

---

# keyestudio WiKi

keyestudio WiKi

Nov 16, 2023





# KS3016 RASPBERRY PI SENSOR STARTER KIT

<b>1</b>	<b>1. Description:</b>	<b>1</b>
<b>2</b>	<b>2. Kit List:</b>	<b>3</b>
<b>3</b>	<b>3. Resources:</b>	<b>7</b>
<b>4</b>	<b>C Language Tutorial</b>	<b>9</b>
4.1	1. Install Raspberry Pi OS System . . . . .	9
4.2	2. Install Raspberry Pi OS on Raspberry Pi 4B . . . . .	26
4.3	3. Preparations for C Language . . . . .	41
4.3.1	Hardware . . . . .	41
4.3.2	GPIO Extension Board . . . . .	43
4.3.3	Install WiringPi GPIO Library . . . . .	46
4.3.4	Run Example Code1 . . . . .	49
4.4	4. Projects . . . . .	54
4.4.1	Project 1Hello World . . . . .	54
4.4.2	Project 2LED Blinks . . . . .	55
4.4.3	Project 3SOS Light . . . . .	57
4.4.4	Project 4Breathing LED . . . . .	59
4.4.5	Project 5Traffic Lights . . . . .	63
4.4.6	Project 6Illuminating Lamp . . . . .	65
4.4.7	Project 7RGB Light . . . . .	67
4.4.8	Project 8Doorbell . . . . .	69
4.4.9	Project 9Passive Buzzer . . . . .	71
4.4.10	Project 10Button-controlled LED . . . . .	76
4.4.11	Project 11PIR Motion Sensor . . . . .	79
4.4.12	Project 12Fire Alarm . . . . .	81
4.4.13	Project 13Electronic Hourglass . . . . .	83
4.4.14	Project 14Collision Alarm . . . . .	87
4.4.15	Project 15Line-tracking Sensor . . . . .	89
4.4.16	Project 16Photo Interrupter Module . . . . .	91
4.4.17	Project 17Magnetic Detection . . . . .	93
4.4.18	Project 185V Relay . . . . .	95
4.4.19	Project19Touch-sensitive Alarm . . . . .	97
4.4.20	Project 20Obstacle Avoidance Sensor . . . . .	99
4.4.21	Project 21Reed Switch Module . . . . .	102
4.4.22	Project 22Vibration Alarm . . . . .	104
4.4.23	Project 23Servo . . . . .	107
4.4.24	Project 24Adjust the Brightness of LED . . . . .	109
4.4.25	Project 25Photoresistor . . . . .	114

4.4.26	Project 26Sound-activated Light . . . . .	116
4.4.27	Project 27I2C LCD1602 . . . . .	118
4.4.28	Project 28Water Level Monitor . . . . .	123
4.4.29	Project 29Flower-watering Device . . . . .	125
4.4.30	Project 30Temperature Alarm . . . . .	127
4.4.31	Project 31Steam in the Air . . . . .	129
4.4.32	Project 32MQ-2 Gas Leakage Alarm . . . . .	131
4.4.33	Project 33Alcohol Tester . . . . .	133
4.4.34	Project 34Joystick Module . . . . .	136
4.4.35	Project 35Ultrasonic Sensor . . . . .	138
4.4.36	Project 36 Light Intensity Detection . . . . .	142
4.4.37	Project 37Pressure Measurement . . . . .	144
4.4.38	Project 38Temperature Detection . . . . .	146
4.4.39	Project 39: Ultraviolet Light Detection . . . . .	148
<b>5</b>	<b>Processing JAVA Tutorial . . . . .</b>	<b>151</b>
5.1	1.Preparations . . . . .	151
5.1.1	(1)Install processing IDE . . . . .	151
5.1.2	(2)Use Processing IDE . . . . .	157
5.1.3	(3)Copy Example Code to Raspberry Pi . . . . .	161
5.2	2.Projects . . . . .	163
5.2.1	Project 1Print Hello World . . . . .	163
5.2.2	Project 2LED Blinks . . . . .	165
5.2.3	Project 3Mouse-controlled LED . . . . .	169
5.2.4	Project 4Breathing LED . . . . .	170
5.2.5	Project 5RGB . . . . .	175
5.2.6	Project 6Active Buzzer . . . . .	180
5.2.7	Project 7Button-controlled LED . . . . .	184
5.2.8	Project 8PIR Motion Sensor . . . . .	187
5.2.9	Project 9Fire Alarm . . . . .	190
5.2.10	Project 10 Collision Alarm . . . . .	194
5.2.11	Project 11 Line-tracking Sensor . . . . .	198
5.2.12	Project 12 Magnetic Detection . . . . .	201
5.2.13	Project 14 Rotary Potentiometer . . . . .	208
5.2.14	Project 15 Photoresistor . . . . .	215
5.2.15	Project 16 Water Level Monitor . . . . .	218
5.2.16	Project 17 Flower-watering Device . . . . .	222
5.2.17	Project 18Joystick . . . . .	225
<b>6</b>	<b>Python Tutorial . . . . .</b>	<b>231</b>
6.1	1. Install Raspberry Pi OS System . . . . .	231
6.1.1	1.1Hardware Tool . . . . .	231
6.1.2	1.2Software Tool . . . . .	231
6.2	2.Install Raspberry Pi OS on Raspberry Pi 4B . . . . .	250
6.2.1	(1) Burn System . . . . .	253
6.2.2	(2)Log in system . . . . .	256
6.2.3	(3) Remote Login . . . . .	260
6.2.4	(4) Check ip and mac address . . . . .	263
6.2.5	(5) Fix ip address of Raspberry Pi . . . . .	265
6.2.6	(6) Log in Desktop on Raspberry Pi Wirelessly . . . . .	267
6.2.7	(7) Open the remote desktop connection on Windows . . . . .	268
6.3	3. Preparations for Python . . . . .	271
6.3.1	3.1Hardware . . . . .	271
6.3.2	3.2Copy Example Code Folder to Raspberry Pi . . . . .	276

6.4	4. Projects	284
6.4.1	Project 1Python3 Shell	284
6.4.2	Project 2LED Blinks	288
6.4.3	Project 3: SOS Light	290
6.4.4	Project 4: Breathing LED	292
6.4.5	Project 5: Traffic Lights	295
6.4.6	Project 6Illuminating Lamp	297
6.4.7	Project 7RGB Light	299
6.4.8	Project 8Doorbell	302
6.4.9	Project 9: Passive Buzzer	304
6.4.10	Project 10Button-controlled LED	308
6.4.11	Project 11PIR Motion Sensor	311
6.4.12	Project 12Fire Alarm	313
6.4.13	Project 13Electronic Hourglass	316
6.4.14	Project 14Collision Alarm	319
6.4.15	Project 15Line Tracking Sensor	321
6.4.16	Project 16Photo Interrupter Module	323
6.4.17	Project 17Magnetic Detection	325
6.4.18	Project 185V Relay	327
6.4.19	Project 19: Touch capacitive Alarm	329
6.4.20	Project 20Obstacle Avoidance Sensor	332
6.4.21	Project 21Reed Switch Module	335
6.4.22	Project 22Vibration Sensor	337
6.4.23	Project 23Servo	340
6.4.24	Project 24Adjust the Brightness of LED	343
6.4.25	Project 25Photoresistor	349
6.4.26	Project 26Sound-activated Light	351
6.4.27	Project 27LCD1602	354
6.4.28	Project 28Water Level Monitor	357
6.4.29	Project 29Flower-watering Device	359
6.4.30	Project 30Temperature Alarm	362
6.4.31	Project 31: Steam Sensor	364
6.4.32	Project 32Gas Leakage Alarm	366
6.4.33	Project 33Alcohol Tester	369
6.4.34	Project 34Joystick Module	372
6.4.35	Project 35Ultrasonic Sensor	374
6.4.36	Project 36Light Intensity Detection	377
6.4.37	Project 37Pressure Detection	380
6.4.38	Project 38Temperature Detection	382
6.4.39	Project 39Ultraviolet Light Detection	385



## **1. DESCRIPTION:**

Raspberry Pi is a credit-card sized computer of low cost with Raspberry Pi OS as its official system and also compatible with other systems like ubuntu and Windows IoT. Furthermore, it extends out 40 pins to link with sensors or modules, which makes conducting all kind of experiments possible. You could get a camera monitor by plugging a camera to Raspberry Pi. Equally, the voice interactive function could be achieved if a microphone or a camera is connected with it.

And this is a purpose-built kit for Raspberry Pi enthusiasts, through which you could acquire knowledge of Linux and Python, Java and other programming languages, as well as the application of sensors or modules.

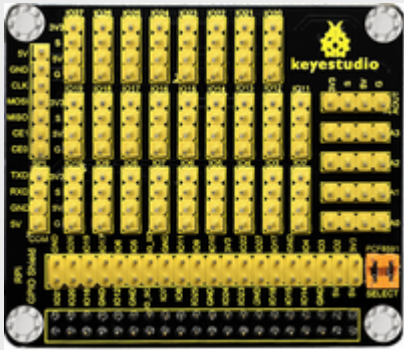

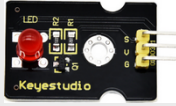
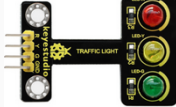


### **Resources**

Download code and more details, please refer to the following link: <https://fs.keyestudio.com/KS3016>







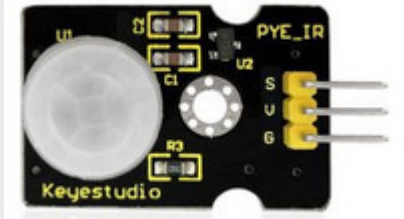







## 2. KIT LIST:

When you get this kit, please confirm whether all components listed below are delivered.

Product Name	Quantity	Picture
RPI GPIO-PCF8591 Shield	1	
White LED Module	1	
Red LED Module	1	
Traffic Light Module	1	
3W LED Module	1	
RGB Module	1	

continues on next page

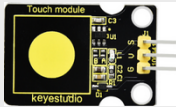




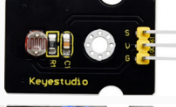








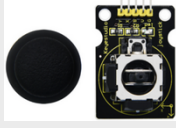
Table 1 – continued from previous page

Product Name	Quantity	Picture
Push Button Module	1	
Active Buzzer Module	1	
IR Obstacle Avoidance Module		
Passive Buzzer Module	1	
PIR Motion Sensor	1	
Flame Sensor	1	
Tilt Sensor	1	
Collision Sensor	1	
Line-tracking Sensor	1	
Photo Interrupter Module	1	
Hall Magnetic Sensor	1	
5V Relay Module	1	

continues on next page








Table 1 – continued from previous page

Product Name	Quantity	Picture
Capacitive Touch Sensor	1	
Reed Switch Sensor	1	
Vibration Sensor	1	
Relay Module	1	
Rotary Potentiometer Module	1	
Photoresistor Sensor	1	
Analog Sound Sensor	1	
I2C LCD1602 Module	1	
Water Level Sensor	1	
Soil Humidity Sensor	1	
LM35 Temperature Sensor	1	
Steam Sensor	1	
MQ-2Gas Sensor	1	
MQ-3 Alcohol Sensor	1	
Joystick Module	1	

continues on next page

Table 1 – continued from previous page

Product Name	Quantity	Picture
Ultrasonic Module	1	
TEMT6000 Ambient Light Sensor	1	
Thin-film Pressure Sensor	1	
Analog Temperature Sensor	1	
GUVA-S12SD Ultraviolet Sensor	1	
F-F DuPont Wire 40P	1	
Screwdriver	1	
USB Cable	1	

**3. RESOURCES:**

<https://fs.keyestudio.com/KS3016>



## C LANGUAGE TUTORIAL

Raspberry Pi and electronic components are controlled via C language here.

### 4.1 1. Install Raspberry Pi OS System

#### Hardware Tool

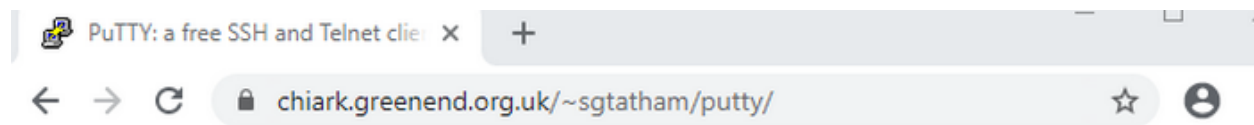
- Raspberry Pi 4B/3B/2B
- Above 8G TFT SD Card
- Card Reader
- Computer and other parts

#### Install Software Tool

#### Windows System

Install putty firstly:

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 [Simon Tatham](#).

The latest version is 0.74 [Download it here](#).

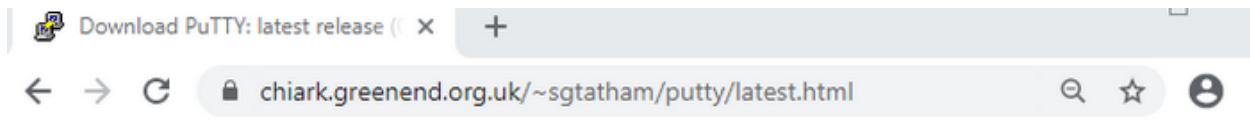
**LEGAL WARNING:** Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. We believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but we are not lawyers, and so if in doubt you should seek legal advice before downloading it. You may find useful information at [cryptolaw.org](#), 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 PuTTY: latest release (0.74)

[Home](#) | [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 [permanent link to the 0.74 release](#).

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 [development snapshots](#), to see if the problem has already been fixed in those versions.

### Package files

You probably want one of these. They include versions of all the PuTTY utilities.


