is **for i in range**().

In the code, we used range(3), which means the variable i starts from 0, increase 1 for each time, to 2.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing, we will see that the green LED will be on for 5s then off, the yellow LED will flash for 3s then go off

and the red one will be on for 5s then off. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

### 5.2.4 Project 4: Laser Sensor



#### Description

Lasers are widely used to cut, weld, surface treat, and more on specific materials. The energy of the laser is very high. The toy laser pointer may cause glare to the human eye, and it may cause retinal damage for a long time. my country also prohibits the use of laser to illuminate the aircraft.

#### **Working Principle**

The laser head sensor module is mainly composed of a laser head with a light-emitting die, a condenser lens, and a copper adjustable sleeve.

We can see the circuit schematic diagram of this module which is very similar to the LED we have learned. They are all driven by triodes. A high-level digital signal is directly input at the signal end, then the sensor will start to work; if inputting low levels, the sensor won't work.

Note: don't point an laser emitter at eyes of people.



Components



**Connection Diagram**


fritzing

**Test Code** 

### **Code Explanation**

Please refer to project 2 above for the code setting instructions.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O "Run current script", the code starts executing, we will see that the laser tube on the module emits a red laser signal for 2 seconds, and stops emitting

a red laser signal for 2 seconds. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.

# 5.2.5 Project 5: Breathing LED



#### Overview

A"breathing LED" is a phenomenon where an LED's brightness smoothly changes from dark to bright and back to dark, continuing to do so and giving the illusion of an LED"breathing. This phenomenon is similar to a lung breathing in and out. So how to control LED's brightness? We need to take advantage of PWM. Please refer to Project 6.

## Components



#### **Connection Diagram**



fritzing

#### **Test Code**

```
import time
from machine import Pin,PWM
#The way that the ESP32 PWM pins output is different from traditionally controllers.
#It can change frequency and duty cycle by configuring PWM's parameters at the.
→initialization stage.
#Define GPIO 0's output frequency as 10000Hz and its duty cycle as 0, and assign them to.
\hookrightarrow PWM.
pwm =PWM(Pin(0,Pin.OUT),10000,0)
try:
    while True:
#The range of duty cycle is 0-1023, so we use the first for loop to control PWM to.
\rightarrow change the duty cycle value, making PWM output 0% -100%; Use the second for loop to.
→make PWM output 100%-0%.
        for i in range(0,1023):
            pwm.duty(i)
            time.sleep_ms(1)
        for i in range(0,1023):
            pwm.duty(1023-i)
            time.sleep_ms(1)
except:
#Each time PWM is used, the hardware Timer will be turned ON to cooperate it. Therefore,
→after each use of PWM, deinit() needs to be called to turned OFF the timer. Otherwise,
\hookrightarrow the PWM may fail to work next time.
    pwm.deinit()
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}^{*}$ Run current script", the code starts executing, we will see that the LED on the module gradually gets dimmer then brighter, cyclically, like human

breathe. Press "Ctrl+C"or click ""Stop/Restart backend" to exit the program.

# 5.2.6 Project 6: RGB Module



#### Overview

Among these modules is a RGB module. It adopts a F10-full color RGB foggy common cathode LED. We connect the RGB module to the PWM port of MCU and the other pin to GND(for common anode RGB, the rest pin will be connected to VCC). So what is PWM?

PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output. In general, the input voltages of ports are 0V and 5V. What if the 3V is required? Or a switch among 1V, 3V and 3.5V? We cannot change resistors constantly. For this reason, we resort to PWM.

For Arduino digital port voltage outputs, there are only LOW and HIGH levels, which correspond to the voltage outputs of 0V and 5V respectively. You can define LOW as"0" and HIGH as"1', and let the Arduino output five hundred'0'or"1" within 1 second. If output five hundred'1', that is 5V; if all of which is '0', that is 0V; if output 250 01 pattern, that is 2.5V.

This process can be likened to showing a movie. The movie we watch are not completely continuous. Actually, it generates 25 pictures per second, which cannot be told by human eyes. Therefore, we mistake it as a continuous process. PWM works in the same way. To output different voltages, we need to control the ratio of 0 and 1. The more '0'or 1' output per unit time, the more accurate the control.

### **Working Principle**

For our experiment, we will control the RGB module to display different colors through three PWM values.



## Components

		RGB LED	~~	<b>\$\$</b>
ESP32	ESP32 Expansion	Keyestudio Common Cathode RGB	4P Dupont	Micro USB Ca-
Board*1	Board*1	Module *1	Wire*1	ble*1

## **Connection Diagram**



fritzing

### **Test Code**



(continues on next page)

(continued from previous page)

```
pwm0 = PWM(Pin(pins[0]),10000)
pwm1 = PWM(Pin(pins[1]), 10000)
pwm2 = PWM(Pin(pins[2]), 10000)
#define a function to set the color of RGBLED.
def setColor(r, g, b):
   pwm0.duty(1023-r)
   pwm1.duty(1023-g)
   pwm2.duty(1023-b)
try:
   while True:
        red = randint(0, 1023)
        green = randint(0, 1023)
       blue = randint((0, 1023))
        setColor(red, green, blue)
        time.sleep_ms(200)
except:
   pwm0.deinit()
   pwm1.deinit()
   pwm2.deinit()
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing, we will see that the RGB LED on the module starts to display random colors. Press "Ctrl+C" or click

"Stop/Restart backend" to exit the program.

# 5.2.7 Project 7: Button Sensor



## Overview

In this kit, there is a Keyestudio single-channel button module, which mainly uses a tact switch and comes with a yellow button cap.

In previous lessons, we learned how to make the pins of our single-chip microcomputer output a high level or low level. In this experiment, we will read the high level (3.3V) and low level (0V).

We can determine whether the button on the sensor is pressed by reading the high and low level of the S terminal on the sensor.

#### Working Principle

The button module has four pins. The pin 1 is connected to the pin 3 and the pin 2 is linked with the pin 4. When the button is not pressed, they are disconnected. Yet, when the button is pressed, they are connected. If the button is released, the signal end is high level.



Components



**Connection Diagram** 



fritzing

**Test Code** 

```
from machine import Pin
import time
button = Pin(15, Pin.IN, Pin.PULL_UP)
while True:
    if button.value() == 0:
        print("You pressed the button!") #Press to print the corresponding information.
    else:
        print("You loosen the button!")
    time.sleep(0.1) #delay 0.1s
```

## **Code Explanation**

button = Pin(15, Pin.IN, Pin.PULL\_UP), we define the pin of the button as GP15 and set to PULL-UP mode

We can use **button = Pin(15, Pin.IN) to set INPUT mode,** at this time, the pins are in high resistance state.

1). button.value(), read levels of buttons. Function returns High or Low

2). **if...else...** sentence, when the logic judge is TRUE, the code under the if will be activated; otherwise, the code udder the else will be activated.

3). When ESP32 detects the button pressed, the signal end is low level (GP 15 is low level). **button.value**() is **0**. If the ESP32 detects the button unpressed, **button.value**() is 1 and else sentence will be activated.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing, the string will be displayed on the "Shell" window.

When the button is pressed, the "Shell" window will show "You pressed the button!" when the button is released the

"Shell" window will show "Loosen the button"; as shown below. Press "Ctrl+C"or click "" "Stop/Restart backend" to exit the program.

Shell 🛛		
104	Losen the Button.	
You	loosen the button!	_
You	loosen the button!	
You	pressed the button!	
You	pressed the button!	
You	pressed the button!	1.1

# 5.2.8 Project 8: Capacitive Sensor



### Description

In this kit, there is a capacitive touch module which mainly uses a TTP223-BA6 chip. It is a touch detection chip, which provides a touch button, and its function is to replace the traditional button with a variable area button. When we power on, the sensor needs about 0.5 seconds to stabilize.

Do not touch the keys during this time period. At this time, all functions are disabled, and self-calibration is always performed. The calibration period is about 4 seconds. We display the test results in the shell.



## **Working Principle**

When our fingers touch the module, the signal S outputs high levels, the red LED on the module flashes. We can determine if the button is pressed or not by reading high and low levels on the sensor.

## **Required Components**

	ELE-12		Yanth					<b>\$</b>
ESP32 Board*1	ESP32 Board*1	Expansion	Keyestudio Module*1	DIY	Capacitive	3P Wire*	Dupont 1	Micro USB Ca- ble*1

## **Connection Diagram**



fritzing

**Test Code** 

```
from machine import Pin
import time
touch = Pin(15, Pin.IN, Pin.PULL_UP)
while True:
    if touch.value() == 1:
        print("You pressed the button!") #Press to print the corresponding information.
    else:
        print("You loosen the button!")
    time.sleep(0.1) #delay0.1s
```

## **Code Explanation**

When we touch the sensor, the Shell monitor will show "You pressed the button!", if not, "You loosen the button!" will be shown on the monitor.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing, the string will be displayed in the "Shell" window. when the button is pressed, the red LED lights up and val is 1. Then the shell shows "You pressed the button!"; if the button is released, the red LED is off and val is

0; "You loosen the button!" will be displayed, as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

```
Shell X
You loosen the button!
You loosen the button!
You loosen the button!
You pressed the button!
```

# 5.2.9 Project 9: Obstacle Avoidance Sensor



### Overview

In this kit, there is a Keyestudio obstacle avoidance sensor, which mainly uses an infrared emitting and a receiving tube. In the experiment, we will determine whether there is an obstacle by reading the high and low level of the S terminal on the sensor.

## Working Principle

NE555 circuit provides IR signals with frequency to the emitter TX, then the IR signals will fade with the increase of transmission distance. If encountering the obstacle, it will be reflected back.

When the receiver RX meets the weak signals reflected back, the receiving pin will output high levels, which indicates the obstacle is far away. On the contrary, it the reflected signals are stronger, low levels will be output, which represents the obstacle is close. There are two potentiometers on the module, and by adjusting the two potentiometers, we can adjust its effective distance.



## Components

	CSP-32 Krywatedia			<b>\$</b>
ESP32	ESP32 Expansion	Keyestudio DIY Obstacle Avoid-	3P Dupont	Micro USB Ca-
Board*1	Board*1	ance Sensor*1	Wire*1	ble*1

## **Connection Diagram**



fritzing

**Test Code** 

```
from machine import Pin
import time
sensor = Pin(15, Pin.IN)
while True:
    if sensor.value() == 0:
        print("There are obstacles")
    else:
        print("All going well")
    time.sleep(0.1)
```

## **Code Explanation**

## Note:

Connect the wires according to the connection diagram. After powering on, we start to adjust the two potentiometers to sense distance.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing, the string will be displayed in the "Shell" window. When the sensor detects the obstacle, sensor.value() is 0, the shell will show "There are obstacles", if the obstacle is not detected, sensor.value () is 1, "All going well" will be shown, as shown below.

Press "Ctrl+C"or click "Stop/Restart backend" to exit the program.

Shell ≍					
A11	going well				<b>^</b>
A11	going well				
A11	going well				
A11	going well				
A11	going well				
A11	going well				
Ther	e are obstacl	25			
Ther	e are obstacl	25			
Ther	e are obstacl	25			
Ther	e are obstacl	23			
Ther	e are obstacl	23			_



# 5.2.10 Project 10: Line Tracking Sensor

## Description

In this kit, there is a DIY electronic building block single-channel line tracking sensor which mainly uses a TCRT5000 reflective black and white line recognition sensor element.

In the experiment, we judge the color (black and white) of the object detected by the sensor by reading the high and low levels of the S terminal on the module; and display the test results on the shell.

## **Working Principle**



When a black or no object is detected, the signal terminal will output high levels; when white object is detected, the signal terminal is low level; its detection height is 0-3cm.

We can adjust the sensitivity by rotating the potentiometer on the sensor. When the potentiometer is rotated, the sensitivity is best when the red LED on the sensor is at the critical point between off and on.

### **Required Components**



**Connection Diagram** 



fritzing

**Test Code** 

```
from machine import Pin
import time
sensor = Pin(15, Pin.IN, Pin.PULL_UP)
while True:
    if sensor.value() == 0:
        print("0 White") #Press to print the corresponding information.
    else:
        print("1 Black")
    time.sleep(0.1) #delay 0.1s
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing, the string and data will be displayed in the "Shell" window.

When the sensor doesn't detect an object or detects a black object, the val is 1, and the shell will display "Black"; when a white object (can reflect light) is detected, the val is 0, and the shell displays "White", as shown below. Press

"Ctrl+C"or click ""Stop/Restart backend"to exit the program.

Shell ⊠		
-	DIUCA	
1	Black	
0	White	
		÷

# 5.2.11 Project 11: Photo Interrupter



## Description

This kit contains a photo interrupter which mainly uses 1 ITR-9608 photoelectric switch. It is a photoelectric switch optical switch sensor.

## **Working Principle**

When the paper is put in the slot, C is connected with VCC and the signal end S of the sensor are high levels; then the red LED will be off. Otherwise, the red LED will be on.



## **Required Components**



## **Connection Diagram**



fritzing

Test Code

```
from machine import Pin
import time
sensor = Pin(15, Pin.IN, Pin.PULL_UP)
lastState = 0
PushCounter = 0
while True:
    State = sensor.value()
    if State != lastState:
        if State == 1:
            PushCounter += 1
            print(PushCounter) #Press to print the corresponding information.
lastState = State
```

## **Code Explanation**

Logic setting :

Set State to 0 (value of the sensor)

Set lastState to 0

Condition	Value	Result
When an object enters the slot	lastState is 0State turns into 1; lastState turns into 1	Set PushCounter to PushCounter+1print the value of PushCounter
When the object leaves	lastState is 1State becomes 0two data are	PushCounterdoesn't change;Don't print
the slot	not equallastState turns into 0.	the value of PushCounter
When the object goes	lastState is 0, State becomes 1two data are	Set PushCounter to PushCounter+1And
through this slot again	not equallastState turns into 1.	print the value of PushCounter
When the object leaves this slot again	lastState is 1State turns into 0two data are not equal lastState turns into 0	PushCounter doesn't change;Don't print the PushCounter value

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O "Run current script", the code starts executing, data will be displayed in the "Shell" window. Every time when the object passes through the slot of the

sensor, the PushCounter data will increase by 1 continuously, as shown below. Press"Ctrl+C"or click<sup>99</sup>"Stop/Restart backend"to exit the program.

Shell Ж	
1	•
2	
3	
4	
5	
6	
7	
8	
9	
	 ŀ

## 5.2.12 Project 12: Tilt Module



### Overview

In this kit, there is a Keyestudio tilt sensor. The tilt switch can output signals of different levels according to whether the module is tilted. There is a ball inside. When the switch is higher than the horizontal level, the switch is turned on, and when it is lower than the horizontal level, the switch is turned off. This tilt module can be used for tilt detection, alarm or other detection.

### **Working Principle**

The working principle is pretty simple. When pin 1 and 2 of the ball switch P1 are connected, the signal S is low level and the red LED will light up; when they are disconnected, the pin will be pulled up by the 4.7K R1 and make S a high level, then LED will be off.



Components



**Connection Diagram** 



fritzing

Test Code

```
from machine import Pin
import time
TiltSensor = Pin(15, Pin.IN)
while True:
   value = TiltSensor.value()
   print(value, end = " ")
   if value== 0:
        print("The switch is turned on")
   else:
        print("The switch is turned off")
   time.sleep(0.1)
```

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing, the string and the data will be displayed in the "Shell" window. When the tilt module is inclined to one side, the red LED on the module will be off and the Shell" window will display"1. the switch is turned off.

In contrast, if you make it incline the other side, the red LED will light up and the monitor will display"0, the switch is

turned on", as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

Shell	×				
0	The	switch	is	turned	on
0	The	switch	is	turned	on
0	The	switch	is	turned	on
0	The	switch	is	turned	on
1	The	switch	is	turned	off
1	The	switch	is	turned	off
1	The	switch	is	turned	off
1	The	switch	is	turned	off
1	The	switch	is	turned	off
1	The	switch	is	turned	off
1	The	switch	is	turned	off

# 5.2.13 Project 13: Collision Sensor



### Description

The collision sensor uses a tact switch. This sensor is often used as a limit switch in 3D printers. In the experiment, we judge whether the sensor shrapnel is pressed down by reading the high and low levels of the S terminal on the module; and, we display the test results in the shell.

## **Working Principle**

It mainly uses a tact switch. When the shrapnel of the tact switch is pressed, 2 and 3 are connected, the signal terminal S is low level, and the red LED on the module lights up; when the touch switch is not pressed, 2 and 3 are not connected, and 3 is pulled up to a high level by the 4.7K resistor R1, that is, the sensor signal terminal S is a high level, and the built-in red LED will be off at this time.



### **Components Required**



**Connection Diagram** 



fritzing

Test Code

```
from machine import Pin
import time
CollisionSensor = Pin(15, Pin.IN)
while True:
   value = CollisionSensor.value()
   print(value, end = " ")
   if value== 0:
        print("The end of this!")
   else:
        print("All going well")
   time.sleep(0.1)
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing, the string and the data will be displayed in the "Shell" window. When the shrapnel on the sensor is pressed down, val is 0, the red LED of the module is on, and "The end of his!" is printed.

When the shrapnel is released, the val is 1, the red LED of the module is off, and "All going well" is printed. !" character, as shown below. Press "Ctrl+C" click "":Stop/Restart backend" to exit the program.

```
Shell X

1 All going well

1 All going well

1 All going well

1 All going well

0 The end of his!

0 The end of his!
```

## 5.2.14 Project 14: Hall Sensor



#### Description

In this kit, there is a Hall sensor which mainly adopts a A3144 linear Hall element. The element P1 are composed of a voltage regulator, a Hall voltage generator, a differential amplifier, a Schmitt trigger, a temperature compensation circuit and an open-collector output stage. In the experiment, we will use the Hall sensor to detect the magnetic field and display the test results on the shell.

## **Working Principle**

When the sensor detects no magnetic field or a north pole magnetic field, the signal terminal will be high level; when it senses a south pole magnetic field, the signal terminal will be low levels.

The stronger the magnetic field strength is, induction distance is longer.



**Required Components** 



### **Connection Diagram**



fritzing

**Test Code** 

```
from machine import Pin
import time
hall = Pin(15, Pin.IN)
while True:
   value = hall.value()
   print(value, end = " ")
   if value == 0:
        print("A magnetic field")
   else:
        print("There is no magnetic field")
   time.sleep(0.1)
```

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing, the string and the data will be displayed in the "Shell" window. When the sensor detects no magnetic fields or the north pole magnetic field, Shell will show "1 There is no magnetic field" and the LED on the sensor will be off.

When it detects the south pole magnetic field, the Shell will show "0 A magnetic field" and the LED on the sensor will be on, as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

Shell	×	
-		ere io no magneere riera
1	тh	ere is no magnetic field
1	тh	ere is no magnetic field
0	Α	magnetic field
0	Α	magnetic field
0	Α	magnetic field
0	Α	magnetic field
0	Α	magnetic field
0	Α	magnetic field
0	А	magnetic field

# 5.2.15 Project 15: Reed Switch Module



### Overview

In this kit, there is a Keyestudio reed switch module, which mainly uses a MKA10110 green reed component.

The reed switch is the abbreviation of the dry reed switch. It is a passive electronic switch element with contacts.

It has the advantages of simple structure, small size and easy control. Its shell is a sealed glass tube with two iron elastic reed electric plates.

In the experiment, we will determine whether there is a magnetic field near the module by reading the high and low level of the S terminal on the module; and, we display the test result in the shell.

## **Working Principle**



In normal conditions, the glass tube in the two reeds made of special materials are separated. When a magnetic substance close to the glass tube, in the role of the magnetic field lines, the pipe within the two reeds are magnetized to attract each other in contact, the reed will suck together, so that the junction point of the connected circuit communication.

After the disappearance of the outer magnetic reed because of their flexibility and separate, the line is disconnected. The sensor uses this characteristic to build a circuit to convert magnetic field signal into high and low level signal.

## Components



## **Connection Diagram**



fritzing

Test Code

```
from machine import Pin
import time
ReedSensor = Pin(15, Pin.IN)
while True:
   value = ReedSensor.value()
   print(value, end = " ")
   if value == 0:
        print("A magnetic field")
   else:
        print("There is no magnetic field")
   time.sleep(0.1)
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing, the string and the data will be displayed in the "Shell" window.

When the sensor detects a magnetic field, val is 0 and the red LED of the module lights up, "0 A magnetic field" will be displayed. When no magnetic field is detected, val is 1, and the LED on the module goes out, "1 There is no magnetic

field" will be shown, as shown below. Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

Shell	×							
1	The	re	is	no	magne	etic	field	
1	The	re	is	no	magne	etic	field	
1	The	re	is	no	magne	etic	field	
1	The	re	is	no	magne	etic	field	
1	The	re	is	no	magne	etic	field	
0	Απ	agn	eti	c f	field			
0	Απ	agn	eti	c f	field			
0	Αm	agn	eti	c f	field			
0	Αm	agn	eti	c f	field			
0	Αm	agn	eti	c f	field			
0	Αm	agn	eti	c f	field			

# 5.2.16 Project 16: PIR Motion Sensor



#### Overview

In this kit, there is a Keyestudio PIR motion sensor, which mainly uses an RE200B-P sensor elements. It is a human body pyroelectric motion sensor based on pyroelectric effect, which can detect infrared rays emitted by humans or animals, and the Fresnel lens can make the sensor's detection range farther and wider.

In the experiment, we determine if there is someone moving nearby by reading the high and low levels of the S terminal on the module. The detected results will be displayed on the Shell.

## **Working Principle**

The upper left part is voltage conversion(VCC to 3.3V). The working voltage of sensors we use is 3.3V, therefore we can't use 5V directly. The voltage conversion circuit is needed.

When no person is detected or no infrared signal is received, and pin 1 of the sensor outputs low level. At this time, the LED on the module will light up and the MOS tube Q1 will be connected and the signal terminal S will detect Low levels.

When one is detected or an infrared signal is received, and pin 1 of the sensor outputs a high level. Then LED on the module will go off, the MOS tube Q1 is disconnected and the signal terminal S will detect high levels.



## **Required Components**



#### **Connection Diagram**



fritzing

Test Code

from machine import Pin
import time
PIR = Pin(15, Pin.IN)
(continues on next page)

(continued from previous page)

```
while True:
    value = PIR.value()
    print(value, end = " ")
    if value == 1:
        print("Some body is in this area!")
    else:
        print("No one!")
    time.sleep(0.1)
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing, the string and the data will be displayed in the "Shell" window. When the sensor detects someone nearby, value is 1, the LED will go off and the "Shell" window will show "1 Somebody is in this area!".

On the contrary, the value is 0, the LED will go up and "0 No one!" will be shown, as shown below. Press "Ctrl+C" or

click<sup>\*\*\*</sup> "Stop/Restart backend" to exit the program.

Shell X 0 No one! 0 No one! 0 No one! 1 Some body is in this area! 1 Some body is in this area!

## 5.2.17 Project 17: Active Buzzer



#### Overview

In this kit, it contains an active buzzer module and a power amplifier module (the principle is equivalent to a passive buzzer). In this experiment, we control the active buzzer to emit sounds. Since it has its own oscillating circuit, the buzzer will automatically sound if given large voltage.

### **Working Principle**

From the schematic diagram, the pin of buzzer is connected to a resistor R2 and another port is linked with a NPN triode Q1. So, if this triode Q1 is powered, the buzzer will sound.

If the base electrode of the triode connected to the R1 resistor is a high level, the triode Q1 will be connected. If the base electrode is pulled down by the resistor R3, the triode is disconnected.

When we output a high level from the IO port to the triode, the buzzer will emit sounds; if outputting low levels, the buzzer won't emit sounds.



## Components



### **Connection Diagram**



fritzing

**Test Code** 

```
from machine import Pin
import time
buzzer = Pin(15, Pin.OUT)
while True:
    buzzer.value(1)
    time.sleep(1)
    buzzer.value(0)
    time.sleep(1)
```

### **Code Explanation**

In the experiment, we set the pin to GPIO15. When setting to high, the active buzzer will beep. When setting to low, the active buzzer will stop emitting sounds.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing. The active buzzer will emit sound for 1 second, and stop for 1 second. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

# 5.2.18 Project 18: 8002b Audio Power Amplifier



### Overview

In this kit, there is a Keyestudio 8002b audio power amplifier. The main components of this module are an adjustable potentiometer, a speaker, and an audio amplifier chip;

The main function of this module is: it can amplify the output audio signal, with a magnification of 8.5 times, and play sound or music through the built-in low-power speaker, as an external amplifying device for some music playing equipment.

In the experiment, we used the 8002b power amplifier speaker module to emit sounds of various frequencies.

### **Working Principle**

In fact, it is similar to a passive buzzer. The active buzzer has its own oscillation source. Yet, the passive buzzer does not have internal oscillation. When controlling the circuit, we need to input square waves of different frequencies to the positive pole of the component and ground the negative pole to control the buzzer to chime sounds of different frequencies.


#### Components



#### **Connection Diagram**





Test Code

```
from machine import Pin, PWM
from time import sleep
buzzer = PWM(Pin(15))
buzzer.duty(1000)
buzzer.freq(523)#DO
sleep(0.5)
buzzer.freq(586)#RE
sleep(0.5)
buzzer.freq(658)#MI
sleep(0.5)
buzzer.freq(697)#FA
sleep(0.5)
buzzer.freq(783)#SO
sleep(0.5)
buzzer.freq(879)#LA
sleep(0.5)
buzzer.freq(987)#SI
sleep(0.5)
buzzer.duty(◊)
```

#### **Code Explanation**

In this experiment, we use the PWM class of the machine module, buzzer = PWM(Pin(15)) to create an instance of the PWM class, and the buzzer pin is connected to GPIO15.

The buzzer.duty(1000): set the duty cycle, and the duty cycle is 1000/4095. The larger the value, the louder the buzzer. When set to 0, the buzzer does not emit sound. **buzzer.freq**() is the frequency setting method.

In the experiment, we use the PWM on the machine module. **buzzer = PWM(Pin(15))** 

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing. The power amplifier module will emit the sound of the corresponding frequency corresponding to the beat :DO for 0.5s, Re for 0.5s, Mi for 0.5s, Fa for 0.5s, So for 0.5s, La 0.5s and Si for 0.5s.

Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

# 5.2.19 Project 19: 130 Motor



#### Description

The 130 motor driver module is compatible with servo motors, which has high efficiency and good quality fans.

It adopts a HR1124S motor control chip. HR1124S is a single-channel H-bridge driver chip for DC motor solutions. In addition, this chip has low standby current and low quiescent current.

The module is compatible with various single-chip control boards. In the experiment, we can control the rotation direction of the motor by outputting the voltage directions of the two signal terminals IN+ and IN- to make the motor rotate.

#### Working Principle

The chip is used to help drive the motor. We can't drive it with a triode or an IO port due to its a large current of need. It is very simple to make the motor rotate. Just apply voltage to both ends of the motor. The direction of the motor is different in different voltage directions. Within the rated voltage, the higher the voltage, the faster the motor rotates; on the contrary, the lower the voltage, the slower the motor rotates, or even unable to rotate.

So we can use the PWM port to control the speed of the motor. We haven't learned PWM here, so we use the high and low levels to control the motor first.



**Required Components** 



Note: the motor is separated with its fan, you need to assemble it first.

#### **Connection Diagram**

130 Motor	ESP32 Expansion Board
G	G
V	5V
IN+	IO15
IN-	IO4



fritzing

### **Test Code**

from machine import Pin import time
<pre>#Two pins of the motor INA = Pin(15, Pin.OUT) #INA corresponds to IN+</pre>
<pre>INB = Pin(4, Pin.OUT)#INB corresponds to IN-</pre>
while True:
#Counterclockwise 2s
INA.value(1)
INB.value(0)
time.sleep(2)
#stop 1s
INA.value(0)
INB.value(0)
time.sleep(1)
#Turn clockwise for 2s
INA.value(0)
INB.value(1)
time.sleep(2)
#stop 1s
INA.value(0)
INB.value(0)
time.sleep(1)

## **Code Explanation**

Set pins to GPIO4, GPIO15, when the pin GPIO4 outputs low levels and the pin GPIO15 outputs high levels, the motor

will rotate counterclockwise; when both pins are set to low, the motor stops rotating.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch ON the ESP32 expansion board to the ON end, after powering on, click O "Run current script", the code starts executing, then the fan will rotate counterclockwise for 2 s, stop for 1 s; and rotate clockwise for 2 s and stop for 1 s, cycle alternately. Press "Ctrl+C" or click O "Stop/Restart backend" to exit the program.

# 5.2.20 Project 20: Potentiometer



#### Overview

The following we will introduce is the Keyestudio rotary potentiometer which is an analog sensor.

The digital IO ports can read the voltage value between 0 and 3.3V and the module only outputs high levels. However, the analog sensor can read the voltage value through 16 ADC analog ports on the ESP32 board. In the experiment, we will display the test results on the Shell.

#### **Working Principle**



It uses a 10K adjustable resistor. We can change the resistance by rotating the potentiometer. The signal S can detect the voltage changes(0-3.3V) which are analog quantity.

**ADC** The more bits an ADC has, the denser the partitioning of the simulation, the higher the accuracy of the final conversion.



Subsection 1: The analog value within 0V-3.3/4095 V corresponds to the number 0; Subsection 2: The analog value within 3.3/4095V-2\*3.3/4095V corresponds to the number 1; .....

The conversion formula is as follows:

DAC The higher the precision of DAC, the higher the precision of the output voltage value.

The conversion formula is as follows:

Analog Voltage = 
$$\frac{DACValue}{255}$$
 \* 3.3(V)

. \_.\_\_ -

#### ADC on ESP32

The ESP32 has 16 pins that can be used to measure analog signals. GPIO pin serial numbers and analog pin definitions are shown below:

ADC number in ESP32	ESP32 GPIO number
ADC0	GPIO 36
ADC3	GPIO 39
ADC4	GPIO 32
ADC5	GPIO33
ADC6	GPIO34
ADC7	GPIO 35
ADC10	GPIO 4
ADC11	GPIO0
ADC12	GPIO2
ADC13	GPIO15
ADC14	GPIO13
ADC15	GPIO 12
ADC16	GPIO 14
ADC17	GPIO27
ADC18	GPIO25
ADC19	GPIO26

#### DAC on ESP32

The ESP32 has two 8-bit digital-to-analog converters connected to GPIO25 and GPIO26 pins, which are immutable, as shown below :

Simulate pin number	GPIO number
DAC1	GPIO25
DAC2	GPIO26

### Components



#### **Connection Diagram**



fritzing

#### **Test Code**

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
# and print these data to "Shell".
try:
    while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep((0.1)
except:
   pass
```

#### **Code Explanation**

1). In the experiment, add "From Machine import ADC" to the top of your Python file every time you use the ACD module, the same goes for DAC modules.

2). **machine.ADC(pin):** Create an ADC object associated with the given pin. 3). **pin:** The available pins are Pin(36)Pin(39)Pin(34Pin(35)Pin(32)Pin(33). DAC(pin). Create an DAC object associated with the given pin.

4). machine.ADC(pin): The available pins are pin (25) pin (26).

5). ADC. Read():Read ADC value and return ADC value.

6).ADC.atten(db): Set attenuation ration (that is, the full range voltage, such as the voltage of 11db full range is 3.3V)

**db**attenuation ratio

ADC.ATTIN\_0DB —full range of 1.2V

ADC.ATTN\_2\_5\_DB —full range of 1.5V

ADC.ATTN\_6DB —full range of 2.0 V

ADC.ATTN\_11DB — full range of 3.3V

ADC.width(bit): Set data width.

**bit**data bit

ADC.WIDTH\_9BIT —9 data width

ADC.WIDTH\_10BIT — 10 data width

ADC.WIDTH\_11BIT — 11 data width

ADC.WIDTH\_12BIT — 12 data width

7). The read()method reads the ADC value rang is 0~4095the adc.read() reads the ADC value input by the ADC(Pin(34)) Pin and assigns it to a variable named adcVal.

8). DAC.write(value):Output the voltage value, the data rang : 0-255the corresponding output voltage is 0-3.3V.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and poweron. Click O"Run current script", the code starts executing. The "Shell" window prints and displays the potentiometer ADC value, DAC value and voltage value. Rotating the potentiometer handle, the ADC value, DAC value and voltage value will change. Press "Ctrl+C" or

click ""Stop/Restart backend" to exit the program.

Shell  $\times$ 

```
_____
ADC Val: 0 DACVal: 0 Voltage: 0.0 V
ADC Val: 23 DACVal: 1 Voltage: 0.0185348 V
ADC Val: 48 DACVal: 3 Voltage: 0.03868132 V
ADC Val: 268 DACVal: 16 Voltage: 0.2159707 V
ADC Val: 559 DACVal: 34 Voltage: 0.4504762 V
ADC Val: 0 DACVal: 0 Voltage: 0.0 V
ADC Val: 553 DACVal: 34 Voltage: 0.445641 V
ADC Val: 810 DACVal: 50 Voltage: 0.6527472 V
ADC Val: 1294 DACVal: 80 Voltage: 1.042784 V
ADC Val: 1280 DACVal: 80 Voltage: 1.031502 V
ADC Val: 1287 DACVal: 80 Voltage: 1.037143 V
ADC Val: 1514 DACVal: 94 Voltage: 1.220073 V
ADC Val: 2160 DACVal: 135 Voltage: 1.740659 V
ADC Val: 2162 DACVal: 135 Voltage: 1.742271 V
ADC Val: 2171 DACVal: 135 Voltage: 1.749524 V
ADC Val: 2467 DACVal: 154 Voltage: 1.988059 V
ADC Val: 2642 DACVal: 165 Voltage: 2.129084 V
ADC Val: 2640 DACVal: 165 Voltage: 2.127473 V
ADC Val: 2723 DACVal: 170 Voltage: 2.194359 V
ADC Val: 2911 DACVal: 181 Voltage: 2.345861 V
ADC Val: 3008 DACVal: 188 Voltage: 2.424029 V
ADC Val: 3029 DACVal: 189 Voltage: 2.440952 V
ADC Val: 3140 DACVal: 196 Voltage: 2.530403 V
ADC Val: 3271 DACVal: 204 Voltage: 2.635971 V
ADC Val: 3583 DACVal: 223 Voltage: 2.887399 V
ADC Val: 3664 DACVal: 229 Voltage: 2.952674 V
```

## 5.2.21 Project 21: Steam Sensor



#### Description

This is a DIY electronic building block water drop sensor. It is an analog (digital) input module, also called rain, rain sensor. It can be used to monitor various weather conditions, detect whether it is raining and the amount of rain, convert it into digital signal (DO) and analog signal (AO) output, and is widely used in Arduino robot kits, raindrops, rain sensors, and can be used for various circumstances. It can monitor various weather conditions, and convert it into digital signal and AO output, and can also be used for automobile automatic wiper system, intelligent lighting system and intelligent sunroof system.

In the experiment, we input the sensor signal terminal (S terminal) to the analog port of the ESP32 development board, sense the change of the analog value, and display the corresponding analog value on the shell.

#### Working Principle

Its principle is to detect the amount of water through the exposed printed parallel lines on the circuit board. The more water there is, the more wires will be connected, and the conductive contact area increases. The voltage output by pin 2 will gradually increase. The larger the analog value detected by the signal terminal S is.

It can also detect steam in the air. Two position holes are used to install on the other devices.



## **Required Components**



#### **Connection Diagram**



fritzing

## Test Code

# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC

(continues on next page)

(continued from previous page)

```
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
# and print these data to "Shell".
try:
    while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep(0.1)
except:
   pass
```

#### Test Result

Wiring up and powering on, then click O"Run current script", the code starts executing. The Shell will display ADC value, DAC value and voltage value of the sensor. When a few drops of water are placed in the sensor sensing area, the ADC value, DAC value and voltage value will change. The more water volume, the greater the output voltage value ,

ADC value and the DAC value. Press"Ctrl+C" or click "Stop/Restart backend" to exit the program.

```
Shell ×
                       . . . . . . . . .
 ADC Val: 0 DACVal: 0 Voltage: 0.0 V
 ADC Val: 23 DACVal: 1 Voltage: 0.0185348 V
 ADC Val: 48 DACVal: 3 Voltage: 0.03868132 V
 ADC Val: 268 DACVal: 16 Voltage: 0.2159707 V
 ADC Val: 559 DACVal: 34 Voltage: 0.4504762 V
 ADC Val: 0 DACVal: 0 Voltage: 0.0 V
 ADC Val: 553 DACVal: 34 Voltage: 0.445641 V
 ADC Val: 810 DACVal: 50 Voltage: 0.6527472 V
 ADC Val: 1294 DACVal: 80 Voltage: 1.042784 V
 ADC Val: 1280 DACVal: 80 Voltage: 1.031502 V
 ADC Val: 1287 DACVal: 80 Voltage: 1.037143 V
 ADC Val: 1514 DACVal: 94 Voltage: 1.220073 V
 ADC Val: 2160 DACVal: 135 Voltage: 1.740659 V
 ADC Val: 2162 DACVal: 135 Voltage: 1.742271 V
 ADC Val: 2171 DACVal: 135 Voltage: 1.749524 V
 ADC Val: 2467 DACVal: 154 Voltage: 1.988059 V
 ADC Val: 2642 DACVal: 165 Voltage: 2.129084 V
 ADC Val: 2640 DACVal: 165 Voltage: 2.127473 V
 ADC Val: 2723 DACVal: 170 Voltage: 2.194359 V
 ADC Val: 2911 DACVal: 181 Voltage: 2.345861 V
 ADC Val: 3008 DACVal: 188 Voltage: 2.424029 V
 ADC Val: 3029 DACVal: 189 Voltage: 2.440952 V
 ADC Val: 3140 DACVal: 196 Voltage: 2.530403 V
 ADC Val: 3271 DACVal: 204 Voltage: 2.635971 V
 ADC Val: 3583 DACVal: 223 Voltage: 2.887399 V
 ADC Val: 3664 DACVal: 229 Voltage: 2.952674 V
```

## 5.2.22 Project 22: Sound Sensor



#### Overview

In this kit, there is a Keyestudio DIY electronic block and a sound sensor. In the experiment, we test the analog value corresponding to the sound level in the current environment with it. The louder the sound, the larger the ADC, DAC and the voltage value, and the "shell" window will display the test results.

#### **Working Principle**

It uses a high-sensitive microphone component and an LM386 chip. We build the circuit with the LM386 chip and amplify the sound through the high-sensitive microphone. In addition, we can adjust the sound volume by the potentiometer. Rotate it clockwise, the sound will get louder.



#### Components

	CORPORATION		Vicrophase		~		
ESP32 Board*1	ESP32 Board*1	Expansion	Keyestudio DIY Sound Sen- sor*1	3P Wire*1	Dupont	Micro USB ( ble*1	Ca-

## **Connection Diagram**



fritzing

#### Test Code

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
# and print these data to "Shell".
try:
   while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep((0.1)
except:
   pass
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. The "Shell" window will display the sound sensor ADC value, DAC value and voltage value.

Rotate the potentiometer clockwise and speak at the MIC. Then you can see the analog value get larger, as shown below.

Press "Ctrl+C"or click <sup>₩</sup> "Stop/Restart backend" to exit the program.

Shell $ imes$	
>>> %Run -c \$EDITOR_CONTENT	^
ADC Val: 0 DACVal: 0 Voltage: 0.0 V ADC Val: 23 DACVal: 1 Voltage: 0.0185348 V ADC Val: 1520 DACVal: 95 Voltage: 1.224908 V ADC Val: 551 DACVal: 34 Voltage: 0.4440293 V ADC Val: 2285 DACVal: 142 Voltage: 1.841392 V ADC Val: 1395 DACVal: 87 Voltage: 1.124176 V ADC Val: 0 DACVal: 0 Voltage: 0.0 V ADC Val: 1902 DACVal: 118 Voltage: 1.532747 V ADC Val: 0 DACVal: 0 Voltage: 0.0 V ADC Val: 0 DACVal: 0 Voltage: 0.0 V ADC Val: 0 DACVal: 0 Voltage: 0.0 V	
ADC Val. 000 DACVAL. 05 VOIDage. 0.5101055 V	$\sim$

# 5.2.23 Project 23: Photoresistor



## Description

In this kit, there is a photoresistor which consists of photosensitive resistance elements. Its resistance changes with the light intensity. Also, it converts the resistance change into a voltage change through the characteristic of the photosensitive resistive element. When wiring it up, we interface its signal terminal (S terminal) with the analog port of ESP32, so as to sense the change of the analog value, and display the corresponding analog value in the shell.

## Working Principle

If there is no light, the resistance is 0.2M and the detected voltage at the terminal 2 is close to 0. When the light intensity increases, the resistance of photoresistor and detected voltage will diminish, and the detected voltage is increasing.



Components



#### **Connection Diagram**



fritzing

## Test Code



(continues on next page)

(continued from previous page)

```
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
# and print these data to "Shell".
try:
    while True:
        adcVal=adc.read()
        dacVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep(0.1)
except:
        pass
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram and poweron. Click W "Run current script", the code starts executing. The "Shell" window will display the photoresistor ADC value, DAC value and voltage value. The

brighter the light, the greater the analog value, as shown below. Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

```
Shell ×
 ADC Val: 1472 DACVal: 92 Voltage: 1.186227 V
 ADC Val: 1645 DACVal: 102 Voltage: 1.325641 V
 ADC Val: 1847 DACVal: 115 Voltage: 1.488425 V
 ADC Val: 2043 DACVal: 127 Voltage: 1.646374 V
 ADC Val: 2254 DACVal: 140 Voltage: 1.81641 V
 ADC Val: 2442 DACVal: 152 Voltage: 1.967912 V
 ADC Val: 2625 DACVal: 164 Voltage: 2.115385 V
 ADC Val: 2752 DACVal: 172 Voltage: 2.217729 V
 ADC Val: 2832 DACVal: 177 Voltage: 2.282198 V
 ADC Val: 2880 DACVal: 180 Voltage: 2.320879 V
 ADC Val: 2887 DACVal: 180 Voltage: 2.32652 V
 ADC Val: 2873 DACVal: 179 Voltage: 2.315238 V
 ADC Val: 2922 DACVal: 182 Voltage: 2.354725 V
 ADC Val: 2991 DACVal: 186 Voltage: 2.41033 V
 ADC Val: 3051 DACVal: 190 Voltage: 2.458681 V
 ADC Val: 3103 DACVal: 193 Voltage: 2.500586 V
 ADC Val: 3145 DACVal: 196 Voltage: 2.534432 V
 ADC Val: 3163 DACVal: 197 Voltage: 2.548938 V
 ADC Val: 3181 DACVal: 198 Voltage: 2.563443 V
 ADC Val: 3187 DACVal: 199 Voltage: 2.568278 V
 ADC Val: 3214 DACVal: 200 Voltage: 2.590037 V
 ADC Val: 3314 DACVal: 207 Voltage: 2.670623 V
```

# 5.2.24 Project 24: NTC-MF52AT Thermistor



#### Overview

In the experiment, there is a NTC-MF52AT analog thermistor. We connect its signal terminal to the analog port of the ESP32 mainboard and read the corresponding ADC value, voltage value and thermistor value.

We can use analog values to calculate the temperature of the current environment through specific formulas. Since the temperature calculation formula is more complicated, we only read the corresponding analog value.

#### **Working Principle**



This module mainly uses NTC-MF52AT thermistor element, which can can sense the changes of the surrounding environment temperature. Resistance changes with the temperature, causing the voltage of the signal terminal S to change.

This sensor uses the characteristics of NTC-MF52AT thermistor element to convert resistance changes into voltage changes.

#### Components



#### **Connection Diagram**



fritzing

Test Code

```
from machine import Pin, ADC
import time
import math
#Set ADC
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
try:
   while True:
        adcValue = adc.read()
        voltage = adcValue / 4095 * 3.3
       Rt = (3.3 - voltage) / voltage * 4.7;
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = (tempK - 273.15)
       print("ADC value:",adcValue," Voltage:",voltage,"V"," Temperature: ",tempC,"C
time.sleep(1)
except:
   pass
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. The "Shell" window will display the thermistor ADC value, voltage value and temperature value, as

shown below. Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

```
Shell ×>>> %Run -c $EDITOR_CONTENTADC value: 2305Voltage: 1.857509 V<br/>1.843004 V<br/>ADC value: 2287Temperature: 19.41592 C<br/>Temperature: 19.80316 C<br/>Temperature: 20.47055 C<br/>ADC value: 2266ADC value: 2256Voltage: 1.818022 V<br/>Temperature: 20.47055 C<br/>ADC value: 2246Temperature: 20.47055 C<br/>Temperature: 20.14752 C<br/>ADC value: 2269ADC value: 2269Voltage: 1.83011 V<br/>Temperature: 20.14752 C<br/>ADC value: 2197Voltage: 1.828498 V<br/>Temperature: 20.19058 C<br/>Temperature: 21.74371 C<br/>ADC value: 218ADC value: 2218Voltage: 1.770476 V<br/>Temperature: 21.29001 C<br/>ADC value: 2251Voltage: 1.81393 V<br/>Temperature: 21.0958 C<br/>Temperature: 21.0958 C<br/>ADC value: 2227ADC value: 2227Voltage: 1.794652 V<br/>Temperature: 21.0958 C<br/>ADC value: 2227ADC value: 2227Voltage: 1.794652 V<br/>Temperature: 21.0958 C<br/>ADC value: 2227ADC value: 2227Voltage: 1.810769 V<br/>Temperature: 21.0958 C<br/>ADC value: 2247ADC value: 2257Voltage: 1.810769 V<br/>Temperature: 20.66449 C<br/>ADC value: 2257ADC value: 2257Voltage: 1.818828 V<br/>Temperature: 20.44904 C
```

# 5.2.25 Project 25: Thin-film Pressure Sensor



#### Overview

In this kit, there is a Keyestudio thin-film pressure sensor. The thin-film pressure sensor composed of a new type of nano pressure-sensitive material and a comfortable ultra-thin film substrate, has waterproof and pressure-sensitive functions.

In the experiment, we determine the pressure by collecting the analog signal on the S end of the module. The smaller the ADC value, DAC value and voltage value, the greater the pressure; and the displayed results will shown on the Shell.

#### **Working Principle**

When the sensor is pressed by external forces, the resistance value of sensor will vary. We convert the pressure signals detected by the sensor into the electric signals through a circuit. Then we can obtain the pressure changes by detecting voltage signal changes.



Components



**Connection Diagram** 



fritzing

#### **Test Code**

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
# and print these data to "Shell".
try:
    while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep((0.1)
except:
   pass
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O "Run current script", the code starts executing. The "Shell" window will display the thin-film pressure sensor ADC value, voltage value and DAC value. When the thin-film is pressed by fingers, the analog value will decrease, as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

\_\_\_\_\_

Shell ×		
ADC Val:	: 2278 DACVal: 142 Voltage: 1.835751 V	~
ADC Val:	: 2278 DACVal: 142 Voltage: 1.835751 V	
ADC Val:	: 2275 DACVal: 142 Voltage: 1.833333 V	
ADC Val:	: 2277 DACVal: 142 Voltage: 1.834945 V	
ADC Val:	: 2278 DACVal: 142 Voltage: 1.835751 V	
ADC Val:	: 1755 DACVal: 109 Voltage: 1.414286 V	
ADC Val:	: 1153 DACVal: 72 Voltage: 0.9291575 V	
ADC Val:	: 808 DACVal: 50 Voltage: 0.6511355 V	
ADC Val:	: 851 DACVal: 53 Voltage: 0.6857876 V	
ADC Val:	: 593 DACVal: 37 Voltage: 0.4778755 V	
ADC Val:	: 339 DACVal: 21 Voltage: 0.2731868 V	
ADC Val:	: 400 DACVal: 25 Voltage: 0.3223443 V	
ADC Val:	: 349 DACVal: 21 Voltage: 0.2812454 V	
ADC Val:	: 483 DACVal: 30 Voltage: 0.3892307 V	
ADC Val:	: 318 DACVal: 19 Voltage: 0.2562637 V	
ADC Val:	: 325 DACVal: 20 Voltage: 0.2619048 V	
ADC Val:	: 368 DACVal: 23 Voltage: 0.2965568 V	
ADC Val:	: 424 DACVal: 26 Voltage: 0.341685 V	
		~

# 5.2.26 Project 26: Flame Sensor



## Description

In daily life, it is often seen that a fire broke out without any precaution. It will cause great economic and human loss. So how can we avoid this situation? Right, install a flame sensor and a speaker in those places that easily break out a fire. When the flame sensor detects a fire, the speaker will alarm people quickly to put out the fire.

So in this project, you will learn how to use a flame sensor and an active buzzer module to simulate the fire alarm

system.

#### **Working Principle**

This flame sensor can be used to detect fire or other light sources with wavelength stands at 700nm  $\sim$  1000nm. Its detection angle is about 60°. You can rotate the potentiometer on the sensor to control its sensitivity. Adjust the potentiometer to make the LED at the critical point between on and off state. The sensitivity is the best.

From the below figure, power up. When detecting fire, the digital pin outputs low levels, the red LED2 will light up first, the digital signal terminal D0 outputs a low level, and the red LED1 will light up. The stronger the external infrared light, the smaller the value; the weaker the infrared light, the larger the value.



#### **Required Components**



#### **Connection Diagram**



#### Test Code

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
flame_D = Pin(13, Pin.IN)
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read digital value and ADC value once every 0.1seconds, convert ADC value to DAC value.
→and Voltage value and output it,
# and print these data to "Shell".
try:
   while True:
        digitalVal = flame_D.value()
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("digitalVal:",digitalVal,"ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",
→voltage,"V")
        time.sleep(0.1)
except:
   pass
```

#### **Code Explanation**

Two pins we use are defined as GPIO13 and GPIO34 according to the wiring-up diagram, and print digital signals and

analog signals respectively.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}^{*}$ Run current script", the code starts executing. After powering on, rotating the potentiometer on the sensor, we can adjust the red LED bright and not bright critical point. The red LED2 on the sensor module is lit, while the red LED1 is not.

The "Shell" window will print and display the digital value, ADC value, DAC value and voltage value of the flame sensor. When fire is detected, the LED1 will be on. the digital value will change from 1 to 0, and the analog value will

become smaller, as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

Shell ×	
digitalVal: 1 ADC Val: 2832 DACVal: 177 Voltage: 2.282198 V	~
digitalVal: 1 ADC Val: 2957 DACVal: 184 Voltage: 2.382931 V	
digitalVal: 1 ADC Val: 2890 DACVal: 180 Voltage: 2.328938 V	
digitalVal: 1 ADC Val: 2806 DACVal: 175 Voltage: 2.261245 V	
digitalVal: 1 ADC Val: 2736 DACVal: 171 Voltage: 2.204835 V	
digitalVal: 1 ADC Val: 2235 DACVal: 139 Voltage: 1.801099 V	
digitalVal: 0 ADC Val: 1968 DACVal: 123 Voltage: 1.585934 V	
digitalVal: 0 ADC Val: 1837 DACVal: 114 Voltage: 1.480366 V	
digitalVal: 0 ADC Val: 1872 DACVal: 117 Voltage: 1.508571 V	
digitalVal: 0 ADC Val: 1461 DACVal: 91 Voltage: 1.177363 V	
digitalVal: 0 ADC Val: 1134 DACVal: 70 Voltage: 0.9138462 V	
digitalVal: 0 ADC Val: 957 DACVal: 59 Voltage: 0.7712088 V	
digitalVal: 0 ADC Val: 1009 DACVal: 63 Voltage: 0.8131136 V	
digitalVal: 0 ADC Val: 1053 DACVal: 65 Voltage: 0.8485715 V	
digitalVal: 0 ADC Val: 939 DACVal: 58 Voltage: 0.7567033 V	
digitalVal: 1 ADC Val: 3414 DACVal: 213 Voltage: 2.751209 V	
	<b>b.4</b>

## 5.2.27 Project 27: MQ-2 Gas Sensor

#### Description

This analog gas sensor - MQ2 is used in gas leakage detecting equipment in consumer electronics and industrial markets.

This sensor is suitable for detecting LPG, I-butane, propane, methane, alcohol, Hydrogen and smoke. It has high sensitivity and quick response.

In addition, the sensitivity can be adjusted by rotating the potentiometer.

In the experiment, we read the analog value at the A0 port and the D0 port to determine the content of gas.

#### **Working Principle**

The greater the concentration of smoke, the greater the conductivity, the lower the output resistance, the greater the output analog signal.

When in use, the A0 terminal reads the analog value of the corresponding gas; the D0 terminal is connected to an LM393 chip (voltage comparator), we can adjust the alarm threshold of the measured gas through the potentiometer, and output the digital value at D0. When the measured gas content exceeds the critical point, the D0 terminal outputs a low level. When the measured gas content does not exceed the critical point, the D0 terminal outputs a high level.



**Required Components** 



**Connection Diagram** 



# fritzing

Test Code

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
# Turn on and configure the ADC with the range of 0-3.3V
mq2_D = Pin(13, Pin.IN)
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read digital value and ADC value once every 0.1seconds, convert ADC value to DAC value.
\rightarrow and Voltage value and output it,
# and print these data to "Shell".
while True:
    digitalVal = mq2_D.value()
    adcVal=adc.read()
    dacVal=adcVal//16
    voltage = adcVal / 4095.0 * 3.3
    print("digitalVal:",digitalVal,"ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,
\rightarrow "V", end = " ")
    if digitalVal == 0:
        print("Exceeding")
    else:
        print("Normal")
    time.sleep((0.1)
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code

starts executing. The "shell" window will display the corresponding data and string. After powering on, by rotating the potentiometer on the sensor, we can adjust the red LED bright and not bright critical point.

When the sensor detects the smoke or combustible gas, the red LED lights up and the digital value in the "Shell" window changes from 1 to 0, the ADC value, DAC value and voltage value increase, as shown below. Press "Ctrl+C" or

click <sup>•••</sup> "Stop/Restart backend" to exit the program.

Shell ×	
digitalVal: 1 ADC Va	al: 47 DACVal: 2 Voltage: 0.03787546 V Normal 🔨 🔺
digitalVal: 1 ADC Va	al: 34 DACVal: 2 Voltage: 0.02739927 V Normal
digitalVal: 1 ADC Va	al: 35 DACVal: 2 Voltage: 0.02820513 V Normal
digitalVal: 1 ADC Va	al: 39 DACVal: 2 Voltage: 0.03142857 V Normal
digitalVal: 1 ADC Va	al: 37 DACVal: 2 Voltage: 0.02981685 V Normal
digitalVal: 1 ADC Va	al: 35 DACVal: 2 Voltage: 0.02820513 V Normal
digitalVal: 1 ADC Va	al: 36 DACVal: 2 Voltage: 0.02901099 V Normal
digitalVal: 1 ADC Va	al: 39 DACVal: 2 Voltage: 0.03142857 V Normal
digitalVal: 1 ADC Va	al: 62 DACVal: 3 Voltage: 0.04996337 V Normal
digitalVal: 1 ADC Va	al: 50 DACVal: 3 Voltage: 0.04029304 V Normal
digitalVal: 0 ADC Va	al: 144 DACVal: 9 Voltage: 0.116044 V Exceeding
digitalVal: 0 ADC Va	al: 400 DACVal: 25 Voltage: 0.3223443 V Exceeding
digitalVal: 0 ADC Va	al: 787 DACVal: 49 Voltage: 0.6342124 V Exceeding
digitalVal: 0 ADC Va	al: 1327 DACVal: 82 Voltage: 1.069377 V Exceeding
digitalVal: 0 ADC Va	al: 2007 DACVal: 125 Voltage: 1.617363 V Exceeding
digitalVal: 0 ADC Va	al: 2576 DACVal: 161 Voltage: 2.075897 V Exceeding
	~

# 5.2.28 Project 28: MQ-3 Alcohol Sensor



#### Description

In this kit, there is a MQ-3 alcohol sensor, which uses the gas-sensing material is tin dioxide (SnO2) which has a low conductivity in clean air. When there is alcohol vapor in the environment where the sensor is located, the conductivity of the sensor increases with the increase of the alcohol gas concentration in the air. The change in conductivity can be converted into an output signal corresponding to the gas concentration using a simple circuit.

In the experiment, we read the analog value at the A0 end of the sensor and the digital value at the D0 end to judge the content of alcohol vapor in the air and whether they exceed the standard.

#### **Working Principle**

At a certain temperature, the conductivity changes with the composition of the ambient gas. When in use, A0 terminal reads the analog value corresponding to alcohol vapor; D0 terminal is connected to an LM393 chip (comparator), we can adjust and measure the alcohol vapor alarm threshold through the potentiometer, and output the digital value at D0.

When the measured alcohol vapor content exceeds the critical point, the D0 terminal outputs a low level; when the measured alcohol vapor content does not exceed the critical point, the D0 terminal outputs a high level.



**Components Required** 



**Connection Diagram** 



#### **Test Code**

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
# Turn on and configure the ADC with the range of 0-3.3V
mq3_D = Pin(13, Pin.IN)
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
# Read digital value and ADC value once every 0.1seconds, convert ADC value to DAC value.
\rightarrow and Voltage value and output it,
# and print these data to "Shell".
while True:
    digitalVal = mq3_D.value()
    adcVal=adc.read()
    dacVal=adcVal//16
    voltage = adcVal / 4095.0 * 3.3
    print("digitalVal:",digitalVal,"ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,
\rightarrow "V", end = " ")
    if digitalVal == 0:
        print("Exceeding")
    else:
        print("Normal")
```

(continues on next page)

(continued from previous page)

time.sleep(0.1)

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. The "shell" window will display the corresponding data and string. After powering on, by rotating the potentiometer on the sensor, we can adjust the yellow and green LED bright and not bright critical point.

When the sensor detects the alcohol gas, the yellow and green LED lights up and the digital value in the "Shell" window changes from 1 to 0, the ADC value, DAC value and voltage value increase, as shown below. Press "Ctrl+C" or

click ""Stop/Restart backend" to exit the program.

Shell ×	
digitalVal: 1 ADC Val: 47 DACVal: 2 Voltage: 0.03787546 V Normal	^
digitalVal: 1 ADC Val: 34 DACVal: 2 Voltage: 0.02739927 V Normal	
digitalVal: 1 ADC Val: 35 DACVal: 2 Voltage: 0.02820513 V Normal	
digitalVal: 1 ADC Val: 39 DACVal: 2 Voltage: 0.03142857 V Normal	
digitalVal: 1 ADC Val: 37 DACVal: 2 Voltage: 0.02981685 V Normal	
digitalVal: 1 ADC Val: 35 DACVal: 2 Voltage: 0.02820513 V Normal	
digitalVal: 1 ADC Val: 36 DACVal: 2 Voltage: 0.02901099 V Normal	
digitalVal: 1 ADC Val: 39 DACVal: 2 Voltage: 0.03142857 V Normal	
digitalVal: 1 ADC Val: 62 DACVal: 3 Voltage: 0.04996337 V Normal	
digitalVal: 1 ADC Val: 50 DACVal: 3 Voltage: 0.04029304 V Normal	
digitalVal: 0 ADC Val: 144 DACVal: 9 Voltage: 0.116044 V Exceeding	
digitalVal: 0 ADC Val: 400 DACVal: 25 Voltage: 0.3223443 V Exceeding	
digitalVal: 0 ADC Val: 787 DACVal: 49 Voltage: 0.6342124 V Exceeding	
digitalVal: 0 ADC Val: 1327 DACVal: 82 Voltage: 1.069377 V Exceeding	
digitalVal: 0 ADC Val: 2007 DACVal: 125 Voltage: 1.617363 V Exceeding	
digitalVal: 0 ADC Val: 2576 DACVal: 161 Voltage: 2.075897 V Exceeding	
	Υ.


# 5.2.29 Project 29: Five-key AD Button Module

### Description

When we talked about analog and digital sensors earlier, we talked about the single-channel key module. When we press the key, it outputs a low level, and when we release the key, it outputs a high level. We can only read these two digital signals. In fact, the key module ADC acquisition can also be performed. In this kit, a DIY electronic building block five-way AD button module is included.

We can judge which key is pressed through the analog value. In the experiment, we print out the key press information in the shell.

### **Working Principle**

Let's look at the schematic diagram, when we do not press the key, the OUT of S output to the signal end is pulled down by R1. At this time, we read the low level 0V. When we press the key SW1, the OUT of the output to the signal end S is directly connected to the VCC. At this time, we read the high level 3.3V(the figure is marked as a 12-bit ADC(0~4095) and VCC is 5V. The principle is the same. Here we have VCC of 3.3V and ADC mapped to 12 bits), which is an analog value of 4095.

Next, when we press the key SW2, the OUT terminal voltage of the signal we read is the voltage between R2 and R1, namely VCC\*R1/(R2+R1), which is about 2.64V, and the analog value is about 3276.

When we press the key SW3, the OUT terminal voltage of the signal we read is the voltage between R2+R3 and R1, namely VCC\*R1/(R3+R2+R1), which is about 1.99V, and the analog value is about 2469.

When we press the key SW4, the OUT terminal voltage of the signal we read is the voltage between R2+R3+R4 and R1, namely VCC\*R1/(R4+R3+R2+R1), about 1.31V, and the analog value is about 1626.

Similarly, when we press the key SW5, the OUT terminal voltage of the signal we read is the voltage between R2+R3+R4+R5 and R1, namely VCC\*R1/(R5+R4+R3+R2+R1), which is about 0.68V, and the analog value is about 844.



# **Components Required**



## **Connection Diagram**



fritzing

**Test Code** 

```
# Import Pin and ADC modules.
from machine import ADC,Pin
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
while True:
    adcvalue = adc.read()
    print(adcvalue, end = '')
    if adcvalue <= 500:</pre>
        print(" no key is pressed")
    elif adcvalue <= 1000:</pre>
        print(" SW5 is pressed")
    elif adcvalue <= 2000:</pre>
        print(" SW4 is pressed")
    elif adcvalue <= 3000:</pre>
        print(" SW3 is pressed")
    elif adcvalue <= 4000:</pre>
        print(" SW2 is pressed")
    else:
        print(" SW1 is pressed")
    time.sleep((0.5))
```

### **Code Explanation**

We assign the read analog value to the variable val, and the shell displays the value of val, (our default setting is 9600, which can be changed). We judge the read analog value. When the analog value is lower than 6000, we judge that the button is not pressed. When the analog value is between 6000 and 20000, we judge that the button SW5 is pressed. Between 20000 and 32000, we judge that the button SW4 is pressed.

when the analog value is between 32000 and 45000, we judge that the button SW3 is pressed. When the analog value is between 45000 and 59000, we judge that the button SW2 is pressed. Press. Otherwise, when the analog value is above 59000, we judge that the button SW1 is pressed. If we only use a fixed value, there will inevitably be errors, so we use the interval to judge.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing. when the button is pressed, the shell prints out the corresponding information, as shown in the figure

below. Press "Ctrl+C"or click "" "Stop/Restart backend" to exit the program.

$\mathbf{Shell} \ \times$		
1492	SW4 is pressed	
0 no	) key is pressed	
0 no	) key is pressed	
0 no	) key is pressed	
0 no	) key is pressed	
2321	SW3 is pressed	
0 no	) key is pressed	
0 no	) key is pressed	
0 no	) key is pressed	
3170	SW2 is pressed	
3166	SW2 is pressed	
0 no	) key is pressed	
0 no	) key is pressed	
4095	SW1 is pressed	
4095	SW1 is pressed	
0 no	) key is pressed	

# 5.2.30 Project 30: Joystick Module



#### Overview

Game handle controllers are ubiquitous. It mainly uses PS2 joysticks. When controlling it, we need to connect the X and Y ports of the module to the analog port of the single-chip microcomputer, port B to the digital port of the single-chip microcomputer, VCC to the power output port(3.3-5V), and GND to the GND of the MCU. We can read the high and low levels of two analog values and one digital port) to determine the working status of the joystick on the module.

In the experiment, two analog values(x axis and y axis) will be shown on Shell.

### **Working Principle**



In fact, its working principle is very simple. Its inside structure is equivalent to two adjustable potentiometers and a button. When this button is not pressed and the module is pulled down by R1, low levels will be output ; on the contrary, when the button is pressed, VCC will be connected (high levels). When we move the joystick, the internal potentiometer will adjust to output different voltages, and we can read the analog value.

## Components



### **Connection Diagram**



fritzing

### Test Code

```
from machine import Pin, ADC
import time
# Initialize the joystick module (ADC function)
rocker_x=ADC(Pin(34))
rocker_y=ADC(Pin(35))
button_z=Pin(13,Pin.IN,Pin.PULL_UP)
# Set the acquisition range of voltage of the two ADC channels to 0-3.3V,
# and the acquisition width of data to 0-4095.
rocker_x.atten(ADC.ATTN_11DB)
rocker_y.atten(ADC.ATTN_11DB)
rocker_x.width(ADC.WIDTH_12BIT)
rocker_y.width(ADC.WIDTH_12BIT)
# In the code, configure Z_Pin to pull-up input mode.
# In loop(), use Read () to read the value of axes X and Y
# and use value() to read the value of axis Z, and then display them.
while True:
    print("X,Y,Z:",rocker_x.read(),",",rocker_y.read(),",",button_z.value())
    time.sleep((0.5))
```

### **Code Explanation**

In the experiment, according to the wiring diagram, the x pin is set to GPIO34, the y pin is set to GPIO35 and the pin of the joystick is set to GPIO13.

### **Test Result**

Wire up, power on and click  $\mathbf{Q}$  "Run current script", the code starts executing. The "Shell" window will print the analog and digital values of the current joystick. Moving the joystick or pressing it will change the analog and digital

values in "Shell". Press"Ctrl+C"or click ""Stop/Restart backend" to exit the program.



# 5.2.31 Project 31: Relay Module

## Overview

In our daily life, we usually use communication to drive electrical equipments, and sometimes we use switches to control electrical equipments. If the switch is connected directly to the ac circuit, leakage occurs and people are in danger. Therefore, from the perspective of safety, we specially designed this relay module with NO(normally open) end and NC(normally closed) end.

## **Working Principle**

Relay is compatible with a variety of micro-controller control board, such as Arduino series micro-controller, which is a small current to control the operation of large current "automatic switch".

Input Voltage3.3V-5V



It can let the MCU control board drive 3A load, such as an LED lamp belt, a DC motor, a micro water pump and a solenoid valve plugable interface design, which is easy to use.

### **Components Required**



### **Connection Diagram**



**Test Code** 

```
from machine import Pin
import time
# create relay from Pin 15, Set Pin 15 to output
relay = Pin(15, Pin.OUT)
# The relay is opened, COM and NO are connected on the relay, and COM and NC are.
\rightarrow disconnected.
def relay_on():
    relay(1)
# The relay is closed, the COM and NO on the relay are disconnected, and the COM and NC.
\rightarrow are connected.
def relay_off():
    relay(≬)
# Loop, the relay is on for one second and off for one second
while True:
    relay_on()
    time.sleep(1)
    relay_off()
    time.sleep(1)
```

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}^{*}$ Run current script", the code starts executing. The relay will cycle on and off, on for 1 second, off for 1 second. At the same time, you can hear the sound of the relay on and off as well as see the change of the indicator light on the relay.

Press"Ctrl+C"or click <sup>22</sup> "Stop/Restart backend" to exit the program.

# 5.2.32 Project 32: SK6812 RGB Module



### Overview

In previous lessons, we learned about the plug-in RGB module and used PWM signals to color the three pins of the module.

There is a Keyestudio 6812 RGB module whose the driving principle is different from the plug-in RGB module. It can only control with one pin. This is a set. It is an intelligent externally controlled LED light source with the control circuit and the light-emitting circuit. Each LED element is the same as a 5050 LED lamp bead, and each component is a pixel. There are four lamp beads on the module, which indicates four pixels.

In the experiment, we make different lights show different colors.

### **Working Principle**

From the schematic diagram, we can see that these four pixel lighting beads are all connected in series. In fact, no matter how many they are, we can use a pin to control a light and let it display any color. The pixel point contains a data latch signal shaping amplifier drive circuit, a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.

The data protocol adopts a single-wire zero-code communication method. After the pixel is powered up and reset, the S terminal receives the data transmitted from the controller. The first 24bit data sent is extracted by the first pixel and sent to the data latch of the pixel.



### Components

	CONTRACTOR							
ESP32 Board*1	ESP32 Board*1	Expansion	Keyestudio 6812 RGB Mod- ule*1	3P Wire*1	Dupont	Micro ble*1	USB	Ca-

### **Connection Diagram**



fritzing

**Test Code** 

```
#Import Pin, neopiexl and time modules.
from machine import Pin
import neopixel
import time
#Define the number of pin and LEDs connected to neopixel.
pin = Pin(15, Pin.OUT)
np = neopixel.NeoPixel(pin, 4)
#brightness :0-255
brightness=100
colors=[[brightness,0,0],
                                              #red
        [0, brightness, 0],
                                              #green
        [0,0,brightness],
                                              #blue
        [brightness, brightness], #white
        [0, 0, 0]]
                                              #close
#Nest two for loops to make the module repeatedly display five states of red, green,
\rightarrow blue, white and OFF.
while True:
    for i in range(0,5):
        for j in range((0, 4)):
            np[j]=colors[i]
            np.write()
            time.sleep_ms(50)
        time.sleep_ms(500)
    time.sleep_ms(500)
```

### **Code Explanation**

A few function ports and functions:

**np = neopixel.NeoPixel(pin, 4)**, there are four LED beads, so we set to 4.

pin = Pin(15, Pin.OUT), this is the pin number, we connect to GP15.

brightness = 100, brightness setting 255 implies brightest.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{O}$  "Run current script", the code starts executing. Then we can see the four RGB LEDs show various colors.

Press "Ctrl+C"or click "" "Stop/Restart backend" to exit the program.

# 5.2.33 Project 33: Rotary Encoder



### Overview

In this kit, there is a Keyestudio rotary encoder, dubbed as switch encoder. It is applied to automotive electronics, multimedia audio, instrumentation, household appliances, smart home, medical equipment and so on.

In the experiment, it it used for counting. When we rotate the rotary encoder clockwise, the set data falls by 1. If you rotate it anticlockwise, the set data is up 1, and when the middle button is pressed, the value will be show on Shell.

### **Working Principle**

The incremental encoder converts the displacement into a periodic electric signal, and then converts this signal into a counting pulse, and the number of pulses indicates the size of the displacement.

This module mainly uses 20pulse rotary encoder components. It can calculate the number of pulses output during clockwise and reverse rotation. There is no limit to count rotation. It resets to the initial state, that is, starts counting from 0.



Components



## **Connection Diagram**



fritzing

## Add Library

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. ESP32\_code\_MicroPython"  $\rightarrow$  "lesson 30. Rotary encoder counting". Select"rotary.py"and"rotary\_irq\_rp2.py"right-click"Upload to /"

ाद्ध Thonny File Edit View Run	Tools Help		-		×
	0-3e 🕨 👳				
Files ×					
This computer D: \ESP32_Python_code	e(Windows)				
🕀 🔃 lesson 31. Relay	P.C.P.				
H Iesson 32. SK6812	KGB encoder count				
lesson 33 Rot	arv encoder c				
🚭 rotary.py	Open in Thonny	<pre>pp on v1.17 on 2021-09-02; ESP32 module with ESP32 p()" for more information. v MicroPython (ESP32)</pre>			
🖏 rotary_irq_r	<pre>e Edit View Run Tools Help  Files ×  Files</pre>				
Image: Second state in the second s					
MicroPython device	Upload to /	-			
File Edit View Run Tools Help Files × This computer D: \ESP32_Python_code(Windows) P \[ lesson 31. Relay P \[ lesson 32. SK6812 RGB P \[ lesson 33. Rotary encoder count P lesson 33. Rotary encoder count P lesson 33. Rotary encoder count P lesson 34. Serve MicroPython device P boot.py Move to Recycle Bin New directory Properties P on v1.17 on 2021-09-02; ESP32 module with ESP32 p()" for more information.	^				
	New directory	on VI.17 on 2021-09-02; ESP32 module Wi	- C ×		
	Properties	-P() for more information.			*
		-	MicroPyt	thon (ESP	32)

द्धि Thonny		— 🗆	×
File Edit View Run Tool	s Help		
🗋 🚰 📕 🔘 🌴 👁 S	3. <u>.</u> e 🕩 🚥		
Files ×			
This computer D: \ESP32_Python_code(Wind	≡ ^ dows)		
<ul> <li></li></ul>			
🖃 🔑 lesson 33. Rotary encod	ler count		
desson_33_Rotary_er	ncoder_c		
de rotary.py			
🚭 rotary_irq_rp2.py	Open in Thonny		
🕀 🕌 lesson 34. Servo	Open in system default app		□ ×
	Configure .pv files		
MicroPython device	configure ipy mesin		
😓 boot.py	Upload to /		^
🖶 rotary.py	View Run Tools Help Nuter Python_code(Windows) on 31. Relay on 32. SK6812 RGB on 33. Rotary encoder count lesson_33_Rotary_encoder_c rotary.py rotary.irq_rp2.py Open in Thonny Open in Thonny Open in system default app Configure .py files Upload to / New directory 17 on 2021-09-02; ESP32 module with ESP32 for more information.		
	New directory	for more information	
		TOP MORE INTORMALION.	~
	Properties	N Care Diale and	(ECD22)
		IVII CTOPVTNON	(ESP32)

**Test Code** 

(continues on next page)

(continued from previous page)

```
reverse=False,
              range_mode=RotaryIRQ.RANGE_UNBOUNDED)
val_old = r.value()
while True:
    trv:
        val_new = r.value()
        if SW.value()==0 and n==0:
            print("Button Pressed")
            print("Selected Number is : ",val_new)
            n=1
            while SW.value()==0:
                continue
        n=0
        if val_old != val_new:
            val_old = val_new
            print('result =', val_new)
        time.sleep_ms(50)
    except KeyboardInterrupt:
        break
```

### **Code Explanation**

1). We will see the file rotary.py and rotary\_irq\_rp2.py. This means that we save them in the ESP32 successfully. Then we can use **from rotary\_irq\_rp2 import RotaryIRQ**.

2). **SW=Pin(20,Pin.\IN,Pin.PULL\_UP)** indicates that the SW pin is connected to GPIO27, **pin\_num\_clk=12** indicates that the pin CLK is connected to GPIO12, and **pin\_num\_dt=14** means that the DT pin is connected to GPIO14. We can change these pin numbers.

3). **try/except** is the python language exception capture processing statement, **try** executes the code, **except** executes the code when an exception occurs, and when we press Ctrl+C, the program exits.

4). **r.value()** returns the value of the encoder

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{O}$  "Run current script", the code starts executing. Rotate the encoder clockwise, the displayed data decrease, rotate the encoder counterclockwise, the displayed data increase.

Press the middle button of the encoder, the displayed data is the value of the encoder, as shown in the figure below.

Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

Shell ×	
result = 1	^
result = 2	
result = 3	
result = 4	
result = 3	
result = 2	
result = 1	
result = 0	
result = -1	
result = -2	
result = -3	
Button Pressed	
Selected Number is : -3	
Button Pressed	
Selected Number is : -3	
Button Pressed	
Selected Number is : -3	
	~

# 5.2.34 Project 34: Servo Control



### Overview

Servo is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor.

In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.



### **Working Principle**

The rotation angle of servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms (50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle from  $0^{\circ}$  to  $180^{\circ}$ . But note that for different brand motors, the same signal may have different rotation angles.



# Components



## **Connection Diagram**



fritzing

Test Code 1

```
from machine import Pin, PWM
import time
pwm = PWM(Pin(4))
pwm.freq(50)
m
Duty cycle corresponding to the Angle
0°----2.5%----25
45°----5%----51.2
90°----7.5%----77
135°----10%----102.4
180°----12.5%----128
angle_0 = 25
angle_{90} = 77
angle_{180} = 128
while True:
    pwm.duty(angle_0)
    time.sleep(1)
    pwm.duty(angle_90)
```

(continues on next page)

(continued from previous page)

```
time.sleep(1)
pwm.duty(angle_180)
time.sleep(1)
```

### **Code Explanation 1**

According to the angle of the signal pulse width, it is converted into a duty cycle. The formula is: 2.5+angle/180\*10. The PWM pin resolution of ESP32 is  $2^{10} = 1024$ . When converted to 0 degree, its duty cycle is  $1024 \times 2.5\% = 25.6$ , when the angle is 180 degrees, its duty cycle value is  $1024 \times 12.5\% = 128$ , these two values will be related to the program, considering the error and rotation angle, I set the duty cycle at between 10 and 150, the servo can rotate smoothly 0~180 degrees

### **Test Result 1**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code

starts executing, the servo will rotate 0°90° and 180° cyclically. Press "Ctrl+C"or click "Stop/Restart backend" to exit the program.

Test Code 2

```
from utime import sleep
from machine import Pin
from machine import PWM
pwm = PWM(Pin(4))#Steering gear pin is connected to GP4.
pwm.freq(50)#20ms period, so the frequency is 50Hz
Duty cycle corresponding to the Angle
0°----2.5%----25
45°----5%----51.2
90°----7.5%----77
135°----10%----102.4
180°----12.5%----128
# Set the servo motor rotation Angle
def setServoCycle (position):
   pwm.duty(position)
    sleep(0.01)
# Convert the rotation Angle to duty cycle
def convert(x, i_m, i_M, o_m, o_M):
   return max(min(o_M, (x - i_m) * (o_M - o_m) // (i_M - i_m) + o_m), o_m)
while True:
    for degree in range(0, 180, 1):#servo goes from 0 to 180
        pos = convert(degree, 0, 180, 20, 150)
        setServoCycle(pos)
    for degree in range(180, 0, -1):#servo goes from 180 to 0
        pos = convert(degree, 0, 180, 20, 150)
        setServoCycle(pos)
```

### **Code Explanation 2**

convert(x, i\_m, i\_M, o\_m, o\_M): x is the value we want to map; i\_m, i\_M are the lower and upper limits of the current

value; o\_m, o\_M are the lower and upper limits of the target range we want to map to.

### Test Result 2

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{O}$  "Run current script", the code starts executing. The servo rotates from 0° to 180° by moving 1° for each 15ms.

Press "Ctrl+C"or click ""Stop/Restart backend" to exit the program.

# 5.2.35 Project 35: Ultrasonic Sensor



Bats and some marine animals are able to use high frequencies of sound for echolocation or communication. They can emit ultrasonic waves from the larynx through the mouth or nose and use the sound waves that bounce back to orient and determine the position, size and whether nearby objects are moving.

Ultrasonic is a frequency higher than 20000 Hz sound wave, which has a good direction, a strong penetration ability, and is easy to obtain more concentrated sound energy as well as spread far in the water. It can be used for ranging, speed measurement, cleaning, welding, gravel, sterilization and disinfection. What's more, it has many applications in medicine, military, industry and agriculture.

### Overview

In this kit, there is a keyes HC-SR04 ultrasonic sensor, which can detect obstacles in front and the detailed distance between the sensor and the obstacle. Its principle is the same as that of bat flying. It can emit the ultrasonic signals that cannot be heard by humans. When these signals hit an obstacle and come back immediately. The distance between the sensor and the obstacle can be calculated by the time gap of emitting signals and receiving signals.

In the experiment, we use the sensor to detect the distance between the sensor and the obstacle, and print the test result.

### **Working Principle**

The most common ultrasonic ranging method is the echo detection. As shown below; when the ultrasonic emitter emits the ultrasonic waves towards certain direction, the counter will count. The ultrasonic waves travel and reflect back once encountering the obstacle. Then the counter will stop counting when the receiver receives the ultrasonic waves coming back.

The ultrasonic wave is also sound wave, and its speed of sound V is related to temperature. Generally, it travels 340m/s in the air. According to time t, we can calculate the distance s from the emitting spot to the obstacle.

s = 340t/2

The HC-SR04 ultrasonic ranging module can provide a non-contact distance sensing function of 2cm-400cm, and the ranging accuracy can reach as high as 3mm; the module includes an ultrasonic transmitter, receiver and control circuit. Basic working principle:

1). First pull down the TRIG, and then trigger it with at least 10us high level signal;

2). After triggering, the module will automatically transmit eight 40KHZ square waves, and automatically detect whether there is a signal to return.

3). If there is a signal returned back, through the ECHO to output a high level, the duration time of high level is actually the time from emission to reception of ultrasonic.



### Components

	Discontinue de la contraction de la contractione				~	
ESP32 Board*1	ESP32 Board*1	Expansion	keyestudio S Sensor*1	R01 Ultrasonic	4P Dupont Wire*1	Micro USB Ca- ble*1

### **Connection Diagram**



fritzing

**Test Code** 