(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

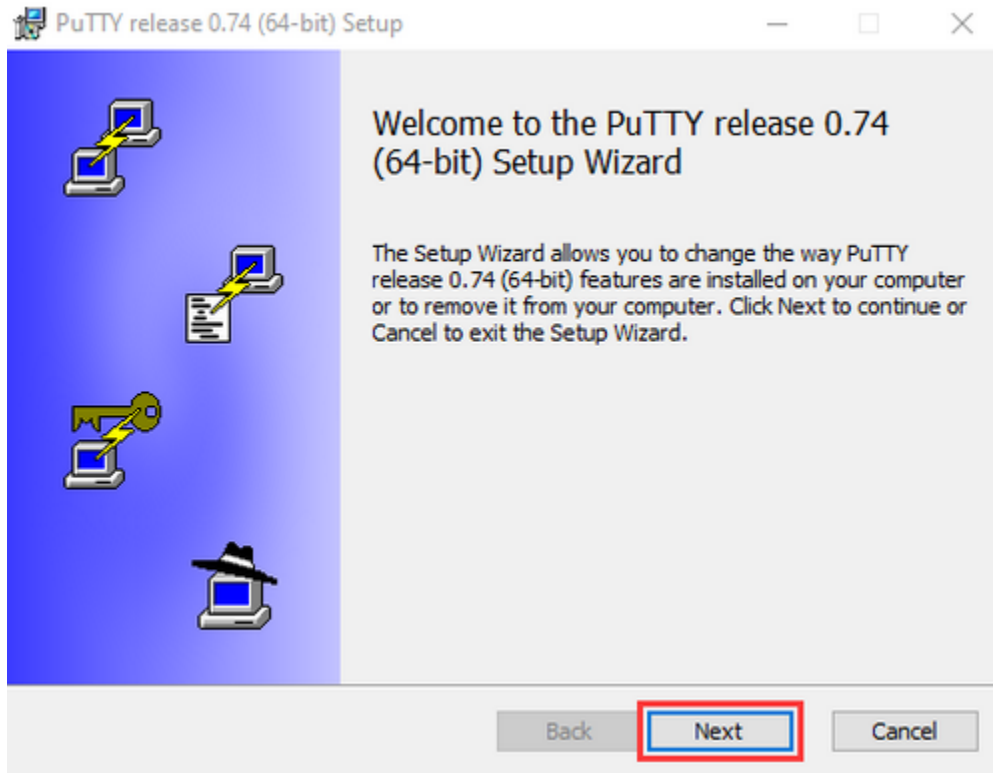
#### MSI ('Windows Installer')

32-bit:	<a href="#">putty-0.74-installer.msi</a>	<a href="#">(or by FTP)</a>	<a href="#">(signature)</a>
64-bit:	<a href="#">putty-64bit-0.74-installer.msi</a>	<a href="#">(or by FTP)</a>	<a href="#">(signature)</a>

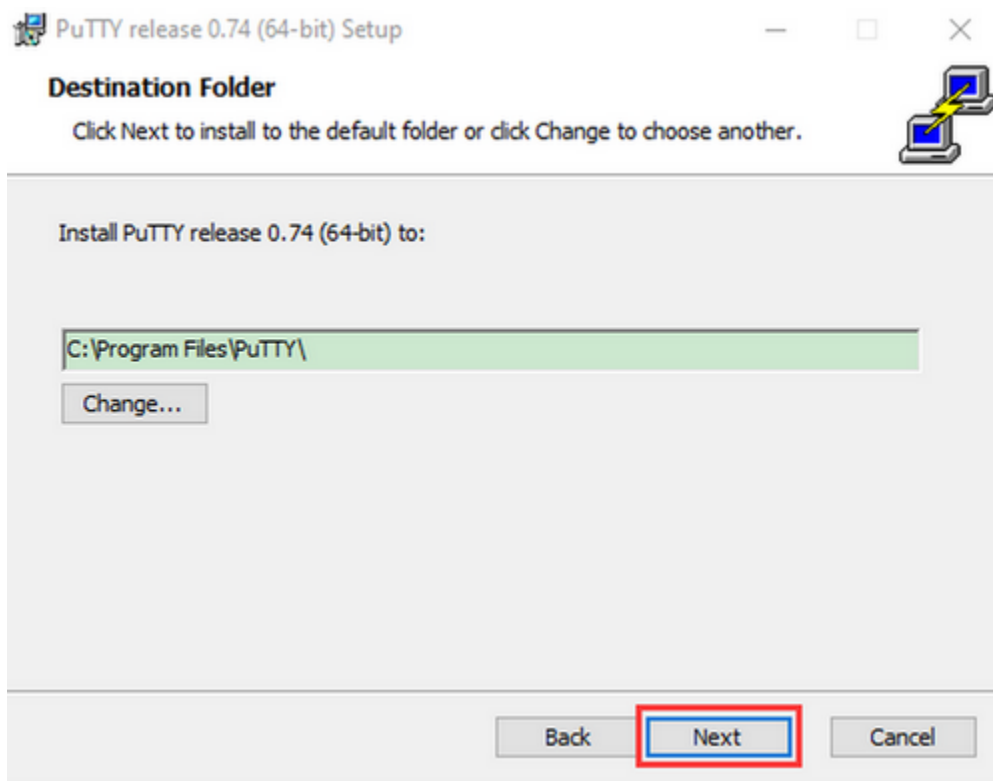
#### Unix source archive

.tar.gz:	<a href="#">putty-0.74.tar.gz</a>	<a href="#">(or by FTP)</a>	<a href="#">(signature)</a>
----------	-----------------------------------	-----------------------------	-----------------------------

After downloading the driver file  `putty-64bit-0.74-installer` double-click it and tap "Next"

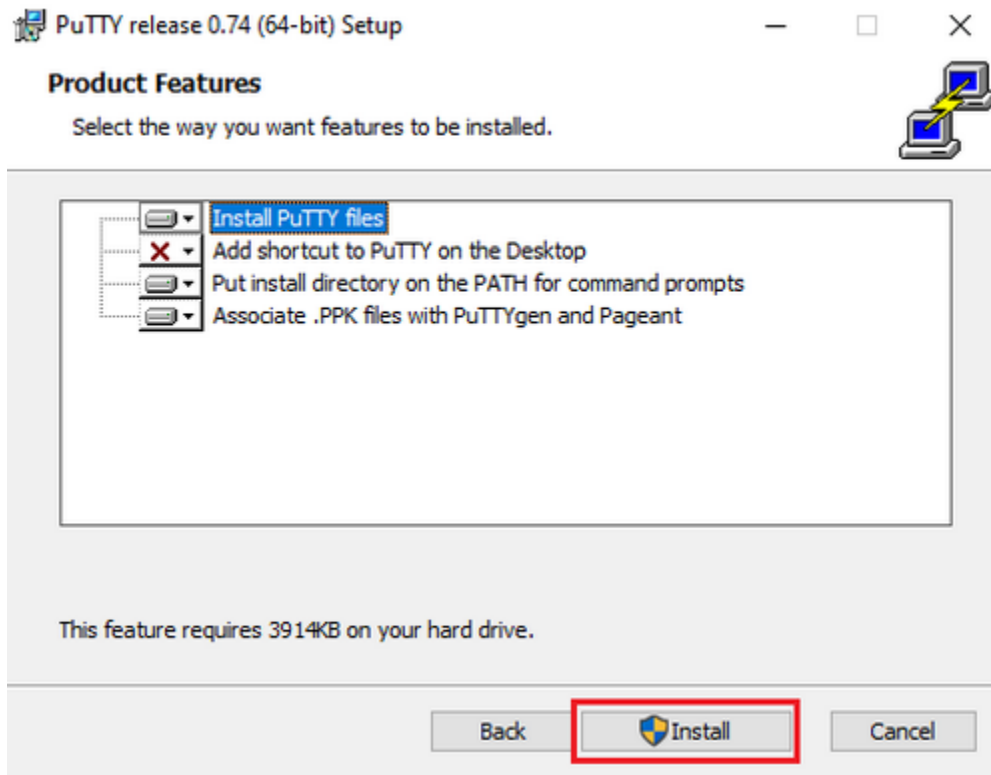


Click "Next"

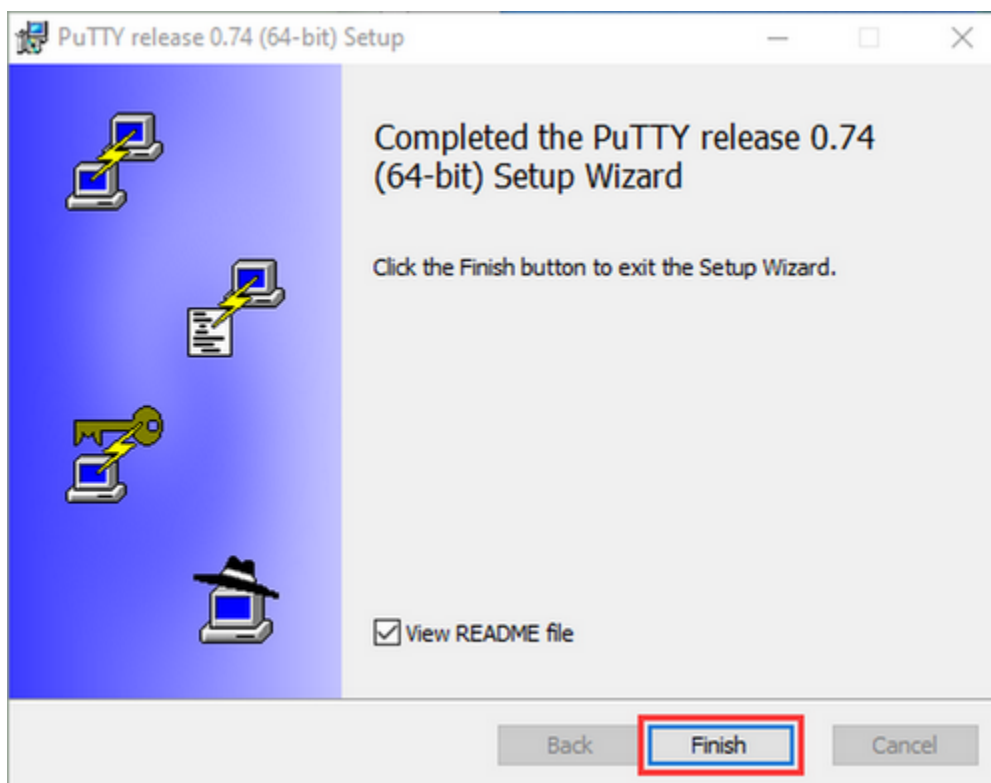


Select "Install Putty files" and click "Install".







After a few seconds, click "Finish".



### SSH Remote Login software -WinSCP

Download WinSCP: <https://winscp.net/eng/download.php>

After the download, click  **WinSCP-5.17.9-Setup.exe** and  **Install for all users (recommended)**.

Select Setup Install Mode



## Select install mode

WinSCP can be installed for all users (requires administrative privileges), or for you only.

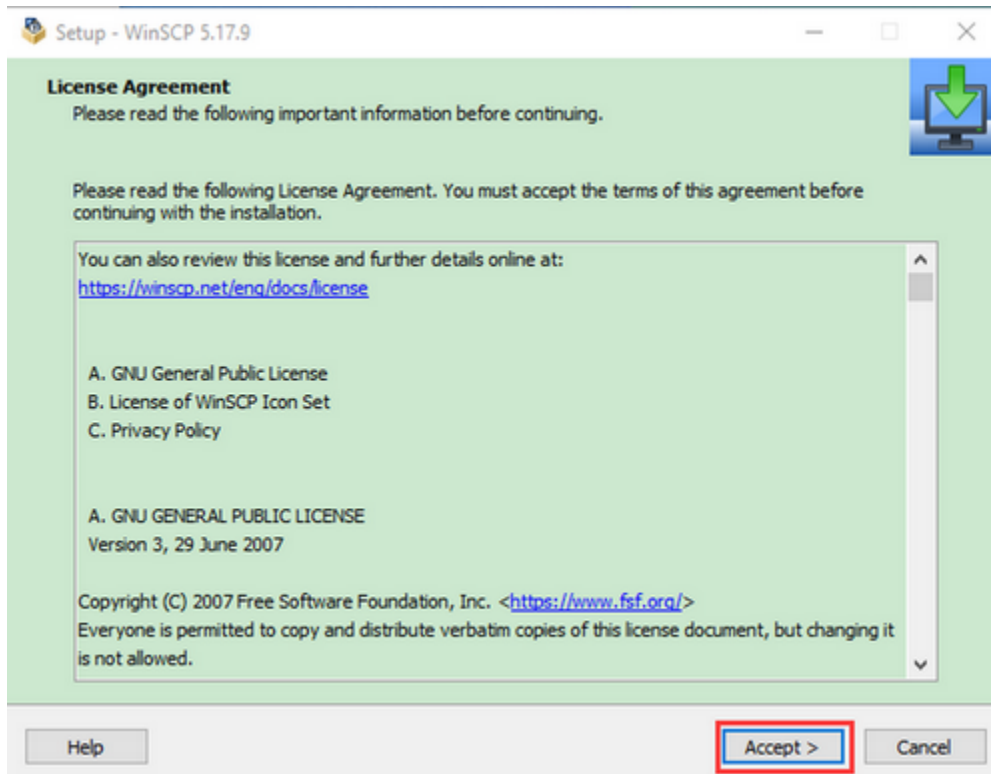


**Install for all users (recommended)**

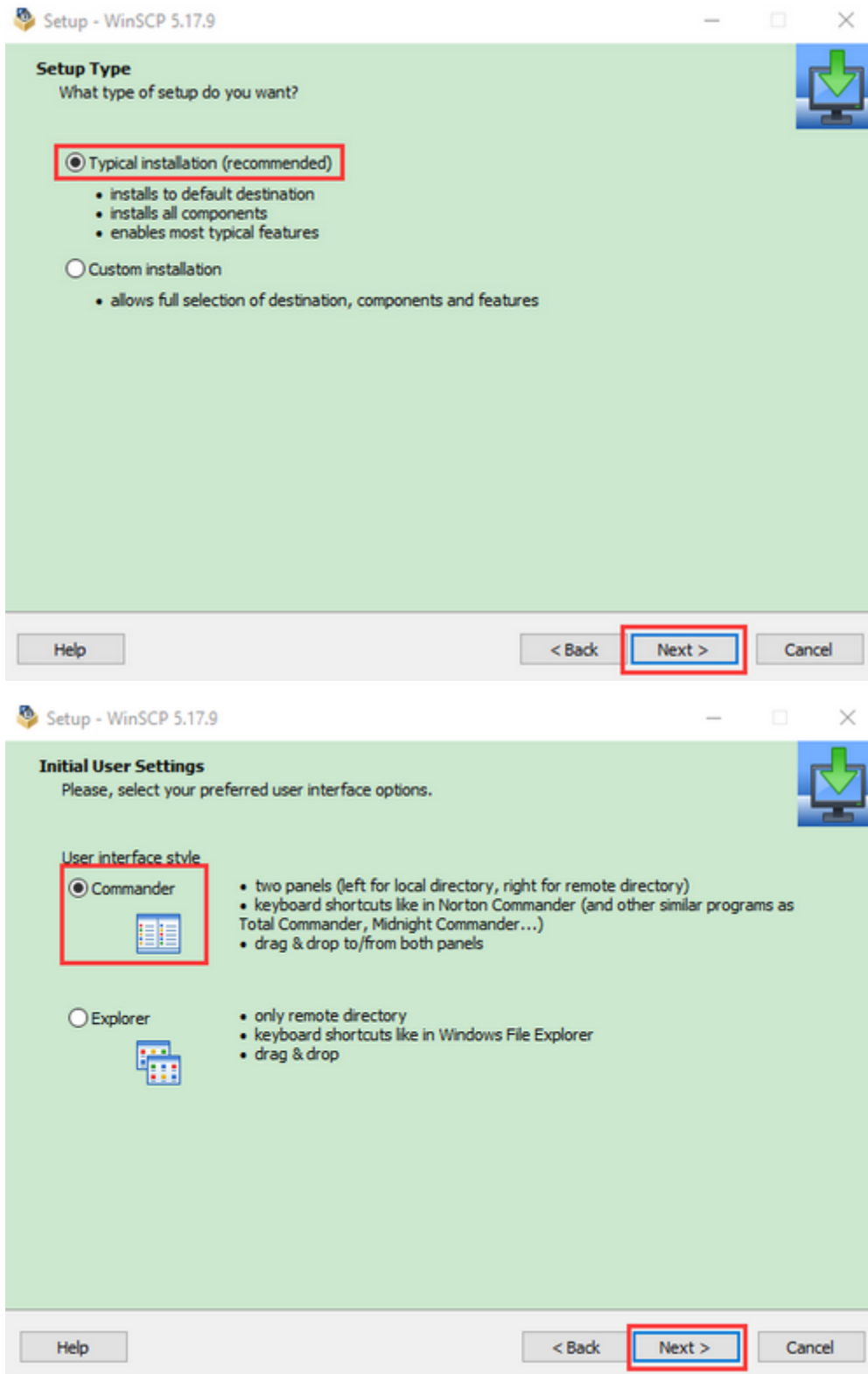
→ Install for me only

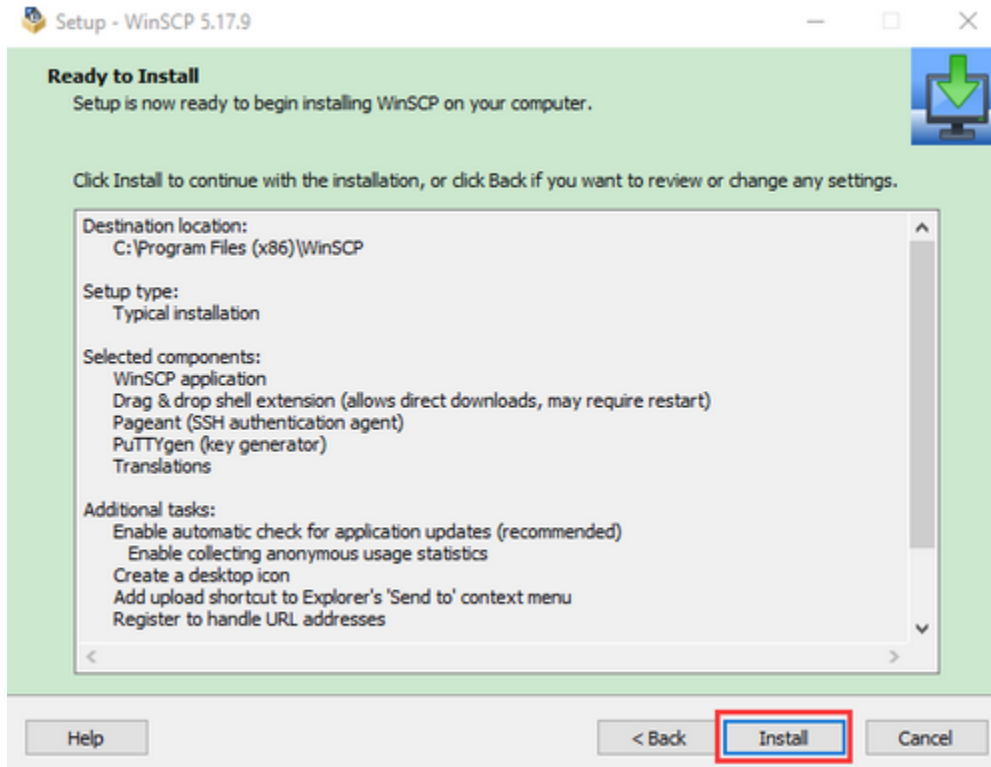
Cancel

Click "Accept"



Follow the below steps to finish the installation.





After a few seconds, the installation is completed and click“Finish”;



## SD Card Formatter

Format TFT card tool


## Download SD Card Formatter

[http://www.canadiancontent.net/tech/download/SD\\_Card\\_Formatter.html](http://www.canadiancontent.net/tech/download/SD_Card_Formatter.html)


### SD Card Formatter

Free Formatter Download

Software Downloads > Hardware Software > Hard Drive Software > Hard Drive Formatters >



**SD Card Formatter 5.0.1**  
 Update Submitted 12 May 2019




**Software Review:**

SD Card Formatter is a simple and basic formatted which is designed to be used with SD, SDHC and SDXC memory cards.

The application itself isn't too different from the format utility included with Windows and includes two modes: Quick format and Overwrite format. CHS format size adjustment is the only other option.


Once the appropriate card and volume label has been selected, the format can begin after hitting "Format".

Version 5.0.1 is a freeware program which does not have restrictions and it's free so it doesn't cost anything.



SD Card Formatter screenshot

### Download File



**Download SD Card Formatter**  
 6 MB - Filesize

### Details

<b>Publisher:</b>	Tuxera
<b>License:</b>	Freeware
<b>OS/Platform:</b>	Windows 7, Windows 8 (64-bit, 32-bit) / Vista / XP
<b>Filesize:</b>	6 MB
<b>Filename:</b>	SDCardFormatterv5_WinEN.z...
<b>Cost (Full Version):</b>	Free
<b>Rating:</b>	3 out of 5 based on 1 rating.
<b>Notes</b>	This file download is licensed as freeware for Windows 7, Windows 8 (64-bit, 32-bit) / Vista / XP.
<b>TrustRank</b>	Based on many factors, we give this program a Trust rating of 5 / 10.


CanadianContent

Register Account

Software Downloads » Hardware Software » Hard Drive Software » Hard Drive Formatters » SD Card Formatter »

Download SD Card Formatter

## Download SD Card Formatter 5.0.1 (x64 & x32) Free



Have you tried the SD Card Formatter before? If yes, please consider recommending it by clicking the Facebook "Recommend" button!

Download SD Card Formatter 5.0.1 from **Hosted by Sdcard.org**


SD Card Formatter has been tested for viruses and malware

This download is 100% clean of viruses. It was tested with 24 different antivirus and anti-malware programs and was clean **100%** of the time. View the full SD Card Formatter homepage for virus test results.


The file that was tested: SDCardFormatterv5\_WinEN.zip.

Tip: If you're experiencing trouble downloading this file, please disable any download managers to SD Card Formatter you may be using.

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.


### SD Card Formatter - InstallShield Wizard



## Preparing to Install...

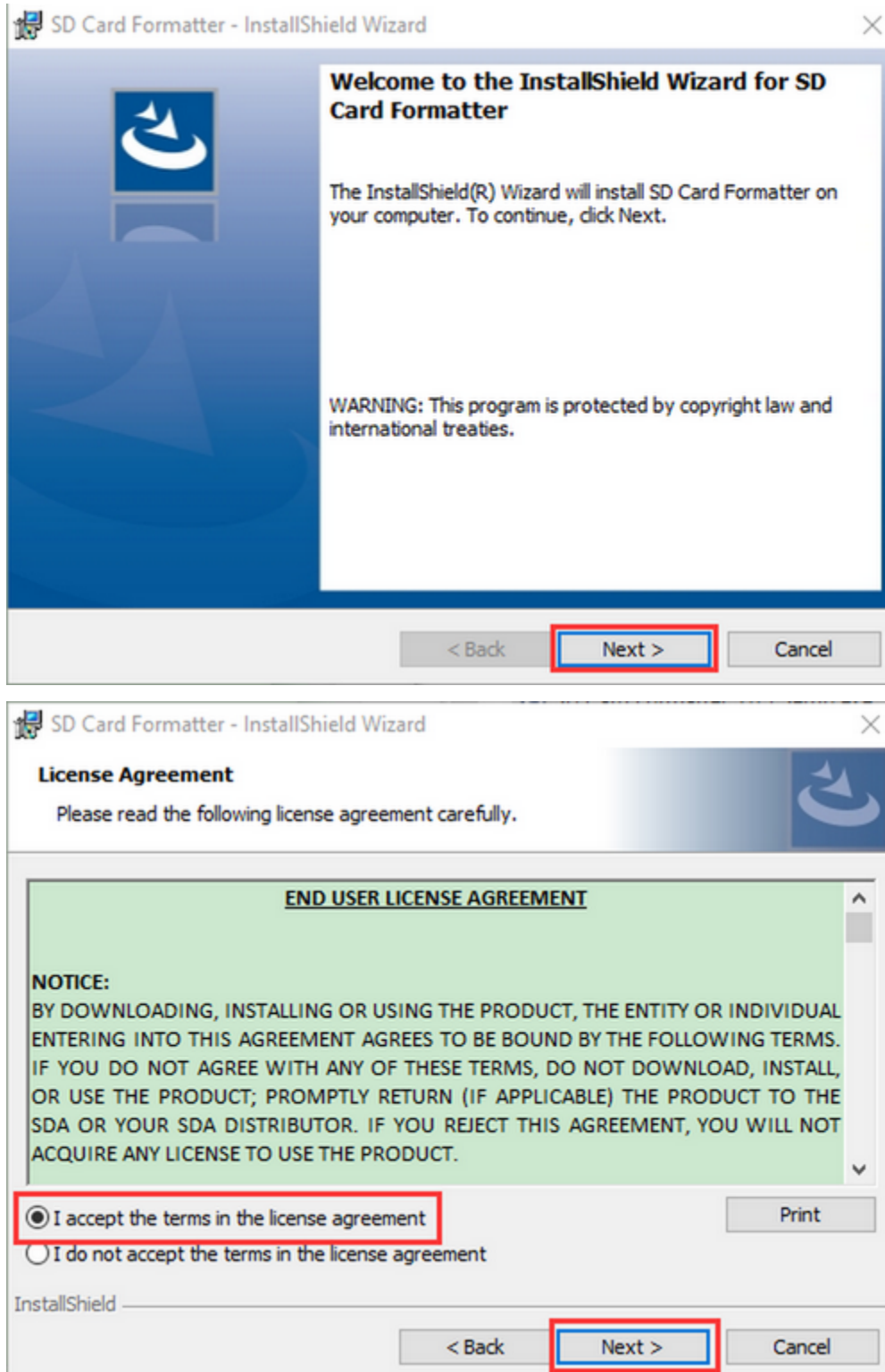
SD Card Formatter Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.

Extracting: SD Card Formatter Setup.msi



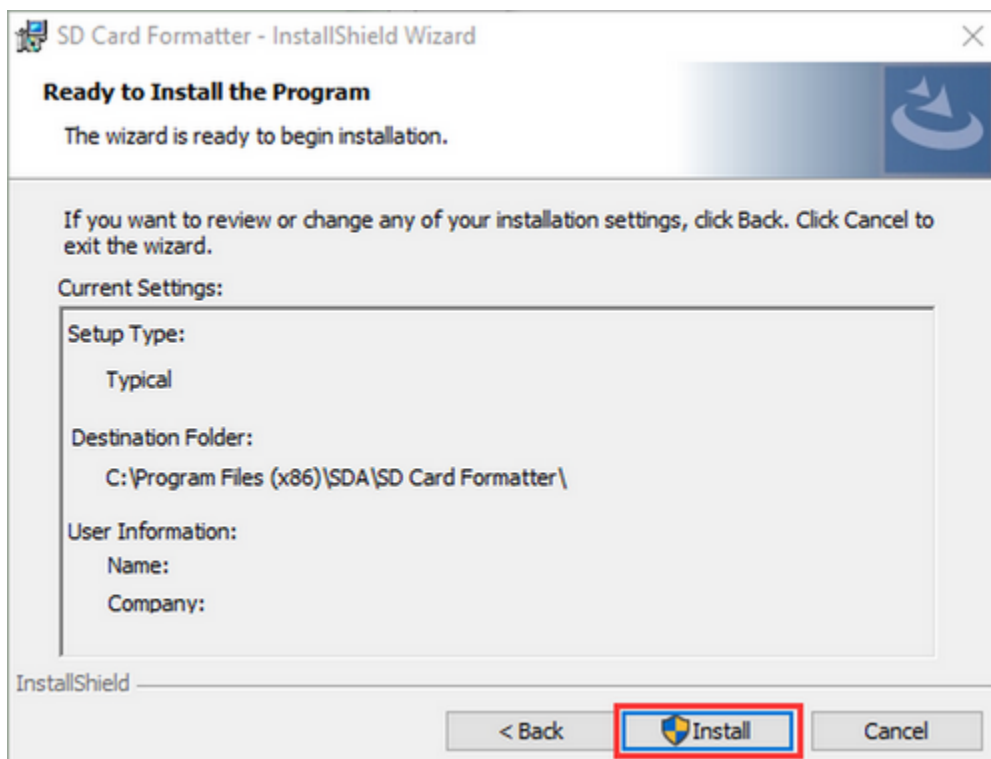
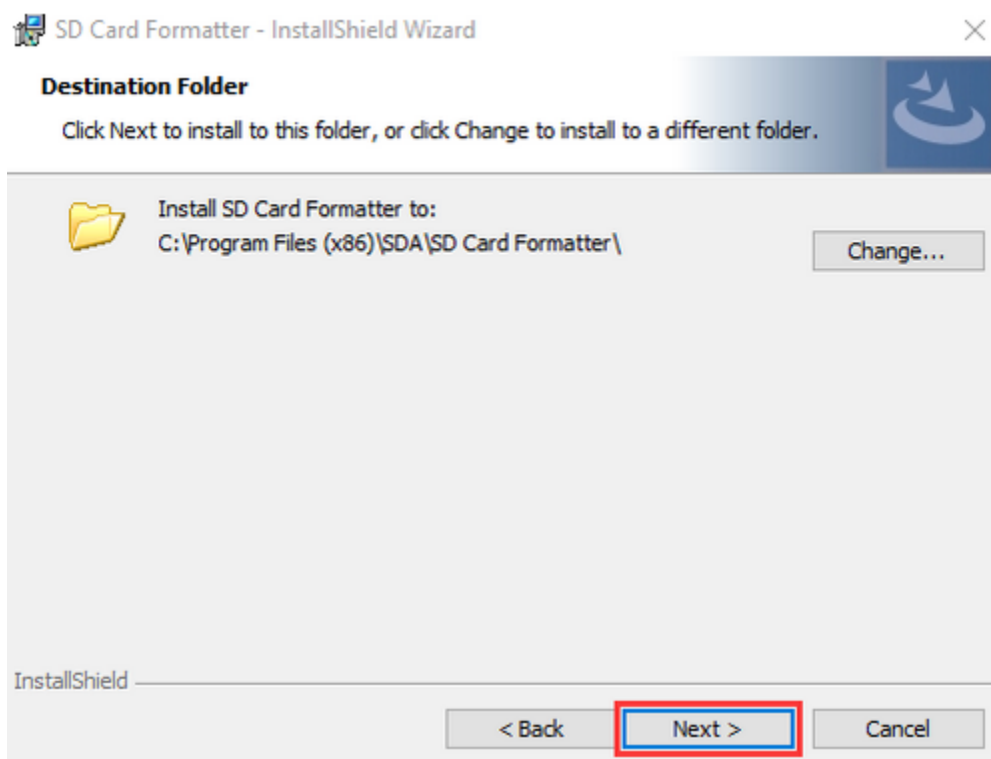
Cancel

Click "Next" and choose ☒ **I accept the terms in the license agreement**, then tap "Next"



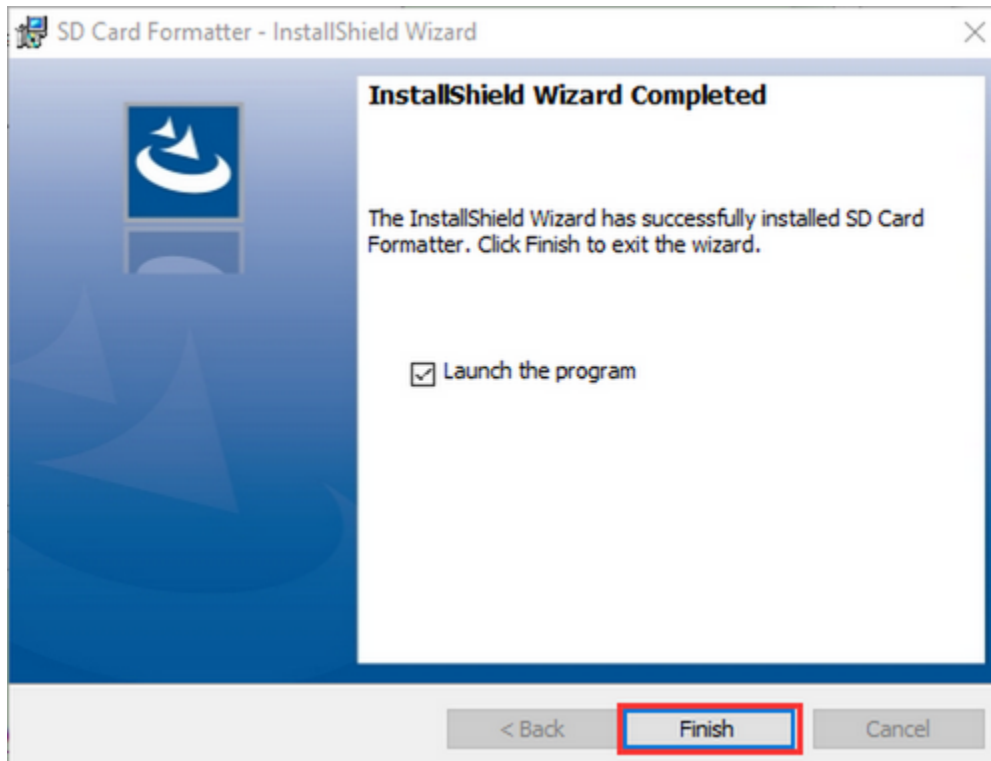
Click "Next" and "Install".