```
from machine import Pin
import time
# Define the control pins of the ultrasonic ranging module.
Trig = Pin(13, Pin.OUT, ◊)
Echo = Pin(14, Pin.IN, \emptyset)
distance = 0 # Define the initial distance to be 0.
soundVelocity = 340 #Set the speed of sound.
# The getDistance() function is used to drive the ultrasonic module to measure distance,
# the Trig pin keeps at high level for 10us to start the ultrasonic module.
# Echo.value() is used to read the status of ultrasonic module's Echo pin,
# and then use timestamp function of the time module to calculate the duration of Echo
# pin's high level,calculate the measured distance based on time and return the value.
def getDistance():
   Trig.value(1)
   time.sleep_us(10)
   Trig.value(♥)
   while not Echo.value():
        pass
   pingStart = time.ticks_us()
   while Echo.value():
       pass
   pingStop = time.ticks_us()
   pingTime = time.ticks_diff(pingStop, pingStart) // 2
   distance = int(soundVelocity * pingTime // 10000)
   return distance
# Delay for 2 seconds and wait for the ultrasonic module to stabilize,
# Print data obtained from ultrasonic module every 500 milliseconds.
time.sleep(2)
while True:
    time.sleep_ms(500)
```

(continues on next page)

(continued from previous page)

```
distance = getDistance()
print("Distance: ", distance, "cm")
```

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. The "Shell" window will print the distance between the ultrasonic sensor and the object.

Press"Ctrl+C"or click <sup>999</sup> "Stop/Restart backend" to exit the program.

Shell ×	
Distance:	4 Cm
Distance:	3 cm
Distance:	4 cm
Distance:	5 cm
Distance:	6 cm
Distance:	6 cm
Distance:	7 cm
Distance:	7 cm
Distance:	8 cm
Distance:	9 cm
Distance:	9 cm
Distance:	10 cm
Distance:	11 cm
Distance:	11 cm
Distance:	12 cm

# 5.2.36 Project 36: IR Receiver Module



### Overview

Infrared remote control is currently the most widely used means of communication and remote control, which has the characteristics of small volume, low power consumption, strong function and low cost. Therefore, recorder, audio equipment, air conditioning machine and toys and other small electrical devices have also used the infrared remote control.

Its transmitting circuit is the use of infrared light emitting diode to emit modulated infrared light wave. The circuit is composed of infrared receiving diode, triode or silicon photocell. They convert infrared light emitted by infrared emitter into corresponding electrical signal, and then send back amplifier.

In this experiment, we need to know how to use the infrared receiving sensor, which mainly uses the VS1838B infrared receiving sensor element. It integrates receiving, amplifying, and demodulating. The internal IC has already completed the demodulation, and the output is a digital signal. It can receive 38KHz modulated remote control signal. In the experiment, we use the IR receiver to receive the infrared signal emitted by the external infrared transmitting device, and display the received signal in the shell.

### **Working Principle**

The main part of the IR remote control system is modulation, transmission and reception. The modulated carrier frequency is generally between 30khz and 60khz, and most of them use a square wave of 38kHz and a duty ratio of 1/3. A 4.7K pull-up resistor R3 is added to the signal end of the infrared receiver.



Components



# **Connection Diagram**



fritzing

Test Code

import utime
from machine import Pin
ird = Pin(15,Pin.IN)

(continues on next page)

```
(continued from previous page)
act = {"1": "LLLLLLLHHHHHHHHHHHHHHLHHLLHLLHLHHHH","2": "LLLLLLLLHHHHHHHHHHHHLLHHLLLHHHLHHH","3
↔": "LLLLLLLHHHHHHHHHHHHHLHHLLLLLHLLHHHH",
      "4": "LLLLLLLHHHHHHHHHLLHHLLLHHLLLHHHH", "5": "LLLLLLLLHHHHHHHHHHLLLHHLLLHHHLLHHHL
"7": "LLLLLLLHHHHHHHHLLLHLLLLHHHHLHHHH", "8": "LLLLLLLLHHHHHHHHHLLHHHLLLHHHLLLHHH", "9
↔": "LLLLLLLHHHHHHHHHHHHHLHLHLHLHLHLHL",
      "0": "LLLLLLLLHHHHHHHHLHLLHLHLHLHLHLHHLHH", "Up": "LLLLLLLLHHHHHHHHHHHLHHLLLHLHLHLHHHLH",
→ "Down": "LLLLLLLHHHHHHHHHHHHHHLHLLLLHLHLHHHH",
      "Left": "LLLLLLLHHHHHHHHHLLHLLHLHLHHHHHHH", "Right":
→"LLLLLLLHHHHHHHHHHLLLLHLLHHHHHHH", "Ok": "LLLLLLLHHHHHHHHHLLLLLLHHHHHHHHHH,
       "*": "LLLLLLLLHHHHHHHHLHLLLLHLHLHHHHHLH","#": "LLLLLLLLHHHHHHHHHHLHLHLHLHLHLHHLH"}
def read_ircode(ird):
   wait = 1
   complete = 0
   seq0 = []
   seq1 = []
   while wait == 1:
        if ird.value() == 0:
            wait = 
    while wait == 0 and complete == 0:
        start = utime.ticks_us()
        while ird.value() == 0:
           ms1 = utime.ticks_us()
        diff = utime.ticks_diff(ms1,start)
        seq0.append(diff)
        while ird.value() == 1 and complete == 0:
            ms2 = utime.ticks_us()
            diff = utime.ticks_diff(ms2,ms1)
            if diff > 10000:
                complete = 1
        seq1.append(diff)
   code = ""
    for val in seq1:
        if val < 2000:
            if val < 700:
                code += "L"
            else:
                code += "H"
    # print(code)
   command = ""
    for k,v in act.items():
        if code == v:
            command = k
   if command == "":
        command = code
   return command
while True:
```

command = read\_ircode(ird)

(continued from previous page)

print(command)
utime.sleep(0.5)

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing. Find the infrared remote control, pull out the insulating sheet, and press the button at the receiving head of the infrared receiving sensor. After receiving the signal, the LED on the infrared receiving sensor also starts to flash, as shown in the figure below.

Press "Ctrl+C"or click "" "Stop/Restart backend" to exit the program.

Shell ×	
Up	^
Left	
Down	
Right	
Ok	
1	
2	
3	
4	
5	
6	
7	
8	
9	
*	
0	
ŧ	
	$\sim$



# 5.2.37 Project 37: DS18B20 Temperature Sensor

### Description

In this kit, there is a DS18B20 temperature sensor, which is from maxim. The MCU can communicate with the DS18B20 through 1-Wire protocol, and finally read the temperature. In this experiment, we will use this temperature sensor to measure the temperature in the current environment. The test result is °C, ranging from -55°C to +125°C. We will display the test result on shell.

### **Working Principle**

The hardware interface of the 1-Wire bus is very simple, just connect the data pin of the DS18B20 to an IO port of the microcontroller. The timing of the 1-Wire bus is relatively complex. Many students can't understand the timing diagram independently here. We have encapsulated the complex timing operations in the library, and you can use the library functions directly.

#### Schematic Diagram of DS18B20

This can save up to 12-bit temperature vale. In the register, save in code complement. As shown below;



A total of 2 bytes, LSB is the low byte, MSB is the high byte, where MSb is the high byte of the byte, LSb is the low byte of the byte. As you can see, the binary number, the meaning of the temperature represented by each bit, is expressed. Among them, S represents the sign bit, and the lower 11 bits are all powers of 2, which are used to represent the final temperature. The temperature measurement range of DS18B20 is from -55 degrees to +125 degrees, and the expression form of temperature data, S represents positive and negative temperature, and the resolution is 2, which is 0.0625.



### **Required Components**

		LEE22 Terrpersture		
ESP32	ESP32 Expansion	Keyestudio DIY 18B20 Tempera-	3P Dupont	Micro USB Ca-
Board*1	Board*1	ture Sensor*1	Wire*1	ble*1

## **Required Components**



fritzing

## Add Library

Open"Thonny", click"This computer"→"D:"→"2. ESP32\_code\_MicroPython"→"lesson 37. DS18B20".

Select"ds18x20.py"and"onewire.py"right-click and select"Upload to"waiting for the "ds18x20.py" and "onewire.py"to be uploaded to the ESP32.



ाति Thonny File Edit View Run	Tools Help	- 0	×
🗋 🚰 🔜 🌑 🌞	33. R 🕨 🐡		
This computer D: \ESP32_Python_code	(Windows) nic iver 0 8b20.py		
🤪 onewire.py 관 🚺 lesson 38. XHT11	Open in Thonny Open in system default app Configure, py files		
MicroPython device	Upload to / Move to Recycle Bin New directory	n v1.17 on 2021-09-02; ESP32 module with ESP32 ()" for more information.	^
	Properties	MicroPython (	ESP32)

### **Test Code**

```
import machine, onewire, ds18x20, time
ds_pin = machine.Pin(15)
ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))
roms = ds_sensor.scan()
print('Found DS devices: ', roms)
while True:
ds_sensor.convert_temp()
time.sleep_ms(750)
for rom in roms:
    #print(rom)
    print(ds_sensor.read_temp(rom))
time.sleep(1)
```

### **Code Explanation**

1). We set the pin to GPIO15 and obtain the temperature in the unit of  $^{\circ}$ C.

2). The Shell window displays the temperature value. Ds\_sensor. Read\_temp (ROM) indicates the temperature value.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{O}$  "Run current script", the code starts executing, the shell displays the temperature of the current environment, as shown below. Press "Ctrl+C" or

click <sup>•••</sup> "Stop/Restart backend" to exit the program.

Shell 🛛	
21.0	
21.0	
21.0	
21.0	
21.0	
22.1	25
25.4	375
27.1	25
28.3	125
29.4	375
	1



# 5.2.38 Project 38: XHT11 Temperature and Humidity Sensor

### Description

This DHT11 temperature and humidity sensor is a composite sensor which contains a calibrated digital signal output of the temperature and humidity.

DHT11 temperature and humidity sensor uses the acquisition technology of the digital module and temperature and humidity sensing technology, ensuring high reliability and excellent long-term stability. It includes a resistive element and a NTC temperature measuring device.



### **Working Principle**

The communication and synchronization between the single-chip microcomputer and XHT11 adopts the single bus data format. The communication time is about 4ms. The data is divided into fractional part and integer part.

Operation process: A complete data transmission is 40bit, high bit first out. Data format: 8bit humidity integer data + 8bit humidity decimal data + 8bit temperature integer data + 8bit temperature decimal data + 8bit checksum

8-bit checksum: 8-bit humidity integer data + 8-bit humidity decimal data + 8-bit temperature integer data + 8-bit temperature decimal data "Add the last 8 bits of the result.

### **Required Components**



#### **Connection Diagram**



fritzing

Test Code

```
# Import machine, time and dht modules.
import machine
import time
import dht
#Associate DHT11 with Pin(15).
DHT = dht.DHT11(machine.Pin(15))
# Obtain temperature and humidity data once per second and print them out.
while True:
DHT.measure() # Start DHT11 to measure data once.
# Call the built-in function of DHT to obtain temperature
# and humidity data and print them in "Shell".
print('temperature:',DHT.temperature(),'C','humidity:',DHT.humidity(),'%')
time.sleep_ms(1000)
```

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing, the shell displays the temperature and humidity data of the current environment, as shown below.

Press"Ctrl+C"or click <sup>•••</sup> "Stop/Restart backend" to exit the program.

Shell ×		
ocmpersource	n' o namidiol. or e	
temperature:	27 °C humidity: 65 %	
temperature:	27 °C humidity: 66 %	
temperature:	27 °C humidity: 65 %	
temperature:	27 °C humidity: 65 %	
temperature:	28 °C humidity: 69 %	
temperature:	28 °C humidity: 68 %	
temperature:	28 °C humidity: 76 %	
temperature:	28 °C humidity: 74 %	
temperature:	28 °C humidity: 78 %	
temperature:	28 °C humidity: 84 %	
temperature:	29 °C humidity: 85 %	
temperature:	29 °C humidity: 88 %	
temperature:	29 °C humidity: 88 %	
temperature:	29 °C humidity: 88 %	
temperature:	30 °C humidity: 91 %	


## 5.2.39 Project 39: DS1307 Clock Module

#### Overview

This module mainly uses the real-time clock chip DS1307, which is the I2C bus interface chip that has second, minute, hour, day, month, year and other functions as well as leap year automatic adjustment function introduced by DALLAS. It can work independently of CPU, and won't' affected by the CPU main crystal oscillator and capacitance as well as keep accurate time. What's more, monthly cumulative error is generally less than 10 seconds.

The chip also has a clock protection circuit in case of main power failure and runs on a back-up battery that denies the CPU read and write access. At the same time, it contains automatic switching control circuit of standby power supply, so it can guarantee the accuracy of system clock in case of power failure of main power supply and other bad environment.

Going forward, the DS1307 chip internal integration has a certain capacity, with power failure protection characteristics of static RAM, which can be used to save some key data.



the experiment, we use the DS1307 clock module to obtain the system time and print the test results.

#### **Working Principle**

Serial real-time clock records year, month, day, hour, minute, second and week; AM and PM indicate morning and afternoon respectively; 56 bytes of NVRAM store data; 2-wire serial port; programmable square wave output; power failure detection and automatic switching circuit; battery current is less than 500nA.

Pins description

X1, X232.768kHz crystal terminal;

VBAT: +3V input;

SDAserial data;

SCLserial clock;

SQW/OUTsquare waves/output drivers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	1	0 Second	S		Sec	onds		Seconds	00–59
01h	0		10 Minutes	6		Min	utes		Minutes	00–59
02h 0		12		10		На	ure		Hours	1–12
		24	PM/ AM	Hour		no	uis		TIOUIS	00–23
03h	0	0	0	0	0		DAY		Day	01–07
04h	0	0	10 [	Date		Da	ate		Date	01–31
05h	0	0	0	10 Month		Month			Month	01–12
06h		10	Year			Year				00–99
07h	OUT	0	0	SQWE	0	0 0 RS1 RS0		Control		
08h–3Fh									RAM 56 x 8	00h-FFh

#### Components



## **Connection Diagram**



### Add Library

Open "Thonny", click "This computer" $\rightarrow$ "D:" $\rightarrow$ "2. ESP32\_code\_MicroPython" $\rightarrow$ "lesson 39. DS1307 Real Time Clock". Select"urtc.py"right-click and select"Upload to /"waiting for the"urtc.py"to be uploaded to the ESP32.

Image: Thonny     −     □     >       File     Edit     View     Run     Tools     Help	×
Files $\times$	
This computer = D: \ ESP32_Python_code(Windows)	
🗄 📜 lesson 36. IR Receiver	
😑 💫 lesson 39. DS1307 Real Time Clo	
esson_39_DS1307.py	
Open in Thonny	
🗄 🐙 lesson 40. AD, Open in system default app	
MicroPython device Configure .py files	
le boot.py Upload to /	^
Move to Recycle Bin thon v1.17 on 2021-09-02; ESP32 module with ESP32	
New directory <pre>p()" for more information.</pre>	
Properties	Υ
MicroPython (ESP3	32)

**Test Code** 

```
from machine import I2C, Pin
from urtc import DS1307
import utime
i2c = I2C(1,scl = Pin(22),sda = Pin(21),freq = 400000)
rtc = DS1307(i2c)
year = int(input("Year : "))
month = int(input("month (Jan --> 1 , Dec --> 12): "))
date = int(input("date : "))
day = int(input("day (1 --> monday , 2 --> Tuesday ... 0 --> Sunday): "))
hour = int(input("hour (24 Hour format): "))
minute = int(input("minute : "))
second = int(input("second : "))
now = (year,month,date,day,hour,minute,second,0)
rtc.datetime(now)
#(year,month,date,day,hour,minute,second,p1) = rtc.datetime()
while True:
    DateTimeTuple = rtc.datetime()
    print(DateTimeTuple[0], end = '-')
    print(DateTimeTuple[1], end = '-')
print(DateTimeTuple[2], end = ' ')
    print(DateTimeTuple[4], end = ':')
    print(DateTimeTuple[5], end = ':')
    print(DateTimeTuple[6], end = ' week:')
    print(DateTimeTuple[3])
    utime.sleep(1)
```

#### **Code Explanation**

rtc.datetime()Return a tuple of time. When the program is running, we set the "please input" program, run the code, it will prompt us to input the time and date, after the input is completed, the data will be printed every second.

DateTimeTuple[0]: save years

DateTimeTuple[1]: save months

DateTimeTuple[2]: save days

DateTimeTuple[3]: save weeks

Rtc.GetDateTime().Month(): return months

DateTimeTuple[4]: save hours

DateTimeTuple[5]: save minutes

DateTimeTuple[6]: save seconds

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing, the shell will display "Year". Then we enter year, month, day, hour, minute and second, once complete,

printed the data every second, as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

```
Shell ×
>>> %Run -c $EDITOR CONTENT
                                                                             ~
  Year : 2022
 month (Jan --> 1 , Dec --> 12): 3
  date : 30
  day (1 --> monday , 2 --> Tuesday ... 0 --> Sunday): 3
 hour (24 Hour format): 12
 minute : 30
  second : 23
  2022-3-30 12:30:23 week:3
  2022-3-30 12:30:24 week:3
  2022-3-30 12:30:25 week:3
  2022-3-30 12:30:26 week:3
  2022-3-30 12:30:27 week:3
  2022-3-30 12:30:28 week:3
  2022-3-30 12:30:29 week:3
  2022-3-30 12:30:30 week:3
  2022-3-30 12:30:31 week:3
  2022-3-30 12:30:32 week:3
  2022-3-30 12:30:33 week:3
  2022-3-30 12:30:34 week:3
```



## 5.2.40 Project 40: ADXL345 Acceleration Sensor

#### Overview

In this kit, there is a DIY electronic building block ADXL345 acceleration sensor module, which uses the ADXL345BCCZ chip. The chip is a small, thin, low-power 3-axis accelerometer with a high resolution (13 bits) and a measurement range of  $\pm 16g$  that can measure both dynamic acceleration due to motion or impact as well as stationary acceleration such as gravitational acceleration, making the device usable as a tilt sensor.

#### **Working Principle**

The ADXL345 is a complete 3-axis acceleration measurement system with a selection of measurement ranges of  $\pm 2$  g,  $\pm 4$  g,  $\pm 8$  g or  $\pm 16$  g. Its digital output data is in 16-bit binary complement format and can be accessed through an SPI (3-wire or 4-wire) or I2C digital interface.



The sensor can measure static acceleration due to gravity in tilt detection applications, as well as dynamic acceleration due to motion or impact. Its high resolution (3.9 mg/LSB) enables measurement of tilt Angle changes of less than  $1.0^{\circ}$ .

## **Components Required**



## **Connection Diagram**



# fritzing

## Add Library

Open"Thonny", click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. ESP32\_code\_MicroPython" $\rightarrow$ "lesson 40. ADXL345". Select"ADXL345.py"right-click and select"Upload to /"waiting for the "ADXL345.py" to be uploaded to the ESP32.

ाद्धि Thonny File Edit View Run	Tools Help	— C	) ×
🗋 💕 🖩 🛛 🔅	3-3t 🕨 🚥		
This computer         D: \ESP32_Python_code(	Windows) Real Time Clov Den in Thonny Open in system default app Configure .py files		
MicroPython device	Upload to / Move to Recycle Bin New directory		^
	Properties	h v1.17 on 2021-09-02; ESP32 module with ESP32 ()" for more information.	•
		MicroPyth	on (ESP32)

**Test Code** 

```
from machine import Pin
import time
from ADXL345 import adxl345
scl = Pin(22)
sda = Pin(21)
bus = 0
snsr = adxl345(bus, scl, sda)
while True:
    x,y,z = snsr.readXYZ()
    print('x:',x,'y:',y,'z:',z,'uint:mg')
    time.sleep(0.1)
```

## **Code Explanation**

Set IIC pins, select IIC0sda->21, scl->22then assign the value to x, y and z. The shell shows the value of x, y and zunit is mg.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing, the shell will display the corresponding value of the three-axis acceleration in mg, as shown in the

following figure, as shown below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

~

Shell	× .	
SHEIL	~	
83		
adz	x1345	found
x:	-347.	.1 y:
x:	-971.	.1 y:
		-

x:	-347.1 y: 491.4 z: 920.4 uint:mg
x:	-971.1 y: 315.9 z: 249.6 uint:mg
x:	-756.6 y: -947.7 z: -612.3 uint:mg
x:	-374.4 y: -491.4 z: -265.2 uint:mg
x:	-362.7 y: -292.5 z: 237.9 uint:mg
x:	-234.0 y: 405.6 z: 89.7 uint:mg
x:	42.9 y: 600.6 z: 1778.4 uint:mg
x:	-296.4 y: -803.4 z: 93.60001 uint:mg
x:	-257.4 y: -1060.8 z: 491.4 uint:mg
x:	-522.6 y: 951.6 z: 666.9 uint:mg
x:	-378.3 y: 819.0 z: 464.1 uint:mg
x:	-869.7 y: -764.4 z: 967.2 uint:mg
x:	982.8 y: -1692.6 z: -249.6 uint:mg
x:	721.5 y: -854.1 z: 276.9 uint:mg
x:	323.7 y: -655.2 z: 604.5 uint:mg
x:	-561.6 y: 31.2 z: 284.7 uint:mg
x:	-889.2 y: 612.3 z: 432.9 uint:mg
x:	-569.4 y: 19.5 z: 1567.8 uint:mg
x:	-413.4 y: 11.7 z: -288.6 uint:mg
x:	-978.9 y: 1930.5 z: 440.7 uint:mg
x:	-175.5 y: -542.1 z: 304.2 uint:mg
x:	-245.7 y: -448.5 z: 877.5 uint:mg
x:	-499.2 y: 850.2 z: 873.6 uint:mg
x:	7.8 y: 163.8 z: 939.9 uint:mg
x:	11.7 y: 159.9 z: 947.7 uint:mg
x:	11.7 y: 156.0 z: 947.7 uint:mg
x:	15.6 v: 159.9 z: 947.7 uint:mg

~

# 5.2.41 Project 41: TM1650 4-Digit Tube Display



#### Overview

This module is mainly composed of a 0.36 inch red common cathode 4-digit digital tube, and its driver chip is TM1650. When using it, we only need two signal lines to make the single-chip microcomputer control a 4-bitdigit tube, which greatly saves the IO port resources of the control board.

TM1650 is a special circuit for LED (light emitting diode display) drive control. It integrates MCU input and output control digital interface, data latch, LED drivers, keyboard scanning, brightness adjustment and other circuits.

TM1650 has stable performance, reliable quality and strong anti-interference ability.

It can be applied to the application of long-term continuous working for 24 hours.

TM1650 uses 2-wire serial transmission protocol for communication (note that this data transmission protocol is not a standard I2C protocol). The chip can drive the digital tube and save MCU pin resources through two pins and MCU communication.

## **Working Principle**

TM1650 adopts IIC treaty, which uses DIO and CLK buses.



Data command setting: 0x48 means that we light up the digital tube, instead of enable the function of key scanning

B7	B6	B5	Β4	В3	B2	B1	во	Function	Description
$\times$	0	0	0		×	×			Eight-level brightness
$\times$	0	0	1		×	×			One-level brightness
$\times$	0	1	0		×	×			Two-level brightness
$\times$	0	1	1		×	×			Three-level brightness
$\times$	1	0	0		×	×		Brightness setting	Four-level brightness
$\times$	1	0	1		×	×			Five-level brightness
$\times$	1	1	0		×	×			Six-level brightness
$\times$	1	1	1		×	×			Seven-level brightness
$\times$				0	×	×		7/8 segment	8-segment display way
$\times$				1	×	×		display control bit	7-segment display way
$\times$					×	×	0	ON/OFF display hit	Off display
$\times$					×	×	1	ON/OFF display bit	On display

### **Command display setting:**

bit[6:4]set the brightness of tube display, and 000 is brightest

bit[3]set to show decimal points

bit[0]start the display of the tube display

Components



## **Connection Diagram**



fritzing

## Test Code

```
from machine import Pin
import time
# definitions for TM1650
ADDR_DIS = 0x48 #mode command
ADDR_KEY = 0x49 #read key value command
# definitions for brightness
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHTEST
          = 7
on = 1
off = 0
# number:0~9
NUM = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]
# DIG = [0x68,0x6a,0x6c,0x6e]
DIG = [0x6e,0x6c,0x6a,0x68]
DOT = [0, 0, 0, 0]
```

```
clkPin = 22
dioPin = 21
clk = Pin(clkPin, Pin.OUT)
dio = Pin(dioPin, Pin.OUT)
DisplayCommand = 0
def writeByte(wr_data):
    global clk,dio
    for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(♥)
        clk.value(0)
        time.sleep(0.0001)
        clk.value(1)
        time.sleep(0.0001)
        clk.value(♥)
        wr_data <<= 1</pre>
    return
def start():
    global clk,dio
    dio.value(1)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(♥)
    return
def ack():
    global clk,dio
    dy = ≬
    clk.value(♥)
    time.sleep(0.0001)
    dio = Pin(dioPin, Pin.IN)
    while(dio.value() == 1):
        time.sleep(0.0001)
        dy += 1
        if(dy>5000):
            break
    clk.value(1)
    time.sleep(0.0001)
    clk.value(♥)
    dio = Pin(dioPin, Pin.OUT)
    return
def stop():
    global clk,dio
    dio.value(♥)
    clk.value(1)
```

```
time.sleep(0.0001)
    dio.value(1)
    return
def displayBit(bit, num):
    global ADDR_DIS
    if(num > 9 and bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    if(DOT[bit-1] == 1):
        writeByte(NUM[num] | 0x80)
    else:
        writeByte(NUM[num])
    ack()
    stop()
    return
def clearBit(bit):
    if(bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    writeByte(0x00)
    ack()
    stop()
    return
    def setBrightness(b = BRIGHT_TYPICAL):
        global DisplayCommand, brightness
        DisplayCommand = (DisplayCommand & 0x0f)+(b<<4)
        return
    def setMode(segment = 0):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xf7)+(segment<<3)
    return
def displayOnOFF(OnOff = 1):
```

```
global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xfe)+OnOff
    return
def displayDot(bit, 0n0ff):
    if(bit > 4):
        return
    if(0n0ff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return
def InitDigitalTube():
    setBrightness(2)
    setMode(0)
    displayOnOFF(1)
    for _ in range(4):
        clearBit(_)
    return
def ShowNum(num): #0~9999
    displayBit(1,num%10)
    if(num < 10):
        clearBit(2)
        clearBit(3)
        clearBit(4)
    if(num > 9 and num < 100):
        displayBit(2,num//10%10)
        clearBit(3)
        clearBit(4)
    if(num > 99 \text{ and } num < 1000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        clearBit(4)
    if(num > 999 and num < 10000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        displayBit(4,num//1000)
InitDigitalTube()
while True:
    #displayDot(1,on)
                           # on or off, DigitalTube.Display(bit,number); bit=1---4 _
\rightarrownumber=0---9
    for i in range(0,9999):
        ShowNum(i)
        time.sleep(0.01)
```

#### **Code Explanation**

**clkPin = 22dioPin = 21is pin number**CLK is connected to GPIO22DIO is connected to GPIO21. We can set any two pins at random.

**displayBit(bit, num):** show numbers at bit(1~4) bit num(0~9)

clearBit(bit): clear up bit(1~4)

setBrightness(): brightness setting

displayOnOFF() 0 means OFF, 1 means ON

displayDot(bit, OnOff) shows dots0 means OFF, 1 means ON

ShowNum(num): show integer numin the range of 0~9999

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing. The 4-digit tube display will show integer from 0 to 99999, an increase of 1 for each 10ms, then start from 0 once reaching 99999.

Press "Ctrl+C"or click ""Stop/Restart backend"to exit the program.

# 5.2.42 Project 42: HT16K33\_8X8 Dot Matrix Module



#### Overview

What is the dot matrix display?

If we apply the previous circuit, there will be must one IO port to control only one LED. When more LED need to be controlled, we may adopt a dot matrix.

The 8X8 dot matrix is composed of 64 light-emitting diodes, and each light-emitting diode is placed at the intersection of the row line and the column line. Refer to the experimental schematic diagram below, when the corresponding column is set to a high level and a certain row to low, the corresponding diode will light up. For instance, set pin 13 to a high level and pin 9 to low, and then the first LED will light up.

In the experiment, we display icons via this dot matrix.

#### **Working Principle**

As the schematic diagram shown, to light up the LED at the first row and column, we only need to set C1 to high level and R1 to low level. To turn on LEDs at the first row, we set R1 to low level and C1-C8 to high level.

16 IO ports are needed, which will highly waste the MCU resources.

Therefore, we designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, which greatly saves the resources of the single-chip microcomputer.

There are three DIP switches on the module, all of which are set to I2C communication address. The setting method is shown below. A0A1 and A2 are grounded, that is, the address is 0x70.

A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
0(OFF)	O(OFF)	O(OFF)	1 (ON)	O(OFF)	O(OFF)	O(OFF)	1 (ON)	0(OFF)
0X70			0X71			0X72		
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
1 (ON)	1 (ON)	O(OFF)	O(OFF)	O(OFF)	1 (ON)	1 (ON)	0(OFF)	1 (ON)
0X73			0X74			0X75		
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)			
O(OFF)	1 (ON)							
	0X76			0X77				

#### Components

	CONTRACTOR				•	~		
ESP32 Board*1	ESP32 Board*1	Expansion	Keyestudio HT10 Matrix*1	6K338X8 Dot	4P Wire*	Dupont 1	Micro USE ble*1	Ca-

#### **Connection Diagram**



#### **Add Library**

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. ESP32\_code\_MicroPython"  $\rightarrow$  "lesson 42. HT16K33 dot matrix". Select "ht16k33.py" and "ht16k33matrix.py", right-click and select "Upload to /", waiting for the "ht16k33.py" and "ht16k33matrix.py" to be uploaded to the ESP32.



【限 Ihonny		—	υх
File Edit View Run Tools	Help		
	. <u>r</u> IÞ 🚥		
Files ×			
This computer D: \ESP32_Python_code(Wind	e Allows)		
🗄 🐌 lesson 40. ADXL345			
🕀 🕖 lesson 41. TM1650 Four d	digital tı		
🖃 🐌 lesson 42. HT16K33 dot r	matrix		
👘 ht16k33.py			
ht16k33matrix.py	Open in Thonny		
lesson_42_matrix_d	Open in system default app		
	Configure .py files		
MicroPython device	Upload to /		
de boot.py	Move to Recycle Bin		<b>^</b>
😓 ht16k33.py	New directory	17 on 2021-09-02: ESP32 module with ESP3	32
-	Properties	for more information.	
L	>>>	1	¥
		MicroPyt	hon (ESP32)

**Test Code** 

```
# IMPORTS
import utime as time
from machine import I2C, Pin, RTC
from ht16k33matrix import HT16K33Matrix
# CONSTANTS
DELAY = 0.01
PAUSE = 3
# START
if ___name___ == '___main___':
    i2c = I2C(scl=Pin(22), sda=Pin(21))
    display = HT16K33Matrix(i2c)
    display.set_brightness(2)
    # Draw a custom icon on the LED
    icon = b"\x00\x66\x00\x00\x18\x42\x3c\x00"
    display.set_icon(icon).draw()
    # Rotate the icon
    display.set_angle(0).draw()
    time.sleep(PAUSE)
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O "Run current script", the code starts executing. The dot matrix displays a" smile "pattern. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

# 5.2.43 Project 43: LCD\_128X32\_DOT Module



## Description

This is a 128\*32 pixel LCD module, which uses IIC communication mode and ST7567A driver chip. At the same time, the code contains all the English letters and common symbols of the library that can be directly called. When used, we can also set English letters and symbols to display different text sizes in our code. To make it easy to set up the pattern display, we also provide a mold capture software that can convert a specific pattern into control code and then copy it directly into the test code for use.

In the experiment, we will set up the display screen to display various English words, common symbols and numbers.

### **Working Principle**



The module uses the IIC communication principle, the underlying functions have been encapsulated in the library surface, we can directly call the library function, if interested, you can also go to understand the underlying driver of the module.

#### Components



### **Connection Diagram**



#### **Add Library**

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. ESP32\_code\_MicroPython"  $\rightarrow$  "lesson 43. lcd128\_32". Select "lcd128\_32.py" and "lcd128\_32\_fonts.py", right-click and select "Upload to *I*", waiting for the "lcd128\_32.py" and "lcd128\_32\_fonts.py" to be uploaded to the ESP32.



🖡 Thonny		-	- 🗆	×
File Edit View Run Tools	Help			
Files ×	= ^			
D: \ESP32_Python_code(Windo	ws)			
<ul> <li></li></ul>	gital tı atrix			
<ul> <li>Icd128_32.py</li> <li>Icd128_32_fonts.py</li> <li>Iesson_43_Icd128_32_</li> <li>Iesson 44. RFID RC522</li> </ul>	Open in Thonny Open in system default app Configure .py files			
MicroPython device	Upload to /			
🌏 boot.py 🌏 lcd128_32.py	Move to Recycle Bin New directory	17 on 2021-09-02; ESP32 module with	ESP32	^
	Properties >>>	or more information.		~
		M	icroPython (E	SP32)

**Test Code** 

```
import machine
import time
import lcd128_32_fonts
from lcd128_32 import lcd128_32
#i2c config
clock_pin = 22
data_pin = 21
bus = 
i2c_addr = 0x3f
use_i2c = True
def scan_for_devices():
    i2c = machine.I2C(bus,sda=machine.Pin(data_pin),scl=machine.Pin(clock_pin))
    devices = i2c.scan()
    if devices:
        for d in devices:
            print(hex(d))
    else:
        print('no i2c devices')
if use_i2c:
    scan_for_devices()
    lcd = lcd128_32(data_pin, clock_pin, bus, i2c_addr)
    lcd.Clear()
    lcd.Cursor(0, 7)
lcd.Display("KEYES")
lcd.Cursor(1, ∅)
```

```
lcd.Display("ABCDEFGHIJKLMNOPQR")
lcd.Cursor(2, 0)
lcd.Display("123456789+-*/<>=$@")
lcd.Cursor(3, 0)
lcd.Display("%^&() {}:;'|?,.~\\[]")
while True:
```

```
#scan_for_devices()
time.sleep(0.5)
```

#### **Code Explanation**

#### Scan\_for\_devices()

This function is an IIC addressing function; if an IIC device is identified, the IIC address of the device is printed, as shown in the figure:

```
Shell X

MicroPython v1.17 on 2021-09-02; Raspberry Pi Pico with RP2040

Type "help()" for more information.

>>> %Run -c $EDITOR_CONTENT

0x3f
```

If the device is not recognized, print no i2c devices, and then report an error, as shown in the figure:

```
Shell X
File "<stdin>", line 40, in <module>
File "lcd128_32.py", line 229, in Display
File "lcd128_32.py", line 71, in WriteFont
File "lcd128_32.py", line 22, in WriteByte_dat
File "lcd128_32.py", line 27, in reg_write
OSError: [Errno 5] EIO
>>>
```

### lcd.Cursor(0, 7)

In order to set the cursor function, that is, to set the position where the character is displayed on the lcd, the first parameter is the parameter of the row, the second is the parameter of the column, then it is expressed as, the first row, the seventh column starts to display the characters.

#### lcd.Display("KEYES")

In order to set the character content to be displayed, "KEYES" is displayed here

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on.

Click Click

Press "Ctrl+C"or click ""Stop/Restart backend"to exit the program.

# 5.2.44 Project 44: RFID Module



#### Description

RFIDRFID-RC522 radio frequency module adopts a Philips MFRC522 original chip to design card reading circuit, easy to use and low cost, suitable for equipment development and card reader development and so on.

RFID or Radio Frequency Identification system consists of two main components, a transponder/tag attached to an object to be identified, and a transceiver also known as interrogator/Reader.

In the experiment, the data read by the card swipe module is 4 hexadecimal numbers, and we print these four hexadecimal numbers as strings. For example, we read the data of the IC card below: 237, 247,148,90 and the data read from the keychain is: 76, 9, 107, 110. Different IC cards and different key chains have diverse data.

#### **Working Principle**

Radio frequency identification, the card reader is composed of a radio frequency module and a high-level magnetic field. The Tag transponder is a sensing device, and this device does not contain a battery. It only contains tiny integrated circuit chips and media for storing data and antennas for receiving and transmitting signals. To read the data in the tag, first put it into the reading range of the card reader. The reader will generate a magnetic field, and because the magnetic energy generates electricity according to Lenz's law, the RFID tag will supply power, thereby activating the device.





## **Connection Diagram**



### Add Library

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. ESP32\_code\_MicroPython"  $\rightarrow$  "lesson 44. RFID RC522".

Select "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py", right-click and select "Upload to /", waiting for the "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py" to be uploaded to the ESP32.



Te Thonny —	
File Edit View Run Tools Help	
Files ×	
This computer = D: \ ESP32_Python_code(Windows)	
E Disson 44. RFID RC522	
esson_44_mtrc522.py	
mfrc522_i2c.py	
😴 soft_iic.py Open in Thonny	
🗄 🕖 lesson 45. buttor Open in system default app	
MicroPython device Configure .py files	
boot.pv Upload to /	
mfrc522_config. Move to Recycle Bin	<u> </u>
mfrc522_i2c.py New directory pn v1.17 on 2021-09-02; ESP32 module with ESP3	32
Properties ()" for more information.	
MicroPyt	thon (ESP32)

### **Test Code**

```
import machine
import time
from mfrc522_i2c import mfrc522
#i2c config
addr = 0x28
scl = 22
sda = 21
rc522 = mfrc522(scl, sda, addr)
rc522.PCD_Init()
rc522.ShowReaderDetails()
                                     # Show details of PCD - MFRC522 Card Reader details
while True:
   if rc522.PICC_IsNewCardPresent():
        #print("Is new card present!")
        if rc522.PICC_ReadCardSerial() == True:
            print("Card UID:")
           print(rc522.uid.uidByte[0 : rc522.uid.size])
    #time.sleep(1)
```

#### **Code Explanation**

mfrc522\_config.py; This is a configuration file that defines some parameters and commands

mfrc522\_i2c.py; Initialization and read and write functions

Soft\_iic.py; It is the bottom-level read and write function of software I2C. We use the io port to simulate I2C here.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click <sup>(1)</sup> "Run current script", the code

~

starts executing. When we make the IC card and key chain close to the RFID module, the information will be printed out, as shown in the figure below. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

```
Shell ×

MFRC522 Software Version:146 = v2.0

Card UID:

[237, 247, 148, 90]

Card UID:

[237, 247, 148, 90]

Card UID:

[237, 247, 148, 90]

Card UID:

[76, 9, 107, 110]

Card UID:

[76, 9, 107, 110]

Card UID:

[76, 9, 107, 110]

Card UID:

[76, 9, 107, 110]
```

Note: Different RFID-RC522 door cards and key chains have diverse values.

# 5.3 3. Comprehensive Experiments:

The previous projects are related to single sensor or module. In the following part, we will combine various sensors and modules to create some comprehensive experiments to perform special functions.

# 5.3.1 Project 45: Button-controlled LED



#### Overview

In this lesson, we will make an extension experiment with a button and an LED. When the button is pressed and low levels are output, the LED will light up; when the button is released, the LED will go off. Then we can control a module with another module.

### Components



### **Connection Diagram**



fritzing

## Test Code

```
from machine import Pin
import time
led = Pin(4, Pin.OUT) # create LED object from Pin 4,Set Pin 4 to output
                                                                                              . .
button = Pin(15, Pin.IN, Pin.PULL_UP) #Create button object from Pin15,Set GP15 to input
#Customize a function and name it reverseGPIO(), which reverses the output level of the
\hookrightarrow LED
def reverseGPIO():
    if led.value():
        led.value(♥)
                          #Set led turn off
    else:
        led.value(1)
                          #Set led turn on
try:
    while True:
        if not button.value():
             time.sleep_ms(2♥)
                                                                                (continues on next page)
```

```
if not button.value():
    reverseGPIO()
    while not button.value():
        time.sleep_ms(20)
except:
    pass
```

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O "Run current script", the code starts executing. When the button is pressed, the LED will light up, when pressed again, the LED will go off, cycle this operation. Press "Ctrl+C" or "Stop/Restart backend" to exit the program.

# 5.3.2 Project 46: Alarm Experiment



#### Overview

In the previous experiment, we control an output module though an input module. In this lesson, we will make an experiment that the active buzzer will emit sounds once an obstacle appears.

#### Components



## **Connection Diagram**



fritzing

## Test Code

```
from machine import Pin
import time
buzzer = Pin(4, Pin.OUT)
sensor = Pin(15, Pin.IN)
while True:
    buzzer.value(not(sensor.value()))
    time.sleep(0.01)
```

#### **Code Explanation**

When an obstacle is detected, **sensor.value**() will return a low level signal. So when an obstacle is detected, the GPIO4 connected to the buzzer pin will output a high level signal, the buzzer will emit sounds.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing. The active buzzer will emit sound if detecting obstacles; otherwise, it won't emit sound. Press

"Ctrl+C" or click ""Stop/Restart backend" to exit the program.

# 5.3.3 Project 47: Intrusion Detection



### Description

In this experiment, we use a PIR motion sensor to control an active buzzer to emit sounds and the onboard LED to flash rapidly.

## **Required Components**



#### **Connection Diagram**



fritzing

### **Test Code**

```
# Import Pin and time modules.
from machine import Pin
import time
# Define the pins of the Human infrared sensor, led and Active buzzer.
sensor_pir = Pin(15, Pin.IN)
led = Pin(22, Pin.OUT)
buzzer = Pin(4, Pin.OUT)
while True:
      if sensor_pir.value():
          print("Warning! Intrusion detected")
          buzzer.value(1)
          led.value(1)
          time.sleep(0.2)
          buzzer.value(0)
          led.value(♥)
          time.sleep(0.2)
      else:
          buzzer.value(0)
          led.value(♥)
```

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click "Run current script", the code starts executing. If the PIR Motion sensor detects someone moving nearby, the buzzer will emit an alarm, and the LED will flash continuously. At the same time, the "shell" will display Warning! Intrusion detected".

Press "Ctrl+C"or click ""Stop/Restart backend"to exit the program.

$\mathbf{Shell} \times$			
Warning!	Intrusion	detected !	~
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
Warning!	Intrusion	detected !	
TT	T		*

# 5.3.4 Project 48: Extinguishing Robot



## Description

Today we will use Arduino simulation to build an extinguishing robot that will automatically sense the fire and start the fan. In this project we will learn how to build a very simple robot using ESP32, (detecting flames with a flame sensor, blowing out candles with a fan) can teach us basic concepts about robotics. Once you understand the basics below, you can build more complex robots.

### **Components Required**
ESP32 Board*1	ESP32 Expansion Board*1	130 Motor*1
	~	
3P Dupont Wire*1	4P Dupont Wire*1	Micro USB Cable*1
+ Store		Ava and
Battery (provided by yourself)*6	Flame Sensor*1	Battery Holder*1

# **Connection Diagram**



fritzing

# Test Code

# Import Pin and ADCmodules.
from machine import ADC,Pin

```
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
#Two pins of the moto
INA = Pin(15, Pin.OUT) #INA corresponds to IN+
INB = Pin(4, Pin.OUT) #INB corresponds to IN-
while True:
    adcVal=adc.read()
    print(adcVal)
    if adcVal < 3000:</pre>
        #open
        INA.value(♥)
        INB.value(1)
    else:
        #stop
        INA.value(♥)
        INB.value(♥)
    time.sleep(0.1)
```

#### **Code Explanation**

In the code, we set the threshold value to 3000. When the ADC value detected by the flame sensor is lower than the threshold value, the fan will be automatically turned on; otherwise, it will be turned off. For the driving method of the fan, please refer to the 130 Motor.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch on the ESP32 expansion board to the ON end, click  $\bigcirc$  "Run current script", the code starts executing. The shell prints the flame value. When this value is less than 3000, the fan will work to blow out the fire. Basically, the flame value can be set by

yourself. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.

$\mathbf{Shell} \times$		
1665	^	
1814		
1617		
3911		
1742		
1846		
1553		
1488		
2814	×	



# 5.3.5 Project 49: Rotary Encoder control RGB

# Introduction

In this lesson, we will control the LED on the RGB module to show different colors through a rotary encoder.

When designing the code, we need to divide the obtained values by 3 to get the remainders. The remainder is 0 and the LED will become red. The remainder is 1, the LED will become green. The remainder is 2, the LED will turn blue.

# Components



# **Connection Diagram**



# Add Library

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. ESP32\_code\_MicroPython"  $\rightarrow$  "lesson 49. Encoder control RGB".

Select "rotary.py" and "rotary\_irq\_rp2.py", right-click and select "Upload to /", waiting for the "rotary.py" and "rotary\_irq\_rp2.py" to be uploaded to the ESP32.

🖫 Thonny				- 0	×
File Edit View Run	Tools Help				
D 📂 🖩 🛛 🚸	9 3e 🕨 👳				
Files ×					
This computer D: \ESP32_Python_com lesson 47. PIR al. lesson 48. Fire-fi lesson 49. Encom lesson 49. Encom rotary.py rotary.irq_r lesson 50. adjus	le(Windows) srm ghting robot er control RGB icoder_control_F Open in Thonny Open in system default app Configure_ny files				
MicroPython device	Unload to /				
aboot ny	Move to Recycle Bin				
- soonpy	New directory				^
-	Properties p(	v1.17 on 2021-09 )" for more info	9-02; ESP32 module wi rmation.	th ESP32	
	* >>>				~
				MicroPython	(ESP32)

The Thonny		- 0	×
File Edit View Run Tools File Edit View Run Tools Files × This computer D: \ESP32_Python_code(Wind ↓ lesson 47. PIR alarm ↓ lesson 48. Fire-fighting r ↓ lesson 49. Encoder contr ↓ lesson 49. Encoder contr ↓ lesson 49. Encoder contr ↓ lesson 49. Encoder contr ↓ lesson 50. adjust the lite MicroPython device ↓ lesson 50. adjust the lite ↓ lesson 50. ad	Help Help Image: Participation of the second sec	1.17 on 2021-09-02; ESP32 module with ESP32	
	Properties	for more information.	~
		MicroPytho	in (ESP32)

Test Code

```
import time
from rotary_irq_rp2 import RotaryIRQ
from machine import Pin, PWM
```

pwm\_r = PWM(Pin(0))

```
pwm_g = PWM(Pin(2))
pwm_b = PWM(Pin(15))
pwm_r.freq(1000)
pwm_g.freq(1000)
pwm_b.freq(1000)
def light(red, green, blue):
    pwm_r.duty(red)
    pwm_g.duty(green)
    pwm_b.duty(blue)
SW=Pin(27,Pin.IN,Pin.PULL_UP)
r = RotaryIRQ(pin_num_clk=12,
              pin_num_dt=14,
              min_val=0.
              reverse=False,
              range_mode=RotaryIRQ.RANGE_UNBOUNDED)
while True:
    val = r.value()
    print(val%3)
    if val%3 == 0:
        light(4950, 0, 0)
    elif val%3 == 1:
        light(0, 4950, 0)
    elif val%3 == 2:
        light(0, 0, 4950)
    time.sleep(0.1)
```

# **Code Explanation**

In the experiment, we set the val to the remainder of Encoder\_Count divided by 3. Encoder\_Count is the value of the encoder. Then we can set pin GPIO0 (red), GPIO2 (green) and GPIO15 (blue) according to remainders.

Colors of the LEDs can be controlled by remainders.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{O}$  "Run current script", the code starts executing. Rotate the knob of the rotary encoder to display the reminders, which can control colors of LED(red

green blue). Press "Ctrl+C"or click "Stop/Restart backend" to exit the program.

Shell ⊠	
+	
1	
1	
0	
2	
0	
2	
1	
1	
0	
	•

# 5.3.6 Project 50: Rotary Potentiometer



# Introduction

In the previous courses, we did experiments of breathing light and controlling LED with button. In this course, we do these two experiments by controlling the brightness of LED through an adjustable potentiometer. The brightness of LED is controlled by PWM values, and the range of analog values is 0 to 4095 and the PWM value range is 0-255.

After the code is set successfully, we can control the brightness of the LED on the module by rotating the potentiometer.

# **Required Components**



# **Connection Diagram**



fritzing

Test Code

```
from machine import Pin,PWM,ADC
import time

pwm =PWM(Pin(15,Pin.OUT),1000)
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_10BIT)

try:
    while True:
        adcValue=adc.read()
        pwm.duty(adcValue)
        print(adc.read())
        time.sleep_ms(100)

except:
        pwm.deinit()
```

# **Code Explanation**

It is easy to control the brightness of the LED light by a potentiometer. Here we can find that MicroPython unifies the value range of the ADC between 0 and 1023, and assigns values directly, which is simple and convenient.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. Rotating the potentiometer on the module can adjust the brightness of the LED on the LED module.

Press "Ctrl+C"or click "Stop/Restart backend" to exit the program.

# 5.3.7 Project 51: Smart Windows



Description

In life, we can see all kinds of smart products, such as smart home. Smart homes include smart curtains, smart windows, smart TVs, smart lights, and more. In this experiment, we use a steam sensor to detect rainwater, and then achieve the effect of closing and opening the window by a servo.

# **Required Components**



# **Connection Diagram**



fritzing

# Test Code

```
# Import Pin and ADC modules.
from machine import ADC,Pin,PWM
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
```

```
pwm = PWM(Pin(15))#Steering pin connected to GP15
pwm.freq(50)#20ms period, so the frequency is 50Hz
Duty cycle corresponding to the Angle
0°−−−−2. 5%−−−−25
45°----5%----51.2
90°----7.5%----77
135°----10%----102.4
180°----12.5%----128
In consideration of the error, the duty cycle is set at 1000~9000, which can smoothly
→rotate 0~180 degrees
....
angle_0 = 25
angle_{90} = 77
angle_{180} = 128
while True:
    adcVal=adc.read()
    print(adcVal)
    if adcVal > 2000:
        pwm.duty(angle_0)
        time.sleep((0.5))
    else:
        pwm.duty(angle_180)
        time.sleep((0.5))
```

# **Code Explanation**

We can control a servo to rotate by a threshold.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{O}$  "Run current script", the code starts executing. When the sensor detects a certain amount of water, the servo rotates to achieve the effect of closing

or opening windows. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.

# 5.3.8 Project 52: Sound Activated Light



# Introduction

In this lesson, we will make a smart sound activated light using a sound sensor and an LED module. When we make a sound, the light will automatically turn on; when there is no sound, the lights will automatically turn off. How it works? Because the sound-controlled light is equipped with a sound sensor, and this sensor converts the intensity of external sound into a corresponding value. Then set a threshold, when the threshold is exceeded, the light will go on, and when it is not exceeded, the light will turn off.

# Components



# **Connection Diagram**





# Test Code

```
from machine import ADC, Pin
import time
# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
led = Pin(15,Pin.OUT)
while True:
    adcVal=adc.read()
    print(adcVal)
    if adcVal > 600:
        led.value(1)
        time.sleep(3)
    else:
```

led.value(♥)		
time.sleep(0,1)		

### **Code Explanation**

We set the ADC threshold value to 600. If more than 600, LED will be on 3s; on the contrary, it will be off.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing. The shell monitor displays the corresponding volume ADC value. When the analog value of sound is greater than 600, the LED on the LED module will light up, otherwise it will go off. Press "Ctrl+C" or click

"Stop/Restart backend" to exit the program.

$\textbf{Shell} \times$		
U	^	
0		
0		
0		
272		
0		
0		
1349		
545		
0		
64		
805		
0		
714		
	×	

# 5.3.9 Project 53: Fire Alarm



#### Description

In this experiment, we will make a fire alarm system. Just use a flame sensor to control an active buzzer to emit sounds.

**Required Components** 



# **Connection Diagram**



fritzing

Test Code

```
from machine import Pin
import time
buzzer = Pin(15, Pin.OUT)
sensor = Pin(4, Pin.IN)
while True:
    Val = sensor.value()
    print(Val)
    if Val == 0:
        buzzer.value(1)
    else:
        buzzer.value(0)
    time.sleep(0.5)
```

**Code Explanation** 

This flame sensor uses an analog pin and a digital pin. When a flame is detected, the digital pin outputs a low level. In this experiment we will use the digital port.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\bigcirc$  "Run current script", the code starts executing. When the sensor detects the flame, the external active buzzer will emit sounds, otherwise the active buzzer will not emit sounds.

Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.

# 5.3.10 Project 54: Smoke Alarm



Description

In this experiment, we will make a smoke alarm by a TM16504-Digit segment module, a gas sensor and an active buzzer.

# **Required Components**



# **Connection Diagram**



fritzing

# Test Code

# Import Pin and ADC modules.
from machine import ADC,Pin
import time

# Turn on and configure the ADC with the range of 0-3.3V adc=ADC(Pin(34)) adc.atten(ADC.ATTN\_11DB)

```
adc.width(ADC.WIDTH_12BIT)
buzzer = Pin(15, Pin.OUT)
# definitions for TM1650
ADDR DIS = 0x48 #mode command
ADDR_KEY = 0x49 #read key value command
# definitions for brightness
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHTEST
              = 7
on = 1
off = 0
# number:0~9
NUM = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]
# DIG = [0x68,0x6a,0x6c,0x6e]
DIG = [0x6e,0x6c,0x6a,0x68]
DOT = [0, 0, 0, 0]
clkPin = 22
dioPin = 21
clk = Pin(clkPin, Pin.OUT)
dio = Pin(dioPin, Pin.OUT)
DisplayCommand = 0
def writeByte(wr_data):
    global clk,dio
    for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(♥)
        clk.value(♥)
        time.sleep(0.0001)
        clk.value(1)
        time.sleep(0.0001)
        clk.value(♥)
        wr data <<= 1
    return
def start():
    global clk,dio
    dio.value(1)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(♥)
    return
def ack():
```

```
global clk,dio
    dy = ≬
    clk.value(♥)
    time.sleep(0.0001)
    dio = Pin(dioPin, Pin.IN)
    while(dio.value() == 1):
        time.sleep(0.0001)
        dy += 1
        if(dy>5000):
            break
    clk.value(1)
    time.sleep(0.0001)
    clk.value(♥)
    dio = Pin(dioPin, Pin.OUT)
    return
def stop():
    global clk,dio
    dio.value(♥)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(1)
    return
def displayBit(bit, num):
    global ADDR_DIS
    if(num > 9 and bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    if(DOT[bit-1] == 1):
        writeByte(NUM[num] | 0x80)
    else:
        writeByte(NUM[num])
    ack()
    stop()
    return
def clearBit(bit):
    if(bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
```

```
ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    writeByte(0x00)
    ack()
    stop()
    return
    def setBrightness(b = BRIGHT_TYPICAL):
        global DisplayCommand, brightness
        DisplayCommand = (DisplayCommand & 0x0f)+(b<<4)
        return
    def setMode(segment = 0):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xf7)+(segment<<3)</pre>
    return
def displayOnOFF(OnOff = 1):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xfe)+OnOff
    return
def displayDot(bit, OnOff):
    if(bit > 4):
        return
    if(0n0ff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return
def InitDigitalTube():
    setBrightness(2)
    setMode(◊)
    displayOnOFF(1)
    for _ in range(4):
        clearBit(_)
    return
def ShowNum(num): #0~9999
    displayBit(1,num%10)
    if(num < 10):
        clearBit(2)
        clearBit(3)
        clearBit(4)
    if(num > 9 and num < 100):
        displayBit(2,num//10%10)
        clearBit(3)
        clearBit(4)
    if(num > 99 \text{ and } num < 1000):
```

```
displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        clearBit(4)
    if(num > 999 \text{ and } num < 10000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        displayBit(4,num//1000)
InitDigitalTube()
while True:
    adcVal=adc.read()
    print(adcVal)
    ShowNum(adcVal)
    if adcVal > 1000:
        buzzer.value(1)
    else:
        buzzer.value(0)
    time.sleep(0.1)
```

# **Code Explanation**

Define an integer variable val to store the ADC value of the smoke sensor, and then we display the analog value in the four-digit digital tube, and then set a threshold, and when the threshold is reached, the buzzer will sound.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbf{Q}$  "Run current script", the code starts executing. When the concentration of combustible gas exceeds the standard, the active buzzer module will give

an alarm, and the four-digit digital tube will display the concentration value. Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

# 5.3.11 Project 55: Alcohol Sensor



#### Description

In the last experiment, we made a smoke alarm. In this experiment, we combine the active buzzer, the MQ-3 alcohol sensor, and a four-digit digital tube to test the alcohol concentration through the alcohol sensor. Then, the concentration to control the active buzzer alarm and the four-digit digital tube to display the concentration. So as to achieve the simulation effect of alcohol detector.

# **Components Required**



# **Connection Diagram**



# fritzing

# **Test Code**

# Import Pin and ADC modules.
from machine import ADC,Pin
import time

adc=ADC(Pin(34))
adc.atten(ADC.ATTN\_11DB)

```
adc.width(ADC.WIDTH_12BIT)
buzzer = Pin(15, Pin.OUT)
# definitions for TM1650
ADDR_DIS = 0x48 #mode command
ADDR_KEY = 0x49 #read key value command
# definitions for brightness
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHTEST
              = 7
on = 1
off = 0
# number:0~9
NUM = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]
# DIG = [0x68,0x6a,0x6c,0x6e]
DIG = [0x6e,0x6c,0x6a,0x68]
DOT = [0, 0, 0, 0]
clkPin = 22
dioPin = 21
clk = Pin(clkPin, Pin.OUT)
dio = Pin(dioPin, Pin.OUT)
DisplayCommand = 0
def writeByte(wr_data):
    global clk,dio
    for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(♥)
        clk.value(♥)
        time.sleep(0.0001)
        clk.value(1)
        time.sleep(0.0001)
        clk.value(♥)
        wr_data <<= 1
    return
def start():
    global clk,dio
    dio.value(1)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(♥)
    return
def ack():
```

```
global clk,dio
    dy = ≬
    clk.value(♥)
    time.sleep(0.0001)
    dio = Pin(dioPin, Pin.IN)
    while(dio.value() == 1):
        time.sleep(0.0001)
        dy += 1
        if(dy>5000):
            break
    clk.value(1)
    time.sleep(0.0001)
    clk.value(♥)
    dio = Pin(dioPin, Pin.OUT)
    return
def stop():
    global clk,dio
    dio.value(♥)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(1)
    return
def displayBit(bit, num):
    global ADDR_DIS
    if (num > 9 \text{ and } bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    if(DOT[bit-1] == 1):
        writeByte(NUM[num] | 0x80)
    else:
        writeByte(NUM[num])
    ack()
    stop()
    return
def clearBit(bit):
    if(bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
```

ack()

(continued from previous page)

```
stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    writeByte(0x00)
    ack()
    stop()
    return
    def setBrightness(b = BRIGHT_TYPICAL):
        global DisplayCommand, brightness
        DisplayCommand = (DisplayCommand & 0x0f)+(b<<4)
        return
    def setMode(segment = 0):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xf7)+(segment<<3)</pre>
    return
def displayOnOFF(OnOff = 1):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xfe)+OnOff
    return
def displayDot(bit, OnOff):
    if(bit > 4):
        return
    if(0n0ff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return
def InitDigitalTube():
    setBrightness(2)
    setMode(◊)
    displayOnOFF(1)
    for _ in range(4):
        clearBit(_)
    return
def ShowNum(num): #0~9999
    displayBit(1,num%10)
    if(num < 10):
        clearBit(2)
        clearBit(3)
        clearBit(4)
    if(num > 9 and num < 100):
        displayBit(2,num//10%10)
        clearBit(3)
        clearBit(4)
    if(num > 99 \text{ and } num < 1000):
```

```
displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        clearBit(4)
    if(num > 999 \text{ and } num < 10000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        displayBit(4,num//1000)
InitDigitalTube()
while True:
    adcVal=adc.read()
    print(adcVal)
    ShowNum(adcVal)
    if adcVal > 1000:
        buzzer.value(1)
    else:
        buzzer.value(0)
    time.sleep(0.1)
```

# **Code Explanation**

Define an integer variable val to store the ADC value of the alcohol sensor, then we display the analog value in the four-digit display module and set a threshold.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. When different alcohol concentrations are detected, the active buzzer module will alarm, and the

four-digit digital display will show the concentration value. Press "Ctrl+C"or click ""Stop/Restart backend" to exit the program.



# 5.3.12 Project 56: Ultrasonic Radar

# Description



We know that bats use echoes to determine the direction and the location of their preys. In real life, sonar is used to detect sounds in the water. Since the attenuation rate of electromagnetic waves in water is very high, it cannot be used to detect signals, however, the attenuation rate of sound waves in the water is much smaller, so sound waves are most commonly used underwater for observation and measurement.

In this experiment, we will use a speaker module, an RGB module and a 4-digit tube display to make a device for detection through ultrasonic.

# **Required Components**



### **Connection Diagram**



# fritzing

# **Test Code**

from machine import Pin, PWM
import utime
# definitions for TM1650
ADDR\_DIS = 0x48 #mode command
ADDR\_KEY = 0x49 #read key value command

```
# definitions for brightness
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHTEST
              = 7
on = 1
off = 0
# number:0~9
NUM = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]
# DIG = [0x68,0x6a,0x6c,0x6e]
DIG = [0x6e, 0x6c, 0x6a, 0x68]
DOT = [0, 0, 0, 0]
clkPin = 22
dioPin = 21
clk = Pin(clkPin, Pin.OUT)
dio = Pin(dioPin, Pin.OUT)
DisplayCommand = 0
def writeByte(wr_data):
    global clk,dio
    for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(♥)
        clk.value(♥)
        utime.sleep(0.0001)
        clk.value(1)
        utime.sleep(0.0001)
        clk.value(0)
        wr_data <<= 1
    return
def start():
    global clk,dio
    dio.value(1)
    clk.value(1)
    utime.sleep(0.0001)
    dio.value(♥)
    return
def ack():
    global clk,dio
    dy = 0
    clk.value(♥)
    utime.sleep(0.0001)
    dio = Pin(dioPin, Pin.IN)
    while(dio.value() == 1):
        utime.sleep(0.0001)
```

```
dy += 1
        if(dy>5000):
            break
    clk.value(1)
    utime.sleep(0.0001)
    clk.value(♥)
    dio = Pin(dioPin, Pin.OUT)
    return
def stop():
    global clk,dio
    dio.value(♥)
    clk.value(1)
    utime.sleep(0.0001)
    dio.value(1)
    return
def displayBit(bit, num):
    global ADDR_DIS
    if(num > 9 and bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    if(DOT[bit-1] == 1):
        writeByte(NUM[num] | 0x80)
    else:
        writeByte(NUM[num])
    ack()
    stop()
    return
def clearBit(bit):
    if(bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    writeByte(0x00)
    ack()
```

```
stop()
    return
    def setBrightness(b = BRIGHT_TYPICAL):
        global DisplayCommand, brightness
        DisplayCommand = (DisplayCommand & 0x0f)+(b<<4)
        return
    def setMode(segment = 0):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xf7)+(segment<<3)
    return
def displayOnOFF(OnOff = 1):
    global DisplayCommand
    DisplayCommand = (DisplayCommand \& 0xfe)+OnOff
    return
def displayDot(bit, 0n0ff):
    if(bit > 4):
        return
    if(0n0ff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return
def InitDigitalTube():
    setBrightness(2)
    setMode(♥)
    displayOnOFF(1)
    for _ in range(4):
        clearBit(_)
    return
def ShowNum(num): #0~9999
    displayBit(1,num%10)
    if(num < 10):
        clearBit(2)
        clearBit(3)
        clearBit(4)
    if(num > 9 and num < 100):
        displayBit(2,num//10%10)
        clearBit(3)
        clearBit(4)
    if(num > 99 and num < 1000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        clearBit(4)
    if(num > 999 \text{ and } num < 10000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        displayBit(4,num//1000)
```

```
pwm_r = PWM(Pin(0))
pwm_g = PWM(Pin(2))
pwm_b = PWM(Pin(15))
pwm_r.freq(1000)
pwm_g.freq(1000)
pwm_b.freq(1000)
def light(red, green, blue):
    pwm_r.duty(red)
    pwm_g.duty(green)
    pwm_b.duty(blue)
# Ultrasonic ranging, unit: cm
def getDistance(trigger, echo):
    # Generates a 10us square wave
    trigger.value(\emptyset) #A short low level is given beforehand to ensure a clean high-
→pulse:
    utime.sleep_us(2)
    trigger value(1)
    utime.sleep_us(10)#After pulling high, wait 10 microseconds and immediately set it.
→to low
    trigger.value())
    while echo.value() == 0: #Establish a while loop to detect whether the echo pin_
\leftrightarrow value is 0 and record the time at that time
        start = utime.ticks_us()
    while echo.value() == 1: #Establish a while loop to check whether the echo pin value.
\rightarrow is 1 and record the time at that time
        end = utime.ticks_us()
    d = (end - start) * 0.0343 / 2 #The travel time of the sound wave x the speed of_
\rightarrow sound (343.2 m/s, 0.0343 cm/microsecond), and the distance back and forth divided by 2.
    return d
# set the pin
trigger = Pin(13, Pin.OUT)
echo = Pin(14, Pin.IN)
buzzer = PWM(Pin(18))
def playtone(frequency):
    buzzer.duty(1000)
    buzzer.freq(frequency)
def bequiet():
    buzzer.duty(\emptyset)
# main program
InitDigitalTube()
while True:
    distance = int(getDistance(trigger, echo))
    ShowNum(distance)
```

```
if distance <= 10:
    playtone(880)
    utime.sleep(0.1)
    bequiet()
    light(1023, 0, 0)
elif distance <= 20:
    playtone(532)
    utime.sleep(0.2)
    bequiet()
    light(0, 0, 1023)
else:
    light(0, 1023, 0)</pre>
```

# **Code Explanation**

We set sound frequency and light color by adjusting different distance range.

We can adjust the distance range in the code.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click O"Run current script", the code starts executing. When the ultrasonic sensor detects different distances, the buzzer will produce different frequencies of sound(within 20 cm), the RGB will show different colors, and the measured distances are displayed on the 4-digit

tube display. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.



# 5.3.13 Project 57: IR Remote Control

# Introduction

In the previous experiments, we learned how to turn on/off the LED and adjust its brightness via PWM and print the button value of the IR remote control in the Shell window. Herein, we use an infrared remote control to turn on/off an LED.

# Components



# **Connection Diagram**



fritzing

Test Code

import time from machine import Pin
led = Pin(4, Pin.OUT) ird = Pin(15 Pin IN)
$\frac{110 - rm(13, rm, m)}{10}$
<pre>act = {"1": "LLLLLLLHHHHHHHHHHHHHHHHHHHHHHHHHHHHH</pre>
"4": "LLLLLLLHHHHHHHHLLHHLLLHHLLHHHH","5": "LLLLLLLHHHHHHHHHHLLLHHLLHHHLLHHH","6
↔": "LLLLLLHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
"7": "LLLLLLLHHHHHHHHLLLHLLLHHHLHHHH","8": "LLLLLLLHHHHHHHHHHHLLHHHLLHHLLHHH","9
"0": "LLLLLLLHHHHHHHHLHLLHLHLHLHLHLHLH","Up": "LLLLLLLHHHHHHHHHHHLHHLLLHLHLHHHLH",
→"Down": "LLLLLLHHHHHHHHHHHHLHLLLLHLHLHHHH",
(continues on next page)

```
"Left": "LLLLLLLHHHHHHHHHLLHLLHLHLHHHHHHH", "Right":
"*": "LLLLLLLHHHHHHHHHLHLLLLHLHHHHHHLH","#": "LLLLLLLHHHHHHHHHHHHHHHHHHHHHHHHHH
def read_ircode(ird):
   wait = 1
   complete = 0
   seq0 = []
   seq1 = []
   while wait == 1:
       if ird.value() == 0:
           wait = 🛛
   while wait == 0 and complete == 0:
       start = time.ticks_us()
       while ird.value() == 0:
           ms1 = time.ticks_us()
       diff = time.ticks_diff(ms1,start)
       seq0.append(diff)
       while ird.value() == 1 and complete == 0:
           ms2 = time.ticks_us()
           diff = time.ticks_diff(ms2,ms1)
           if diff > 10000:
               complete = 1
       seq1.append(diff)
   code = ""
   for val in seq1:
       if val < 2000:
           if val < 700:
               code += "L"
           else:
               code += "H"
   # print(code)
   command = ""
   for k,v in act.items():
       if code == v:
           command = k
   if command == "":
       command = code
   return command
flag = False
while True:
#
     global flag
   command = read_ircode(ird)
   print(command, end = " ")
   print(flag, end = " ")
   if command == "Ok":
       if flag == True:
           led.value(1)
           flag = False
```

```
print("led on")
else:
    led.value(0)
    flag = True
    print("led off")
time.sleep(0.1)
```

# **Code Explanation**

We define a boolean variable. There are two boolean variables. true (true) or false (false).

When we press the OK button, the value of infrared reception is OK. At this time, we need to set a boolean variable flag. When the flag is true (true), the LED is turned on, and when it is false (false), the LED is turned off and turned on. After the LED is on and set it to false. We press the OK key, the LED will be off.

# **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbb{Q}$  "Run current script", the code starts executing. Press the OK button of the remote, the LED will be on, press it again, the LED will be off.

Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

```
Shell X

Type help() for more information.

>>> %Run -c $EDITOR_CONTENT

Ok False led off

Ok True led on

Ok False led off

Ok True led on

Ok False led off

Ok True led on

Ok False led off
```
# 5. 14 Project 58: Heat Dissipation Device

#### Description

We will use a temperature sensor and some modules to make a smart cooling device in this experiment. When the ambient temperature is higher than a certain value, the motor is turned on, thereby reducing the ambient temperature and achieving the heat dissipation effect. Then display the temperature value in the four-digit segment display.

#### **Required Components**



**Connection Diagram** 



#### Add Library

 $Open ``Thonny", click ``This computer" \rightarrow ``D:" \rightarrow ``2. ESP32\_code\_MicroPython" \rightarrow ``lesson 58. heat abstractor".$ 

Select"ds18x20.py"and"ds18x20.py"right-click and select "Upload to/"waiting for the "ds18x20.py"and"ds18x20.py"to be uploaded to the ESP32.

😱 Thonny		_		×
File Edit View Run	Tools Help			
🗋 💕 🖩 🛛 🔅	🖘 3 et 🕪 🚥			
Files ×				
This computer D: \ESP32_Python_code	(Windows)			
표 🔑 lesson 56. Ultrasor	nic radar			
🕀 🕖 lesson 57. lR contr	rol LED			
esson 58. heat ab	stractor			
lesson 58 hei	Open in Thonny			
onewire.py	Open in system default app			
표 📜 lesson 59. Intellig	Configure .py files			
	Upload to /			
MicroPython device	Move to Recycle Bin			
😔 boot.py	New directory			
	Properties			- ^
	microrycodn v1.17 on 2021-09-02; ESP32 module wit	h ESP	32	
	Type "help()" for more information.			~
	222	MicroPvt	hon (ES	P32)
				,

The Thonny —	o x
File Edit View Run Tools Help	
Image: Second secon	
D: \ESP32_Python_code(Windows)	
🗄 📜 lesson 56. Ultrasonic radar	
Iesson 57. IR control LED	
E lesson 58. heat abstractor	
😴 ds lascon 58, heat abstractor m	
onewire.pv	
Configure ny files	
MicroPython device	
boot.py	
ds18x20.py Move to Recycle Bin	^
n v1.17 on 2021-09-02; ESP32 module with ESP3	2
Properties ()" for more information.	~
MicroPyth	non (ESP32)

### Test Code

```
from machine import Pin
import machine, onewire, ds18x20, time
ds_pin = machine.Pin(13)
ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))
```

```
roms = ds_sensor.scan()
#Two pins of the motor
INA = Pin(15, Pin.OUT) #INA corresponds to IN+
INB = Pin(4, Pin.OUT)#INB corresponds to IN-
# definitions for TM1650
ADDR_DIS = 0x48 #mode command
ADDR_KEY = 0x49 #read key value command
# definitions for brightness
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHTEST
            = 7
on = 1
off = 0
# number:0~9
NUM = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]
# DIG = [0x68,0x6a,0x6c,0x6e]
DIG = [0x6e,0x6c,0x6a,0x68]
DOT = [0, 0, 0, 0]
clkPin = 22
dioPin = 21
clk = Pin(clkPin, Pin.OUT)
dio = Pin(dioPin, Pin.OUT)
DisplayCommand = 0
def writeByte(wr_data):
   global clk,dio
   for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(♥)
       clk.value(♥)
        time.sleep(0.0001)
        clk.value(1)
        time.sleep(0.0001)
        clk.value(♥)
        wr_data <<= 1
   return
def start():
   global clk,dio
   dio.value(1)
   clk.value(1)
   time.sleep(0.0001)
   dio.value(♥)
```

```
return
def ack():
    global clk,dio
    dy = ≬
    clk.value(♥)
    time.sleep(0.0001)
    dio = Pin(dioPin, Pin.IN)
    while(dio.value() == 1):
        time.sleep(0.0001)
        dy += 1
        if(dy>5000):
            break
    clk.value(1)
    time.sleep(0.0001)
    clk.value(♥)
    dio = Pin(dioPin, Pin.OUT)
    return
def stop():
    global clk,dio
    dio.value(♥)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(1)
    return
def displayBit(bit, num):
    global ADDR_DIS
    if(num > 9 and bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    if(DOT[bit-1] == 1):
        writeByte(NUM[num] | 0x80)
    else:
        writeByte(NUM[num])
    ack()
    stop()
    return
def clearBit(bit):
    if(bit > 4):
        return
    start()
```

```
writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    writeByte(0x00)
    ack()
    stop()
    return
    def setBrightness(b = BRIGHT_TYPICAL):
        global DisplayCommand, brightness
        DisplayCommand = (DisplayCommand & 0x0f)+(b<<4)
        return
    def setMode(segment = 0):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xf7)+(segment<<3)</pre>
    return
def displayOnOFF(OnOff = 1):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xfe)+OnOff
    return
def displayDot(bit, OnOff):
    if(bit > 4):
        return
    if(0n0ff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return
def InitDigitalTube():
    setBrightness(2)
    setMode(♥)
    displayOnOFF(1)
    for _ in range(4):
        clearBit(_)
    return
def ShowNum(num): #0~9999
    displayBit(1,num%10)
    if(num < 10):
        clearBit(2)
        clearBit(3)
        clearBit(4)
    if(num > 9 and num < 100):
        displayBit(2,num//10%10)
```

```
clearBit(3)
        clearBit(4)
    if(num > 99 \text{ and } num < 1000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        clearBit(4)
    if(num > 999 and num < 10000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        displayBit(4,num//1000)
InitDigitalTube()
print('Found DS devices: ', roms)
while True:
    ds_sensor.convert_temp()
    time.sleep_ms(750)
    for rom in roms:
        value = ds_sensor.read_temp(rom)
        print(value)
        ShowNum(int(value))
        if value > 28:
            INA.value(♥)
            INB.value(1)
        else:
            INA.value(♥)
            INB.value(♥)
```

#### **Code Explanation**

The setting of variables and the storage of detection values are the same as what we learned earlier. We also set a temperature threshold and control the rotation of the motor when the threshold is exceeded, and then we use the digital tube to display the temperature value.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch on the ESP32 expansion board to the ON end. Click O"Run current script", the code starts executing. We can see the temperature of the current environment (unit is Celsius) on the four-digit segment display, as shown in the figure below. If this value

exceeds the value we set, the fan will rotate to dissipate heat. Press"Ctrl+C"or click <sup>100</sup> "Stop/Restart backend" to exit the program.



## 5.3.15 Project 59: Intelligent Entrance Guard System

#### Description

In this project, we use the RFID522 card swiping module and the servo to set up an intelligent access control system. The principle is very simple. We use RFID522 swipe card module, an IC card or key card to unlock.

#### **Required Components**



#### **Connection Diagram**



#### Add Library

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. ESP32\_code\_MicroPython"  $\rightarrow$  "lesson 59. Intelligent access control". Select "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py", right-click and select "Upload to *l*", waiting for the "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py" to be uploaded to the ESP32.

The Thonny	_		Х
File Edit View Run Tools Help			
Image: Second Property   Image: Second			
lesson_59_Intelligent_access_			
mfrc522_config.py Open in Thonny			
soft iic.pv Open in system default app			
Upload to /			
MicroPython device Move to Recycle Bin			
le boot.py New directory			•
Properties	ith FC	020	
Type "help()" for more information.	ICH LO	r JZ	
			Υ
	MicroP	ython (E	SP32)

The Thomas		_		×
				$\sim$
File Edit View Run Too	ls Help			
🗋 🚰 🖩 🛛 🚸 👁 🕄	3. 🥐 🕨 🚥			
Files $\times$				
This computer D: \ESP32_Python_code(Win	≡ ^ dows)			
<ul> <li>Iesson 58. heat abstract</li> <li>Iesson 59. Intelligent ac</li> <li>Iesson_59_Intelligent</li> <li>Iesson_59_Intelligent</li> <li>mfrc522_config.py</li> </ul>	tor ccess cont nt_access_			
<ul> <li>mfrc522_i2c.py</li> <li>soft_iic.py</li> <li>Iesson 60. WiFi Statio</li> </ul>	Open in Thonny Open in system default app Configure .py files			
MicroPython device	Upload to /			
😔 boot.py 🌏 mfrc522_config.py	Move to Recycle Bin New directory			^
	Properties	/1.17 on 2021-09-02; ESP32 module with " for more information.	ESP32	
	* >>>	Mic	oPython (f	-SP32)

隔 Thonny	_		Х
File Edit View Run Tools Help			
🗋 🚰 📕 💿 🌞 🕾 🗈 🕨 🚥			
Files ×			
This computer       ■         D: \ESP32_Python_code(Windows)       ■         Image: Disconstructure       ■         Image: Disconstru			
soft_iic.py Open in Thonny			
MicroPython device Configure .py files			
boot.py Upload to /			
😔 mfrc522_config.p Move to Recycle Bin			-^
mfrc522_i2c.py New directory n v1.17 on 2021-09-02; ESP32 module wi	th ESP	32	
Properties ()" for more information.			~
	MicroPy	thon (E	SP32)

#### Test Code

Note: Different RFID-RC522 modules, ID cards and keys may different uid1 values and uid2 values.

The uID1 and UID2 values of the white card and key chain read by your RRFID RC522 module can be replaced by the corresponding values in the program code. If not, click **"Run current script"** to run the code may cause your own white card and key chain to fail to control the servo.

For example: You can replace the UID1 and UID2 values In the program code

```
uid1 = [237, 247, 148, 90]
uid2 =
        [76, 9, 107, 110]
                                   with your own white card and key chain values.
from machine import Pin, PWM
import time
from mfrc522_i2c import mfrc522
pwm = PWM(Pin(15))
pwm.freq(50)
Duty cycle corresponding to the Angle
Duty cycle corresponding to the Angle
0°----2.5%----25
45°----5%----51.2
90°----7.5%----77
135°----10%----102.4
180°----12.5%----128
angle_0 = 25
angle_{90} = 77
angle_{180} = 128
#i2c config
addr = 0x28
scl = 22
sda = 21
rc522 = mfrc522(scl, sda, addr)
rc522.PCD_Init()
                                    # Show details of PCD - MFRC522 Card Reader details
rc522.ShowReaderDetails()
uid1 = [237, 247, 148, 90]
uid2 = [76, 9, 107, 110]
pwm.duty(angle_180)
time.sleep(1)
while True:
   if rc522.PICC_IsNewCardPresent():
       print("Is new card present!")
        if rc522.PICC_ReadCardSerial() == True:
            print("Card UID:", end=' ')
            print(rc522.uid.uidByte[0 : rc522.uid.size])
            if rc522.uid.uidByte[0 : rc522.uid.size] == uid1 or rc522.uid.uidByte[0 :_

→rc522.uid.size] == uid2:

                pwm.duty(angle_0)
            else :
                pwm.duty(angle_180)
            time.sleep(500) from machine import Pin, PWM
import time
from mfrc522_i2c import mfrc522
```

```
pwm = PWM(Pin(15))
pwm.freq(50)
Duty cycle corresponding to the Angle
Duty cycle corresponding to the Angle
0°----2.5%----25
45°----5%----51.2
90°----7.5%----77
135°----10%----102.4
180°----12.5%----128
angle_0 = 25
angle_{90} = 77
angle_{180} = 128
#i2c config
addr = 0x28
scl = 22
sda = 21
rc522 = mfrc522(scl, sda, addr)
rc522.PCD_Init()
                                                                                                                               # Show details of PCD - MFRC522 Card Reader details
rc522.ShowReaderDetails()
uid1 = [237, 247, 148, 90]
uid2 = [76, 9, 107, 110]
pwm.duty(angle_180)
time.sleep(1)
while True:
             if rc522.PICC_IsNewCardPresent():
                           print("Is new card present!")
                           if rc522.PICC_ReadCardSerial() == True:
                                         print("Card UID:", end=' ')
                                         print(rc522.uid.uidByte[0 : rc522.uid.size])
                                         if rc522.uid.uidByte[0 : rc522.uid.size] == uid1 or rc522.uid.uidByte[0 :_

where the state of the s
                                                       pwm.duty(angle_0)
                                         else :
                                                       pwm.duty(angle_180)
                                         time.sleep(500)
```

#### **Code Explanation**

In the previous experiment, our card swipe module has tested the information of IC card and key. Then we use this corresponding information to control the door.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Click  $\mathbf{Q}$  "Run current script", the code starts executing. When we use the IC card or blue key to swipe the card, the shell displays the card and the key

information, at the same time, the servo rotates to the corresponding angle to simulate opening the door.

Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

## 5.3.16 Project 60WIFI Station Mode

#### Description

ESP32 has three different WiFi modes: Station mode, AP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi running mode before using, otherwise the WiFi cannot be used. In this project, we are going to learn the WiFi Station mode of the ESP32.

#### Components



#### Wiring Diagram

Plug the ESP32 to the USB port of your PC



#### **Component Knowledge**

#### Station mode

When setting Station mode, the ESP32 is taken as a WiFi client. It can connect to the router network and communicate with other devices on the router via a WiFi connection. As shown in the figure below, the PC and the router have been connected. If the ESP32 wants to communicate with the PC, the PC and the router need to be connected.



Test Code

有	lesson		
^	1 2 3	<pre>import time import network # Import networhameland password.</pre>	^
	4 5 6	<pre>ssidRouter = 'ChinaNet-2.4G-0DF0' # Enter the router name passwordRouter = 'ChinaNet@233' # Enter the router password</pre>	
~	7 8 9 10	<pre>def STA_Setup(ssidRouter,passwordRouter):     print("Setup start")     sta_if = network.WLAN(network.STA_IF) # Set ESP32 in Station mode.     if not sta_if.isconnected():</pre>	
	11 12	<pre>print('connecting to',ssidRouter) # Activate ESP32's Station mode, initiate a connection request to the router</pre>	~
	Shell	×	^
	Micro Type	oPython v1.17 on 2021-09-02; ESP32 module with ESP32 "help()" for more information.	~
		MicroPuthon (ESE	9321

```
import time
import network # Import network module.
               = 'ChinaNet-2.4G-0DF0' # Enter the router name
ssidRouter
passwordRouter = 'ChinaNet@233' # Enter the router password
def STA_Setup(ssidRouter,passwordRouter):
    print("Setup start")
    sta_if = network.WLAN(network.STA_IF) # Set ESP32 in Station mode.
    if not sta_if.isconnected():
        print('connecting to',ssidRouter)
 # Activate ESP32's Station mode, initiate a connection request to the router and enter.
\rightarrow the password to connect.
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
  #Wait for ESP32 to connect to router until they connect to each other successfully.
                                                                                           ш
\rightarrow
        while not sta_if.isconnected():
            pass
  # Print the IP address assigned to ESP32-WROVER in "Shell".
    print('Connected, IP address:', sta_if.ifconfig())
    print("Setup End")
try:
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()
```

#### **Test Result**

Since the router name and password are different in various places, so before running the code, the user needs to enter

the correct router name and password in the red box shown above.

After entering the correct router name and password, click O"Run current script", the code will start executing.

The Shell monitor will print the IP address of the ESP32 when connecting the ESP32 to your router.

```
Shell ×
>>> %Run -c $EDITOR_CONTENT
Setup start
connecting to ChinaNet-2.4G-0DF0
Connected, IP address: ('192.168.1.147', '255.255.255.0', '192.168.1.1'
, '114.114.114.114')
Setup End
>>>
```

## 5.3.17 Project 61WIFI AP Mode

#### Description

In this project, we are going to learn the WiFi AP mode of the ESP32.

#### Components



#### Wiring Diagram

Plug the ESP32 mainboard to the USB port of your PC



#### **Component Knowledge**

#### **AP Mode:**

When setting AP mode, a hotspot network will be created, waiting for other WiFi devices to connect. As shown below;

Take the ESP32 as the hotspot, if a phone or PC needs to communicate with the ESP32, it must be connected to the ESP32's hotspot. Communication is only possible after a connection is established via the ESP32.



Test Code

> *	• •	3. 12 🕨 🚥		
	lesson	_61_WiFi_AP_Mode.py	Set a name and a	
^	1 2	import network	#Import network module password for ESP32 AP.	^
	3 4 5	#Enter correct ssidAP passwordAP	<pre>router base and password = 'ESP32_Wifi' #Enter the router name = '12345678' #Enter the router password</pre>	
* <	7 8 9 10	local_IP gateway subnet dns	= '192.168.1.147' = '192.168.1.1' = '255.255.255.0' = '8.8.8.8'	
	11 12	#Set ESP32 in	AP mode.	~
	Shell	<		
	Micro Type >>>	Python v1.17 o "help()" for m	n 2021-09-02; ESP32 module with ESP32 ore information.	^
¥			Mino Dation (	V

```
import network #Import network module.
#Enter correct router name and password.
           = 'ESP32_Wifi' #Enter the router name
ssidAP
passwordAP = '12345678' #Enter the router password
local_IP
             = '192.168.1.147'
gateway
             = '192.168.1.1'
              = '255.255.255.0'
subnet
              = '8.8.8.8'
dns
#Set ESP32 in AP mode.
ap_if = network.WLAN(network.AP_IF)
def AP_Setup(ssidAP,passwordAP):
   ap_if.ifconfig([local_IP,gateway,subnet,dns])
   print("Setting soft-AP ... ")
   ap_if.config(essid=ssidAP,authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
   ap_if.active(True)
   print('Success, IP address:', ap_if.ifconfig())
   print("Setup End\n")
try:
   AP_Setup(ssidAP,passwordAP)
except:
   print("Failed, please disconnect the power and restart the operation.")
```

~

ap\_if.disconnect()

#### **Test Result**

You can modify the AP name and password or keep them unchanged.

Click Circle Click

```
Shell ×
>>> %Run -c $EDITOR_CONTENT
Setting soft-AP ...
Success, IP address: ('192.168.1.147', '192.168.1.1', '255.255.255.0',
'8.8.8.8')
Setup End
>>>
```

Turn on your phone's WiFi search function, then you can see the ssid\_AP which is called "ESP32\_Wifi" in this code. You can enter the password "12345678" to connect it, or you can modify its AP name and password by code.



### 5.3.18 Project 62WIFI AP+Station Mode

#### Description

In this project, we are going to learn the AP+Station mode of the ESP32.

Components



## Wiring Diagram

Plug the ESP32 mainboard to the USB port of your PC



#### **Component Knowledge**

#### **AP+Station mode**

In addition to the AP mode and the Station mode, **AP+Station mode** can be used at the same time. Turn on the Station mode of the ESP32, connect it to the router network, and it can communicate with the Internet through the router. Then turn on the AP mode to create a hotspot network. Other WiFi devices can be connected to the router network or the hotspot network to communicate with the ESP32.

#### Test Code

) *	k 👁	3e 🕨 📼	Please enter the correct names	
^	1 2	<pre>import network</pre>	#Import network module.	^
	3 4 5	ssidRouter passwordRouter	<pre>= 'ChinaNet-2.4G-0DF0' #Enter the router name = 'ChinaNet@233' #Enter the router password</pre>	
J	6 7	ssidAP passwordAP	<pre>= 'ESP32_Wifi'#Enter the AP name = '12345678' #Enter the AP password</pre>	
^	9 10 11 12 13	local_IP gateway subnet dns	<pre>= '192.168.4.147' = '192.168.1.1' = '255.255.255.0' = '8.8.8.8'</pre>	
	14 15 16	<pre>sta_if = networ ap_if = networ</pre>	rk.WLAN(network.STA_IF) «.WLAN(network.AP_IF)	*
	Shell	×		
~	Micro Type >>>	oPython v1.17 or "help()" for mo	1 2021-09-02; ESP32 module with ESP32 ore information.	•
			MicroPvthon	(ESP32)

```
import network #Import network module.
ssidRouter = 'ChinaNet-2.4G-0DF0' #Enter the router name
passwordRouter = 'ChinaNet@233' #Enter the router password
ssidAP
              = 'ESP32_Wifi'#Enter the AP name
             = '12345678' #Enter the AP password
passwordAP
local_IP
              = '192.168.4.147'
              = '192.168.1.1'
gateway
subnet
              = '255.255.255.0'
              = '8.8.8.8'
dns
sta_if = network.WLAN(network.STA_IF)
ap_if = network.WLAN(network.AP_IF)
def STA_Setup(ssidRouter,passwordRouter):
    print("Setting soft-STA ... ")
    if not sta_if.isconnected():
       print('connecting to',ssidRouter)
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
        while not sta_if.isconnected():
                                                                          (continues on next page)
```

```
pass
   print('Connected, IP address:', sta_if.ifconfig())
   print("Setup End")
def AP_Setup(ssidAP,passwordAP):
    ap_if.ifconfig([local_IP,gateway,subnet,dns])
   print("Setting soft-AP ... ")
   ap_if.config(essid=ssidAP,authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
   ap_if.active(True)
   print('Success, IP address:', ap_if.ifconfig())
   print("Setup End\n")
try:
   AP_Setup(ssidAP,passwordAP)
   STA_Setup(ssidRouter,passwordRouter)
except:
   sta_if.disconnect()
   ap_if.idsconnect()
```

#### **Test Result**

Before running the code, you need to modify ssidRouter, passwordRouter, ssidAP, and passwordAP. After making sure that the code is modified correctly, click O"Run current script" and the "Shell" window will display the following:

```
Shell ×
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
Setting soft-AP ...
Success, IP address: ('192.168.4.147', '192.168.1.1', '255.255.255.0',
'8.8.8.8')
Setup End
Setting soft-STA ...
connecting to ChinaNet-2.4G-0DF0
Connected, IP address: ('192.168.1.147', '255.255.255.0', '192.168.1.1'
, '114.114.114.114')
Setup End
>>>
```

Then you can see the ssid\_A on the ESP32

	Wi–Fi	
Settings	Wi-Fi View help	
Q Search	More settings	>
Log in to	AVAILABLE NETWORKS	
Access Cloud, > AppGallery, and	ESP32_Wifi Connected	Ŷ
more	ChinaNet-2.4G-0DF0	<b></b>
🛜 WLAN ESP32_Wifi >	ChinaNet-Dsvv	6
	Encrynterl	

## 5.3.19 Project 63: Comprehensive Experiment



#### Introduction

We did a lot of experiments, and for each one we needed to re-upload the code, so can we achieve different functions through an experiment? In this experiment, we will use an external button module to achieve different functions.

### **Components Required**





ESP32

pansion

Board\*1

Ex-Keyestudio DIY Purple LED Module\*1



Keyestudio Button Keyestudio

Potentiometer\*1



Rotary

and



Keyestudio Obstacle Avoidance Sensor\*1



ESP32

Board\*1



Keyestudio Line Tracking Sensor\*1

Keyestudio Keyestudio DIY Joystick HC-SR04 Ul-Module\*1 trasonic sensor

Keyestudio

Module\*1



DIY Keyestudio XHT11 Common Cathode Temperature RGB Module \*1 Humidity Sensor \*1



Keyestudio ADXL345 Acceleration Sensor\*1







\*1











4P 3P Dupont Wire\*6 Wire\*3

5P Dupont Wire\*1 Dupont

Wiring Diagram



fritzing

**Test Code** 

```
from machine import ADC, Pin, PWM
import time
import machine
import random
import dht
from ADXL345 import adxl345
scl = Pin(22)
sda = Pin(21)
bus = 0
snsr = adxl345(bus, scl, sda)
pwm_r = PWM(Pin(4))
pwm_g = PWM(Pin(0))
pwm_b = PWM(Pin(2))
```

```
pwm_r.freq(1000)
pwm_g.freq(1000)
pwm_b.freq(1000)
DHT = dht.DHT11(machine.Pin(15))
potentiometer_adc=ADC(Pin(33))
potentiometer_adc.atten(ADC.ATTN_11DB)
potentiometer_adc width(ADC WIDTH_12BIT)
button = Pin(23, Pin.IN)
led = PWM(Pin(5))
led.freq(1000)
tracking = Pin(14, Pin.IN, Pin.PULL_UP)
button_z=Pin(32,Pin.IN,Pin.PULL_UP)
rocker_x=ADC(Pin(35))
rocker_y=ADC(Pin(34))
rocker_x.atten(ADC.ATTN_11DB)
rocker_y.atten(ADC.ATTN_11DB)
rocker_x width(ADC WIDTH_12BIT)
rocker_y.width(ADC.WIDTH_12BIT)
avoid = Pin(27, Pin.IN)
# Set ultrasonic pins
trigger = Pin(13, Pin.OUT)
echo = Pin(12, Pin.IN)
def light(red, green, blue):
    pwm_r.duty(red)
    pwm_g.duty(green)
    pwm_b.duty(blue)
# Ultrasonic ranging, unit: cm
def getDistance(trigger, echo):
    # Generates a 10us square wave
    trigger.value(\emptyset) #A short low level is given beforehand to ensure a clean high.
\rightarrow pulse:
    time.sleep_us(2)
    trigger.value(1)
   time.sleep_us(10) #After pulling high, wait 10 microseconds and immediately set it to.
→low
    trigger.value(♥)
    while echo.value() == 0: #Establish a while loop to detect whether the echo pin_
→value is 0 and record the time at that time
        start = time.ticks_us()
    while echo.value() == 1: #Establish a while loop to check whether the echo pin value.
\rightarrow is 1 and record the time at that time
        end = time.ticks_us()
```

```
(continued from previous page)
    d = (end - start) * 0.0343 / 2 #The travel time of the sound wave x the speed of_
\rightarrow sound (343.2 m/s, 0.0343 cm/microsecond), and the distance back and forth divided by 2
    return d
    kevs = 0
    nums = 
    print(keys % 8)
    def toggle_handle(pin):
        global keys
        keys += 1
        print(keys % 7)
    button.irq(trigger = Pin.IRQ_FALLING, handler = toggle_handle)
def showRGB():
    R = random.randint(0, 1023)
    G = random.randint(0, 1023)
    B = random.randint(0, 1023)
    light(R, G, B)
    time.sleep(0.3)
def showsht11():
    DHT.measure()
    print('temperature:',DHT.temperature(),'C','humidity:',DHT.humidity(),'%')
    time.sleep(1)
def showtracking():
    if tracking value() == 0:
        print("0
                  White") #Press to print the corresponding information.
    else:
        print("1 Black")
    time.sleep(0.1) #delay 0.1s
def showJoystick():
    B_value = button_z.value()
    X_value = rocker_x.read()
    Y_value = rocker_y.read()
    print("button:", end = " ")
    print(B_value, end = " ")
    print("X:", end = " ")
    print(X_value, end = " ")
    print("Y:", end = " ")
    print(Y_value)
    time.sleep(0.1)
def adjustLight():
    pot_value = potentiometer_adc.read()
    print(pot_value)
    led.duty(pot_value)
    time.sleep((0.1)
def showAvoid():
    if avoid.value() == 0:
                                                                             (continues on next page)
```

```
print("There are obstacles")
   else:
       print("All going well")
   time.sleep(0.1)
def showDistance():
   distance = getDistance(trigger, echo)
   print("The distance is {:.2f} cm".format(distance))
   time.sleep(0.1)
def showADXL345():
   x,y,z = snsr.readXYZ()
   print('x:',x,'y:',y,'z:',z,'uint:mg')
   time.sleep((0.1))
while True:
   nums = keys % 8 #number of keystrokes mod 7 to get 0, 1, 2, 3, 4, 5, 6
   if nums == 0: #According to RGB
        showRGB()
    elif nums == 1: #Displays the high and low level of the tracking sensor
        showtracking()
   elif nums == 2: #Display temperature and humidity
        showxht11()
   elif nums == 3: #Displays the rocker value
        showJoystick()
   elif nums == 4: #The potentiometer adjusts the LED
        adjustLight()
   elif nums == 5: #Display obstacle information
        showAvoid()
   elif nums == 6: #Display ultrasonic ranging value
        showDistance()
   elif nums == 7: #Display ADXL345_x/y/z value
        showADXL345()
```

#### **Code Explanation**

Calculate how many times the button is pressed, divide it by 8, and get the remainder which is 0, 1 2, 3, 4, 5, 6 and 7. According to different remainders, construct five unique functions to control the experiment and realize different functions.

We add adx1345 library files in this project.

Following the instructions, we can add or remove sensors/modules in the wiring, and then change the experimental function in the code.

#### **Test Result**

Connect the wires according to the wiring diagram, use the USB to power on, and then click  $\bigcirc$  run the test code. At the beginning, the number of keys is 0, the remainder is 0, and the four lamp beads on the RGB module flash with random colors.

~

¥

```
Shell ×

>>> %Run -c $EDITOR_CONTENT

[83]

b'\xe5'

adx1345 found

83

adx1345 found

0
```

Press the button, the RGB stops flashing, press once, the remainder is the function of the experiment is to track the sensor according to black and white objects read high and low levels, the following information is displayed.

$Shell \times$		
1	Black	^
1	Black	
1	Black	
0	White	
1	Black	
1	Black	
		v

Press the key twice, the time of pressing buttons is 2 and the remainder is 2. Read temperature and humidity values. As shown below;

Shell ×						
temperature:	27 °C	humidity:	47	1	8	~
temperature:	27 °C	humidity:	47	5	8	
temperature:	<b>27</b> ℃	humidity:	55	ş	8	
temperature:	<b>27</b> ℃	humidity:	76	ş	8	
temperature:	28 °C	humidity:	82	ş	8	
temperature:	28 °C	humidity:	86	ş	8	
temperature:	28 °C	humidity:	90	ş	8	
temperature:	29 °C	humidity:	91	ş	8	
temperature:	29 °C	humidity:	92	ş	8	_
temperature:	29 °C	humidity:	92	ş	8	
temperature:	29 °C	humidity:	93	ş	8	
temperature:	29 °C	humidity:	93	ş	8	
townowstuwo	20 00	humiditur	0.2		و	~

Press the key again, the time of pressing buttons is 3 and the remainder is 3. Read digital values at x, y and z axis of the joystick module. As shown below;

$\mathbf{Shell} \times$				
button	: 0 X	: 4095	Y: 1952	~
button	: 0 X	: 4027	Y: 4095	
button	: 0 X	: O Y:	4095	
button	: 0 X	: 0 Y:	1939	
button	: 0 X	: 2130	Y: 0	
button	: 0 X	: 1933	Y: 1934	
button	: 0 X	: 1936	Y: 1939	
button	: 0 X	: 1936	Y: 1936	
button	: 0 X	: 1932	Y: 1937	
button	: 1 X	: 1935	Y: 1941	
button	: 1 X	: 1936	Y: 1938	
button	: 1 X	: 1934	Y: 1936	
button	: 1 X	: 1934	Y: 1940	
				$\checkmark$

Press the key for the fourth time, the remainder is 4. Then the potentiometer can adjust the PWM value at the GPI05 port to control LED brightness of the purple LED;

Shell ×	
U	
256	
773	
1359	
1811	
2283	
2791	
3375	
4095	
4005	

Press the key for the fifth time, the remainder is 5. Then the obstacle avoidance sensor can detect obstacles, as shown below;

Shell ×	
All going well	^
All going well	
All going well	
All going well	
There are obstacles	
	¥

Press the key for the sixth time, the remainder is 6. Then the ultrasonic sensor can detect distance away from obstacles, as shown below;

~

$\mathbf{Shell} \times$				
The distan	ce is	: 6.98	cm	^
The distan	ce is	: 6.12	cm	
The distan	ce is	: 6.98	cm	
The distan	ce is	: 6.98	cm	
The distan	ce is	: 8.09	cm	
The distan	ce is	: 8.56	cm	
The distan	ce is	: 8.97	cm	
The distan	ce is	: 8.16	cm	
The distan	ce is	: 8.51	cm	
The distan	ce is	: 8.97	cm	
The distan	ce is	: 9.36	cm	
				¥

Press the key for seventh time and the remainder is 7. The shell will print out the acceleration value;

```
Shell ×
```

```
x: -639.6 y: 830.7 z: 230.1 uint:mg
x: -686.4 y: 538.2 z: 358.8 uint:mg
x: -542.1 y: 666.9 z: -323.7 uint:mg
x: -276.9 y: 783.9 z: -795.6 uint:mg
x: -163.8 y: 444.6 z: -744.9 uint:mg
x: -62.4 y: 284.7 z: -1099.8 uint:mg
x: 93.60001 y: -93.60001 z: -31.2 uint:mg
x: 214.5 y: -308.1 z: -819.0 uint:mg
x: 140.4 y: -378.3 z: -858.0 uint:mg
x: 226.2 y: -370.5 z: -920.4 uint:mg
x: 226.2 y: -315.9 z: -861.9 uint:mg
```

Press the key for eighth time and the remainder is 0. Then the RGB will flash. If you press keys incessantly, remainders will change in a loop way. So does functions.

Press "Ctrl+C"or click "Stop/Restart backend" to exit the program.

## CHAPTER

SIX

## **ARDUINO TUTORIAL**

## 6.1 1. Get started with Arduino C:

## 6.1.1 1. Windows System



### 1.1 Installing Arduino IDE:

When you get control board, you need to download Arduino IDE and driver firstly.

You could download Arduino IDE from the official website: https://www.arduino.cc/, click the "**SOFTWARE**" on the browse bar, click "**DOWNLOADS**" to enter download page, as shown below:

PROFESSIONAL	EDUCATION	STORE	Q Search on Arduino.cc							SIGN IN
ΘO	HARDWARE	SOFTWARE	CLOUD	DOCUMENTATION -	COMMUNIT	IY 👻	BLOG	ABOUT		
WHAT BUY AN	T IS ARDU		AR ED WH & J Oct Sign	ADUINO DUVISION	се ST (Ге СС) п? т	Time Check o product electror Enter	to play sut the best sits, for all this here fit	st of our ki he fun of ti	ds he	
	HARDWARE	SOFTWARE	Sigi CLOUD	DOCUMENTATION	сомми	Enter	here f	or fun! G ABO	DUT	

# Downloads



There are various versions of IDE for Arduino. Just download a version compatible with your system. Here we will show you how to download and install the windows version of Arduino IDE.


There are two versions of IDE for Windows system: Windows Win7 and newer and Windows ZIP file. The former needs to install manually, while the latter can be directly downloaded, without the need of installing it manually.



You just need to click "JUST DOWNLOAD".

After the Arduino is downloaded, click "I Agree" to continue installing.



Click "Next".

💿 Arduino Setup: Installation	Options	_	
Check the components you don't want to instal	you want to install I. Click Next to con	and uncheck the tinue.	components
Select components to install:	<ul> <li>✓ Install Ard</li> <li>✓ Install USE</li> <li>✓ Create State</li> <li>✓ Create De</li> <li>✓ Associate</li> </ul>	uino software driver art Menu shortcu sktop shortcut .ino files	t
Space required: 543.4MB			
Cancel Nullsoft Inst	all System v3.0	< Back	Next >

Then click "Install".

💿 Arduino Setup: Installation Folder			$\times$
Setup will install Arduino in the following fo folder, dick Browse and select another fold installation.	lder. To insta der. Click Inst	ll in a differe all to start t	ent he
Destination Folder			
C: \Users \Administrator \Desktop \Arduino		Browse.	
Space required: 543.4MB			
Space available: 14.7GB			
Cancel Nullsoft Install System v3.0	< Back	Ins	tall

If the following page appears, click "Install".

Windows Security	٢
Would you like to install this device software?	
Name: Arduino USB Driver Publisher: Arduino LLC	
Always trust software from "Arduino LLC". Install Don't Install	ן
You should only install driver software from publishers you trust. <u>How can I</u> <u>decide which device software is safe to install?</u>	

🤕 Arduino Setu	ıp: Installing	_	
Extract: i	iox128a1u.h		
	1		
Show details			
Cancel	Nullsoft Install System v3.0	< Back	Close
💿 Arduino Setu	ıp: Completed		
Arduino Setu Complete	up: Completed	-	• ×
Arduino Setu     Complete	up: Completed		
Show details	up: Completed		
Show details	ıp: Completed :d		
Show details	ıp: Completed :d		
Show details	ıp: Completed		
Show details	ıp: Completed		
Show details	ıp: Completed		

#### 1.2 Install a driver on Windows

If you have installed the driver, just skip it.

Before using the ESP32 board, you must install a driver, otherwise it will not communicate with computer.

Unlike the USB series chip (ATMEGA8U2) of the Arduino UNO R3, the ESP32 board is used the CP2102 chip USB series chip and USB type C interface.

The driver of the CP2102 chip is included in 1.8.0 version and newer version of Arduino IDE. Usually, you connect the board to the computer and wait for Windows to begin its driver installation process. After a few moments, the process will succeed.

Right click "**Computer**"— Click "**Properties**"—Click "**Device Manager**". Look under Ports (COM & LPT) or other devices, The driver of CP2102 is installed successfully. As shown below:

ᡖ Device Manager	_	$\times$
File Action View Help		
V 🗄 DESKTOP-980K7TG		
> 4 Audio inputs and outputs		
> 💻 Computer		
> 👝 Disk drives		
> 🏣 Display adapters		
> 🐺 Human Interface Devices		
> 📷 IDE ATA/ATAPI controllers		
> 🔤 Keyboards		
> III Mice and other pointing devices		
> 🛄 Monitors		
> 🚽 Network adapters		
V 💭 Ports (COM & LPT)		
Silicon Labs CP210x USB to UART Bridge (COM3)		
> 📇 Print queues		
> Processors		
> F Software components		
> 📱 Software devices		
> 4 Sound, video and game controllers		
> ُ Storage controllers		
> 🏣 System devices		
> 🏺 Universal Serial Bus controllers		

If the driver installation process fail, you need to install the driver manually.

Note:

1). Please make sure that your IDE is updated to 1.8.0 or newer version.

2). If the version of Arduino IDE you download is below 1.8, you should download the driver of CP2102 and install it manually.

Link to download the driver of CP2102 : https://fs.keyestudio.com/CP2102-WIN

To install the drive manually, open the device manager of computer. A yellow exclamation mark means that the CP2102 driver installation failed.

📇 Device Manager		$\times$
File Action View Help		
<ul> <li>DESKTOP-98OK7TG</li> <li>Audio inputs and outputs</li> <li>Computer</li> <li>Disk drives</li> <li>Display adapters</li> <li>Display adapters</li> <li>Human Interface Devices</li> <li>IDE ATA/ATAPI controllers</li> <li>Keyboards</li> <li>Mice and other pointing devices</li> <li>Monitors</li> </ul>		
<ul> <li>Network adapters</li> <li>Other devices</li> <li>CP2102 USB to UART Bridge Controller</li> </ul>		
<ul> <li>Print queues</li> <li>Processors</li> <li>Software components</li> <li>Software devices</li> <li>Sound, video and game controllers</li> <li>Storage controllers</li> <li>System devices</li> <li>Universal Serial Bus controllers</li> </ul>		

Double-click I CP2102 USB to UART Bridge Controller and click " Update drive..."

占 Device	Manager —	$\times$
File Acti	CP2102 USB to UART Bridge Controller Properties ×	
<	General Driver Details Events	
→ 📇 DES	CP2102 USB to UART Bridge Controller	
	Device type: Other devices	
> 🔜 I	Manufacturer: Unknown	
> 🎮	Location: Port_#0012.Hub_#0001	
> 🗃		
	Device status	
> 🙂	The drivers for this device are not installed. (Code 28)	
	There are no compatible drivers for this device	
	There are no compauble drivers for this device.	
Y № 1	To find a driver for this device, click Update Driver	
	To find a driver for this device, click opdate briver.	
	✓	
	Update Unver	
G		
	OK Cancel	
· *		

Click "Browse my computer for drivers" for updated driver software.

$\rightarrow$	Search automatically for drivers Windows will search your computer for the best	available driver and install it on	
	your device.		
$\rightarrow$	Browse my computer for drivers		
	Locate and install a driver manually.		

There is a DRIVERS folder in Arduino software installed package Arduino ), open driver folder and you can see the driver of CP210X series chips.

Click "Browse", then find the driver folder, or you could enter "driver" to search in rectangular box, then click "Next",

÷	Update Drivers - CP2102 USB to UART Bridge Controller	×
	Browse for drivers on your computer	
	Search for drivers in this location:	
	C:\Users\Administrator\Desktop\Arduino\drivers  v Browse 1	
	Include subfolders	
	→ Let me pick from a list of available drivers on my computer This list will show available drivers compatible with the device, and all drivers in the same category as the device.	
	Next Cano	el

Update Drivers - Silicon Labs CP210x USB to UART Bridge (COM3)

Windows has successfully updated your drivers

Windows has finished installing the drivers for this device:



Silicon Labs CP210x USB to UART Bridge



 $\times$ 

Open **device manager**, you will find the yellow exclamation mark disappear. The driver of CP2102 is installed successfully.

🗄 Device Manager	_	$\times$
File Action View Help		
V 🛔 DESKTOP-980K7TG		
> 🐗 Audio inputs and outputs		
> 💻 Computer		
> 👝 Disk drives		
> 🏣 Display adapters		
> 🐺 Human Interface Devices		
> 📹 IDE ATA/ATAPI controllers		
> 🔤 Keyboards		
> III Mice and other pointing devices		
> 💻 Monitors		
> 🚽 Network adapters		
V 💭 Ports (COM & LPT)		
Silicon Labs CP210x USB to UART Bridge (COM3)		
> 🖻 Print queues		
> Processors		
> Foftware components		
> Software devices		
> 4 Sound, video and game controllers		
> 🍇 Storage controllers		
> 🏣 System devices		
> 🏺 Universal Serial Bus controllers		

#### 1.3. Install the ESP32 on Arduino IDE

Note: you need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.



1). Click Arduino to open Arduino IDE



2). Click "File"  $\rightarrow$  "Preferences" copy the website address https://dl.espressif.com/dl/package\_esp32\_index.json in the "Additional Boards Manager URLs:" and click "OK"



# keyestudio WiKi

Preferences	×			
Settings Network				
Sketchbook location:				
C:\Users\Administrator\Documents\Arduino				
Editor language:	System Default $\checkmark$ (requires restart of Arduino)			
Editor font size:	12			
Interface scale:	Automatic 100 * % (requires restart of Arduino)			
Theme:	Default theme $\checkmark$ (requires restart of Arduino)			
Show verbose output during	: compilation upload			
Compiler warnings:	None $\sim$			
Display line numbers	Enable Code Folding			
✓ Verify code after uplo	ad 🗌 Use external editor			
✓ Check for updates on s	artup Save when verifying or uploading			
Use accessibility feat	res			
Additional Boards Manager	URLs: https://dl.espressif.com/dl/package_esp32_index.json 1			
More preferences can be ed	ited directly in the file			
C:\Users\Administrator\App	Data\Local\Arduino15\preferences.txt			
(edit only when Arduino is	not running)			
	OK Cancel			

3). Click "Tools"  $\rightarrow$  "Board:" then click "Boards Manager..." to enter "Boards Manager". Enter "ESP32" as follows, then click "Install".

🐵 Boards Manager	0	×
Type All $\checkmark$ ESP32		
esp32 by Espressif Systems Boards included in this package: ESP32 Dev Module, WEMOS LoLin3 <u>More Info</u>	2, WEMOS D1 MINI ESP32.	~
	Cl	V

🥯 Boards Manager	×
Type All ~ ESP32	
esp32 by Espressif Systems Boards included in this package: ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32. <u>More Info</u>	^
Installing	
	-
	*
Downloading boards definitions. Downloaded 17,637kb of 51,126kb. Canc	el

4). After installing, click "Close".

# 1.4. Arduino IDE Setting:



1). Click Arduino icon to pen Arduino IDE.



2). When downloading the sketch to the board, you must select the correct name of Arduino board that matches the board connected to your computer. As shown below;

(Note: we use the ESP32 board in this tutorial; therefore, we select ESP32\*\*)\*\*

💿 sketch_jan10a   Ai	duino 1.8.16		x i			
File Edit Sketch Too	ls Help					
<pre>sketch_jan10a void setup() // put your } uoid loop() (</pre>	Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries Serial Monitor Serial Plotter WiFi101 / WiFiNINA Firmware Updater	Ctrl+T Ctrl+Shift+I Ctrl+Shift+M Ctrl+Shift+L				△         ESP32 Dev Module         ESP32 Wrover Module         ESP32 Pico Kit         TTGO LoRa32-OLED V1         XinaBox CW02
<pre>// put your }</pre>	Board: "Arduino Uno" Port: "COM3" Get Board Info Programmer: "AVRISP mkll" Burn Bootloader	:	Board Ardui Ardui	s Manager no AVR Board no Mbed OS F Arduino	ls > RP2040 Boards > >	SparkFun ESP32 Thing u-blox NINA-W10 series (ESP32) Widora AIR Electronic SweetPeas - ESP320 Nano32 LOLIN D32 LOLIN D32 PRO WEMOS LOLIN32
1		Arduino Uno	on COM3			Dongsen Tech Pocket 32 "WeMos" WiFi&Bluetooth Battery ESPea32 Noduino Quantum Node32s Hornbill FSP32 Dev



3). Set the board type as follows;



4). Then select the correct COM port (you can see the corresponding COM port after the driver is successfully installed).

🛔 Device Manager	_	$\times$
File Action View Help		
🗢 🔿 п 🛛 🖬 💭		
V 🗄 DESKTOP-980K7TG		
> 4 Audio inputs and outputs		
> 💻 Computer		
> 🚘 Disk drives		
> 🏣 Display adapters		
> 🛺 Human Interface Devices		
> 📹 IDE ATA/ATAPI controllers		
> 🔤 Keyboards		
> III Mice and other pointing devices		
> 🛄 Monitors		
> 🚽 Network adapters		
V 🛱 Ports (COM & LPT)		
Silicon Labs CP210x USB to UART Bridge (COM3)		
> 🖻 Print queues		
Processors		
> Foftware components		
Software devices		
> 🖏 Sound, video and game controllers		
> 🍇 Storage controllers		
> 🏣 System devices		
> 🏺 Universal Serial Bus controllers		





- A- Used to verify whether there is any compiling mistakes or not.
- B- Used to upload the sketch to your Arduino board.
- C- Used to create shortcut window of a new sketch.
- D- Used to directly open an example sketch.
- E- Used to save the sketch.
- F- Used to send the serial data received from board to the serial monitor.

# 6.1.2 2. Mac System:



# 2.1. Download Arduino IDE:

# Downloads



The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the Getting Started page for Installation instructions.

#### SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

#### DOWNLOAD OPTIONS

Windows Win 7 and newer Windows ZIP file

Windows app Win 8.1 or 10 Get 🚦

Linux 32 bits Linux 64 bits Linux ARM 32 bits Linux ARM 64 bits

Mac OS X 10.10 or newer

Release Notes Checksums (sha512)

#### 2.2. How to install the CP2102 driver

If you have installed the driver, just skip it.

1). Connect the ESP32 board to your computer, and open Arduino IDE.



2). Click "Tools-Board:ESP32 Dev Module" and "/dev/cu.usbserial-0001".

🗯 Arduino File Edit Sketch	Tools Help	. 6	۲	<b>N</b>	拼	* (	D ?
• • •	Auto Format						
	Archive Sketch						
	Fix Encoding & Reload						
sketch_jan19a	Manage Libraries						
void setup() {	Serial Monitor						
77 pur your setup coue nere, co ru	Serial Plotter						
ŀ	WiFi101 / WiFiNINA Firmware Updater						
<pre>void loop() {     // put your main code here, to run</pre>	ArduBlock		L				
F	Board: "ESP32 Dev Module"						
	Upload Speed: "921600"						
	CPU Frequency: "240MHz (WiFi/BT)"						
	Flash Mode: "QIO"						
	Flash Size: "4MB (32Mb)"						
	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"						
	Core Debug Level: "None"						
	PSRAM: "Disabled"			_	_	_	_
			/d	ev/cu.Blue	tooth-li	ncomin	g-Port
	Programmer		/4	iev/cu.usbs	erial-0	001	

3). Click to upload code.

Note: If code is uploaded unsuccessfully, you need to install driver of CP2102, please continue to follow the instructions as below:

Download the driver of CP2102:

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Select Mac OSX edition, as shown below;

## Download for WinCE

Platform	Software	Release Notes
🙀 WinCE 6.0 (2.1)	Download VCP (276 KB)	Download WINCE 6.0 Revision History
🏨 WinCE 5.0 (2.1)	Download VCP (271 KB)	Download WinCE 5.0 Revision History

# Download for Macintosh OSX (v5.3.5)

Platform	Software	Release Notes
Kac OSX	Download VCP (832 KB)	Download Mac VCP Revision History

#### **Download for Linux**

Platform	Software	Release Notes
∆ Linux 3.x.x and 4.x.x	Download VCP (10.0 KB)	Download Linux 3.x.x and 4.x.x VCP Revision History
🛕 Linux 2.6.x	Download VCP (10.2 KB)	Download Linux 2.6.x VCP Revision History

\*Note: The Linux 3.x.x and 4.x.x version of the driver is maintained in the current Linux 3.x.x and 4.x.x tree at www.kernel.org.

Unzip the downloaded package.



Open folder and double-click "SiLabsUSBDriverDisk.dmg" file.



You will view the following files as follows:



Double-click "Install CP210x VCP Driver", tick "Don't warn me when opening application on this disk image" and tap "Open".



Click "Continue".



Tap "Agree" and "Continue".

0 • 0	Install CP210x VCP Driver
To install this softwa license agreement. Click Agree to continu	re you must agree to the terms of this software e or Disagree to cancel the installation.
Read License	Disagree Agree TO
• Install	AGREEMENT CONSTITUTES YOUR AND (IF APPLICABLE) YOUR COMPANY'S ASSENT TO AND ACCEPTANCE OF THIS END-USER LICENSE AGREEMENT (THE "LICENSE" OR "AGREEMENT") JE YOU DO NOT AGREE WITH ALL OF THE
Guidance     Rebuild Cache	TERMS, YOU MUST NOT USE THIS PRODUCT. WRITTEN APPROVAL IS NOT A PREREQUISITE TO THE VALIDITY OR ENFORCEABILITY OF
<ul> <li>Summary</li> </ul>	THIS AGREEMENT, AND NO SOLICITATION OF SUCH WRITTEN APPROVAL BY OR ON BEHALF OF SILICON LABORATORIES, INC. ("SILICON LABS") SHALL BE CONSTRUED AS AN INFERENCE TO THE CONTRARY. IF THESE TERMS ARE CONSIDERED AN OFFER BY SILICON LABS, ACCEPTANCE IS EXPRESSLY LIMITED TO THESE TERMS.
(5)	LICENSE AND WARRANTY: The Licensed Product and the embedded Software which is made the subject of this License is either the property of SILICON LABS or a third party from whom
SILICON LABS	Back Continue

Click "Continue" and input your password.

	Install CP210x VCP Driver
	To Be Installed: Version 5.3.5
License	Currently Installed: None
Info	version 5.3.5 will be installed in /Library/Extensions/.
Install	
Guidance	
Rebuild Cache	
Summary	
SILICON LABS	You will be prompted to enter your password. Back Continue
Install ( helper to Enter you Usernar Passwo	CP210x VCP Driver is trying to install a new tool. ur password to allow this. me:
	Cancel Install Helper

Select "Open Security Preferences".

# System Extension Blocked

A program tried to load new system extension(s) signed by "Silicon Laboratories Inc", which will be incompatible with a future version of macOS. If you want to enable these extensions, open Security & Privacy System Preferences.

Open Security Preferences

Click the lock to unlock "security & privacy preference".



Tap Unlock and enter your Username and password

OK



Then click "Allow".

	Security & Privacy	Q Search
General	FileVault Firewall Privacy	
A login password has been set	for this user Change Password	
<ul> <li>✓ Require password 1 h</li> <li>Show a message when</li> <li>✓ Disable automatic login</li> </ul>	our ᅌ after sleep or screen sa the screen is locked Set Lock Mess	aver begins sage
Allow apps downloaded from: App Store App Store and identified Anywhere	d developers	
System software from develop blocked from loading.	er "Silicon Laboratories Inc" was	Allow
lick the lock to prevent further cl	nanges.	Advanced ?

Back to installation page, and wait to install.



Successfully installed.

	Install CP210x VCP Driver
<ul> <li>Introduction</li> <li>License</li> <li>Info</li> <li>Install</li> <li>Guidance</li> <li>Rebuild Cache</li> <li>Summary</li> </ul>	Installation of the driver succeeded.
SILICON LABS	Back Close

Then enter ArduinoIDE, click Tools and select Board ESP32 Dev Module and the serial port

#### is"/dev/cu.SLAB\_USBtoUAPT.



Click to upload code and show "Done uploading".

• • •	sketch_sep14a   Arduino 1.8.13	
		ø
sketch_sep14a	a §	
<pre>void setup() {     // put your s     pinMode(13,0)</pre>	setup code here, to run once: OUTPUT);	
}		
<pre>void loop() {    // put your r    digitalWrited    delay(200);    digitalWrited    delay(200); }</pre>	<pre>main code here, to run repeatedly: (13,HIGH); (13,LOW);</pre>	
Dama un las dina	٠ ٦	
Done uploading.		
Invalid library Invalid library Invalid library Invalid library Invalid library	found in /Users/lisagu/Documents/Arduino/libraries/HT16K3 found in /Users/lisagu/Documents/Arduino/libraries/Danger found in /Users/lisagu/Documents/Arduino/libraries/PS2X: found in /Users/lisagu/Documents/Arduino/libraries/Psx_ar	33: no r_shiel no hea nalog: neNet:

13

Arduino Uno on /dev/cu.SLAB\_USBtoUART

# 6.1.3 3. How to Add Libraries? :

## 3.1. What are Libraries ? :

Libraries are a collection of code that make it easy for you to connect sensors, displays, modules, etc.

For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

Invalid library found in /Users/lisagu/Documents/Arduino/libraries/Self\_balanci

The built-in libraries and some of these additional libraries are listed in the reference. https://www.arduino.cc/en/Reference/Libraries

# 3.2. How to Install a Library ? :

Here we will introduce the most simple way to add libraries .

**Step 1:** After downloading well the Arduino IDE, you can right-click the icon of Arduino IDE. Find the option "**Open file location**".





- □ ×									
File Home Shar	re View			~ ?					
$\leftarrow \rightarrow \checkmark \uparrow$ > This PC > Desktop > Arduino $\checkmark$ O Search Arduino									
A Quick course	Name	Date modified	Туре	Size					
	drivers	10/21/2021 9:20 AM	File folder						
Desktop 🖈	examples	10/21/2021 9:20 AM	File folder						
🕂 Download 🖈	hardware	10/21/2021 9:20 AM	File folder						
🔮 Document 🖈	📊 java	10/21/2021 9:20 AM	File folder						
📰 Pictures 🖈	lib	10/21/2021 9:20 AM	File folder						
OneDrive	libraries	12/20/2021 2:58 PM	File folder						
Onebrive	tools	10/21/2021 9:20 AM	File folder						
💻 This PC	tools-builder	10/21/2021 9:20 AM	File folder						
A Network	💿 arduino.exe	9/3/2021 3:54 PM	Application	72 KB					
	🚮 arduino.l4j	9/3/2021 3:54 PM	Configuration sett	1 KB					
	💿 arduino_debug.exe	9/3/2021 3:54 PM	Application	69 KB					
DESKTOP-7R4	📓 arduino_debug.l4j	9/3/2021 3:54 PM	Configuration sett	1 KB					
DESKTOP-BD	📧 arduino-builder.exe	9/3/2021 3:53 PM	Application	23,156 KB					
DESKTOP-CM	🚳 libusb0.dll	9/3/2021 3:53 PM	Application exten	43 KB					
DESKTOP-D2:	🚳 msvcp100.dll	9/3/2021 3:53 PM	Application exten	412 KB					
	🚳 msvcr100.dll	9/3/2021 3:53 PM	Application exten	753 KB					
	revisions	9/3/2021 3:53 PM	Text Document	96 KB					
	👹 uninstall.exe	10/21/2021 9:20 AM	Application	404 KB					
- IVIW-	📄 wrapper-manifest	9/3/2021 3:54 PM	XML Document	1 KB					
19 items									

📙   💆 📙 🖛   libraries			_	
File Home Shar	e View			~ 🕐
← → × ↑ 📕 → TI	ට 🔎 Searc	h libraries		
^	Name	Date modified	Туре	Size 🔺
📌 Quick access	Adofruit Circuit Playaround	10/21/2021 0-20 AM	Eile felder	
📃 Desktop  🖈	Reides	10/21/2021 9:20 AIVI	File folder	
👆 Download 🖈	Db+11	11/8/2021 11:22 AM	File folder	
🔮 Document 🖈	Esplora	10/21/2021 0.20 AM	File folder	
Pictures 🖈		10/21/2021 9:20 AM	File folder	
	Firmata	10/21/2021 9:20 AM	File folder	
OneDrive	GSM	10/21/2021 9:20 AM	File folder	
This PC	Keyboard	10/21/2021 9:20 AM	File folder	
	Keypad	11/8/2021 11:23 AM	File folder	
💣 Network	LCD 128X32	11/8/2021 10:48 AM	File folder	
DDDD-PC	 LiquidCrystal	10/21/2021 9:20 AM	File folder	
DESKTOP-7R4	LiquidCrystal_I2C	10/21/2021 3:10 PM	File folder	
DESKTOP-BD	MFRC522_I2C	11/8/2021 5:36 PM	File folder	
DESKTOP-CM	Mouse	10/21/2021 9:20 AM	File folder	
DESKTOP-D2	TOP-D2: Robot_Control 10/21/2021 9:20 AM File folde		File folder	
	Robot_Motor	10/21/2021 9:20 AM	File folder	
	RobotlRremote	10/21/2021 9:20 AM	File folder	
	SD SD	10/21/2021 9:20 AM	File folder	~
🔜 MW- 🗸 🤞	C			>
25 items				

Step 3: Next, find out the "libraries" folder.

bard		Organize	New	Open			Select	
ŀ		> 5.Libraries_Driver_Firmware_a	and_APP	୍ <b>ଓ ୪</b>		,⊂ Se	Search 5.Libraries_	
	Name	^		Date mod	dified		Туре	
	Android APP		11/10/2023 2:17 PM			File folder		
	Arduin	o_C_Libraries(Raspberry-Pi)		11/13/202	23 8:42 AM		File folder	
	Arduin	o_C_Libraries(Windows)		11/13/202	23 8:42 AM		File folder	
	CP2102	Driver File		11/10/202	23 1:41 PM		File folder	
	Python	_Firmware		11/10/202	23 1:46 PM		File folder	
Clipboard O	rganize New	Open	1	Select				
-------------------------------------	----------------------------------	------------------	-----------------	------------				
S.Libraries_Driver_Firmware_and_APP	> Arduino_C_Libraries(Windows)	) ~	Q O	Search Ar				
Name	Date mo	dified	Туре					
Adafruit_NeoPixel	11/13/20	23 8:42 AM	File folder					
adx1345_io	11/13/20	23 8:42 AM	File folder					
DS18B20	11/13/20	23 8:42 AM	File folder					
ESP32Servo	11/13/20	23 8:42 AM	File folder					
HT16K33_Lib_For_ESP32	11/13/20	23 8:42 AM	File folder					
IRremoteESP8266	11/13/20	23 8:42 AM	File folder					
lcd128_32_io	11/13/20	23 8:42 AM	File folder					
MFRC522_I2C	11/13/20	23 8:42 AM	File folder					
Rtc_by_Makuna	11/13/20	23 8:42 AM	File folder					
	11/13/20	23 8:42 AM	File folder					
	11/13/20	23 8:42 AM	File folder					
📊 xht11	11/13/20	23 8:42 AM	File folder					
_	-							
Adafruit_NeoPixel								
adx1345_io								
DS18B20								
ESP32Servo								
HT16K33_Lib_For_ESP32								
IRremoteESP8266								
lcd128_32_io								
MFRC522_I2C								
📊 Rtc_by_Makuna								
TM1650								
Wire								
xht11								
Copy destination"	in the libraries folder of Ardui	no. Then click "	Replace the fil	les in the				



libraries

Share View

📙 > This PC > Desktop > Arduino > lib	raries		~
Name	Date modified	Туре	Size
Adafruit_Circuit_Playground	10/21/2021 9:20 AM	File folder	
Adafruit_NeoPixel	4/2/2022 3:23 PM	File folder	
adxI345_io	3/30/2022 3:28 PM	File folder	
Bridge	10/21/2021 9:20 AM	File folder	
Dht11	11/8/2021 11:22 AM	File folder	
DS18B20	3/30/2022 2:09 PM	File folder	
ESP32Servo	3/30/2022 1:29 PM	File folder	
Esplora	10/21/2021 9:20 AM	File folder	
Ethernet	10/21/2021 9:20 AM	File folder	
examples	7/26/2021 11:46 AM	File folder	
📙 Firmata	10/21/2021 9:20 AM	File folder	
GSM	10/21/2021 9:20 AM	File folder	
HT16K33_Lib_For_ESP32	4/2/2022 3:23 PM	File folder	
RremoteESP8266	3/9/2022 3:06 PM	File folder	
LCD 128X32	2/28/2022 6:51 PM	File folder	
	4/2/2022 3:23 PM	File folder	
LiquidCrystal	10/21/2021 9:20 AM	File folder	
Matrix	2/16/2022 6:04 PM	File folder	
MFRC522_I2C	3/10/2022 1:22 PM	File folder	
Mouse	10/21/2021 9:20 AM	File folder	
Robot_Motor	10/21/2021 9:20 AM	File folder	
RobotlRremote	10/21/2021 9:20 AM	File folder	
Rtc_by_Makuna	3/30/2022 2:33 PM	File folder	
SD SD	10/21/2021 9:20 AM	File folder	
SpacebrewYun	10/21/2021 9:20 AM	File folder	
src	3/28/2022 2:47 PM	File folder	
Temboo	10/21/2021 9:20 AM	File folder	
TFT	10/21/2021 9:20 AM	File folder	
	4/2/2022 10:16 AM	File folder	
UltrasonicSensor-1.1.0	1/11/2022 11:05 AM	File folder	
WiFi	10/21/2021 9:20 AM	File folder	
Wire	3/10/2022 1:22 PM	File folder	
📙 xht11	3/28/2022 1:17 PM	File folder	

# 6.2 2. Basic Projects

When we get the kit, we can see that there are 42 sensors/modules in the kit, which contain the corresponding ESP32 mainboard, ESP32 Expansion Board and wirings. Here, we will connect the 42 sensors individually to the ESP32 mainboard and the ESP32 Expansion Board using wirings. Then run the corresponding test code to test the function of each sensor separately. Our next lesson is to study the principles of individual modules/sensors from simple to complex as well as some extended applications of sensors to consolidate and deepen our understanding of the kits.

Note : When connecting the module/sensor wirings in the projects, the wiring method and position must be followed in the document. What's more, do not misconnect the power supply and signal pin, otherwise there may be no experimental results or damage to the modules/sensors.

## 6.2.1 Project 1: Hello World

## Overview

For ESP32 beginners, we will start with some simple things. In this project, you only need a ESP32 mainboard, a USB cable and a computer to complete the "Hello World!" project, which is a test of communication between the ESP32 mainboard and the computer as well as a primary project.

### Components



## Wiring Diagram

In this project, we will use a USB cable to connect the ESP32 to a computer.



### **Test Code**

(continues on next page)

(continued from previous page)

```
Serial.begin(9600);// sets baudrate to 9600
}
void loop()
{
    if (Serial.available() > 0) {
        val=Serial.read();// reads symbols assigns to "val"
        if(val=='R')// checks input for the letter "R"
        { // if so,
        Serial.println("Hello World!");// shows "Hello World !".
      }
    }
}
```

💿 HelloWorld | Arduino 1.8.16  $\times$ File Edit Sketch Tools Help Auto Format Ctrl+T Ð Ø Archive Sketch -HelloWorld Fix Encoding & Reload ESP32 Dev Module //\*\*\*\*\*\*\* Ctrl+Shift+1 Manage Libraries... • ESP32 Wrover Module Serial Monitor Ctrl+Shift+M ESP32 Pico Kit \* Filename Serial Plotter Ctrl+Shift+L \* Descriptio lays"Hello World". TTGO LoRa32-OLED V1 \* Auther XinaBox CW02 WiFi101 / WiFiNINA Firmware Updater SparkFun ESP32 Thing char val;// d Board: "ESP32 Wrover Module" void setup() Boards Manager... u-blox NINA-W10 series (ESP32) Upload Speed: "921600' Arduino AVR Boards Widora AIR Serial.begin( Flash Frequency: "80MHz" Arduino Mbed OS RP2040 Boards Electronic SweetPeas - ESP320 ŀ void loop() Flash Mode: "QIO" ESP32 Arduino Nano32 ł ESP8266 Boards (2.5.0) Partition Scheme: "Default" LOLIN D32 if (Serial. Core Debug Level: "None" val=Seria LOLIN D32 PRO if(val== Port: "COM3" WEMOS LOLIN32 { // if Get Board Info Dongsen Tech Pocket 32 Serial.p } "WeMos" WiFi&Bluetooth Battery Programmer ł ESPea32 Burn Bootloader < > Noduino Quantum Done Saving. Node32s Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressi: Hornbill ESP32 Dev Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressi Hornbill ESP32 Minima FireBeetle-ESP32 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3 IntoRobot Fig

Before uploading the test code to the ESP32click "Tools"  $\rightarrow$  "Board" select "ESP32 Wrover Module".

Select the correct serial port.

💿 HelloWorld   Ardu	ino 1.8.16		_		×	
File Edit Sketch Too	ls Help					
	Auto Format	Ctrl+T			Ø	
	Archive Sketch					
HelloWorld	Fix Encoding & Reload					
//********	Manage Libraries	Ctrl+Shift+I	*****	*****	**** 🔨	
/*	Serial Monitor	Ctrl+Shift+M				
* Descriptio	Serial Plotter	Ctrl+Shift+L	lays"Hello	World"		
* Auther						
*/ char val:// d	WIFITUT / WIFININA Firmware Upd	ater				
<pre>void setup()</pre>	Board: "ESP32 Wrover Module"	>				
{	Upload Speed: "921600"	>				
Serial.begin(	Flash Frequency: "80MHz"	>				
void loop()	Flash Mode: "QIO"	>				
{ 	Partition Scheme: "Default"	>				
val=Seria	Core Debug Level: "None"	>				
if(val=='	Port: "COM3"	;	Serial por	ts		
{ // if	Get Board Info		COM21			
}		[	COM3			
}	Programmer				~	
<	Burn Bootloader				>	
Done Saving.						
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif						
invalid library f	ound in C:\Users\Administra	tor Desktop Ardu:	ino\nardware	vespre		
<					>	
2	ESP32 Wrover	Module, Default, QIO, 80	0MHz, 921600, N	one on C	юмз	

Note: For macOS users, if the upload fails, set the baud rate to 115200 before clicking

💿 HelloWorld   Ardu	ino 1.8.16				_		×	<
File Edit Sketch Tool	s Help		_					
	Auto Format	Ctrl+T					Ø	
	Archive Sketch							
HelloWorld	Fix Encoding & Reload							
//*******	Manage Libraries	Ctrl+Shift+I		****	*****	*****	****	• •
/* * Filename	Serial Monitor	Ctrl+Shift+M						
* Descriptio	Serial Plotter	Ctrl+Shift+L		lays	"Hello	World	".	
* Auther								
char val;// d	WIFITUT / WIFININA FIRMWa	re Opdater	_					
void setup()	Board: "ESP32 Wrover Modu	le"	>			-		
{	Upload Speed: "921600"		2	•	921600			
<pre>Serial.begin()</pre>	Flash Frequency: "80MHz"		2		115200			
void loop()	Flash Mode: "QIO"		1		256000			
{	Partition Scheme: "Default"		;	1	230400			
val=Seria	Core Debug Level: "None"		;	1	512000			
if(val=='	Port: "COM3"		7					
{ // if Serial.p	Get Board Info			-				
}	Programmer		>					
<	Burn Bootloader						>	Ť
			_					
Done Saving.								
Invalid library fo	ound in C:\Users\Admini ound in C:\Users\Admini	lstrator\Desktop\Ard lstrator\Desktop\Ard	ui ui	.no\h .no\h	ardwar	e\espr e\espr	essi essi	^
		······						~
<							>	_
2	ESP32 V	Vrover Module, Default, QIO,	80	MHz,	921600, 1	None on	сомз	
Click to upload the test code to the ESP32.								

🕺 HelloWorld   Arduino 1.8.16			×
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp			
			ø
HelloWorld			
//*************************************	******	******	**** 🔺
/*			
* Filename : Hello World			
* Description : Enter the letter R, and the serial port displa	ays"Hello	World"	-
* Auther :http// <u>www.keyestudio.com</u>			
*/			
char val;// defines variable "val"			
void setup()			
1 Serial begin(9600):// sets baudrate to 9600			
l			
void loop()			
{			
<pre>if (Serial.available() &gt; 0) {</pre>			
<pre>val=Serial.read();// reads symbols assigns to "val"</pre>			
<pre>if(val=='R')// checks input for the letter "R"</pre>			
{ // if so,			
<pre>Serial.println("Hello World!");// shows "Hello World !".</pre>			
}			~
<			>
Compiling sketch			
2 ESP32 Wrover Module Default OIO 80M	H <del>7</del> 921600	None on C	:омз
	12, 82 1000,	None on c	

Note: If the uploading code fails, you can press and hold the Boot button on the ESP32 after clicking and release the Boot button after the percentage of uploading progress appears., as shown below:





### The code is uploaded successfully.

📨 HelloWorld   Arduino 1.8.16	_		×
<u>Eile Edit Sketch Tools H</u> elp			
			ø
HelloWorld			
//*************************************	*****	******	**** 🔺
/*			
* Filename : Hello World			
* Description : Enter the letter R, and the serial port displays	"Hello	World"	
* Auther :http// <u>www.keyestudio.com</u>			
*/			
<pre>char val;// defines variable "val"</pre>			
<pre>void setup()</pre>			
{			
<pre>Serial.begin(9600);// sets baudrate to 9600</pre>			
}			
void loop()			
{			
<pre>if (Serial.available() &gt; 0) {</pre>			
<pre>val=Serial.read();// reads symbols assigns to "val"</pre>			
<pre>if(val=='R')// checks input for the letter "R"</pre>			
{ // if so,			
<pre>Serial.println("Hello World!");// shows "Hello World !".</pre>			
}	_		× *
2			>
Done uploading.			
Invalid library found in C:\Users\Administrator\Desktop\Arduino\k	ardwar	e\espre	essif 🔨
Invalid library found in C:\Users\Administrator\Desktop\Arduino\B	ardwar	e\espre	essif
			$\sim$
<			>
2 ESP32 Wrover Module, Default, QIO, 80MHz,	921600, 1	None on C	омз

## **Test Result**

After uploading successfullywe will use a USB cable to power on, click set the baud rate to 9600enter the letter "R"click "Send"then the serial monitor prints "Hello World!".

💿 сомз		_	
R 2			Send
Hello World!			3
		10	v
🗹 Autoscroll 🗌 Show timestamp	Newline $\vee$	9600 baud $\sim$	Clear output

## 6.2.2 Project 2: Lighting up LED



### Overview

In this kit, we have a Keyestudio Purple Module, which is very simple to control. If you want to light up the LED, you just need to make a certain voltage across it.

In the project, we will control the high and low level of the signal end S through programming, so as to control the LED on and off.

### **Working Principle**

The two circuit diagrams are given.

The left one is wrong wiring-up diagram. Why? Theoretically, when the S terminal outputs high levels, the LED will receive the voltage and light up.

Due to limitation of IO ports of ESP32 board, weak current can't make LED brighten.

The right one is correct wiring-up diagram. GND and VCC are powered up. When the S terminal is a high level, the triode Q1 will be connected and LED will light up(note: current passes through LED and R3 to reach GND by VCC not IO ports). Conversely, when the S terminal is a low level, the triode Q1 will be disconnected and LED will go off.



Components

	CSP-32 Krywertolis			<b>\$</b>
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Purple LED Module*1	3P Dupont Wire*1	Micro USB Cable*1

Wiring Diagram



fritzing

**Test Code** 



## **Code Explanation**

1). PinMode(pin,mode): Pin is the ESP32 GPIO pin number used to set the mode, here we set pin 0 as output mode.

2). **DigitalWrite(pin, value**): Pin is the GPIO pin, which is defined GP0 here. Valueis the digital level that we will outputHIGH/LOW. If the pin is configured to OUTPUT using pinMode(), its voltage is set to the corresponding value: 3.3V is HIGH, low level is 0V (ground). When connect the LEDs to the pins, using the digitalWriteHIGH, the LEDs will get dim.

3). Setup() executes once, while loop() executes all the time. Delay (ms) is delay function, ms is the number of milliseconds to pause. Data type: unsigned longrange  $0 \sim 4,294,967,295$  (2^32 - 1).

4). Firstly, we connect the module signal to ledPIN, namely GP0, and set it to a high level to light the LEDs on the module. Then delay 1000 ms, controlling the LEDs on the module light up for 1s and off for 1s to achieve the flashing effect.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the LED in the circuit will flash alternately.

Note: If the uploading code fails, you can press and hold the Boot button on the ESP32 after clicking and release the Boot button after the percentage of uploading progress appears.



as shown below:



## 6.2.3 Project 3: Traffic Lights Module



### Overview

In this lesson, we will learn how to control multiple LED lights and simulate the operation of traffic lights.

Traffic lights are signal devices positioned at road intersections, pedestrian crossings, and other locations to control flows of traffic.

In this kit, we will use the traffic light module to simulate the traffic light.

#### **Working Principle**

In previous lesson, we already know how to control an LED. In this part, we only need to control three separated LEDs. Input high levels to the signal R(3.3V), then the red LED will be on.



Components

	CLA-22 Heynordia	View of the second seco		
ESP32	ESP32 Expansion	DIY Traffic Lights	5P Dupont	Micro USB
Board*1	Board*1	Module*1	Wire*1	Cable*1

Wiring Diagram



fritzing

#### **Test Code**

```
/*
* Filename : Traffic_Light
* Description : Simulated traffic lights
* Auther
          : http://www.keyestudio.com
*/
int redPin = 15; //Red LED connected to GPI015
int yellowPin = 2; //Yellow LED connected to GPI02
int greenPin = 0; //Green LED connected to GPIO0
void setup() {
 //LED interfaces are set to output mode
 pinMode(greenPin, OUTPUT);
 pinMode(yellowPin, OUTPUT);
 pinMode(redPin, OUTPUT);
}
void loop() {
 digitalWrite(greenPin, HIGH); //Lighting green LED
 delay(5000); //Delay for 5 seconds
 digitalWrite(greenPin, LOW); //Turn off green LEDS
 for (int i = 1; i <= 3; i = i + 1) { //run three times</pre>
   digitalWrite(yellowPin, HIGH); //Lighting yellow LED
   delay(500); //Delay for 0.5 seconds
   digitalWrite(yellowPin, LOW); //Turn off yellow LED
   delay(500); //Delay for 0.5 seconds
 }
 digitalWrite(redPin, HIGH); //Lighting red LED
 delay(5000); //Delay5s
 digitalWrite(redPin, LOW); //Turn off red LED
}
```

**Code Explanation** 

Create pins, set pins mode and delayed functions.

We use the function for(). for (int i = 1;  $i \le 3$ ; i = i + 1) represents the variable i adds 1 fir each time from 1 to 3.

The function for (int i = 255; i >= 0; i = i - 1) indicates that i reduces by 1 each time. When i<0, exit the for() loop and execute 256 times

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the green LED will be on for 5s then off, the yellow LED will flash for 3s then go off and the red one will be on for 5s then off, the three LED modules will simulate the circulation of traffic lights automatically.

## 6.2.4 Project 4: Laser Sensor



#### Description

Lasers are widely used to cut, weld, surface treat, and more on specific materials. The energy of the laser is very high. The toy laser pointer may cause glare to the human eye, and it may cause retinal damage for a long time. my country also prohibits the use of laser to illuminate the aircraft.

### **Working Principle**

The laser head sensor module is mainly composed of a laser head with a light-emitting die, a condenser lens, and a copper adjustable sleeve. We can see the circuit schematic diagram of this module which is very similar to the LED we have learned. They are all driven by triodes. A high-level digital signal is directly input at the signal end, then the sensor will start to work; if inputting low levels, the sensor won't work.



Components

ESP32	ESP32 Expansion	DIY Laser	3P Dupont	Micro USB
Board*1	Board*1	Module*1	Wire*1	Cable*1

**Connection Diagram** 



fritzing

**Test Code** 

/\* \* Filename : Laser sensor \* Description : Laser light flashing \* Auther : http://www.keyestudio.com \*/ int laserPin = 0; //Define the laser pin as GPIO 0 void setup() { pinMode(laserPin, OUTPUT);//Define laser pin as output mode } void loop() { digitalWrite(laserPin, HIGH); //Open the laser delay(2000); //Delay 2 seconds digitalWrite(laserPin, LOW); //Shut down the laser delay(2000); //Delay 2 seconds }

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the laser module will emit red laser signals for 2 seconds and stop emitting signals for 2 seconds on a cycle.

## 6.2.5 Project 5: Breathing LED



### Overview

A"breathing LED" is a phenomenon where an LED's brightness smoothly changes from dark to bright and back to dark, continuing to do so and giving the illusion of an LED"breathing. This phenomenon is similar to a lung breathing in and out. So how to control LED's brightness? We need to take advantage of PWM. You may refer to Project 6.

### Components

ESP32	ESP32 Expansion	Purple LED	3P Dupont	MicroUSB
Board*1	Board*1	Module*1	Wire*1	Cable*1

### **Connection Diagram**



fritzing

Test Code

```
/*
* Filename
           : Breathing Led
* Description : Make led light fade in and out, just like breathing.
* Auther
        : http//www.keyestudio.com
*/
#define PIN_LED
              0
                //define the led pin
#define CHN
              0 //define the pwm channel
#define FRO
              1000 //define the pwm frequency
#define PWM_BIT
              8
                   //define the pwm precision
void setup() {
 ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel
 ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel
}
void loop() {
 for (int i = 0; i < 255; i++) { //make light fade in</pre>
   ledcWrite(CHN, i);
   delay(10);
 }
 for (int i = 255; i > -1; i--) { //make light fade out
   ledcWrite(CHN, i);
   delay(10);
 }
}
```

## Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the LED on the module gradually gets dimmer then brighter, cyclically, like human breathe.

## 6.2.6 Project 6: RGB Module



### Overview

Among these modules is a RGB module. It adopts a F10-full color RGB foggy common cathode LED. We connect the RGB module to the PWM port of MCU and the other pin to GND(for common anode RGB, the rest pin will be connected to VCC). So what is PWM?

PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output.

In general, the input voltages of ports are 0V and 5V. What if the 3V is required? Or a switch among 1V, 3V and 3.5V? We cannot change resistors constantly. For this reason, we resort to PWM.

For Arduino digital port voltage outputs, there are only LOW and HIGH levels, which correspond to the voltage outputs of 0V and 5V respectively. You can define LOW as"0" and HIGH as"1', and let the Arduino output five hundred '0' or "1" within 1 second. If output five hundred '1', that is 5V; if all of which is '0', that is 0V; if output 250 01 pattern, that is 2.5V.

This process can be likened to showing a movie. The movie we watch are not completely continuous. Actually, it generates 25 pictures per second, which cannot be told by human eyes. Therefore, we mistake it as a continuous process. PWM works in the same way. To output different voltages, we need to control the ratio of 0 and 1. The more '0' or '1' output per unit time, the more accurate the control.

## **Working Principle**

For our experiment, we will control the RGB module to display different colors through three PWM values.



## Components

	CLA-22 Keyenida		~	2
ESP32 Board*1	ESP32 Expansion Board*1	Common Cathode RGB Module *1	4P Dupont Wire*1	Micro USB Cable*1

### **Connection Diagram**



fritzing

**Test Code** 

(continues on next page)

(continued from previous page)

```
* Auther
             : http//www.keyestudio.com
*/
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
                               //define the pwm channels
const byte chns[] = \{0, 1, 2\};
int red, green, blue;
void setup() {
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit</pre>
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
 red = random(0, 256);
 green = random(\emptyset, 256);
 blue = random(\emptyset, 256);
 setColor(red, green, blue);
 delay(200);
}
void setColor(byte r, byte g, byte b) {
 ledcWrite(chns[0], 255 - r); //Common anode LED, low level to turn on the led.
 ledcWrite(chns[1], 255 - g);
 ledcWrite(chns[2], 255 - b);
}
 11
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power onwe will see that the RGB LED on the module starts to display random colors.

## 6.2.7 Project 7: Button Sensor



### Overview

In this kit, there is a Keyestudio single-channel button module, which mainly uses a tact switch and comes with a yellow button cap.

In previous lessons, we learned how to make the pins of our single-chip microcomputer output a high level or low level. In this experiment, we will read the high level (3.3V) and low level (0V).

We can determine whether the button on the sensor is pressed by reading the high and low level of the S terminal on the sensor.

### Working Principle

The button module has four pins. The pin 1 is connected to the pin 3 and the pin 2 is linked with the pin 4. When the button is not pressed, they are disconnected. Yet, when the button is pressed, they are connected. If the button is released, the signal end is high level.



Components

ESP32	ESP32 Expansion	DIY Button	3P Dupont	Micro USB
Board*1	Board*1	Module*1	Wire*1	Cable*1

## **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename
            : button
* Description : Read key value
* Auther
            : http://www.keyestudio.com
*/
int val = 0; //Useto store key values
int button = 15; //The pin of the button is connected to GP15
void setup() {
 Serial.begin(9600); //Start the serial port monitor and set baud rate to 9600
 pinMode(button, INPUT); //Set key pin to input mode
}
void loop() {
 val = digitalRead(button); //Read the value of the key and assign it to the variable.
→val
 Serial.print(val); //Print it on the serial port
 if (val == 0) { //Press the key to read the low level and print the press related.
→information
   Serial.print("
                     "):
   Serial.println("Press the botton");
   delay(100);
 }
 else { //Print information about key release
   Serial.print("
                     ");
   Serial.println("Loosen the botton");
   delay(100);
 }
}
```

## **Code Explanation**

1). pinMode(button, INPUT); set the pin of the button module to GP15 and INPUT.

Configure INPUT through pinMode(). INPUT must use the pull-up or pull-down resistor(ours module has the pull-up resistor RI).

2). Serial.begin(9600): Initialize serial communication and set the baud rate to 9600.

3). **digitalRead(button)**: read the digital level of the button(HIGH or LOW). If this pin is not connected to pins, the digitalRead() will return HIGH or LOW.

4). if...else...if the logic behind () is true, execute the code of (); otherwise execute the code of else.

5). If the button is pressed, the signal end is low level, GP15 is low level and Val is 0. Then the monitor will show the corresponding value and characters; otherwise, the sensor is released, val is 1 and monitor will show 1 and other characters

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. The serial monitor will display the corresponding data and characters. When the button is pressed, val is 0, the monitor

	COM3 —		$\times$
			Send
1	Loosen the potton		A
1	Loosen the botton		
1	Loosen the botton		
1	Loosen the botton		
1	Loosen the botton		
1	Loosen the botton		
1	Loosen the botton		
1	Loosen the botton		
1	Loosen the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0	Press the botton		
0			¥
	Autoscroll Show timestamp Newline Vervice 9600 baud V	Clean	r output

will show "Press the button" when the button is released, val is 1 the monitor will show "Loosen the button"; as shown below:

## 6.2.8 Project 8: Capacitive Sensor



## Description

In this kit, there is a capacitive touch module which mainly uses a TTP223-BA6 chip. It is a touch detection chip, which provides a touch button, and its function is to replace the traditional button with a variable area button. When we power on, the sensor needs about 0.5 seconds to stabilize.

Do not touch the keys during this time period. At this time, all functions are disabled, and self-calibration is always performed. The calibration period is about 4 seconds. We display the test results in the shell.

### **Working Principle**

When our fingers touch the module, the signal S outputs high levels, the red LED on the module flashes. We can determine if the button is pressed or not by reading high and low levels on the sensor.



**Required Components** 

	CD-30 			
ESP32	ESP32 Expansion	DIY Capacitive	3P Dupont	Micro USB
Board*1	Board*1	Module*1	Wire*1	Cable*1

## **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename : Touch sensor
* Description : Reading touch value
* Auther : http://www.keyestudio.com
*/
int val = 0;
int touch = 15; //The key of PIN
void setup() {
 Serial.begin(9600);//Baud rate is 9600
 pinMode(touch, INPUT);//Setting input mode
}
void loop() {
 val = digitalRead(touch);//Read the value of the key
 Serial.print(val);//Print out key values
 if (val == 1) {//Press for high level
               ");
  Serial.print("
  Serial.println("Press the button");
  delay(100);
 }
 else {//Release to low level
  Serial.print("
               ");
   Serial.println("Loosen the button");
  delay(100);
 }
}
```

## **Code Explanation**

When we touch the sensor, the Shell monitor will show "Pressed the button!", if not, "Loosen the button!" will be shown on the monitor.

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600.

The serial monitor will display the corresponding data and characters. when the button is pressed, the red LED lights up and val is 1. Then the shell shows "Pressed the button!"; if the button is released, the red LED is off and val is 0, "Loosen the button!" will be displayed.

(	∞ COM3 —		$\times$
			Send
0	Loosen the button		^
0	Loosen the button		
0	Loosen the button		
0	Loosen the button		
0	Loosen the button		
0	Loosen the button		
1	Press the button		
1	Press the button		
1	Press the button		
1	Press the button		
1	Press the button		
1	Press the button		
1	Press the button		
1	Press the button		
1	Press the button		
			¥
6	🗸 Autoscroll 🗌 Show timestamp 🛛 Newline 🗸 9600 baud 🗸	Clear	output

## 6.2.9 Project 9: Obstacle Avoidance Sensor



### Overview

In this kit, there is a Keyestudio obstacle avoidance sensor, which mainly uses an infrared emitting and a receiving tube.

In the experiment, we will determine whether there is an obstacle by reading the high and low level of the S terminal on the sensor.

### Working Principle

NE555 circuit provides IR signals with frequency to the emitter TX, then the IR signals will fade with the increase of transmission distance. If encountering the obstacle, it will be reflected back.

When the receiver RX meets the weak signals reflected back, the receiving pin will output high levels, which indicates the obstacle is far away. On the contrary, it the reflected signals are stronger, low levels will be output, which represents the obstacle is close. There are two potentiometers on the module, and by adjusting the two potentiometers, we can adjust its effective distance.



Components

				<b>30</b>
ESP32 Board*1	ESP32 Expansion Board*1	DIY Obstacle Avoidance Sensor*1	3P Dupont Wire*1	Micro USB Cable*1

**Connection Diagram** 



fritzing

Test Code

```
/*
* Filename : obstacle avoidance sensor
* Description : Reading the obstacle avoidance value
        : http://www.keyestudio.com
* Auther
*/
int val = 0;
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(15, INPUT);//Set pin GP15 to input mode
}
void loop() {
 val = digitalRead(15);//Read digital level
 Serial.print(val);//Print the level signal read
 if (val == 0) {//Obstruction detected
   Serial.print("
                    ");
   Serial.println("There are obstacles");
   delay(100);
 }
 else {//No obstructions detected
   Serial.print("
                    ");
   Serial.println("All going well");
   delay(100);
 }
}
```

### **Code Explanation**

### Note:

Upload the test code and wire up according to the connection diagram. After powering on, we start to adjust the two potentiometers to sense distance.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600.

The serial monitor will display the corresponding data and characters. When the sensor detects the obstacle, the val is 0, the monitor will show "There are obstacles"; if the obstacle is not detected, the val is 1, "All going well" will be shown.

💿 сом	3	—		$\times$
				Send
1	All going well			^
1	All going well			
1	All going well			
1	All going well			
1	All going well			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
0	There are obstacles			
				~
Autos	roll Show timestamp Newline ~ 9600 baud	~	Clear	output

## 6.2.10 Project 10: Line Tracking Sensor



### Description

In this kit, there is a DIY electronic building block single-channel line tracking sensor which mainly uses a TCRT5000 reflective black and white line recognition sensor element.

In the experiment, we judge the color (black and white) of the object detected by the sensor by reading the high and low levels of the S terminal on the module; and display the test results on the shell.

### Working Principle

When a black or no object is detected, the signal terminal will output high levels; when white object is detected, the signal terminal is low level; its detection height is 0-3cm. We can adjust the sensitivity by rotating the potentiometer on the sensor. When the potentiometer is rotated, the sensitivity is best when the red LED on the sensor is at the critical point between off and on.



**Required Components** 

		Line Tresking		
ESP32 Board*1	ESP32 Expansion	DIY Line Tracking	3P Dupont	MicroUSB
	Board*1	Sensor*1	Wire*1	Cable*1

### **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename : line tracking
* Description : Reading the tracking sensor value
* Auther
           : http://www.keyestudio.com
*/
int val = 0;
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(15, INPUT);//Sets sensor pin to input mode
}
void loop() {
 val = digitalRead(15);//Read the digital level output by the patrol sensor
 Serial.print(val);//Serial port print value
 if (val == 0) {//White val is 0 detected
   Serial.print("
                     ");
```

(continues on next page)
(continued from previous page)

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600.

The serial monitor will display the corresponding data and characters. when the sensor doesn't detect an object or detects a black object, the val is 1, and the monitor will display "1 Black"; when a white object (can reflect light) is detected, the val is 0, and the monitor will display "0 White".

	COM3				_		$\times$
							Send
1	Black						^
1	Black						
1	Black						
1	Black						
1	Black						
1	Black						
0	White						
0	White						
0	White						
0	White						
0	White						
0	White						
0	White						_
0	White						
0	White						
							¥
	Autoscroll 🗌 Show timestamp	Ne	wline $\vee$	9600 baud	$\sim$	Clear	output

## 6.2.11 Project 11: Photo Interrupter



### Description

This kit contains a photo interrupter which mainly uses 1 ITR-9608 photoelectric switch. It is a photoelectric switch optical switch sensor.

### **Working Principle**

When the paper is put in the slot, C is connected with VCC and the signal end S of the sensor are high levels; then the red LED will be off. Otherwise, the red LED will be on.



### **Required Components**

	CH-22 Rependence Concesso Rependence Concesso			<b>\$0</b>
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY Photo Interrupter*1	3P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

#### **Test Code**

```
/*
* Filename : Photo_Interrupt
* Description : Light snap sensor counting
          : http://www.keyestudio.com
* Auther
*/
int PushCounter = 0; //The count variable is assigned an initial value of 0
int State = 0; //Store the current state of the sensor output
int lastState = 0; //Stores the state of the last sensor output
void setup() {
 Serial.begin(9600);//Set the baud rate to 9600
 pinMode(15, INPUT);//Set the light snap sensor pin to input mode
}
void loop() {
 State = digitalRead(15);//Read current state
 if (State != lastState) {//If the state is different from the last read
   if (State == 1) {//block the light
     PushCounter = PushCounter + 1;//Count + 1
                                                                  (continues on next page)
```

(continued from previous page)

## **Code Explanation**

Logic setting:

	Set PushCounter to 0			
Initial Setting	Set State to 0 (value of the s	ensor)		
	Set lastState to 0			
when an object	lastState is 0 , State turns	Set PushCounter to PushCounter+1		
enters the slot	into 1; lastState turns into 1	print the value of PushCounter		
when the object	lastState is 1 , State becomes 0 , two data are	PushCounterdoesn't change;		
leaves the slot	not equal , lastState turns into 0.	Don't print the value of PushCounter		
When the object goes through this slot again	lastState is 0, State becomes 1 , two data are not equal , lastState turns into 1.	SetPushCounter to PushCounter+1. And print the value of PushCounter		
When the object leaves this slot again	lastState is 1 , State turns into 0 , two data are not equal lastState turns into 0	PushCounter doesn 't change;Don 't print the PushCounter value		

## Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600.

The serial monitor will display the PushCounter data. Every time when the object passes through the slot of the sensor, the PushCounter data will increase by 1 continuously, as shown below;

💿 СОМЗ			_		$\times$
					Send
1					^
2					
4					
5					
6					
7					
8					
10					
11					
12					
13					
15					
16					~
Autoscroll 🗌 Show timestamp	Newline $\checkmark$	9600 baud	$\sim$	Clea	r output

## 6.2.12 Project 12: Tilt Module



#### Overview

In this kit, there is a Keyestudio tilt sensor. The tilt switch can output signals of different levels according to whether the module is tilted. There is a ball inside. When the switch is higher than the horizontal level, the switch is turned on, and when it is lower than the horizontal level, the switch is turned off. This tilt module can be used for tilt detection, alarm or other detection.

#### **Working Principle**

The working principle is pretty simple. When pin 1 and 2 of the ball switch P1 are connected, the signal S is low level and the red LED will light up; when they are disconnected, the pin will be pulled up by the 4.7K R1 and make S a high level, then LED will be off.