After a few seconds, click“Finish”






### Burn Win32DiskImager

Download Link <https://sourceforge.net/projects/win32diskimager/>


Home / Browse / System Administration / Storage / Win32 Disk Imager




## Win32 Disk Imager

A Windows tool for writing images to USB sticks or SD/CF cards

Brought to you by: [gruemaster](#), [tuxinator2009](#)




★★★★★ 112 Reviews
Downloads: 42,251 This Week
Last Update: 2018-06-07

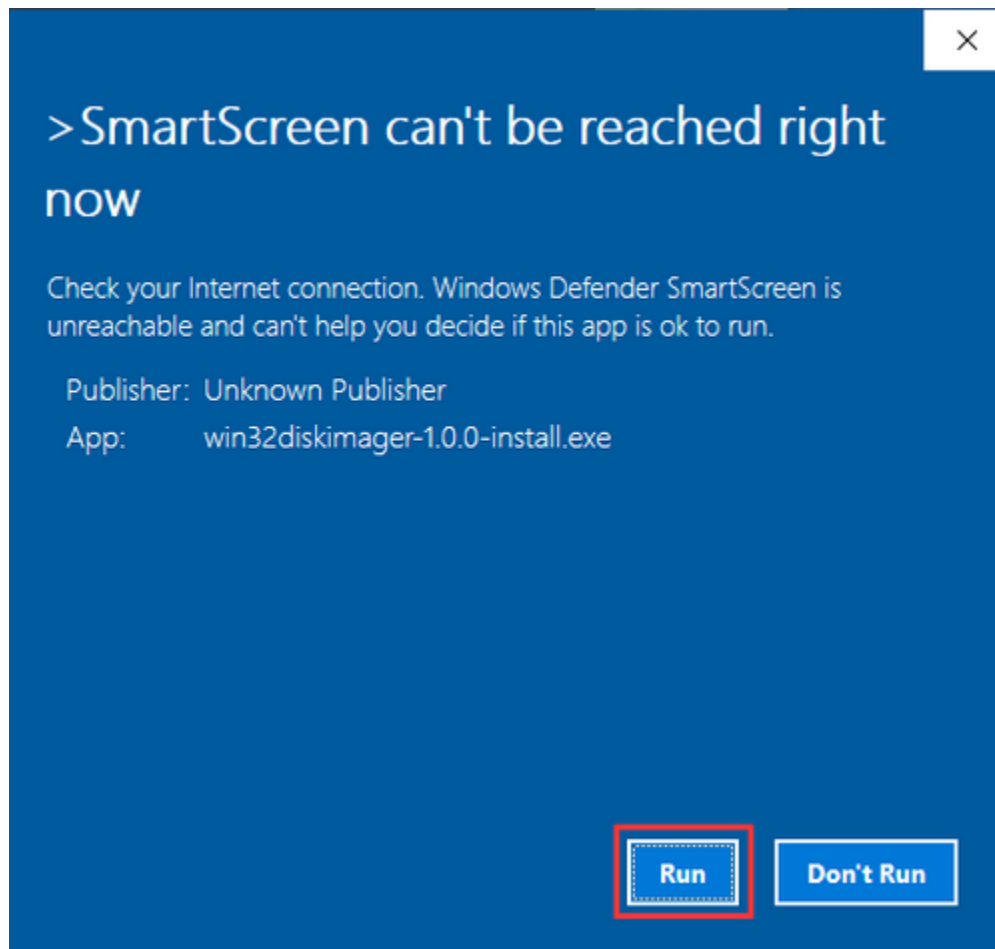

Get Updates
Share This

Summary Files Reviews Support Wiki Feature Requests Bugs Code Mailing Lists Blog

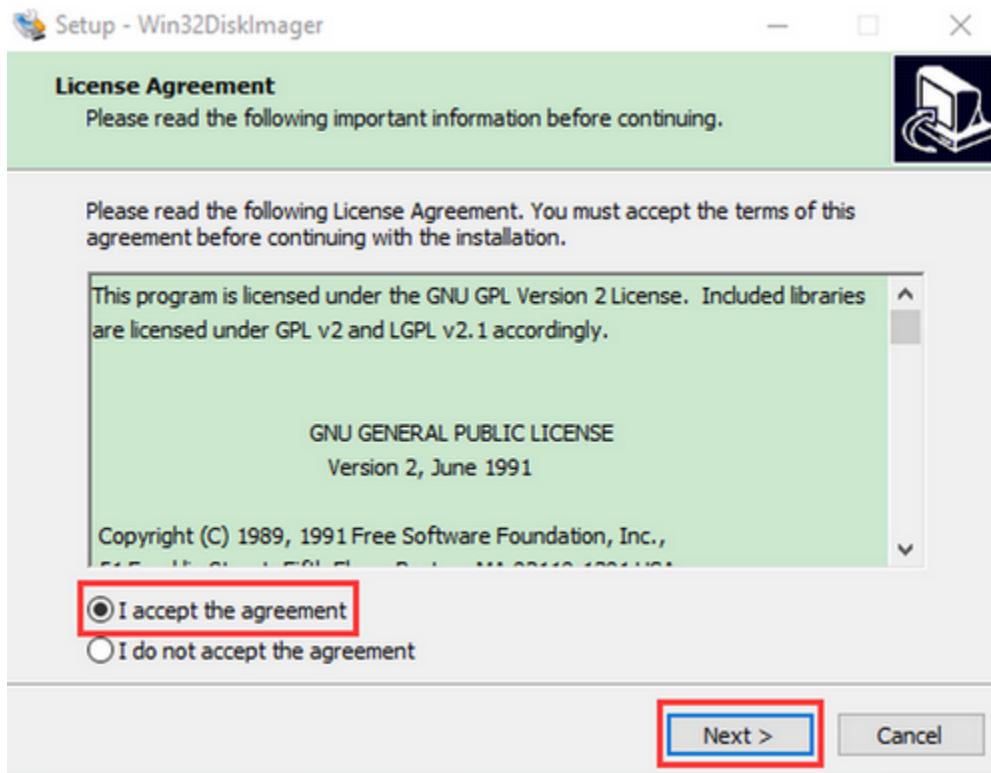
This program is designed to write a raw disk image to a removable device or backup a removable device to a raw image file. It is very useful for embedded development, namely Arm development projects (Android, Ubuntu on Arm, etc). Anyone is free to branch and modify this program. Patches are always welcome.

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 developers). For Windows XP/Vista, please use v0.9 (in the files archive).

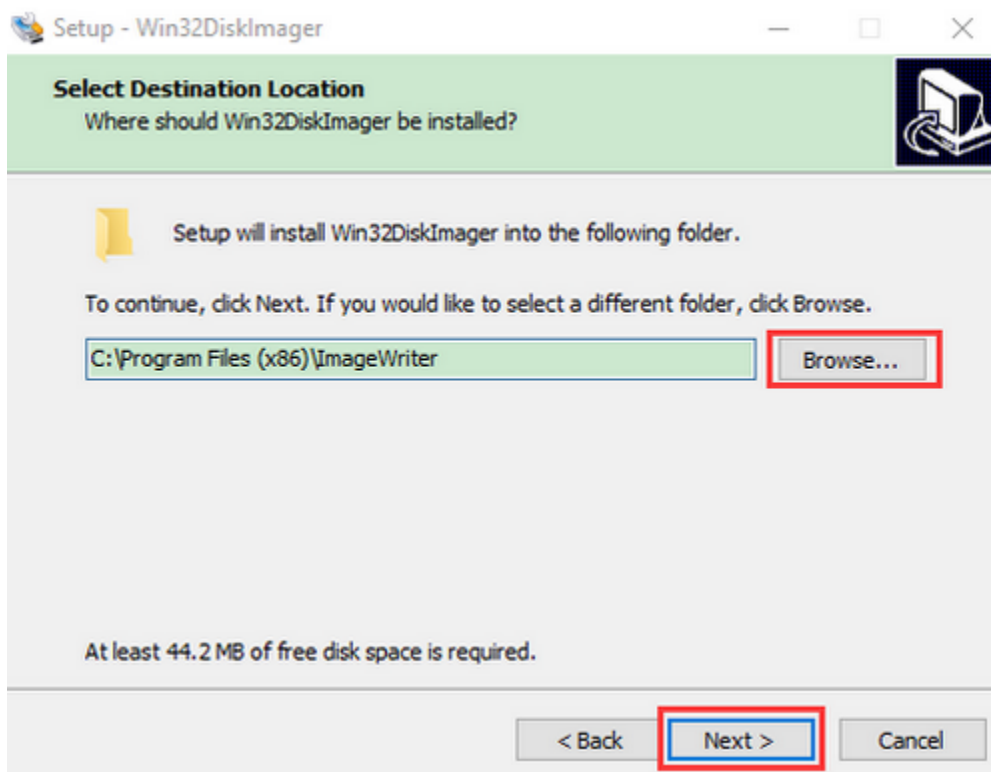
- a. After the download, double-click  **win32diskimager-1.0.0-install.exe** and tap “Run”



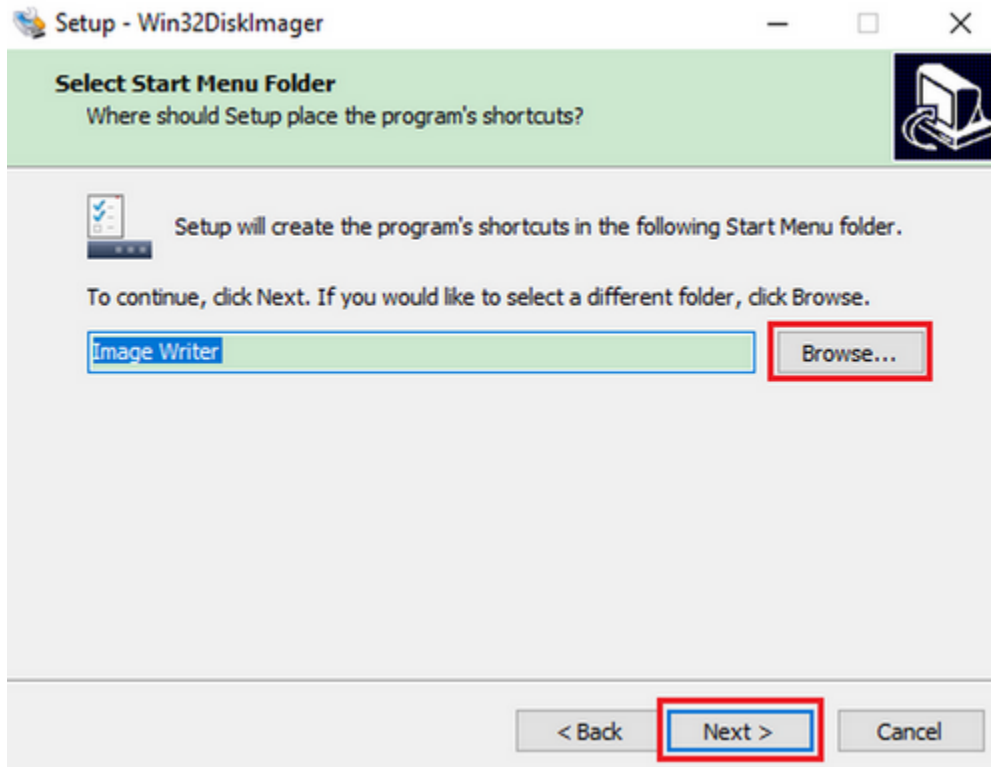
b. Select ☒ I accept the agreement and tap "Next".



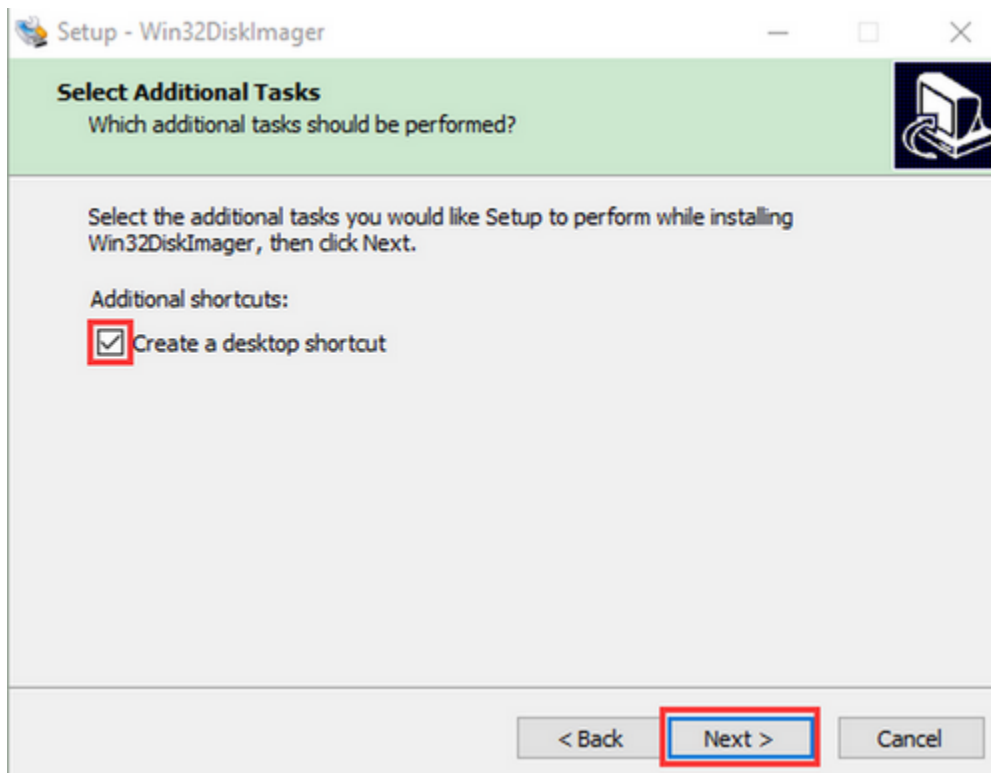
c. Click "Browse..." and find out the folder where the Win32DiskImager is located, tap "Next".

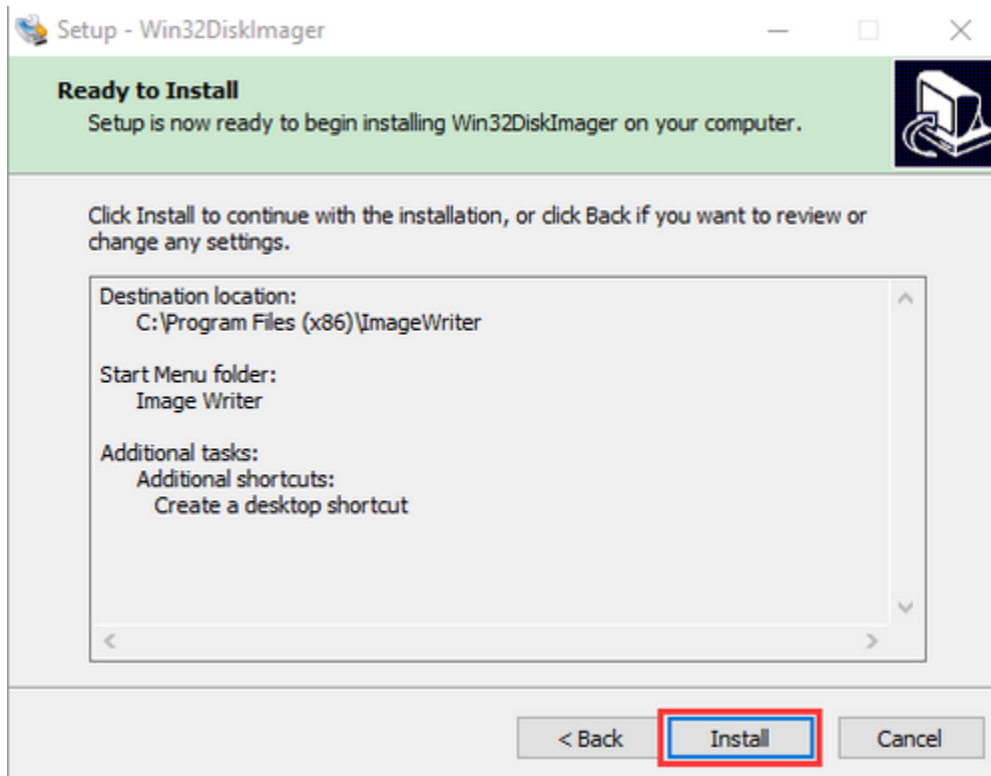


d. Tick **Create a desktop shortcut**, click "Next" and "Install"

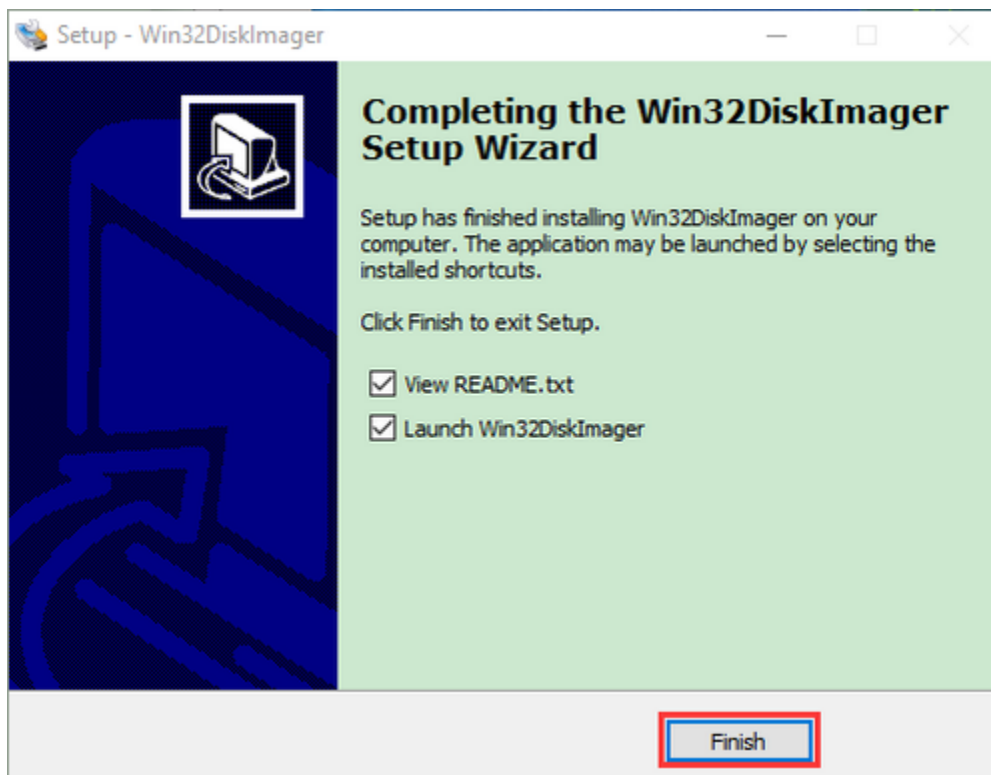


d. Tick **Create a desktop shortcut**, click "Next" and "Install"





e. After a few seconds, click“Finish”. The installation is finished



Scan to search ip address software tool—WNetWatcher

Download Link<http://www.nirsoft.net/utis/wnetwatcher.zip>

## Raspberry Pi Imager

<https://www.raspberrypi.org/downloads/raspberry-pi-os/>

(recommend downloading the version with desktop and commonly used software)

# Operating system images

Many operating systems are available for Raspberry Pi, including Raspberry Pi OS, our official supported operating system, and operating systems from other organisations.

[Raspberry Pi Imager](#) is the quick and easy way to install an operating system to a microSD card ready to use with your Raspberry Pi. Alternatively, choose from the operating systems below, available to download and install manually.

Download:  
[Raspberry Pi OS \(32-bit\)](#)  
[Raspberry Pi Desktop](#)  
[Third-Party operating systems](#)

## Raspberry Pi OS

Compatible with:  
[All Raspberry Pi models](#)



### Raspberry Pi OS with desktop and recommended software

Release date: December 2nd 2020  
Kernel version: 5.4  
Size: 2.949MB  
[Show SHA256 file integrity hash:](#)  
[Release notes](#)

Download

[Download torrent](#)

### Raspberry Pi OS with desktop

Release date: December 2nd 2020  
Kernel version: 5.4  
Size: 1.177MB  
[Show SHA256 file integrity hash:](#)  
[Release notes](#)

Download

[Download torrent](#)

### Raspberry Pi OS Lite

Release date: December 2nd 2020  
Kernel version: 5.4  
Size: 438MB  
[Show SHA256 file integrity hash:](#)  
[Release notes](#)

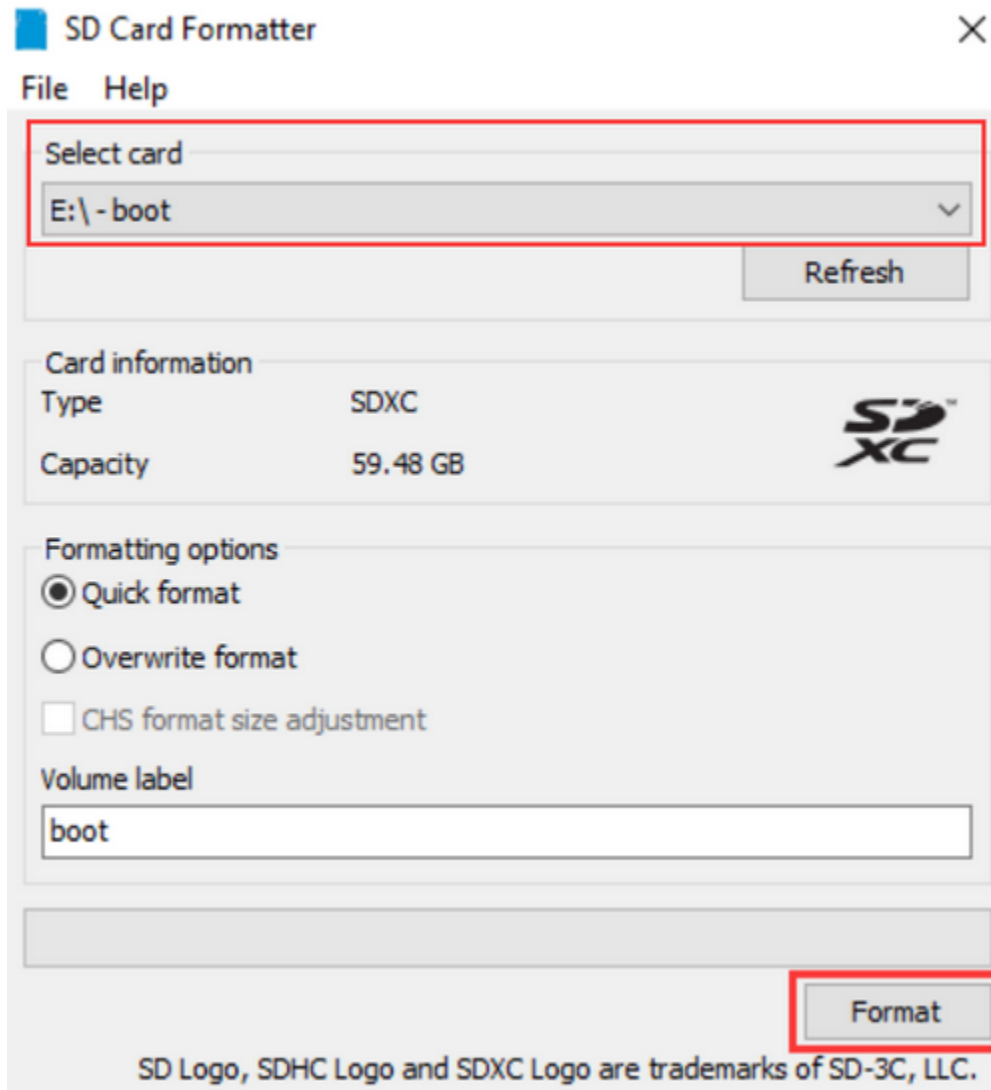
Download

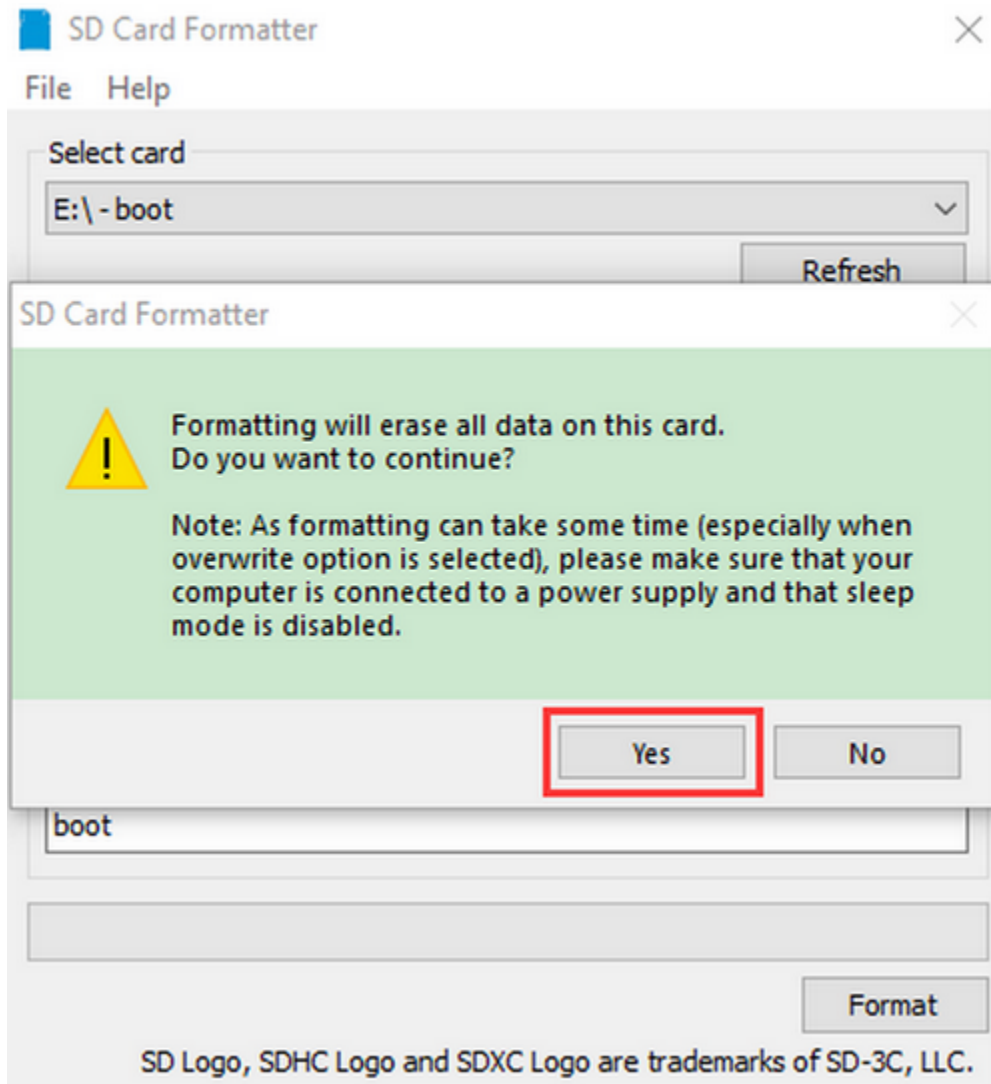
[Download torrent](#)

## 4.2 2. Install Raspberry Pi OS on Raspberry Pi 4B

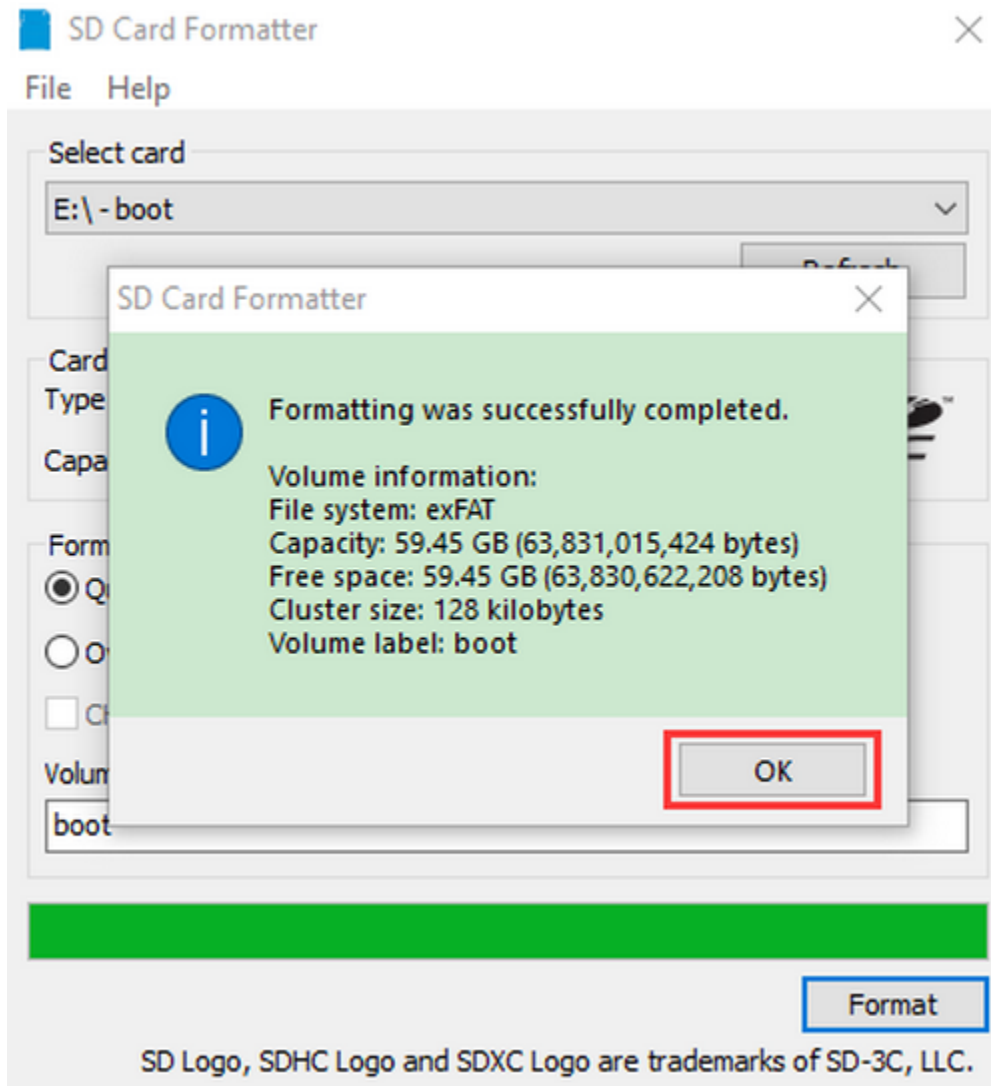
Insert TFT RAM card to card reader, then interface card reader to USB port of computer.