Components

				<b>50</b>
ESP32	ESP32 Expansion	Keyestudio	3P Dupont	Micro USB
Board*1	Board*1	Tilt Sensor*1	Wire*1	Cable*1

**Connection Diagram** 



fritzing

## **Test Code**



#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. Make the tilt module incline to one side, the red LED on the module will be off and the monitor will display"1". In contrast, if you make it incline the other side, the red LED will light up and the monitor will display "0".

💿 СОМЗ			_		$\times$
					Send
0					^
0					
0					
0					
0					
1					
1					
1					
					~
└─ Autoscroll └─ Show timestamp	Newline 🗸 🗸	9600 baud	~	Clear	r output

# 6.2.13 Project 13: Collision Sensor



### Description

The collision sensor uses a tact switch. This sensor is often used as a limit switch in 3D printers. In the experiment, we judge whether the sensor shrapnel is pressed down by reading the high and low levels of the S terminal on the module; and, we display the test results in the shell.

#### **Working Principle**

It mainly uses a tact switch. When the shrapnel of the tact switch is pressed, 2 and 3 are connected, the signal terminal S is low level, and the red LED on the module lights up; when the touch switch is not pressed, 2 and 3 are not connected, and 3 is pulled up to a high level by the 4.7K resistor R1, that is, the sensor signal terminal S is a high level, and the built-in red LED will be off at this time.



## **Components Required**

ESP32	ESP32 Expansion	Keyestudio	3P Dupont	Micro USB
Board*1	Board*1	Collision Sensor*1	Wire*1	Cable*1

## **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename : collision sensor
* Description : Reading the value of the collision sensor
* Auther
        : http://www.keyestudio.com
*/
int val = 0;
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(15, INPUT);//Set collision sensor pin 15 to input mode
}
void loop() {
 val = digitalRead(15);//Read the value of the collision sensor
 Serial.print(val);//Newline print
                                                             (continues on next page)
```

(continued from previous page)

#### Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. The serial monitor will display the corresponding data and characters.

In the experiment, when the shrapnel on the sensor is pressed down, val is 0, the red LED of the module is on, and "0 The end of his!" is printed; when the shrapnel is released, the val is 1, the red LED of the module is off, and "1 All going well" is printed. !" character, as shown below.

0	COM3			_		$\times$
						Send
1	All going well					^
1	All going well					
1	All going well					
1	All going well					
1	All going well					
1	All going well					
1	All going well					
0	The end of his!					
0	The end of his!					
0	The end of his!					
0	The end of his!					
0	The end of his!					
0	The end of his!					
0	The end of his!					
0	The end of his!					
						¥
	Autoscroll 🗌 Show timestamp Newline	~	9600 ba	ud v	Clear	output

# 6.2.14 Project 14: Hall Sensor



#### Description

In this kit, there is a Hall sensor which mainly adopts a A3144 linear Hall element. The element P1 is composed of a voltage regulator, a Hall voltage generator, a differential amplifier, a Schmitt trigger, a temperature compensation circuit and an open-collector output stage. In the experiment, we use the Hall sensor to detect the magnetic field and display the test results on the shell.

#### **Working Principle**

When the sensor detects no magnetic field or a north pole magnetic field, the signal terminal will be high level; when it senses a south pole magnetic field, the signal terminal will be low levels. The stronger the magnetic field strength is, induction distance is longer.



**Required Components** 

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY Hall Sensor*1	3P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

#### Test Code

```
/*
* Filename : Hall magnetic
* Description : Reading the value of hall magnetic sensor
* Auther
         : http://www.keyestudio.com
*/
int val = 0;
int hallPin = 15; //Hall sensor pin is connected to GPI015
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(hallPin, INPUT);//Set pin to input mode
}
void loop() {
 val = digitalRead(hallPin);//Read the level value of hall sensor
 Serial.print(val);//Print val
 if (val == 0) {//There is a South Pole magnetic field
   Serial.println(" The magnetic field at the South Pole!");
 }
```

(continues on next page)

(continued from previous page)

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. when the sensor detects no magnetic fields or the north pole magnetic field, the monitor l will show "1 Just be all normal!" and the LED on the sensor will be off; When it detects the south pole magnetic field, "0 The magnetic field at the South Pole!" and the LED on the sensor will be on.

💿 co	M3			—		$\times$
						Send
1	Just be all normal!					^
1	Just be all normal!					
1	Just be all normal!					
1	Just be all normal!					
1	Just be all normal!					
1	Just be all normal!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South Pole!					
0	The magnetic field at the South					~
Auto	oscroll 🗌 Show timestamp	Newline $\sim$	9600 baud	~	Clear	output

## 6.2.15 Project 15: Reed Switch Module



#### Overview

In this kit, there is a Keyestudio reed switch module, which mainly uses a MKA10110 green reed component.

The reed switch is the abbreviation of the dry reed switch. It is a passive electronic switch element with contacts.

It has the advantages of simple structure, small size and easy control.

Its shell is a sealed glass tube with two iron elastic reed electric plates.

In the experiment, we will determine whether there is a magnetic field near the module by reading the high and low level of the S terminal on the module; and, we display the test result in the shell.

#### Working Principle

In normal conditions, the glass tube in the two reeds made of special materials are separated. When a magnetic substance close to the glass tube, in the role of the magnetic field lines, the pipe within the two reeds are magnetized to attract each other in contact, the reed will suck together, so that the junction point of the connected circuit communication.



After the disappearance of the outer magnetic reed because of their flexibility and separate, the line is disconnected. The sensor uses this characteristic to build a circuit to convert magnetic field signal into high and low level signal.

#### Components

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY Reed Switch Module*1	3P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

## Test Code

//*************************************
/*
* Filename : Reed Switch
* Description : Read the value of the reed sensor
* Auther : http://www.keyestudio.com
*/
<pre>int val = 0;</pre>
<pre>int reedPin = 15; //Define dry reed module signal pin connected to GPI015</pre>
<pre>void setup() {</pre>
Serial.begin(9600);//Set baud rate to 9600
<pre>pinMode(reedPin, INPUT);//Set mode to input</pre>
}
<pre>void loop() {   val = digitalRead(reedPin);//Read digital level   Serial.print(val);//Serial port shows up</pre>
<pre>if (val == 0) {//There's a magnetic field nearby</pre>
<pre>Serial.print(" ");</pre>
<pre>Serial.println("A magnetic field");</pre>
delay(100);
}
<pre>else {//There is no magnetic field</pre>
<pre>Serial.print(" ");</pre>
<pre>Serial.println("There is no magnetic field"); delay(100);</pre>
}
}
//*************************************

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. The serial monitor will display the corresponding data and characters.

When the sensor detects a magnetic field, val is 0 and the red LED of the module lights up, "0 A magnetic field" will be displayed; when no magnetic field is detected, val is 1, and the LED on the module goes out, "1 There is no magnetic field" will be shown, as shown below.

💿 сомз				_		$\times$
						Send
1	There is no magnetic field					^
1	There is no magnetic field					
1	There is no magnetic field					
1	There is no magnetic field					
1	There is no magnetic field					
1	There is no magnetic field					
0	A magnetic field					
0	A magnetic field					
0	A magnetic field					
0	A magnetic field					
0	A magnetic field					
0	A magnetic field					
0	A magnetic field					
0	A magnetic field					_
0	A magnetic field					
0	A					~
Autosci	roll Show timestamp	Newline 🗸 🗸	9600 baud	~	Clear	output

## 6.2.16 Project 16: PIR Motion Sensor



Overview

In this kit, there is a Keyestudio PIR motion sensor, which mainly uses an RE200B-P sensor elements. It is a human body pyroelectric motion sensor based on pyroelectric effect, which can detect infrared rays emitted by humans or animals, and the Fresnel lens can make the sensor's detection range farther and wider.

In the experiment, we determine if there is someone moving nearby by reading the high and low levels of the S terminal on the module. The detected results will be displayed on the Shell.

### **Working Principle**

The upper left part is voltage conversion(VCC to 3.3V). The working voltage of sensors we use is 3.3V, therefore we can't use 5V directly. The voltage conversion circuit is needed.

When no person is detected or no infrared signal is received, and pin 1 of the sensor outputs low level. At this time, the LED on the module will light up and the MOS tube Q1 will be connected and the signal terminal S will detect Low levels.

When one is detected or an infrared signal is received, and pin 1 of the sensor outputs a high level. Then LED on the module will go off, the MOS tube Q1 is disconnected and the signal terminal S will detect high levels.



#### **Required Components**

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY PIR Motion Sensor*1	3P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

Test Code

```
/*
* Filename : PIR motion
* Description : Reading the value of the human body infrared sensor
* Auther
            : http://www.keyestudio.com
*/
int val = 0;
int pirPin = 15; //The pin of PIR motion sensor is defined as GPI015
void setup() {
 Serial.begin(9600); //Set baud rate to 9600
 pinMode(pirPin, INPUT); //Set the sensor to input mode
}
void loop() {
 val = digitalRead(pirPin); //Read the sensor value
 Serial.print(val);//Print val value
 if (val == 1) {//There is movement nearby, output high level
                      ");
   Serial.print("
   Serial.println("Some body is in this area!");
   delay(100);
 }
 else {//If no movement nearby, output low level
   Serial.print("
                      ");
   Serial.println("No one!");
   delay(100);
 }
}
                      ************************************
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. The serial monitor will display the corresponding data and characters.

When the sensor detects someone nearby, value is 1, the LED will go off and the monitor will show"1 Somebody is in

💿 сом	}			_		$\times$
						Send
0	No one!					^
0	No one!					
0	No one!					
0	No one!					
0	No one!					
0	No one!					
0	No one!					
0	No one!					
1	Some body is in this area!					
1	Some body is in this area!					
1	Some body is in this area!					
1	Some body is in this area!					
1	Some body is in this area!					
1	Some body is in this area!					
1	Some body is in this area!					
1	Some body is in this are					¥
Autoso	roll 🗌 Show timestamp	Newline $\sim$	9600 baud	~	Clea	r output

this area!". In contrast, the value is 0, the LED will go up and "0 No one!" will be shown.

## 6.2.17 Project 17: Active Buzzer



#### Overview

In this kit, it contains an active buzzer module and a power amplifier module (the principle is equivalent to a passive buzzer). In this experiment, we control the active buzzer to emit sounds. Since it has its own oscillating circuit, the buzzer will automatically sound if given large voltage.

### **Working Principle**

From the schematic diagram, the pin of buzzer is connected to a resistor R2 and another port is linked with a NPN triode Q1. So, if this triode Q1 is powered, the buzzer will sound.

If the base electrode of the triode connected to the R1 resistor is a high level, the triode Q1 will be connected. If the base electrode is pulled down by the resistor R3, the triode is disconnected.

When we output a high level from the IO port to the triode, the buzzer will emit sounds; if outputting low levels, the buzzer won't emit sounds.



#### Components

				<b>\$</b>
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Active Buzzer*1	3P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

## Test Code



#### **Code Explanation**

In the experiment, we set the pin to GPIO15. When setting to high, the active buzzer will beep; when setting to low, the active buzzer will stop emitting sounds.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The active buzzer will emit sound for 1 second, and stop for 1 second.

## 6.2.18 Project 18: 8002b Audio Power Amplifier



#### Overview

In this kit, there is a Keyestudio 8002b audio power amplifier. The main components of this module are an adjustable potentiometer, a speaker, and an audio amplifier chip.

The main function of this module is: it can amplify the output audio signal, with a magnification of 8.5 times, and play sound or music through the built-in low-power speaker, as an external amplifying device for some music playing equipment.

In the experiment, we used the 8002b power amplifier speaker module to emit sounds of various frequencies.

#### **Working Principle**

In fact, it is similar to a passive buzzer. The active buzzer has its own oscillation source. Yet, the passive buzzer does not have internal oscillation. When controlling the circuit, we need to input square waves of different frequencies to the positive pole of the component and ground the negative pole to control the buzzer to chime sounds of different frequencies.



## Components

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio 8002b Audio Power Amplifier*1	3P Dupont Wire*1	Micro USB Cable*1

**Connection Diagram** 



fritzing

**Test Code** 

/\* \* Filename : Passive Buzzer \* Description : Passive Buzzer sounds the alarm. \* Auther : http//www.keyestudio.com \*/ #define LEDC\_CHANNEL\_0 0 // LEDC timer uses 13 bit accuracy #define LEDC\_TIMER\_13\_BIT 13 // Define tool I/O ports #define BUZZER\_PIN 15 //Create a musical melody list, Super Mario int melody[] = {330, 330, 330, 262, 330, 392, 196, 262, 196, 165, 220, 247, 233, 220, →196, 330, 392, 440, 349, 392, 330, 262, 294, 247, 262, 196, 165, 220, 247, 233, 220, **.** →196, 330, 392,440, 349, 392, 330, 262, 294, 247, 392, 370, 330, 311, 330, 208, 220, **•** →262, 220, 262, 294, 392, 370, 330, 311, 330, 523, 523, 523, 392, 370, 330, 311, 330, 208, 220, 262,220, →262, 294, 311, 294, 262, 262, 262, 262, 262, 294, 330, 262, 220, 196, 262, 262, 262, 2 →262, 294, 330, 262, 262, 262, 262, 294, 330, 262, 220, 196}; //Create a list of tone durations  $\hookrightarrow$ 4,8,4,8,4,8,2,8,4,4,8,4,1,8,4,4,8,4,8,4,8,2}; void setup() {

(continues on next page)

(continued from previous page)

```
pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer to output mode
}
void loop() {
    int noteDuration; //Create a variable of noteDuration
    for (int i = 0; i < sizeof(noteDurations); ++i)
    {
        noteDuration = 800/noteDurations[i];
        ledcSetup(LEDC_CHANNEL_0, melody[i]*2, LEDC_TIMER_13_BIT);
        ledcAttachPin(BUZZER_PIN, LEDC_CHANNEL_0);
        ledcWrite(LEDC_CHANNEL_0, 50);
        delay(noteDuration * 1.30); //delay
    }
}</pre>
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onthen the power amplifier module will emit the sound on a loop.

# 6.2.19 Project 19: 130 Motor



#### Description

The 130 motor driver module is compatible with servo motors, which has high efficiency and good quality fans.

It adopts a HR1124S motor control chip. HR1124S is a single-channel H-bridge driver chip for DC motor solutions. In addition, this chip has low standby current and low quiescent current.

The module is compatible with various single-chip control boards. In the experiment, we can control the rotation direction of the motor by outputting the voltage directions of the two signal terminals IN+ and IN- to make the motor rotate.

#### **Working Principle**

The chip is used to help drive the motor. We can't drive it with a triode or an IO port due to its a large current of need. It is very simple to make the motor rotate. Just apply voltage to both ends of the motor. The direction of the motor is different in different voltage directions. Within the rated voltage, the higher the voltage, the faster the motor rotates; on the contrary, the lower the voltage, the slower the motor rotates, or even unable to rotate.

So we can use the PWM port to control the speed of the motor. We haven't learned PWM here, so we use the high and low levels to control the motor first.

## **Required Components**

			~
ESP32 Board*1	ESP32 Expansion Board*1	keyestudio DIY 130 Motor*1	4P Dupont Wire*1
Micro USB Cable*1 Battety Holder*1		Battety(not included in kit)*6	

Note: the motor is separated with its fan, you need to assemble it first.

#### **Connection Diagram**

130 Motor	ESP32 Expansion Board
G	G
V	5V
IN+	IO15
IN-	IO4



**Test Code** 

```
/*
* Filename
          : 130DC Fan motor
* Description : Motor positive and negative rotation
* Auther
          : http://www.keyestudio.com
*/
//Define two pins interfaces of the motor, respectively 15 and 4
int INA = 15; //INA corresponds to IN+
int INB = 4; //INB corresponds to IN-
void setup() {
 //Set the motor pins as output
 pinMode(INA, OUTPUT);
 pinMode(INB, OUTPUT);
}
void loop() {
 //Turn counterclockwise
 digitalWrite(INA, HIGH);
 digitalWrite(INB, LOW);
 delay(2000);
 //stop
 digitalWrite(INA, LOW);
 digitalWrite(INB, LOW);
 delay(1000);
 //clockwise rotation
 digitalWrite(INA, LOW);
 digitalWrite(INB, HIGH);
 delay(2000);
 //stop
 digitalWrite(INA, LOW);
 digitalWrite(INB, LOW);
 delay(1000);
}
```

#### **Code Explanation**

Set pins to GPIO4GPIO15, when the pin GPIO4 outputs low levels and the pin GPIO15 outputs high levels, the motor will rotate counterclockwise; when both pins are set to low, the motor stops rotating.

#### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch ON the ESP32 expansion board to the ON end, after powering on, compile and upload the code to the ESP32. After uploading

successfully the fan will rotate counterclockwise for 2 seconds, stop for 1 second and clockwise for 2 seconds and stop for 1 second; cycle alternately.

## 6.2.20 Project 20: Potentiometer



#### Overview

The following we will introduce is the Keyestudio rotary potentiometer which is an analog sensor.

The digital IO ports can read the voltage value between 0 and 3.3V and the module only outputs high levels. However, the analog sensor can read the voltage value through 16 ADC analog ports on the ESP32 board. In the experiment, we will display the test results on the Shell.

#### **Working Principle**



It uses a 10K adjustable resistor. We can change the resistance by rotating the potentiometer. The signal S can detect the voltage (0-3.3V) which are analog quantity.

**ADC:** The more bits an ADC has, the denser the partitioning of the simulation, the higher the accuracy of the final conversion.



Subsection 1: The analog value within 0V-3.3/4095 V corresponds to the number 0; Subsection 2: The analog value within 3.3/4095V-2\*3.3/4095V corresponds to the number 1; .....

The conversion formula is as follows:

$$ADCValue = \frac{Ana \log Voltage}{3.3} *4095$$

DAC: The higher the precision of DAC, the higher the precision of the output voltage value.

The conversion formula is as follows:

Ana log Voltage = 
$$\frac{DACValue}{255}$$
 \* 3.3(V)

#### ADC on ESP32

The ESP32 has 16 pins that can be used to measure analog signals. GPIO pin serial numbers and analog pin definitions are shown below:

ADC number in ESP32	ESP32 GPIO number
ADC0	GPIO 36
ADC3	GPIO 39
ADC4	GPIO 32
ADC5	GPIO33
ADC6	GPIO34
ADC7	GPIO 35
ADC10	GPIO 4
ADC11	GPIO0
ADC12	GPIO2
ADC13	GPIO15
ADC14	GPIO13
ADC15	GPIO 12
ADC16	GPIO 14
ADC17	GPIO27
ADC18	GPIO25
ADC19	GPIO26

### DAC on ESP32

The ESP32 has two 8-bit digital-to-analog converters connected to GPIO25 and GPIO26 pins, which are immutable, as shown below :

Simulate pin number	GPIO number
DAC1	GPIO25
DAC2	GPIO26

## Components

		Particular		R
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Rotary Potentiometer*1	3P Dupont Wire*1	Micro USB Cable*1

### **Connection Diagram**



# fritzing

## Test Code

```
/*
* Filename
           : Rotary_potentiometer
* Description : Read the basic usage of ADCDAC and Voltage
           : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 34 //the pin of the Potentiometer
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
```

(continued from previous page)

#### **Code Explanation**

1). analogVal means analog value. The rotary potentiometer outputs analog values  $(0 \sim 4095)$ , therefore, we set pins to analog ports. For example, we connect to GPIO34.

2). analogRead(pin): read the value of the specified analog pin. The ESP32 contains a multi-channel, 12-bit converter. This means that it will map the input voltage between 0 and the working voltage (5V or 3.3V) to an integer value between 0 and 4095. For example, this will produce a resolution among readings: 3.3V/4096 stands for 0.0008V per unit.

3). The map() function converts this 12-bit DAC value to an 8-bit DAC value.

4). Pin: the name of analog input pin.

5). The serial monitor displays the values of adcVal, dacVal, voltage, the baud rate must be set before display (we default to 9600, which can be changed).

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. The serial monitor will display the potentiometer's ADC value, DAC value and voltage value. Rotate the potentiometer handle, the analog values will change.

💿 сомз			_	$\Box$ $\times$
				Send
ADC Val: 456, DAC Val: 28,	Voltage: 0.37V			^
ADC Val: 1092, DAC Val: 68, ADC Val: 1497 DAC Val: 93	Voltage: 0.88V			
ADC Val: 1957, DAC Val: 55, ADC Val: 1919, DAC Val: 119, ADC Val: 2141 DAC Val: 122	Voltage: 1.55V			
ADC Val: 2141, DAC Val: 133, ADC Val: 2379, DAC Val: 148, ADC Val: 2797, DAC Val: 174	Voltage: 1.73V Voltage: 1.92V			
ADC Val: 2757, DAC Val: 174, ADC Val: 3031, DAC Val: 188, ADC Val: 2458, DAC Val: 215	Voltage: 2.23V Voltage: 2.44V			
ADC Val: 3435, DAC Val: 215, ADC Val: 4031, DAC Val: 251, ADC Val: 4005, DAC Val: 251,	Voltage: 2.75V Voltage: 3.25V			
ADC Val: 4095, DAC Val: 255, ADC Val: 4095, DAC Val: 255,	Voltage: 3.30V Voltage: 3.30V			
ADC Val: 4095, DAC Val: 255, ADC Val: 4095, DAC Val: 255,	Voltage: 3.30V Voltage: 3.30V			U
Autoscroll 🗌 Show timestamp	no processi final h	Newline ~	9600 baud $\sim$	Clear output

## 6.2.21 Project 21: Steam Sensor



#### Description

This is a DIY electronic building block water drop sensor. It is an analog (digital) input module, also called rain, rain sensor. It can be used to monitor various weather conditions, detect whether it is raining and the amount of rain, convert it into digital signal (DO) and analog signal (AO) output, and is widely used in Arduino robot kits, raindrops, rain sensors, and can be used for various It can monitor various weather conditions, and convert it into digital signal and AO output, and can also be used for automobile automatic wiper system, intelligent lighting system and intelligent sunroof system.

In the experiment, we input the sensor signal terminal (S terminal) to the analog port of the ESP32 development board, sense the change of the analog value, and display the corresponding analog value on the shell.

#### Working Principle

Its principle is to detect the amount of water through the exposed printed parallel lines on the circuit board. The more water there is, the more wires will be connected, and the conductive contact area increases. The voltage output by pin 2 will gradually increase. The larger the analog value detected by the signal terminal S is.

It can also detect steam in the air. Two position holes are used to install on the other devices.



## **Required Components**

				<b>%</b>
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY Steam Sensor *1	3P Dupont Wire*1	Micro USB Cable*1

## **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename : Steam sensor
* Description : Read the basic usage of ADCDAC and Voltage
          : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 34 //the pin of the Steam sensor
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\leftrightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32.

After uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. The serial monitor will display the steam sensor's ADC value, DAC value and voltage value. When a few drops of water are placed in the sensor sensing area, the values will change. The more water volume, the greater the output voltage value , ADC value and the DAC value .

💿 СОМЗ			_	
				Send
ADC Val: 456, DAC Val: 28,	Voltage: 0.37V			^
ADC Val: 778, DAC Val: 48, ADC Val: 1092, DAC Val: 68,	Voltage: 0.63V Voltage: 0.88V			
ADC Val: 1497, DAC Val: 93,	Voltage: 1.21V			
ADC Val: 1919, DAC Val: 119, ADC Val: 2141, DAC Val: 133,	Voltage: 1.55V Voltage: 1.73V			
ADC Val: 2379, DAC Val: 148,	Voltage: 1.92V			
ADC Val: 2797, DAC Val: 174, ADC Val: 3031, DAC Val: 188,	Voltage: 2.25V Voltage: 2.44V			
ADC Val: 3459, DAC Val: 215, ADC Val: 4031 DAC Val: 251	Voltage: 2.79V Voltage: 3.25V			
ADC Val: 4095, DAC Val: 255,	Voltage: 3.30V			
ADC Val: 4095, DAC Val: 255, ADC Val: 4095, DAC Val: 255,	Voltage: 3.30V Voltage: 3.30V			
ADC Val: 4095, DAC Val: 255,	Voltage: 3.30V			
Autoscroll Show timestern		Newline	9600 haud 🗸	Clear output
C Autoscivii biow timestamp		Mealine v	oooo oadu v	orear output

# 6.2.22 Project 22: Sound Sensor



## Overview

In this kit, there is a Keyestudio DIY electronic block and a sound sensor.
In the experiment, we test the analog value corresponding to the sound level in the current environment with it. The louder the sound, the larger the ADC, DAC and the voltage value, and the serial monitor window will display the test results.

#### Working Principle

It uses a high-sensitive microphone component and an LM386 chip. We build the circuit with the LM386 chip and amplify the sound through the high-sensitive microphone. In addition, we can adjust the sound volume by the potentiometer. Rotate it clockwise, the sound will get louder.



#### Components

		Microphase		
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY Sound Sensor*1	3P Dupont Wire*1	Micro USB Cable*1



fritzing

# **Test Code**

```
/*
* Filename
            : MicroPhone
* Description : Read the basic usage of ADCDAC and Voltage
            : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 34 //the pin of the Sound Sensor
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the.
→information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
               ******
```

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. The serial monitor will display the sound sensor's ADC value, DAC value and voltage value. Rotate the potentiometer clockwise and speak at the MIC. Then you can see the analog value get larger, as shown below:

💿 сомз				_		<
					Send	L
ADC Val: 352,	DAC Val: 21,	Voltage: 0.28V				^
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 288,	DAC Val: 17,	Voltage: 0.23V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 576,	DAC Val: 35,	Voltage: 0.46V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 280,	DAC Val: 17,	Voltage: 0.23V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 244,	DAC Val: 15,	Voltage: 0.20V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
ADC Val: 256,	DAC Val: 15,	Voltage: 0.21V				
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V				
						¥
Autoscroll	Show timestamp	on present fixed b	Newline $\sim$	9600 baud $\sim$	Clear out	put

# 6.2.23 Project 23: Photoresistor



### Description

In this kit, there is a photoresistor consists of photosensitive resistance elements. Its resistance changes with the light intensity. Also, it converts the resistance change into a voltage change through the characteristic of the photosensitive

resistive element. When wiring it up, we interface its signal terminal (S terminal) with the analog port of ESP32, so as to sense the change of the analog value, and display the corresponding analog value in the shell.

#### **Working Principle**

If there is no light, the resistance is 0.2M and the detected voltage at the terminal is close to 0. When the light intensity increases, the resistance of photoresistor and detected voltage will diminish, and the detected voltage is increasing.



#### Components

		CONTRACTOR		
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY Photoresistor*1	3P Dupont Wire*1	MicroUSB Cable*1



fritzing

# Test Code

```
/*
* Filename
            : Photoresistance
* Description : Read the basic usage of ADCDAC and Voltage
* Auther
            : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 34 //the pin of the Photoresistance
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the.
→information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
             ******
```

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32.

After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. The serial monitor will display the photoresistor's ADC value, DAC value and voltage value. When the light intensity gets stronger, the analog values will get larger, as shown below:

💿 сомз					_		$\times$
							Send
ADC Val: 343,	DAC Val: 21,	Voltage: 0.28V					^
ADC Val: 348,	DAC Val: 21,	Voltage: 0.28V					
ADC Val: 464,	DAC Val: 28,	Voltage: 0.37V					
ADC Val: 1128,	DAC Val: 70,	Voltage: 0.91V					
ADC Val: 2042,	DAC Val: 127,	Voltage: 1.65V					
ADC Val: 2153,	DAC Val: 134,	Voltage: 1.74V					
ADC Val: 2182,	DAC Val: 135,	Voltage: 1.76V					
ADC Val: 2241,	DAC Val: 139,	Voltage: 1.81V					
ADC Val: 2327,	DAC Val: 144,	Voltage: 1.88V					
ADC Val: 2407,	DAC Val: 149,	Voltage: 1.94V					
ADC Val: 2702,	DAC Val: 168,	Voltage: 2.18V					
ADC Val: 2607,	DAC Val: 162,	Voltage: 2.10V					
ADC Val: 2595,	DAC Val: 161,	Voltage: 2.09V					
ADC Val: 2511,	DAC Val: 156,	Voltage: 2.02V					
ADC Val: 2475,	DAC Val: 154,	Voltage: 1.99V					
							¥
Autoscroll	Show timestamp		Newline	∨ 9600 baud	$\sim$	Clear	output



# 6.2.24 Project 24: NTC-MF52AT Thermistor

#### Overview

In the experiment, there is a NTC-MF52AT analog thermistor. We connect its signal terminal to the analog port of the ESP32 mainboard and read the corresponding ADC value, voltage value and thermistor value.

We can use analog values to calculate the temperature of the current environment through specific formulas. Since the temperature calculation formula is more complicated, we only read the corresponding analog value.

### **Working Principle**



This module mainly uses NTC-MF52AT thermistor element, which can sense the changes of the surrounding environment temperature. Resistance changes with the temperature, causing the voltage of the signal terminal S to change.

This sensor uses the characteristics of NTC-MF52AT thermistor element to convert resistance changes into voltage changes.

# Components

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio NTC-MF52AT Thermistor*1	3P Dupont Wire*1	Micro USB Cable*1



fritzing

# Test Code



#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. The serial monitor will display the thermistor's ADC value, DAC value and voltage value, as shown below:

🐵 сомз			- 🗆 X
			Send
ADC value : 1408,	Voltage : 1.13V,	Temperature : 22.57C	^
ADC value : 1390,	Voltage : 1.12V,	Temperature : 23.01C	
ADC value : 1408,	Voltage : 1.13V,	Temperature : 22.57C	
ADC value : 1396,	Voltage : 1.12V,	Temperature : 22.86C	
ADC value : 1392,	Voltage : 1.12V,	Temperature : 22.96C	
ADC value : 1389,	Voltage : 1.12V,	Temperature : 23.03C	
ADC value : 1414,	Voltage : 1.14V,	Temperature : 22.43C	
ADC value : 1501,	Voltage : 1.21V,	Temperature : 20.39C	
ADC value : 1625,	Voltage : 1.31V,	Temperature : 17.62C	
ADC value : 1554,	Voltage : 1.25V,	Temperature : 19.19C	
ADC value : 1562,	Voltage : 1.26V,	Temperature : 19.01C	
ADC value : 1566,	Voltage : 1.26V,	Temperature : 18.92C	
ADC value : 1571,	Voltage : 1.27V,	Temperature : 18.81C	
ADC value : 1571,	Voltage : 1.27V,	Temperature : 18.81C	
ADC value : 1571,	Voltage : 1.27V,	Temperature : 18.81C	
			×
🗹 Autoscroll 🗌 Show tim	estamp	Newline 🗸 9600 baud	∨ Clear output

# 6.2.25 Project 25: Thin-film Pressure Sensor



# Overview

In this kit, there is a Keyestudio thin-film pressure sensor, which is composed of a new type of nano pressure-sensitive material and a comfortable ultra-thin film substrate, has waterproof and pressure-sensitive functions.

In the experiment, we determine the pressure by collecting the analog signal on the S end of the module. The smaller the ADC value, DAC value and voltage value, the greater the pressure; and the displayed results will shown on the

Shell.



# **Working Principle**

When the sensor is pressed by external forces, the resistance value of sensor will vary. We convert the pressure signals detected by the sensor into the electric signals through a circuit. Then we can obtain the pressure changes by detecting voltage signal changes.

### Components

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Thin-film Pressure Sensor*1	3P Dupont Wire*1	Micro USB Cable*1



fritzing

### **Test Code**

```
/*
* Filename
          : Film pressure sensor
* Description : Read the basic usage of ADCDAC and Voltage
* Auther
            : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 34 //the pin of the Film pressure sensor
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC_
⇔value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After

uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. The serial monitor will display the thin-film's ADC value, DAC value and voltage value, when the thin-film is pressed by fingers, the analog value will decrease, as shown below;

💿 COM3				—		$\times$
						Send
ADC Val: 2239, DAC Val: 139,	Voltage: 1.80V					^
ADC Val: 12255, DAC Val: 135, ADC Val: 1046, DAC Val: 65, ADC Val: 1285, DAC Val: 60,	Voltage: 1.80V Voltage: 0.84V					
ADC Val: 1295, DAC Val: 80, ADC Val: 577, DAC Val: 35,	Voltage: 1.04V Voltage: 0.46V					
ADC Val: 372, DAC Val: 23, ADC Val: 336, DAC Val: 20,	Voltage: 0.30V Voltage: 0.27V					
ADC Val: 250, DAC Val: 15, ADC Val: 204, DAC Val: 12,	Voltage: 0.20V Voltage: 0.16V					
ADC Val: 209, DAC Val: 13, ADC Val: 186, DAC Val: 11,	Voltage: 0.17V Voltage: 0.15V					
ADC Val: 272, DAC Val: 16, ADC Val: 394, DAC Val: 24,	Voltage: 0.22V Voltage: 0.32V					
ADC Val: 800, DAC Val: 49, ADC Val: 2242, DAC Val: 139,	Voltage: 0.64V Voltage: 1.81V					
Autoscroll Show timestamp		Newline	✓ 9600 baud	~	Clear	v output
						-

# 6.2.26 Project 26: Flame Sensor



Description

In daily life, it is often seen that a fire broke out without any precaution. It will cause great economic and human loss. So how can we avoid this situation? Right, install a flame sensor and a speaker in those places that easily break out a fire. When the flame sensor detects a fire, the speaker will alarm people quickly to put out the fire.

So in this project, you will learn how to use a flame sensor and an active buzzer module to simulate the fire alarm system.

#### Working Principle

This flame sensor can be used to detect fire or other light sources with wavelength stands at 700nm  $\sim$  1000nm. Its detection angle is about 60°. You can rotate the potentiometer on the sensor to control its sensitivity. Adjust the potentiometer to make the LED at the critical point between on and off state. The sensitivity is the best.

From the below figure, power up. When detecting fire, the digital pin outputs low levels, the red LED2 will light up first, the digital signal terminal D0 outputs a low level, and the red LED1 will light up. The stronger the external infrared light, the smaller the value; the weaker the infrared light, the larger the value.



**Required Components** 

			2	
ESP32 Board*1	ESP32 Expansion Board*1	keyestudio DIY Flame Sensor*1	4P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

# **Test Code**

(continues on next page)

(continued from previous page)

```
void setup() {
 Serial.begin(9600);
 pinMode(digitalPin, INPUT); //Digital pin 13 is set to input mode
}
//In loop()the digitalRead()function is used to obtain the digital value,
//the analogRead() function is used to obtain the ADC value.
//the map() function is used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the
→information is finally printed out.
void loop() {
 int digitalVal = digitalRead(digitalPin); //Read digital signal;
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("digitalVal: %d, \t ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n",
→digitalVal, adcVal, dacVal, voltage);
 delay(200);
}
```

# **Code Explanation**

Two pins we use are defined as GPIO13 and GPIO34 according to the wiring-up diagram, and print digital signals and analog signals respectively.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the sensor, we can adjust the red LED bright and not bright critical point. The red LED2 on the sensor module is lit, while the red LED1 is not. Open the monitor and set baud rate to 9600. The "Shell" window will display the digital value, ADC value, DAC value and voltage value of the flame sensor. When fire is detected, the LED1 will be on. the digital value will change from 1 to 0, and the analog value will become smaller, as shown below.

💿 сомз		_	
			Send
digitalVal: 1,	ADC Val: 4095,	DAC Val: 255, Voltage: 3.30V	^
digitalVal: 1,	ADC Val: 3583,	DAC Val: 223, Voltage: 2.89V	
digitalVal: 1,	ADC Val: 3491,	DAC Val: 217, Voltage: 2.81V	
digitalVal: 1,	ADC Val: 2639,	DAC Val: 164, Voltage: 2.13V	
digitalVal: 1,	ADC Val: 2803,	DAC Val: 174, Voltage: 2.26V	
digitalVal: 1,	ADC Val: 3671,	DAC Val: 228, Voltage: 2.96V	
digitalVal: 1,	ADC Val: 3277,	DAC Val: 204, Voltage: 2.64V	
digitalVal: 1,	ADC Val: 2334,	DAC Val: 145, Voltage: 1.88V	
digitalVal: 0,	ADC Val: 1130,	DAC Val: 70, Voltage: 0.91V	
digitalVal: 0,	ADC Val: 672,	DAC Val: 41, Voltage: 0.54V	
digitalVal: 0,	ADC Val: 458,	DAC Val: 28, Voltage: 0.37V	
digitalVal: 0,	ADC Val: 1456,	DAC Val: 90, Voltage: 1.17V	
digitalVal: 0,	ADC Val: 1216,	DAC Val: 75, Voltage: 0.98V	
digitalVal: 0,	ADC Val: 1152,	DAC Val: 71, Voltage: 0.93V	
digitalVal: 0,	ADC Val: 1050,	DAC Val: 65, Voltage: 0.85V	
			¥
🗹 Autoscroll 🗌	Show timestamp	Newline $\sim$ 9600 baud $\sim$	Clear output

# 6.2.27 Project 27: MQ-2 Gas Sensor

#### Description

This analog gas sensor - MQ2 is used in gas leakage detecting equipment in consumer electronics and industrial markets.

This sensor is suitable for detecting LPG, I-butane, propane, methane, alcohol, Hydrogen and smoke. It has high sensitivity and quick response.

In addition, the sensitivity can be adjusted by rotating the potentiometer.

In the experiment, we read the analog value at the A0 port and the D0 port to determine the content of gas.

#### **Working Principle**

The greater the concentration of smoke, the greater the conductivity, the lower the output resistance, the greater the output analog signal.

When in use, the A0 terminal reads the analog value of the corresponding gas; the D0 terminal is connected to an LM393 chip (voltage comparator), we can adjust the alarm threshold of the measured gas through the potentiometer, and output the digital value at D0. When the measured gas content exceeds the critical point, the D0 terminal outputs a low level; when the measured gas content does not exceed the critical point, the D0 terminal outputs a high level.



### **Required Components**

			3	
ESP32 Board*1	ESP32 Expansion Board*1	keyestudio DIY Analog Gas Sensor*1	4P Dupont Wire*1	Micro USB Cable*1



**Test Code** 

```
/*
* Filename : MQ2
* Description : Read the basic usage of Digital, ADCDAC and Voltage
* Auther : http//www.keyestudio.com
*/
//MQ_2 two pins 13, 34, respectively
#define PIN_ANALOG_IN 34
int digitalPin = 13;
//The following two variables hold the digital signal and adc values respectively
int analogVal = 0;
int adcVal = 0;
void setup() {
 Serial.begin(9600);
 pinMode(digitalPin, INPUT); //Digital pin 13 is set to input mode
}
//In loop()the digitalRead()function is used to obtain the digital value,
//the analogRead() function is used to obtain the ADC value.
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
```

(continues on next page)

```
(continued from previous page)
```

```
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int digitalVal = digitalRead(digitalPin); //Read digital signal;
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("digitalVal: %d, \t ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n",
→digitalVal, adcVal, dacVal, voltage);
 if (digitalVal == 1) {
   Serial.println(" Normal");
 }
 else {
   Serial.println(" Exceeding");
 }
 delay(100); //Delay time 100 ms
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32.

After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the sensor, we can adjust the red LED bright and not bright critical point. Open the monitor, set baud rate to 9600 and display the corresponding data and characters. When the sensor detects the smoke or combustible gas, the red LED lights up and the digital value changes from 1 to 0, the ADC value, DAC value and voltage value increase, as shown below.

🚥 COM3		- 🗆 X
		Send
digitalVal: 1, ADC Val: 80,	DAC Val: 4, Voltage: 0.06V	^
digitalVal: 1, ADC Val: 76, Normal	DAC Val: 4, Voltage: 0.06V	
digitalVal: 1, ADC Val: 76, Normal	DAC Val: 4, Voltage: 0.06V	
digitalVal: 1, ADC Val: 80, Normal	DAC Val: 4, Voltage: 0.06V	
digitalVal: 1, ADC Val: 80, Normal	DAC Val: 4, Voltage: 0.06V	
digitalVal: 0, ADC Val: 1769 Exceeding	, DAC Val: 110, Voltage: 1.43V	
digitalVal: 0, ADC Val: 3111 Exceeding	, DAC Val: 193, Voltage: 2.51V	
digitalVal: 0, ADC Val: 3411 Exceeding	, DAC Val: 212, Voltage: 2.75V	
digitalVal: 0, ADC Val: 3455 Exceeding	, DAC Val: 215, Voltage: 2.78V	
digitalVal: 0, ADC Val: 3474 Exceeding	, DAC Val: 216, Voltage: 2.80V	
digi		¥
🗹 Autoscroll 🗌 Show timestamp	Newline $\vee$ 960	) baud v Clear output

# 6.2.28 Project 28: MQ-3 Alcohol Sensor



#### Description

In this kit, there is a MQ-3 alcohol sensor, which uses the gas-sensing material is tin dioxide (SnO2) which has a low conductivity in clean air. When there is alcohol vapor in the environment where the sensor is located, the conductivity of the sensor increases with the increase of the alcohol gas concentration in the air. The change in conductivity can be converted into an output signal corresponding to the gas concentration using a simple circuit.

In the experiment, we read the analog value at the A0 end of the sensor and the digital value at the D0 end to judge the content of alcohol vapor in the air and whether they exceed the standard.

#### **Working Principle**

At a certain temperature, the conductivity changes with the composition of the ambient gas. When in use, A0 terminal reads the analog value corresponding to alcohol vapor; D0 terminal is connected to an LM393 chip (comparator), we can adjust and measure the alcohol vapor alarm threshold through the potentiometer, and output the digital value at D0. When the measured alcohol vapor content exceeds the critical point, the D0 terminal outputs a low level; when the measured alcohol vapor content does not exceed the critical point, the D0 terminal outputs a high level.



# **Components Required**

			3	
ESP32 Boar d*1	ESP32 Expansion Board*1	kevestudio Alcohol Sensor*1	4P <u>Dupont</u> Wire*1	MicroUSBCable*1



# fritzing

# Test Code

```
/*
* Filename
            : MQ3
* Description : Read the basic usage of Digital, ADCDAC and Voltage
* Auther
            : http//www.keyestudio.com
*/
//MQ_3 two pins 13, 34, respectively
#define PIN_ANALOG_IN 34
int digitalPin = 13;
//The following two variables hold the digital signal and adc values respectively
int analogVal = \emptyset;
int adcVal = 0;
void setup() {
 Serial.begin(9600);
 pinMode(digitalPin, INPUT); //Digital pin 13 is set to input mode
}
//In loop()the digitalRead()function is used to obtain the digital value,
//the analogRead() function is used to obtain the ADC value.
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int digitalVal = digitalRead(digitalPin); //Read digital signal;
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
```

(continues on next page)

(continued from previous page)

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the sensor, we can adjust the yellow and green LED bright and not bright critical point. Open the monitor, set baud rate to 9600 and display the corresponding data and characters. When the sensor detects the alcohol gas, the yellow and green LED lights up and the digital value changes from 1 to 0, the ADC value, DAC value and voltage value decrease, as shown below.

💿 COM3				_		$\times$
					S	Gend
digitalVal: 1, ADC V Normal	Val: 80, DAC Val	: 4, Voltage	: 0.06V			^
digitalVal: 1, ADC V Normal	Val: 76, DAC Val	: 4, Voltage	: 0.06V			
digitalVal: 1, ADC V Normal	Val: 76, DAC Val	: 4, Voltage	: 0.06V			
digitalVal: 1, ADC V Normal	Val: 80, DAC Val	: 4, Voltage	: 0.06V			
digitalVal: 1, ADC V Normal	Val: 80, DAC Val	: 4, Voltage	: 0.06V			
digitalVal: 0, ADC V Exceeding	Val: 1769,	DAC Val: 110,	Voltage: 1.43V			
digitalVal: 0, ADC V Exceeding	Val: 3111,	DAC Val: 193,	Voltage: 2.51V			
digitalVal: 0, ADC V Exceeding	Val: 3411,	DAC Val: 212,	Voltage: 2.75V			
digitalVal: 0, ADC V Exceeding	Val: 3455,	DAC Val: 215,	Voltage: 2.78V			
digitalVal: 0, ADC V Exceeding	Val: 3474,	DAC Val: 216,	Voltage: 2.80V			
digi						~
Autoscroll Show tim	mestamp	Newlin	e ~ 9600 baud	$\sim$	Clear	output



# 6.2.29 Project 29: Five-key AD Button Module

#### Description

When we talked about analog and digital sensors earlier, we talked about the single-channel key module. When we press the key, it outputs a low level, and when we release the key, it outputs a high level. We can only read these two digital signals. In fact, the key module ADC acquisition can also be performed. In this kit, a DIY electronic building block five-way AD button module is included.

We can judge which key is pressed through the analog value. In the experiment, we print out the key press information in the shell.

#### Working Principle

Let's look at the schematic diagram, when we do not press the key, the OUT of S output to the signal end is pulled down by R1. At this time, we read the low level 0V. When we press the key SW1, the OUT of the output to the signal end S is directly connected to the VCC. At this time, we read the high level 3.3V(the figure is marked as a 12-bit ADC(0~4095) and VCC is 5V. The principle is the same. Here we have VCC of 3.3V and ADC mapped to 12 bits), which is an analog value of 4095.

Next, when we press the key SW2, the OUT terminal voltage of the signal we read is the voltage between R2 and R1, namely VCC\*R1/(R2+R1), which is about 2.64V, and the analog value is about 3276.

When we press the key SW3, the OUT terminal voltage of the signal we read is the voltage between R2+R3 and R1, namely VCC\*R1/(R3+R2+R1), which is about 1.99V, and the analog value is about 2469.

When we press the key SW4, the OUT terminal voltage of the signal we read is the voltage between R2+R3+R4 and R1, namely VCC\*R1/(R4+R3+R2+R1), about 1.31V, and the analog value is about 1626.

Similarly, when we press the key SW5, the OUT terminal voltage of the signal we read is the voltage between R2+R3+R4+R5 and R1, namely VCC\*R1/(R5+R4+R3+R2+R1), which is about 0.68V, and the analog value is about 844. R4+R3+R2+R1), which is about 0.68V, and the analog value is about 844.



# **Components Required**

		: 101 : 6 101 101 101 <sup>21</sup> : 101 :	8	
ESP32 Board*1	ESP32 Expansion Board*1	keyestudio 5-Channel AD Button Module*1	3P Dupont Wire*1	Micro USB Cable*1



fritzing

# Test Code

```
/*
* Filename : Five AD Keys
* Description : Read the value of Five AD Keys
* Auther
         : http//www.keyestudio.com
*/
int val = 0;
int ADkey = 34; //Define five AD keys connected to GPI036
void setup() {
 Serial.begin(9600); //Set baud rate to 9600
}
void loop() {
 val = analogRead(ADkey); //Read the simulated value of the AD key and assign it to.
\rightarrow the variable val
 Serial.print(val); //A newline prints the variable val
 if (val <= 500) { //Val is less than or equal to 500 when no button is pressed
   Serial.println(" no key is pressed");
 } else if (val <= 1000) { //When key 5 is pressed,val is between 500 and 1000
   Serial.println(" SW5 is pressed");
 } else if (val <= 2000) { //When pressed,val is between 1000 and 2000
   Serial.println(" SW4 is pressed");
 } else if (val <= 3000) { //When pressed,val is between 2000 and 3000</pre>
   Serial.println(" SW3 is pressed");
 } else if (val <= 4000) { //When key 2 is pressed, val is between 3000 and 4000
   Serial.println(" SW2 is pressed");
 } else { //When key 1 is pressed, val is greater than 4000
   Serial.println(" SW1 is pressed");
 }
}
```

#### **Code Explanation**

We assign the read analog value to the variable val, and the serial monitor displays the value of val, (we set to 9600).

When the analog value is in the range of 500 and 1000, the button SW5 is pressed; when the analog value is in the 1000 and 2000, the button SW4 is pressed; when the analog value is between 2000 and 3000, the button SW3 is pressed; when the analog value is between 3000 and 4000, the button SW2 is pressed. When the analog value is above 4000, we judge that the button SW1 is pressed.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600; when the button is pressed, the serial monitor prints out the corresponding information, as shown in the figure below.

0	COI	ИЗ			_		$\times$
							Send
0	no	key is pressed					~
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
3141	L	SW2 is pressed					
3127	7	SW2 is pressed					
3139	5	SW2 is pressed					
3139	5	SW2 is pressed					
3129	5	SW2 is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
0	no	key is pressed					
4099	5	SW1 is pressed					
4099	5	SW1 is pressed					
4099	5	SW1 is pressed					
4099	5						¥
	Auto	scroll Show timestamp	ewline $\vee$	9600 baud	$\sim$	Clear	output

# 6.2.30 Project 30: Joystick Module



#### Overview

Game handle controllers are ubiquitous. There is a joystick module in this kit, which mainly uses PS2 joysticks. When controlling it, we need to connect the X and Y ports of the module to the analog port of the single-chip microcomputer, port B to the digital port of the single-chip microcomputer, VCC to the power output port(3.3-5V), and GND to the GND of the MCU. We can read the high and low levels of two analog values and one digital port) to determine the working status of the joystick on the module.

In the experiment, two analog values(x axis and y axis) will be shown on the Shell.

# Working Principle



In fact, its working principle is very simple. Its inside structure is equivalent to two adjustable potentiometers and a button. When this button is not pressed and the module is pulled down by R1, low levels will be output ; on the contrary, when the button is pressed, VCC will be connected (high levels). When we move the joystick, the internal potentiometer will adjust to output different voltages, and we can read the analog value.

# Components

		Control of the second s	• <u>·</u> ···	
	ESP32	Keyestudio		
ESP32	F	-	5P Dupont	Micro USB
Board*1	Expansion	JOYSTICK	Wire*1	Cable*1
	Board*1	Module*1		



# fritzing

#### **Test Code**

```
/*
* Filename : Joystick
* Description : Read data from Rocker.
* Auther
        : http//www.keyestudio.com
*/
int xyzPins[] = {34, 35, 13}; //x,y,z pins
void setup() {
 Serial.begin(9600);
 pinMode(xyzPins[0], INPUT); //x axis.
 pinMode(xyzPins[1], INPUT); //y axis.
 pinMode(xyzPins[2], INPUT_PULLUP); //z axis is a button.
}
// In loop(), use analogRead () to read the value of axes X and Y
//and use digitalRead () to read the value of axis Z, then display them.
void loop() {
 int xVal = analogRead(xyzPins[0]);
 int yVal = analogRead(xyzPins[1]);
 int zVal = digitalRead(xyzPins[2]);
 Serial.println("X,Y,Z: " + String(xVal) + ", " + String(yVal) + ", " + String(zVal));
 delay(500);
}
```

#### **Code Explanation**

In the experiment, according to the wiring diagram, the x pin is set to GPIO34, the y pin is set to GPIO35 and the pin of the joystick is set to GPIO13.

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After

uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600;

The serial monitor will show the corresponding value. Moving the joystick or pressing it will change the analog and digital values in the serial monitor .





# 6.2.31 Project 31: Relay Module

### Overview

In our daily life, we usually use communication to drive electrical equipment, and sometimes we use switches to control electrical equipment. If the switch is connected directly to the ac circuit, leakage occurs and people are in danger. Therefore, from the perspective of safety, we specially designed this relay module with NO(normally open) end and NC(normally closed) end.

### Working Principle

Relay is compatible with a variety of microcontroller control board, such as Arduino series microcontroller, which is a small current to control the operation of large current "automatic switch".

Input Voltage3.3V-5V



It can let the MCU control board drive 3A load, such as an LED lamp belt, a DC motor, a micro water pump and a solenoid valve pluggable interface design, which is easy to use.

#### **Components Required**

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Relay Module*1	3P Dupont Wire*1	Micro USB Cable*1



# fritzing

# **Test Code**

```
/*
* Filename : Relay
* Description : Relay turn on and off.
* Auther
         : http//www.keyestudio.com
*/
#define Relay 15 // defines digital 15
void setup()
{
pinMode(Relay, OUTPUT); // sets "Relay" to "output"
}
void loop()
{
digitalWrite(Relay, HIGH); // turns on the relay
delay(1000); //delays 1 seconds
digitalWrite(Relay, LOW); // turns off the relay
delay(1000); // delays 1 seconds
}
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The relay will cycle on and off, on for 1 second, off for 1 second. At the same time, you can hear the sound of the relay on and off as well as see the change of the indicator light on the relay.

# 6.2.32 Project 32: SK6812 RGB Module



#### Overview

In previous lessons, we learned about the plug-in RGB module and used PWM signals to color the three pins of the module.

There is a Keyestudio 6812 RGB module whose the driving principle is different from the plug-in RGB module. It can only control with one pin. This is a set. It is an intelligent externally controlled LED light source with the control circuit and the light-emitting circuit. Each LED element is the same as a 5050 LED lamp bead, and each component is a pixel. There are four lamp beads on the module, which indicates four pixels.

In the experiment, we make different lights show different colors.

#### Working Principle

From the schematic diagram, we can see that these four pixel lighting beads are all connected in series. In fact, no matter how many they are, we can use a pin to control a light and let it display any color. The pixel point contains a data latch signal shaping amplifier drive circuit, a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.

The data protocol adopts a single-wire zero-code communication method. After the pixel is powered up and reset, the S terminal receives the data transmitted from the controller. The first 24bit data sent is extracted by the first pixel and sent to the data latch of the pixel.



Components

	CLP-32 Neversitie			
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio 6812 RGB Module*1	3P Dupont Wire*1	MicroUSB Cable*1

#### **Connection Diagram**



fritzing

**Test Code**
```
/*
* Filename : sk6812 RGB LED
* Description : turn on sk6812 RGB LED
* Auther : http//www.keyestudio.com
*/
#include <Adafruit_NeoPixel.h>
#define PIN 15
// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
// NEO_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
// NEO_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
// NEO_GRB
               Pixels are wired for GRB bitstream (most NeoPixel products)
// NEO_RGB
              Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across
// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input
// and minimize distance between Arduino and first pixel. Avoid connecting
// on a live circuit...if you must, connect GND first.
void setup() {
 strip.begin();
 strip.show(); // Initialize all pixels to 'off'
}
void loop() {
 // Some example procedures showing how to display to the pixels:
 colorWipe(strip.Color(255, 0, 0), 50); // Red
 colorWipe(strip.Color(0, 255, 0), 50); // Green
 colorWipe(strip.Color(0, 0, 255), 50); // Blue
 // Send a theater pixel chase in...
 theaterChase(strip.Color(127, 127, 127), 50); // White
 theaterChase(strip.Color(127, 0, 0), 50); // Red
 theaterChase(strip.Color( 0, 0, 127), 50); // Blue
 rainbow(20);
 rainbowCycle(20);
 theaterChaseRainbow(50);
}
// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
 for(uint16_t i=0; i<strip.numPixels(); i++) {</pre>
     strip.setPixelColor(i, c);
     strip.show();
     delay(wait);
 }
}
```

(continues on next page)

```
void rainbow(uint8_t wait) {
 uint16_t i, j;
  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {</pre>
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
 }
}
// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
 uint16_t i, j;
  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel</pre>
    for(i=0; i< strip.numPixels(); i++) {</pre>
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
 }
}
//Theatre-style crawling lights.
void theaterChase(uint32_t c, uint8_t wait) {
  for (int j=0; j<10; j++) { //do 10 cycles of chasing</pre>
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
        strip.setPixelColor(i+q, c); //turn every third pixel on
      }
      strip.show();
      delay(wait);
      for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
        strip.setPixelColor(i+q, 0);
                                       //turn every third pixel off
      }
    }
 }
}
//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow(uint8_t wait) {
  for (int j=0; j < 256; j++) {
                                   // cycle all 256 colors in the wheel
    for (int q=0; q < 3; q++) {
        for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
          strip.setPixelColor(i+q, Wheel( (i+j) % 255)); //turn every third pixel on
        }
        strip.show();
```

(continues on next page)

(continued from previous page)

```
delay(wait);
       for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
        strip.setPixelColor(i+q, ∅);
                                      //turn every third pixel off
       }
   }
 }
}
// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
 if(WheelPos < 85) {</pre>
  return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
 } else if(WheelPos < 170) {</pre>
  WheelPos -= 85;
  return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
 } else {
  WheelPos -= 170;
  return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
 }
}
```

#### **Test Code**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Then we can see the four RGB LED are lighting in various colors.

## 6.2.33 Project 33: Rotary Encoder



#### Overview

In this kit, there is a Keyestudio rotary encoder, dubbed as switch encoder. It is applied to automotive electronics, multimedia audio, instrumentation, household appliances, smart home, medical equipment and so on.

In the experiment, it is used for counting. When we rotate the rotary encoder clockwise, the set data adds 1; if you rotate it anticlockwise, the set data substructs1; and when the middle button is pressed, the value will be show in the serial monitor.

#### **Working Principle**



The incremental encoder converts the displacement into a periodic electric signal, and then converts this signal into a counting pulse, and the number of pulses indicates the size of the displacement.

This module mainly uses 20-pulse rotary encoder components. It can calculate the number of pulses output during clockwise and reverse rotation. There is no limit to count rotation. It resets to the initial state, that is, starts counting from 0.

## Components

		Forary seconder		
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Rotary Encoder*1	5P Dupont Wire*1	Micro USB Cable*1

## **Connection Diagram**



fritzing

## **Test Code**

```
/*
* Filename : Encoder
* Description : Rotary encoder module counting.
* Auther : http//www.keyestudio.com
*/
//Interfacing Rotary Encoder with Arduino
//Encoder Switch -> pin 27
//Encoder DT -> pin 14
//Encoder CLK -> pin 12
int Encoder_DT = 14;
int Encoder_CLK = 12;
int Encoder_Switch = 27;
int Previous_Output;
int Encoder_Count;
void setup() {
 Serial.begin(9600);
 //pin Mode declaration
 pinMode (Encoder_DT, INPUT);
 pinMode (Encoder_CLK, INPUT);
 pinMode (Encoder_Switch, INPUT);
 Previous_Output = digitalRead(Encoder_DT); //Read the inital value of Output A
}
void loop() {
 //aVal = digitalRead(pinA);
 if (digitalRead(Encoder_DT) != Previous_Output)
```

(continues on next page)

(continued from previous page)

```
{
   if (digitalRead(Encoder_CLK) != Previous_Output)
   {
     Encoder_Count ++;
     Serial.println(Encoder_Count);
   }
   else
   {
     Encoder_Count--;
     Serial.println(Encoder_Count);
   }
 }
 Previous_Output = digitalRead(Encoder_DT);
 if (digitalRead(Encoder_Switch) == 0)
 {
   delay(5);
   if (digitalRead(Encoder_Switch) == 0) {
     Serial.println("Switch pressed");
     while (digitalRead(Encoder_Switch) == 0);
   }
 }
}
```

#### **Code Explanation**

Set CLK to GPIO12 and DAT to GPIO14.

This code is set well in the library file. When CLK descends, read the voltage of DAT, when DAT is a HIGH level, the value of the rotary encoder is added by 1; when DAT is a LOW level, the value of the rotary encoder is cut down 1.

Set the pin of the button(GPIO27) to LOW and print.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600;

Rotate the knob on the rotary encoder clockwise, the displayed data will increase; on the contrary, in anticlockwise way, the data will decrease. Equally, press the button on the rotary encoder, "Switch pressed" will be shown.

COM3		_		$\times$
				Send
-2				~
-1				
0				
1				
2				
3				
4				
5				
4				
3				
2				
1				
0				
-1				
-2				
-3				
Switch pressed				
				×
Autoscroll 🗌 Show timestamp	Newline $\sim$	9600 baud $\sim$	Clear	r output

# 6.2.34 Project 34: Servo Control



## Overview

Servo motor is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor.

In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.

#### **Working Principle**



The rotation angle of servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms (50Hz).

Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle from  $0^{\circ}$  to  $180^{\circ}$ . But note that for different brand motors, the same signal may have different rotation angles.





### Components

	LILLING CONTRACTOR CON		
ESP32 Board*1	ESP32 Expansion Board*1	Servo*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

Test Code 1

```
/*
* Filename : Servo_1
* Description : Steering gear rotation Angle 0-90-180, repeatly
* Auther
           : http//www.keyestudio.com
*/
int servoPin = 4;//steering gear PIN
void setup() {
 pinMode(servoPin, OUTPUT);//steering pin is set to output
}
void loop() {
 servopulse(servoPin, 0);//Rotate it to zero degrees
 delay(1000);//delay 1S
 servopulse(servoPin, 90);//Rotate it to 90 degrees
 delay(1000);
 servopulse(servoPin, 180);//Rotate it to 180 degrees
 delay(1000);
}
void servopulse(int pin, int myangle) { //Impulse function
 int pulsewidth = map(myangle, 0, 180, 500, 2500); //Map Angle to pulse width
 for (int i = 0; i < 10; i++) { //Output a few more pulses</pre>
   digitalWrite(pin, HIGH);//Set the steering gear interface level to high
   delayMicroseconds(pulsewidth);//The number of microseconds of delayed pulse width_
\rightarrow value
   digitalWrite(pin, LOW);//Lower the level of steering gear interface
   delay(20 - pulsewidth / 1000);
 }
}
```

## **Code Explanation 1**

map(value, fromLow, fromHigh, toLow, toHigh)

Value is the value we map. fromLow, fromHigh is the maximum and minimum value, toLow, toHigh are the upper limit and lower limit we map.

For example, map(myangle, 0, 180, 500, 2500) means that an angle value myangle (0°-180° the mapping range is from 500us to 2500us.

We use the function **servopulse()** to make the servo move. We also make the servo rotate 0°, 90° and 180° cyclically.

#### Test Result 1

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, the servo will rotate  $0^{\circ}90^{\circ}$  and  $180^{\circ}$  cyclically.

#### Test Code 2

```
/*
* Filename : Servo Sweep
* Description : Control the servo motor for sweeping
* Auther : http//www.keyestudio.com
*/
#include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int posVal = 0; // variable to store the servo position
int servoPin = 4; // Servo motor pin
void setup() {
 myservo.setPeriodHertz(50);
                           // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→object
}
void loop() {
 for (posVal = 0; posVal <= 180; posVal += 1) { // goes from 0 degrees to 180 degrees
   // in steps of 1 degree
                         // tell servo to go to position in variable 'pos'
   myservo.write(posVal);
                           // waits 15ms for the servo to reach the position
   delay(15);
 }
 for (posVal = 180; posVal >= 0; posVal -= 1) { // goes from 180 degrees to 0 degrees
   myservo.write(posVal); // tell servo to go to position in variable 'pos'
   delay(15);
                           // waits 15ms for the servo to reach the position
 }
}
```

#### **Code Explanation 2**

myservo. write (pos) is the rotation angle to POS. myservo. read () reads the current angle value of the servo.

#### **Test Result 2**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, the servo will rotate from  $0^{\circ}$  to  $180^{\circ}$  by moving  $1^{\circ}$  for each 15ms.



# 6.2.35 Project 35: Ultrasonic Sensor

Bats and some marine animals are able to use high frequencies of sound for echolocation or communication. They can emit ultrasonic waves from the larynx through the mouth or nose and use the sound waves that bounce back to orient and determine the position, size and whether nearby objects are moving.

Ultrasonic is a frequency higher than 20000 Hz sound wave, which has a good direction, a strong penetration ability, and is easy to obtain more concentrated sound energy as well as spread far in the water. It can be used for ranging, speed measurement, cleaning, welding, gravel, sterilization and disinfection. What's more, it has many applications in medicine, military, industry and agriculture.

## Overview

In this kit, there is a keyes HC-SR04 ultrasonic sensor, which can detect obstacles in front and the detailed distance between the sensor and the obstacle. Its principle is the same as that of bat flying. It can emit the ultrasonic signals that cannot be heard by humans. When these signals hit an obstacle and come back immediately. The distance between the sensor and the obstacle can be calculated by the time gap of emitting signals and receiving signals.

In the experiment, we use the sensor to detect the distance between the sensor and the obstacle, and print the test result.

#### **Working Principle**

The most common ultrasonic ranging method is the echo detection. As shown below; when the ultrasonic emitter emits the ultrasonic waves towards certain direction, the counter will count. The ultrasonic waves travel and reflect back once encountering the obstacle. Then the counter will stop counting when the receiver receives the ultrasonic waves coming back.

The ultrasonic wave is also sound wave, and its speed of sound V is related to temperature. Generally, it travels 340m/s in the air. According to time t, we can calculate the distance s from the emitting spot to the obstacle. s = 340t/2 The HC-SR04 ultrasonic ranging module can provide a non-contact distance sensing function of 2cm-400cm, and the

ranging accuracy can reach as high as 3mm; the module includes an ultrasonic transmitter, receiver and control circuit. Basic working principle:

1). First pull down the TRIG, and then trigger it with at least 10us high level signal;

2). After triggering, the module will automatically transmit eight 40KHZ square waves, and automatically detect whether there is a signal to return.

3). If there is a signal returned back, through the ECHO to output a high level, the duration time of high level is actually the time from emission to reception of ultrasonic.

TestDistance = HighLevelDuration \* 340m/s \* 0.5



## Components

			3	
ESP32 Board*1	ESP32 Expansion Board*1	keyestudio SR01 Ultrasonic Sensor*1	4P Dupont Wire*1	MicroUSB Cable*1

#### **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename
            : Ultrasonic
* Description : Use the ultrasonic module to measure the distance.
           : http//www.keyestudio.com
* Auther
*/
const int TrigPin = 13; // define TrigPin
const int EchoPin = 14; // define EchoPin.
int duration = 0; // Define the initial value of the duration to be 0
int distance = 0;//Define the initial value of the distance to be 0
void setup()
{
 pinMode(TrigPin , OUTPUT); // set trigPin to output mode
 pinMode(EchoPin , INPUT); // set echoPin to input mode
 Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
}
void loop()
{
// make trigPin output high level lasting for 10s to triger HC_SR04
 digitalWrite(TrigPin , HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin , LOW);
 // Wait HC-SR04 returning to the high level and measure out this waitting time
 duration = pulseIn(EchoPin , HIGH);
 // calculate the distance according to the time
 distance = (duration/2) / 28.5;
 Serial.print("Distance: ");
 Serial.print(distance); //Serial port print distance value
 Serial.println("cm");
 delay(300); // Wait 100ms between pings (about 20 pings/sec).
}
             ******
```

**Test Result** 

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600.

The serial monitor will print the distance between the ultrasonic sensor and the object.

💿 СОМЗ	- 🗆 ×
	Send
Distance. Jem	
Distance: 3cm	
Distance: 4cm	
Distance: 4cm	
Distance: 6cm	
Distance: 6cm	
Distance: 8cm	
Distance: 7cm	
Distance: 7cm	
Distance: 8cm	
Distance: 9cm	
Distance: 9cm	
Distance: 53cm	
Distance: 55cm	
Distance: 10cm	
	~
Autoscroll Show timestamp Newline	$\sim$ 9600 baud $\sim$ Clear output



## 6.2.36 Project 36: IR Receiver Module

#### Overview

Infrared remote control is currently the most widely used means of communication and remote control, which has the characteristics of small volume, low power consumption, strong function and low cost. Therefore, recorder, audio equipment, air conditioning machine and toys and other small electrical devices have also used the infrared remote control.

Its transmitting circuit is the use of infrared light emitting diode to emit modulated infrared light wave. The circuit is composed of infrared receiving diode, triode or silicon photocell. They convert infrared light emitted by infrared emitter into corresponding electrical signal, and then send back amplifier.

In this experiment, we need to know how to use the infrared receiving sensor. The infrared receiving sensor mainly uses the VS1838B infrared receiving sensor element. It integrates receiving, amplifying, and demodulating. The internal IC has already completed the demodulation, and the output is a digital signal. It can receive 38KHz modulated remote control signal.

In the experiment, we use the IR receiver to receive the infrared signal emitted by the external infrared transmitting device, and display the received signal in the shell.

#### Working Principle

The main part of the IR remote control system is modulation, transmission and reception. The modulated carrier frequency is generally between 30khz and 60khz, and most of them use a square wave of 38kHz and a duty ratio of 1/3. A 4.7K pull-up resistor R3 is added to the signal end of the infrared receiver.



Components

ESP32 Board*11	ESP32 Expansion Board*1	DIY IR Receiver*1
3P Dupont Wire*1	Micro USB Cable*1	Remote Control*1

**Connection Diagram** 



fritzing

**Test Code** 

```
/*
* Filename : IR Receiver
* Description : Decode the infrared remote control and print it out through the serial.
\rightarrow port.
* Auther
           : http//www.keyestudio.com
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
const uint16_t recvPin = 15; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
                       // Create a decoding results class object
decode_results results;
void setup() {
                      // Initialize the serial port and set the baud rate to 9600
 Serial.begin(9600);
                        // Start the receiver
 irrecv.enableIRIn();
 Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
 Serial.println(recvPin); //print the infrared receiving pin
}
void loop() {
 if (irrecv.decode(&results)) { // Waiting for decoding
   serialPrintUint64(results.value, HEX);// Print out the decoded results
   Serial.println("");
   irrecv.resume();
                                   // Release the IRremote. Receive the next value
 }
 delay(1000);
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32.

After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set baud rate to 9600; Find the infrared remote control, pull out the insulating sheet, and press the button at the receiving head of the infrared receiving sensor. After receiving the signal, the LED on the infrared receiving sensor also starts to flash, as shown in the figure below.

💿 СОМЗ			_		$\times$
					Send
FF629D					~
FF22DD					
FF22DD					
FFA857					
FFC23D					
FF629D					
FF6897					
FF9867					
FFB04F					
FF30CF					
986E09D					
FF7A85					
FF10EF					
FF38C7					
FF5AA5					
FF42BD					
FF4AB5					
FF52AD					~
Autoscroll 🗌 Show timestamp	Newline	$\sim$ 9600 baud	$\sim$	Clear	output

Write down the key code value associated with the infrared remote with each key, as you will need this information later.





## 6.2.37 Project 37: DS18B20 Temperature Sensor

### Description

In this kit, there is a DS18B20 temperature sensor, which is from maxim. The MCU can communicate with the DS18B20 through 1-Wire protocol, and finally read the temperature.

In this experiment, we will use this temperature sensor to measure the temperature in the current environment. The test result is  $^{\circ}$ C, ranging from -55 $^{\circ}$ C to +125 $^{\circ}$ C. We will display the test result on shell.

#### **Working Principle**



The hardware interface of the 1-Wire bus is very simple, just connect the data pin of the DS18B20 to an IO port of the microcontroller. The timing of the 1-Wire bus is relatively complex. Many students can't understand the timing diagram independently here. We have encapsulated the complex timing operations in the library, and you can use the library functions directly.

## Schematic Diagram of DS18B20

This can save up to 12-bit temperature vale. In the register, save in code complement. As shown below;



A total of 2 bytes, LSB is the low byte, MSB is the high byte, where MSb is the high byte of the byte, LSb is the low byte of the byte. As you can see, the binary number, the meaning of the temperature represented by each bit, is expressed. Among them, S represents the sign bit, and the lower 11 bits are all powers of 2, which are used to represent the final temperature. The temperature measurement range of DS18B20 is from -55 degrees to +125 degrees, and the expression form of temperature data, S represents positive and negative temperature, and the resolution is 2, which is 0.0625.

#### **Required Components**

		Lillon Dim Sampersture		
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY 18B20 Temperature Sensor*1	3P Dupont Wire*1	MicroUSB Cable*1

#### **Required Components**



fritzing

**Test Code** 



(continues on next page)

```
(continued from previous page)
```

#### **Code Explanation**

We set the pin to GPIO15 and obtain the temperature in the unit of °C.

Set a double decimal variable to temp, and assign the measured result to temp.

The serial monitor displays the temp value, and the baud rate needs to be set before displaying (our default setting is 9600, which can be changed).

We add the unit behind the data. If the unit is directly set to  $^{\circ}C$ , the test result will be garbled. So we directly replace  $^{\circ}C$  with C.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600; The monitor will display the temperature of the current environment, as shown below.

💿 СОМЗ			_		$\times$	
					Send	
Temperature. 20.370					_	
Temperature: 28.37C						<u> </u>
Temperature: 28.50C						
Temperature: 28.81C						
Temperature: 29.00C						
Temperature: 29.19C						
Temperature: 29.31C						
Temperature: 29.37C						
Temperature: 29.44C					_	
Temperature: 29.50C						
Temperature: 29.56C						
Temperature: 29.62C						
Temperature: 29.62C						
Temperature: 29.69C						
Temperature: 29.69C						
Temperature: 29.69C						
Temperature: 29.75C						
Temperature: 29.75C						
					_	¥
Autoscroll 🗌 Show timestamp	Newline ~	9600 baud	$\sim$	Clear	outpu	ıt



## 6.2.38 Project 38: XHT11 Temperature and Humidity Sensor

#### Description

This DHT11 temperature and humidity sensor is a composite sensor which contains a calibrated digital signal output of the temperature and humidity.

DHT11 temperature and humidity sensor uses the acquisition technology of the digital module and temperature and humidity sensing technology, ensuring high reliability and excellent long-term stability.

It includes a resistive element and a NTC temperature measuring device.



#### **Working Principle**

The communication and synchronization between the single-chip microcomputer and XHT11 adopts the single bus data format. The communication time is about 4ms. The data is divided into fractional part and integer part.

Operation process: A complete data transmission is 40bit, high bit first out. Data format: 8bit humidity integer data + 8bit humidity decimal data + 8bit temperature integer data + 8bit temperature decimal data + 8bit checksum

8-bit checksum: 8-bit humidity integer data + 8-bit humidity decimal data + 8-bit temperature integer data + 8-bit temperature decimal data "Add the last 8 bits of the result.

### **Required Components**

ESP32Board*1	Keyestudio XHT11 Temperature and Humidity Sensor *1 (compatible with DHT11)		
ESP32 Expansion Board*1	3P Dupont Wire*1	Micro USB Cable*1	

## **Connection Diagram**





**Test Code** 



(continues on next page)

(continued from previous page)

```
*/
#include "xht11.h"
//gpio15
xht11 xht(15);
unsigned char dht[4] = {0, 0, 0, 0}; //Only the first 32 bits of data are received, not
\rightarrow the parity bits
void setup() {
 Serial.begin(9600);//Start the serial port monitor and set baud rate to 9600
}
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [1] is the fractional part
   Serial.print("% ");
   Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
→part
   Serial.println("C");
 } else {
           //Read error
   Serial.println("sensor error");
 }
 delay(1000); //It takes 1000ms to wait for the device to read
}
```

#### **Code Explanation**

1). We set the pin to GPIO15, and store the detected temperature and humidity data in the dht[4] array. 2). We add units behind the data. If the temperature unit is directly set to °C, the test results may be wrong, so we directly replace °C with C; the humidity unit is directly set to %.

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set baud rate to 9600; The monitor will display the temperature and humidity data of the current environment, as shown below.

💿 CON	13			_		$\times$
						Send
Rui. 578	Temp.200					^
RH:62%	Temp:28C					
RH:67%	Temp:28C					
RH:72%	Temp:28C					
RH:74%	Temp:28C					
RH:80%	Temp:28C					
RH:80%	Temp:28C					
RH:87%	Temp:28C					
RH:88%	Temp:29C					
RH:91%	Temp:29C					
RH:92%	Temp:29C					
RH:94%	Temp:29C					
RH:94%	Temp:30C					
RH:95%	Temp:30C					
RH:95%	Temp:30C					
RH:95%	Temp:30C					
RH:95%	Temp:30C					
RH:95%	Temp:31C					
	-					~
Autos	croll 🗌 Show timestamp	Newline $\sim$	9600 baud	$\sim$	Clear	r output



## 6.2.39 Project 39: DS1307 Clock Module

#### Overview

This module mainly uses the real-time clock chip DS1307, which is the I2C bus interface chip that has second, minute, hour, day, month, year and other functions as well as leap year automatic adjustment function introduced by DALLAS. It can work independently of CPU, and won't' affected by the CPU main crystal oscillator and capacitance as well as keep accurate time. What's more, monthly cumulative error is generally less than 10 s. The chip also has a clock protection circuit in case of main power failure and runs on a back-up battery that denies the CPU read and write access.

At the same time, it contains automatic switching control circuit of standby power supply, making it guarantees the accuracy of system clock in case of power failure of main power supply and other bad environment.

Going forward, the DS1307 chip internal integration has a certain capacity, with power failure protection characteristics of static RAM, which can be used to save some key data.

In the experiment, we use the DS1307 clock module to obtain the system time and print the test results.



## **Working Principle**

Serial real-time clock records year, month, day, hour, minute, second and week; AM and PM indicate morning and afternoon respectively; 56 bytes of NVRAM store data; 2-wire serial port; programmable square wave output; power failure detection and automatic switching circuit; battery current is less than 500nA.

Pins description:

X1, X2: 32.768kHz crystal terminal

VBAT: +3V input

SDA: serial data

SCL: serial clock

SQW/OUT: square waves/output drivers

#### Components

			3	
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DS1307 Clock Module*1	4P Dupont Wire*1	MicroUSB Cable*1

## **Connection Diagram**



fritzing

## **Test Code**



\_\_\_\_\_

(continued from previous page)

```
Serial.print(" ");
Serial.println(Rtc.GetDateTime().DayOfWeek());
delay(1000);//Delay 1 second
```

### 

#### **Code Explanation**

}

Rtc.GetDateTime(): the obtained current time and date.

Rtc.Begin(): enable DS1307 real-time clock.

Rtc.SetIsRunning(true): run the DS1307 real-time clock, if true changes into false, time will stop.

Rtc.SetDateTime(): set time.

Rtc.GetDateTime().Year(): return year.

Rtc.GetDateTime().Month(): return month.

Rtc.GetDateTime().Day(): return date.

Rtc.GetDateTime().Hour(): return hour.

Rtc.GetDateTime().Minute(): return minute.

Rtc.GetDateTime().Second(): return second.

Rtc.GetDateTime().DayOfWeek(): return week.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, attach the DS1307 sensor to a battery, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on.

Open the serial monitor and set baud rate to 57600. We can see the displayed year, month, day, hour, minute, second and week on the serial monitor, and set the time and date to refresh every second, as shown below:

💿 COM3			_		$\times$
					Send
2021/12/0	11.37.33	-			~
2021/12/9	11:37:36	4			~
2021/12/9	11:37:37	4			
2021/12/9	11:37:38	4			
2021/12/9	11:37:39	4			
2021/12/9	11:37:40	4			
2021/12/9	11:37:41	4			
2021/12/9	11:37:42	4			
2021/12/9	11:37:43	4			
2021/12/9	11:37:44	4			
2021/12/9	11:37:45	4			
2021/12/9	11:37:46	4			
2021/12/9	11:37:47	4			
2021/12/9	11:37:48	4			
2021/12/9	11:37:49	4			
2021/12/9	11:37:50	4			
2021/12/9	11:37:51	4			
2021/12/9	11:37:52	4			
					¥
Autoscroll	Show times	stamp	Newline $\sim$ 57600 baud $\sim$	Clea	r output

# 6.2.40 Project 40: ADXL345 Acceleration Sensor



### Overview

In this kit, there is a DIY electronic building block ADXL345 acceleration sensor module, which uses the ADXL345BCCZ chip. The chip is a small, thin, low-power 3-axis accelerometer with a high resolution (13 bits) and a measurement range of  $\pm 16g$  that can measure both dynamic acceleration due to motion or impact as well as stationary acceleration such as gravitational acceleration, making the device usable as a tilt sensor.

#### **Working Principle**

The ADXL345 is a complete 3-axis acceleration measurement system with a selection of measurement ranges of  $\pm 2$  g,  $\pm 4$  g,  $\pm 8$  g or  $\pm 16$  g. Its digital output data is in 16-bit binary complement format and can be accessed through an SPI (3-wire or 4-wire) or I2C digital interface.

The sensor can measure static acceleration due to gravity in tilt detection applications, as well as dynamic acceleration due to motion or impact. Its high resolution (3.9mg/LSB) enables measurement of tilt Angle changes of less than  $1.0^{\circ}$ .



#### **Components Required**

		ADXL345	3	
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio ADXL345 Acceleration Module*1	4P Dupont Wire*1	MicroUSBCable*1

#### **Connection Diagram**



fritzing

#### **Test Code**

```
/*
* Filename
           : ADXL345
* Description : Read the X/Y/Z value of ADXL345
* Auther
           : http//www.keyestudio.com
*/
#include "adx1345_io.h"
//The port is sda-->21,scl-->22
adx1345 adx1345(21, 22);
float out_X, out_Y, out_Z;
void setup() {
 Serial.begin(57600);//Start serial port monitoring and set baud rate to 57600
 adxl345.Init();
}
void loop() {
 adx1345.readXYZ(&out_X, &out_Y, &out_Z);
 Serial.print(out_X);
 Serial.print("g
               ");
 Serial.print(out_Y);
 Serial.print("g
               ");
 Serial.print(out_Z);
 Serial.println("g");
 delay(100);
}
//*******
```

#### **Code Explanation**

Set 3 decimal variables out\_X out\_Y out\_Z, and assign the measured result to out\_X out\_Y out\_Z. The serial monitor displays the value of out\_X out\_Y out\_Z, and the baud rate needs to be set before displaying (our default setting is 57600, which can be changed).
Adxl345.Init; Initialize the ADXX345 accelerometer.