Format TFT RAM card with SD Card Formatter software, as shown below:



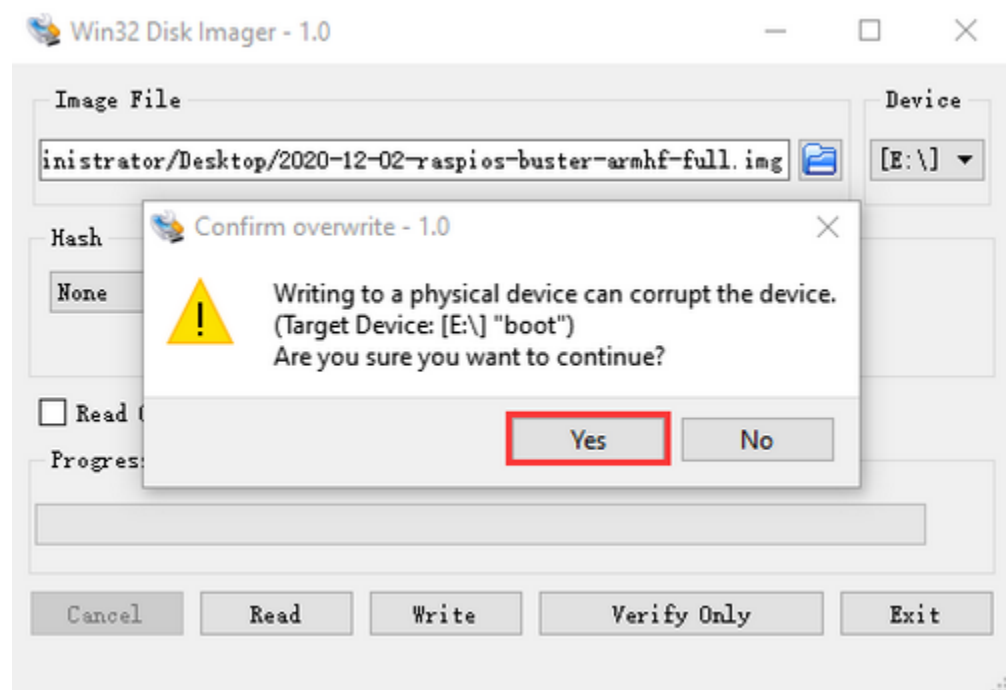
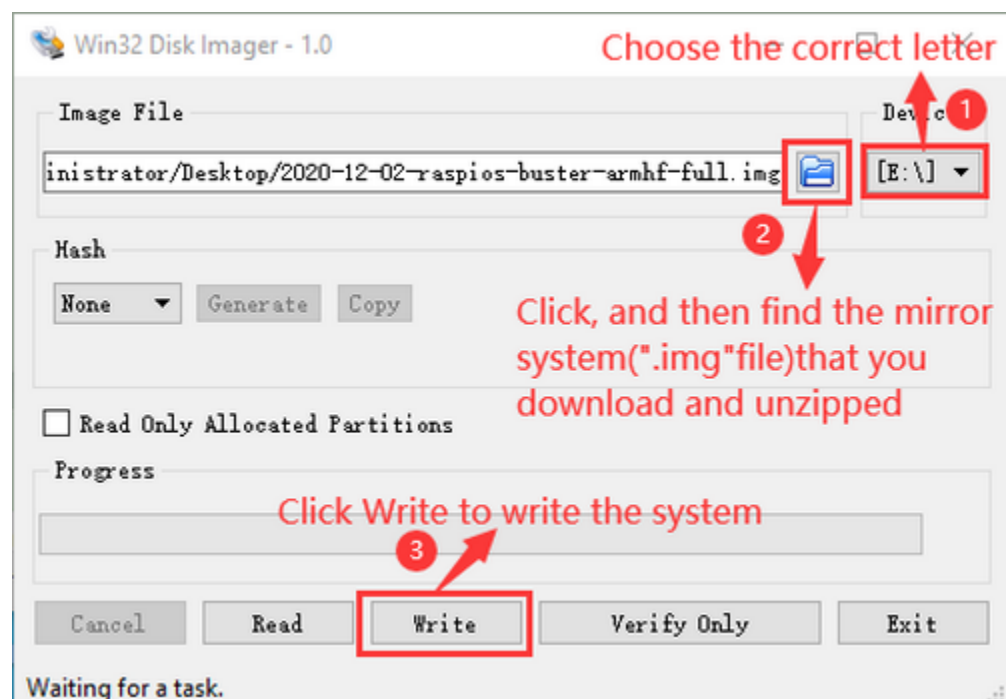


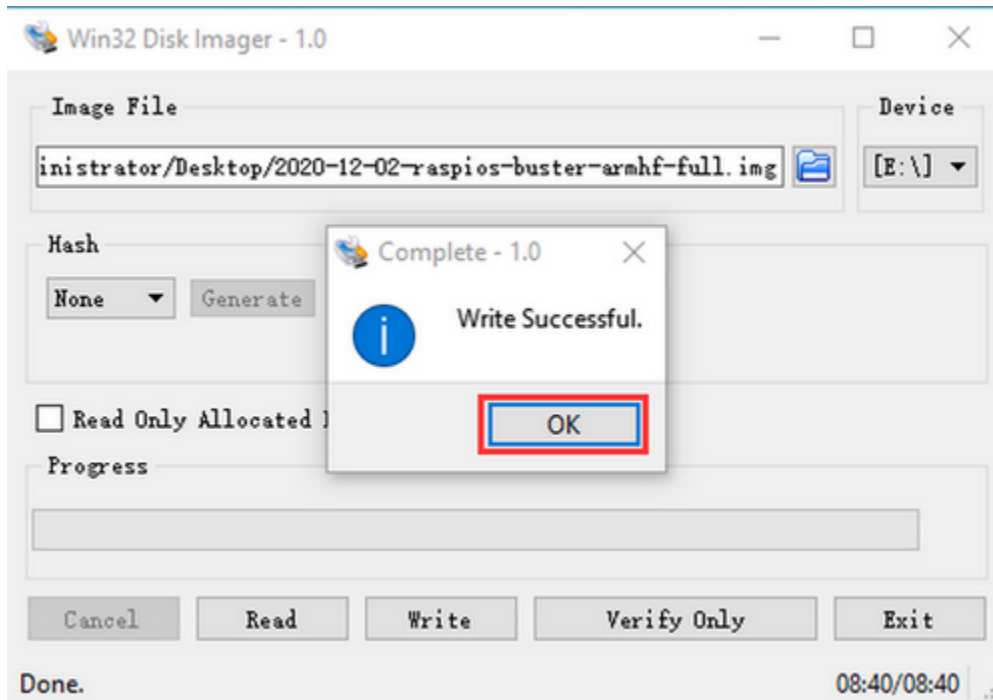




### Burn System

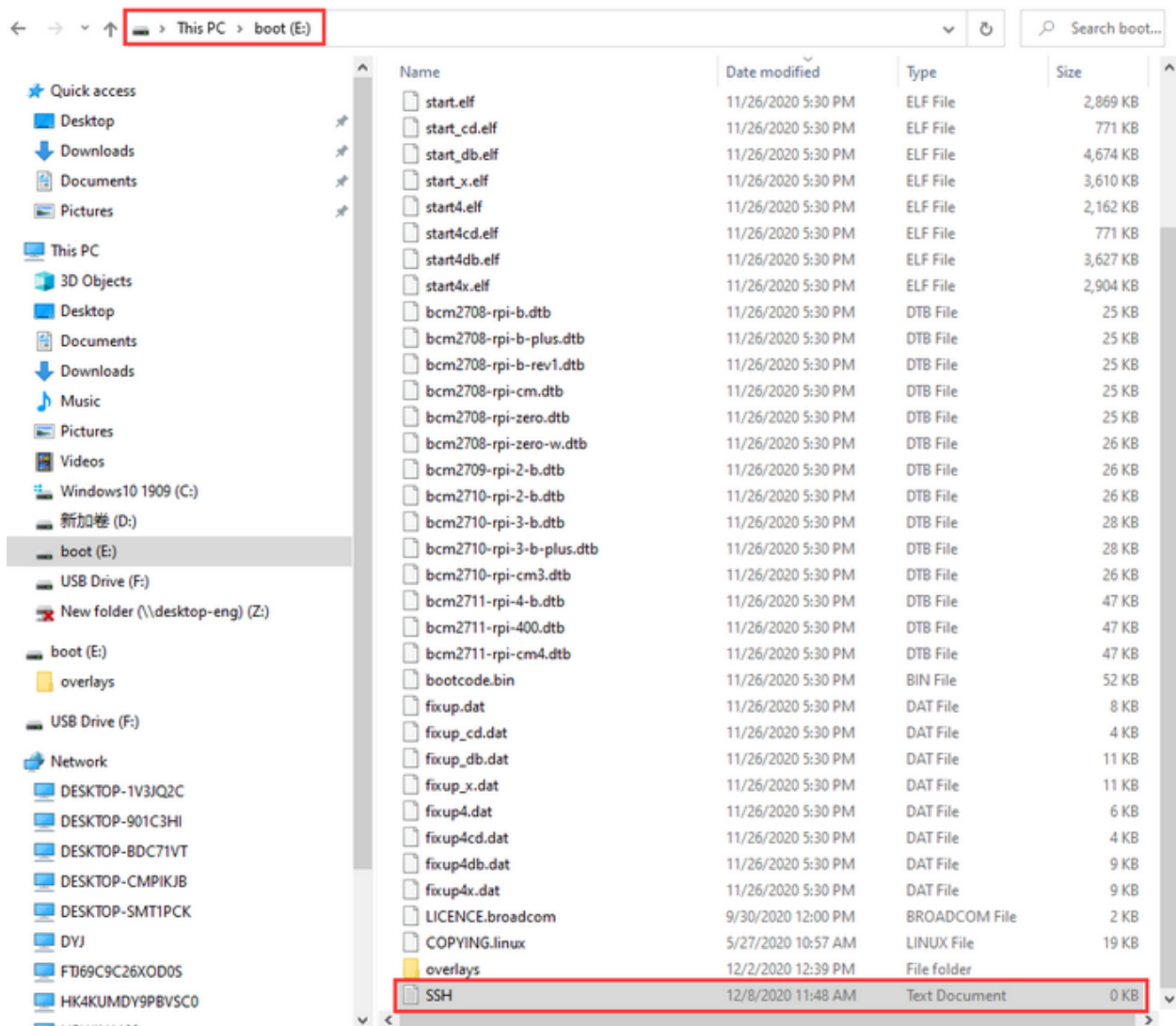
Burn the Raspberry Pi OS system to TFT card using Win32DiskImager software





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.



## Eject Card Reader

Log in systemraspberry and PC should be in the same local area network

Insert TFT card into Raspberry, connect internet cable and plug in power.

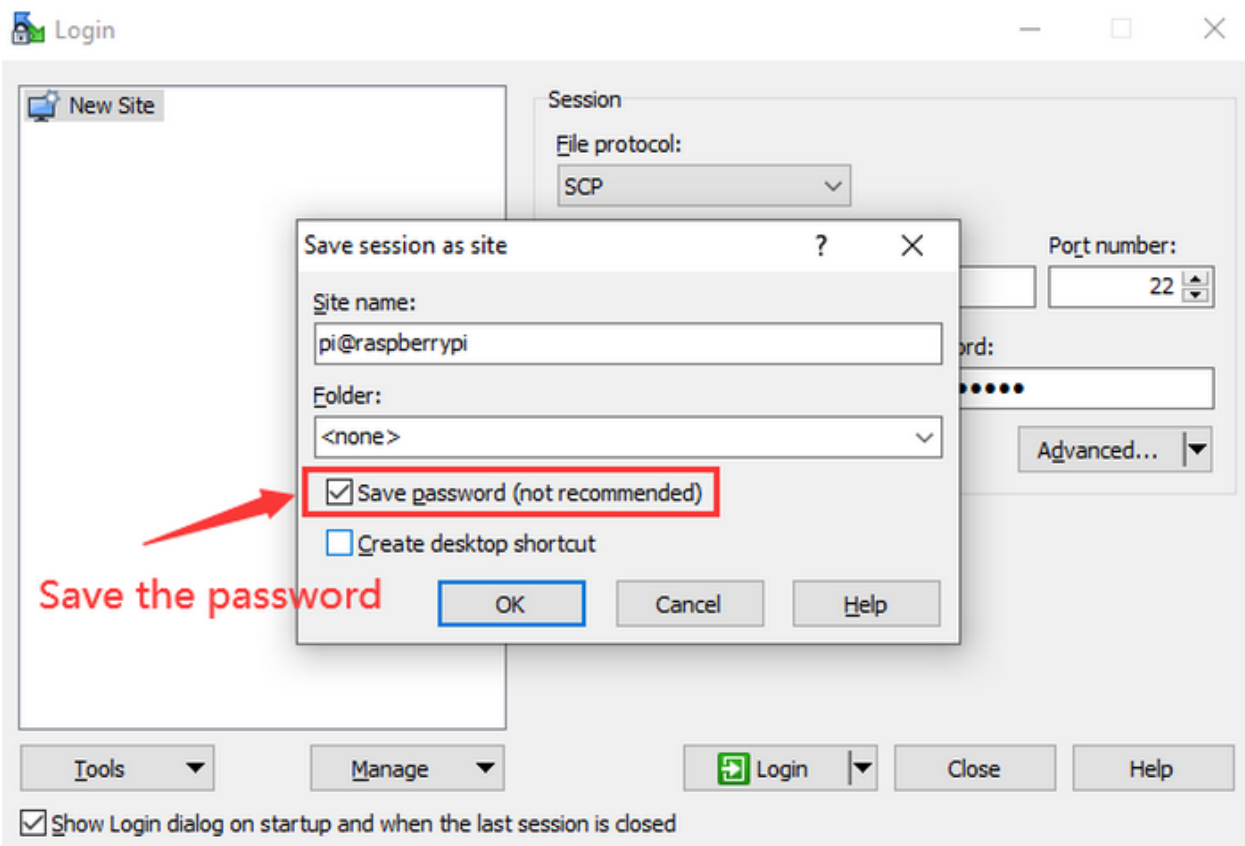
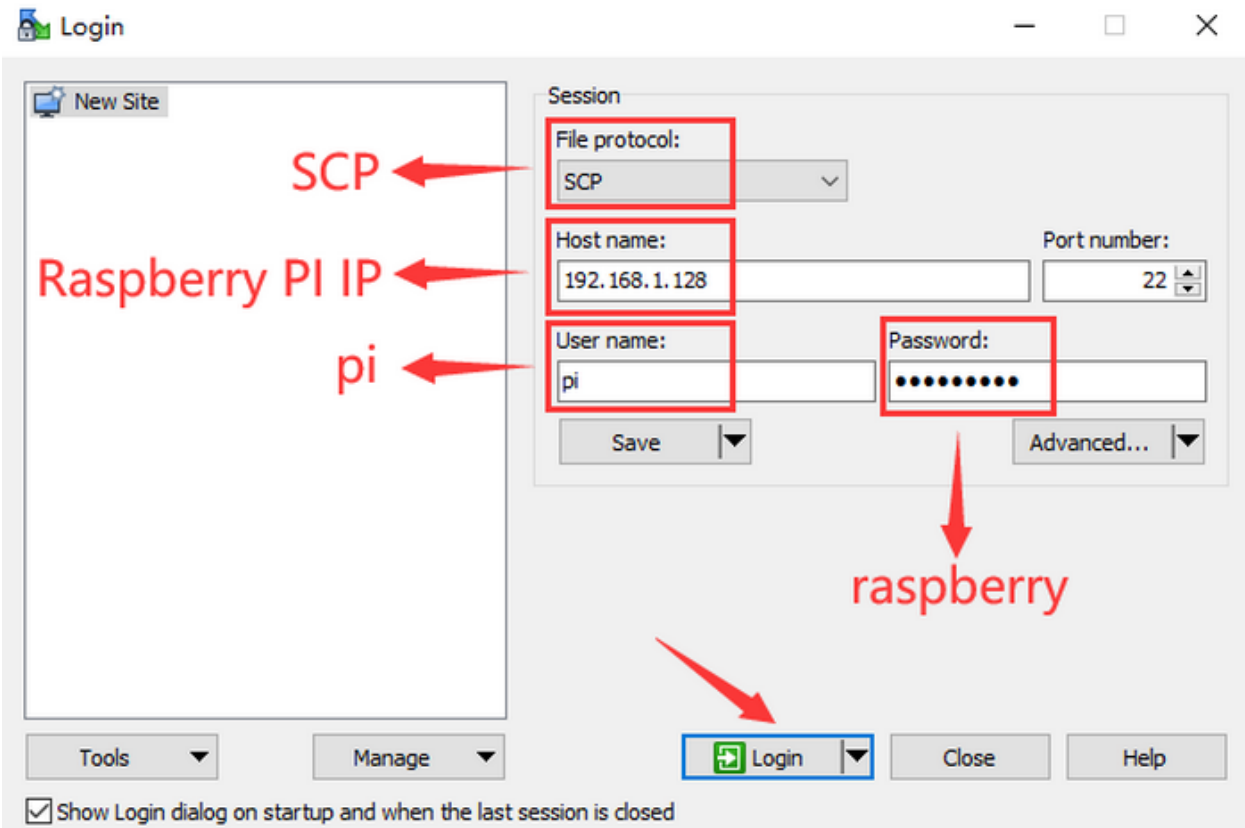
If you have screen and HDMI cable of Raspberry Pi, you could view Raspberry Pi OS system activating.