**adxl345.readXYZ(&out\_X, &out\_Y, &out\_Z);** Get the acceleration value of the X axis and return it to the variables out\_X, out\_Y, out\_Z.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set baud rate to 57600.

The serial monitor displays the value corresponding to the sensor, the unit is mg, as shown in the figure below.

💿 сома	}				_		$\times$
							Send
00.009	00.009	200.00g					^
-107.00g	14.00g	241.00g					
-95.00g	23.00g	218.00g					
-81.00g	-104.00g	214.00g					
-28.00g	-159.00g	143.00g					
15.00g	-261.00g	198.00g					
106.00g	-119.00g	231.00g					
128.00g	11.00g	265.00g					
-57.00g	66.00g	233.00g					
-45.00g	82.00g	270.00g					
-61.00g	111.00g	209.00g					
-58.00g	190.00g	211.00g					
-225.00g	30.00g	95.00g					
-171.00g	-84.00g	-27.00g					
-89.00g	-175.00g	-17.00g					
10.00g	-180.00g	27.00g					
-8.00g	-229.00g	92.00g					
							$\sim$
<							>
Autosc	roll 🗌 Show	v timestamp	Newline ~	57600	baud 🗸	Clear	r output

# 6.2.41 Project 41: TM1650 4-Digit Tube Display



#### Overview

This module is mainly composed of a 0.36 inch red common cathode 4-digit digital tube, and its driver chip is TM1650. When using it, we only need two signal lines to make the single-chip microcomputer control a 4-bitdigit tube, which greatly saves the IO port resources of the control board.

TM1650 is a special circuit for LED (light emitting diode display) drive control. It integrates MCU input and output control digital interface, data latch, LED drivers, keyboard scanning, brightness adjustment and other circuits.

TM1650 has stable performance, reliable quality and strong anti-interference ability.

It can be applied to the application of long-term continuous working for 24 hours.

TM1650 uses 2-wire serial transmission protocol for communication (note that this data transmission protocol is not a standard I2C protocol). The chip can drive the digital tube and save MCU pin resources through two pins and MCU communication.

### **Working Principle**

TM1650 adopts IIC treaty, which uses DIO and CLK buses.



Data command setting: 0x48 means that we light up the digital tube, instead of enable the function of key scanning

B7	B6	B5	Β4	В3	B2	B1	во	Function	Description
$\times$	0	0	0		×	×			Eight-level brightness
$\times$	0	0	1		×	×			One-level brightness
$\times$	0	1	0		×	×			Two-level brightness
$\times$	0	1	1		×	×		Duichte and active	Three-level brightness
$\times$	1	0	0		×	×		Brightness setting	Four-level brightness
$\times$	1	0	1		×	×			Five-level brightness
$\times$	1	1	0		×	×			Six-level brightness
$\times$	1	1	1		×	×			Seven-level brightness
$\times$				0	×	×		7/8 segment	8-segment display way
$\times$				1	×	×		display control bit	7-segment display way
$\times$					×	×	0	ON/OFF display bit	Off display
$\times$					×	×	1	ON/OFF display bit	On display

### **Command display setting:**

bit[6:4]: set the brightness of tube display, and 000 is brightest

bit[3]: set to show decimal points

bit[0]: start the display of the tube display

Components

		CEEB Control Control C	3	
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio TM16504-Digit Segment Display*1	4P Dupont Wire*1	Micro USB Cable*1

#### **Connection Diagram**



fritzing

### **Test Code**

(continues on next page)

(continued from previous page)

```
for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                                //DigitalTube.clearBit(0 to 3); Clear bit display.
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0);
                               //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>→</u>--9
}
void loop(){
 for(int num=0; num<10000; num++){</pre>
   displayFloatNum(num);
   delay(100);
 }
}
void displayFloatNum(float num){
 if(num > 9999)
   return:
 int dat = num*10;
  //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return:
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return;
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The 4-digit tube display will show integer from 0 to 99999, add 1 for each 10ms. Increase to 9999 then start from 0.



## 6.2.42 Project 42: HT16K33 8X8 Dot Matrix Module

### Overview

What is the dot matrix display? If we apply the previous circuit, there will be must one IO port to control only one LED. When more LED need to be controlled, we may adopt a dot matrix. The 8X8 dot matrix is composed of 64 light-emitting diodes, and each light-emitting diode is placed at the intersection of the row line and the column line. Refer to the experimental schematic diagram below, when the corresponding column is set to a high level and a certain row to low, the corresponding diode will light up. For instance, set pin 13 to a high level and pin 9 to low, and then the first LED will light up. In the experiment, we display icons via this dot matrix.

### **Working Principle**

As the schematic diagram shown, to light up the LED at the first row and column, we only need to set C1 to high level and R1 to low level. To turn on LEDs at the first row, we set R1 to low level and C1-C8 to high level.

16 IO ports are needed, which will highly waste the MCU resources.

Therefore, we designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, which greatly saves the resources of the single-chip microcomputer.



There are three DIP switches on the module, all of which are set to I2C communication address. The setting method is shown below. A0A1 and A2 are grounded, that is, the address is 0x70.

A0 (1)	A1 (2)	A2 (3)	0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	0 (OFF)
	0X70			0X71			0X72	
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
1 (ON)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	1 (ON)	0 (OFF)	1 (ON)
	OX73			0X74			0X75	
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)			
0 (OFF)	1 (ON)							

### Components

			~	
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio HT16K33_ 8X8 Dot Matrix*1	4P Dupont Wire*1	Micro USB Cable*1

### **Connection Diagram**



fritzing

### **Test Code**

(continues on next page)

(continued from previous page)

```
byte result[8][8];
byte test1[8] = {0x00,0x42,0x41,0x09,0x09,0x41,0x42,0x00};
void setup()
{
 matrix.init(0x70, SDA, SCL);//Initialize matrix
 matrix.showLedMatrix(test1,0,0);
 matrix.show();
}
void loop()
{
 for (int i = 0; i <= 7; i++)
 {
   matrix.setBrightness(i);
   delay(100);
 }
 for (int i = 7; i > 0; i--)
 {
   matrix.setBrightness(i);
   delay(100);
 }
}
```

### **Code Explanation**

First we need to import the library file.

The pattern in our code is an array of byte data type, which is shown in the table below.

We convert  $\{0x00,0x42,0x41,0x09,0x09,0x41,0x42,0x00\}$  into binary, and fill in the 8\*8 form below to make it clear. 1 means on, 0 means off. Then we can see that it is a smile shape.

0	0	0	0	0	0	0	0
o	1	1	0	0	1	1	0
o	0	0	0	0	0	0	0
о	0	0	0	0	0	0	0
o	0	0	1	1	0	0	0
о	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Then the dot matrix displays a "smile "pattern.

# 6.2.43 Project 43: LCD\_128X32\_DOT Module



### Description

This is a 128\*32 pixel LCD module, which uses IIC communication mode and ST7567A driver chip. At the same time, the code contains all the English letters and common symbols of the library that can be directly called. When used, we can also set English letters and symbols to display different text sizes in our code.

To make it easy to set up the pattern display, we also provide a mold capture software that can convert a specific pattern into control code and then copy it directly into the test code for use.

In the experiment, we will set up the display screen to display various English words, common symbols and numbers.

### **Working Principle**



The module uses the IIC communication principle, the underlying functions have been encapsulated in the library surface, we can directly call the library function, if interested, you can also go to understand the underlying driver of the module.

### Components



**Connection Diagram** 



### **Test Code**

```
/*
* Filename : lcd128*32
* Description : Lcd128 *32 Displays character strings
* Auther : http//www.keyestudio.com
*/
#include "lcd128_32_io.h"
//Create lcd12832 examples,sda--->21 scl--->22
lcd lcd(21, 22);
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //cls
}
void loop() {
 lcd.Cursor(0, 7); //Set display position
 lcd.Display("KEYES"); //Setting the display
 lcd.Cursor(1, ♥);
 lcd.Display("ABCDEFGHIJKLMNOPQR");
 lcd.Cursor(2, ∅);
 lcd.Display("123456789+-*/<>=$@");
 lcd.Cursor(3, ∅);
 lcd.Display("%^&(){}:;'|?,.~\\[]");
}
```

### **Code Explanation**

First import the library file 1.Init() initializes the display screen; .Clear() clears the display; .Cursor() sets the display position; .Display() displays characters.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The first line of the 128X32LCD module display shows "KEYES", the second line shows "ABCDEFGHIJKLMNOPQR", and the third line shows "123456789±\*/<> =\$@", the fourth line displays " $%^{0}[{;;']}?, ~[]$ ", as shown in the following figure:

## 6.2.44 Project 44: RFID Module



#### Description

RFIDRFID-RC522 radio frequency module adopts a Philips MFRC522 original chip to design card reading circuit, easy to use and low cost, suitable for equipment development and card reader development and so on.

RFID or Radio Frequency Identification system consists of two main components, a transponder/tag attached to an object to be identified, and a transceiver also known as interrogator/Reader.

In the experiment, the data read by the card swipe module is 4 hexadecimal numbers, and we print these four hexadecimal numbers as strings. For example, we read the data of the IC card below:  $0 \times ED0 \times F70 \times 940 \times 5A$  and the information string displayed in the serial monitor is ED F7 94 5A; the data read from the keychain is:  $0 \times 4C0 \times 090 \times 6B0 \times 6E$ . Different IC cards and different key chains have diverse data.

### **Working Principle**

Radio frequency identification, the card reader is composed of a radio frequency module and a high-level magnetic field. The Tag transponder is a sensing device, and this device does not contain a battery. It only contains tiny integrated circuit chips and media for storing data and antennas for receiving and transmitting signals. To read the data in the tag, first put it into the reading range of the card reader. The reader will generate a magnetic field, and because the magnetic energy generates electricity according to Lenz's law, the RFID tag will supply power, thereby activating the device.



Micro USB Cable\*1 Key\*1

**Connection Diagram** 



### **Test Code**

```
/*
* Filename : RFID
* Description : RFID reader UID
* Auther : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
void setup() {
 Serial.begin(115200); // initialize and PC's serial communication
                              // initialize I2C
 Wire.begin();
 mfrc522.PCD_Init();
                               // initialize MFRC522
                              // dispaly PCD - MFRC522 read carder
 ShowReaderDetails();
 Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
}
void loop() {
 //
 if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return;
 }
 // select one of door cards. UID and SAK are mfrc522.uid.
 // save UID
 Serial.print(F("Card UID:"));
 for (byte i = 0; i < mfrc522.uid.size; i++) {</pre>
   Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   Serial.print(mfrc522.uid.uidByte[i], HEX);
 }
 Serial.println();
}
void ShowReaderDetails() {
 // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print(F("MFRC522 Software Version: 0x"));
 Serial.print(v, HEX);
 if (v == 0x91)
   Serial.print(F(" = v1.0"));
 else if (v == 0x92)
   Serial.print(F(" = v2.0"));
 else
   Serial.print(F(" (unknown)"));
 Serial.println("");
 // when returning to 0x00 or 0xFF, may fail to transmit communication signals
```

(continues on next page)

(continued from previous page)

#### **Code Explanation**

Wire.begin(); The module we use is the IIC interface, so we first initialize the IIC

mfrc522.PCD\_Init(); initialize MFRC522

String(mfrc522.uid.uidByte[i], HEX); A string to convert the value read into hexadecimal format.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 115200. When we make the IC card close to the RFID module, the information will be printed out, as shown in the figure below.

🐵 COM3 —		
	Send	
Card UID: ED F7 94 5A		^
Card UID: ED F7 94 5A		
Card UID: ED F7 94 5A		
Card UID: ED F7 94 5A		
Card UID: ED F7 94 5A		
Card UID: 4C 09 6B 6E		
Card UID: 4C 09 6B 6E		
Card UID: 4C 09 6B 6E		
Card UID: 4C 09 6B 6E		
		$\mathbf{v}$
🗸 Autoscroll 🗌 Show timestamp 🛛 Newline 🗸 115200 baud 🗸	Clear outp	ut

Note: Different RFID-RC522 door cards and key chains have diverse values.

# 6.3 3. Comprehensive Experiments:

The previous projects are related to single sensor or module. In the following part, we will combine various sensors and modules to create some comprehensive experiments to perform special functions.

## 6.3.1 Project 45: Button-controlled LED



### Overview

In this lesson, we will make an extension experiment with a button and an LED. When the button is pressed and low levels are output, the LED will light up; when the button is released, the LED will go off. Then we can control a module with another module.

### Components

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Purple LED Module*1
Keyestudio DIY Button Module*1	3P Dupont Wire*2	Micro USB Cable*1

#### **Connection Diagram**



#### **Test Code**

```
/*
* Filename : button_control_LED
* Description : Make a table lamp.
* Auther : http//www.keyestudio.com
*/
#define PIN_LED
                4
#define PIN_BUTTON 15
bool ledState = false;
void setup() {
 // initialize digital pin PIN_LED as an output.
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_BUTTON, INPUT);
}
// the loop function runs over and over again forever
void loop() {
 if (digitalRead(PIN_BUTTON) == LOW) {
   delay(20);
   if (digitalRead(PIN_BUTTON) == LOW) {
     reverseGPIO(PIN_LED);
   }
   while (digitalRead(PIN_BUTTON) == LOW);
 }
}
```

(continues on next page)

(continued from previous page)

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. When the button is pressed, the LED will light up; when pressed again, the LED will go off.

# 6.3.2 Project 46: Alarm Experiment



### Overview

In the previous experiment, we control an output module though an input module. In this lesson, we will make an experiment that the active buzzer will emit sounds once an obstacle appears.

### Components



### **Connection Diagram**



**Test Code** 

```
* Filename
          : Avoiding alarm
* Description : Obstacle avoidance sensor controls the buzzer
         : http//www.keyestudio.com
* Auther
*/
int item = 0;
void setup() {
 pinMode(15, INPUT); //Obstacle avoidance sensor is connected to GPI015 and set to.
→input mode
 pinMode(4, OUTPUT); //The buzzer is connected to GPI04 and set to output mode
}
void loop() {
 item = digitalRead(15);//Read the level value output by the obstacle avoidance sensor
 if (item == 0) {//Obstruction detected
   digitalWrite(4, HIGH);//The buzzer sounded
 } else {//No obstacles detected
   digitalWrite(4, LOW);//The buzzer is off
 3
 delay(100);//Delay 100ms
}
 *****************
```

### **Code Explanation**

Set IO ports according to connection diagram then configure pins mode.

The value is 0 when pressing the button, So, we can determine the key value(0through if (item == 0) and make the buzzer beep digitalWrite(4, HIGH).

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. If the obstacle is detected, the active buzzer will chime; if not, it won't beep.

# 6.3.3 Project 47: Intrusion Detection



## Description

In this experiment, we use a PIR motion sensor to control an active buzzer to emit sounds and the onboard LED to flash rapidly.

### **Required Components**

		V V V V V V V V V V V V V V V V V V V	
ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio DIY PIR Motion Sensor*1	Keyestudio DIY Active Buzzer*1
Keyestudio Purple LED Module*1	3P Dupont Wire*3	Micro USB Cable*1	

### **Connection Diagram**



### **Test Code**

```
/*
* Filename
            : PIR alarm
* Description : PIR control buzzer
* Auther
        : http//www.keyestudio.com
*/
int item = 0;
void setup() {
 pinMode(15, INPUT); //PIR motion sensor is connected to GPI015 and set as the input_
→mode
 pinMode(4, OUTPUT);//The active buzzer is connected to GPI04 and set to output mode
 pinMode(22, OUTPUT);//LED is connected to GPI022 and set to output mode
}
void loop() {
 item = digitalRead(15);//Read digital level signal output by infrared pyrorelease.
⇔sensor
 if (item == 1) { //Movement detected
   digitalWrite(4, HIGH); //Turn on the buzzer
   digitalWrite(22, HIGH); //Turn on the LED
   delay(200);//Delay 200ms
   digitalWrite(4, LOW); //Turn off the buzzer
   digitalWrite(22, LOW); //Turn off the LED
   delay(200);//Delay 200ms
 } else { //No detection
   digitalWrite(4, LOW); //Turn off the buzzer
   digitalWrite(22, LOW); //Turn off the LED
 }
}
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. If the sensor detects people moving, the buzzer will emit an alarm, and the LED will flash continuously.

# 6.3.4 Project 48: Extinguishing Robot



### Description

Today we will use Arduino simulation to build an extinguishing robot that will automatically sense the fire and start the fan. In this project we will learn how to build a very simple robot using ESP32, (detecting flames with a flame sensor, blowing out candles with a fan) can teach us basic concepts about robotics. Once you understand the basics below, you can build more complex robots.

### **Components Required**

ESP32 Board*1	ESP32 Expansion Board*1	130 Motor*1
3P Dupont Wire*1	4P Dupont Wire*1	Micro USB Cable*1
+ STOP OF		
Battery(provide for yourself)*6	Flame Sensor*1	Battery Holder*1

### **Connection Diagram**



**Test Code** 

```
/*
* Filename
          : Fire-fighting robot
* Description : Flame sensor controls the 130 fan module
* Auther
           : http//www.keyestudio.com
*/
int item = 0;
void setup() {
 Serial.begin(9600);
 pinMode(15, OUTPUT);//INA corresponds to IN+, and sets GPI015 to output mode
 pinMode(4, OUTPUT);//INB corresponds to IN-, and sets GPI04 to output mode
}
void loop() {
 item = analogRead(34);//The flame sensor is connected to GPI034, and read the
→ simulated value to Item
 Serial.println(item);//Serial port display analog value
 if (item < 3000) {//Less than 3000
   digitalWrite(15, LOW);//Turn on the fan
   digitalWrite(4, HIGH);
 } else {//Otherwise, turn off the fan.
   digitalWrite(15, LOW);
   digitalWrite(4, LOW);
 }
 delay(100);
}
```

### **Code Explanation**

In the code, we set the threshold value to 3000. When the ADC value detected by the flame sensor is lower than the threshold value, the fan will be automatically turned on; otherwise, it will be turned off. For the driving method of the fan, please refer to the 130 Motor.

### **Test Result**

Connect the wires according to the experimental wiring diagram, switch the DIP switch on the ESP32 expansion board to the ON end and power up, compile and upload the code to the ESP32. After uploading successfully, open the serial monitor and set baud rate to 9600, then the ADC value of the flame will be printed. When this value is less than 3000, the fan will work to blow out the fire, otherwise, it will be turned off. Basically, the ADC value can be set by yourself.

## keyestudio WiKi

💿 СОМ30			_		×
					Send
3218					~
3109					
3027					
2983					
2919					
2917					
2843					
2867					
2859					
2871					
2807					
2923					
2831					
					~
🗹 Autoscroll 🗌 Show timestamp	Newline ~	9600 baud	$\sim$	Clear	output



# 6.3.5 Project 49: Rotary Encoder control RGB

## Introduction

In this lesson, we will control the LED on the RGB module to show different colors through a rotary encoder.

When designing the code, we need to divide the obtained values by 3 to get the remainders. The remainder is 0 and the LED will become red. The remainder is 1, the LED will become green. The remainder is 2, the LED will turn blue.

### Components



### **Connection Diagram**



fritzing

**Test Code** 

```
/*
* Filename : Encoder control RGB
* Description : Rotary encoder controls RGB to present different effects
* Auther : http//www.keyestudio.com
*/
//Interfacing Rotary Encoder with Arduino
//Encoder Switch -> pin 27
//Encoder DT -> pin 14
//Encoder CLK -> pin 12
int Encoder_DT = 14;
int Encoder_CLK = 12;
int Encoder_Switch = 27;
int Previous_Output;
int Encoder_Count;
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
const byte chns[] = {0, 1, 2}; //define the pwm channels
int red, green, blue;
int val;
                                                                (continues on next page)
```

(continued from previous page)

```
void setup() {
  Serial.begin(9600);
  //pin Mode declaration
 pinMode (Encoder_DT, INPUT);
  pinMode (Encoder_CLK, INPUT);
  pinMode (Encoder_Switch, INPUT);
 Previous_Output = digitalRead(Encoder_DT); //Read the inital value of Output A
  for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
  }
}
void loop() {
 //aVal = digitalRead(pinA);
 if (digitalRead(Encoder_DT) != Previous_Output)
  {
   if (digitalRead(Encoder_CLK) != Previous_Output)
   {
      Encoder_Count ++;
      Serial.print(Encoder_Count);
      Serial.print(" ");
     val = Encoder_Count % 3;
     Serial.println(val);
   }
   else
    {
      Encoder_Count--;
      Serial.print(Encoder_Count);
      Serial.print(" ");
     val = Encoder_Count % 3;
      Serial.println(val);
   }
  }
 Previous_Output = digitalRead(Encoder_DT);
 if (digitalRead(Encoder_Switch) == 0)
  {
   delay(5);
   if (digitalRead(Encoder_Switch) == 0) {
      Serial.println("Switch pressed");
     while (digitalRead(Encoder_Switch) == 0);
   }
  }
  if (val == 0) {
   //RED(255, 0, 0)
   ledcWrite(chns[0], 255 );
   ledcWrite(chns[1], 0);
```

(continues on next page)

(continued from previous page)

		<pre>ledcWrite(chns[2],</pre>	0);
	}	<pre>else if (val == 1)</pre>	{
		//GREEN(0, 255, 0)	
		<pre>ledcWrite(chns[0],</pre>	0);
		<pre>ledcWrite(chns[1],</pre>	255);
		<pre>ledcWrite(chns[2],</pre>	0);
	}	else {	
		//BLUE(0, 0, 255)	
		<pre>ledcWrite(chns[0],</pre>	0);
		<pre>ledcWrite(chns[1],</pre>	0);
		<pre>ledcWrite(chns[2],</pre>	255);
	}		
}			
11	*	******	* * * * * * * * * * * * * * * * * * * *

#### **Code Explanation**

1). In the experiment, we set the val to the remainder of Encoder\_Count divided by 3. Encoder\_Count is the value of the encoder. Then we can set pin GPIO0 (red), GPIO2 (green) and GPIO15 (blue) according to remainders.

2). Referring to the control method learned in the previous experiment, use the LED on the remainder control module to display the corresponding light color. The value obtained by taking the remainder of 3 for any number is 0 or 1 or 2. We use these three values to judge, and display the corresponding color.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600, then rotate the knob of the rotary encoder to display the reminders, which can control colors of LED(red green blue).



# 6.3.6 Project 50: Rotary Potentiometer



### Introduction

In the previous courses, we did experiments of breathing light and controlling LED with button. In this course, we do these two experiments by controlling the brightness of LED through an adjustable potentiometer. The brightness of LED is controlled by PWM values, and the range of analog values is 0 to 4095 and the PWM value range is 0-255.

After the code is set successfully, we can control the brightness of the LED on the module by rotating the potentiometer.

### **Required Components**

ESP32 Board*1	ESP32 Expansion Board*1	Keyestudio Purple LED*1
Attentional of the second seco		
Keyestudio Rotary Potentiometer*1	3P Dupont Wire*2	Micro USB Cable*1

### **Connection Diagram**



fritzing

### **Test Code**

(continued from previous page)

### **Code Explanation**

In the experiment, the mapping function maps adcVal from the range of 0-4095 to 0-255, and assigns it to pwmVal.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the module can adjust the brightness of the LED on the LED module.

## 6.3.7 Project 51: Smart Windows



#### Description

In life, we can see all kinds of smart products, such as smart home. Smart homes include smart curtains, smart windows, smart TVs, smart lights, and more. In this experiment, we use a steam sensor to detect rainwater, and then achieve the effect of closing and opening the window by a servo.

### **Required Components**



**Connection Diagram** 



**Test Code** 

```
/*
* Filename : smart window
* Description : Water drop sensor controls steering gear rotation.
* Auther
          : http//www.keyestudio.com
*/
#include <ESP32Servo.h>//Import the steering gear library file
int adcVal = 0;//A variable that holds the ADC value output by the droplet sensor
int servoPin = 15; // Define the servo pin
Servo myservo;//Defines an instance of the steering gear class
#define PIN_ADC 34 //the pin of the Water drop sensor
void setup(){
 Serial.begin(9600);
 pinMode(PIN_ADC, INPUT);
                           // standard 50 hz servo
 myservo.setPeriodHertz(50);
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→object
}
void loop(){
```

(continues on next page)

### **Code Explanation**

We can control a servo to rotate by a threshold.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. When the sensor detects a certain amount of water, the servo rotates to achieve the effect of closing or opening windows.

# 6.3.8 Project 52: Sound Activated Light



### Introduction

In this lesson, we will make a smart sound activated light using a sound sensor and an LED module. When we make a sound, the light will automatically turn on; when there is no sound, the light will automatically turn off. How it works? Because the sound-controlled light is equipped with a sound sensor, and this sensor converts the intensity of external sound into a corresponding value. Then set a threshold, when the threshold is exceeded, the light will go on, and when it is not exceeded, the light will go off.

### Components



### **Connection Diagram**



fritzing

### **Test Code**

```
/*
* Filename : sound-controlled lights
* Description : Sound sensor controls LED on and off
* Auther
           : http//www.keyestudio.com
*/
int ledPin = 15;//LED is connected to GP15
int microPin = 34;//Sound sensor is connected to GPI034
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(ledPin, OUTPUT);//LED is the output mode
}
void loop() {
 int val = analogRead(microPin);//Read analog value
 Serial.print(val);// Serial port print
 if(val > 600){//exceed the threshold value
   digitalWrite(ledPin, HIGH);//Lighting LED 3sand print the corresponding information
   Serial.println(" led on");
   delay(3000);
 }else{//otherwise
   digitalWrite(ledPin, LOW);//Turn off the LED and print the corresponding information
   Serial.println(" led off");
 }
 delay(100);
}
```
## **Code Explanation**

We set the ADC threshold value to 600. If more than 600, LED will be on 3s; on the contrary, it will be off.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600, then the corresponding volume ADC value will be displayed.

When the analog value of sound is greater than 600, the LED on the LED module will light up 3s, otherwise it will go off.

∞ COM21			_		$\times$
					Send
0 led off					~
0 led off					
0 led off					
0 led off					
768 led on					
0 led off					
0 led off					
0 led off					
613 led on					
200 led off					
0 led off					
0 led off					
1300 led on					
					~
Autoscroll Show timestamp	Newline $\vee$	9600 bau	d ~	Clear	output

## 6.3.9 Project 53: Fire Alarm



### Description

In this experiment, we will make a fire alarm system. Just use a flame sensor to control an active buzzer to emit sounds.

### **Required Components**



#### **Connection Diagram**



#### **Test Code**

```
/*
* Filename : Flame Alarm
* Description : Controlling the buzzer by flame sensor.
* Auther
         : http//www.keyestudio.com
*/
int item = 0;
void setup() {
 Serial.begin(9600);
 pinMode(4, INPUT);//Flame sensor digital pin is connected to GPI04
 pinMode(15, OUTPUT);//Buzzer pin is connected to GPI015
}
void loop() {
 item = digitalRead(4);//Read the digital level output by the flame sensor
 Serial.println(item);//Newline print level signal
 if (item == 0) {//Flame detected
   digitalWrite(15, HIGH);//Turn on the buzzer
 } else {//Otherwise, turn off the buzzer
   digitalWrite(15, LOW);
 }
 delay(100);//Delay 100ms
}
```

**Code Explanation** 

This flame sensor uses an analog pin and a digital pin. When a flame is detected, the digital pin outputs a low level. In this experiment we will use the digital port.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. When the sensor detects the flame, the external active buzzer will emit sounds, otherwise the active buzzer will not emit sounds.

## 6.3.10 Project 54: Smoke Alarm



### Description

In this experiment, we will make a smoke alarm by a TM16504-Digit segment module, a gas sensor and an active

buzzer.

## **Required Components**



## **Connection Diagram**



**Test Code** 

```
/*
* Filename : smoke alarm
* Description : MQ2 controls a buzzer and a four-digit analog smoke tester
* Auther
            : http//www.keyestudio.com
*/
#include "TM1650.h" //Import the TM1650 library file
int adcVal = 0; //display ADC value
//the interfaces are GPI021 and GPI022
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
void setup() {
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, 0=display off, 1=display on,__
→default : 1
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b); //DigitalTube.clearBit(0 to 3); Clear bit display.
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
```

(continues on next page)

```
(continued from previous page)
```

```
DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
\rightarrow --9
 pinMode(15, OUTPUT);//the buzzer is connected to GPI015
}
void loop() {
 adcVal = analogRead(34);//Read the ADC values of MQ2
 displayFloatNum(adcVal);;//Four digit tube display adcVal values
 if (adcVal > 1000) {//ADC value is greater than 1000
   digitalWrite(15, HIGH); // buzzer alarming
 } else {//or else
   digitalWrite(15, LOW); //Turn off the buzzer
 3
 delay(100);//delay 100ms
}
void displayFloatNum(float adcVal){
 if(adcVal > 9999)
   return;
 int dat = adcVal*10;
   //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return:
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return;
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
```

#### **Code Explanation**

Define an integer variable val to store the analog value of the smoke sensor, and then we display the analog value in the four-digit digital tube, and then set a threshold, and when the threshold is reached, the buzzer will sound.

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. When the concentration of combustible gas exceeds the standard, the active buzzer module will give an alarm, and the four-digit digital tube will display the concentration value.

## 6.3.11 Project 55: Alcohol Sensor



### Description

In the last experiment, we made a smoke alarm. In this experiment, we combine the active buzzer, the MQ-3 alcohol sensor, and a four-digit digital tube to test the alcohol concentration through the alcohol sensor. Then, the concentration to control the active buzzer alarm and the four-digit digital tube to display the concentration. So as to achieve the simulation effect of alcohol detector.

#### **Components Required**



**Connection Diagram** 



fritzing

### **Test Code**

```
/*
* Filename : breathalyzer
* Description : MQ3 controls a buzzer and a four-digit tube to simulate a breathalyzer.
            : http//www.keyestudio.com
* Auther
*/
#include "TM1650.h" //Import the TM1650 library file
int adcVal = 0; //display ADC value
//the interfaces are GPI021 and GPI022
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
void setup() {
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, 0=display off, 1=display on,
→default : 1
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                             //DigitalTube.clearBit(0 to 3); Clear bit display.
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>→</u>--9
 pinMode(15, OUTPUT);//the buzzer is connected to GPI015
}
void loop() {
 adcVal = analogRead(34);//Read the ADC values of MQ3
 displayFloatNum(adcVal);//Four digit tube display adcVal values
 if (adcVal > 1000) {//ADC value is greater than 1000
   digitalWrite(15, HIGH); // buzzer alarming
 } else {//or else
   digitalWrite(15, LOW); //Turn off the buzzer
 }
 delay(100);//delay 100ms
}
void displayFloatNum(float adcVal){
 if(adcVal > 9999)
   return;
 int dat = adcVal*10;
                                                                      (continues on next page)
```

```
(continued from previous page)
```

```
//DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return;
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
```

## **Code Explanation**

Define an integer variable val to store the ADC value of the alcohol sensor, then we display the analog value in the four-digit display module and set a threshold.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. When different alcohol concentrations are detected, the active buzzer module will alarm, and the four-digit digital display will show the concentration value.

# 6.3.12 Project 56: Ultrasonic Radar



### Description

We know that bats use echoes to determine the direction and the location of their preys. In real life, sonar is used to detect sounds in the water. Since the attenuation rate of electromagnetic waves in water is very high, it cannot be used to detect signals, however, the attenuation rate of sound waves in the water is much smaller, so sound waves are most commonly used underwater for observation and measurement.

In this experiment, we will use a speaker module, an RGB module and a 4-digit tube display to make a device for detection through ultrasonic.

### **Required Components**



### **Connection Diagram**



fritzing

#### **Test Code**

```
/*
* Filename : Ultrasonic radar
* Description : Ultrasonic control four digit tube, buzzer and RGB analog ultrasonic
⇔radar.
* Auther
            : http//www.keyestudio.com
*/
#include "TM1650.h" //Import the TM1650 library file
//the interfaces are GPI021 and GPI022
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
int beeppin = 18; //Define the horn pin as GPI018
int TrigPin = 13; //Set the Trig pin to GPI013
int EchoPin = 14; //Set the Echo pin to GPI014
int distance;//Distance measured by ultrasound
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
const byte chns[] = \{0, 1, 2\};
                             //define the pwm channels
```

(continues on next page)

(continued from previous page)

```
float checkdistance() { //get distance
  // A short low level is given beforehand to ensure a clean high pulse:
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
  // The sensor is triggered by a high pulse of 10 microseconds or more
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  // Read the signal from the sensor: a high level pulse
  //Its duration is the time (in microseconds) from sending the ping command to.
→receiving the echo from the object
  float distance = pulseIn(EchoPin, HIGH) / 58.00; //Convert to distance
  delay(10);
 return distance;
}
void setup() {
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
  DigitalTube.displayOnOFF(); //display on or off, 0=display off, 1=display on,
→default : 1
  for(char b=1;b<5;b++){
   DigitalTube.clearBit(b); //DigitalTube.clearBit(0 to 3); Clear bit display.
  }
  // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>→--9</u>
  pinMode(TrigPin, OUTPUT);//Sets the Trig pin as output
  pinMode(EchoPin, INPUT); //Set the Echo pin as input
  ledcSetup(3, 1000, 8);//setup the pwm channels,1KHz,8bit
  ledcAttachPin(18, 3);
  for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit
   ledcSetup(chns[i], 1000, 8);
    ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
  distance = checkdistance(); //Ultrasonic ranging
  displayFloatNum(distance); //Nixie tube shows distance
  if (distance \leq 10) {
   ledcWrite(3, 100);
   delay(100);
   ledcWrite(3, ∅);
   ledcWrite(chns[0], 255); //Common cathode LED, high level to turn on the led.
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], ∅);
  } else if (distance > 10 && distance <= 20) {</pre>
   ledcWrite(3, 200);
   delay(200);
   ledcWrite(3, 150);
    ledcWrite(chns[0], 0);
```

(continued from previous page)

```
ledcWrite(chns[1], 255);
   ledcWrite(chns[2], 0);
 } else {
   ledcWrite(3, 0);
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 255);
 }
}
void displayFloatNum(float distance){
 if(distance > 9999)
   return:
 int dat = distance*10;
  //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return:
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return:
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
```

### **Code Explanation**

We set sound frequency and light color by adjusting different distance range.

We can adjust the distance range in the code.

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. When the ultrasonic sensor detects different distances(within 20 cm), the buzzer will produce different frequencies of sound, the RGB will show different colors, and the measured distances are displayed on the 4-digit tube display.



# 6.3.13 Project 57: IR Remote Control

## Introduction

In the previous experiments, we learned how to turn on/off the LED and adjust its brightness via PWM and print the button value of the IR remote control in the serial monitor window. Herein, we use an infrared remote control to turn on/off an LED.

### Components



## **Connection Diagram**



**Test Code** 

```
/*
* Filename : IR Control LED
* Description : Remote controls LED on and off
* Auther : http//www.keyestudio.com
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
const uint16_t recvPin = 15; // Infrared receiving pin 15
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results;
                       // Create a decoding results class object
int led = 4;//LED connect to GP4
void setup() {
 Serial.begin(9600);
                                 // Start the receiver
 irrecv.enableIRIn();
 pinMode(led, OUTPUT);
}
(continues on next page)
```

(continued from previous page)

```
void loop() {
 if(irrecv.decode(&results)) {
                                   // Waiting for decoding
   serialPrintUint64(results.value, HEX);// Print out the decoded results
   Serial.print("");
   handleControl(results.value);
                                   // Handle the commands from remote control
   irrecv.resume();
                                   // Receive the next value
 }
}
 void handleControl(unsigned long value) {
   if (value == 0xFF6897) // Receive the number '1'
    {
      digitalWrite(led, HIGH);//turn on LED
      Serial.println(" led on");
    }
   else if (value == 0xFF9867) // Receive the number '2'
    {
       digitalWrite(led, LOW);//turn off LED
      Serial.println(" led off");
   }
}
         ******
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600. Press the button 1 of the remote, which will be displayed on the monitor, and the LED will be on. Similarly, press the button 2, the LED will be off.

💿 CON	A21	_		$\times$
				Send
FF6897	led on			^
FF9867	led off			
FF6897	led on			
FF9867	led off			
				~
Auto:	scroll Show timestamp 9600 baud	$\sim$	Clear	output

6. 14 Project 58: Heat Dissipation Device

## Description

We will use a temperature sensor and some modules to make a smart cooling device in this experiment. When the ambient temperature is higher than a certain value, the motor is turned on, thereby reducing the ambient temperature and achieving the heat dissipation effect. Then display the temperature value in the four-digit segment display.

## **Required Components**



## **Connection Diagram**



### **Test Code**

(continued from previous page)

```
→ Abstractor
* Auther
               : http//www.keyestudio.com
*/
#include <DS18B20.h>
#include "TM1650.h" //Import the TM1650 library file
//The two ports are GP21 and GP22
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
//ds18b20 pin to 13
DS18B20 ds18b20(13);
void setup() {
  Serial.begin(9600);
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, 0=display off, 1=display on,
→default : 1
  for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                                  //DigitalTube.clearBit(0 to 3); Clear bit display.
  3
  // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>→</u>--9
  //Motor is connected to 15 4
 pinMode(15, OUTPUT);
 pinMode(4, OUTPUT);
}
void loop() {
  double temp = ds18b20.GetTemp();//Read the temperature
  temp *= 0.0625;//The conversion accuracy is 0.0625/LSB
  Serial.println(temp);
  displayFloatNum(temp);//4- digit tube display temperature value
  if (temp > 25) {//When the temperature exceeds 25 degrees Celsius, turn on the fan
   digitalWrite(15, LOW);
   digitalWrite(4, HIGH);
  } else {//Otherwise, turn off the fan.
   digitalWrite(15, LOW);
   digitalWrite(4, LOW);
  }
 delay(100);
}
void displayFloatNum(float temp){
  if(temp > 9999)
   return:
  int dat = temp*10;
  //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
  if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
```

(continues on next page)

(continued from previous page)

```
DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return:
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
```

## **Code Explanation**

The setting of variables and the storage of detection values are the same as what we learned earlier. We also set a temperature threshold and control the rotation of the motor when the threshold is exceeded, and then we use the digital tube to display the temperature value.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch on the ESP32 expansion board to the ON end, compile and upload the code to the ESP32. After uploading successfully, we can see the temperature of the current environment (unit is Celsius) on the four-digit segment display, as shown in the figure below. If this value exceeds the value we set, the fan will rotate to dissipate heat.



# 6.3.15 Project 59: Intelligent Entrance Guard System

## Description

In this project, we use the RFID522 card swiping module and the servo to set up an intelligent access control system. The principle is very simple. We use RFID522 swipe card module, an IC card or key card to unlock.

## **Required Components**



#### **Connection Diagram**



## **Test Code**

Note: Different RFID-MFRC522 IC cards and keys have diverse values. You can substitute your own IC cards and keys values for the corresponding values read by the RFID-MFRC522 module in the program, otherwise the servo can't be controlled when uploading the test code to the ESP32.

For example: You can replace the rfid\_str of the **if (rfid\_str == "edf7945a"** || **rfid\_str == "4c96b6e"**) in the program code with your own IC cards and keys values read by the RFID-MFRC522 module.

```
/*
* Filename : Intelligent_access_control
* Description : RFID controlled steering gear simulated door opening
* Auther
          : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
#include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int servoPin = 15; // Servo motor pin
String rfid_str = "";
void setup() {
 Serial.begin(9600);
 Wire.begin();
 mfrc522.PCD_Init();
 ShowReaderDetails();
                              // dispaly PCD - MFRC522 read carder
 Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
 myservo.setPeriodHertz(50);
                                    // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo_
                                                                    (continues on next page)
```

(continued from previous page)

```
→object
 myservo.write(0);
 delay(500);
}
void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return;
 }
 // select one of door cards. UID and SAK are mfrc522.uid.
 // save UID
 rfid_str = ""; //String emptying
 Serial.print(F("Card UID:"));
 for (byte i = 0; i < mfrc522.uid.size; i++) {</pre>
   rfid_str = rfid_str + String(mfrc522.uid.uidByte[i], HEX); //Convert to string
   //Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   //Serial.print(mfrc522.uid.uidByte[i], HEX);
 }
 Serial.println(rfid_str);
 if (rfid_str == "edf7945a" || rfid_str == "4c96b6e") {
   myservo.write(180);
   delay(500);
   Serial.println(" open the door!");
   }
}
void ShowReaderDetails() {
  // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print(F("MFRC522 Software Version: 0x"));
 Serial.print(v, HEX);
 if (v == 0x91)
   Serial.print(F(" = v1.0"));
 else if (v == 0x92)
   Serial.print(F(" = v2.0"));
 else
   Serial.print(F(" (unknown)"));
 Serial.println("");
 // when returning to 0x00 or 0xFF, may fail to transmit communication signals
 if ((v == 0x00) || (v == 0xFF)) \{
   Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?
'));
 }
}
```

#### **Code Explanation**

In the previous experiment, our card swipe module has tested the information of IC card and key. Then we use this

corresponding information to control the door.

## Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600. When we use the IC card or blue key to swipe the card, the monitor displays the card and the key information and "open the door", at the same time, the servo rotates to the corresponding angle to simulate opening the door.

💿 СОМ30		—		$\times$
				Send
Scan PICC to see UID, type, and data blocks				~
Card UID:4c96b6e				
open the door!				
Card UID:edf7945a				
open the door!				
Card UID:edf7945a				
open the door!				
Card UID:edf7945a				
open the door!				
Card UID:4c96b6e				
open the door!				
Card UID:4c96b6e				
open the door!				
				~
Autoscroll Show timestamp Newline	~ 9600	baud v	Clear	output

## 6.3.16 Project 60Bluetooth

This chapter mainly introduces how to use the Bluetooth of ESP32 for simple data transmission with mobile phone. Experiment 60.1 is conventional Bluetooth, and experiment 60.2 is Bluetooth control LED.

## **Project 60.1Classic Bluetooth**

#### Components



In this experiment, we need to use a Bluetooth dabbed serial Bluetooth terminal for a study. If you haven't install it, please click the installation: https://www.appsapk.com/serial-bluetooth-terminal/ .

Here is its sign:



### **Component Knowledge**

Bluetooth is a short-distance communication system that can be divided into two types, namely low power Bluetooth (BLE) and classic Bluetooth. There are two modes for simple data transfer: master mode and slave mode.

## Master Mode:

In this mode, work is done on the master device and can be connected to the slave device. When the device initiates a connection request in the main mode, information such as the address and pairing password of other Bluetooth devices are required. Once paired, you can connect directly to them.

### Slave Mode:

A Bluetooth module in the slave mode can only accept connection requests from the host, but cannot initiate connection requests. After being connected to a host device, it can send and receive data through the host device. Bluetooth devices can interact with each other, when they interact, the Bluetooth device in the main mode searches for nearby devices. While a connection is established, they can exchange data. For example, when a mobile phone exchanges data with ESP32, the mobile phone is usually in master mode and the ESP32 is in slave mode.



master mode

slave mode

## Wiring Diagram

We can use a USB cable to connect ESP32 mainboard to the USB port on a computer.



**Test Code** 

```
/*
* Filename
           : Classic Bluetooth
* Description : ESP32 communicates with the phone by bluetooth and print phone's data.
→via a serial port
* Auther
          : http//www.keyestudio.com
*/
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
String buffer;
void setup() {
 Serial.begin(115200);
 SerialBT.begin("ESP32test"); //Bluetooth device name
 Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
void loop() {
 if (Serial.available()) {
   SerialBT.write(Serial.read());
 }
 if (SerialBT.available()) {
   Serial.write(SerialBT.read());
 }
 delay(20);
}
```

## Test Result

Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200. When you see the serial prints the character, as shown below, it means that the ESP32's bluetooth is waiting for connect ion with a phone. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)



сомз	_		$\times$
			Send
ets Jun 8 2016 00:22:57			^
<pre>rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT) configsip: 0, SPIWP:0xee clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00 mode:DIO, clock div:1 load:0x2fff0010 lon:4</pre>			
load:0x3ff001c,len:4 load:0x40078000,len:8596 load:0x40080400,len:6980 entry 0x400806f4			
The device started, now you can pair it with bluetooth!			
Autoscroll Show timestamp Newline V 115200 baud V	/ Cle	ar oi	1tput 🗸

Ensure that your mobile phone Bluetooth is enabled and the Bluetooth application of "Serial Bluetooth Terminal" is installed.



Click"Search"search for the nearby Bluetooth and select to connect the"ESP32 test".



Open the software APP and click the left side of the terminal, select "Devices".



If you select ESP32test in classic bluetooth mode, a successful connection message will appear as shown below.

← Devices	٠		
BLUETOOTH CLASSIC	BLUETOOTH LE	11:25:41.094 Connecting to ESP32test 11:25:41.195 Connected	
ESP32test C4:4F:33:22:B6:3B		M1 M2 M3 M4 M5	M6
			>

Data can be transferred between your phone and a computer via ESP32 now.

Send "Hello", When the computer receives it, which will reply with "Hi!".

💿 СОМЗ			$\times$
Hi!			Send
ets Jun 8 2016 00:22:57			^
<pre>rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT) configsip: 0, SPIWP:0xee</pre>			
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00 mode:DIO. clock div:1			
load:0x3fff0018,len:4			
load:0x3fff001c,len:1496			
load:0x40078000,len:8596			
entry 0x400806f4			
The device started, now you can pair it with bluetooth! Hello!			
			~
Autoscroll Show timestamp Newline v 115200 ba	ud ~	Clear	output

	11 CMCC ··· 11		:26 🗚	T 🛸 🖪	D 4G 92% 🔜
≡	Term	inal	-	<b>(1)</b> -	<b>i</b> :
11:25:41. 11:25:41. 11:26:11. 11:26:24.	094 Conn 195 Conn 913 Hello 759 Hi!	ecting to I ected !	ESP32te	st	
M1	M2	М3	M4	M5	5 M6

## Project 60.2Bluetooth Control LED

## Components

ESP3	2*1	ESP32 Expansion Board*1
Keyestudio Purple LED Module*1	3P Dupont*1	MicroUSB Cable*1

## Wiring Diagram



fritzing

## Test Code

```
/*
* Filename
           : Bluetooth Control LED
* Description : The phone controls esp32's led via bluetooth.
              When the phone sends "LED_on," ESP32's LED lights turn on.
              When the phone sends "LED_off," ESP32's LED lights turn off.
* Auther
             : http//www.keyestudio.com
*/
#include "BluetoothSerial.h"
#include "string.h"
#define LED 15
BluetoothSerial SerialBT;
char buffer[20];
static int count = 0;
void setup() {
 pinMode(LED, OUTPUT);
 SerialBT.begin("ESP32test"); //Bluetooth device name
 Serial.begin(115200);
 Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
void loop() {
 while(SerialBT.available())
 {
   buffer[count] = SerialBT.read();
   count++;
 }
 if(count>0){
   Serial.print(buffer);
   if(strncmp(buffer, "led_on", 6)==0){
     digitalWrite(LED,HIGH);
   }
   if(strncmp(buffer,"led_off",7)==0){
     digitalWrite(LED,LOW);
```

(continues on next page)

	(continued from previous page)
	}
	count=0;
	<pre>memset(buffer,0,20);</pre>
	}
}	
//	***************************************

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The APP operation is the same as the **project 60.1**. To make the external LED on and off, simply change the sending content to "led\_on" and "led\_off". Moving the APP to send data:



The serial monitor will display as follows:

© COM4			_		×
					Send
abcd					^
led_on					
led_off					
led_on					
led_off					
					~
Autoscroll 🗌 Show timestamp	Newline $\vee$	115200 baud	$\sim$	Clear	c output

# LED Circumstance





#### Note:

If the sent content is not "led-on 'or" led-off ", the status of the LED will not change. If the LED is on, it remains on when irrelevant content is received; Conversely, if the LED is off, it continues to be off when irrelevant content is received.

# 6.3.17 Project 61WIFI Station Mode

## Description

ESP32 has three different WiFi modes: Station mode, AP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi running mode before using, otherwise the WiFi cannot be used. In this project, we are going to learn the WiFi Station mode of the ESP32.

## Components

<b>%</b>	
MicroUSB Cable*1	ESP32*1

## Wiring Diagram

Plug the ESP32 to the USB port of your PC.



## **Component Knowledge**

## Station mode

When setting Station mode, the ESP32 is taken as a WiFi client. It can connect to the router network and communicate with other devices on the router via a WiFi connection. As shown in the figure below, the PC and the router have been connected. If the ESP32 wants to communicate with the PC, the PC and the router need to be connected.



## Test Code

Since WiFi names and passwords vary from place to place, thereby users need to enter the correct WiFi names and passwords in the box shown below before the program code runs.
```
🚥 WiFi Station Mode | Arduino 1.8.16
                                                                    \times
File Edit Sketch Tools Help
             +
                                                                         Ø
     +
  WiFi_Station_Mode
 //*********
               1*
  * Filename : WiFi Station
                                          Enter the correct Router
  * Description : Connect to your router using ESP3
                                            name and password.
  * Auther : http//www.keyestudio.com
 */
 #include <WiFi.h> //Include the WiFi Library header file of ESP32.
 //Enter correct router name and password
                           = "ChinaNet-2.4G-ODFO"; //Enter the router name
 const char *ssid_Router
 const char *password Router = "ChinaNet@233"; //Enter the router password
 void setup(){
  Serial.begin(115200);
  delay(2000);
  Serial.println("Setup start");
  WiFi.begin(ssid_Router, password_Router);//Set ESP32 in Station mode and conne
  Serial.println(String("Connecting to ")+ssid Router);
 <
                                                                          >
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressi
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressi
<
                                                                          х
                              ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM30
//*********
/*
* Filename : WiFi Station
* Description : Connect to your router using ESP32
* Auther : http//www.keyestudio.com
*/
#include <WiFi.h> //Include the WiFi Library header file of ESP32.
//Enter correct router name and password.
const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name
const char *password_Router = "ChinaNet@233"; //Enter the router password
void setup(){
 Serial.begin(115200);
 delay(2000);
 Serial.println("Setup start");
 WiFi.begin(ssid_Router, password_Router);//Set ESP32 in Station mode and connect it to.
\rightarrow your router.
                                                                     (continues on next page)
```

```
(continued from previous page)
```

```
Serial.println(String("Connecting to ")+ssid_Router);
//Check whether ESP32 has connected to router successfully every 0.5s.
while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
}
Serial.println("\nConnected, IP address: ");
Serial.println(WiFi.localIP());//Serial monitor prints out the IP address assigned to_
    -ESP32.
    Serial.println("Setup End");
}
void loop() {
}
```

#### **Test Result**

After entering the correct WiFi names and passwords, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200. When the ESP32 successfully connects to ssid\_WiFi, the serial monitor prints out the IP address, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)



# 6.3.18 Project 62WIFI AP Mode

### Description

In this project, we are going to learn the WiFi AP mode of the ESP32.

### Components



### Wiring Diagram

Plug the ESP32 mainboard to the USB port of your PC.



#### **Component Knowledge**

#### **AP Mode:**

When setting AP mode, a hotspot network will be created, waiting for other WiFi devices to connect. As shown below:

Take the ESP32 as the hotspot, if a phone or PC needs to communicate with the ESP32, it must be connected to the ESP32's hotspot. Communication is only possible after a connection is established via the ESP32.



### Test Code

Before running the code, you can make any changes to the ESP32 AP name and password in the box as shown below, but in a default circumstance, it doesn't need to modify.

```
\times
 🚥 WiFi AP Mode | Arduino 1.8.16
                                                                  File Edit Sketch Tools Help
             Ŧ
                                                                        Ø
     •
  WiFi_AP_Mode
                //*****
 1*
  * Filename : WiFi AP
                                             Set a name and a
  * Description : Set ESP32 to open an access point
                                           password for ESP32 AP.
  * Auther : http//www.keyestudio.com
 */
 #include <WiFi.h> //Include the WiFi Library heade file of ESP32.
                      = "ESP32_Wifi"; //Enter the router name
 const char *ssid AP
 const char *password_AP = "12345678"; //Enter the router password
 IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself
 IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself
 IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself
 void setup(){
  Serial.begin(115200);
  delay(2000);
 <
                                                                        >
                             ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM30
//*********
/*
* Filename : WiFi AP
* Description : Set ESP32 to open an access point
* Auther : http//www.keyestudio.com
*/
#include <WiFi.h> //Include the WiFi Library header file of ESP32.
const char *ssid_AP = "ESP32_Wifi"; //Enter the router name
const char *password_AP = "12345678"; //Enter the router password
IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself
IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself
IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself
void setup(){
 Serial.begin(115200);
 delay(2000);
                                                                   (continues on next page)
```

```
Serial.println("Setting soft-AP configuration ... ");
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
 Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
 Serial.println("Setting soft-AP ... ");
 boolean result = WiFi.softAP(ssid_AP, password_AP);
 if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("Setup End");
}
void loop() {
}
```

#### **Test Result**

Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)



© COM20	_		$\times$
			Send
configsip: 0, SPIWP:0xee			^
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00			
mode:DIO, clock div:1			
load:0x3fff0018,len:4			
load:0x3fff001c,len:1496			
load:0x40078000,len:8596			
load:0x40080400,len:6980			
entry 0x400806f4			
Setting soft-AP configuration			
Ready			
Setting soft-AP			
Ready			
Soft-AP IP address = 192.168.1.108			
MAC address = 58:BF:25:8A:19:D1			
Setup End			
			~
Autoscroll Show timestamp Newline V 115200 baud	~ c	lear o	utput

When observing the printed information of the serial monitor, turn on the WiFi scanning function of the mobile phone, you can see the ssid\_AP on ESP32, which is dubbed "ESP32\_Wifi" in this program code. You can connect to it either by typing the password "12345678" or by modifying the program code to change its AP name and password.



# 6.3.19 Project 63WIFI AP+Station Mode

#### Description

In this project, we are going to learn the AP+Station mode of the ESP32.

#### Components



### Wiring Diagram

Plug the ESP32 mainboard to the USB port of your PC



#### **Component Knowledge**

#### **AP+Station mode:**

In addition to the AP mode and the Station mode, AP+Station mode can be used at the same time. Turn on the Station mode of the ESP32, connect it to the router network, and it can communicate with the Internet through the router. Then turn on the AP mode to create a hotspot network. Other WiFi devices can be connected to the router network or the hotspot network to communicate with the ESP32.

### **Test Code**

Before running the code, you need to modify the ssid\_Routerpassword\_Routerssid\_AP and password\_AP, as shown in the box below:

```
WiFi_Station_AP_Mode | Arduino 1.8.16
                                                                        \times
File Edit Sketch Tools Help
                                                                             Ø
  WiFi_Station_AP_Mode
                  /*
              : WiFi AP+Station Please enter the correct names
  * Filename
 * Description : ESP32 connects to the user's router, turning on an access po

* Auther : http://www.keyestuand.passwords of Router and
 * Description : ESP32 connects to the user
 */
 #include <WiFi.h>
 const char *ssid Router
                               "ChinaNet-2.4G-ODFO"; //Enter the router name
 const char *password Router =
                              "ChinaNet@233"; //Enter the router password
 const char *ssid AP
                              "ESP32 Wifi"; //Enter the router name
                               "12345678"; //Enter the router password
 const char *password AP
 void setup(){
  Serial.begin(115200);
  Serial.println("Setting soft-AP configuration ... ");
  WiFi.disconnect();
  WiFi.mode(WIFI AP);
   Comist swintln/MCotting coft 3D
                                     пι.
 <
                                      ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3
* Filename : WiFi AP+Station
* Description : ESP32 connects to the user's router, turning on an access point
           : http//www.keyestudio.com
* Auther
*/
#include <WiFi.h>
```

Chapter 6. Arduino tutorial

```
const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name
const char *password_Router = "ChinaNet@233"; //Enter the router password
const char *ssid_AP = "ESP32_Wifi"; //Enter the router name
const char *password_AP
                        = "12345678"; //Enter the router password
void setup(){
 Serial.begin(115200);
 Serial.println("Setting soft-AP configuration ... ");
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
 Serial.println("Setting soft-AP ... ");
 boolean result = WiFi.softAP(ssid_AP, password_AP);
 if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("\nSetting Station configuration ... ");
 WiFi.begin(ssid_Router, password_Router);
 Serial.println(String("Connecting to ")+ ssid_Router);
 while (WiFi.status() != WL_CONNECTED){
   delay(500);
   Serial.print(".");
 }
 Serial.println("\nConnected, IP address: ");
 Serial.println(WiFi.localIP());
 Serial.println("Setup End");
}
void loop() {
}
```

#### **Test Result**

Ensure that the code in the program has been modified correctly, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)



💿 сомз			_		$\times$
					Send
load:0x40078000,len:8596					^
load:0x40080400,len:6980					
entry 0x400806f4					
Setting soft-AP configuration					
Setting soft-AP					
Ready					
Soft-AP IP address = 192.168.4.1					
MAC address = 58:BF:25:8A:19:D1					
Setting Station configuration					
Connecting to ChinaNet-2.4G-0DF0					
Connected, IP address:					
192.168.1.157					
Setup End					
					¥
Autoscroll 🗌 Show timestamp	Newline	~ 115200 baud	~ C	lear o	utput

Open the WiFi scanning function of the mobile phone, you can see the ssid\_AP.

	Wi–Fi	
Settings	Wi-Fi View help	
Q Search	More settings	>
Log in to	AVAILABLE NETWORKS	
Access Cloud, > AppGallery, and	ESP32_Wifi Connected	((·
more	ChinaNet-2.4G-0DF0	<b></b>
🛜 WLAN ESP32_Wifi >	Saved, encrypted ChinaNet-Dsvv	
	Encrynted	<u> </u>



# 6.3.20 Project 64: Comprehensive Experiment

#### Introduction

We did a lot of experiments, and for each one we needed to re-upload the code, so can we achieve different functions through an experiment? In this experiment, we will use an external button module to achieve different functions.

#### **Components Required**



Wiring Diagram



**Test Code** 

```
/*
* Filename : Comprehensive experiment
* Description : Multiple sensors/modules work together
* Auther : http//www.keyestudio.com
*/
#include "xht11.h"
#include "adx1345_io.h"
//ADXL345 sda-->21,scl-->22
adx1345 adx1345(21, 22);
//xht11 to gpio15
xht11 xht(15);
//rgb is connected to 4,0,2
int ledPins[] = {4, 0, 2}; //define red, green, blue led pins
const byte chns[] = {0, 1, 2}; //define the pwm channels
int red, green, blue;
//Rocker module port
int X = 35;
int Y = 34;
int KEY = 32;
//Potentiometer pin is connected to analog port 33
int resPin = 33;
//Trace sensor pin connected to IO port 14
int TrackingPin = 14;
//LED is Connected to GP5
#define PIN_LED 5 // the pin of the LED
#define CHAN 3
//Obstacle avoidance sensor is connected to GP27
int Avoid = 27;
//Ultrasonic sensor port
int Trig = 13;
```

```
int Echo = 12;
//Key module port
int button = 23;
int PushCounter = 0;//Store the number of times a key is pressed
int yushu = \emptyset;
unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are received, not
\rightarrow the parity bits
bool ir_flag = 1;
float out_X, out_Y, out_Z;
void counter() {
  delay(10);
 ir_flag = 0;
 if (!digitalRead(button)) {
   PushCounter++;
  }
}
void setup() {
  Serial.begin(9600);//Set baud rate to 9600
  pinMode(KEY, INPUT);//Button of remote sensing module
  ledcSetup(CHAN, 1000, 12);
  ledcAttachPin(PIN_LED, CHAN);
  pinMode(button, INPUT);//The key module
  attachInterrupt(digitalPinToInterrupt(button), counter, FALLING); //External_
→interrupt 0, falling edge fired
  pinMode(Avoid, INPUT);//Obstacle avoidance sensor
  pinMode(Trig, OUTPUT);//Ultrasonic module
  pinMode(Echo, INPUT);
  adxl345.Init();
                                //setup the pwm channels,1KHz,8bit
  for (int i = 0; i < 3; i++) {
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 delay(1000);
}
}
void loop() {
 yushu = PushCounter % 8;
  if (yushu == 0) { //The remainder is 0
   yushu_0(); //rgb displays
  } else if (yushu == 1) { //The remainder is 1
   yushu_1(); //Displays the high and low levels read by the tracking sensor
  } else if (yushu == 2) { //The remainder is 2
   yushu_2(); //Display temperature and humidity value
  } else if (yushu == 3) { //The remainder is 3
   yushu_3(); //Displays the rocker value
  }else if (yushu == 4) { //The remainder is 4
    yushu_4(); //Display potentiometer ADC value and potentiometer control LED
  } else if (yushu == 5) { //The remainder is 5
```

```
(continued from previous page)
   yushu_5(); //Obstacle avoidance sensor detects obstacles
  } else if (yushu == 6) { //The remainder is 6
   yushu_6(); //Shows the distance detected by ultrasound
 } else if (yushu == 7) { //The remainder is 7
   yushu_7(); //ADXL345 triaxial acceleration value
 }
}
//RGB
void yushu_0() {
 red = random(0, 256);
  green = random((0, 256);
 blue = random((0, 256);
  setColor(red, green, blue);
 delay(200);
}
void setColor(byte r, byte g, byte b) {
 ledcWrite(chns[0], 255 - r); //Common anode LED, low level to turn on the led.
 ledcWrite(chns[1], 255 - g);
 ledcWrite(chns[2], 255 - b);
}
void yushu_1() {
 int val = digitalRead(TrackingPin);//Read the digital level output by the tracking.
⇔sensor
  Serial.print(val);//Serial port print value
 if (val == 0) {//White val is 0 detected
                          ");
   Serial.print("
   Serial.println("White");
   delay(100);
  }
 else {//Black val is 1 detected
   Serial.print("
                          ");
   Serial.println("Black");
   delay(100);
  }
}
void yushu_2() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [1] is the fractional part
   Serial.print("% ");
   Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
→part
   Serial.println("C");
 } else { //read error
   Serial.println("sensor error");
  }
  delay(1200);
}
```

```
void yushu_3() {
 int x = analogRead(X);
 int y = analogRead(Y);
 int key = digitalRead(KEY);
  Serial.print("X:");
  Serial.print(x);
  Serial.print("
                    Y:");
  Serial.print(y);
  Serial.print("
                    KEY:");
  Serial.println(key);
  delay(100);
}
void yushu_4() {
 int adcVal = analogRead(resPin); //read adc
  Serial.println(adcVal);
                            // adcVal re-map to pwmVal
  int pwmVal = adcVal;
 ledcWrite(CHAN, pwmVal); // set the pulse width.
  delay(10);
}
void yushu_5() {
 int val = digitalRead(Avoid);
  if (val == 0) {//Obstruction detected
   Serial.println("There are obstacles");
 }
  else {//No obstructions detected
   Serial.println("All going well");
  }
 delay(100);
}
void yushu_6() {
  float distance = checkdistance();
  Serial.print("distance:");
  Serial.print(distance);
 Serial.println("cm");
 delay(100);
}
void yushu_7() {
  adx1345.readXYZ(&out_X, &out_Y, &out_Z);
  Serial.print(out_X);
                  ");
  Serial.print("g
  Serial.print(out_Y);
  Serial.print("g ");
  Serial.print(out_Z);
  Serial.println("g");
  delay(100);
}
```

#### **Code Explanation**

1). Calculate how many times the button is pressed, divide it by 8, and get the remainder which is 0, 1 2, 3, 4, 5, 6 and 7. According to different remainders, construct eight unique functions to control the experiment and realize different functions.

2). Following the instructions, we can add or remove sensors/modules in the wiring, and then change the experimental function in the code.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. At the beginning, the number of the button is 0 and remainder is 0. Open the monitor and set baud rate to 9600.

Press the button, the RGB stops flashing, press once, the remainder is 1. The function of the experiment is to detect black objects and white objects by a line tracking sensor. If the sensor does not detect an object or detects a black object, val is 1, and the serial monitor displays the character "1 Black". When a white object (reflective) is detected, val is 0 and the serial monitor displays the character "0 White", the serial monitor will display as follows:

	COM3				—		$\times$
							Send
1	Black						^
1	Black						
1	Black						
1	Black						
1	Black						
1	Black						
0	White						
0	White						
0	White						
0	White						
0	White						
0	White						
0	White						
0	White						
0	White						
							~
	Autoscroll 🗌 Show timestamp	Ne	where $\sim$	9600 baud	~	Clear	output

Press a key twice, the time of pressing buttons is 2 and the remainder is 2. Read temperature and humidity values. As shown below:

💿 CON	130			_	_		$\times$
							Send
RH:45%	Temp:28C						^
RH:45%	Temp:28C						
RH:45%	Temp:28C						
RH:63%	Temp:28C						
RH:72%	Temp:28C						
RH:75%	Temp:28C						
RH:81%	Temp:28C						
RH:88%	Temp:29C						
RH:90%	Temp:29C						
RH:92%	Temp:29C						
RH:93%	Temp:30C						
RH:95%	Temp:30C						
RH:95%	Temp:30C						
							~
Autos	croll 🗌 Show timestamp	Newline	~ 9600	baud 🕚	~	Clear	output

Press a key again, the time of pressing buttons is 3 and the remainder is 3. Read digital values at x, y and z axis of the joystick module. As shown below:

💿 co	M30		-		$\times$
					Send
X:4095	Y:4095	KEY:0			~
X:1903	Y:4095	KEY:0			
X:0	Y:4095	KEY:0			
X:0	Y:1906	KEY:0			
X:868	¥:0 1	KEY:0			
X:3487	Y:0	KEY:0			
X:1914	Y:2477	KEY:0			
X:1894	Y:2423	KEY:0			
X:1901	Y:1909	KEY:0			
X:1897	Y:1905	KEY:0			
X:1899	Y:1905	KEY:0			
X:1904	Y:1913	KEY:1			
X:1898	Y:1906	KEY:1			
					¥
Auto	oscroll 🗌 S	how timestamp	Newline $\checkmark$ 9600 baud $\checkmark$	Clear	output

Press the key for the fourth time, the remainder is 4. Then the potentiometer can adjust the PWM value at the GPI05 port to control LED brightness of the purple LED.

💿 СОМ30				_		$\times$
						Send
2559						^
2671						
2747						
2815						
2885						
2959						
3027						
3131						
3255						
3391						
3579						
3839						
4095						
		_				×
Autoscroll 🗌 Show timestamp	Newline $\sim$	9600	baud	$\sim$	Clear	output

Press the key for the fifth time, the remainder is 5. Then the ultrasonic sensor can detect obstacles, as shown below:

COM30			_		$\times$
					Send
All going well					~
All going well					
All going well					
All going well					
All going well					
All going well					
All going well					
There are obstacles					
There are obstacles					
There are obstacles					
There are obstacles					
There are obstacles					
There are obstacles					
					~
Autoscroll 🗌 Show timestamp	Newline $\checkmark$	9600 baud	$\sim$	Clear	output

Press the key for the sixth time, the remainder is 6. Then the ultrasonic sensor can detect distance away from obstacles, as shown below:

© COM30			_		$\times$
					Send
distance:4.48cm					^
distance:4.90cm					
distance:568.95cm					
distance:5.12cm					
distance:3.98cm					
distance:4.05cm					
distance:4.78cm					
distance:5.47cm					
distance:6.14cm					
distance:7.24cm					
distance:8.12cm					
distance:9.07cm					
distance:10.09cm					
					¥
🗹 Autoscroll 🗌 Show timestamp	Newline ~	9600 bau	ıd ~	Clear	r output

Press the key for seventh time and the remainder is 7. The monitor will print out the acceleration values.

💿 COM3	0		_		$\times$
					Send
59.00g	-87.00g	-234.00g			~
59.00g	-79.00g	-234.00g			
61.00g	-70.00g	-233.00g			
61.00g	-45.00g	-228.00g			
54.00g	-23.00g	-225.00g			
43.00g	-2.00g	-315.00g			
-30.00g	167.00g	-209.00g			
-55.00g	168.00g	-133.00g			
-72.00g	235.00g	-80.00g			
-107.00g	211.00g	r -4.00g			
10.00g	175.00g	-268.00g			
18.00g	-16.00g	-244.00g			
24.00g	-78.00g	-230.00g			
					¥
Autosc	roll 🗌 Sho	w timestamp	Newline $\sim$ 9600 baud $\sim$	Clear	r output

Press the key for eighth time and the remainder is 0. Then the RGB will flash. If you press keys incessantly, remainders will change in a loop way. So does functions.

# 6.3.21 Project 65: WiFi

#### Description

In the previous experiment, we have learned the WiFi Station mode, WiFi AP mode and WiFi AP+Station mode of the ESP32. In this project, We will use ESP32's WiFi Station mode to control the work of multiple sensors/modules through APP connection with WiFi to achieve the effect of WiFi smart home.

#### Components



#### Wiring Diagram



### Install APP

(1) Android device (mobile phone/PC) APP:

A. We provide the Android APP installation package:

ې <mark>ت 🗸</mark> ۹۹	Search 5.Librarie
Date modified	Туре
11/10/2023 2:17 PM	File folder
11/13/2023 8:42 AM	File folder
11/13/2023 8:42 AM	File folder
11/10/2023 1:41 PM	File folder
11/10/2023 1:46 PM	File folder
	PP C C C C C C C C C C C C C C C C C C

B. Now transfer the **keyes wifi.apk** file in the Android APP installation package to the Android phone or PC, click the **keyes wifi.apk** file to enter the installation page, click "**ALLOW**" key, and then click "**INSTALL**" button. After installation, click "**OPEN**" button to enter the APP interface.

oboa	rd	Organize	New	Open		Sele	ect	
↑	🔤 « 5.Librar	ies_Driver_Firmware_and_APP	Android APP	~	ē	,⊂ Se	arch Android A	PP
^	Name	^	Date modified		Тур	e	Size	
	📄 keyes w	ifi.apk	8/30/2021 4:13	PM	AP	K File	2,362 KB	]



keyes wifi.apk







Webpage not available	192.168.1.148	WIFI	
		$\mathbf{A}$	
1		3	
4		6	

(2) IOS device (mobile phone /iPad) APP:

A. Open App Store



B. Enter keyes link in the search box and click search, the download interface appears. Click " V" to download and install the APP of the keyes link. The following operations are similar to those of Android system. You can refer to the steps of Android system above for operation.

#### Test Code

Note: You need to change the Wifi name and default Wifi password of the experimental code to your own Wifi name and Wifi password.

```
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233"; //the password of user's wifi
```

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>
#include "xht11.h"
//gpio15
xht11 xht(27);
unsigned char dht[4] = {0, 0, 0, 0};
#include <ESP32Servo.h>
Servo myservo;
int servoPin = 21;
#define Relay 4
#define IN1 2 //IN1 corresponds to IN+
#define IN2 15 //IN2 corresponds to IN-
#define trigPin 12
#define echoPin 13
int distance1;
String dis_str;
int ip_flag = 1;
int ultra_state = 1;
int temp_state = 1;
int humidity_state = 1;
String item = "0";
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233"; //the password of user's wifi
WiFiServer server(80);
String unoData = "";
void setup() {
  Serial.begin(115200);
  pinMode(Relay, OUTPUT);
  myservo.setPeriodHertz(50);
 myservo.attach(servoPin, 500, 2500);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
```

```
digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(Relay, LOW);
  pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
}
void loop() {
  WiFiClient client = server.available();
  if (!client) {
     return:
  }
  while(client.connected() && !client.available()){
      delay(1);
  }
  String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1) {
      Serial.print("Invalid request: ");
      Serial.println(req);
     return;
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")
  {
      IPAddress ip = WiFi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +...

→String(ip[3]);

      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
→Hello from ESP32 at ";
      s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);
  }
  else if(req == "/btn/0")
  {
   Serial.write('a');
   client.println(F("turn on the relay"));
   digitalWrite(Relay, HIGH);
  }
  else if(req == "/btn/1")
  {
   Serial.write('b');
   client.println(F("turn off the relay"));
   digitalWrite(Relay, LOW);
  }
```

```
(continued from previous page)
```

```
else if(req == "/btn/2")
{
 Serial.write('c');
 client.println("Bring the steering gear over 180 degrees");
 myservo.write(180);
 delay(200);
}
else if(req == "/btn/3")
{
 Serial.write('d');
 client.println("Bring the steering gear over 0 degrees");
 myservo.write(0);
 delay(200);
}
else if(req == "/btn/4")
{
 Serial.write('e');
 client.println("esp32 already turn on the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
}
else if(req == "/btn/5")
{
 Serial.write('f');
 client.println("esp32 already turn off the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
}
else if(req == "/btn/6")
{
 Serial.write('g');
 while(Serial.available() > 0)
  {
   unoData = Serial.readStringUntil('#');
    client.println("Data");
 }
 while(ultra_state>0)
     {
        Serial.print("Distance = ");
        Serial.print(checkdistance());
        Serial.println("#");
        Serial1.print("Distance = ");
        Serial1.print(checkdistance());
        Serial1.println("#");
        int t_val1 = checkdistance();
        client.print("Distance(cm) = ");
        client.println(t_val1);
        ultra_state = 0;
      }
}
else if(req == "/btn/7")
{
```

```
Serial.write('h');
 client.println("turn off the ultrasonic");
 ultra_state = 1;
}
else if(req == "/btn/8")
{
 Serial.write('i');
 while(Serial.available() > 0)
   {
   unoData = Serial.readStringUntil('#');
    client.println(unoData);
   }
 while(temp_state>0)
    {
      if (xht.receive(dht)) {
        Serial.print("Temperature = ");
        Serial.print(dht[2],1);
        Serial.println("#");
        Serial1.print("Temperature = ");
        Serial1.print(dht[2],1);
        Serial1.println("#");
        int t_val2 = dht[2];
        client.print("Temperature(°) = ");
        client.println(t_val2);
      }
      temp_state = 0;
   }
}
else if(req == "/btn/9")
{
 Serial.write('j');
 client.println("turn off the temperature");
 temp_state = 1;
}
else if(req == "/btn/10")
{
  Serial.write('k');
 while(Serial.available() > 0)
   {
     unoData = Serial.readStringUntil('#');
     client.println(unoData);
  }
 while(humidity_state > 0)
    {
      if (xht.receive(dht)) {
        Serial.print("Humidity = ");
        Serial.print(dht[0],1);
        Serial.println("#");
        Serial1.print("Humidity = ");
        Serial1.print(dht[0],1);
        Serial1.println("#");
        int t_val3 = dht[0];
```

```
client.print("Humidity(%) = ");
        client.println(t_val3);
      }
      humidity_state = 0;
     }
 }
 else if(req == "/btn/11")
 {
   Serial.write('1');
   client.println("turn off the humidity");
   humidity_state = 1;
   }
 //client.print(s);
 client.stop();
}
int checkdistance() {
 digitalWrite(12, LOW);
 delayMicroseconds(2);
 digitalWrite(12, HIGH);
 delayMicroseconds(10);
 digitalWrite(12, LOW);
 int distance = pulseIn(13, HIGH) / 58;
 delay(10);
 return distance;
}
```

### **Test Result**

After the code has been modified correctly, connect the external power supply and power on. Switch the DIP switch ON the ESP32 expansion board to the ON end, compile and upload the code to the ESP32 mainboard.If uploading the

code is not successfulpress the Boot button on the ESP32 mainboard with your hand after click , release it when the upload progress percentage appears.)



Open the serial monitor and set baud rate to 115200, then the monitor prints the detected WiFi IP address. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)



💿 СОМЗ			_		$\times$
					Send
					^
Connected to ChinaNet-2.4G-0DF0					
IP address: 192.168.0.156					
TCP server started					
					~
Autoscroll 🗌 Show timestamp	Newline $\sim$	115200 baud	~	Clear	output

Open WiFi APP, enter the detected WIFI IP address in the text box in front of the WIFI button (for example, the IP address detected by the serial monitor above is 192.168.0.156). Next, click the WIFI button to connect to WIFI, at the same time, the corresponding WiFi IP address will be displayed in the text box :"Hello from ESP32 at 192.168.0.156", then the APP has connected to WiFi.(WiFi IP address sometimes changes, if the original IP address can not use, you need to re-check it.)



After the APP is connected to WiFi, the following operations are performed:





## CHAPTER

# SEVEN

# **ARDUINO(RASPBERRY-PI) TUTORIAL**

# 7.1 1. Install Raspberry Pi OS System

# 7.1.1 1.1. Hardware Tool

- Raspberry Pi 4B/3B/2B
- Above 8G TFT SD Card
- Card Reader
- Computer and other parts

# 7.1.2 1.2. Software Tool

## Windows System

### (1) Install putty:

Download Putty: https://www.chiark.greenend.org.uk/~sgtatham/putty/



# **PuTTY: a free SSH and Telnet client**

Home | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team Download: Stable · Snapshot | Docs | Changes | Wishlist

PuTTY is a free implementation of SSH and Telnet for Windows and Unix platforms, along with an xterm terminal emulator. It is written and maintained primarily by <u>Simon Tatham</u>.

The latest version is 0.74 **Download it here**.

**LEGAL WARNING**: Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. We believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but we are not lawyers, and so if in doubt you should seek legal advice before downloading it. You may find useful information at <u>cryptolaw.org</u>, which collects information on cryptography laws in many countries, but we can't vouch for its correctness.

Use of the Telnet-only binary (PuTTYtel) is unrestricted by any cryptography laws.

# Latest news

## 2020-11-22 Primary git branch renamed

The primary branch in the PuTTY git repository is now called main, instead of git's default of master. For now, both branch names continue to exist, and are kept automatically in sync by a symbolic-ref on the server. In a few months' time, the alias master will be withdrawn.
🛃 Download P	uTTY: latest release ( ) × +	
$\leftarrow \   \rightarrow \   G$	chiark.greenend.org.uk/~sgtatham/putty/latest.html	ବ 🖈 \varTheta

## Download PuTTY: latest release (0.74)

<u>Home</u> | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team Download: Stable - Snapshot | Docs | Changes | Wishlist

This page contains download links for the latest released version of PuTTY. Currently this is 0.74, released on 2020-06-27.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a <u>permanent link to the 0.74 release</u>.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the <u>development snapshots</u>, to see if the problem has already been fixed in those versions.

Package file	25			
You probably (Not sure when	want one of these. They include versions of ther you want the 32-bit or the 64-bit version	f all the PuTTY utili on? Read the <u>FAQ er</u>	ties. <u>ntry</u> .)	
MSI ('Windo	ws Installer')			
32-bit:	putty-0.74-installer.msi	(or by FTP)	(signature)	
64-bit:	putty-64bit-0.74-installer.msi	<u>(or by FTP)</u>	(signature)	
Unix source a	rchive			
.tar.gz:	<u>putty-0.74.tar.gz</u>	<u>(or by FTP)</u>	(signature)	

After downloading the driver file putty-64bit-0.74-installer double-click it and tap "Next".

Putty release 0.74 (64-bit) Setup Wizard Putty release 0.74 (64-bit) Setup Wizard Putty release 0.74 (64-bit) Setup Wizard Putty release 0.74 (64-bit) features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Click "Next".

🛃 PuTTY release 0.74 (64-bit) Setup	_		$\times$
Destination Folder			P
Click Next to install to the default folder or click Change to	o choose another.	Ĺ	₫,
Install PuTTY release 0.74 (64-bit) to:			
C:\Program Files\PuTTY\			
Change			
Back	Next	Cano	el

Select "Install Putty files" and click "Install".

🔀 PuTTY release 0.74 (64-bit) Setup	_		$\times$
Product Features Select the way you want features to be installed.		ĺ	P
		2	2
Add shortcut to PuTTY on the Desktop  Put install directory on the PATH for command prompts			
Associate .PPK files with PuTTYgen and Pageant			
This feature requires 3914KB on your hard drive.			
Back 💱 Install		Cano	el

After a few seconds, click "Finish".



### (2) SSH Remote Login software -WinSCP

Download WinSCP: https://winscp.net/eng/download.php





Follow the below steps to finish the installation.





Setup - WinSCP 5.17.9		$\times$
Ready to Install Setup is now ready to begin installing WinSCP on your computer.	Ę	
Click Install to continue with the installation, or click Back if you want to review or change any settings.		
Destination location: C:\Program Files (x86)\WinSCP	^	
Setup type: Typical installation		
Selected components: WinSCP application Drag & drop shell extension (allows direct downloads, may require restart) Pageant (SSH authentication agent) PuTTYgen (key generator) Translations		
Additional tasks: Enable automatic check for application updates (recommended) Enable collecting anonymous usage statistics Create a desktop icon Add upload shortcut to Explorer's 'Send to' context menu Register to handle URL addresses	~	
< >		
Help < Back Install	Cance	1



## (3) SD Card Formatter

Format TFT card tool

Download SD Card Formatter

http://www.canadiancontent.net/tech/download/SD\_Card\_Formatter.html

### keyestudio WiKi



If you're receiving a 404 File Not Found error, this means the publisher has taken the file offline and has not updated their links with us for SD Card Formatter. Please do drop us a note in the event of a missing file.