If not, you can enter the desktop of Raspberry Pi via SSH remote login software—WinSCP and xrdp login.

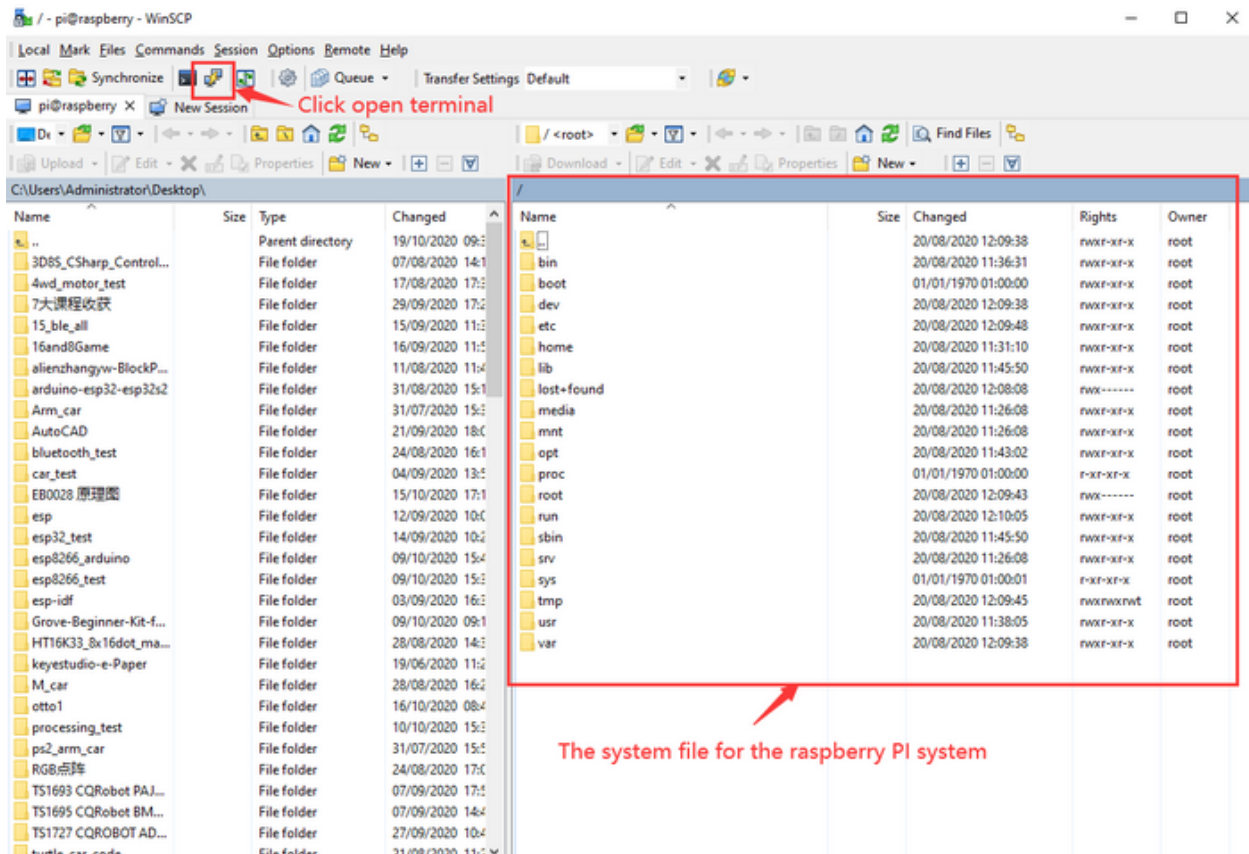
## Remote Login

Enter default user name, password and host name on WinSCP to log in.

Only a Raspberry Pi is connected in same network.

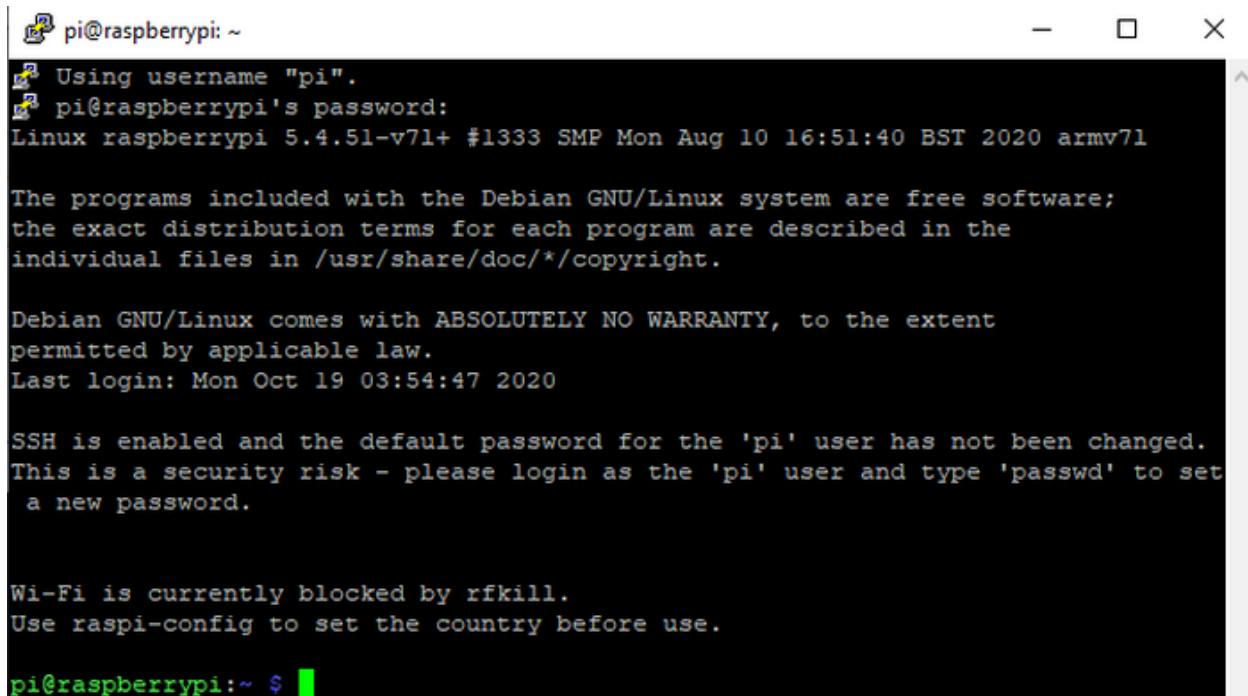


## Check ip and mac address



Click to open terminal input the password raspberry, and press "Enter" on keyboard.





```

pi@raspberrypi: ~
Using username "pi".
pi@raspberrypi's password:
Linux raspberrypi 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv7l

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

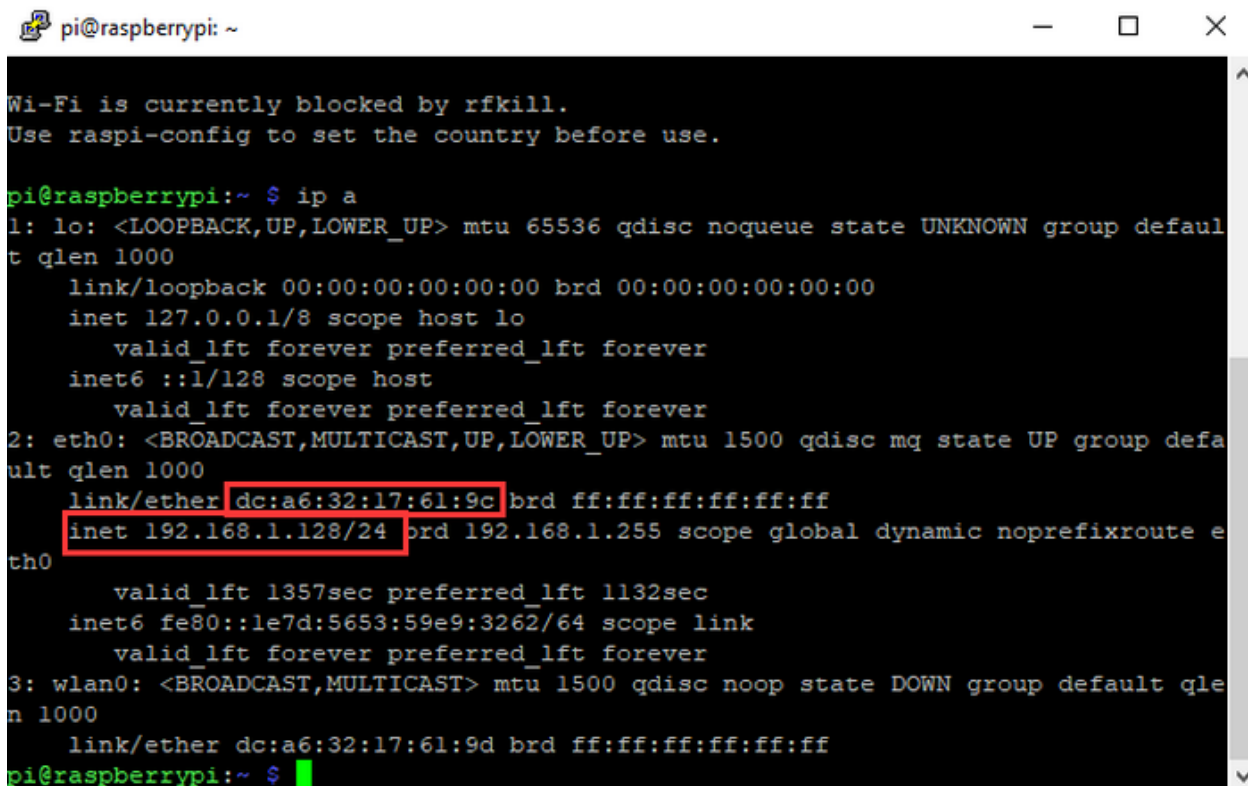
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

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: ~
Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~ $ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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 default qlen 1000
    link/ether dc:a6:32:17:61:9c brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.128/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 1357sec preferred_lft 1132sec
    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 qlen 1000
    link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff

pi@raspberrypi:~ $

```

From the above figure, mac address of this Raspberry Pi is `dc:a6:32:17:61:9c`, and ip address is `192.168.1.128` (use ip address to finish xrdp login)

Since mac address never changes, you could confirm ip via it.

**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 password

Input password in the terminal `sudo passwd root` to set password

Switch to root user

`su root`

Fix the configuration file of ip address

Firstly change ip address of the following configuration file

#New ip address `address 192.168.1.99`

Copy the above new address to terminal and press "Enter"

Configuration File

`echo -e '`

`auto eth0`

`iface eth0 inet static`

#Change IP address

`address 192.168.1.99`

`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 0`

`mtu 1492`

`'>/etc/network/interfaces.d/eth0`

As shown below:



```

pi@raspberrypi:~ $ su root
Password:
root@raspberrypi:/home/pi# echo -e '
> auto eth0
> iface eth0 inet static
>     #Change IP address
>     address 192.168.1.99
>     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 0
> mtu 1492
> '/etc/network/interfaces.d/eth0
root@raspberrypi:/home/pi#

```

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: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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 1492 qdisc mq state UP group default qlen 1000
    link/ether dc:a6:32:17:61:9c brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.99/24 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 qlen 1000
    link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff
pi@raspberrypi:~ $

```

### 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. Let's take example of Xrdp.

### Install Xrdp Service in the terminal

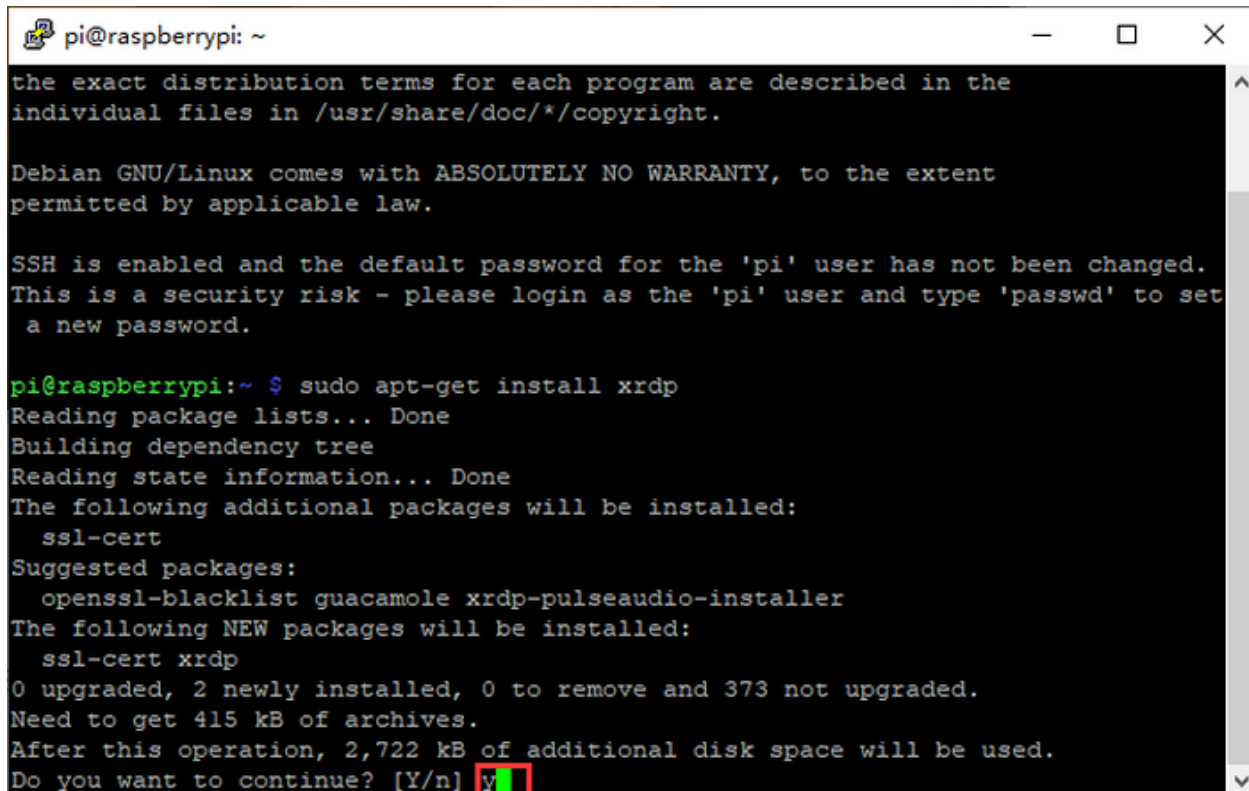
Install Command

Switch to Root User: `su root`

Install apt-get install xrdp

Enter y and press “Enter”

As shown below:

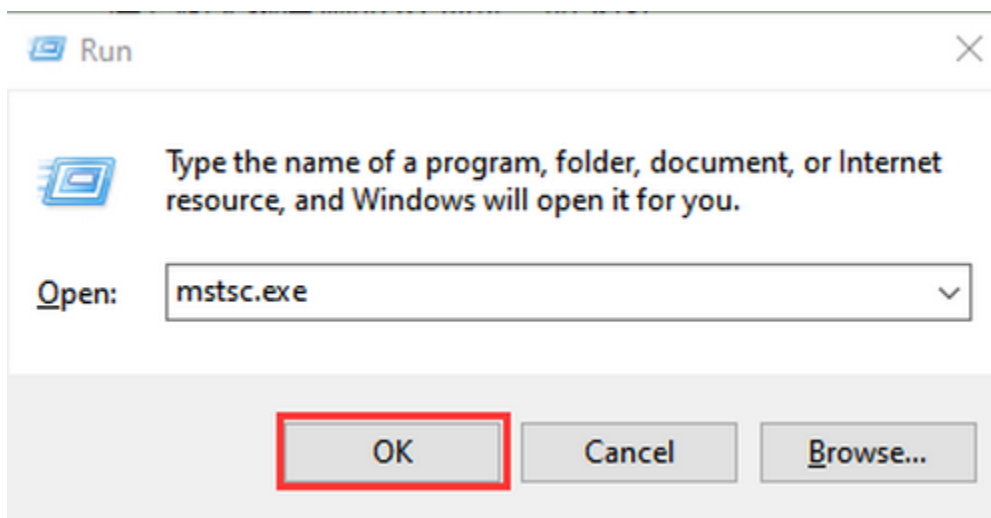