Unzip the SDCardFormatterv5	_WinEN package, double-click SD Card Formatter 5.0.1 Setup.exe to run it.
docs/4.Arduino_C_Tutoria	al(Raspberry-Pi)/media/046c67e4072093ee3dad27e8088fcf9f.png
docs	/4.Arduino_C_Tutorial(Raspberry-Pi)/media/13dc08ae2b5cb52ae3d7ea198134d778.png
SD Card Formatter - InstallShiel	Id Wizard
く	Preparing to Install SD Card Formatter Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.
	Extracting: SD Card Formatter Setup.msi
	Cancel

🚼 SD Card Formatter - InstallSł	nield Wizard X
2	Welcome to the InstallShield Wizard for SD Card Formatter
	The InstallShield(R) Wizard will install SD Card Formatter on your computer. To continue, click Next.
	WARNING: This program is protected by copyright law and international treaties.
	< Back Next > Cancel
SD Card Formatter - InstallS	aidd Wittard
w so cara contractor installo	
License Agreement Please read the following licen:	se agreement carefully.
License Agreement Please read the following license ENI	se agreement carefully.
License Agreement Please read the following licen  NOTICE: BY DOWNLOADING, INSTALLIN ENTERING INTO THIS AGREEM IF YOU DO NOT AGREE WITH OR USE THE PRODUCT; PRON SDA OR YOUR SDA DISTRIBUT ACQUIRE ANY LICENSE TO USE	A se agreement carefully.
License Agreement Please read the following licen  NOTICE: BY DOWNLOADING, INSTALLIN ENTERING INTO THIS AGREEM IF YOU DO NOT AGREE WITH OR USE THE PRODUCT; PROM SDA OR YOUR SDA DISTRIBUT ACQUIRE ANY LICENSE TO USE  I accept the terms in the license	se agreement carefully.
License Agreement Please read the following licen  NOTICE: BY DOWNLOADING, INSTALLIN ENTERING INTO THIS AGREEM IF YOU DO NOT AGREE WITH OR USE THE PRODUCT; PRON SDA OR YOUR SDA DISTRIBUT ACQUIRE ANY LICENSE TO USE  I accept the terms in the licens I do not accept the terms in the	se agreement carefully.
License Agreement Please read the following licen  ENI NOTICE: BY DOWNLOADING, INSTALLIN ENTERING INTO THIS AGREEM IF YOU DO NOT AGREE WITH OR USE THE PRODUCT; PRON SDA OR YOUR SDA DISTRIBUT ACQUIRE ANY LICENSE TO USE I accept the terms in the licens I do not accept the terms in the InstallShield	se agreement carefully.

Click "Next" and "Install".

B Card Formatter - Instalishield Wiz	ard		$\times$
Destination Folder Click Next to install to this folder, or clic	k Change to install	to a different folde	
Install SD Card Formatter to: C:\Program Files (x86)\SDA\S	D Card Formatter\	i.	Change
InstallShield	< Back	Next >	Cancel
P Card Formatter - InstallShield Wig	ard		~
Ready to Install the Program The wizard is ready to begin installation			3
If you want to review or change any of exit the wizard.	your installation s	ettings, dick Back. (	Click Cancel to
Current Settings:			
Setup Type:			
Setup Type: Typical			
Setup Type: Typical			
Setup Type: Typical Destination Folder: C:\Program Files (x86)\SDA\SD Car	'd Formatter\		
Setup Type: Typical Destination Folder: C:\Program Files (x86)\SDA\SD Car User Information: Name:	d Formatter\		
Setup Type: Typical Destination Folder: C:\Program Files (x86)\SDA\SD Car User Information: Name: Company:	d Formatter\		
Setup Type: Typical Destination Folder: C:\Program Files (x86)\SDA\SD Car User Information: Name: Company:	d Formatter\		

After a few seconds, click "Finish".

👹 SD Card Formatter - InstallSh	ield Wizard	$\times$
	InstallShield Wizard Completed The InstallShield Wizard has successfully installed SD Card Formatter. Click Finish to exit the wizard.  ✓ Launch the program	
	< Back Finish Cancel	

### (4) Burn Win32DiskImager

Download Linkhttps://sourceforge.net/projects/win32diskimager/

Home/Browse/S	ystem Adminis	stration / Storage /	Win32 Disk Image	r					PROJECT
Win32 Disk Imager								3	MONTH MAR 2014
A Windows tool for writing images to USB sticks or SD/CF cards									
	Brought to you by: gruemaster, tuxinator2009								
****	★ 112	Reviews		Downle	oads: 42,251 This W	/eek	I	.ast Update: 20	18-06-07
Get Updates Share This									
Summary	Files	Reviews	Support	Wiki	Feature Requests	Bugs	Code	Mailing Lists	Blog

This program is designed to write a raw disk image to a removable device or backup a removable device to a raw image file. It is very useful for embedded development, namely Arm development projects (Android, Ubuntu on Arm, etc). Anyone is free to branch and modify this program. Patches are always welcome.

This release is for Windows 7/8.1/10. It will should also work on Windows Server 2008/2012/2016 (although not tested by the developmers). For Windows XP/Vista, please use v0.9 (in the files archive).

After the download, double-click win32diskimager-1.0.0-install.exe and tap "Run".

 $\times$ 

>SmartScreen can't be reached right now
Check your Internet connection. Windows Defender SmartScreen is unreachable and can't help you decide if this app is ok to run.
Publisher: Unknown Publisher App: win32diskimager-1.0.0-install.exe
Run Don't Run

Select I accept the agreement and tap "Next".

🗞 Setup - Win32Disklmager	_		$\times$
License Agreement Please read the following important information before continuing.		¢	
Please read the following License Agreement. You must accept the t agreement before continuing with the installation.	erms of t	nis	_
This program is licensed under the GNU GPL Version 2 License. Inclu are licensed under GPL v2 and LGPL v2.1 accordingly.	uded librar	ries ^	
GNU GENERAL PUBLIC LICENSE Version 2, June 1991			
Copyright (C) 1989, 1991 Free Software Foundation, Inc.,		v	
<ul> <li>I accept the agreement</li> <li>I do not accept the agreement</li> </ul>			
Nex	t >	Can	ncel

 $Click \ ``Browse...'' \ and \ find \ out \ the \ folder \ where \ the \ Win 32DiskImager \ is \ located, \ tap \ ``Next'' \ .$ 

酸 Setup - Win32DiskImager	—		$\times$
Select Destination Location Where should Win32DiskImager be installed?			Ð
Setup will install Win32DiskImager into the following folder.			
To continue, click Next. If you would like to select a different folder, c	lick Bro	wse.	
C:\Program Files (x86)\ImageWriter	Bro	owse	
At least 44.2 MB of free disk space is required.			
< Back Next	>	Ca	ncel

👒 Setup - Win32Disklmager			_		×
Select Start Menu Folder Where should Setup place the program's short	tcuts?			) ©	Ì
Setup will create the program's short	cuts in the fol	lowing Start	t Menu	folder.	
To continue, dick Next. If you would like to se	lect a differen	it folder, die	ck Brow	ise.	
Image Writer			Brow	wse	
[	< Back	Next >		Cance	el
Tick Create a desktop shortcut , click "Next" and	"Install".				
😻 Setup - Win32Disklmager		-	_		$\times$
Select Additional Tasks Which additional tasks should be performed?					D
Select the additional tasks you would like Setu Win32DiskImager, then click Next.	p to perform v	vhile installir	ng		
Additional shortcuts:					
Create a desktop shortcut					
	< Back	Next >		Cance	el



The installation is finished.

### (5) WNetWatcher

Scan to search ip address software tool-WNetWatcher

Download Linkhttp://www.nirsoft.net/utils/wnetwatcher.zip

### (6) Raspberry Pi Imager

### Download Address

https://www.raspberrypi.org/downloads/raspberry-pi-os/

Old Version:

Raspbianhttps://downloads.raspberrypi.org/raspbian/images/ Raspbian fullhttps://downloads.raspberrypi.org/ raspbian\_full/images/ Raspbian litehttps://downloads.raspberrypi.org/raspbian\_lite/images/ We use the 2020.05.28 version in the tutorial and recommend you to use this version. (Please download this version as shown in the picture below.) https://downloads.raspberrypi.org/raspios\_full\_armhf/images/raspios\_full\_armhf-2021-05-28/

# Index of /raspios\_full\_armhf/images/raspios\_full\_armhf-2021-05-28

	Name	Last modified		<u>Size</u>	<b>Description</b>
-	Parent Directory			-	
?	2021-05-07-raspios-buster-armhf-full.info	2021-05-07 16:2	23 2	88K	_
Ð	2021-05-07-raspios-buster-armhf-full.zip	2021-05-07 16:3	35 2	2.8G	
?	2021-05-07-raspios-buster-armhf-full.zip.sha1	2021-05-28 15:4	19	83	-
?	2021-05-07-raspios-buster-armhf-full.zip.sha256	2021-05-28 15:4	9	107	
?	2021-05-07-raspios-buster-armhf-full.zip.sig	2021-05-28 15:0	00	488	
?	2021-05-07-raspios-buster-armhf-full.zip.torrent	2021-05-28 15:5	<b>0</b>	28K	



# 7.1.3 1.3. Install Raspberry Pi OS on Raspberry Pi 4B:

Format TFT RAM card with SD Card Formatter software, as shown below:

SD Card Forn	natter	×
File Help		
Select card		
E:\-boot		~
		Refresh
Card information	n	
Туре	SDXC	52
Capacity	59.48 GB	XC
Formatting optic	vns	
Overwrite for	rmat	
CHS format s	ize adjustment	
Volume label		
boot		
		Format
SD Logo	, SDHC Logo and SDXC Logo are t	trademarks of SD-3C, LLC.

SD Card Formatter	×
File Help	
Select card	
E:\-boot	~
	Refresh
SD Card Formatter	
Formatting will erase all Do you want to continue Note: As formatting can overwrite option is selec computer is connected to mode is disabled.	data on this card. e? take some time (especially when ted), please make sure that your o a power supply and that sleep
	Yes No
lbest	
boot	
	Format
SD Logo, SDHC Logo and SD	XC Logo are trademarks of SD-3C, LLC.
SD Card Formatter	×
Select card	
E:\-boot	~
	Defeat
SD Card Formatter	× —
Card Type Capa Form Form Q Q O O	uccessfully completed. on: B (63,831,015,424 bytes) GB (63,830,622,208 bytes) kilobytes ot
Volun boot	ОК
SD Logo, SDHC Logo and SD	Format XC Logo are trademarks of SD-3C, LLC.

# (1) Burn System

Burn the Raspberry Pi OS system to TFT card using Win32DiskImager software

💖 Win32 Disk Imager - 1.0	Choose th <del>e</del> correct letter
Image File	Devic
inistrator/Desktop/2020-12-02-ras	pios-buster-armhf-full.img 📔 [E:\] 🔻
Hash	0
None - Generate Copy	Click, and then find the mirror
	download and unzinned
Read Only Allocated Partitions	download and unzipped
Progress	
Click Write	to write the system
Cancel Read Writ	te Verify Only Exit
Waiting for a task.	
👒 Win32 Disk Imager - 1.0	- 🗆 X
Win32 Disk Imager - 1.0	— 🗆 🗙 Device
Win32 Disk Imager - 1.0 Image File inistrator/Desktop/2020-12-02-rasp	→ □ × Device pios-buster-armhf-full.img 📄 [E:\] ▼
Win32 Disk Imager - 1.0 Image File inistrator/Desktop/2020-12-02-rasp Hash Confirm overwrite - 1.0	— □ × Device pios-buster-armhf-full.img ≧ [E:\] ▼ ×
Win32 Disk Imager - 1.0 Image File inistrator/Desktop/2020-12-02-rasp Hash None None Writing to a phys (Target Device: [E Are you sure you	□ × Device pios=buster=armhf=full.img  [] [E:\] ▼ (E:\] ▼ (ical device can corrupt the device. (X) "boot") want to continue?
Win32 Disk Imager - 1.0  Image File  ini strator/Desktop/2020-12-02-rasp Hash None None None None Read Read	- □ × Device pios=buster=armhf=full.img
Win32 Disk Imager - 1.0  Image File  inistrator/Desktop/2020-12-02-rasp Hash None Confirm overwrite - 1.0 None Writing to a phys (Target Device: [E Are you sure you Read ( Progres:	—       □       ×         pios=buster=armhf=full.img       [E:\] ▼         [X:\] "boot")       ×         want to continue?
Win32 Disk Imager - 1.0  Image File  inistrator/Desktop/2020-12-02-rasp Hash None Confirm overwrite - 1.0 None None None None None None None None	□ × Device pios=buster=armhf=full.img ≧ [E:\] ▼ Exical device can corrupt the device. (X) "boot") want to continue? Yes No
Win32 Disk Imager - 1.0  Image File  inistrator/Desktop/2020-12-02-rasp Hash Confirm overwrite - 1.0  Writing to a phys (Target Device: [E Are you sure you Read Progres: Cancel Read Writ	- □ × Device pios=buster=armhf=full.img

👒 Win32 Disk Imager - 1.0	_	$\Box$ $\times$
Image File		Device
inistrator/Desktop/2020-12-02-raspios-buster-armhf	-full.img 📔	[E:\] ▼
Hash 💊 Complete - 1.0 🗙		
None  Generate Write Successful.		
Read Only Allocated ) OK Progress	ļ	
Cancel Read Write Verif	y Only	Exit
Done.		08:40/08:40

Don't eject card reader after burning mirror system, build a file named SSH, then delete .txt .

The SSH login function can be activated by copying SSH file to boot category, as shown below.

$\leftarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ $\blacksquare$ > This PC > boot (E:)				~ Õ	, Search boot
	^	Name	Date modified	Туре	Size
★ Quick access		start.elf	11/26/2020 5:30 PM	ELF File	2,869 KB
Desktop	*	start_cd.elf	11/26/2020 5:30 PM	ELF File	771 KB
👆 Downloads	*	start_db.elf	11/26/2020 5:30 PM	ELF File	4,674 KB
🖹 Documents	*	start_x.elf	11/26/2020 5:30 PM	ELF File	3,610 KB
E Pictures	*	start4.elf	11/26/2020 5:30 PM	ELF File	2,162 KB
This DC		start4cd.elf	11/26/2020 5:30 PM	ELF File	771 KB
Inis PC		start4db.elf	11/26/2020 5:30 PM	ELF File	3,627 KB
3D Objects		start4x.elf	11/26/2020 5:30 PM	ELF File	2,904 KB
Desktop		bcm2708-rpi-b.dtb	11/26/2020 5:30 PM	DTB File	25 KB
Documents		bcm2708-rpi-b-plus.dtb	11/26/2020 5:30 PM	DTB File	25 KB
🖶 Downloads		bcm2708-rpi-b-rev1.dtb	11/26/2020 5:30 PM	DTB File	25 KB
h Music		bcm2708-rpi-cm.dtb	11/26/2020 5:30 PM	DTB File	25 KB
Pictures		bcm2708-rpi-zero.dtb	11/26/2020 5:30 PM	DTB File	25 KB
Videos		bcm2708-rpi-zero-w.dtb	11/26/2020 5:30 PM	DTB File	26 KB
Windows10 1909 (Cr)		bcm2709-rpi-2-b.dtb	11/26/2020 5:30 PM	DTB File	26 KB
**************************************		bcm2/10-rpi-2-b.dtb	11/26/2020 5:30 PM	DIB File	26 KB
		bcm2/10-rpi-3-b.dtb	11/20/2020 5:30 PM	DTB File	28 KB
boot (E:)		bcm2710-rpi-s-b-plus.dtb	11/20/2020 5:30 PM	DTR File	20 ND
USB Drive (F:)		bcm2711-mi-4.h dth	11/26/2020 5:30 PM	DTR File	20 ND
🛬 New folder (\\desktop-eng) (Z:)		bcm2711-mi-400 dth	11/26/2020 5:30 PM	DTB File	47 KB
boot (E:)		bcm2711-mi-cm4.dtb	11/26/2020 5:30 PM	DTB File	47 KB
overlays		bootcode.bin	11/26/2020 5:30 PM	BIN File	52 KB
or only s		fixup.dat	11/26/2020 5:30 PM	DAT File	8 KB
USB Drive (F:)		fixup_cd.dat	11/26/2020 5:30 PM	DAT File	4 KB
Petwork		fixup_db.dat	11/26/2020 5:30 PM	DAT File	11 KB
DESKTOP-1V3IO2C		fixup_x.dat	11/26/2020 5:30 PM	DAT File	11 KB
		ifixup4.dat	11/26/2020 5:30 PM	DAT File	6 KB
		itxup4cd.dat	11/26/2020 5:30 PM	DAT File	4 KB
DESKIOP-BDC/IVI		itxup4db.dat	11/26/2020 5:30 PM	DAT File	9 KB
DESKTOP-CMPIKJB		📄 fixup4x.dat	11/26/2020 5:30 PM	DAT File	9 KB
DESKTOP-SMT1PCK		LICENCE.broadcom	9/30/2020 12:00 PM	BROADCOM File	2 KB
LYJ		COPYING.linux	5/27/2020 10:57 AM	LINUX File	19 KB
FTI69C9C26XOD0S		overlays	12/2/2020 12:39 PM	File folder	
HK4KUMDY9PBVSC0		SSH SSH	12/8/2020 11:48 AM	Text Document	0 KB 🗸
1014010100	~	<			>

Eject Card Reader

### (2) Log in system

(Raspberry and PC should be in the same local area network.)

1). Insert TFT memory card into Raspberry Pi, connect internet cable and plug in power. If you have screen and HDMI cable of Raspberry Pi, you could view Raspberry Pi OS activating. If not, you can enter the desktop of Raspberry Pi via SSH remote login software—WinSCP and xrdp.



2). Use the WNetWatcher software to find the IP address of the Raspberry Pi.

Wireless	Network Watcher
----------	-----------------

Wireless Network Water Strength W	atcher						-		$\times$
File Edit View Optio	ns Help								
🕨 🔲 🔚 🖿 🖌 🖆	n 🗐								
IP Address	Device Name	MAC	Address	Network Adapter Co	Device Information	User Text			
<b>a</b> 192.168.0.1		A4-1	A-3A-DF-24-7E	TP-LINK TECHNOLOG	Your Router	AUTOBVT	-LEFJQ	7B	
< .									>
7 item(s)		12	NirSoft Freeware, ht	tps://www.nirsoft.net					

If there is no IP address as shown in the figure above, follow the following steps to set it.

Wireless Network File Edit View Or	Watcher			- 0	×
	MAC Address Format				
IP Address	Put Icon On Tray Start As Hidden Show Advanced Options Automatically Tray Balloon On New Device	dapter Co CHNOLOG	Device Information Your Router	User Text AUTOBVT-LEFJQ7B	^
	Background Scan Clear ARP Cache On Every Scan Beep On New Device				
	Beep On Disconnected Device Scan On Program Start Automatically Copy Device Name To User Text Show Inactive Devices Show All Previous Devices				
	Always On Top Auto Size Columns On Every Scan Automatic Sort On Every Scan				
	Select Another Font Use Default Font				
	Device Options         Ctrl+F9           Advanced Options         F9	1			~
54 item(s)	74 NirSoft Freeware, https://w	w.nirsoft.net			

Advanced Options	$\times$
Use the following network adapter:	
[192.168.0.49] Realtek PCIe GbE Family Controller: 以太网	~
Scan the following IP addresses range:	
From: 192.168.0.1 To: 192.168.0.255	
Use the following audio file for new device notification:	
Use the following audio file for disconnected device notification:	
Execute the following command when a new device is detected:	
Execute the following command when a device is disconnected	
You can use the following variables in the command string:	
%device_name% %user_text% %adapter_company% %mac_addr% %ip_addr% %detect count%	
Activate the beep/tray alert/command-execution only if the device is detected in the	
Background scan interval: 0 Seconds	
Automatically export all items to a file every 30 Seconds	
File type: Comma Delimited Text File	
Filename:	
Export to a file only when there is a change since the previous export	
Always overwrite the previous file $\sim$ Generate filename with numeric counter	$\sim$
	-
OK	Cancel

Once the setup is complete, record the IP and MAC addresses of the Raspberry PI. As shown in the red box below, the MAC address of the Raspberry PI is b8:27:eb:17:16:01, and the ip address is 192.168.0.57.

Wireless Network W	Vatcher				- 0	×
File Edit View Optio	ons Help					
🕨 🔲 🖻 🛄 🗞 🖆	े 🔊 - <b>म</b>					
IP Address	Device Name	MAC Address	Network Adapter Company	Device Information	User Text	^
192.168.0.12		C8-BF-4C-D0-B4-0F	Beijing Xiaomi Mobile Software		ZHANG	
192.168.0.13	dddd-PC	08-60-6E-59-37-B5	ASUSTek COMPUTER INC.		dddd-PC	
192.168.0.14		6C-0B-84-07-45-F5	Universal Global Scientific Indus		WIN-28LBGL340	60
192.168.0.18	WIN-D92OC35Q3AS	30-5A-3A-52-25-01	ASUSTek COMPUTER INC.		WIN-D92OC35Q	3AS
192.168.0.19	AUTOBVT-LEFJQ7B	08-62-66-47-C9-9F	ASUSTek COMPUTER INC.		AUTOBVT-RE50V	/R4
192.168.0.20	po-PC	FC-AA-14-E9-47-90	GIGA-BYTE TECHNOLOGY CO., L		po-PC	
192.168.0.21		00-15-5D-00-99-0D	Microsoft Corporation		ygjghj-PC	
192.168.0.22	ygjghj-PC	E0-D5-5E-6C-C6-82	GIGA-BYTE TECHNOLOGY CO., L		ygjghj-PC	
192.168.0.24	DESKTOP-NJT3RNC	F4-6B-8C-01-EF-9C	Hon Hai Precision Industry Co.,		DESKTOP-NJT3R	NC
192.168.0.26	lin	F4-B5-20-14-59-A7	Biostar Microtech international		LIN	
192.168.0.27	DESKTOP-S73F5DH	40-8D-5C-BF-16-D8	GIGA-BYTE TECHNOLOGY CO., L		DESKTOP-8HL55	т
192.168.0.31	PZ-XJX	18-C0-4D-58-71-F7	GIGA-BYTE TECHNOLOGY CO., L		PZ-XJX	
192.168.0.32	DESKTOP-901C3HI	94-C6-91-58-4A-F8	EliteGroup Computer Systems C		DESKTOP-901C3	н
192.168.0.35	DESKTOP-8JF2T7R	F4-6B-8C-02-25-B0	Hon Hai Precision Industry Co.,		lifan	
192.168.0.36	PZ-IQC	18-C0-4D-90-20-AD	GIGA-BYTE TECHNOLOGY CO.,L		PZ-IQC	
192.168.0.38	DESKTOP-RE514OU	A8-A1-59-B1-13-22	ASRock Incorporation		DESKTOP-RE514	ou
192.168.0.40	DESKTOP-MHK2NO9	18-C0-4D-99-04-55	GIGA-BYTE TECHNOLOGY CO., L.,		DESKTOP-MHK2	NO9
192.168.0.57		B8-27-EB-17-16-01	Raspberry Pi Foundation			
192.168.0.42	DESKTOP-VSQORHH	40-B0-76-44-A6-06	ASUSTek COMPUTER INC.		DESKTOP-VSQO	RHH
192.168.0.43	shengchan-ling	F4-6B-8C-05-15-F2	Hon Hai Precision Industry Co.,		shengchan-ling	
192.168.0.45	XTZJ-20230106WH	14-DD-A9-EA-60-32	ASUSTek COMPUTER INC.		XTZJ-20230106W	/H
588 192 168 0 46 <	限设共享	56-6F-FR-94-00-03			閑☆共言	>
56 item(s)		115 NirSoft Freeware.	https://www.nirsoft.net			

If you do not know the mac address and the ip address of the Raspberry PI, then unplug the network cable of the Raspberry PI first, open the **WNetWatcher** query, and the detection times will be displayed on the right side of the interface. Connect the Raspberry PI cable and query it once using **WNetWatcher**, and the Raspberry PI address is detected one less time than the other addresses. Then write down the ip and mac addresses.

## (3) Remote Login

Enter default user name, password and host name on WinSCP to log in.

The same network only receives one Raspberry Pi.



🔁 Login	- 🗆 X
New Site	Session   File protocol:   SCP   Host name:   192.168.0.57   User name:   Password:   pi   Save   ▼     Advanced
<u>T</u> ools ▼ <u>M</u> anage ▼ Show Login dialog on startup and when the last s Warning	Session is closed
Continue connecting to an unhost key to a cache? The server's host key was not found in the server is the computer you think it The server's Ed25519 key details are:	the cache. You have no guarantee that is.
Algorithm: ssh-ed25519 255 SHA-256: 7PyYMa8IW2IQuwz4ofn MD5: 0f:90:dd:a3:61:22:58:93:b If you trust this host, press Yes. To com cache, press No. To abandon the conn	ndJMSJdzwE820gJI56gSRL95Y 1:46:24:31:a8:6e:89:28 nect without adding host key to the ection press Cancel.
<u>Y</u> es	No Cancel <u>H</u> elp

## (4) Check ip and mac address

🔁 Documents – pi@192.168.	0.26 - Wins	SCP						- 0	×
Local Mark Files Comman	ds Tabs	Options Remote H	lelp						
🕂 🤮 🐹 Synchronize 💿	💽 🖸	🔅 🕼 Queue	Transfer Settings De	fault	• 🧝 •				
📮 pi@192.168.0.26 🗙 📑	New Tab	Click i	it to open terr	mi	nal				
🖹 My documents 🔹 📁 🔹	<b>Y</b> - 1	() -   🖬 🖬	1 A Q		🔤 pi 🔹 🖬 🕈 🍸	• • • •	-> - 🗈 🖬 🏫 🖏	🗎 Find Files	÷.
🛗 Upload 👻 📝 Edit 🔹 🗲	A D	Properties -	New - + - V		Download + Z Edi	- ×	D Properties - 12 New		V
C:\Users\Administrator\Docum	nents\				/home/pi/				
Name	Size	Туре	Changed	^	Name	Size	Changed	Rights	Owner
t		Parent directory	4/20/2023 2:56:44 PM		t		5/7/2021 3:42:11 PM	DWXT-XT-X	root
360js Files		File folder	11/28/2022 7:29:39 PM		Bookshelf		5/7/2021 3:52:54 PM	rwxr-xr-x	pi
Arduino		File folder	7/21/2023 8:24:54 AM		Desktop		8/11/2023 2:15:21 AM	rwxr-xr-x	pi
Corel		File folder	3/30/2020 2:38:41 PM		Documents		8/11/2023 2:15:24 AM	rwxr-xr-x	pi
Fax		File folder	4/2/2020 3:58:17 PM		Downloads		8/11/2023 2:15:24 AM	rwxr-xr-x	pi
FormatFactory		File folder	7/14/2020 8:51:41 AM		Music		8/11/2023 2:15:24 AM	rwxr-xr-x	pi
Fritzing		File folder	4/20/2023 2:56:44 PM		Pictures		8/11/2023 2:15:24 AM	rwxr-xr-x	pi
M_PdfEdit		File folder	9/10/2021 1:33:50 PM		Public		8/11/2023 2:15:24 AM	rwxr-xr-x	pi
MakeCode		File folder	3/31/2021 11:22:04 AM		Templates		8/11/2023 2:15:24 AM	DWXT-XT-X	pi
Processing		File folder	12/16/2020 8:30:17 AM		Videos		8/11/2023 2:15:24 AM	DWXT-XT-X	pi
QQPCMgr		File folder	6/9/2021 5:49:13 PM						
Scanned Documents		File folder	4/2/2020 3:58:17 PM		The system	filos	for the raceh	orny ni	
SOLIDWORKS DownL		File folder	8/1/2022 2:25:37 PM		The system	mes	ior the raspu	eny pi	
Tencent Files		File folder	8/11/2023 10:10:44 AM						
Visual Studio 2019		File folder	12/17/2020 5:29:15 PM	¥					
¢			>		¢				
B of 4.82 MB in 0 of 24			7 hid	den	0 B of 0 B in 0 of 9				10 hidd
							S	P Q	0:00:59

Click to open terminal input the passwordraspberry, and press"Enter" on keyboard.



🗬 pi@raspberrypi: ~  $\times$ Using username "pi". pi@raspberrypi's password: Linux raspberrypi 5.4.51-v71+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv71 The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright. Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. Last login: Mon Oct 19 03:54:47 2020 SSH is enabled and the default password for the 'pi' user has not been changed. This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password. Wi-Fi is currently blocked by rfkill. Use raspi-config to set the country before use. pi@raspberrypi:~ 🖇

Logging in successfully, open the terminal, input ip a and tap"Enter"to check ip and mac address.

🗬 pi@raspberrypi: ~  $\times$ Wi-Fi is currently blocked by rfkill. Use raspi-config to set the country before use. pi@raspberrypi:~ \$ ip a 1: 1o: <LOOPBACK, UP, LOWER UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul t qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo valid\_lft forever preferred lft forever inet6 ::1/128 scope host valid lft forever preferred lft forever 2: eth0: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc mq state UP group defa ult qlen 1000 link/ether dc:a6:32:17:5b:cb brd ff:ff:ff:ff:ff:ff inet 192.168.0.57 24 brd 192.168.0.255 scope global dynamic noprefixroute et h0valid 1ft 3569sec preferred 1ft 2819sec inet6 fe80::977f:5abe:e49c:78c4/64 scope link valid lft forever preferred lft forever 3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qle n 1000 link/ether dc:a6:32:17:5b:cc brd ff:ff:ff:ff:ff:ff i@raspberrypi:~ 💲

### (5) Fix ip address of Raspberry Pi

Ip address is changeable, therefore, we need to make ip address fixed for convenient use.

Follow the below steps
Switch to root user
If without root user's password
Set root passward
Input passwordin the terminalsudo passwd root to set password
Switch to root user
Input su root
Fix the configuration file of ip address
Firstly change ip address of the following configuration file.
#New ip addressaddress 192.168.0.57
Copy the above new address to terminal and press"Enter".
Configuration File
echo -e '
auto eth0

iface eth0 inet static

HChange IP address

address 192.168.0.57

netmask 255.255.255.0

gateway 192.168.1.1

network 192.168.1.0

broadcast 192.168.1.255

dns-domain 119.29.29.29

dns-nameservers 119.29.29.29

metric 🛇

mtu 1492

\>/etc/network/interfaces.d/eth0

As shown below:



Reboot the system and activate the configuration file

Input the restart command in the terminal: sudo reboot

You could log in via fixed ip afterwards.

Check IP and insure ip address fixed well

```
pi@raspberrypi:~ $ ip a
1: 10: <LOOPBACK, UP, LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t glen 1000
    link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid lft forever preferred lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1492 qdisc mq state UP group defa
Sult glen 1000
    link/ether dc:a6:32:17:61:9c brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.57/24 brd 192.168.1.255 scope global eth0
       valid lft forever preferred lft forever
    inet 192.168.1.128/24 brd 192.168.1.255 scope global secondary dynamic nopre
fixroute eth0
       valid_lft 1730sec preferred_lft 1505sec
    inet6 fe80::1e7d:5653:59e9:3262/64 scope link
       valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qle
n 1000
    link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff
 pi@raspberrypi:~ 💲
```

### (6) Log in Desktop on Raspberry Pi Wirelessly

In fact, we can log in desktop on Raspberry Pi Wirelessly even without screen and HDMI cable.

VNC and Xrdp are commonly used to log in desktop of Raspberry Pi wirelessly.

#### Install Xrdp Service in the terminal

Installation commands:

Switch to Root User: su root

Install apt-get install xrdp

Enter y and press "Enter"
As shown below:

🖉 pi@raspberrypi: ~	_		×
pi@raspberrypi:~ \$ su root			^
Password:			
root@raspberrypi:/home/pi# apt-get install xrdp			
Reading package lists Done			
Building dependency tree			
Reading state information Done			
The following additional packages will be installed:			
xll-apps xll-session-utils xbitmaps xfonts-75dpi xfonts-base xf	onts-s	calable	
xorg xorg-docs-core xorgxrdp			
Suggested packages:			
mesa-utils xorg-docs xll-xfs-utils guacamole xrdp-pulseaudio-in	stalle	r	
The following NEW packages will be installed:			
xll-apps xll-session-utils xbitmaps xfonts-75dpi xfonts-base xf	onts-s	calable	
xorg xorg-docs-core xorgxrdp xrdp			
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.			
Need to get 10.9 MB of archives.			
After this operation, 17.4 MB of additional disk space will be us	ed.		
Do you want to continue? [Y/n] y			
Get:1 http://mirrors.sjtug.sjtu.edu.cn/raspbian/raspbian buster/m	ain ar	mhf xll	a
pps armhf 7.7+7 [541 kB]			
Get:2 http://mirrors.sjtug.sjtu.edu.cn/raspbian/raspbian buster/m	ain ar	mhf xll	-8
ession-utils armhf 7.7+3 [61.2 kB]			
Get:3 http://mirrors.zju.edu.cn/raspbian/raspbian buster/main arm	hf xbi	tmaps a	11
1.1.1-2 [32.1 kB]			$\sim$

#### (7) Open the remote desktop connection on Windows

Press WIN+R on keyboard and enter mstsc.exe As shown below Run × Type the name of a program, folder, document, or Internet resource, and Windows will open it for you. Open: mstsc.exe ×

Input ip address of Raspberry Pi, as shown below.

Click"Connect" and tap"Connect".

192.168.0.57 is ip address we use, you could change into yours ip address.

nemote Desktop Connection	—		$\times$
Remote Desktop Connection			
Computer: 192.168.0.57	~		
User name: None specified			
You will be asked for credentials when you connect.			
Show Options	onnect	He	lp
Click "Yes".			
Nemote Desktop Connection	$\times$		
The identity of the remote computer cannot be verified. Do you connect anyway?	i want to		
This problem can occur if the remote computer is running a version of Windows that is earlier than Windows Vista, or if the remote computer configured to support server authentication.	is not		
For assistance, contact your network administrator or the owner of the computer.	remote		
Don't ask me again for connections to this computer			

Input user name: pi, default password: raspberry, as shown below:

Login to raspberrypi		^
Just		
Session Xorg viewername pi password viewername		
OK Cancel		
<	>	

Click "OK" or "Enter", you will view the desktop of Raspberry Pi OS, as shown below:



Now, we finish the basic configuration of Raspberry Pi OS.

## 7.2 2. Preparations for C language:

C language is a programming language with a considerably fast running speed. There are numerous software and system core code written in it, such as Linux system. Notably, hardware MCU and embedded class are not exception. Thereby, it makes sense to learn the C language to control hardware.

### 7.2.1 2.1. Hardware

#### (1) Raspberry Pi 4B:



Hardware Interfaces



(2) ESP32 Expansion Board:



(3) Raspberry Pi+ESP32 mainboard+ESP32 Expansion Board+USB Cableare as follows



## 7.2.2 2.2. Copy Example Code Folder to Raspberry Pi:

Place example code folder to the pi folder of Raspberry Pi. and extract the example code from **ESP32\_C\_code(Raspberry-Pi).zip** file(the default is .zip file), as shown below:



			pi				~ ^ X
File Edit View Sort Go Tool	s						
📰   🔛   👬 🧱 📗	$\blacktriangleright \leftrightarrow \Rightarrow$	1/home/	/pi				•
A Home Folder						po]]p	*
Ellesystem Root	Arduino	Bookshelf	Desktop	Documents	Downloads	Music	Pictures
thinclient_drives     ≜		INTERN.			C .		
nome				0	Open	I	
👻 🏚 pi	Public	Templates	thinclient_dr	Videos	Archi	ver	
Arduino	2 2 2 2		IVes		ESP3 Open	With	
Bookshelf					Extra	ct To	
🔁 Desktop					Extra	ct Here	
T Documents					Cut		
Downloads					Сору		
Music					Move	e to Wastebas	ket
					Сору	Path(s)	0.0
"2. ESP32_C_code.zip" (80.3 KiB) 2	Lip archive				Free		GIB)
			pi				~ ^ X
<u>File Edit View Sort Go Tool</u>	S						
	$\blacktriangleright \leftrightarrow \rightarrow$	↑ /home/	/pi				-
觰 Home Folder				1729			ndin
🚐 Filesystem Root	ESD22	Arduino	Rooksholf	Desktop	Documente	Dowploada	Music
limit_drives	C code(Ras	Arduno	DOOKSHEII	Desktop	Documents	DOWINOAUS	WUSIC
nome	berry-Pi)	P					
🝷 📻 pi		111	NAME OF COLUMN		0	1	
▶ 2. ESP32_C_code	Dictures	Public	Tomplates	thinglight dr	Videon		
Arduino	Fictures	Public	Templates	ives	videos	:SP32	
Bookshelf					-	C_code(Ras	р
C Desktop					ł	perry-Pi).zip	
13 items (21 hidden)					Free space: 1	17.3 GiB (Total	l: 28.4 GiB)

Double-click ESP32\_C\_code(Raspberry-Pi), as shown below.

			2. ESP	32_C_code					~ ~	×
<u>File Edit View Sort Go Tool</u>	s									
📑   📑   👬 🧱 🔠	★ -	↑ /ho	me/pi/:ESP	32_C_cod	e(Raspber	ry-Pi)				•
🖌 Home Folder										Î
🕮 Filesystem Root	lesson	lesson	lesson	lesson	lesson	lesson	lesson	lesson	lesson	- 1
thinclient_drives	01. Hello World	02. LED	03. Traffic Li	04. Laser sensor	05. Breath	06. RGB	07. Button	08. Touch s	09. Obstacle	
▼ home										
🗢 🙍 pi	lesson 10. Line	lesson 11. Photo	lesson 12. Tilt	lesson 13.	lesson 14. Hall	lesson 15. Reed	lesson 16. PIR	lesson 17.	lesson 18.	
2. ESP32_C_code	tracking	Interrupt	switch	collision	sensor	Switch	motion	Active b	Passive	- 1
🕨 🚞 lesson 01. Hello '										- 1
Iesson 02. LED	lesson	lesson	lesson	lesson	lesson	lesson	lesson	lesson	lesson	- 1
Iesson 03. Traffic	19.130	20.	21.	22.	23.	24.	25. Film	26.	27. MQ-2	- 1
Iesson 04. Laser	Motor	Rotary p	Steam s	MicroPh	Photore	Tempera	pressure	Flame s		- 1
Iesson 05. Breath									lane and	- 1
lesson 06. RGB	28. MO-3	29. Five	iesson 30.	31. Relay	32	33.	34. Servo	iesson 35	36. IR	
Iesson 07. Buttor	20. 110	AD Keys	Joystick	ee,	SK6812	Rotary e		Ultrasonic	Receiver	
Iesson 08. Touch										
▶ 🔚 lesson 09. Obsta	lesson 37.	lesson 38.	lesson 39.	lesson 40.	lesson 41.	lesson 42. Matrix	lesson 43.	lesson 44. RFID	lesson 45.	
lesson 10. Line tr	DS18B20	XHT11	DS1307	ADXL345	TM1650		lcd128	RC522	Button c	
Iesson 11 Photo*										÷
65 items						F	Free space:	17.3 GiB (T	otal: 28.4 Gi	B)

## 7.3 3. Linux SystemRaspberry Pi:

## 7.3.1 3.1. Download and install Arduino IDE

1First, click on Raspberry Pi's browser.



2Download Arduino IDE from the Arduino official websitewww.arduino.cc/en/software, as shown below:



(3) There are various versions of IDE for Arduino. Just download a version compatible with your system. (install the lasted Arduino IDE 1.8.19) and click "Linux ARM 32 bits".

# Downloads



## Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the Getting Started page for Installation instructions.

#### SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

#### (4) You just need to click "JUST DOWNLOAD".



Windows Win 7 and newer Windows ZIP file

Windows app Win 8.1 or 10 Get 🚦

Linux 32 bits Linux 64 bits Linux ARM 32 bits Linux ARM 64 bits

Mac OS X 10.10 or newer

Release Notes

Checksums (sha512)



After a few seconds, the lasted Arduino IDEArduino 1.8.19 versionzip file can be directly downloaded.

(5) Click , then find the Downloads file from the pi and tap it. Then we can see the downloaded package "arduino-1.8.19-linuxarm.tar.xz" and unzip it.











## 7.3.2 3.2. Install the ESP32 on Arduino IDE

Noteyou need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.



sketch_apr06a   Arduino 1.8.19	~	^	×
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp			
		ø	
sketch_apr06a		•	
<pre>void setup() {     // put your setup code here, to run once:</pre>			Î
}			I
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>			I
}			I
			Ŧ
1 Ardu	iino I	Uno	

(2 Click "File"  $\rightarrow$  "Preferences" copy the website address https://dl.espressif.com/dl/package\_esp32\_index.json in the "Additional Boards Manager URLs:" and click "OK".

	sketch_a	apr06a   Arduino 1.8.19 🔹 🔺 🔺	×
<u>File</u> Edit <u>S</u> ketc	h <u>T</u> ools <u>H</u> elp		
New	Ctrl+N	Q.	
Open	Ctrl+O		
Open Recent	>	·	
Sketchbook	>	e here, to run once:	Î
Examples	>		ı
Close	Ctrl+W		ı
Save	Ctrl+S	here to run repeatedly:	ı
Save As	Ctrl+Shift+S	nere, to full repeateury.	I
Page Setup	Ctrl+Shift+P		ı
Print	Ctrl+P		Ŧ
Preferences	Ctrl+Comma		
Quit	Ctrl+Q		
1		Arduino Uno	

	Preferences 🗸 🔨
Settings Network	
Sketchbook location:	
/home/pi/Arduino	Browse
Editor language:	System Default
Editor font size:	12
Interface scale:	✓ Automatic 100 <sup>▲</sup> / <sub>♥</sub> % (requires restart of Arduino)
Theme:	Default theme    (requires restart of Arduino)
Show verbose output during:	compilation upload
Compiler warnings:	None 💌
<ul> <li>Display line numbers</li> <li>Verify code after upload</li> <li>Check for updates on start</li> <li>Use accessibility features</li> </ul>	<ul> <li>□ Enable Code Folding</li> <li>□ Use external editor</li> <li>up</li> <li>☑ Save when verifying or uploading</li> </ul>
Additional Boards Manager U	RLs: https://dl.espressif.com/dl/package_esp32_index.json 1
More preferences can be edite	d directly in the file
/home/pi/.arduino15/preferen	ces.txt
(edit only when Arduino is not	running) (2
	OK Cance

3 Click "Tools"  $\rightarrow$  "Board:" then click "Boards Manager..." to enter "Boards Manager". Enter "ESP32" as follows, then click "Install".

<u>Eile Edit Sketch</u> Tools <u>H</u> elp	
Auto Format Ctrl+T	
Archive Sketch	Boards Manager
void setun()	
// put your Manage Libraries Ctrl+Shift+I	Δ
Serial Monitor Ctrl+Shift+M	Arduino Yún
Serial Plotter Ctrl+Shift+L	<ul> <li>Arduino Uno</li> </ul>
void loop() { // put your WiFi101 / WiFiNINA Firmware Updater	Arduino Duemilanove or Diecimila Arduino Nano
} Board: "Arduino Uno"	Arduino Mera or Mera 2560
Port >	Arduino Mega ADK
Get Board Info	Arduino Mega ADK
	Arduino Leonardo
Programmer: "AVRISP mkli"	Arduino Leonardo ETH
Burn Bootloader	Arduino Micro
	Arduino Esplora
1 Arduino Uno	Arduino Mini
Boards Manager	~ ^ X
Type All esp32	
esp32	î
by Espressin Systems Boards included in this package:	
More Info	0
	1.0.6 Vinstali

Boards Manager	~	~ ×
Type All · esp32		
esp32		î
by Espressif Systems Boards included in this package: ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.		
More info	etelling	٦
	stalling	1
		- 1
		- 1
		- 1
		- 1
		- 1
		- 1
		- 1
		- 1
		ļ
Downloading boards definitions. Downloaded 23,526kb of 51,126kb.	Ca	ancel

(4) After installing, click "Close".

	Boards Manager 👻	~ ×
Type All 🔹 esp32		
esp32 by Espressif Systems version 1.0.6 INSTALLED Boards included in this package: ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32. <u>More info</u>		Î
		Close

## 7.3.3 3.3. Arduino IDE Setting



When downloading the sketch to the board, you must select the correct name of Arduino board that matches the board connected to your computer. As shown below: (Note: we use the ESP32 board in this tutorial; therefore, we select ESP32)



Then select the correct COM port (you can see the corresponding COM port after the ESP32 is connected to the

### Raspberry Pi via a USB cable.).

	sketch_apr06a   Arduino 1.8.19 🛛 🗸 🔺 🗙		-
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	Tools Help		
	Auto Format	Ctrl+T	
	Archive Sketch		
sketch_apr06	Fix Encoding & Reload		
<pre>void setup()    // put your</pre>	Manage Libraries	Ctrl+Shift+I	
	Serial Monitor	Ctrl+Shift+M	
}	Serial Plotter	Ctrl+Shift+L	
<pre>void loop() {     // put your</pre>	WiFi101 / WiFiNINA Firmware Updater		-
}	Board: "ESP32 Wrover Module"	>	
	Upload Speed: "921600"	>	
	Flash Frequency: "80MHz"	>	-
	Flash Mode: "QIO"	>	
	Partition Scheme: 'Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)	" >	
	Core Debug Level: "None"	>	
	Port	>	Serial ports
	Get Board Info		/dev/ttyAMA0
	Programmer	>	/dev/ttyUSB0
Default 4MB with :	Burn Bootloader		

sketch_apr06a   Arduino 1.8.19 🛛 🗸 🔺	×
<u>File Edit Sketch Tools H</u> elp	
sketch_apr06a	
<pre>void setup() {     // put your setup code here, to run once:</pre>	Î
}	1
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>	I
}	I
	I
	I
	Ŧ
1 spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0	

sketch_apr06a   Arduino 1.8.19 🛛 🗸 🔺 🗙	
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp	
sketch_apr06a	
// put your setup code here, to run once:	Ì
A B C D E	
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>	
}	
	•
n spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0	

A- Used to verify whether there is any compiling mistakes or not. B- Used to upload the sketch to your Arduino board. C- Used to create shortcut window of a new sketch. D- Used to directly open an example sketch. E- Used to save the sketch. F- Used to send the serial data received from board to the serial monitor.

## 7.4 4. How to Add Libraries? :

## 7.4.1 4.1. What are Libraries ?:

Librariesare a collection of code that make it easy for you to connect sensors, displays, modules, etc. For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. (https://www.arduino.cc/en/Reference/Libraries)

### 7.4.2 4.2. How to Install a Library ?:

Here we will introduce the most simple way to add libraries .



click"libraries" file libraries from the "arduino-1.8.19" file.







**Step 2:** Copy and paste the **Arduino\_C\_Libraries(Raspberry-Pi)** file (default .ZIP file) from the provided Arduino Libraries folder into the Libraries file opened in the first stepthe route is/home/pi/Downloads/arduino-1.8.19/libraries.

Clipboard	Organize	New	Open		
	ware_and_APP > Arduino_C_Libra	ries(Raspberry-f	Pi) 🗸	Ū	Q
Name	Date modif	ied	Туре	Size	
🏭 Adafruit_NeoPixel	4/6/2022 3:	12 PM	WinRAR ZIP		63 KB
adxI345_io	4/6/2022 3:	13 PM	WinRAR ZIP		4 KB
DS18B20	4/6/2022 3:	12 PM	WinRAR ZIP		3 KB
ESP32Servo	4/6/2022 3:	13 PM	WinRAR ZIP	1	24 KB
HT16K33_Lib_For_ESP32	2 4/6/2022 3:	14 PM	WinRAR ZIP		7 KB
🙀 IRremoteESP8266	4/6/2022 3:	42 PM	WinRAR ZIP	6	62 KB
🚾 lcd128_32_io	4/6/2022 3:	42 PM	WinRAR ZIP		5 KB
MFRC522_12C	4/6/2022 3:	43 PM	WinRAR ZIP	1	27 KB
🚾 Rtc_by_Makuna	4/6/2022 3:	43 PM	WinRAR ZIP	4	46 KB
1M1650	4/7/2022 11	:06 AM	WinRAR ZIP		4 KB
🙀 Wire	4/6/2022 3:	43 PM	WinRAR ZIP		14 KB
🚾 xht11	4/6/2022 3:-	43 PM	WinRAR ZIP		3 KB



Step 3: Unzip the Arduino C package in the libraries folderfor exampleclick "Adafruit\_NeoPixel.zip"file

## Adafruit\_Ne

oPixel.zip select and tap"Extract Here"to unzip the "Adafruit\_NeoPixel.zip"file.

#### keyestudio WiKi



		libra	aries			~ ^ X
File Edit View Sort Go	Tools					
	🝖 ←	ightarrow  /h	ome/pi/Dov	vnloads/arduin	o-1.8.19/libra	ries 🔹
🔒 Home Folder	GSM	кеуроаго	Liquiacryst	ta Mouse	Robot_Contr ol	ote
🚐 Filesystem Root						
👝 thinclient_drives 🛛 🔺						
r IIdiuwale 🛓	Robot_Moto	SD	Servo	SpacebrewY	Stepper	Temboo
▶ java	r	_		un		
▶ lib			10	Open		3
👻 📃 libraries 📘	TFT	WiFi	Adafrui	Archiver	).zi	ESP32Servo
👂 🔚 Adafrui			oPixel	Open With		.zip
Bridge		- 10 A	- 100	Extract To		
Esplora	HT16K33_Li	IRremoteES	lcd128	Extract Here	lak	TM1650.zip
Etherne	b_For_ESP3	P8266.zip	o.zij	Cut	)	
▶ 📄 exampl	2.zip			Сору		
Firmata		10 m		Move to Waste	basket	
▶ GSM -	Wire.zip	xht11.zip		Copy Path(s)		I
"Adafruit_NeoPixel.zip" (62.6 H	<ib) archive<="" td="" zip=""><td>9</td><td></td><td>Rename</td><td>В (1</td><td>Total: 58.2 GiB)</td></ib)>	9		Rename	В (1	Total: 58.2 GiB)



## 7.5 5. Basic Projects

When we get the kit, we can see that there are 42 sensors/modules in the kit, which contain the corresponding ESP32 mainboard, ESP32 Expansion Board and wirings. Here, we will connect the 42 sensors individually to the ESP32 mainboard and the ESP32 Expansion Board using wirings. Then run the corresponding test code to test the function of each sensor separately. Our next lesson is to study the principles of individual modules/sensors from simple to complex as well as some extended applications of sensors to consolidate and deepen our understanding of the kits.

Note: When connecting the module/sensor wirings in the projects, the wiring method and position must be followed in the document. What's more, do not misconnect the power supply and signal pin, otherwise there may be no experimental results or damage to the modules/sensors.

## 7.5.1 Project 1: Hello World

#### Overview

For ESP32 beginners, we will start with some simple things. In this project, you only need a ESP32 mainboard, a USB cable and Raspberry Pi to complete the "Hello World!" project, which is a test of communication between the ESP32 mainboard and the Raspberry Pi as well as a primary project.

#### Wiring Diagram

In this project, we will use a USB cable to connect the ESP32 to Raspberry Pi.



#### **Test Code**

```
/*
* Filename : Hello World
* Description : Enter the letter R, and the serial port displays"Hello World".
* Auther
         :http//www.keyestudio.com
*/
char val;// defines variable "val"
void setup()
{
Serial.begin(9600);// sets baudrate to 9600
}
void loop()
{
 if (Serial.available() > 0) {
  val=Serial.read();// reads symbols assigns to "val"
  if(val=='R')// checks input for the letter "R"
  { // if so,
   Serial.println("Hello World!");// shows "Hello World !".
  }
 }
}
```

Before uploading the test code to the ESP32click "Tools"  $\rightarrow$  "Board" select "ESP32 Wrover Module".

	HelloWorld   Arduino 1.8.19		~ ^ X		
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	<u>Tools H</u> elp				Contraction of the second s
	Auto Format	Ctrl+T	<b>10</b>		
	Archive Sketch				
HelloWorld	Fix Encoding & Reload			-	Δ
//*	Manage Libraries	Ctrl+Shift+I	· · · · · ·		ESP32 Dev Module
* Filename	Serial Monitor	Ctrl+Shift+M			<ul> <li>ESP32 Wrover Module</li> </ul>
* Auther	Serial Plotter	Ctrl+Shift+L			ESP32 Pico Kit
<pre>*/ char val;// d</pre>	WiFi101 / WiFiNINA Firmware Updater				TinyPICO
<pre>void setup()</pre>					S.ODI Ultra v1
{ Serial.begin	Board: ESP32 wrover Module	,	Boards Manage	er	MagicBit
}	Upload Speed: "921600"	>	Arduino AVR B	oards >	Turta IoT Node
Void Loop()	Flash Frequency: "80MHz"	>	ESP32 Arduino		TTGO LoBa32-OLED V1
if (Serial	Flash Mode: "QIO"	>			TTGO TI
val=Seria if(val==	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>			TTCO T7 V1 2 Mini22
2.(141	Core Debug Level: "None"	>		A.M.	TTGO T7 V1.5 WIIII52
	Port: "/dev/ttyUSB0"	>		and the second s	11G0 17 V1.4 Mini32
Invalid libra	Get Board Info		eaders f		XinaBox CW02
Invalid libra			iders fil		SparkFun ESP32 Thing
4	Programmer	>		-	SparkFun ESP32 Thing Plus
1 ESP32 Wr	Burn Bootloader		v/ttyUSB0		u-blox NINA-W10 series (ESP32)

#### Select the correct serial port

	HelloWorld   Arduino 1.8.19		~ ^	×	
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	Tools Help				
	Auto Format	Ctrl+T	3	<b>D</b>	
	Archive Sketch				
HelloWorld	Fix Encoding & Reload				
//*********	Manage Libraries	Ctrl+Shift+I	****	Î	
* Filename	Serial Monitor	Ctrl+Shift+M			
* Description	Serial Plotter	Ctrl+Shift+L			
<pre>*/ char val;// c void cotup()</pre>	WiFi101 / WiFiNINA Firmware Updater				
{	Board: "ESP32 Wrover Module"	>			
Serial.begin(	Upload Speed: "921600"	>			
void loop()	Flash Frequency: "80MHz"	>			
if (Serial	Flash Mode: "QIO"	>			
val= <mark>Seria</mark> if(val==	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>			
	Core Debug Level: "None"	>		-	wh
	Port: "/dev/ttyUSB0"	>	Seria	al por	ts
Invalid libra Invalid libra	Get Board Info		/dev	/ttyAl	MAO
	Programmer	>	✓ /dev	/ttyU	SB0
1 ESP32 Wr	Burn Bootloader		v/ttyUSB	0	1944 (M) 1944 (M) 1944 (M)

💿 HelloWorld   Ardu	ino 1.8.16				_		×	
File Edit Sketch Too	ls Help							
	Auto Format	Ctrl+T					Ø	
	Archive Sketch							
HelloWorld	Fix Encoding & Reload							
//********	Manage Libraries	Ctrl+Shift+I		***	******	*****	****	^
/*	Serial Monitor	Ctrl+Shift+M						
* Descriptio	Serial Plotter	Ctrl+Shift+L		lay	s"Hello	World"		
* Auther								
*/	WiFi101 / WiFiNINA Firmwa	are Updater						
void setup()	Board: "ESP32 Wrover Mod	ule"	>					
{	Upload Speed: "921600"		;	•	921600	]		
Serial.begin(	Flash Frequency: "80MHz"		;		115200	1		
void loop()	Flash Mode: "QIO"		;		256000			
{	Partition Scheme: "Default"		;		230400			
if (Serial.	Core Debug Level: "None"		;		512000			
if(val=='	Port: "COM3"		>					
{ // if	Get Board Info							
Serial.p				-				
}	Programmer		>					
<	Burn Bootloader						>	
Dana Antina								
Done Saving.	ound in Ct\llease\ldate		diar		bardura			
Invalid library f	ound in C:\Users\Admin	istrator\Desktop\Arc	du:	ino\	hardware	e\espre	essi	^
								¥
<							>	
2	ESP32	Wrover Module, Default, QIO	, 80	)MH2	z, 921600, N	lone on (	сомз	
Click 💽 to upload th	e test code to the ESP32.							


Note: If the uploading code fails, you can press and hold the Boot button on the ESP32 after clicking and release the Boot button after the percentage of uploading progress appears, as shown below:



The code is uploaded successfully

HelloWorld   Arduino 1.8.19	~ ^ X
<u>File Edit Sketch T</u> ools <u>H</u> elp	
	<b>₽</b>
HelloWorld	
//*************************************	* * *
<pre>* Filename : Hello World * Description : Enter the letter R,and the serial port displays"Hello World". * Auther :http//www.keyestudio.com */</pre>	
<pre>char val;// defines variable "val" void setup() {</pre>	
Serial.begin(9600);// sets baudrate to 9600	
} void loop() {	- 1
<pre>if (Serial.available() &gt; 0) {    val=Serial.read();// reads symbols assigns to "val"    if(val=='R')// checks input for the letter "R"</pre>	
Done uploading.	
Invalid library found in /home/pi/Downloads/arduino-1.8.19/libraries/examples: no he Invalid library found in /home/pi/Downloads/arduino-1.8.19/libraries/TM1650: no head	aders f≜ ers fil
	•
1 ESP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/	ttyUSB0

## **Test Result**

After uploading successfullywe will use a USB cable to power onclicked, set the baud rate to 9600we need to press the reset button on the ESP32 motherboard and enter the letter "R"click "Send"then the serial monitor prints "Hello World!".

/0	dev/ttyUSB0	~	~ X
R 2			Send
Ϋ́Hello World! Hello World!		1	3
Autoscroll 🗌 Show timestamp	Newline • 9600	baud 🔻 Clear	output

# 7.5.2 Project 2: Lighting up LED



#### Overview

In this kit, we have a Keyestudio Purple Module, which is very simple to control. If you want to light up the LED, you just need to make a certain voltage across it.

In the project, we will control the high and low level of the signal end S through programming, so as to control the LED on and off.

### Working Principle

The two circuit diagrams are given.

The left one is wrong wiring-up diagram. Why? Theoretically, when the S terminal outputs high levels, the LED will receive the voltage and light up.

Due to limitation of IO ports of ESP32 board, weak current can't make LED brighten.

The right one is correct wiring-up diagram. GND and VCC are powered up. When the S terminal is a high level, the triode Q1 will be connected and LED will light up(note: current passes through LED and R3 to reach GND by VCC not IO ports). Conversely, when the S terminal is a low level, the triode Q1 will be disconnected and LED will go off.



## Components



Wiring Diagram



## **Test Code**

(continues on next page)

```
* Description : led Flashing 1 s
* Auther : http://www.keyestudio.com
*/
int ledPin = 0; //Define LED pin connection to GPI00
void setup() {
    pinMode(ledPin, OUTPUT);//Set mode to output
}
void loop() {
    digitalWrite(ledPin, HIGH); //Output high level, turn on led
    delay(1000);//Delay 1000 ms
    digitalWrite(ledPin, LOW); //Output low level,turn off led
    delay(1000);//Delay 1000 ms
}
```

#### **Code Explanation**

1). PinMode(pin,mode): Pin is the ESP32 GPIO pin number used to set the mode, here we set pin 0 as output mode.

2). DigitalWrite(pin, value): Pin is the GPIO pin, which is defined GP0 here. Valueis the digital level that we will outputHIGH/LOW. If the pin is configured to OUTPUT using pinMode(), its voltage is set to the corresponding value: 3.3V is HIGH, low level is 0V (ground). When connect the LEDs to the pins, using the digitalWriteHIGH, the LEDs will get dim.

3). Setup() executes once, while loop() executes all the time. Delay (ms) is delay function, ms is the number of milliseconds to pause. Data type: unsigned longrange  $0 \sim 4,294,967,295$  ( $2^32 - 1$ ).

4). Firstly, we connect the module signal to ledPIN, namely GP0, and set it to a high level to light the LEDs on the module. Then delay 1000 ms, controlling the LEDs on the module light up for 1s and off for 1s to achieve the flashing effect.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the LED in the circuit will flash alternately.

## 7.5.3 Project 3: Traffic Lights Module



#### Overview

In this lesson, we will learn how to control multiple LED lights and simulate the operation of traffic lights.

Traffic lights are signal devices positioned at road intersections, pedestrian crossings, and other locations to control flows of traffic.

In this kit, we will use the traffic light module to simulate the traffic light.

#### **Working Principle**

In previous lesson, we already know how to control an LED. In this part, we only need to control three separated LEDs. Input high levels to the signal R(3.3V), then the red LED will be on.



Components



Wiring Diagram



//************************************
/* * Filename : Traffic Light
* Description : Simulated traffic lights
* Auther : http://www.keyestudio.com
int redPin = 15: //Red LED connected to GPT015
int yellowPin = 2; //Yellow LED connected to GPI02
<pre>int greenPin = 0; //Green LED connected to GPIO0</pre>
<pre>void setup() {     //LED interfaces are set to output mode     pinMode(greenPin, OUTPUT);     pinMode(yellowPin, OUTPUT);     pinMode(redPin, OUTPUT); }</pre>
<pre>void loop() {     digitalWrite(greenPin, HIGH); //Lighting green LED     delay(5000); //Delay for 5 seconds     digitalWrite(greenPin, LOW); //Turn off green LEDS     for (int i = 1; i &lt;= 3; i = i + 1) { //run three times         digitalWrite(yellowPin, HIGH); //Lighting yellow LED</pre>

(continues on next page)

## **Code Explanation**

Create pins, set pins mode and delayed functions.

We use the function for(). for (int i = 1;  $i \le 3$ ; i = i + 1) represents the variable i adds 1 fir each time from 1 to 3.

The function for (int i = 255; i >= 0; i = i - 1) indicates that i reduces by 1 each time. When i<0, exit the for() loop and execute 256 times.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the green LED will be on for 5s then off, the yellow LED will flash for 3s then go off and the red one will be on for 5s then off, the three LED modules will simulate the circulation of traffic lights automatically .

## 7.5.4 Project 4: Laser Sensor



#### Description

Lasers are widely used to cut, weld, surface treat, and more on specific materials. The energy of the laser is very high. The toy laser pointer may cause glare to the human eye, and it may cause retinal damage for a long time. my country also prohibits the use of laser to illuminate the aircraft.

#### **Working Principle**

The laser head sensor module is mainly composed of a laser head with a light-emitting die, a condenser lens, and a copper adjustable sleeve. We can see the circuit schematic diagram of this module which is very similar to the LED

we have learned. They are all driven by triodes. A high-level digital signal is directly input at the signal end, then the sensor will start to work; if inputting low levels, the sensor won't work.



## Components



#### **Connection Diagram**



```
/*
* Filename
        : Laser sensor
* Description : Laser light flashing
        : http://www.keyestudio.com
* Auther
*/
int laserPin = 0; //Define the laser pin as GPIO 0
void setup() {
 pinMode(laserPin, OUTPUT);//Define laser pin as output mode
}
void loop() {
 digitalWrite(laserPin, HIGH); //Open the laser
 delay(2000); //Delay 2 seconds
 digitalWrite(laserPin, LOW); //Shut down the laser
 delay(2000); //Delay 2 seconds
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the laser module will emit red laser signals for 2 seconds and stop emitting signals for 2 seconds on a cycle.

# 7.5.5 Project 5: Breathing LED



## Overview

A"breathing LED" is a phenomenon where an LED's brightness smoothly changes from dark to bright and back to dark, continuing to do so and giving the illusion of an LED"breathing. This phenomenon is similar to a lung breathing in and out. So how to control LED's brightness? We need to take advantage of PWMyou can refer to experiment six.

## Components



**Connection Diagram** 



```
/*
* Filename : Breathing Led
* Description : Make led light fade in and out, just like breathing.
* Auther
          : http//www.keyestudio.com
*/
              0 //define the led pin
#define PIN_LED
#define CHN
              0 //define the pwm channel
#define FRQ
             1000 //define the pwm frequency
                 //define the pwm precision
#define PWM_BIT
             8
void setup() {
 ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel
 ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel
}
void loop() {
 for (int i = 0; i < 255; i++) { //make light fade in
   ledcWrite(CHN, i);
   delay(10);
 }
 for (int i = 255; i > -1; i--) { //make light fade out
   ledcWrite(CHN, i);
   delay(10);
 }
}
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the LED on the module gradually gets dimmer then brighter, cyclically, like human breathe.

# 7.5.6 Project 6: RGB Module



#### Overview

Among these modules is a RGB module. It adopts a F10-full color RGB foggy common cathode LED. We connect the RGB module to the PWM port of MCU and the other pin to GND(for common anode RGB, the rest pin will be connected to VCC). So what is PWM?

PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output. In general, the input voltages of ports are 0V and 5V. What if the 3V is required? Or a switch among 1V, 3V and 3.5V? We cannot change resistors constantly. For this reason, we resort to PWM.

For Arduino digital port voltage outputs, there are only LOW and HIGH levels, which correspond to the voltage outputs of 0V and 5V respectively. You can define LOW as "0" and HIGH as "1", and let the Arduino output five hundred "0" or "1" within 1 second. If output five hundred "1", that is 5V; if all of which is "0", that is 0V; if output 250 01 pattern, that is 2.5V.

This process can be likened to showing a movie. The movie we watch are not completely continuous. Actually, it generates 25 pictures per second, which cannot be told by human eyes. Therefore, we mistake it as a continuous process. PWM works in the same way. To output different voltages, we need to control the ratio of 0 and 1. The more '0' or '1' output per unit time, the more accurate the control.

#### Working Principle

For our experiment, we will control the RGB module to display different colors through three PWM values.



Components



**Connection Diagram** 



```
/*
* Filename : RGB LED
* Description : Use RGBLED to show random color.
* Auther : http//www.keyestudio.com
*/
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
const byte chns[] = {0, 1, 2}; //define the pwm channels
int red, green, blue;
void setup() {
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit</pre>
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
 red = random(0, 256);
 green = random((0, 256);
 blue = random(\emptyset, 256);
 setColor(red, green, blue);
 delay(200);
}
void setColor(byte r, byte g, byte b) {
```

(continues on next page)

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onwe will see that the RGB LED on the module starts to display random colors.

# 7.5.7 Project 7: Button Sensor



### Overview

In this kit, there is a Keyestudio single-channel button module, which mainly uses a tact switch and comes with a yellow button cap.

In previous lessons, we learned how to make the pins of our single-chip microcomputer output a high level or low level. In this experiment, we will read the high level (3.3V) and low level (0V).

We can determine whether the button on the sensor is pressed by reading the high and low level of the S terminal on the sensor.

## Working Principle

The button module has four pins. The pin 1 is connected to the pin 3 and the pin 2 is linked with the pin 4. When the button is not pressed, they are disconnected. Yet, when the button is pressed, they are connected. If the button is released, the signal end is high level.



Components



**Connection Diagram** 



```
/*
* Filename : button
* Description : Read key value
* Auther : http://www.keyestudio.com
*/
int val = 0; //Useto store key values
int button = 15; //The pin of the button is connected to GP15
void setup() {
 Serial.begin(9600); //Start the serial port monitor and set baud rate to 9600
 pinMode(button, INPUT); //Set key pin to input mode
}
void loop() {
 val = digitalRead(button); //Read the value of the key and assign it to the variable.
→val
 Serial.print(val); //Print it on the serial port
 if (val == 0) { //Press the key to read the low level and print the press related.
→information
   Serial.print("
                       ");
   Serial.println("Press the botton");
   delay(100);
 }
 else { //Print information about key release
   Serial.print("
                       ");
```

(continues on next page)

#### **Code Explanation**

1). pinMode(button, INPUT); set the pin of the button module to GP15 and INPUT.

Configure INPUT through pinMode(). INPUT must use the pull-up or pull-down resistor(ours module has the pull-up resistor R1).

2). Serial.begin(9600): Initialize serial communication and set the baud rate to 9600.

3). **digitalRead(button)**: read the digital level of the button(HIGH or LOW). If this pin is not connected to pins, the digitalRead() will return HIGH or LOW.

4). if...else...if the logic behind () is true, execute the code of (); otherwise execute the code of else.

5). If the button is pressed, the signal end is low level, GP15 is low level and Val is 0. Then the monitor will show the corresponding value and characters; otherwise, the sensor is released, val is 1 and monitor will show 1 and other characters

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters. When the button is pressed, val is 0, the monitor will show "Press the button" when the button is released, val is 1 the monitor will show "Loosen the button"; as shown below

		/dev/ttyUSB1		×
			Ser	nd
1	Loosen the botton			•
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
0	Press the botton			
0	Press the botton			
0	Press the botton			
0	Press the botton			
0	Press the botton			
0	Press the botton			
Θ	Press the botton			
0	Press the botton			
0	Press the botton			
				Ŧ
•				•
	Autoscroll 🗌 Show timestamp	Newline 🔹 9600 baud 💌 Clea	ar outp	ut

# 7.5.8 Project 8: Capacitive Sensor



### Description

In this kit, there is a capacitive touch module which mainly uses a TTP223-BA6 chip. It is a touch detection chip, which provides a touch button, and its function is to replace the traditional button with a variable area button. When we power on, the sensor needs about 0.5 seconds to stabilize. Do not touch the keys during this time period. At this time, all functions are disabled, and self-calibration is always performed. The calibration period is about 4 seconds. We display the test results in the shell.

#### **Working Principle**

When our fingers touch the module, the signal S outputs high levels, the red LED on the module flashes. We can determine if the button is pressed or not by reading high and low levels on the sensor.



**Required Components** 



**Connection Diagram** 



#### **Test Code**

```
* Auther
             : http://www.keyestudio.com
*/
int val = 0;
int touch = 15; //The key of PIN
void setup() {
 Serial.begin(9600);//Baud rate is 9600
 pinMode(touch, INPUT);//Setting input mode
}
void loop() {
 val = digitalRead(touch);//Read the value of the key
 Serial.print(val);//Print out key values
 if (val == 1) {//Press for high level
                       ");
   Serial.print("
   Serial.println("Press the button");
   delay(100);
 }
 else {//Release to low level
   Serial.print("
                       ");
   Serial.println("Loosen the button");
   delay(100);
 }
}
```

### **Code Explanation**

When we touch the sensor, the Shell monitor will show "Pressed the button!", if not, "Loosen the button!" will be shown on the monitor.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters. when the button is pressed, the red LED lights up and val is 1. Then the shell shows "Pressed the button!"; if the button is released, the red LED is off and val is 0, "Loosen the button!" will be displayed.

	/dev/ttyUSB1	~	^	×
			Sei	nd
0	Press the botton			•
0	Press the botton			
0	Press the botton			
0	Press the botton			
0	Press the botton			
Θ	Press the botton			
0	Press the botton			
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
1	Loosen the botton			
				-
•				•
•	Autoscroll 🗌 Show timestamp Newline 💌 9600 baud 💌	Clear	outp	but

# 7.5.9 Project 9: Obstacle Avoidance Sensor



## Overview

In this kit, there is a Keyestudio obstacle avoidance sensor, which mainly uses an infrared emitting and a receiving tube. In the experiment, we will determine whether there is an obstacle by reading the high and low level of the S terminal on the sensor.

## Working Principle

NE555 circuit provides IR signals with frequency to the emitter TX, then the IR signals will fade with the increase of transmission distance. If encountering the obstacle, it will be reflected back.

When the receiver RX meets the weak signals reflected back, the receiving pin will output high levels, which indicates the obstacle is far away. On the contrary, it the reflected signals are stronger, low levels will be output, which represents the obstacle is close. There are 2 potentiometers on the sensor, and by adjusting the 2 potentiometers, we can adjust its effective distance.



## Components



**Connection Diagram** 



```
//
_ *****
                  /*
   * Filename : Touch sensor
   * Description : Reading touch value
   * Auther : http://www.keyestudio.com
   */
  int val = 0;
  int touch = 15; //The key of PIN
  void setup() {
    Serial.begin(9600);//Baud rate is 9600
    pinMode(touch, INPUT);//Setting input mode
  }
  void loop() {
    val = digitalRead(touch);//Read the value of the key
    Serial.print(val);//Print out key values
    if (val == 1) {//Press for high level
      Serial.print("
                         ");
      Serial.println("Press the button");
      delay(100);
    }
    else {//Release to low level
      Serial.print(" ");
      Serial.println("Loosen the button");
      delay(100);
```

(continues on next page)



#### **Code Explanation**

Note:

Upload the test code and wire up according to the connection diagram. After powering on, we start to adjust the two potentiometers to sense distance.

#### Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters. When the sensor detects the obstacle, the val is 0,the monitor will show"There are obstacles"; if the obstacle is not detected, the val is 1,"All going well" will be shown.

	/dev/ttyUSB1	- ^	×
		S	end
1 1 1 1 1 0	All going well All going well All going well All going well All going well All going well There are obstacles		•
	There are obstacles There are obstacles There are obstacles There are obstacles There are obstacles There are obstacles There are obstacles		ļ
•	Autoscroll 🗌 Show timestamp Newline 🔹 9600 baud 💌 Clea	r ou	• tput

# 7.5.10 Project 10: Line Tracking Sensor



#### Description

In this kit, there is a DIY electronic building block single-channel line tracking sensor which mainly uses a TCRT5000 reflective black and white line recognition sensor element.

In the experiment, we judge the color (black and white) of the object detected by the sensor by reading the high and low levels of the S terminal on the module; and display the test results on the shell.

#### **Working Principle**

When a black or no object is detected, the signal terminal will output high levels; when white object is detected, the signal terminal is low level; its detection height is 0-3cm. We can adjust the sensitivity by rotating the potentiometer on the sensor. When the potentiometer is rotated, the sensitivity is best when the red LED on the sensor is at the critical point between off and on.



**Required Components** 



**Connection Diagram** 



Test Code

(continues on next page)

```
* Description : Reading the tracking sensor value
* Auther
             : http://www.keyestudio.com
*/
int val = 0;
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(15, INPUT);//Sets sensor pin to input mode
}
void loop() {
 val = digitalRead(15);//Read the digital level output by the patrol sensor
 Serial.print(val);//Serial port print value
 if (val == 0) {//White val is 0 detected
   Serial.print("
                       ");
   Serial.println("White");
   delay(100);
 }
 else {//Black val is 1 detected
   Serial.print("
                       ");
   Serial.println("Black");
   delay(100);
 }
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters. when the sensor doesn't detect an object or detects a black object, the val is 1, and the monitor will display "1 Black"; when a white object (can reflect light) is detected, the val is 0, and the monitor will display "0 White";

	/dev/ttyUSB1	~	^	×
			Sen	nd
1	Black			•
1	Black			
0	White			
Θ	White			
0	White			
				Ŧ
•			1	•
•	Autoscroll 🗌 Show timestamp Newline 🔹 9600 baud 💌 Cle	ear c	outp	ut

# 7.5.11 Project 11: Photo Interrupter



## Description

This kit contains a photo interrupter which mainly uses 1 ITR-9608 photoelectric switch. It is a photoelectric switch optical switch sensor.

## **Working Principle**

When the paper is put in the slot, C is connected with VCC and the signal end S of the sensor are high levels; then the red LED will be off. Otherwise, the red LED will be on.



## **Required Components**



**Connection Diagram** 



```
/*
* Filename : Photo_Interrupt
* Description : Light snap sensor counting
* Auther
           : http://www.keyestudio.com
*/
int PushCounter = 0; //The count variable is assigned an initial value of 0
int State = 0; //Store the current state of the sensor output
int lastState = 0; //Stores the state of the last sensor output
void setup() {
 Serial.begin(9600);//Set the baud rate to 9600
 pinMode(15, INPUT);//Set the light snap sensor pin to input mode
}
void loop() {
 State = digitalRead(15);//Read current state
 if (State != lastState) {//If the state is different from the last read
   if (State == 1) {//block the light
    PushCounter = PushCounter + 1;//Count + 1
    Serial.println(PushCounter);//Print count
   }
 }
 lastState = State;//Update state
}
```

## **Code Explanation**

Logic setting:

Initial Setting	Set PushCounter to 0Set State to 0 (value of the sensor)Set lastState to 0	
when an object enters the slot	lastState is 0State turns into 1; lastState turns into 1	Set PushCounter to Push- Counter+1print the value of Push- Counter
when the object leaves the slot	lastState is 1State becomes 0two data are not equallastState turns into 0.	PushCounterdoesn't change;Don't print the value of PushCounter
When the object goes through this slot again	lastState is 0, State becomes 1two data are not equallastState turns into 1.	SetPushCounter to PushCounter+1. And print the value of PushCounter
When the object leaves this slot again	lastState is 1State turns into 0two data are not equal lastState turns into 0	PushCounter doesn't change;Don't print the PushCounter value

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the PushCounter data. Every time when the object passes through the slot of the sensor, the PushCounter data will increase by 1 continuously, as shown below;

/dev.	/ttyUSB1				~	^	×
						Sei	nd
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
10							
				_			-
✓ Autoscroll	Newline	•	9600 baud	•	Clear	outp	out

## 7.5.12 Project 12: Tilt Module



#### Overview

In this kit, there is a Keyestudio tilt sensor. The tilt switch can output signals of different levels according to whether the module is tilted. There is a ball inside. When the switch is higher than the horizontal level, the switch is turned on, and when it is lower than the horizontal level, the switch is turned off. This tilt module can be used for tilt detection, alarm or other detection.

#### Working Principle

The working principle is pretty simple. When pin 1 and 2 of the ball switch P1 are connected, the signal S is low level and the red LED will light up; when they are disconnected, the pin will be pulled up by the 4.7K R1 and make S a high level, then LED will be off.



#### Components



**Connection Diagram** 



Test Code

```
* Description : Reading the tilt sensor value
* Auther
            :http://www.keyestudio.com
*/
int val; //Store the level value output by the tilt sensor
void setup() {
 Serial.begin(9600);
 pinMode(15, INPUT); //Connect the pin of the tilt sensor to GP15 and set GP15 to the.
→input mode
}
void loop() {
 val = digitalRead(15); //Read module level signal
 Serial.println(val); //Newline print
 delay(100); //Delay for 100 ms
}
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then make the tilt module incline to one side, the red LED on the module will be off and the monitor will display "1". In contrast, if you make it incline the other side, the red LED will light up and the monitor will display "0".

/dev/t	tyUSB1				×
				Sen	d
Θ					٠
Θ					
0					
Θ					
0					
1					
1					
1					
1					
1					
1					
1					
1					Ŧ
✓ Autoscroll	Newline 🔻	9600 baud 🔻	Clear	outp	ut

(continued from previous page)
# 7.5.13 Project 13: Collision Sensor



## Description

The collision sensor uses a tact switch. This sensor is often used as a limit switch in 3D printers. In the experiment, we judge whether the sensor shrapnel is pressed down by reading the high and low levels of the S terminal on the module; and, we display the test results in the shell.

## **Working Principle**

It mainly uses a tact switch. When the shrapnel of the tact switch is pressed, 2 and 3 are connected, the signal terminal S is low level, and the red LED on the module lights up; when the touch switch is not pressed, 2 and 3 are not connected, and 3 is pulled up to a high level by the 4.7K resistor R1, that is, the sensor signal terminal S is a high level, and the built-in red LED will be off at this time.



**Components Required** 



**Connection Diagram** 



```
Test Code
```

```
/*
* Filename : collision sensor
* Description : Reading the value of the collision sensor
* Auther : http://www.keyestudio.com
*/
int val = 0;
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(15, INPUT);//Set collision sensor pin 15 to input mode
}
void loop() {
 val = digitalRead(15);//Read the value of the collision sensor
 Serial.print(val);//Newline print
 if (val == 0) {//Collision val is 0
   Serial.print("
                      ");
   Serial.println("The end of his!");
   delay(100);
 }
 else {// No collision val is 1
   Serial.print("
                    ");
   Serial.println("All going well");
   delay(100);
 }
```

(continues on next page)

(continued from previous page)

#### **Test Result**

}

1

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters.

\*\*\*\*\*\*

In the experiment, when the shrapnel on the sensor is pressed down, val is 0, the red LED of the module is on, and "0 The end of his!" is printed; when the shrapnel is released, the val is 1, the red LED of the module is off, and "1 All going well" is printed. "!" character, as shown below.

	/dev/ttyUSB1	~	<b>^</b> :	×
			Sen	d
1	All going well			•
1	All going well			
1	All going well			
1	All going well			
1	All going well			
0	The end of his!			
0	The end of his!			
0	The end of his!			
0	The end of his!			
0	The end of his!			
0	The end of his!			
Θ	The end of his!			
Θ	The end of his!			L
0	The end of his!			
				Ŧ
41			•	
•	Autoscroll 🗌 Show timestamp Newline 🔻 9600 baud 🔻 Clea	ar o	utpu	Jt

## 7.5.14 Project 14: Hall Sensor



### Description

In this kit, there is a Hall sensor which mainly adopts a A3144 linear Hall element. The element P1 is composed of a voltage regulator, a Hall voltage generator, a differential amplifier, a Schmitt trigger, a temperature compensation circuit and an open-collector output stage. In the experiment, we use the Hall sensor to detect the magnetic field and display the test results on the shell.

### **Working Principle**

When the sensor detects no magnetic field or a north pole magnetic field, the signal terminal will be high level; when it senses a south pole magnetic field, the signal terminal will be low levels. The stronger the magnetic field strength is, induction distance is longer.



**Required Components** 



## **Connection Diagram**

_																						
GND		•					•									7	•	К		1		
3.3V							E		E	E	-			ø		E		WE		(6-9)		
8	3	2			-	<b>0</b>		2	2	9	4		2	2	-	0	<b>.</b>	РС		Ö		
	102	102	TXD	RXD	102	101	101	<u>0</u>	101	101	Q	0	0	101	SD	SD	CL					
											:			1	Ŀ.	Ξ.	Q	C1			0 < 0	
E		:		a		20 M 32		4	io I							-						
	7	1		ц И	ר א גער א	ESPWR			ŭ		f	-		•		-	-	- U1				,
	н.										- 27				1	_ 1						
			) ES		<u>ا</u> و				Ľ	1		_		111		- 1					+ + <b>-</b>	2
	ו		ME) ES		ء ر ال	FCC 90:2AC7	•		keule	•		-	:		т. 1.	-		C3				
	1	- F	MF) ES		∍ ر ⊕	FCC 9D:2AC7	•		kerle	•		-	:		": . 11-	- '	þ				U W	
						FCC 90-2AC7	•		Kerle	•	- -		:					C3 C2				
		svp	SVN			<b>1032</b>				1027	1014 📵 - 📑	1012 🕕 : -	1013 🕕 🕇 💼 :	sd2	so3 🗻			C3 C2	R1			
S S		SVP (				<b>1032</b>				<ul> <li>IO27 (1)</li> </ul>	- Io14 📵 - 🚆 : -	- IO12 🕕	- Io13 🕕 🕇 -	<ul> <li>sd2</li> <li>sd2</li> </ul>					R1			
S 3.3V		SVP 0							П 1026 🕕 keule	<b>II I IO27</b>	<b>1 1 1014 (1) 1 1</b>	a 1012 🗊 : 🗧 -	🖬 💼 1013 📵 🕇 📼 :						R1	ON		

### **Test Code**

(continues on next page)

```
(continued from previous page)
 * Description : Reading the value of hall magnetic sensor
 * Auther
             : http://www.keyestudio.com
*/
int val = 0;
int hallPin = 15; //Hall sensor pin is connected to GPI015
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(hallPin, INPUT);//Set pin to input mode
}
void loop() {
 val = digitalRead(hallPin);//Read the level value of hall sensor
 Serial.print(val);//Print val
 if (val == 0) {//There is a South Pole magnetic field
                       The magnetic field at the South Pole!");
   Serial.println("
 }
 else {//If not
   Serial.println("
                       Just be all normal!");
 }
}
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, when the sensor detects no magnetic fields or the north pole magnetic field, the monitor l will show"1 Just be all normal!"and the LED on the sensor will be off; When it detects the south pole magnetic field, "0 The magnetic field at the South Pole!"and the LED on the sensor will be on.

	/dev/ttyUSB1			×
			Sen	d
1	Just be all normal!			•
1	Just be all normal!			
1	Just be all normal!			
1	Just be all normal!			
1	Just be all normal!			
1	Just be all normal!			
0	The magnetic field at the South Pole!			
0	The magnetic field at the South Pole!			
Θ	The magnetic field at the South Pole!			
0	The magnetic field at the South Pole!			
0	The magnetic field at the South Pole!			
Θ	The magnetic field at the South Pole!			
0	The magnetic field at the South Pole!			
0	The magnetic field at the South Pole!			
0	The magnetic field at the South Po			Ŧ
			•	
•	Autoscroll 🗌 Show timestamp Newline 🔹 9600 baud 💌	Clear o	outpu	Jt

# 7.5.15 Project 15: Reed Switch Module



### Overview

In this kit, there is a Keyestudio reed switch module, which mainly uses a MKA10110 green reed component.

The reed switch is the abbreviation of the dry reed switch. It is a passive electronic switch element with contacts.

It has the advantages of simple structure, small size and easy control.

Its shell is a sealed glass tube with two iron elastic reed electric plates.

In the experiment, we will determine whether there is a magnetic field near the module by reading the high and low level of the S terminal on the module; and, we display the test result in the shell.

### **Working Principle**

In normal conditions, the glass tube in the two reeds made of special materials are separated. When a magnetic substance close to the glass tube, in the role of the magnetic field lines, the pipe within the two reeds are magnetized to attract each other in contact, the reed will suck together, so that the junction point of the connected circuit communication.

After the disappearance of the outer magnetic reed because of their flexibility and separate, the line is disconnected. The sensor uses this characteristic to build a circuit to convert magnetic field signal into high and low level signal.

#### Components



**Connection Diagram** 



```
Test Code
```

```
/*
* Filename : Reed Switch
* Description : Read the value of the reed sensor
* Auther
            : http://www.keyestudio.com
*/
int val = 0;
int reedPin = 15; //Define dry reed module signal pin connected to GPI015
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(reedPin, INPUT);//Set mode to input
}
void loop() {
 val = digitalRead(reedPin);//Read digital level
 Serial.print(val);//Serial port shows up
 if (val == 0) {//There's a magnetic field nearby
   Serial.print("
                      ");
   Serial.println("A magnetic field");
   delay(100);
 }
 else {//There is no magnetic field
   Serial.print("
                      ");
   Serial.println("There is no magnetic field");
```

(continues on next page)

	(continued from previous page
	delay(100);
	}
}	
1	/ * * * * * * * * * * * * * * * * * * *

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onopen the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters.

When the sensor detects a magnetic field, val is 0 and the red LED of the module lights up, "0 A magnetic field" will be displayed; when no magnetic field is detected, val is 1, and the LED on the module goes out, "1 There is no magnetic field" will be shown, as shown below.

	/dev/ttyUSB1	~	^	×
			Ser	id
1 1 1 1 1 1 1 0	There is no magnetic field There is no magnetic field A magnetic field			•
	A magnetic field A			
🖌 Autoso	croll 🗌 Show timestamp Newline 🔹 9600 baud 💌 Clea	ar o	outp	ut

# 7.5.16 Project 16: PIR Motion Sensor



## Overview

In this kit, there is a Keyestudio PIR motion sensor, which mainly uses an RE200B-P sensor elements. It is a human

body pyroelectric motion sensor based on pyroelectric effect, which can detect infrared rays emitted by humans or animals, and the Fresnel lens can make the sensor's detection range farther and wider.

In the experiment, we determine if there is someone moving nearby by reading the high and low levels of the S terminal on the module. The detected results will be displayed on the Shell.

#### Working Principle

The upper left part is voltage conversion(VCC to 3.3V). The working voltage of sensors we use is 3.3V, therefore we can't use 5V directly. The voltage conversion circuit is needed.

When no person is detected or no infrared signal is received, and pin 1 of the sensor outputs low level. At this time, the LED on the module will light up and the MOS tube Q1 will be connected and the signal terminal S will detect Low levels.

When one is detected or an infrared signal is received, and pin 1 of the sensor outputs a high level. Then LED on the module will go off, the MOS tube Q1 is disconnected and the signal terminal S will detect high levels.



**Required Components** 



## **Connection Diagram**



**Test Code** 

```
/*
* Filename : PIR motion
* Description : Reading the value of the human body infrared sensor
* Auther
          : http://www.keyestudio.com
*/
int val = 0;
int pirPin = 15; //The pin of PIR motion sensor is defined as GPI015
void setup() {
 Serial.begin(9600); //Set baud rate to 9600
 pinMode(pirPin, INPUT); //Set the sensor to input mode
}
void loop() {
 val = digitalRead(pirPin); //Read the sensor value
 Serial.print(val);//Print val value
 if (val == 1) {//There is movement nearby, output high level
                      ");
   Serial.print("
   Serial.println("Some body is in this area!");
   delay(100);
 }
 else {//If no movement nearby, output low level
   Serial.print("
                      ");
   Serial.println("No one!");
```

(continues on next page)

	(continued from previous page)
	delay(100);
	}
}	
/	·/************************************

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the corresponding data and characters.

When the sensor detects someone nearby, value is 1, the LED will go off and the monitor will show "1 Somebody is in this area!". In contrast, the value is 0, the LED will go up and "0 No one!" will be shown.

	/dev/ttyUSB1	~ <i>/</i>	××
		S	Send
0	No one!		•
0	No one!		
0	No one!		
0	No one!		
0	No one! No one!		
1	Some body is in this area!		
1	Some body is in this area! Some body is in this area!		
1	Some body is in this area!		
1	Some body is in this area!		
1			
•	Autoscroll 🗌 Show timestamp Newline 🔹 9600 baud 💌 Clea	ar ou	ıtput

# 7.5.17 Project 17: Active Buzzer



Overview

In this kit, it contains an active buzzer module and a power amplifier module (the principle is equivalent to a passive buzzer).

In this experiment, we control the active buzzer to emit sounds. Since it has its own oscillating circuit, the buzzer will automatically sound if given large voltage.

### **Working Principle**

From the schematic diagram, the pin of buzzer is connected to a resistor R2 and another port is linked with a NPN triode Q1. So, if this triode Q1 is powered, the buzzer will sound.

If the base electrode of the triode connected to the R1 resistor is a high level, the triode Q1 will be connected. If the base electrode is pulled down by the resistor R3, the triode is disconnected.

When we output a high level from the IO port to the triode, the buzzer will emit sounds; if outputting low levels, the buzzer won't emit sounds.



#### Components



**Connection Diagram** 



**Test Code** 

```
/*
* Filename
           : Active buzzer
* Description : An active buzzer produces sound
          : http://www.keyestudio.com
* Auther
*/
int buzzer = 15; //Define buzzer receiver pin GPI015
void setup() {
 pinMode(buzzer, OUTPUT);//Set the output mode
}
void loop() {
 digitalWrite(buzzer, HIGH); //sound production
 delay(1000);
 digitalWrite(buzzer, LOW); //Stop the sound
 delay(1000);
}
```

## **Code Explanation**

In the experiment, we set the pin to GPIO15. When setting to high, the active buzzer will beep; when setting to low, the active buzzer will stop emitting sounds.

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. The active buzzer will emit sound for 1 second, and stop for 1 second.



# 7.5.18 Project 18: 8002b Audio Power Amplifier

#### Overview

In this kit, there is a Keyestudio 8002b audio power amplifier. The main components of this module are an adjustable potentiometer, a speaker, and an audio amplifier chip;

The main function of this module is: it can amplify the output audio signal, with a magnification of 8.5 times, and play sound or music through the built-in low-power speaker, as an external amplifying device for some music playing equipment.

In the experiment, we used the 8002b power amplifier speaker module to emit sounds of various frequencies.

### **Working Principle**

In fact, it is similar to a passive buzzer. The active buzzer has its own oscillation source. Yet, the passive buzzer does not have internal oscillation. When controlling the circuit, we need to input square waves of different frequencies to the positive pole of the component and ground the negative pole to control the buzzer to chime sounds of different frequencies.



## Components



**Connection Diagram** 



Test Code

```
/*
* Filename
            : Passive Buzzer
* Description : Passive Buzzer sounds the alarm.
          : http//www.keyestudio.com
* Auther
*/
#define LEDC_CHANNEL_0 0
// LEDC timer uses 13 bit accuracy
#define LEDC_TIMER_13_BIT 13
// Define tool I/O ports
#define BUZZER_PIN 15
//Create a musical melody list, Super Mario
int melody[] = {330, 330, 330, 262, 330, 392, 196, 262, 196, 165, 220, 247, 233, 220,
→196, 330, 392, 440, 349, 392, 330, 262, 294, 247, 262, 196, 165, 220, 247, 233, 220, .
→196, 330, 392,440, 349, 392, 330, 262, 294, 247, 392, 370, 330, 311, 330, 208, 220, J
→262, 220, 262, 294, 392, 370, 330, 311, 330, 523, 523, 523, 392, 370, 330, 311, 330, .
→208, 220, 262,220, 262, 294, 311, 294, 262, 262, 262, 262, 262, 262, 294, 330, 262, 220, .
→196, 262, 262, 262, 262, 294, 330, 262, 262, 262, 262, 294, 330, 262, 220, 196};
//Create a list of tone durations
→4,8,4,8,4,8,2,8,4,4,8,4,1,8,4,4,8,4,8,4,8,2};
void setup() {
pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer to output mode
}
void loop() {
 int noteDuration; //Create a variable of noteDuration
 for (int i = 0; i < sizeof(noteDurations); ++i)</pre>
 {
    noteDuration = 800/noteDurations[i];
    ledcSetup(LEDC_CHANNEL_0, melody[i]*2, LEDC_TIMER_13_BIT);
    ledcAttachPin(BUZZER_PIN, LEDC_CHANNEL_0);
    ledcWrite(LEDC_CHANNEL_0, 50);
    delay(noteDuration * 1.30); //delay
 }
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power onthen the power amplifier module will emit the sound on a

loop.

## 7.5.19 Project 19: 130 Motor



### Description

The 130 motor driver module is compatible with servo motors, which has high efficiency and good quality fans.

It adopts a HR1124S motor control chip. HR1124S is a single-channel H-bridge driver chip for DC motor solutions. In addition, this chip has low standby current and low quiescent current.

The module is compatible with various single-chip control boards. In the experiment, we can control the rotation direction of the motor by outputting the voltage directions of the two signal terminals IN+ and IN- to make the motor rotate.

### **Working Principle**

The chip is used to help drive the motor. We can't drive it with a triode or an IO port due to its a large current of need. It is very simple to make the motor rotate. Just apply voltage to both ends of the motor. The direction of the motor is different in different voltage directions. Within the rated voltage, the higher the voltage, the faster the motor rotates; on the contrary, the lower the voltage, the slower the motor rotates, or even unable to rotate.

So we can use the PWM port to control the speed of the motor. We haven't learned PWM here, so we use the high and low levels to control the motor first.

### **Required Components**



Note: the motor is separated with its fan, you need to assemble it first.

## **Connection Diagram**

130 Motor	ESP32 Expansion Board
G	G
V	5V
IN+	IO15
IN-	IO4



**Test Code** 

(continues on next page)

(continued from previous page)

```
* Auther
             : http://www.keyestudio.com
*/
//Define two pins interfaces of the motor, respectively 15 and 4
int INA = 15; //INA corresponds to IN+
int INB = 4; //INB corresponds to IN-
void setup() {
 //Set the motor pins as output
 pinMode(INA, OUTPUT);
 pinMode(INB, OUTPUT);
}
void loop() {
 //Turn counterclockwise
 digitalWrite(INA, HIGH);
 digitalWrite(INB, LOW);
 delay(2000);
 //stop
 digitalWrite(INA, LOW);
 digitalWrite(INB, LOW);
 delay(1000);
 //clockwise rotation
 digitalWrite(INA, LOW);
 digitalWrite(INB, HIGH);
 delay(2000);
 //stop
 digitalWrite(INA, LOW);
 digitalWrite(INB, LOW);
 delay(1000);
}
```

### **Code Explanation**

Set pins to GPIO4GPIO15, when the pin GPIO4 outputs low levels and the pin GPIO15 outputs high levels, the motor will rotate counterclockwise; when both pins are set to low, the motor stops rotating.

### **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch ON the ESP32 expansion board to the ON end, after powering on, compile and upload the code to the ESP32. After uploading successfully the fan will rotate counterclockwise for 2 seconds, stop for 1 second and clockwise for 2 seconds and stop for 1 second; cycle alternately.

# 7.5.20 Project 20: Potentiometer



Overview

The following we will introduce is the Keyestudio rotary potentiometer which is an analog sensor.

The digital IO ports can read the voltage value between 0 and 3.3V and the module only outputs high levels. However, the analog sensor can read the voltage value through 16 ADC analog ports on the ESP32 board. In the experiment, we will display the test results on the Shell.

### **Working Principle**



It uses a 10K adjustable resistor. We can change the resistance by rotating the potentiometer. The signal S can detect the voltage changes(0-3.3V) which are analog quantity.

**ADC** The more bits an ADC has, the denser the partitioning of the simulation, the higher the accuracy of the final conversion.



Section 1: 0V – 3.3/4095 V analog quantity corresponding to digital 0;

Section 2: Analog quantities in the range 3.3/4095V - 2\*3.3/4095V correspond to digital 1;

•••

The conversion formula is as follows:

$$ADCValue = \frac{Ana \log Voltage}{3.3} *4095$$

DAC The higher the precision of DAC, the higher the precision of the output voltage value.

The conversion formula is as follows:

Ana log Voltage = 
$$\frac{DACValue}{255}$$
 \* 3.3(V)

## ADC on ESP32

The ESP32 has 16 pins that can be used to measure analog signals. GPIO pin serial numbers and analog pin definitions are shown below:

ADC number in ESP32	ESP32 GPIO number
ADC0	GPIO 36
ADC3	GPIO 39
ADC4	GPIO 32
ADC5	GPIO33
ADC6	GPIO34
ADC7	GPIO 35
ADC10	GPIO 4
ADC11	GPIO0
ADC12	GPIO2
ADC13	GPIO15
ADC14	GPIO13
ADC15	GPIO 12
ADC16	GPIO 14
ADC17	GPIO27
ADC18	GPIO25
ADC19	GPIO26

## DAC on ESP32

The ESP32 has two 8-bit digital-to-analog converters connected to GPIO25 and GPIO26 pins, which are immutable, as shown below :

Simulate pin number	GPIO number
DAC1	GPIO25
DAC2	GPIO26

#### Components



## **Connection Diagram**



## **Test Code**

(continued from previous page)

```
#define PIN_ANALOG_IN 34 //the pin of the Potentiometer
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC_
\leftrightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

## **Code Explanation**

1). analogVal means analog value. The rotary potentiometer outputs analog values  $(0 \sim 4095)$ , therefore, we set pins to analog ports. For example, we connect to GPIO34.

2). analogRead(pin): read the value of the specified analog pin. The ESP32 contains a multi-channel, 12-bit converter. This means that it will map the input voltage between 0 and the working voltage (5V or 3.3V) to an integer value between 0 and 4095. For example, this will produce a resolution among readings: 3.3V/4096 stands for 0.0008V per unit.

3). The map() function converts this 12-bit DAC value to an 8-bit DAC value.

4). Pin: the name of analog input pin.

5). The serial monitor displays the values of adcVal, dacVal, voltage, the baud rate must be set before display (we default to 9600, which can be changed).

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the potentiometer's ADC value, DAC value and voltage value. Rotate the potentiometer handle, the analog values will change.

						,	/dev/ttyl	JSB0				~	^	×
													Sen	id
ADC	Val:	282,	DAC	Val:	18,	Voltage:	0.23V							•
ADC	Val:	387,	DAC	Val:	24,	Voltage:	0.31V							
ADC	Val:	545,	DAC	Val:	34,	Voltage:	0.44V							
ADC	Val:	723,	DAC	Val:	45,	Voltage:	0.58V							
ADC	Val:	928,	DAC	Val:	58,	Voltage:	0.75V							
ADC	Val:	1190,	DAC	Val:	74,	Voltage:	0.96V							
ADC	Val:	1409,	DAC	Val:	88,	Voltage:	1.14V							
ADC	Val:	1679,	DAC	Val:	105,	Voltage:	1.35V							
ADC	Val:	2007,	DAC	Val:	125,	Voltage:	1.62V							
ADC	Val:	2257,	DAC	Val:	141,	Voltage:	1.82V							
ADC	Val:	2530,	DAC	Val:	158,	Voltage:	2.04V							
ADC	Val:	2592,	DAC	Val:	161,	Voltage:	2.09V							
ADC	Val:	2882,	DAC	Val:	179,	Voltage:	2.32V							1
ADC	Val:	3366,	DAC	Val:	210,	Voltage:	2.71V							
						0								÷
•													)	•
. ⊻	Autoso	croll 🗌	Show ti	mestai	mp			Newline	•	9600 baud	•	Clear	outp	ut

# 7.5.21 Project 21: Steam Sensor



## Description

This is a DIY electronic building block water drop sensor. It is an analog (digital) input module, also called rain, rain sensor. It can be used to monitor various weather conditions, detect whether it is raining and the amount of rain, convert it into digital signal (DO) and analog signal (AO) output, and is widely used in Arduino robot kits, raindrops, rain sensors, and can be used for various It can monitor various weather conditions, and convert it into digital signal and AO output, and can also be used for automobile automatic wiper system, intelligent lighting system and intelligent sunroof system.

In the experiment, we input the sensor signal terminal (S terminal) to the analog port of the ESP32 development board, sense the change of the analog value, and display the corresponding analog value on the shell.

## **Working Principle**

Its principle is to detect the amount of water through the exposed printed parallel lines on the circuit board. The more water there is, the more wires will be connected, and the conductive contact area increases. The voltage output by pin 2 will gradually increase. The larger the analog value detected by the signal terminal S is.

It can also detect steam in the air. Two position holes are used to install on the other devices.



**Required Components** 



**Connection Diagram** 



## **Test Code**

```
/*
* Filename
            : Steam sensor
* Description : Read the basic usage of ADCDAC and Voltage
            : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 34 //the pin of the Steam sensor
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
→ function is used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the
→information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We

need to press the reset button on the ESP32, then the serial monitor will display the steam sensor's ADC value, DAC value and voltage value. When a few drops of water are placed in the sensor sensing area, the values will change. The more water volume, the greater the output voltage value, ADC value and the DAC value.

						/dev/ttyl	USB0				~	^	×
												Ser	id
ADC ADC ADC ADC ADC ADC ADC	Val: Val: Val: Val: Val: Val:	282, 387, 545, 723, 928, 1190,	DAC DAC DAC DAC DAC DAC	Val: Val: Val: Val: Val: Val:	18, 24, 34, 45, 58, 74,	Voltage: 0.23V Voltage: 0.31V Voltage: 0.44V Voltage: 0.58V Voltage: 0.75V Voltage: 0.96V							•
ADC ADC ADC ADC ADC ADC ADC ADC ADC	Val: Val: Val: Val: Val: Val: Val: Val:	1409, 1679, 2007, 2257, 2530, 2592, 2882, 3366,	DAC DAC DAC DAC DAC DAC DAC DAC DAC	Val: Val: Val: Val: Val: Val: Val: Val:	88, 105, 125, 141, 158, 161, 179, 210,	Voltage: 1.14V Voltage: 1.35V Voltage: 1.62V Voltage: 1.82V Voltage: 2.04V Voltage: 2.09V Voltage: 2.32V Voltage: 2.71V							ļ
•	Autoso	croll 🗌 :	Show tir	mestai	mp		Newline	•	9600 baud	•	Clear	outp	ut

## 7.5.22 Project 22: Sound Sensor



## Overview

In this kit, there is a Keyestudio DIY electronic block and a sound sensor. In the experiment, we test the analog value corresponding to the sound level in the current environment with it. The louder the sound, the larger the ADC, DAC and the voltage value, and the "shell" window will display the test results.

### **Working Principle**

It uses a high-sensitive microphone component and an LM386 chip. We build the circuit with the LM386 chip and



amplify the sound through the high-sensitive microphone. In addition, we can adjust the sound volume by the potentiometer. Rotate it clockwise, the sound will get louder.

Components



**Connection Diagram** 



### **Test Code**

```
//******
                      /*
* Filename : MicroPhone
* Description : Read the basic usage of ADCDAC and Voltage
           : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 34 //the pin of the Sound Sensor
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
→value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
          ****
```

**Test Result** 

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the sound sensor's ADC value, DAC value and voltage value. Rotate clockwise the potentiometer and speak at the MIC. Then you can see the analog value get larger, as shown below:

						/dev/ttyUSB0	~ ^	×	:
							S	end	ł
ADC ADC	Val: Val:	0, 1440,	DAC DAC	Val: Val:	0, 90,	Voltage: 0.00V Voltage: 1.16V			•
ADC ADC	Val: Val:	0, 688,	DAC DAC	Val: Val:	0, 43,	Voltage: 0.00V Voltage: 0.55V			
ADC ADC	Val: Val:	0, 950, 1107	DAC	Val: Val:	0, 59,	Voltage: 0.00V Voltage: 0.77V			
ADC ADC ADC	Val: Val: Val:	0, 233.	DAC DAC DAC	Val: Val: Val:	09, 0, 15,	Voltage: 0.89V Voltage: 0.00V Voltage: 0.19V			
ADC ADC	Val: Val:	0, 0,	DAC DAC	Val: Val:	0, 0,	Voltage: 0.00V Voltage: 0.00V			i
ADC ADC	Val: Val:	909, 1130, 1264	DAC DAC	Val: Val:	57, 70, 70	Voltage: 0.73V Voltage: 0.91V Voltage: 1.02V			I
ADC	vat.	1204,	DAC	val.	19,	Vollage. 1.02V		•	Ŧ
✓ Autoscroll						Newline   9600 baud  Cle	ar ou	Itpu	t





## Description

In this kit, there is a photoresistor which consists of photosensitive resistance elements. Its resistance changes with the light intensity. Also, it converts the resistance change into a voltage change through the characteristic of the photosen-

sitive resistive element. When wiring it up, we interface its signal terminal (S terminal) with the analog port of ESP32, so as to sense the change of the analog value, and display the corresponding analog value in the shell.

### **Working Principle**

If there is no light, the resistance is 0.2M and the detected voltage at the terminal 2 is close to 0. When the light intensity increases, the resistance value of the light sensor is getting smaller and smaller, so the voltage detected at the signal end is getting larger and larger...



Components



**Connection Diagram** 



## **Test Code**

```
/*
* Filename
            : Photoresistance
* Description : Read the basic usage of ADCDAC and Voltage
* Auther
           : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 34 //the pin of the Photoresistance
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the
→information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After

	/dev/ttyUSB0		~ .	~ ×
•			5	Send
ADC Val: 282,         DAC Val: 18,           ADC Val: 387,         DAC Val: 24,           ADC Val: 545,         DAC Val: 34,           ADC Val: 723,         DAC Val: 45,           ADC Val: 928,         DAC Val: 58,           ADC Val: 1190,         DAC Val: 74,           ADC Val: 1409,         DAC Val: 74,           ADC Val: 1679,         DAC Val: 105,           ADC Val: 2007,         DAC Val: 125,	Voltage: 0.23V Voltage: 0.31V Voltage: 0.44V Voltage: 0.58V Voltage: 0.75V Voltage: 0.96V Voltage: 1.14V Voltage: 1.35V Voltage: 1.62V			
ADC Val: 2257, DAC Val: 141, ADC Val: 2530, DAC Val: 158, ADC Val: 2592, DAC Val: 161, ADC Val: 2882, DAC Val: 179, ADC Val: 3366, DAC Val: 210,	Voltage: 2.04V Voltage: 2.09V Voltage: 2.32V Voltage: 2.71V			•
Autoscroll Show timestamp	Newline 🔻 9600 baud	▼ CI	lear ou	utput

uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the photoresistor's ADC value, DAC value and voltage value. When the light intensity gets stronger, the analog values will get larger, as shown below:



# 7.5.24 Project 24: NTC-MF52AT Thermistor

## Overview

In the experiment, there is a NTC-MF52AT analog thermistor. We connect its signal terminal to the analog port of the ESP32 mainboard and read the corresponding ADC value, voltage value and thermistor value.

We can use analog values to calculate the temperature of the current environment through specific formulas. Since the temperature calculation formula is more complicated, we only read the corresponding analog value.

## **Working Principle**



This module mainly uses NTC-MF52AT thermistor element, which can sense the changes of the surrounding environment temperature. Resistance changes with the temperature, causing the voltage of the signal terminal S to change.

This sensor uses the characteristics of NTC-MF52AT thermistor element to convert resistance changes into voltage changes.

## Components



**Connection Diagram**


#### **Test Code**

```
/*
* Filename : Temperature sensor
* Description : Making a thermometer by thermistor.
* Auther
          : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN
                   34
void setup() {
 Serial.begin(9600);
}
void loop() {
 int adcValue = analogRead(PIN_ANALOG_IN);
                                                    //read ADC pin
 double voltage = (float)adcValue / 4095.0 * 3.3;
                                                   // calculate voltage
 double Rt = (3.3 - voltage) / voltage * 4.7;
                                                     //calculate
→resistance value of thermistor
 double tempK = 1 / (1 / (273.15 + 25) + log(Rt / 10) / 3950.0); //calculate_
→temperature (Kelvin)
double tempC = tempK - 273.15;
                                                    //calculate
→temperature (Celsius)
 Serial.printf("ADC value : %d,\tVoltage : %.2fV, \tTemperature : %.2fC\n", adcValue,_
→voltage, tempC);
 delay(1000);
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After

ADC value : 1353,       Voltage : 1.09V,       Temperature : 23.91C         ADC value : 1360,       Voltage : 1.10V,       Temperature : 23.74C         ADC value : 1363,       Voltage : 1.10V,       Temperature : 23.66C         ADC value : 1358,       Voltage : 1.09V,       Temperature : 23.79C         ADC value : 1387,       Voltage : 1.12V,       Temperature : 23.79C         ADC value : 1387,       Voltage : 1.12V,       Temperature : 23.08C         ADC value : 1444,       Voltage : 1.16V,       Temperature : 23.08C         ADC value : 1445,       Voltage : 1.20V,       Temperature : 20.76C         ADC value : 1493,       Voltage : 1.20V,       Temperature : 20.76C         ADC value : 1495,       Voltage : 1.20V,       Temperature : 20.53C         ADC value : 1495,       Voltage : 1.20V,       Temperature : 20.53C         ADC value : 1502,       Voltage : 1.21V,       Temperature : 20.37C         ADC value : 1505,       Voltage : 1.21V,       Temperature : 20.30C         ADC value : 1511,       Voltage : 1.22V,       Temperature : 20.16C         ADC value : 1511,       Voltage : 1.22V,       Temperature : 20.16C         ADC value : 1514,       Voltage : 1.22V,       Temperature : 20.09C	/dev/ttyUSB0							
ADC value : 1353,Voltage : 1.09V,Temperature : 23.91CADC value : 1360,Voltage : 1.10V,Temperature : 23.74CADC value : 1363,Voltage : 1.10V,Temperature : 23.66CADC value : 1358,Voltage : 1.09V,Temperature : 23.79CADC value : 1387,Voltage : 1.12V,Temperature : 23.08CADC value : 1444,Voltage : 1.16V,Temperature : 21.72CADC value : 1485,Voltage : 1.20V,Temperature : 20.76CADC value : 1493,Voltage : 1.20V,Temperature : 20.58CADC value : 1495,Voltage : 1.20V,Temperature : 20.53CADC value : 1502,Voltage : 1.21V,Temperature : 20.37CADC value : 1505,Voltage : 1.21V,Temperature : 20.30CADC value : 1511,Voltage : 1.22V,Temperature : 20.16CADC value : 1514,Voltage : 1.22V,Temperature : 20.16C					Send			
ADC value : 1511, Voltage : 1.22V, Temperature : 20.16C ADC value : 1511, Voltage : 1.22V, Temperature : 20.16C ADC value : 1514, Voltage : 1.22V, Temperature : 20.09C	C value : 1353,       Value : 1360,       Value : 1360,       Value : 1363,       Value : 1363,       Value : 1363,       Value : 1358,       Value : 1358,       Value : 1358,       Value : 1387,       Value : 1387,       Value : 1387,       Value : 1444,       Value : 1444,       Value : 1444,       Value : 1445,       Value : 1493,       Value : 1493,       Value : 1493,       Value : 1495,       Value : 1502,       Value : 1502,       Value : 1505,       Value : 1505	tage : 1.09V, tage : 1.10V, tage : 1.10V, tage : 1.09V, tage : 1.09V, tage : 1.12V, tage : 1.16V, tage : 1.20V, tage : 1.20V, tage : 1.20V, tage : 1.21V, tage : 1.21V,	Temperature : Temperature :	23.91C 23.74C 23.66C 23.79C 23.08C 21.72C 20.76C 20.58C 20.53C 20.37C 20.30C				
Autoscroll Show timestamp	C value : 1511, Vo C value : 1511, Vo C value : 1514, Vo	tage : 1.22V, tage : 1.22V, tage : 1.22V,	Temperature : Temperature : Temperature :	20.16C 20.16C 20.09C	•			

uploading successfully, we will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the thermistor's ADC value, DAC value and voltage value, as shown below:

## 7.5.25 Project 25: Thin-film Pressure Sensor



#### Overview

In this kit, there is a Keyestudio thin-film pressure sensor. The thin-film pressure sensor composed of a new type of nano pressure-sensitive material and a comfortable ultra-thin film substrate, has waterproof and pressure-sensitive functions.

In the experiment, we determine the pressure by collecting the analog signal on the S end of the module. The smaller the ADC value, DAC value and voltage value, the greater the pressure; and the displayed results will shown on the Shell.

#### **Working Principle**



When the sensor is pressed by external forces, the resistance value of sensor will vary. We convert the pressure signals detected by the sensor into the electric signals through a circuit. Then we can obtain the pressure changes by detecting voltage signal changes.

#### Components



**Connection Diagram** 



#### **Test Code**

```
/*
* Filename
             : Film pressure sensor
* Description : Read the basic usage of ADCDAC and Voltage
* Auther
          : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 34 //the pin of the Film pressure sensor
void setup() {
 Serial.begin(9600);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map() \Box
\rightarrow function is used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the
→information is finally printed out.
```

(continued from previous page)

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then the serial monitor will display the thin-film's ADC value, DAC value and voltage value, when the thin-film is pressed by fingers, the analog value will decrease, as shown below;

						/0	dev/ttyL	JSB0				~	^	×
													Ser	nd
ADC	Val:	282,	DAC	Val:	18,	Voltage: (	0.23V							•
ADC	Val:	387,	DAC	Val:	24,	Voltage: (	0.31V							
ADC	Val:	545,	DAC	Val:	34,	Voltage: 0	0.44V							
ADC	Val:	723,	DAC	Val:	45,	Voltage: (	0.58V							
ADC	Val:	928,	DAC	Val:	58,	Voltage: (	0.75V							
ADC	Val:	1190,	DAC	Val:	74,	Voltage: (	0.96V							
ADC	Val:	1409,	DAC	Val:	88,	Voltage: :	1.14V							
ADC	Val:	1679,	DAC	Val:	105,	Voltage: :	1.35V							
ADC	Val:	2007,	DAC	Val:	125,	Voltage: :	1.62V							
ADC	Val:	2257,	DAC	Val:	141,	Voltage: :	1.82V							
ADC	Val:	2530,	DAC	Val:	158,	Voltage: 2	2.04V							
ADC	Val:	2592,	DAC	Val:	161,	Voltage:	2.09V							
ADC	Val:	2882,	DAC	Val:	179,	Voltage: 2	2.32V							10
ADC	Val:	3366,	DAC	Val:	210,	Voltage: 2	2.71V							
						-								-
•														•
	Autoso	croll 🗌 S	Show tir	mestai	mp			Newline	•	9600 baud	•	Clear	outp	ut

### 7.5.26 Project 26: Flame Sensor



#### Description

In daily life, it is often seen that a fire broke out without any precaution. It will cause great economic and human loss. So how can we avoid this situation? Right, install a flame sensor and a speaker in those places that easily break out a fire. When the flame sensor detects a fire, the speaker will alarm people quickly to put out the fire.

So in this project, you will learn how to use a flame sensor and an active buzzer module to simulate the fire alarm system.

#### **Working Principle**

This flame sensor can be used to detect fire or other light sources with wavelength stands at 700nm  $\sim$  1000nm. Its detection angle is about 60°. You can rotate the potentiometer on the sensor to control its sensitivity. Adjust the potentiometer to make the LED at the critical point between on and off state. The sensitivity is the best.

From the below figure, power up. When detecting fire, the digital pin outputs low levels, the red LED2 will light up first, the digital signal terminal D0 outputs a low level, and the red LED1 will light up. The stronger the external infrared light, the smaller the value; the weaker the infrared light, the larger the value.



**Required Components** 



#### **Connection Diagram**



#### Test Code



```
(continued from previous page)
```

```
*/
//Flame sensor two pins 13, 34, respectively
#define PIN_ANALOG_IN 34
int digitalPin = 13;
//The following two variables hold the digital signal and adc values respectively
int analogVal = 0;
int adcVal = 0;
void setup() {
 Serial.begin(9600);
 pinMode(digitalPin, INPUT); //Digital pin 13 is set to input mode
}
//In loop()the digitalRead()function is used to obtain the digital value,
//the analogRead() function is used to obtain the ADC value.
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int digitalVal = digitalRead(digitalPin); //Read digital signal;
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("digitalVal: %d, \t ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n",
→digitalVal, adcVal, dacVal, voltage);
 delay(200);
}
```

#### **Code Explanation**

Two pins we use are defined as GPIO13 and GPIO34 according to the wiring-up diagram, and print digital signals and analog signals respectively.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the sensor, we can adjust the red LED bright and not bright critical point. The red LED2 on the sensor module is lit, while the red LED1 is not. Open the monitor and set baud rate to 9600.

We need to press the reset button on the ESP32, then the "Shell" window will display the digital value, ADC value, DAC value and voltage value of the flame sensor. When fire is detected, the LED1 will be on. the digital value will change from 1 to 0, and the analog value will become smaller, as shown below.

	-			/dev/ttyUSB0	~ ^ X
					Send
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	<b>^</b>
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	1, ADC	Val:	4095,	DAC Val: 255, Voltage: 3.30V	
digitalVal:	0, ADC	Val:	142,	DAC Val: 9, Voltage: 0.11V	
digitalVal:	0, ADC	Val:	2129,	DAC Val: 133, Voltage: 1.72V	
digitalVal:	0, ADC	Val:	1823,	DAC Val: 114, Voltage: 1.47V	
digitalVal:	0, ADC	Val:	2086,	DAC Val: 130, Voltage: 1.68V	
digitalVal:	0, ADC	Val:	1397,	DAC Val: 87, Voltage: 1.13V	
digitalVal:	0, ADC	Val:	2001,	DAC Val: 125, Voltage: 1.	-
<					•
Autoscroll [	Show t	mesta	mp	Newline 🔻 9600 baud	<ul> <li>Clear output</li> </ul>

# 7.5.27 Project 27: MQ-2 Gas Sensor

#### Description

This analog gas sensor - MQ2 is used in gas leakage detecting equipment in consumer electronics and industrial markets.

This sensor is suitable for detecting LPG, I-butane, propane, methane, alcohol, Hydrogen and smoke. It has high sensitivity and quick response.

In addition, the sensitivity can be adjusted by rotating the potentiometer.

In the experiment, we read the analog value at the A0 port and the D0 port to determine the content of gas.

#### **Working Principle**

The greater the concentration of smoke, the greater the conductivity, the lower the output resistance, the greater the output analog signal.

When in use, the A0 terminal reads the analog value of the corresponding gas; the D0 terminal is connected to an LM393 chip (voltage comparator), we can adjust the alarm threshold of the measured gas through the potentiometer, and output the digital value at D0. When the measured gas content exceeds the critical point, the D0 terminal outputs a low level; when the measured gas content does not exceed the critical point, the D0 terminal outputs a high level.



**Required Components** 



**Connection Diagram** 



#### Test Code

```
/*
* Filename : MQ2
* Description : Read the basic usage of Digital, ADCDAC and Voltage
* Auther : http//www.keyestudio.com
*/
//MQ_2 two pins 13, 34, respectively
#define PIN_ANALOG_IN 34
int digitalPin = 13;
//The following two variables hold the digital signal and adc values respectively
int analogVal = 0;
int adcVal = 0;
void setup() {
 Serial.begin(9600);
 pinMode(digitalPin, INPUT); //Digital pin 13 is set to input mode
}
//In loop()the digitalRead()function is used to obtain the digital value, the.
\rightarrow analogRead() function is used to obtain the ADC value. and then the map() function is
→used to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the
→information is finally printed out.
```

(continued from previous page)

```
void loop() {
 int digitalVal = digitalRead(digitalPin); //Read digital signal;
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("digitalVal: %d, \t ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n",
→digitalVal, adcVal, dacVal, voltage);
 if (digitalVal == 1) {
   Serial.println(" Normal");
 }
 else {
   Serial.println(" Exceeding");
 }
 delay(100); //Delay time 100 ms
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the sensor, we can adjust the red LED bright and not bright critical point. Open the monitor, set baud rate to 9600.

We need to press the reset button on the ESP32, then the monitor displays the corresponding data and characters. When the sensor detects the smoke or combustible gas, the red LED lights up and the digital value changes from 1 to 0, the ADC value, DAC value and voltage value increase, as shown below.

					/dev	/ttyUS	B0				•	^	×
												Sen	nd
digitalVal: Normal	1,	ADC	Val:	1783,	DAC	Val:	111,	Voltage:	1.44V				•
digitalVal: Normal	1,	ADC	Val:	1801,	DAC	Val:	112,	Voltage:	1.45V				
digitalVal: Normal	1,	ADC	Val:	1856,	DAC	Val:	116,	Voltage:	1.50V				
digitalVal: Normal	1,	ADC	Val:	1963,	DAC	Val:	122,	Voltage:	1.58V				
digitalVal: Exceeding	Θ,	ADC	Val:	2093,	DAC	Val:	130,	Voltage:	1.69V				
digitalVal: Exceeding	Θ,	ADC	Val:	2199,	DAC	Val:	137,	Voltage:	1.77V				
digitalVal: Exceeding	Θ,	ADC	Val:	2289,	DAC	Val:	143,	Voltage:	1.84V				I
digital												)	Ŧ
Autoscroll	Sh	ow tin	nestar	np		1	Vewline	•	9600 baud	▼ Cle	ear c	outp	ut

# 7.5.28 Project 28: MQ-3 Alcohol Sensor



#### Description

In this kit, there is a MQ-3 alcohol sensor, which uses the gas-sensing material is tin dioxide (SnO2) which has a low conductivity in clean air. When there is alcohol vapor in the environment where the sensor is located, the conductivity of the sensor increases with the increase of the alcohol gas concentration in the air. The change in conductivity can be converted into an output signal corresponding to the gas concentration using a simple circuit.

In the experiment, we read the analog value at the A0 end of the sensor and the digital value at the D0 end to judge the content of alcohol vapor in the air and whether they exceed the standard.

#### **Working Principle**

At a certain temperature, the conductivity changes with the composition of the ambient gas. When in use, A0 terminal reads the analog value corresponding to alcohol vapor; D0 terminal is connected to an LM393 chip (comparator), we can adjust and measure the alcohol vapor alarm threshold through the potentiometer, and output the digital value at D0. When the measured alcohol vapor content exceeds the critical point, the D0 terminal outputs a low level; when the measured alcohol vapor content does not exceed the critical point, the D0 terminal outputs a high level.



**Components Required** 



**Connection Diagram** 



#### Test Code

```
/*
* Filename
           : MQ3
* Description : Read the basic usage of Digital, ADCDAC and Voltage
* Auther
          : http//www.keyestudio.com
*/
//MQ_3 two pins 13, 34, respectively
#define PIN_ANALOG_IN 34
int digitalPin = 13;
//The following two variables hold the digital signal and adc values respectively
int analogVal = 0;
int adcVal = 0;
void setup() {
 Serial.begin(9600);
 pinMode(digitalPin, INPUT); //Digital pin 13 is set to input mode
}
//In loop()the digitalRead()function is used to obtain the digital value, the
\rightarrow analogRead() function is used to obtain the ADC value. and then the map() function is
\hookrightarrowused to convert the value into an 8-bit precision DAC value.
//The input and output voltage are calculated according to the previous formula, and the
→information is finally printed out.
```

(continued from previous page)

```
void loop() {
 int digitalVal = digitalRead(digitalPin); //Read digital signal;
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("digitalVal: %d, \t ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n",
→digitalVal, adcVal, dacVal, voltage);
 if (digitalVal == 1) {
   Serial.println(" Normal");
 }
 else {
   Serial.println(" Exceeding");
 }
 delay(100); //Delay time 100 ms
}
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Rotating the potentiometer on the sensor, we can adjust the yellow and green LED bright and not bright critical point. Open the monitor, set baud rate to 9600.

We need to press the reset button on the ESP32, then the monitor displays the corresponding data and characters. When the sensor detects the alcohol gas, the yellow and green LED lights up and the digital value changes from 1 to 0, the ADC value, DAC value and voltage value decrease, as shown below.

					/dev	/ttyUS	BO						×
												Sen	d
digitalVal: Normal	1,	ADC	Val:	1783,	DAC	Val:	111,	Voltage:	1.44V				•
digitalVal: Normal	1,	ADC	Val:	1801,	DAC	Val:	112,	Voltage:	1.45V				
digitalVal: Normal	1,	ADC	Val:	1856,	DAC	Val:	116,	Voltage:	1.50V				
digitalVal: Normal	1,	ADC	Val:	1963,	DAC	Val:	122,	Voltage:	1.58V				
digitalVal: Exceeding	Θ,	ADC	Val:	2093,	DAC	Val:	130,	Voltage:	1.69V				
digitalVal: Exceeding	Θ,	ADC	Val:	2199,	DAC	Val:	137,	Voltage:	1.77V				
digitalVal: Exceeding	Θ,	ADC	Val:	2289,	DAC	Val:	143,	Voltage:	1.84V				l
digital												•	Ŧ
Autoscroll	Sh	ow tin	nestar	np		1	Vewline	•	9600 baud	▼ Cle	ar o	utpi	Jt



# 7.5.29 Project 29: Five-key AD Button Module

#### Description

When we talked about analog and digital sensors earlier, we talked about the single-channel key module. When we press the key, it outputs a low level, and when we release the key, it outputs a high level. We can only read these two digital signals. In fact, the key module ADC acquisition can also be performed. In this kit, a DIY electronic building block five-way AD button module is included.

We can judge which key is pressed through the analog value. In the experiment, we print out the key press information in the shell.

#### Working Principle

Let's look at the schematic diagram, when we do not press the key, the OUT of S output to the signal end is pulled down by R1. At this time, we read the low level 0V. When we press the key SW1, the OUT of the output to the signal end S is directly connected to the VCC. At this time, we read the high level 3.3V(the figure is marked as a 12-bit ADC(0~4095) and VCC is 5V. The principle is the same. Here we have VCC of 3.3V and ADC mapped to 12 bits), which is an analog value of 4095.

Next, when we press the key SW2, the OUT terminal voltage of the signal we read is the voltage between R2 and R1, namely VCC\*R1/(R2+R1), which is about 2.64V, and the analog value is about 3276.

When we press the key SW3, the OUT terminal voltage of the signal we read is the voltage between R2+R3 and R1, namely VCC\*R1/(R3+R2+R1), which is about 1.99V, and the analog value is about 2469.

When we press the key SW4, the OUT terminal voltage of the signal we read is the voltage between R2+R3+R4 and R1, namely VCC\*R1/(R4+R3+R2+R1), about 1.31V, and the analog value is about 1626.

Similarly, when we press the key SW5, the OUT terminal voltage of the signal we read is the voltage between R2+R3+R4+R5 and R1, namely VCC\*R1/(R5+R4+R3+R2+R1), which is about 0.68V, and the analog value is about 844.



#### **Components Required**



**Connection Diagram** 



#### **Test Code**

```
/*
* Filename : Five AD Keys
* Description : Read the value of Five AD Keys
* Auther
            : http//www.keyestudio.com
*/
int val = 0;
int ADkey = 34; //Define five AD keys connected to GPI036
void setup() {
  Serial.begin(9600); //Set baud rate to 9600
}
void loop() {
 val = analogRead(ADkey); //Read the simulated value of the AD key and assign it to.
\rightarrow the variable val
  Serial.print(val); //A newline prints the variable val
 if (val <= 500) { //Val is less than or equal to 500 when no button is pressed
   Serial.println(" no key is pressed");
  } else if (val <= 1000) { //When key 5 is pressed, val is between 500 and 1000
   Serial.println(" SW5 is pressed");
  } else if (val <= 2000) { //When pressed, val is between 1000 and 2000
   Serial.println(" SW4 is pressed");
  } else if (val <= 3000) { //When pressed, val is between 2000 and 3000
   Serial.println(" SW3 is pressed");
  } else if (val <= 4000) { //When key 2 is pressed, val is between 3000 and 4000
   Serial.println(" SW2 is pressed");
                                                                     (continues on next page)
```

```
(continued from previous page)
```

#### **Code Explanation**

We assign the read analog value to the variable val, and the serial monitor displays the value of val, (we set to 9600).

When the analog value is in the range of 500 and 1000, the button SW5 is pressed; when the analog value is in the 1000 and 2000, the button SW4 is pressed; when the analog value is between 2000 and 3000, the button SW3 is pressed; when the analog value is between 3000 and 4000, the button SW2 is pressed. When the analog value is above 4000, we judge that the button SW1 is pressed.

#### Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600. We need to press the reset button on the ESP32, when the button is pressed, the serial monitor prints out the corresponding information, as shown in the figure below.

			/dev/ttyUSB0			~		×
							Sen	nd
2326	SW3 is	pressed						
2346 2321	SW3 is SW3 is	pressed pressed						
2323	SW3 is	pressed						
2322	SW3 is	pressed						
2325 2314	SW3 is SW3 is	pressed pressed						
4095 4095	SW1 is SW1 is	pressed						
4095	SW1 is	pressed						
4095 4095	SW1 1s SW1 is	pressed pressed						ł
4095 4095	SW1 is	pressed						Į
•							)	•
🖌 Au	toscroll	Show timestamp	Newline	•	9600 baud	▼ Clea	r outp	ut

# 7.5.30 Project 30: Joystick Module



#### Overview

Game handle controllers are ubiquitous.

It mainly uses PS2 joysticks. When controlling it, we need to connect the X and Y ports of the module to the analog port of the single-chip microcomputer, port B to the digital port of the single-chip microcomputer, VCC to the power output port(3.3-5V), and GND to the GND of the MCU. We can read the high and low levels of two analog values and one digital port) to determine the working status of the joystick on the module.

In the experiment, two analog values(x axis and y axis) will be shown on Shell.

#### VCC 10 0 4 Z 8 3 0603 100NF C1 GND R1 4.7K 0603 100NF Joy VCC VCC GND GND GND GND 220R PJ1 5 4 FD 3 0603-LED VCC 2 GND 1 5pin

**Working Principle** 

In fact, its working principle is very simple. Its inside structure is equivalent to two adjustable potentiometers and a button. When this button is not pressed and the module is pulled down by R1, low levels will be output ; on the

contrary, when the button is pressed, VCC will be connected (high levels), When we move the joystick, the internal potentiometer will adjust to output different voltages, and we can read the analog value.

#### Components



#### **Connection Diagram**



#### **Test Code**

```
/*
* Filename : Jovstick
* Description : Read data from Rocker.
        : http//www.keyestudio.com
* Auther
*/
int xyzPins[] = {34, 35, 13}; //x,y,z pins
void setup() {
 Serial.begin(9600);
 pinMode(xyzPins[0], INPUT); //x axis.
 pinMode(xyzPins[1], INPUT); //y axis.
 pinMode(xyzPins[2], INPUT_PULLUP); //z axis is a button.
}
// In loop(), use analogRead () to read the value of axes X and Y and use digitalRead ().
\hookrightarrow to read the value of axis Z, then display them.
void loop() {
 int xVal = analogRead(xyzPins[0]);
 int yVal = analogRead(xyzPins[1]);
 int zVal = digitalRead(xyzPins[2]);
 Serial.println("X,Y,Z: " + String(xVal) + ", " + String(yVal) + ", " + String(zVal));
 delay(500);
}
```

#### **Code Explanation**

In the experiment, according to the wiring diagram, the x pin is set to GPIO34, the y pin is set to GPIO35 and the pin of the joystick is set to GPIO13.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600.

We need to press the reset button on the ESP32, then the serial monitor will show the corresponding value. Moving the joystick or pressing it will change the analog and digital values in the serial monitor.



# 7.5.31 Project 31: Relay Module

#### Overview

In our daily life, we usually use communication to drive electrical equipment, and sometimes we use switches to control electrical equipment. If the switch is connected directly to the ac circuit, leakage occurs and people are in danger. Therefore, from the perspective of safety, we specially designed this relay module with NO(normally open) end and NC(normally closed) end.

#### **Working Principle**

Relay is compatible with a variety of microcontroller control board, such as Arduino series microcontroller, which is a small current to control the operation of large current "automatic switch".

Input Voltage3.3V-5V



It can let the MCU control board drive 3A load, such as an LED lamp belt, a DC motor, a micro water pump and a solenoid valve plugable interface design, which is easy to use.

#### **Components Required**



**Connection Diagram** 



#### Test Code

//******************************
/*
* Filename : Relay
* Description : Relay turn on and off.
* Auther : http://www.keyestudio.com
*/
#define Relay 15 // defines digital 15
<pre>void setup()</pre>
{
<pre>pinMode(Relay, OUTPUT); // sets "Relay" to "output"</pre>
}
void loop()
{
digitalWrite(Relay, HIGH); // turns on the relay
delay(1000); //delays 1 seconds
digitalWrite(Relay, LOW); // turns off the relay
delay(1000); // delays 1 seconds
}
//*************************************

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The relay will cycle on and off, on for 1 second, off for 1 second. At the same time, you can hear the sound of the relay on and off as well as see the change of the indicator

light on the relay.

# 7.5.32 Project 32: SK6812 RGB Module



#### Overview

In previous lessons, we learned about the plug-in RGB module and used PWM signals to color the three pins of the module.

There is a Keyestudio 6812 RGB module whose the driving principle is different from the plug-in RGB module. It can only control with one pin. This is a set. It is an intelligent externally controlled LED light source with the control circuit and the light-emitting circuit. Each LED element is the same as a 5050 LED lamp bead, and each component is pixel. There are four lamp beads on the module, which indicates four pixels.

In the experiment, we make different lights show different colors.

#### **Working Principle**

From the schematic diagram, we can see that these four pixel lighting beads are all connected in series. In fact, no matter how many they are, we can use a pin to control a light and let it display any color. The pixel point contains a data latch signal shaping amplifier drive circuit, a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.

The data protocol adopts a single-wire zero-code communication method. After the pixel is powered up and reset, the S terminal receives the data transmitted from the controller. The first 24bit data sent is extracted by the first pixel and sent to the data latch of the pixel.



Components



**Connection Diagram** 



#### **Test Code**

(continued from previous page)

```
* Auther
               : http//www.keyestudio.com
*/
#include <Adafruit_NeoPixel.h>
#define PTN 15
// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
// NEO_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
// NEO_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
// NEO_GRB
             Pixels are wired for GRB bitstream (most NeoPixel products)
// NEO_RGB
             Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across pixel power.
→leads, add 300 - 500 Ohm resistor on first pixel's data input and minimize distance.
→between Arduino and first pixel. Avoid connecting on a live circuit...if you must,
→connect GND first.
void setup() {
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}
void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
  // Send a theater pixel chase in...
  theaterChase(strip.Color(127, 127, 127), 50); // White
  theaterChase(strip.Color(127, 0, 0), 50); // Red
  theaterChase(strip.Color( 0, 0, 127), 50); // Blue
  rainbow(20);
  rainbowCycle(20);
  theaterChaseRainbow(50);
}
// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {</pre>
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
 }
}
void rainbow(uint8_t wait) {
  uint16_t i, j;
```

```
(continued from previous page)
```

```
for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {</pre>
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}
// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
 uint16_t i, j;
  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel</pre>
    for(i=0; i< strip.numPixels(); i++) {</pre>
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
 }
}
//Theatre-style crawling lights.
void theaterChase(uint32_t c, uint8_t wait) {
  for (int j=0; j<10; j++) { //do 10 cycles of chasing</pre>
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
        strip.setPixelColor(i+q, c); //turn every third pixel on
      ł
      strip.show();
      delay(wait);
      for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
                                         //turn every third pixel off
        strip.setPixelColor(i+q, ◊);
      }
    }
 }
}
//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow(uint8_t wait) {
                                    // cycle all 256 colors in the wheel
  for (int j=0; j < 256; j++) {
    for (int q=0; q < 3; q++) {
        for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
          strip.setPixelColor(i+q, Wheel( (i+j) % 255)); //turn every third pixel on
        }
        strip.show();
        delay(wait);
        for (int i=0; i < strip.numPixels(); i=i+3) {</pre>
```

(continued from previous page)

```
strip.setPixelColor(i+q, 0);
                                        //turn every third pixel off
       }
   }
 }
}
// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
 if(WheelPos < 85) {</pre>
  return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
 } else if(WheelPos < 170) {</pre>
  WheelPos -= 85;
  return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
 } else {
  WheelPos -= 170;
  return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
 }
}
```

#### Test Code

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Then We can see 4 RGB leds on the module emitting various color lighting effects.

### 7.5.33 Project 33: Rotary Encoder



#### Overview

In this kit, there is a Keyestudio rotary encoder, dubbed as switch encoder. It is applied to automotive electronics, multimedia audio, instrumentation, household appliances, smart home, medical equipment and so on.

In the experiment, it it used for counting. When we rotate the rotary encoder clockwise, the set data is up 1; if you

rotate it anticlockwise, the set data falls by 1; and when the middle button is pressed, the value will be show in the serial monitor.

#### **Working Principle**



The incremental encoder converts the displacement into a periodic electric signal, and then converts this signal into a counting pulse, and the number of pulses indicates the size of the displacement. This module mainly uses 20-pulse rotary encoder components. It can calculate the number of pulses output during clockwise and reverse rotation. There is no limit to count rotation. It resets to the initial state, that is, starts counting from 0.

#### Components



**Connection Diagram** 



#### **Test Code**

```
/*
* Filename : Encoder
* Description : Rotary encoder module counting.
* Auther : http//www.keyestudio.com
*/
//Interfacing Rotary Encoder with Arduino
//Encoder Switch -> pin 27
//Encoder DT -> pin 14
//Encoder CLK -> pin 12
int Encoder_DT = 14;
int Encoder_CLK = 12;
int Encoder_Switch = 27;
int Previous_Output;
int Encoder_Count;
void setup() {
 Serial.begin(9600);
 //pin Mode declaration
 pinMode (Encoder_DT, INPUT);
 pinMode (Encoder_CLK, INPUT);
                                                                (continues on next page)
```

(continued from previous page)

```
pinMode (Encoder_Switch, INPUT);
 Previous_Output = digitalRead(Encoder_DT); //Read the inital value of Output A
}
void loop() {
 //aVal = digitalRead(pinA);
 if (digitalRead(Encoder_DT) != Previous_Output)
 {
   if (digitalRead(Encoder_CLK) != Previous_Output)
   {
     Encoder_Count ++;
     Serial.println(Encoder_Count);
   }
   else
    {
     Encoder_Count--;
     Serial.println(Encoder_Count);
   }
 }
 Previous_Output = digitalRead(Encoder_DT);
 if (digitalRead(Encoder_Switch) == 0)
 {
   delay(5);
   if (digitalRead(Encoder_Switch) == 0) {
     Serial.println("Switch pressed");
     while (digitalRead(Encoder_Switch) == 0);
    }
 }
}
                           *************************
```

#### **Code Explanation**

Set CLK to GPIO12 and DAT to GPIO14.

This code is set well in the library file. When CLK descends, read the voltage of DAT, when DAT is a HIGH level, the value of the rotary encoder is added by 1; when DAT is a LOW level, the value of the rotary encoder is cut down 1.

Set the pin of the button(GPIO27) to LOW and print.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600.

We need to press the reset button on the ESP32, then rotate the knob on the rotary encoder clockwise, the displayed data will rise; on the contrary, in anticlockwise way, the data will decrease. Equally, press the button on the rotary encoder, "Switch pressed" will be shown.

/dev/ttyUSB0	~	^	×
		Ser	nd
0			•
1 2			_
1			ł
1			I
0 -1			ł
-2			ł
-2			I
-3 Switch pressed			I
Switch pressed			
•			>
✓ Autoscroll □ Show timestamp       Newline       ♥ 9600 baud       ✔ Cleater	ar o	utp	ut

# 7.5.34 Project 34: Servo Control



#### Overview

Servo is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor.

In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.



#### **Working Principle**

When the motor speed is constant, the potentiometer is driven to rotate through the cascade reduction gear, which leads that the voltage difference is 0, and the motor stops rotating. Generally, the angle range of servo rotation is  $0^{\circ} -180^{\circ}$ 

The rotation angle of servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms (50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle from  $0^{\circ}$  to  $180^{\circ}$ . But note that for different brand motors, the same signal may have different rotation angles.



Components


### **Connection Diagram**



#### Test Code 1

```
//***************
/*
* Filename : Servo_1
* Description : Steering gear rotation Angle 0-90-180, repeatly
* Auther : http//www.keyestudio.com
*/
int servoPin = 4;//steering gear PIN
void setup() {
 pinMode(servoPin, OUTPUT);//steering pin is set to output
}
void loop() {
 servopulse(servoPin, 0);//Rotate it to zero degrees
 delay(1000);//delay 1S
 servopulse(servoPin, 90);//Rotate it to 90 degrees
 delay(1000);
 servopulse(servoPin, 180);//Rotate it to 180 degrees
 delay(1000);
}
void servopulse(int pin, int myangle) { //Impulse function
 int pulsewidth = map(myangle, 0, 180, 500, 2500); //Map Angle to pulse width
 for (int i = 0; i < 10; i++) { //Output a few more pulses
```

(continues on next page)

#### **Code Explanation 1**

#### 1). map(value, fromLow, fromHigh, toLow, toHigh)

Value is the value we map. fromLow, fromHigh is the maximum and minimum value

toLow, toHigh are the upper limit and lower limit we map. For example, map(myangle, 0, 180, 500, 2500) means that an angle value myangle  $(0^{\circ}-180^{\circ}$ the mapping range is from 500us to 2500us.

#### 2). servopulse()

We use the function servopulse() to make the servo move. We also make the servo rotate  $0^\circ$ ,  $90^\circ$  and  $180^\circ$  cyclically.

#### **Test Result 1**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, the servo will rotate  $0^{\circ}90^{\circ}$  and  $180^{\circ}$  cyclically.

#### Test Code 2

```
/*
* Filename : Servo Sweep
* Description : Control the servo motor for sweeping
* Auther
           : http//www.keyestudio.com
*/
#include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int posVal = 0; // variable to store the servo position
int servoPin = 4; // Servo motor pin
void setup() {
                                   // standard 50 hz servo
 myservo.setPeriodHertz(50);
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→ object
}
void loop() {
 for (posVal = 0; posVal <= 180; posVal += 1) { // goes from 0 degrees to 180 degrees
   // in steps of 1 degree
   myservo.write(posVal);
                              // tell servo to go to position in variable 'pos'
   delay(15);
                              // waits 15ms for the servo to reach the position
 }
 for (posVal = 180; posVal >= 0; posVal -= 1) { // goes from 180 degrees to 0 degrees
   myservo.write(posVal);
                             // tell servo to go to position in variable 'pos'
```

### **Code Explanation 2**

myservo. write (pos) is the rotation angle to POS. myservo. read () reads the current angle value of the servo.

#### Test Result 2

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on, the servo will rotate from  $0^{\circ}$  to  $180^{\circ}$  by moving  $1^{\circ}$  for each 15ms.

# 7.5.35 Project 35: Ultrasonic Sensor



#### Overview

Bats and some marine animals are able to use high frequencies of sound for echolocation or communication. They can emit ultrasonic waves from the larynx through the mouth or nose and use the sound waves that bounce back to orient and determine the position, size and whether nearby objects are moving.

Ultrasonic is a frequency higher than 20000 Hz sound wave, which has a good direction, a strong penetration ability, and is easy to obtain more concentrated sound energy as well as spread far in the water. It can be used for ranging, speed measurement, cleaning, welding, gravel, sterilization and disinfection. What's more, it has many applications in medicine, military, industry and agriculture.

In this kit, there is a keyes HC-SR04 ultrasonic sensor, which can detect obstacles in front and the detailed distance between the sensor and the obstacle. Its principle is the same as that of bat flying. It can emit the ultrasonic signals that cannot be heard by humans. When these signals hit an obstacle and come back immediately. The distance between the sensor and the obstacle can be calculated by the time gap of emitting signals and receiving signals.

In the experiment, we use the sensor to detect the distance between the sensor and the obstacle, and print the test result.

#### **Working Principle**

The most common ultrasonic ranging method is the echo detection. As shown below; when the ultrasonic emitter emits the ultrasonic waves towards certain direction, the counter will count. The ultrasonic waves travel and reflect back once encountering the obstacle. Then the counter will stop counting when the receiver receives the ultrasonic waves coming back.

The ultrasonic wave is also sound wave, and its speed of sound V is related to temperature. Generally, it travels 340m/s in the air. According to time t, we can calculate the distance s from the emitting spot to the obstacle. s = 340t/2 The HC-SR04 ultrasonic ranging module can provide a non-contact distance sensing function of 2cm-400cm, and the ranging accuracy can reach as high as 3mm; the module includes an ultrasonic transmitter, receiver and control circuit. Basic working principle:

1). First pull down the TRIG, and then trigger it with at least 10us high level signal;

2). After triggering, the module will automatically transmit eight 40KHZ square waves, and automatically detect whether there is a signal to return.

3). If there is a signal returned back, through the ECHO to output a high level, the duration time of high level is actually the time from emission to reception of ultrasonic. TestDistance = HighLevelDuration \* 340m/s \* 0.5



#### Components



**Connection Diagram** 



### **Test Code**

```
/*
* Filename : Ultrasonic
* Description : Use the ultrasonic module to measure the distance.
* Auther : http//www.keyestudio.com
*/
const int TrigPin = 13; // define TrigPin
const int EchoPin = 14; // define EchoPin.
int duration = 0; // Define the initial value of the duration to be 0
int distance = 0;//Define the initial value of the distance to be 0
void setup()
{
 pinMode(TrigPin , OUTPUT); // set trigPin to output mode
 pinMode(EchoPin , INPUT); // set echoPin to input mode
 Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
}
void loop()
{
// make trigPin output high level lasting for 10s to triger HC_SR04
 digitalWrite(TrigPin , HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin , LOW);
 // Wait HC-SR04 returning to the high level and measure out this waitting time
 duration = pulseIn(EchoPin , HIGH);
 // calculate the distance according to the time
 distance = (duration/2) / 28.5;
 Serial.print("Distance: ");
 Serial.print(distance); //Serial port print distance value
 Serial.println("cm");
```

(continues on next page)

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600.

We need to press the reset button on the ESP32, then the serial monitor will print the distance between the ultrasonic sensor and the object.

	/dev/ttyUSB0	•	^	×
			Ser	nd
Distance:	8cm			•
Distance:	7cm			
Distance:	7cm			
Distance:	7cm			
Distance:	6cm			
Distance:	5cm			
Distance:	5cm			
Distance:	6cm			
Distance:	7cm			
Distance:	8cm			
Distance:	8cm			
Distance:	12cm			
Distance:	10cm			
Distance:	24cm			
				Ŧ
•				•
🖌 Autoscr	oll 🗌 Show timestamp Newline 🔹 9600 baud 💌 Cl	ear	outp	ut



# 7.5.36 Project 36: IR Receiver Module

# Overview

Infrared remote control is currently the most widely used means of communication and remote control, which has the characteristics of small volume, low power consumption, strong function and low cost. Therefore, recorder, audio equipment, air conditioning machine and toys and other small electrical devices have also used the infrared remote control.

Its transmitting circuit is the use of infrared light emitting diode to emit modulated infrared light wave. The circuit is composed of infrared receiving diode, triode or silicon photocell. They convert infrared light emitted by infrared emitter into corresponding electrical signal, and then send back amplifier.

In this experiment, we need to know how to use the infrared receiving sensor. The infrared receiving sensor mainly uses the VS1838B infrared receiving sensor element. It integrates receiving, amplifying, and demodulating. The internal IC has already completed the demodulation, and the output is a digital signal. It can receive 38KHz modulated remote control signal.

In the experiment, we use the IR receiver to receive the infrared signal emitted by the external infrared transmitting device, and display the received signal in the shell.

# Working Principle

The main part of the IR remote control system is modulation, transmission and reception. The modulated carrier frequency is generally between 30khz and 60khz, and most of them use a square wave of 38kHz and a duty ratio of 1/3. A 4.7K pull-up resistor R3 is added to the signal end of the infrared receiver.



Components



**Connection Diagram** 



```
Test Code
```

```
/*
* Filename : IR Receiver
* Description : Decode the infrared remote control and print it out through the serial
\rightarrow port.
             : http//www.keyestudio.com
* Auther
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
const uint16_t recvPin = 15; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results;
                         // Create a decoding results class object
void setup() {
 Serial.begin(9600);
                        // Initialize the serial port and set the baud rate to 9600
 irrecv.enableIRIn(); // Start the receiver
 Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
 Serial.println(recvPin); //print the infrared receiving pin
}
void loop() {
 if (irrecv.decode(&results)) { // Waiting for decoding
   serialPrintUint64(results.value, HEX);// Print out the decoded results
   Serial.println("");
                                                                   (continues on next page)
```

				(contin	ued from p	orevia	ous page)
<pre>irrecv.resume();</pre>	// Release	the	IRremote.	Receive	the ne	xt	value
}							
delay(1000);							
}							
//*************************************	****	****	*****	*****	*****	* * *	

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600; Find the infrared remote control, pull out the insulating sheet, and press the button at the receiving head of the infrared receiving sensor. After receiving the signal, the LED on the infrared receiving sensor also starts to flash, as shown in the figure below.

/dev/ttyUSB0		^	×
	5	Sen	nd
<pre>\$</pre>	\$ □		~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Autoscroll Show timestamp Newline   9600 baud   Clea	r ou	utp	• ut

Write down the key code value associated with the infrared remote with each key, as you will need this information later.



7.5.37 Project 37: DS18B20 Temperature Sensor



# Description

In this kit, there is a DS18B20 temperature sensor, which is from maxim. The MCU can communicate with the DS18B20 through 1-Wire protocol, and finally read the temperature. In this experiment, we will use this temperature sensor to measure the temperature in the current environment. The test result is °C, ranging from -55°C to +125°C. We will display the test result on shell.

# **Working Principle**



The hardware interface of the 1-Wire bus is very simple, just connect the data pin of the DS18B20 to an IO port of the microcontroller. The timing of the 1-Wire bus is relatively complex. Many students can't understand the timing diagram independently here. We have encapsulated the complex timing operations in the library, and you can use the library functions directly.

## Schematic Diagram of DS18B20

This can save up to 12-bit temperature vale. In the register, save in code complement. As shown below;



A total of 2 bytes, LSB is the low byte, MSB is the high byte, where MSb is the high byte of the byte, LSb is the low byte of the byte. As you can see, the binary number, the meaning of the temperature represented by each bit, is expressed. Among them, S represents the sign bit, and the lower 11 bits are all powers of 2, which are used to represent the final temperature. The temperature measurement range of DS18B20 is from -55 degrees to +125 degrees, and the expression form of temperature data, S represents positive and negative temperature, and the resolution is 2, which is 0.0625.

#### **Required Components**



**Required Components** 



**Test Code** 

(continues on next page)

```
* Auther
            : http//www.keyestudio.com
*/
#include <DS18B20.h>
//ds18b20 pin to 15
DS18B20 ds18b20(15);
void setup() {
 Serial.begin(9600);
}
void loop() {
 double temp = ds18b20.GetTemp();//Read the temperature
 temp *= 0.0625;//The conversion accuracy is 0.0625/LSB
 Serial.print("Temperature: ");
 Serial.print(temp);
 Serial.println("C");
 delay(1000);
}
```

### **Code Explanation**

1). We set the pin to GPIO15 and obtain the temperature in the unit of °C.

2). Set a double decimal variable to temp, and assign the measured result to temp.

3). The serial monitor displays the temp value, and the baud rate needs to be set before displaying (our default setting is 9600, which can be changed).

4). We add the unit behind the data. If the unit is directly set to °C, the test result will be garbled. So we directly replace °C with C.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600. We need to press the reset button on the ESP32, then the monitor will display the temperature of the current environment, as shown below.

		/dev/ttyUSB0				~	~ :	×
							Sen	d
Temperature:	29.310							•
Temperature:	29.37C							
Temperature:	29.37C							
Temperature:	29.44C							
Temperature:	29.50C							I
Temperature:	29.620							
Temperature:	29.620							
Temperature:	29.69C							
Temperature:	29.69C							
Temperature:	29.81C							
Temperature:	30.06C							
Temperature:	30.12C							
Temperature:	30.62C							
Temperature:	30.62C							
								Ŧ
•							•	
Autoscroll	Show timestamp	Newline	•	9600 baud	▼ Cle	ear o	utpu	Jt

# 7.5.38 Project 38: XHT11 Temperature and Humidity Sensor



# Description

This DHT11 temperature and humidity sensor is a composite sensor which contains a calibrated digital signal output

of the temperature and humidity.

DHT11 temperature and humidity sensor uses the acquisition technology of the digital module and temperature and humidity sensing technology, ensuring high reliability and excellent long-term stability.

It includes a resistive element and a NTC temperature measuring device.



#### **Working Principle**

The communication and synchronization between the single-chip microcomputer and XHT11 adopts the single bus data format. The communication time is about 4ms. The data is divided into fractional part and integer part.

Operation process: A complete data transmission is 40bit, high bit first out. Data format: 8bit humidity integer data + 8bit humidity decimal data + 8bit temperature integer data + 8bit temperature decimal data + 8bit checksum 8-bit checksum: 8-bit humidity integer data + 8-bit humidity decimal data + 8-bit temperature integer data + 8-bit temperature decimal data + 8-bit temperature data + 8-bit temperature data + 8-bit temperature data + 8-bit temperature

#### **Required Components**



#### **Connection Diagram**



#### **Test Code**

```
/*
* Filename : xht11
* Description : Read the temperature and humidity values of XHT11.
* Auther
           : http//www.keyestudio.com
*/
#include "xht11.h"
//gpio15
xht11 xht(15);
unsigned char dht[4] = {0, 0, 0, 0}; //Only the first 32 bits of data are received, not
→the parity bits
void setup() {
 Serial.begin(9600);//Start the serial port monitor and set baud rate to 9600
}
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [1] is the fractional part
   Serial.print("% ");
   Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
→part
   Serial.println("C");
 } else {
           //Read error
```

(continues on next page)

## **Code Explanation**

1). We set the pin to GPIO15, and store the detected temperature and humidity data in the dht[4] array.

2). We add units behind the data. If the temperature unit is directly set to  $^{\circ}$ C, the test results may be wrong, so we directly replace  $^{\circ}$ C with C; the humidity unit is directly set to  $^{\circ}$ .

#### Test Result

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 9600. We need to press the reset button on the ESP32, then the monitor will display the temperature and humidity data of the current environment, as shown below.

	/dev/ttyUSB0		^	×
			Ser	nd
RH:57%	Temp:28C			•
RH:67%	Temp:28C			
RH:73%	Temp:28C			
RH:79%	Temp:28C			
RH:83%	Temp:28C			
RH:85%	Temp:28C			
RH:88%	Temp:29C			
RH:90%	Temp:29C			
RH:90%	Temp:29C			
RH:92%	Temp:29C			
RH:93%	Temp:30C			
RH:94%	Temp:30C			
RH:94%	Temp:30C			
RH:95%	Temp:30C			
				Ŧ
4		1	1	•
🗹 Autos	scroll 🗌 Show timestamp Newline 💌 9600 baud 💌	Clear	outp	ut

(continued from previous page)



# 7.5.39 Project 39: DS1307 Clock Module

#### Overview

This module mainly uses the real-time clock chip DS1307, which is the I2C bus interface chip that has second, minute, hour, day, month, year and other functions as well as leap year automatic adjustment function introduced by DALLAS. It can work independently of CPU, and won't' affected by the CPU main crystal oscillator and capacitance as well as keep accurate time. What's more, monthly cumulative error is generally less than 10 s. The chip also has a clock protection circuit in case of main power failure and runs on a back-up battery that denies the CPU read and write access.

At the same time, it contains automatic switching control circuit of standby power supply, making it guarantees the accuracy of system clock in case of power failure of main power supply and other bad environment.

Going forward, the DS1307 chip internal integration has a certain capacity, with power failure protection characteristics of static RAM, which can be used to save some key data.

In the experiment, we use the DS1307 clock module to obtain the system time and print the test results.



## **Working Principle**

Serial real-time clock records year, month, day, hour, minute, second and week; AM and PM indicate morning and afternoon respectively; 56 bytes of NVRAM store data; 2-wire serial port; programmable square wave output; power failure detection and automatic switching circuit; battery current is less than 500nA.

Pins description

- X1, X2: 32.768kHz crystal terminal ;
- VBAT+3V input;
- SDAserial data;
- SCLserial clock;
- SQW/OUTsquare waves/output drivers

#### Components



**Connection Diagram** 



#### **Test Code**

```
//***********************
/*
* Filename : DS1307 Real Time Clock
* Description : Read the year/month/day/hour/minute/second/week of DS1307 clock module
* Auther : http//www.keyestudio.com
*/
#include <Wire.h>
#include "RtcDS1307.h" //DS1307 clock module library
RtcDS1307<TwoWire> Rtc(Wire);//i2cport
void setup(){
 Serial.begin(57600);//Set baud rate to 57600
 Rtc.Begin();
 Rtc.SetIsRunning(true);
 Rtc.SetDateTime(RtcDateTime(__DATE__, __TIME__));
}
void loop(){
 // Print year/month/day/hour/minute/second/week
                                                                   (continues on next page)
```

```
Serial.print(Rtc.GetDateTime().Year());
Serial.print("/");
Serial.print(Rtc.GetDateTime().Month());
Serial.print("/");
Serial.print(Rtc.GetDateTime().Day());
              ");
Serial.print("
Serial.print(Rtc.GetDateTime().Hour());
Serial.print(":");
Serial.print(Rtc.GetDateTime().Minute());
Serial.print(":");
Serial.print(Rtc.GetDateTime().Second());
Serial.print("
               ");
Serial.println(Rtc.GetDateTime().DayOfWeek());
delay(1000);//Delay 1 second
```

# Code Explanation

}

Rtc.GetDateTime(): the obtained current time and date.

Rtc.Begin(); enable DS1307 real-time clock.

Rtc.SetIsRunning(true); run the DS1307 real-time clock, if true changes into false, time will stop.

Rtc.SetDateTime()set time.

Rtc.GetDateTime().Year(): return year.

Rtc.GetDateTime().Month(): return month.

Rtc.GetDateTime().Day(): return date.

Rtc.GetDateTime().Hour(): return hour.

Rtc.GetDateTime().Minute(): return minute.

Rtc.GetDateTime().Second(): return second.

Rtc.GetDateTime().DayOfWeek(): return week.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, attach the DS1307 sensor to a battery, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 57600. We need to press the reset button on the ESP32, then we can see the displayed year, month, day, hour, minute, second and week on the monitor, and set the time and date to refresh every second, as shown below:

			/dev/ttyUSB0	~	<b>^</b> :	×
					Sen	d
2022/4/7	6:59:5	4				٠
2022/4/7	6:59:6	4				
2022/4/7	6:59:7	4				
2022/4/7	6:59:8	4				
2022/4/7	6:59:9	4				
2022/4/7	6:59:10	4				
2022/4/7	6:59:11	4				
2022/4/7	6:59:12	4				
2022/4/7	6:59:13	4				
2022/4/7	6:59:14	4				
2022/4/7	6:59:15	4				
2022/4/7	6:59:16	4				
2022/4/7	6:59:17	4				
2022/4/7	6:59:18	4				
2022/4/7	6:59:19	4				
2022/4/7	6:59:20	4				
						Ŧ
•					•	-
🖌 Autoscrol	ll 🗌 Show tir	mestamp	Newline 🔻 57600 baud 💌	Clear	outpu	Jt

# 7.5.40 Project 40: ADXL345 Acceleration Sensor



# Overview

In this kit, there is a DIY electronic building block ADXL345 acceleration sensor module, which uses the ADXL345BCCZ chip. The chip is a small, thin, low-power 3-axis accelerometer with a high resolution (13 bits) and a measurement range of  $\pm 16g$  that can measure both dynamic acceleration due to motion or impact as well as stationary acceleration such as gravitational acceleration, making the device usable as a tilt sensor.

# **Working Principle**

The ADXL345 is a complete 3-axis acceleration measurement system with a selection of measurement ranges of  $\pm 2$  g,  $\pm 4$  g,  $\pm 8$  g or  $\pm 16$  g. Its digital output data is in 16-bit binary complement format and can be accessed through an SPI (3-wire or 4-wire) or I2C digital interface.

The sensor can measure static acceleration due to gravity in tilt detection applications, as well as dynamic acceleration due to motion or impact. Its high resolution (3.9mg/LSB) enables measurement of tilt Angle changes of less than  $1.0^{\circ}$ .



## **Components Required**



**Connection Diagram** 



#### **Test Code**

```
/*
* Filename : ADXL345
* Description : Read the X/Y/Z value of ADXL345
* Auther : http//www.keyestudio.com
*/
#include "adx1345_io.h"
//The port is sda-->21,scl-->22
adxl345 adxl345(21, 22);
float out_X, out_Y, out_Z;
void setup() {
 Serial.begin(57600);//Start serial port monitoring and set baud rate to 57600
 adxl345.Init();
}
void loop() {
 adx1345.readXYZ(&out_X, &out_Y, &out_Z);
 Serial.print(out_X);
 Serial.print("g ");
 Serial.print(out_Y);
                                                               (continues on next page)
```

```
Serial.print("g ");
Serial.print(out_Z);
Serial.println("g");
delay(100);
```

#### **Code Explanation**

}

Set 3 decimal variables  $out_X out_Y out_Z$ , and assign the measured result to  $out_X out_Y out_Z$ . The serial monitor displays the value of  $out_X out_Y out_Z$ , and the baud rate needs to be set before displaying (our default setting is 9600, which can be changed).

Adxl345.Init; Initialize the ADXX345 accelerometer

adx1345.readXYZ(&out\_X, &out\_Y, &out\_Z);\*\*

Get the acceleration value of the X axis and return it to the variables out\_X, out\_Y, out\_Z

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set baud rate to 57600.

We need to press the reset button on the ESP32, then the serial monitor displays the value corresponding to the sensor, the unit is mg, as shown in the figure below.

			/dev/ttyUSB0	~	^	×
					Sen	id
48.00g 66.00g 19.00g -52.00g 141.00g 234.00g 190.00g 97.00g 48.00g 127.00g 274.00g 208.00g 118.00g 73.00g 41.00g	-183.00g 4.00g 2 157.00g 140.00g 104.00g 129.00g -189.00g -237.00g -187.00g -153.00g -141.00g 83.00g 202.00g 254.00g 249.00g	208.00g 79.00g 237.00g -44.00g 123.00g 193.00g 72.00g -110.00g -186.00g -92.00g 17.00g 126.00g 99.00g 115.00g 116.00g				Î
•					)	•
🖌 Autos	croll 🗌 Show	/ timestamp	Newline 👻 57600 baud 💌 0	Clear	outp	ut

# 7.5.41 Project 41: TM1650 4-Digit Tube Display



**Overview** This module is mainly composed of a 0.36 inch red common cathode 4-digit digital tube, and its driver chip is TM1650. When using it, we only need two signal lines to make the single-chip microcomputer control a 4-bitdigit tube, which greatly saves the IO port resources of the control board.

TM1650 is a special circuit for LED (light emitting diode display) drive control. It integrates MCU input and output control digital interface, data latch, LED drivers, keyboard scanning, brightness adjustment and other circuits.

TM1650 has stable performance, reliable quality and strong anti-interference ability.

It can be applied to the application of long-term continuous working for 24 hours.

TM1650 uses 2-wire serial transmission protocol for communication (note that this data transmission protocol is not a standard I2C protocol). The chip can drive the digital tube and save MCU pin resources through two pins and MCU communication.

#### **Working Principle**

TM1650 adopts IIC treaty, which uses DIO and CLK buses.



Data command setting: 0x48 means that we light up the digital tube, instead of enable the function of key scanning

B7	B6	B5	Β4	B3	B2	B1	во	Function	Description
$\times$	0	0	0		×	×			Eight-level brightness
$\times$	0	0	1		×	×			One-level brightness
×	0	1	0		×	×			Two-level brightness
$\times$	0	1	1		×	×		Drightness setting	Three-level brightness
$\times$	1	0	0		×	×		Brightness setting	Four-level brightness
$\times$	1	0	1		×	×			Five-level brightness
$\times$	1	1	0		×	×			Six-level brightness
$\times$	1	1	1		×	×			Seven-level brightness
$\times$				0	×	×		7/8 segment	8-segment display way
$\times$				1	×	×		display control bit	7-segment display way
$\times$					×	×	0	ON/OFF display bit	Off display
$\times$					×	×	1	ON/OFF display bit	On display

# **Command display setting:**

bit[6:4]: set the brightness of tube display, and 000 is brightest

bit[3]: set to show decimal points

bit[0]: start the display of the tube display

Components



**Connection Diagram** 



**Test Code** 

```
/*
* Filename : TM1650 Four digital tube
* Description : TM1650 Four Digital Tube shows 0-9999
           : http//www.keyestudio.com
* Auther
*/
#include "TM1650.h"
#define CLK 22
                 //pins definitions for TM1650 and can be changed to other ports
#define DIO 21
TM1650 DigitalTube(CLK,DIO);
void setup(){
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, O=display off, 1=display on,
\rightarrow default : 1
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b); //DigitalTube.clearBit(0 to 3); Clear bit display.
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>⊶--9</u>
}
void loop(){
 for(int num=0; num<10000; num++){</pre>
   displayFloatNum(num);
   delay(100);
 }
}
void displayFloatNum(float num){
 if(num > 9999)
   return;
 int dat = num*10;
  //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return:
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
```

(continues on next page)

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The 4-digit tube display will show integer from 0 to 99999, add 1 for each 10ms. Increase to 9999 then start from 0.

# 7.5.42 Project 42: HT16K33\_8X8 Dot Matrix Module



### Overview

What is the dot matrix display?

If we apply the previous circuit, there will be must one IO port to control only one LED. When more LED need to be controlled, we may adopt a dot matrix. The 8X8 dot matrix is composed of 64 light-emitting diodes, and each light-

emitting diode is placed at the intersection of the row line and the column line. Refer to the experimental schematic diagram below, when the corresponding column is set to a high level and a certain row to low, the corresponding diode will light up... For instance, set pin 13 to a high level and pin 9 to low, and then the first LED will light up. In the experiment, we display icons via this dot matrix.

#### Working Principle

As the schematic diagram shown, to light up the LED at the first row and column, we only need to set C1 to high level and R1 to low level. To turn on LEDs at the first row, we set R1 to low level and C1-C8 to high level.

16 IO ports are needed, which will highly waste the MCU resources.

Therefore, we designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, which greatly saves the resources of the single-chip microcomputer.



There are three DIP switches on the module, all of which are set to I2C communication address. The setting method is shown below. A0A1 and A2 are grounded, that is, the address is 0x70.

A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	
0 ( OFF )	0(OFF)	0(OFF)	1 (ON)	0(OFF)	0 ( OFF )	0(OFF)	1 (ON)	0 ( OFF )	
	0X70			0X71		0X72			
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	
1 (ON)	1 (ON)	0(OFF)	0(OFF)	0(OFF)	1 (ON)	1 (ON)	0(OFF)	1 (ON)	
	0X73			0X74		0X75			
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)				
0 ( OFF )	1 (ON)								
	0X76			0X77					

Components



**Connection Diagram** 



**Test Code** 

```
/*
* Filename : 8×8 Dot-matrix Display
* Description : 8x8 LED dot matrix display"Heart" pattern.
* Auther : http//www.keyestudio.com
*/
#include "HT16K33_Lib_For_ESP32.h"
#define SDA 21
#define SCL 22
ESP32_HT16K33 matrix = ESP32_HT16K33();
//The brightness values can be set from 1 to 15, with 1 darkest and 15 brightest
#define A 15
byte result[8][8];
byte test1[8] = {0x00,0x42,0x41,0x09,0x09,0x41,0x42,0x00};
void setup()
{
 matrix.init(0x70, SDA, SCL);//Initialize matrix
 matrix.showLedMatrix(test1,0,0);
 matrix.show();
}
void loop()
{
 for (int i = 0; i <= 7; i++)
 {
   matrix.setBrightness(i);
   delay(100);
 }
 for (int i = 7; i > 0; i--)
 {
   matrix.setBrightness(i);
   delay(100);
 }
}
```

#### **Code Explanation**

First we need to import the library file.

The pattern in our code is an array of byte data type, which is shown in the table below. We convert into binary, and fill in the 8\*8 form below to make it clear. 1 means on, 0 means off. Then we can see that it is a smile shape.

0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Then the dot matrix displays a "smile" pattern.

# 7.5.43 Project 43: LCD\_128X32\_DOT Module



#### Description

This is a 128\*32 pixel LCD module, which uses IIC communication mode and ST7567A driver chip. At the same time, the code contains all the English letters and common symbols of the library that can be directly called. When used, we can also set English letters and symbols to display different text sizes in our code. To make it easy to set up the pattern display, we also provide a mold capture software that can convert a specific pattern into control code and then copy it directly into the test code for use.

In the experiment, we will set up the display screen to display various English words, common symbols and numbers.

#### **Working Principle**


The module uses the IIC communication principle, the underlying functions have been encapsulated in the library surface, we can directly call the library function, if interested, you can also go to understand the underlying driver of the module.

#### Components



#### **Connection Diagram**



# **Test Code**

```
/*
* Filename : lcd128*32
* Description : Lcd128 *32 Displays character strings
* Auther : http//www.keyestudio.com
*/
#include "lcd128_32_io.h"
//Create lcd12832 examples,sda--->21 scl--->22
lcd lcd(21, 22);
void setup() {
   lcd.Init(); //initialize
   lcd.Clear(); //cls
 }
void loop() {
   lcd.Cursor(0, 7); //Set display position
   lcd.Display("KEYES"); //Setting the display
```

(continues on next page)

```
lcd.Cursor(1, 0);
lcd.Display("ABCDEFGHIJKLMNOPQR");
lcd.Cursor(2, 0);
lcd.Display("123456789+-*/<>=$@");
lcd.Cursor(3, 0);
lcd.Display("%^&(){}:;'|?,.~\\[]");
}
```

### **Code Explanation**

First import the library file

.Init(): initializes the display screen;

.Clear(): clears the display;

.Cursor(): sets the display position;

.Display(): displays characters.

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32.

After uploading successfully, we will use a USB cable to power on. The first line of the 128X32LCD module display shows "KEYES", the second line shows "ABCDEFGHIJKLMNOPQR", and the third line shows "123456789 $\pm$ \*/<> =\$@", the fourth line displays "%^&(){};;']?,.~\[]".

# 7.5.44 Project 44: RFID Module



#### Description

RFIDRFID-RC522 radio frequency module adopts a Philips MFRC522 original chip to design card reading circuit, easy to use and low cost, suitable for equipment development and card reader development and so on.

RFID or Radio Frequency Identification system consists of two main components, a transponder/tag attached to an object to be identified, and a transceiver also known as interrogator/Reader.

In the experiment, the data read by the card swipe module is 4 hexadecimal numbers, and we print these four hexadecimal numbers as strings. For example, we read the data of the IC card below:  $0 \times ED0 \times F70 \times 940 \times 5A$  and the information string displayed in the serial monitor is ED F7 94 5A; the data read from the keychain is:  $0 \times 4C0 \times 090 \times 6B0 \times 6E$ . Different IC cards and different key chains have diverse data.

#### **Working Principle**

Radio frequency identification, the card reader is composed of a radio frequency module and a high-level magnetic field. The Tag transponder is a sensing device, and this device does not contain a battery. It only contains tiny integrated circuit chips and media for storing data and antennas for receiving and transmitting signals. To read the data in the tag, first put it into the reading range of the card reader. The reader will generate a magnetic field, and because the magnetic energy generates electricity according to Lenz's law, the RFID tag will supply power, thereby activating the device.



**Connection Diagram** 



#### **Test Code**

```
/*
* Filename : RFID
* Description : RFID reader UID
* Auther : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
void setup() {
 Serial.begin(115200); // initialize and PC's serial communication
                           // initialize I2C
 Wire.begin();
 mfrc522.PCD_Init();
                           // initialize MFRC522
 ShowReaderDetails();
                           // dispaly PCD - MFRC522 read carder
 Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
}
void loop() {
```

(continues on next page)

```
11
 if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return;
 }
 // select one of door cards. UID and SAK are mfrc522.uid.
 // save UID
 Serial.print(F("Card UID:"));
 for (byte i = 0; i < mfrc522.uid.size; i++) {
   Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   Serial.print(mfrc522.uid.uidByte[i], HEX);
 }
 Serial.println();
}
void ShowReaderDetails() {
 // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print(F("MFRC522 Software Version: 0x"));
 Serial.print(v, HEX);
 if (v == 0x91)
   Serial.print(F(" = v1.0"));
 else if (v == 0x92)
   Serial.print(F(" = v2.0"));
 else
   Serial.print(F(" (unknown)"));
 Serial.println("");
 // when returning to 0x00 or 0xFF, may fail to transmit communication signals
 if ((v == 0x00) || (v == 0xFF)) {
   Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?
→"));
 }
}
```

# **Code Explanation**

Wire.begin(); The module we use is the IIC interface, so we first initialize the IIC

mfrc522.PCD\_Init(); initialize MFRC522

String(mfrc522.uid.uidByte[i], HEX); A string to convert the value read into hexadecimal format.

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set baud rate to 115200. We need to press the reset button on the ESP32, when we make the IC card close to the RFID module, the information will be printed out, as shown in the figure below.

							/dev/tty	USB0				~	^	×
													Ser	nd
Card	UID:	4C	09	6B	6E									-
Card	UID:	4C	09	6B	6E									
Card	UID:	4C	09	6B	6E									
Card	UID:	4C	09	6B	6E									
Card	UID:	4C	09	6B	6E									
Card	UID:	4C	09	6B	6E									
Card	UID:	4C	09	6B	6E									
Card	UID:	4C	09	6B	6E									
Card	UID:	ED	F7	94	5A									
Card	UID:	ED	F7	94	5A									
Card	UID:	ED	F7	94	5A									
Card	UID:	ED	F7	94	5A									
Card	UID:	ED	F7	94	5A									
Card	UID:	ED	F7	94	5A									
Card	UID:	ED	F7	94	5A									
														Ŧ
🗹 A	utoscr	oll	S	how	tim	estamp		Newline	•	115200 baud	•	Clear	outp	ut

# 7.6 6. Comprehensive Projects:

The previous projects are related to single sensor or module. In the following part, we will combine various sensors and modules to create some comprehensive experiments to perform special functions.

# 7.6.1 Project 45: Button-controlled LED



#### Overview

In this lesson, we will make an extension experiment with a button and an LED. When the button is pressed and low levels are output, the LED will light up; when the button is released, the LED will go off. Then we can control a module with another module.

#### Components



# **Connection Diagram**



**Test Code** 



```
#define PIN_BUTTON 15
bool ledState = false;
void setup() {
 // initialize digital pin PIN_LED as an output.
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_BUTTON, INPUT);
}
// the loop function runs over and over again forever
void loop() {
 if (digitalRead(PIN_BUTTON) == LOW) {
   delay(20);
   if (digitalRead(PIN_BUTTON) == LOW) {
     reverseGPIO(PIN_LED);
   }
   while (digitalRead(PIN_BUTTON) == LOW);
 }
}
void reverseGPIO(int pin) {
 ledState = !ledState:
 digitalWrite(pin, ledState);
}
  ******
```

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. When the button is pressed, the LED will light up; when pressed again, the LED will go off.

# 7.6.2 Project 46: Alarm Experiment



#### Overview

In the previous experiment, we control an output module though an input module. In this lesson, we will make an experiment that the active buzzer will emit sounds once an obstacle appears.

# Components



# **Connection Diagram**



# **Test Code**

```
void setup() {
    pinMode(15, INPUT); //Obstacle avoidance sensor is connected to GPI015 and set to_
    input mode
    pinMode(4, OUTPUT); //The buzzer is connected to GPI04 and set to output mode
}
void loop() {
    item = digitalRead(15);//Read the level value output by the obstacle avoidance sensor
    if (item == 0) {//Obstruction detected
        digitalWrite(4, HIGH);//The buzzer sounded
    } else {//No obstacles detected
        digitalWrite(4, LOW);//The buzzer is off
    }
    delay(100);//Delay 100ms
}
```

# **Code Explanation**

Set IO ports according to connection diagram then configure pins mode

The value is 0 when pressing the button, So, we can determine the key value(0through if (item == 0) and make the buzzer beep digitalWrite(4, HIGH).

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. If the obstacle is detected, the active buzzer will chime; if not, it won't beep.

# 7.6.3 Project 47: Intrusion Detection



Description

In this experiment, we use a PIR motion sensor to control an active buzzer to emit sounds and the onboard LED to flash rapidly.

#### **Required Components**



...........

\* Filename : PIR alarm

\* Description : PIR control buzzer \* Auther : http//www.keyestudio.com

**Test Code** 

/\*

\*/

(continues on next page)

```
(continued from previous page)
```

```
int item = 0;
void setup() {
 pinMode(15, INPUT); //PIR motion sensor is connected to GPI015 and set as the input_
→mode
 pinMode(4, OUTPUT);//The active buzzer is connected to GPI04 and set to output mode
 pinMode(22, OUTPUT);//LED is connected to GPI022 and set to output mode
}
void loop() {
 item = digitalRead(15);//Read digital level signal output by infrared pyrorelease.
→sensor
 if (item == 1) { //Movement detected
   digitalWrite(4, HIGH); //Turn on the buzzer
   digitalWrite(22, HIGH); //Turn on the LED
   delay(200);//Delay 200ms
   digitalWrite(4, LOW); //Turn off the buzzer
   digitalWrite(22, LOW); //Turn off the LED
   delay(200);//Delay 200ms
 } else { //No detection
   digitalWrite(4, LOW); //Turn off the buzzer
   digitalWrite(22, LOW); //Turn off the LED
 }
}
```

### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. If the sensor detects people moving, the buzzer will emit an alarm , and the LED will flash continuously.

# 7.6.4 Project 48: Extinguishing Robot



#### Description

Today we will use Arduino simulation to build an extinguishing robot that will automatically sense the fire and start the fan.

In this project, we will learn how to build a very simple robot using ESP32, (detecting flames with a flame sensor, blowing out candles with a fan) can teach us basic concepts about robotics. Once you understand the basics below, you can build more complex robots.

#### **Components Required**



# **Connection Diagram**



### **Test Code**

```
/*
* Filename : Fire-fighting robot
* Description : Flame sensor controls the 130 fan module
* Auther
           : http//www.keyestudio.com
*/
int item = 0;
void setup() {
 Serial.begin(9600);
 pinMode(15, OUTPUT);//INA corresponds to IN+, and sets GPI015 to output mode
 pinMode(4, OUTPUT);//INB corresponds to IN-, and sets GPI04 to output mode
}
void loop() {
 item = analogRead(34);//The flame sensor is connected to GPI034, and read the
→simulated value to Item
 Serial.println(item);//Serial port display analog value
 if (item < 3000) {//Less than 3000
   digitalWrite(15, LOW);//Turn on the fan
   digitalWrite(4, HIGH);
 } else {//Otherwise, turn off the fan.
   digitalWrite(15, LOW);
   digitalWrite(4, LOW);
 }
 delay(100);
}
           ******
```

# **Code Explanation**

In the code, we set the threshold value to 3000. When the ADC value detected by the flame sensor is lower than the threshold value, the fan will be automatically turned on; otherwise, it will be turned off. For the driving method of the fan, please refer to the 130 Motor.

# **Test Result**

Connect the wires according to the experimental wiring diagram, switch the DIP switch on the ESP32 expansion board to the ON end and power up, compile and upload the code to the ESP32. After uploading successfully, open the serial monitor and set baud rate to 9600.

We need to press the reset button on the ESP32, then the ADC value of the flame will be printed. When this value is less than 3000, the fan will work to blow out the fire, otherwise, it will be turned off. Basically, the ADC value can be set by yourself.

/	dev/ttyUSB0		~	^	×
				Sen	nd
4095					•
1299					
289					
754					
705					
1379					
1008					
1498					
1216					
2195					
1578					
1846					
2218					
1797					
					Ŧ
<				1	
✓ Autoscroll	Newline	▼ 9600 baud ▼	Clear	outp	ut

# 7.6.5 Project 49: Rotary Encoder control RGB



# Introduction

In this lesson, we will control the LED on the RGB module to show different colors through a rotary encoder.

When designing the code, we need to divide the obtained values by 3 to get the remainders. The remainder is 0 and the LED will become red. The remainder is 1, the LED will become green. The remainder is 2, the LED will turn blue.

# Components

			Colary encoder
ESP32Board*1	ESP32 Expansion Board*1	KeyestudioCommon Cathode RGB Module*1	KeyestudioRotary En- coder Module*1
5P Dupont Wire*1	4P Dupont Wire*1	Micro USB Cable*1	

**Connection Diagram** 



Test Code

```
/*
* Filename : Encoder control RGB
* Description : Rotary encoder controls RGB to present different effects
* Auther : http//www.keyestudio.com
*/
//Interfacing Rotary Encoder with Arduino
//Encoder Switch -> pin 27
//Encoder DT -> pin 14
//Encoder CLK -> pin 12
int Encoder_DT = 14;
int Encoder_CLK = 12;
int Encoder_Switch = 27;
int Previous_Output;
int Encoder_Count;
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
const byte chns[] = \{0, 1, 2\};
                               //define the pwm channels
int red, green, blue;
int val;
void setup() {
 Serial.begin(9600);
 //pin Mode declaration
 pinMode (Encoder_DT, INPUT);
 pinMode (Encoder_CLK, INPUT);
 pinMode (Encoder_Switch, INPUT);
 Previous_Output = digitalRead(Encoder_DT); //Read the inital value of Output A
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit</pre>
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
  }
}
void loop() {
 //aVal = digitalRead(pinA);
 if (digitalRead(Encoder_DT) != Previous_Output)
 {
   if (digitalRead(Encoder_CLK) != Previous_Output)
   {
     Encoder_Count ++;
     Serial.print(Encoder_Count);
     Serial.print(" ");
     val = Encoder_Count % 3;
     Serial.println(val);
   }
   else
   {
     Encoder_Count--;
```

(continues on next page)

```
Serial.print(Encoder_Count);
     Serial.print(" ");
     val = Encoder_Count % 3;
     Serial.println(val);
   }
 }
 Previous_Output = digitalRead(Encoder_DT);
 if (digitalRead(Encoder_Switch) == 0)
 ł
   delay(5);
   if (digitalRead(Encoder_Switch) == 0) {
     Serial.println("Switch pressed");
     while (digitalRead(Encoder_Switch) == 0);
   }
 }
 if (val == 0) {
   //RED(255, 0, 0)
   ledcWrite(chns[0], 255 );
   ledcWrite(chns[1], ◊);
   ledcWrite(chns[2], 0);
 } else if (val == 1) {
   //GREEN(0, 255, 0)
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 255);
   ledcWrite(chns[2], 0);
 } else {
   //BLUE(0, 0, 255)
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 255);
 }
}
```

# **Code Explanation**

1). In the experiment, we set the val to the remainder of Encoder\_Count divided by 3. Encoder\_Count is the value of the encoder. Then we can set pin GPIO0 (red), GPIO2 (green) and GPIO15 (blue) according to remainders.

2). Referring to the control method learned in the previous experiment, use the LED on the remainder control module to display the corresponding light color. The value obtained by taking the remainder of 3 for any number is 0 or 1 or 2. We use these three values to judge, and display the corresponding color.

# **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then rotate the knob of the rotary encoder to display the reminders, which can control colors of LED(red green blue).

# 7.6.6 Project 50: Rotary Potentiometer



#### Introduction

In the previous courses, we did experiments of breathing light and controlling LED with button. In this course, we do these two experiments by controlling the brightness of LED through an adjustable potentiometer. The brightness of LED is controlled by PWM values, and the range of analog values is 0 to 4095 and the PWM value range is 0-255.

After the code is set successfully, we can control the brightness of the LED on the module by rotating the potentiometer.

#### **Required Components**



#### **Connection Diagram**



# **Test Code**

```
/*
* Filename
           : adjust the light
* Description : Controlling the brightness of LED by potentiometer.
           : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 34 //the pin of the potentiometer
#define PIN_LED
            15 // the pin of the LED
#define CHAN
                  0
void setup() {
 ledcSetup(CHAN, 1000, 12);
 ledcAttachPin(PIN_LED, CHAN);
}
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN); //read adc
 int pwmVal = adcVal; // adcVal re-map to pwmVal
 ledcWrite(CHAN, pwmVal);
                    // set the pulse width.
 delay(10);
}
```

#### **Code Explanation**

In the experiment, the mapping function maps adcVal from the range of 0-4095 to 0-255, and assigns it to pwmVal.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Rotating the potentiometer on the module can adjust the brightness of the LED on the LED module.

# 7.6.7 Project 51: Smart Windows



### Description

In life, we can see all kinds of smart products, such as smart home. Smart homes include smart curtains, smart windows, smart TVs, smart lights, and more. In this experiment, we use a steam sensor to detect rainwater, and then achieve the effect of closing and opening the window by a servo.

# **Required Components**



**Connection Diagram** 



# **Test Code**

```
//*********
/*
* Filename : smart window
* Description : Water drop sensor controls steering gear rotation.
* Auther : http//www.keyestudio.com
*/
#include <ESP32Servo.h>//Import the steering gear library file
int adcVal = 0;//A variable that holds the ADC value output by the droplet sensor
int servoPin = 15; // Define the servo pin
Servo myservo;//Defines an instance of the steering gear class
#define PIN_ADC 34 //the pin of the Water drop sensor
void setup(){
 Serial.begin(9600);
 pinMode(PIN_ADC, INPUT);
 myservo.setPeriodHertz(50);
                              // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→ object
}
void loop(){
 adcVal = analogRead(PIN_ADC);//The droplet sensor is connected to the analog port GP34
 Serial.println(adcVal);
 if (adcVal > 2000) {//The simulated value is greater than 2000
   myservo.write(0);//close the window
   delay(500);//Give the steering gear time to turn
 } else {// no rain
```

(continues on next page)



# **Code Explanation**

We can control a servo to rotate by a threshold.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. When the sensor detects a certain amount of water, the servo rotates to achieve the effect of closing or opening windows.

# 7.6.8 Project 52: Sound Activated Light



Introduction

In this lesson, we will make a smart sound activated light using a sound sensor and an LED module. When we make a sound, the light will automatically turn on; when there is no sound, the lights will automatically turn off. How it works? Because the sound-controlled light is equipped with a sound sensor, and this sensor converts the intensity of external sound into a corresponding value. Then set a threshold, when the threshold is exceeded, the light will turn on, and when it is not exceeded, the light will go out.

#### Components



**Connection Diagram** 





```
/*
* Filename : sound-controlled lights
* Description : Sound sensor controls LED on and off
          : http//www.keyestudio.com
* Auther
*/
int ledPin = 15;//LED is connected to GP15
int microPin = 34;//Sound sensor is connected to GPI034
void setup() {
 Serial.begin(9600);//Set baud rate to 9600
 pinMode(ledPin, OUTPUT);//LED is the output mode
}
void loop() {
 int val = analogRead(microPin);//Read analog value
 Serial.print(val);// Serial port print
 if(val > 600){//exceed the threshold value
   digitalWrite(ledPin, HIGH);//Lighting LED 3sand print the corresponding information
   Serial.println(" led on");
   delay(3000);
 }else{//otherwise
   digitalWrite(ledPin, LOW);//Turn off the LED and print the corresponding information
   Serial.println(" led off");
 }
 delay(100);
}
```

# **Code Explanation**

We set the ADC threshold value to 600. If more than 600, LED will be on 3s; on the contrary, it will be off.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600.

We need to press the reset button on the ESP32, then the corresponding volume ADC value will be displayed. When the analog value of sound is greater than 600, the LED on the LED module will light up 3s, otherwise it will go off.

	/dev/ttyUSB0		~ ^ X
			Send
0 led off 0 led off 0 led off 0 led off 0 led off 438 led off 833 led on 283 led off 330 led off 585 led off 0 led off 0 led off 811 led on			•
<ul> <li>Autoscroll Show timestamp</li> </ul>	Newline	▼ 9600 baud ▼	Clear output

# 7.6.9 Project 53: Fire Alarm



# Description

In this experiment, we will make a fire alarm system. Just use a flame sensor to control an active buzzer to emit sounds.

# **Required Components**



# **Connection Diagram**



# **Test Code**

```
Serial.begin(9600);
pinMode(4, INPUT);//Flame sensor digital pin is connected to GPI04
pinMode(15, OUTPUT);//Buzzer pin is connected to GPI015
}
void loop() {
    item = digitalRead(4);//Read the digital level output by the flame sensor
    Serial.println(item);//Newline print level signal
    if (item == 0) {//Flame detected
        digitalWrite(15, HIGH);//Turn on the buzzer
    } else {//Otherwise, turn off the buzzer
        digitalWrite(15, LOW);
    }
    delay(100);//Delay 100ms
}
```

# **Code Explanation**

This flame sensor uses an analog pin and a digital pin. When a flame is detected, the digital pin outputs a low level. In this experiment we will use the digital port.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. When the sensor detects the flame, the external active buzzer will emit sounds, otherwise the active buzzer will not emit sounds.

# 7.6.10 Project 54: Smoke Alarm



# Description

In this experiment, we will make a smoke alarm by a TM16504-Digit segment module, a gas sensor and an active buzzer.

# **Required Components**



### **Connection Diagram**



#### **Test Code**

```
/*
* Filename : smoke alarm
* Description : MQ2 controls a buzzer and a four-digit analog smoke tester
* Auther : http//www.keyestudio.com
*/
#include "TM1650.h" //Import the TM1650 library file
int adcVal = 0; //display ADC value
//the interfaces are GPI021 and GPI022
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
void setup() {
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.display0n0FF(); //display on or off, 0=display off, 1=display on,
→default : 1
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                            //DigitalTube.clearBit(0 to 3); Clear bit display.
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>→--9</u>
 pinMode(15, OUTPUT);//the buzzer is connected to GPI015
}
void loop() {
```

(continues on next page)

```
adcVal = analogRead(34);//Read the ADC values of MQ2
 displayFloatNum(adcVal);;//Four digit tube display adcVal values
 if (adcVal > 1000) {//ADC value is greater than 1000
   digitalWrite(15, HIGH); // buzzer alarming
 } else {//or else
   digitalWrite(15, LOW); //Turn off the buzzer
 delay(100);//delay 100ms
}
void displayFloatNum(float adcVal){
 if(adcVal > 9999)
   return:
 int dat = adcVal*10;
  //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return:
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return:
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
```

#### **Code Explanation**

Define an integer variable val to store the analog value of thesmoke sensor, and then we display the analog value in the four-digit digital tube, and then set a threshold, and when the threshold is reached, the buzzer will sound.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. When the concentration of combustible gas exceeds the standard, the active buzzer module will give an alarm, and the four-digit digital tube will display the concentration value.

# 7.6.11 Project 55: Alcohol Sensor



# Description

In the last experiment, we made a smoke alarm. In this experiment, we combine the active buzzer, the MQ-3 alcohol sensor, and a four-digit digital tube to test the alcohol concentration through the alcohol sensor. Then, the concentration to control the active buzzer alarm and the four-digit digital tube to display the concentration. So as to achieve the simulation effect of alcohol detector.

#### **Components Required**



**Connection Diagram** 



# **Test Code**

```
/*
* Filename : breathalyzer
* Description : MQ3 controls a buzzer and a four-digit tube to simulate a breathalyzer.
* Auther
           : http//www.keyestudio.com
*/
#include "TM1650.h" //Import the TM1650 library file
int adcVal = 0; //display ADC value
//the interfaces are GPI021 and GPI022
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
void setup() {
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, O=display off, 1=display on,
→default : 1
 for(char b=1;b<5;b++){
                          //DigitalTube.clearBit(0 to 3); Clear bit display.
   DigitalTube.clearBit(b);
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
\rightarrow --9
 pinMode(15, OUTPUT);//the buzzer is connected to GPI015
}
void loop() {
 adcVal = analogRead(34);//Read the ADC values of MQ3
 displayFloatNum(adcVal);//Four digit tube display adcVal values
```

(continues on next page)

```
if (adcVal > 1000) {//ADC value is greater than 1000
   digitalWrite(15, HIGH); // buzzer alarming
 } else {//or else
   digitalWrite(15, LOW); //Turn off the buzzer
 }
 delay(100);//delay 100ms
}
void displayFloatNum(float adcVal){
 if(adcVal > 9999)
   return:
 int dat = adcVal*10;
   //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
 if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
 }
 if(dat%1000/100 != 0){
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.displayBit(3, dat%1000/100);
 DigitalTube.displayBit(4, dat%100/10);
 return:
}
 DigitalTube.clearBit(1);
 DigitalTube.clearBit(2);
 DigitalTube.clearBit(3);
 DigitalTube.displayBit(4, dat%100/10);
}
  ******
```

# **Code Explanation**

Define an integer variable val to store the ADC value of the alcohol sensor, then we display the analog value in the four-digit display module and set a threshold.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. When different alcohol concentrations are detected, the active buzzer module will alarm, and the four-digit digital display will show the concentration value.
# 7.6.12 Project 56: Ultrasonic Radar



Description



We know that bats use echoes to determine the direction and the location of their preys. In real life, sonar is used to detect sounds in the water. Since the attenuation rate of electromagnetic waves in water is very high, it cannot be used to detect signals, however, the attenuation rate of sound waves in the water is much smaller, so sound waves are most commonly used underwater for observation and measurement.

In this experiment, we will use a speaker module, an RGB module and a 4-digit tube display to make a device for detection through ultrasonic.

## **Required Components**





Test Code

```
/*
* Filename : Ultrasonic radar
* Description : Ultrasonic control four digit tube, buzzer and RGB analog ultrasonic.
\rightarrow radar.
* Auther
             : http//www.keyestudio.com
*/
#include "TM1650.h" //Import the TM1650 library file
//the interfaces are GPI021 and GPI022
#define DIO 21
#define CLK 22
TM1650 DigitalTube(CLK,DIO);
int beeppin = 18; //Define the horn pin as GPI018
int TrigPin = 13; //Set the Trig pin to GPI013
int EchoPin = 14; //Set the Echo pin to GPI014
int distance;//Distance measured by ultrasound
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
const byte chns[] = \{0, 1, 2\};
                                //define the pwm channels
float checkdistance() { //get distance
 // A short low level is given beforehand to ensure a clean high pulse:
 digitalWrite(TrigPin, LOW);
 delayMicroseconds(2);
 // The sensor is triggered by a high pulse of 10 microseconds or more
 digitalWrite(TrigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin, LOW);
 // Read the signal from the sensor: a high level pulse
 //Its duration is the time (in microseconds) from sending the ping command to.
\rightarrow receiving the echo from the object
 float distance = pulseIn(EchoPin, HIGH) / 58.00; //Convert to distance
 delay(10);
 return distance;
}
void setup() {
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, O=display off, 1=display on,
\rightarrow default : 1
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                                //DigitalTube.clearBit(0 to 3); Clear bit display.
 }
 // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0); //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>→</u> - - 9
 pinMode(TrigPin, OUTPUT);//Sets the Trig pin as output
 pinMode(EchoPin, INPUT); //Set the Echo pin as input
 ledcSetup(3, 1000, 8);//setup the pwm channels,1KHz,8bit
 ledcAttachPin(18, 3);
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit
                                                                         (continues on next page)
```

```
ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
  distance = checkdistance(); //Ultrasonic ranging
  displayFloatNum(distance); //Nixie tube shows distance
  if (distance \leq 10) {
   ledcWrite(3, 100);
   delay(100);
   ledcWrite(3, ◊);
   ledcWrite(chns[0], 255); //Common cathode LED, high level to turn on the led.
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 0);
  } else if (distance > 10 && distance <= 20) {</pre>
   ledcWrite(3, 200);
   delay(200);
   ledcWrite(3, 150);
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 255);
   ledcWrite(chns[2], 0);
  } else {
   ledcWrite(3, ∅);
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 255);
  }
}
void displayFloatNum(float distance){
  if(distance > 9999)
   return:
  int dat = distance*10;
   //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
  if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
  }
  if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
  }
  if(dat%1000/100 != 0){
  DigitalTube.clearBit(1);
```

## **Code Explanation**

We set sound frequency and light color by adjusting different distance range.

We can adjust the distance range in the code.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. When the ultrasonic sensor detects different distances, the buzzer will produce different frequencies of sound (within 20 cm), the RGB will show different colors, and the measured distances are displayed on the 4-digit tube display.

# 7.6.13 Project 57: IR Remote Control



## Introduction

In the previous experiments, we learned how to turn on/off the LED and adjust its brightness via PWM and print the button value of the IR remote control in the serial monitor window. Herein, we use an infrared remote control to turn on/off an LED.

## Components



## **Connection Diagram**



## **Test Code**

//\*
//\*
\* Filename : IR Control LED
\* Description : Remote controls LED on and off
\* Auther : http//www.keyestudio.com
\*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
(continues on next page)

```
#include <IRutils.h>
const uint16_t recvPin = 15; // Infrared receiving pin 15
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results; // Create a decoding results class object
int led = 4;//LED connect to GP4
void setup() {
 Serial.begin(9600);
 irrecv.enableIRIn();
                                   // Start the receiver
 pinMode(led, OUTPUT);
}
void loop() {
 if(irrecv.decode(&results)) { // Waiting for decoding
   serialPrintUint64(results.value, HEX);// Print out the decoded results
   Serial.print("");
   handleControl(results.value); // Handle the commands from remote control
   irrecv.resume();
                                   // Receive the next value
 }
}
 void handleControl(unsigned long value) {
   if (value == 0xFF6897) // Receive the number '1'
    {
      digitalWrite(led, HIGH);//turn on LED
      Serial.println(" led on");
    }
   else if (value == 0xFF9867) // Receive the number '2'
    {
       digitalWrite(led, LOW);//turn off LED
      Serial.println(" led off");
   }
}
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600. We need to press the reset button on the ESP32, then press the button 1 of the remote, which will be displayed on the monitor, and the LED will be on. Similarly, press the button 2, the LED will be off.

	/dev/ttyUSB0	~ ^ X
		Send
FF6897 led on FF9867 led off		Î
Autoscroll 🗌 Show timestamp	Newline 🔹 9600 baud	▼ Clear output



## Description

We will use a temperature sensor and some modules to make a smart cooling device in this experiment. When the ambient temperature is higher than a certain value, the motor is turned on, thereby reducing the ambient temperature and achieving the heat dissipation effect. Then display the temperature value in the four-digit segment display.

## **Required Components**

	En-22 kayesta				4-Digital Tube	3.8	
ESP32 Board*1	ESP32 Board*1	Expansion	keyestudio 1 tor*1	130 Mo-	Keyestudio Segment Dis	TM1650 play*1	4-Digit
Lilli S S G 18820 Temperature							
Keyestudio 18B20 Temper- ature Sensor*1	3P Dupont	Wire*1	4P Dupont W	/ire*2	Micro USB (	Cable*1	
I ANN OF AN AND AND AND AND AND AND AND AND AND	August 2010 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	+					
Battery Holder*1	Battery (j vourself)*(	provide for					

**Connection Diagram** 



#### **Test Code**

```
//ds18b20 pin to 13
DS18B20 ds18b20(13);
void setup() {
  Serial.begin(9600);
 DigitalTube.setBrightness(); //set brightness, 0---7, default : 2
 DigitalTube.displayOnOFF(); //display on or off, 0=display off, 1=display on,
\rightarrow default : 1
  for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                                  //DigitalTube.clearBit(0 to 3); Clear bit display.
  }
  // DigitalTube.displayDot(1,true); //Bit0 display dot. Use before displayBit().
 DigitalTube.displayBit(1,0);
                                //DigitalTube.Display(bit,number); bit=0---3 number=0-
<u>⊶--9</u>
  //Motor is connected to 15 4
 pinMode(15, OUTPUT);
 pinMode(4, OUTPUT);
}
void loop() {
  double temp = ds18b20.GetTemp();//Read the temperature
  temp *= 0.0625;//The conversion accuracy is 0.0625/LSB
  Serial.println(temp);
  displayFloatNum(temp);//4- digit tube display temperature value
  if (temp > 25) {//When the temperature exceeds 25 degrees Celsius, turn on the fan
    digitalWrite(15, LOW);
   digitalWrite(4, HIGH);
  } else {//Otherwise, turn off the fan.
    digitalWrite(15, LOW);
   digitalWrite(4, LOW);
  }
 delay(100);
}
void displayFloatNum(float temp){
  if(temp > 9999)
   return;
  int dat = temp*10;
   //DigitalTube.displayDot(2,true); //Bit0 display dot. Use before displayBit().
  if(dat/10000 != 0){
   DigitalTube.displayBit(1, dat%100000/10000);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
  }
  if(dat%10000/1000 != 0){
   DigitalTube.clearBit(1);
   DigitalTube.displayBit(2, dat%10000/1000);
   DigitalTube.displayBit(3, dat%1000/100);
   DigitalTube.displayBit(4, dat%100/10);
   return;
  }
```

## **Code Explanation**

The setting of variables and the storage of detection values are the same as what we learned earlier. We also set a temperature threshold and control the rotation of the motor when the threshold is exceeded, and then we use the digital tube to display the temperature value.

## **Test Result**

Connect the wires according to the experimental wiring diagram and power on. Switch the DIP switch on the ESP32 expansion board to the ON end, compile and upload the code to the ESP32. After uploading successfully, we can see the temperature of the current environment (unit is Celsius) on the four-digit segment display, as shown in the figure below. If this value exceeds the value we set, the fan will rotate to dissipate heat.



# 7.6.15 Project 59: Intelligent Entrance Guard System

## Description

In this project, we use the RFID522 card swiping module and the servo to set up an intelligent access control system. The principle is very simple. We use RFID522 swipe card module, an IC card or key card to unlock.

## **Required Components**



## **Connection Diagram**



## **Test Code**

Note: Different RFID-MFRC522 IC cards and keys have diverse values. You can substitute your own IC cards and keys values for the corresponding values read by the RFID-MFRC522 module in the program, otherwise the servo can't be controlled when uploading the test code to the ESP32.

For example: You can replace the rfid\_str of the if (rfid\_str == "edf7945a" || rfid\_str == "4c96b6e")

in the program code with your own IC cards and keys values read by the RFID-MFRC522 module.

```
/*
* Filename : Intelligent_access_control
* Description : RFID controlled steering gear simulated door opening
* Auther : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
#include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int servoPin = 15; // Servo motor pin
String rfid_str = "";
void setup() {
 Serial.begin(9600);
 Wire.begin();
 mfrc522.PCD_Init();
                      // dispaly PCD - MFRC522 read carder
 ShowReaderDetails();
 Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
                                    // standard 50 hz servo
 myservo.setPeriodHertz(50);
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→ object
 myservo.write(0);
 delay(500);
}
void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return;
 }
 // select one of door cards. UID and SAK are mfrc522.uid.
 // save UID
 rfid_str = ""; //String emptying
 Serial.print(F("Card UID:"));
 for (byte i = 0; i < mfrc522.uid.size; i++) {</pre>
   rfid_str = rfid_str + String(mfrc522.uid.uidByte[i], HEX); //Convert to string
   //Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   //Serial.print(mfrc522.uid.uidByte[i], HEX);
 }
 Serial.println(rfid_str);
 if (rfid_str == "edf7945a" || rfid_str == "4c96b6e") {
```

```
myservo.write(180);
   delay(500);
   Serial.println(" open the door!");
   }
}
void ShowReaderDetails() {
 // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print(F("MFRC522 Software Version: 0x"));
 Serial.print(v, HEX);
 if (v == 0x91)
   Serial.print(F(" = v1.0"));
 else if (v == 0x92)
   Serial.print(F(" = v2.0"));
 else
   Serial.print(F(" (unknown)"));
 Serial.println("");
 // when returning to 0x00 or 0xFF, may fail to transmit communication signals
 if ((v == 0x00) || (v == 0xFF)) {
   Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?
'));
 }
}
```

## **Code Explanation**

In the previous experiment, our card swipe module has tested the information of IC card and key. Then we use this corresponding information to control the door.

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 9600.

We need to press the reset button on the ESP32, when we use the IC card or blue key to swipe the card, the monitor displays the card and the key information and "open the door", at the same time, the servo rotates to the corresponding angle to simulate opening the door.

/dev/ttyUSB0 ~				
		Se	end	
<pre>I%M/?∢`&lt;©?=a&gt;ii?\md?a?=?Y?)J???9?MFRC522 Software Version: 0x92 = v2.0 Scan PICC to see UID, type, and data blocks Card UID:4c96b6e open the door! Card UID:4c96b6e open the door! Card UID:edf7945a open the door! Card UID:edf7945a open the door!</pre>			Î	
✓ Autoscroll  Show timestamp    Newline    9600 baud	Clea	ar out	:put	

# 7.6.16 Project 60Bluetooth

This chapter mainly introduces how to use the bluetooth of ESP32 for simple data transmission with mobile phone. Project 60.1 is conventional bluetooth, and Project 60.2 is bluetooth control LED.

## **Project 60.1Classic Bluetooth**

## Components



In this experiment, we need to use a bluetooth dobbed serial bluetooth terminal for a study. If you haven't install it, please click the installation: https://www.appsapk.com/serial-bluetooth-terminal/.



Here is its sign:

#### **Component Knowledge**

Bluetooth is a short-distance communication system that can be divided into two types, namely low power bluetooth (BLE) and classic bluetooth. There are two modes for simple data transfer: master mode and slave mode.

**Master Mode**: In this mode, work is done on the master device and can be connected to the slave device. When the device initiates a connection request in the main mode, information such as the address and pairing password of other bluetooth devices are required. Once paired, you can connect directly to them.

**Slave Mode**: A bluetooth module in the slave mode can only accept connection requests from the host, but cannot initiate connection requests. After being connected to a host device, it can send and receive data through the host device . Bluetooth devices can interact with each other, when they interact, the bluetooth device in the main mode searches for nearby devices. While a connection is established, they can exchange data. For example, when a mobile phone exchanges data with ESP32, the mobile phone is usually in master mode and the ESP32 is in slave mode.



master mode

slave mode

## Wiring Diagram

We can use a USB cable to connect ESP32 mainboard to the USB port on the Raspberry Pi.



## **Test Code**

```
SerialBT.begin("ESP32test"); //Bluetooth device name
Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
void loop() {
    if (Serial.available()) {
        SerialBT.write(Serial.read());
    }
    if (SerialBT.available()) {
        Serial.write(SerialBT.read());
    }
    delay(20);
}
```

## **Test Result**

Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200. We need to press the reset button on the ESP32, when you see the serial prints the character, as shown below, it means that the ESP32's bluetooth is waiting for connection with a phone. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)



E (106) psram: PSRAM ID read error: 0xffffffff					
The device started, now you can pair it with blu	uetooth!				ļ
✓ Autoscroll Show timestamp	Newline	•	115200 baud	•	Clear output

Ensure that your mobile phone bluetooth is enabled and the bluetooth application of "Serial Bluetooth Terminal" is installed.



Click"Search"search for the nearby bluetooth and select to connect the"ESP32 test".

4611 CMCC	11:	:24	* 🛈 🌲 🖽	4G 93% 🎫
<	Bluet	tooth		
Bluetoo	oth			-•
Renam MyBlueto	e this device			>
Paired de	vices			
Ō	ESP32test			ןון
Available	devices			
*	4E:17:CF:AA:8	4:38		
With the E devices.	Bluetooth enable, this de	evice will	be visible to r	nearby

Open the software APP and click the left side of the terminal, select "Devices".



If you select ESP32test in classic bluetooth mode, a successful connection message will appear as shown below.



Data can be transferred between your phone and Raspberry Pi via ESP32 now.

Send "Hello", When the Raspberry Pi receives it, which will reply with "Hi!".

/dev/tty	USB0		~ ^ X
Hit			Send
configsip. 0, SPiwP.0x00 clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00 mode:DIO, clock div:1 load:0x3fff0018 lep:4	,hd_drv:0x00,wp_	_drv:0x00	
load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080400,len:6388 entry 0x400806b4 E (106) psram: PSRAM ID read error: 0xffffffff			
The device started, now you can pair it with bl Hello!	uetooth!		
<ul> <li>✓ Autoscroll Show timestamp</li> </ul>	Newline	▼ 115200 baud ▼	Clear output

46 CMCC		11	:26 🖇	'() 🛸 HD	4G 92% 🛄
≡	Termi	nal	4	())-	<b>i</b> :
11:25:41 11:25:41 11:26:11 11:26:24	.094 Conne .195 Conne .913 Hello! .759 Hi!	ecting to l ected	ESP32tes	st	
M1	M2	М3	M4	M5	M6

## Project 60.2Bluetooth Control LED

## Components



## Wiring Diagram



## **Test Code**

//***************
/*
* Filename : Bluetooth Control LED
* Description : The phone controls esp32's led via bluetooth.
When the phone sends "LED_on," ESP32's LED lights turn on.
When the phone sends "LED_off," ESP32's LED lights turn off.
* Auther : http//www.keyestudio.com
*/
<pre>#include "BluetoothSerial.h"</pre>
<pre>#include "string.h"</pre>
#define LED 15
BluetoothSerial SerialBT;
char buffer[20];
<pre>static int count = 0;</pre>
void setup() {
pinMode(LED, OUTPUT);
SerialBT.begin("ESP32test"); //Bluetooth device name
Serial.begin(115200);
Serial.println("\nlhe device started, now you can pair it with bluetooth!");
}
void loop() {
while(SerialBT available())

```
{
   buffer[count] = SerialBT.read();
   count++;
 }
 if(count>0){
   Serial.print(buffer);
   if(strncmp(buffer,"led_on",6)==0){
     digitalWrite(LED,HIGH);
   }
   if(strncmp(buffer, "led_off", 7)==0){
     digitalWrite(LED,LOW);
   }
   count=0;
   memset(buffer,0,20);
 }
}
```

## **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. The APP operation is the same as the project 60.1. We need to press the reset button on the ESP32, if you want to make the external LED on and off, simply change the sending content to "LED\_on" and "LED\_Off". Moving the APP to send data:



The serial monitor will display as follows:

	/dev/ttyUSB0	~ ^ X
		Send
abcd led on		•
led_off led_on		I
led_off		
✓ Autoscroll	Newline 🔻	115200 baud 🔻 Clear output

## LED Circumstance





Note: If the sent content is not "led\_on 'or" led\_off ", the status of the LED will not change. If the LED is on, it remains on when irrelevant content is received; Conversely, if the LED is off, it continues to be off when irrelevant content is received.

# 7.6.17 Project 61WIFI Station Mode

## Description

ESP32 has three different WiFi modes: Station mode, AP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi running mode before using, otherwise the WiFi cannot be used. In this project, we are going to learn the WiFi Station mode of the ESP32.

## Components



## Wiring Diagram

Plug the ESP32 to the USB port of your Raspberry Pi.



## **Component Knowledge**

## Station mode

When setting Station mode, the ESP32 is taken as a WiFi client. It can connect to the router network and communicate with other devices on the router via a WiFi connection. As shown in the figure below, the PC and the router have been connected. If the ESP32 wants to communicate with the PC, the PC and the router need to be connected.



## **Test Code**

Since WiFi names and passwords vary from place to place, thereby users need to enter the correct WiFi names and passwords in the box shown below before the program code runs.

WiFi_Station_Mode   Arduino 1.8.19 🛛 🗸 🔺 🗙
<u>File E</u> dit <u>Sketch</u> <u>T</u> ools <u>H</u> elp
WiFi_Station_Mode
<pre>//*  * Filename : WiFi Station  * Description : Connect to your router uEnterpthe correct Router  * Auther : http://www.keyestudio.com  */  */ </pre>
<pre>#include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h></pre>
<pre>//Enter correct router name and password. const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name const char *password_Router = "ChinaNet@233"; //Enter the router password</pre>
<pre>void setup(){    Serial.begin(115200);    delay(2000);    Serial.println("Setup start");    WiFi.begin(ssid_Router, password_Router);//Set ESP32 in Station mode and (</pre>
r Module. Default 4MB with spiffs (1.2MB APP/1.5MB SPIFES). OIO. 80MHz. 921600. None on /dev/ttyUSB0
//************************************
* Filename : WiFi Station * Description : Connect to your router using ESP22
* Auther : http://www.keyestudio.com
*/
#include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h>
<pre>//Enter correct router name and password. const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name const char *password_Router = "ChinaNet@233"; //Enter the router password</pre>
<pre>void setup(){    Serial.begin(115200);    delay(2000);    Serial.println("Setup start");    WiFi begin(csid Pouter _ password Pouter)://Set ESP32 in Station mode and connect it to</pre>

```
Serial.println(String("Connecting to ")+ssid_Router);
//Check whether ESP32 has connected to router successfully every 0.5s.
while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
}
Serial.println("\nConnected, IP address: ");
Serial.println(WiFi.localIP());//Serial monitor prints out the IP address assigned to_
    -ESP32.
    Serial.println("Setup End");
}
void loop() {
}
```

## **Test Result**

After entering the correct WiFi names and passwords, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200.

When the ESP32 successfully connects to ssid\_WiFi, the serial monitor prints out the IP address, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)



# 7.6.18 Project 62WIFI AP Mode

## Description

ESP32 has three different WiFi modes: Station mode, AP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi running mode before using, otherwise the WiFi cannot be used. In this project, we are going to learn the WiFi AP mode of the ESP32.

## Components



## Wiring Diagram

Plug the ESP32 mainboard to the USB port of your Raspberry Pi



## **Component Knowledge**

## **AP Mode:**

When setting AP mode, a hotspot network will be created, waiting for other WiFi devices to connect. As shown below;

Take the ESP32 as the hotspot, if a phone or PC needs to communicate with the ESP32, it must be connected to the ESP32's hotspot. Communication is only possible after a connection is established via the ESP32.



## **Test Code**

Before the program code runs, you can make any changes to the ESP32 AP name and password in the box as shown below, but in a default circumstance, it doesn't need to modify.

WiFi_AP_Mode   Arduino 1.8.19 🔹 🗸 🗙
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp
WiFi_AP_Mode
<pre>//* * Filename : WiFi AP * Description : Set ESP32 to open an access point * Auther : http://www.keyestudio.cop */ #include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h></pre>
<pre>const char *ssid_AP = "ESP32_Wifi"; //Enter the router name const char *password_AP = "12345678"; //Enter the router password</pre>
<pre>IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself void setup(){</pre>
<pre>Serial.begin(115200);</pre>
Invalid library found in /home/pi/Downloads/arduino-1.8.19/libraries/example Invalid library found in /home/pi/Arduino/libraries/TM1650: no headers files rIModule, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0
<pre>//***********************************</pre>
<pre>const char *ssid_AP = "ESP32_Wifi"; //Enter the router name const char *password_AP = "12345678"; //Enter the router password</pre>
IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself

IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself

```
void setup(){
   Serial.begin(115200);
   delay(2000);
```

```
Serial.println("Setting soft-AP configuration ... ");
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
 Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
 Serial.println("Setting soft-AP ... ");
 boolean result = WiFi.softAP(ssid_AP, password_AP);
 if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("Setup End");
}
void loop() {
}
         *****************
//*
```

## **Test Result**

Compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200, then monitor will display as follows:

(If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)



/dev	r/ttyUSB0		~ ^ X
			Send
LUAU. UX3111001C, LEII. 1210			<b>▲</b>
ho 0 tail 12 room 4			
load:0x40078000,len:10944			
load:0x40080400,len:6388			
entry 0x400806b4			
E (22) psram: PSRAM ID read error: Oxffffff	f		
Setting soft-AP configuration			
Ready			
Setting soft-AP			
Ready			
Soft-AP IP address = 192.168.1.108			
MAC address = 58:BF:25:8A:2F:BD			
Setup End			
			<b>.</b>
4			•
Autoscroll 🗌 Show timestamp	Newline	<ul> <li>▼ 115200 baud</li> </ul>	Clear output

When observing the printed information of the serial port monitor, turn on the WiFi scanning function of the mobile

phone, you can see the ssid\_AP on ESP32, which is dubbed "ESP32\_Wifi" in this program code. You can connect to it either by typing the password "12345678" or by modifying the program code to change its AP name and password.



## 7.6.19 Project 63WIFI AP+Station Mode

#### Description

In this project, we are going to learn the AP+Station mode of the ESP32.

#### Components



## Wiring Diagram

Plug the ESP32 mainboard to the USB port of your Raspberry Pi



#### **Component Knowledge**
## **AP+Station mode**

In addition to the AP mode and the Station mode, **AP+Station mode** can be used at the same time. Turn on the Station mode of the ESP32, connect it to the router network, and it can communicate with the Internet through the router. Then turn on the AP mode to create a hotspot network. Other WiFi devices can be connected to the router network or the hotspot network to communicate with the ESP32.

## Test Code

Before the program code runs, you need to modify the ssid\_Router, password\_Router, ssid\_AP and password\_AP, as shown in the box below:

```
📼 WiFi Station AP Mode | Arduino 1.8.16
                                                                           \times
File Edit Sketch Tools Help
                                                                                 Ø
  WiFi_Station_AP_Mode
                              ******
 11
 /*
                : WiFi AP+Station Please enter the correct names
  * Filename
  * Description : ESP32 connects to the user's router, turning on an access point
* Auther : http//www.keyestuandopasswords of Router and AP.
 */
 #include <WiFi.h>
 const char *ssid Router
                                "ChinaNet-2.4G-ODFO"; //Enter the router name
 const char *password Router
                                "ChinaNet@233"; //Enter the router password
 const char *ssid AP
                                "ESP32 Wifi"; //Enter the router name
 const char *password AP
                                "12345678"; //Enter the router password
 void setup(){
  Serial.begin(115200);
  Serial.println("Setting soft-AP configuration ... ");
  WiFi.disconnect();
  WiFi.mode(WIFI AP);
 comipl mintle/#Cotting

                                        ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3
/*
* Filename : WiFi AP+Station
* Description : ESP32 connects to the user's router, turning on an access point
* Auther
            : http//www.keyestudio.com
*/
#include <WiFi.h>
```

```
const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name
const char *password_Router = "ChinaNet@233"; //Enter the router password
const char *ssid_AP = "ESP32_Wifi"; //Enter the router name
                        = "12345678"; //Enter the router password
const char *password_AP
void setup(){
 Serial.begin(115200);
 Serial.println("Setting soft-AP configuration ... ");
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
 Serial.println("Setting soft-AP ... ");
 boolean result = WiFi.softAP(ssid_AP, password_AP);
 if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("\nSetting Station configuration ... ");
 WiFi.begin(ssid_Router, password_Router);
 Serial.println(String("Connecting to ")+ ssid_Router);
 while (WiFi.status() != WL_CONNECTED){
   delay(500);
   Serial.print(".");
 }
 Serial.println("\nConnected, IP address: ");
 Serial.println(WiFi.localIP());
 Serial.println("Setup End");
}
void loop() {
}
```

## **Test Result**

Ensure that the code in the program has been modified correctly, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set the baud rate to 115200, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)



	/dev/ttyl	USB0				~	~	×
							Sen	d
E (22) psram: PSRAM ID read error Setting soft-AP configuration Setting soft-AP Ready Soft-AP IP address = 192.168.4.1 MAC address = 58:BF:25:8A:2F:BD Setting Station configuration Connecting to ChinaNet-2.4G-0DF0  Connected, IP address: 192.168.0.167 Setup End	: 0xffffffff							
Autoscroll 🗌 Show timestamp		Newline	•	115200 baud	•	Clear	outpi	Jt

Open the WiFi scanning function of the mobile phone, you can see the ssid\_AP.

	Wi–Fi	
Settings	Wi-Fi View help	
Q Search	More settings	>
Log in to	AVAILABLE NETWORKS	
Access Cloud, > AppGallery, and	ESP32_Wifi Connected	ŝ.
more	ChinaNet-2.4G-0DF0	<b>a</b>
🛜 WLAN ESP32_Wifi >	ChinaNet-Dsvv	6
	Encrynted	- <b>T</b>

## 7.6.20 Project 64: Comprehensive Experiment



## Introduction

We did a lot of experiments, and for each one we needed to re-upload the code, so can we achieve different functions through an experiment? In this experiment, we will use an external button module to achieve different functions.

## **Components Required**



ESP32 Board\*1



ESP32 Expansion Board\*1



Keyestudio DIY Purple LED Module\*1



Keyestudio Button Module\*1



Keyestudio Potentiometer\*1



Keyestudio Obstacle Avoidance Sensor\*1



Keyestudio XHT11 Temperature and Humidity Sensor \*1



Keyestudio ADXL345 Acceleration Sensor\*1



Keyestudio Line Tracking Sensor\*1





Keyestudio DIY Joystick Module\*1



3P Dupont Wire\*6



Keyestudio HC-SR04 Ultrasonic sensor \*1







RGB LED Keyestudio DIY Common Cathode RGB Module \*1



5P Dupont Wire\*1

## Wiring Diagram



**Test Code** 

```
/*
* Filename : Comprehensive experiment
* Description : Multiple sensors/modules work together
* Auther
          : http//www.keyestudio.com
*/
#include "xht11.h"
#include "adx1345_io.h"
//ADXL345 sda-->21,scl-->22
adxl345 adxl345(21, 22);
//xht11 to gpio15
xht11 xht(15);
//rgb is connected to 4,0,2
int ledPins[] = {4, 0, 2}; //define red, green, blue led pins
                                                              (continues on next page)
```

```
(continued from previous page)
const byte chns[] = \{0, 1, 2\};
                                     //define the pwm channels
int red, green, blue;
//Rocker module port
int X = 35;
int Y = 34;
int KEY = 32;
//Potentiometer pin is connected to analog port 33
int resPin = 33;
//Trace sensor pin connected to IO port 14
int TrackingPin = 14;
//LED is Connected to GP5
#define PIN_LED 5 // the pin of the LED
#define CHAN
               3
//Obstacle avoidance sensor is connected to GP27
int Avoid = 27;
//Ultrasonic sensor port
int Trig = 13;
int Echo = 12;
//Key module port
int button = 23;
int PushCounter = 0;//Store the number of times a key is pressed
int yushu = 0;
unsigned char dht[4] = {0, 0, 0}; //Only the first 32 bits of data are received, not.
\leftrightarrow the parity bits
bool ir_flag = 1;
float out_X, out_Y, out_Z;
void counter() {
 delay(10);
 ir_flag = 0;
 if (!digitalRead(button)) {
    PushCounter++;
 }
}
void setup() {
  Serial.begin(9600);//Set baud rate to 9600
  pinMode(KEY, INPUT);//Button of remote sensing module
  ledcSetup(CHAN, 1000, 12);
  ledcAttachPin(PIN_LED, CHAN);
  pinMode(button, INPUT);//The key module
  attachInterrupt(digitalPinToInterrupt(button), counter, FALLING); //External_
→interrupt 0, falling edge fired
  pinMode(Avoid, INPUT);//Obstacle avoidance sensor
```

```
pinMode(Trig, OUTPUT);//Ultrasonic module
  pinMode(Echo, INPUT);
  adxl345.Init();
  for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 delay(1000);
}
}
void loop() {
  yushu = PushCounter % 8;
  if (yushu == 0) { //The remainder is 0
   yushu_0(); //rgb displays
  } else if (yushu == 1) { //The remainder is 1
   yushu_1(); //Displays the high and low levels read by the tracking sensor
  } else if (yushu == 2) { //The remainder is 2
   yushu_2(); //Display temperature and humidity value
  } else if (yushu == 3) { //The remainder is 3
   yushu_3(); //Displays the rocker value
  }else if (yushu == 4) { //The remainder is 4
   yushu_4(); //Display potentiometer ADC value and potentiometer control LED
  } else if (yushu == 5) { //The remainder is 5
   yushu_5(); //Obstacle avoidance sensor detects obstacles
  } else if (yushu == 6) { //The remainder is 6
   yushu_6(); //Shows the distance detected by ultrasound
  } else if (yushu == 7) { //The remainder is 7
   yushu_7(); //ADXL345 triaxial acceleration value
  }
}
//RGB
void yushu_0() {
 red = random(0, 256);
  green = random((0, 256);
 blue = random(\emptyset, 256);
  setColor(red, green, blue);
 delay(200);
}
void setColor(byte r, byte g, byte b) {
 ledcWrite(chns[0], 255 - r); //Common anode LED, low level to turn on the led.
 ledcWrite(chns[1], 255 - g);
  ledcWrite(chns[2], 255 - b);
}
void yushu_1() {
 int val = digitalRead(TrackingPin);//Read the digital level output by the tracking_
→sensor
  Serial.print(val);//Serial port print value
  if (val == 0) {//White val is 0 detected
                          ");
    Serial.print("
    Serial.println("White");
```

```
delay(100);
  }
  else {//Black val is 1 detected
   Serial.print("
                          ");
   Serial.println("Black");
   delay(100);
 }
}
void yushu_2() {
 if (xht.receive(dht)) { //Returns true when checked correctly
    Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [1] is the fractional part
    Serial.print("% ");
    Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
→part
   Serial.println("C");
  } else { //read error
   Serial.println("sensor error");
  }
 delay(1200);
}
void yushu_3() {
 int x = analogRead(X);
  int y = analogRead(Y);
  int key = digitalRead(KEY);
  Serial.print("X:");
  Serial.print(x);
  Serial.print("
                   Y:");
  Serial.print(y);
  Serial.print("
                   KEY:");
  Serial.println(key);
  delay(100);
}
void yushu_4() {
 int adcVal = analogRead(resPin); //read adc
  Serial.println(adcVal);
                            // adcVal re-map to pwmVal
  int pwmVal = adcVal;
 ledcWrite(CHAN, pwmVal); // set the pulse width.
  delay(10);
}
void yushu_5() {
 int val = digitalRead(Avoid);
  if (val == 0) {//Obstruction detected
   Serial.println("There are obstacles");
  }
  else {//No obstructions detected
   Serial.println("All going well");
```

```
}
 delay(100);
}
void yushu_6() {
 float distance = checkdistance();
 Serial.print("distance:");
 Serial.print(distance);
 Serial.println("cm");
 delay(100);
}
void yushu_7() {
 adx1345.readXYZ(&out_X, &out_Y, &out_Z);
 Serial.print(out_X);
                  ");
 Serial.print("g
 Serial.print(out_Y);
 Serial.print("g
                  ");
 Serial.print(out_Z);
 Serial.println("g");
 delay(100);
}
float checkdistance() {
 digitalWrite(Trig, LOW);
 delayMicroseconds(2);
 digitalWrite(Trig, HIGH);
 delayMicroseconds(10);
 digitalWrite(Trig, LOW);
 float distance = pulseIn(Echo, HIGH) / 58.00;
 delay(10);
 return distance;
}
```

#### **Code Explanation**

1). Calculate how many times the button is pressed, divide it by 8, and get the remainder which is 0, 1 2, 3, 4, 5, 6 and 7. According to different remainders, construct eight unique functions to control the experiment and realize different functions.

2). Following the instructions, we can add or remove sensors/modules in the wiring, and then change the experimental function in the code.

#### **Test Result**

Connect the wires according to the experimental wiring diagram, compile and upload the code to the ESP32. After uploading successfully, we will use a USB cable to power on. At the beginning, the number of the button is 0 and remainder is 0. Open the monitor and set baud rate to 9600.

Press the button, the RGB stops flashing, press once, the remainder is 1. The function of the experiment is to detect black objects and white objects by a line tracking sensor. If the sensor does not detect an object or detects a black object, val is 1, and the serial monitor displays the character "1 Black". When a white object (reflective) is detected, val is 0 and the serial monitor displays the character "0 White", the serial monitor will display as follows:

	/dev/ttyUSB1	~	^	×
			Ser	nd
1	Black			
0	White			
1	Black			
1	Black			
				Ŧ
•	Autoscroll 🗌 Show timestamp Newline 💌 9600 baud 💌 Clea	ar o	outp	ut

Press a key twice, the time of pressing buttons is 2 and the remainder is 2. Read temperature and humidity values. As shown below;

		/dev/ttyUSB1			× ^	×
					Se	end
RH:77%	Temp:29C					-
RH:79%	Temp:29C					
RH:84%	Temp:29C					
RH:87%	Temp:30C					
RH:88%	Temp:30C					
RH:89%	Temp:30C					
RH:91%	Temp:30C					
RH:92%	Temp:30C					
RH:93%	Temp:31C					
RH:93%	Temp:31C					
RH:93%	Temp:31C					
RH:93%	Temp:31C					
RH:93%	Temp:31C					
						-
🖌 Auto	scroll 🗌 Show timestamp	Newline	▼ 9600 bau	d 🔻 Cl	ear out	put

Press a key again, the time of pressing buttons is 3 and the remainder is 3. Read digital values at x, y and z axis of the joystick module. As shown below;

			/dev/ttyUSE	31		~	^	×
							Sen	d
X:2571	Y:4095	KEY:0						•
X:976	Y:4095	KEY:0						
X:0	Y:4095	KEY:0						
X:0	Y:1911	KEY:0						
X:0	Y:0 KE	Y:0						
X:1906	Y:0	KEY:0						
X:3091	Y:0	KEY:0						
X:4095	Y:1915	KEY:0						
X:4095	Y:3495	KEY:0						
X:4095	Y:4095	KEY:0						
X:3483	Y:1909	KEY:0						
X:3286	Y:1914	KEY:1						11
X:4095	Y:4095	KEY:1						
								÷
🖌 Auto	oscroll 🗌 Sh	ow timestamp	N	ewline 🔹	9600 baud 🔻	Clear	outp	ut

Press the key for the fourth time, the remainder is 4. Then the potentiometer can adjust the PWM value at the GPI05 port to control LED brightness of the purple LED.

/dev/ttyU	SB1				×
				Sen	d
1516					•
1573					
1632					
1683					
1744					
1802					
1869					
1919					
1987					
2064					
2141					
2223					
2299					
23					Ŧ
✓ Autoscroll	Newline 🔻	9600 baud 🔹	Clear	outp	ut

Press the key for the fifth time, the remainder is 5. Then the ultrasonic sensor can detect obstacles, as shown below;

	/dev/ttyUSB1	~ ^ X
		Send
All going well All going well All going well All going well All going well There are obstacles There are obstacles There are obstacles There are obstacles		
There are obstacles There are obstacles There are obstacles There are obstacles		Į
Autoscroll Show timestamp	Newline • 9600 baud •	Clear output

Press the key for the sixth time, the remainder is 6. Then the ultrasonic sensor can detect distance away from obstacles, as shown below;

/dev/ttyl	USB1				×
				S	end
distance:7.26cm					•
distance:7.09cm					
distance:6.91cm					
distance:7.09cm					
distance:7.34cm					
distance:7.59cm					
distance:7.48cm					
distance:8.16cm					
distance:8.40cm					
distance:8.64cm					
distance:8.84cm					
distance:8.98cm					
distance:9.12cm					Ļ
✓ Autoscroll	Newline	•	9600 baud 🔻	Clear out	tput

Press the key for seventh time and the remainder is 7. The monitor will print out the acceleration values.

			/dev/ttyUSB0	~	^	×
					Sen	nd
39.00y	-203.009	ວ໐.⊎⊎y				
-77.00g	-254.00g	47.00g				
-142.00g	-193.00g	16.00g				
-170.00g	-90.00g	16.00g				
-201.00g	-50.00g	-13.00g				
-196.00g	16.00g	-4.00g				
-141.00g	111.00g	67.00g				
-130.00g	157.00g	288.00g				
-40.00g	-55.00g	178.00g				
15.00g	-321.00g	117.00g				
56.00g	-317.00g	233.00g				
60.00g	-179.00g	161.00g				1
-63.00g	-89.00g	160.00g				
	0	0				-
•						•
🖌 Autosc	roll 🗌 Show t	timestamp	Newline 💌 9600 baud 💌 Cl	ear o	outp	ut

Press the key for eighth time and the remainder is 0. Then the RGB will flash. If you press keys incessantly, remainders will change in a loop way. So does functions.

# 7.6.21 Project 65: WiFi

## Description

In the previous experiment, we have learned the WiFi Station mode, WiFi AP mode and WiFi AP+Station mode of the ESP32. In this project, We will use ESP32's WiFi Station mode to control the work of multiple sensors/modules through APP connection with WiFi to achieve the effect of WiFi smart home.

## Components



ESP32 Board\*1









ESP32 Expansion Board\*1

Keyestudio 130 Motor\*1

Keyestudio 5V Servo\*1 Relay Module\*1





Keyestudio XHT11 Temperature and Humidity Sensor\*1compatible DHT11)



Keyestudio HC-SR04 Ultrasonic Sensor\*1

3P Dupont\*2

4P Dupont\*2



Smart Phone/PC\*1





Battery (provide for yourself)\*6

Micro USB Cable\*1

Wiring Diagram

Battery Holder\*1



## Install APP

(1) Android device (mobile phone/PC) APP:

A. We provide the Android APP installation package.

↑	S.Libraries_Driver_Firmware_and_APP	ې <mark>ت</mark> 🗸	Search 5.Libraries
	Name	Date modified	Туре
:0	Android APP	11/10/2023 2:17 PM	File folder
۴.	Arduino_C_Libraries(Raspberry-Pi)	11/13/2023 8:42 AM	File folder
٢	Arduino_C_Libraries(Windows)	11/13/2023 8:42 AM	File folder
۴	CP2102 Driver File	11/10/2023 1:41 PM	File folder
e	Python_Firmware	11/10/2023 1:46 PM	File folder

B. Now transfer the keyes wifi.apk file in the Android APP installation package to the Android phone or PC, click the keyes wifi.apk file to enter the installation page, click "ALLOW" key, and then click "INSTALL" button. After installation, click "OPEN" button to enter the APP interface.

1	S.Libraries_Driver_Firmware_and_A	APP > Android APP ~	õ ,	Search Android APP
	Name	Date modified	Туре	Size
Þ	📋 keyes wifi.apk	8/30/2021 4:13 PM	APK File	2,362 KB
k	eves wifi.apk			
		<b>Evi</b> Fi keyes wifi		
	Allow Files to ins	stall apps?		
	Downloading apps fr put your device and p By touching ALLOW, these risks.	rom external sources personal data at gre you indicate that yo	s may ater risk. u accept	
	Don't ask me agair	n		
	DENY	ALLO	W	





Webpage not available	192.168.1.148	WIFI
	~22	
		$\bigcirc$
	2	3
	5	6

(2) IOS device (mobile phone /iPad) APP:



A. Open App Store App Store

B. Enter keyes link in the search box and click search, the download interface appears. Click " V" " to download and install the APP of the keyes link. The following operations are similar to those of Android system. You can refer to the steps of Android system above for operation.

#### **Test Code**

```
/*
* Filename
           : WiFi Smart Home.
* Description : WiFi APP controls Multiple sensors/modules work to achieve the effect.
\rightarrow of WiFi smart home.
* Auther
          : http//www.keyestudio.com
*/
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>
#include "xht11.h"
//gpio15
xht11 xht(27);
unsigned char dht[4] = {0, 0, 0, 0};
```

```
#include <ESP32Servo.h>
Servo myservo;
int servoPin = 21;
#define Relay 4
#define IN1 2 //IN1 corresponds to IN+
#define IN2 15 //IN2 corresponds to IN-
#define trigPin 12
#define echoPin 13
int distance1;
String dis_str;
int ip_flag = 1;
int ultra_state = 1;
int temp_state = 1;
int humidity_state = 1;
String item = "0";
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233"; //the password of user's wifi
WiFiServer server(80);
String unoData = "";
void setup() {
  Serial.begin(115200);
  pinMode(Relay, OUTPUT);
  myservo.setPeriodHertz(50);
  myservo.attach(servoPin, 500, 2500);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(Relay, LOW);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

```
void loop() {
 WiFiClient client = server.available();
 if (!client) {
     return;
 }
 while(client.connected() && !client.available()){
      delay(1);
 }
 String req = client.readStringUntil('\r');
 int addr_start = req.indexOf(' ');
 int addr_end = req.indexOf(' ', addr_start + 1);
 if (addr_start == -1 || addr_end == -1) {
      Serial.print("Invalid request: ");
      Serial.println(req);
     return;
 }
 req = req.substring(addr_start + 1, addr_end);
 item=req;
 Serial.println(item);
 String s;
 if (req == "/")
 {
      IPAddress ip = WiFi.localIP();
     String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +.

→String(ip[3]);

      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
→Hello from ESP32 at ";
     s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);
 }
 else if(req == "/btn/0")
 {
   Serial.write('a');
   client.println(F("turn on the relay"));
   digitalWrite(Relay, HIGH);
 }
 else if(req == "/btn/1")
 {
   Serial.write('b');
   client.println(F("turn off the relay"));
   digitalWrite(Relay, LOW);
 }
 else if(req == "/btn/2")
 {
   Serial.write('c');
   client.println("Bring the steering gear over 180 degrees");
   myservo.write(180);
   delay(200);
 }
 else if(req == "/btn/3")
```

{

```
(continued from previous page)
```

```
Serial.write('d');
 client.println("Bring the steering gear over 0 degrees");
 myservo.write(0);
 delay(200);
}
else if(req == "/btn/4")
{
 Serial.write('e');
 client.println("esp32 already turn on the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
}
else if(req == "/btn/5")
{
 Serial.write('f');
 client.println("esp32 already turn off the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
}
else if(req == "/btn/6")
{
 Serial.write('g');
 while(Serial.available() > 0)
  {
   unoData = Serial.readStringUntil('#');
    client.println("Data");
 }
 while(ultra_state>0)
     {
        Serial.print("Distance = ");
        Serial.print(checkdistance());
        Serial.println("#");
        Serial1.print("Distance = ");
        Serial1.print(checkdistance());
        Serial1.println("#");
        int t_val1 = checkdistance();
        client.print("Distance(cm) = ");
        client.println(t_val1);
        ultra_state = 0;
      }
}
else if(req == "/btn/7")
{
 Serial.write('h');
 client.println("turn off the ultrasonic");
 ultra_state = 1;
}
else if(req == "/btn/8")
ł
  Serial.write('i');
 while(Serial.available() > 0)
```

```
{
   unoData = Serial.readStringUntil('#');
    client.println(unoData);
   }
 while(temp_state>0)
    {
      if (xht.receive(dht)) {
        Serial.print("Temperature = ");
        Serial.print(dht[2],1);
        Serial.println("#");
        Serial1.print("Temperature = ");
        Serial1.print(dht[2],1);
        Serial1.println("#");
        int t_val2 = dht[2];
        client.print("Temperature(°) = ");
        client.println(t_val2);
      }
      temp_state = 0;
   }
}
else if(req == "/btn/9")
{
 Serial.write('j');
 client.println("turn off the temperature");
  temp_state = 1;
}
else if(req == "/btn/10")
{
  Serial.write('k');
 while(Serial.available() > 0)
   {
     unoData = Serial.readStringUntil('#');
     client.println(unoData);
   }
 while(humidity_state > 0)
    {
      if (xht.receive(dht)) {
        Serial.print("Humidity = ");
        Serial.print(dht[0],1);
        Serial.println("#");
        Serial1.print("Humidity = ");
        Serial1.print(dht[0],1);
        Serial1.println("#");
        int t_val3 = dht[0];
        client.print("Humidity(%) = ");
        client.println(t_val3);
      }
     humidity_state = 0;
   }
}
else if(req == "/btn/11")
{
```

/the password of user's wifi

```
Serial.write('1');
   client.println("turn off the humidity");
   humidity_state = 1;
   }
 //client.print(s);
 client.stop();
}
int checkdistance() {
 digitalWrite(12, LOW);
 delayMicroseconds(2);
 digitalWrite(12, HIGH);
 delayMicroseconds(10);
 digitalWrite(12, LOW);
 int distance = pulseIn(13, HIGH) / 58;
 delay(10);
 return distance;
}
          const char* ssid = "ChinaNet-2.4G-0DF0";
                                                       //the name of user's wifi
```

Note: You need to const char\* password = "ChinaNet@233";

change the Wifi name and default Wifi password of the experimental code to your own Wifi name and Wifi password.

#### **Test Result**

After the code has been modified correctly, connect the external power supply and power on. Switch the DIP switch ON the ESP32 expansion board to the ON end, compile and upload the code to the ESP32 mainboard. If uploading the

code is not successful, press the Boot button on the ESP32 mainboard with your hand after click, release it when the upload progress percentage appears.)



Open the serial monitor and set baud rate to 115200, then the monitor prints the detected WiFi IP address. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the button RESET of the ESP32)

	/dev/ttyUSB0				~ ^ X
					Send
 Connected to ChinaNet-2.4G-ODFO IP address: 192.168.0.156 TCP server started					
Autoscroll 🗌 Show timestamp	Newline	•	115200 baud	• Clea	ar output

Open WiFi APP, enter the detected WIFI IP address in the text box in front of the WIFI button (for example, the IP address detected by the serial monitor above is 192.168.0.156).

Next, click the WIFI button to connect to WIFI, at the same time, the corresponding WiFi IP address will be displayed in the text box :"Hello from ESP32 at 192.168.0.156", then the APP has connected to WiFi. (WiFi IP address sometimes changes, if the original IP address can not use, you need to re-check it.)



After the APP is connected to WiFi, the following operations are performed:





values, the ambient humidity is 52%click againturn off the sensor, the APP will display turn off the humidity