```
pi@raspberrypi: ~  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
  
SSH is enabled and the default password for the 'pi' user has not been changed.  
This is a security risk - please login as the 'pi' user and type 'passwd' to set  
a new password.  
  
pi@raspberrypi:~ $ sudo apt-get install xrdp  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  ssl-cert  
Suggested packages:  
  openssl-blacklist guacamole xrdp-pulseaudio-installer  
The following NEW packages will be installed:  
  ssl-cert xrdp  
0 upgraded, 2 newly installed, 0 to remove and 373 not upgraded.  
Need to get 415 kB of archives.  
After this operation, 2,722 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

**Open the remote desktop connection on Windows**

Press WIN+R on keyboard and enter mstsc.exe

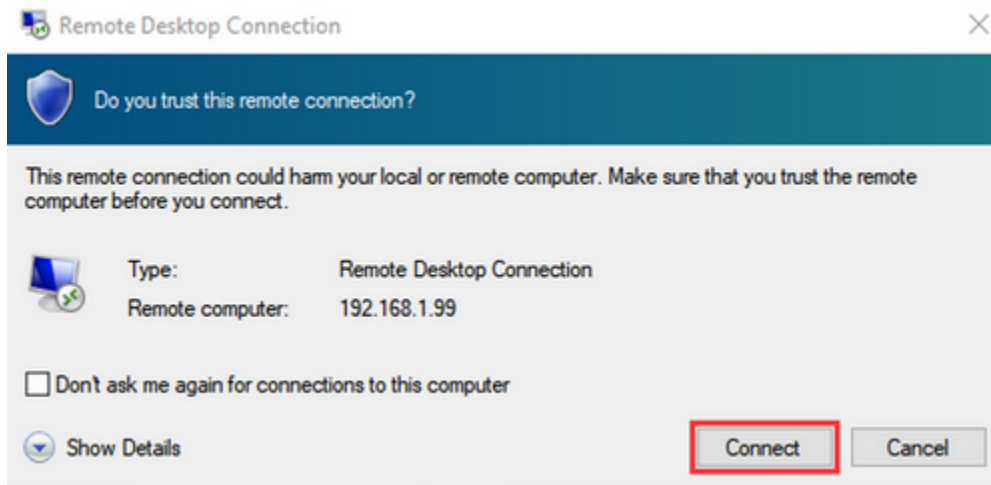
As shown below



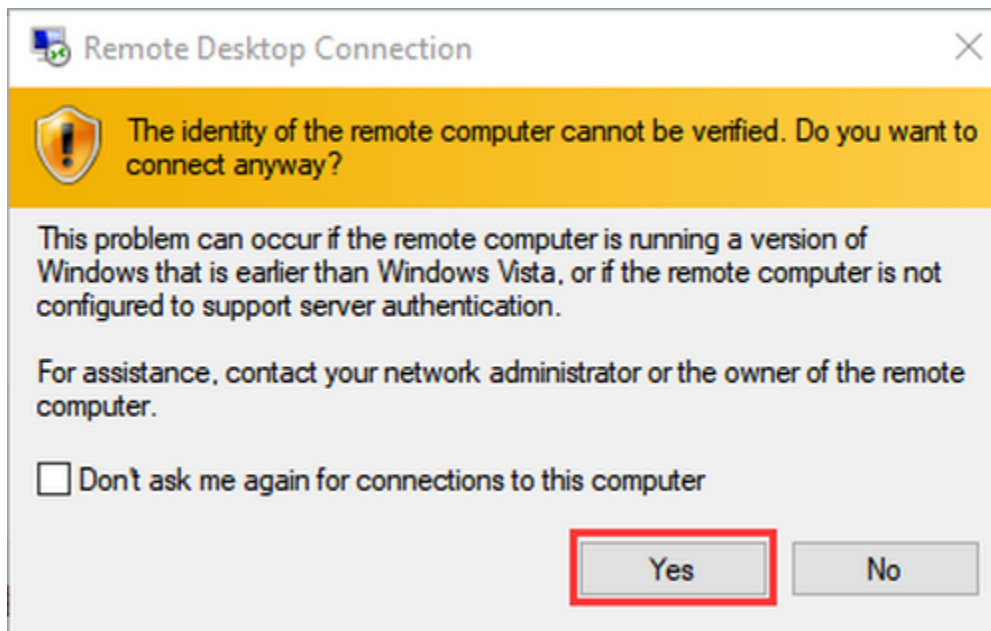
Input ip address of Raspberry Pi, as shown below.

Click “Connect” and tap “Connect”.

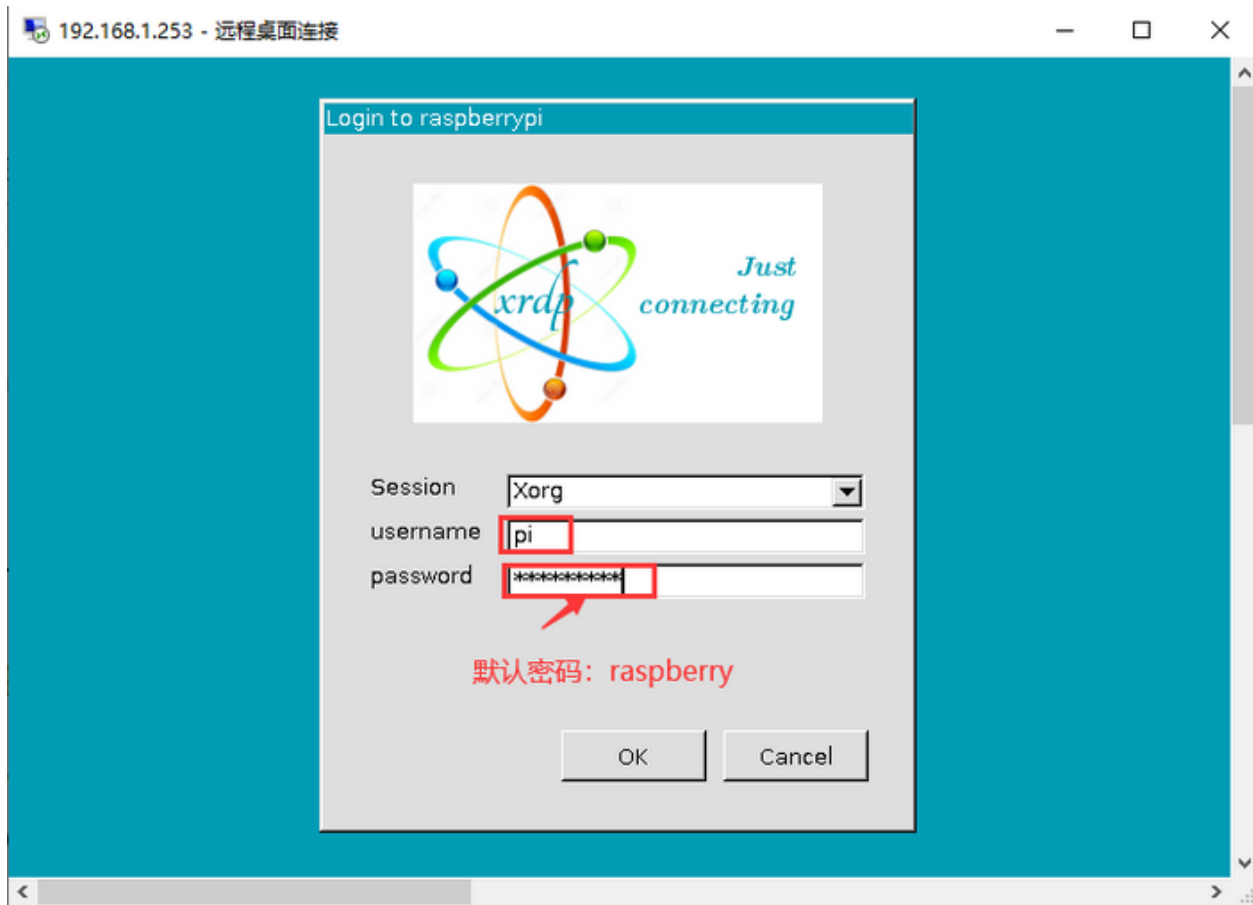
192.168.1.99 is ip address we use, you could change into yours ip address.



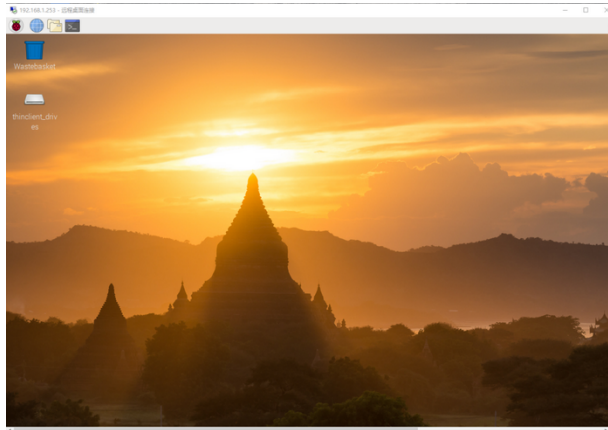
Click "Yes".



Input user name: pi, default password: raspberry, as shown below:



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.

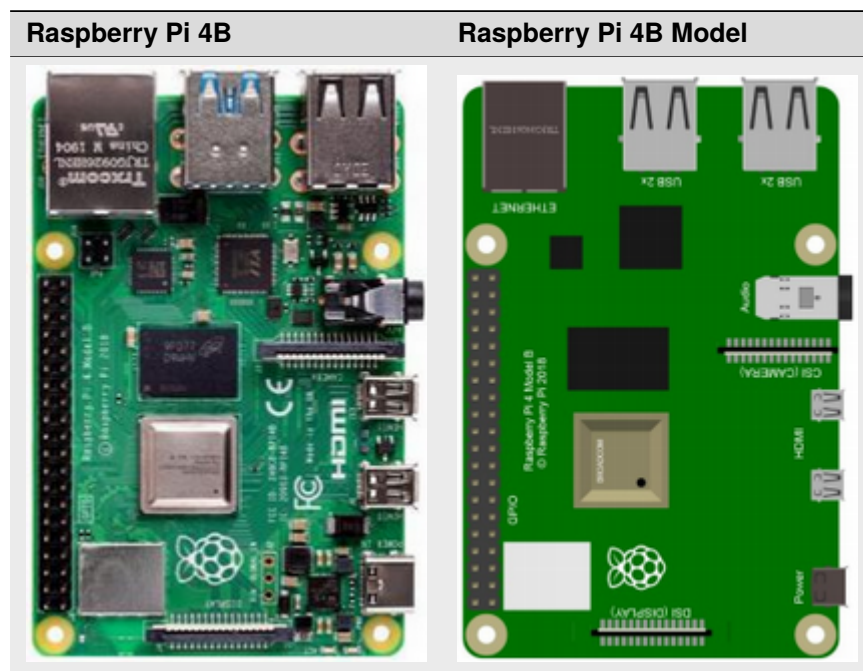
## 4.3 3. Preparations for C Language

C is a procedural programming language. It was initially developed by Dennis Ritchie in the year 1972. It was mainly developed as a system programming language to write an operating system. The main features of C language include low-level access to memory, a simple set of keywords, and clean style, these features make C language suitable for system programmings like an operating system or compiler development.

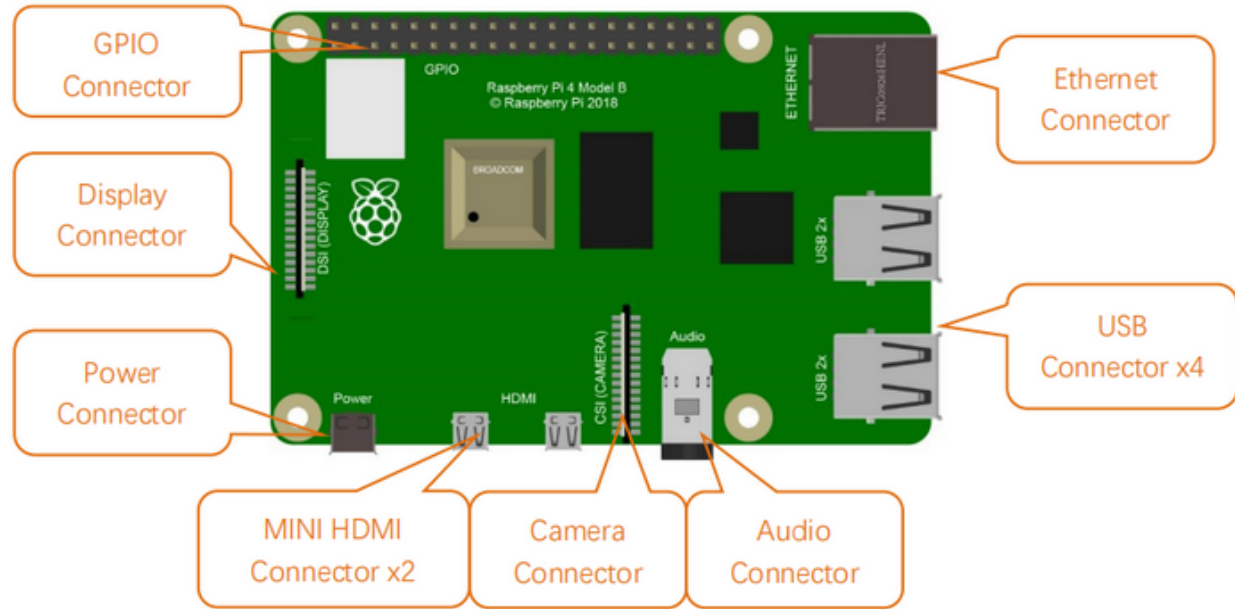
Next to control 40 pins of Raspberry Pi via C language

### 4.3.1 Hardware

#### Raspberry Pi 4B



#### Hardware Interfaces



#### 40-Pin GPIO Header Description

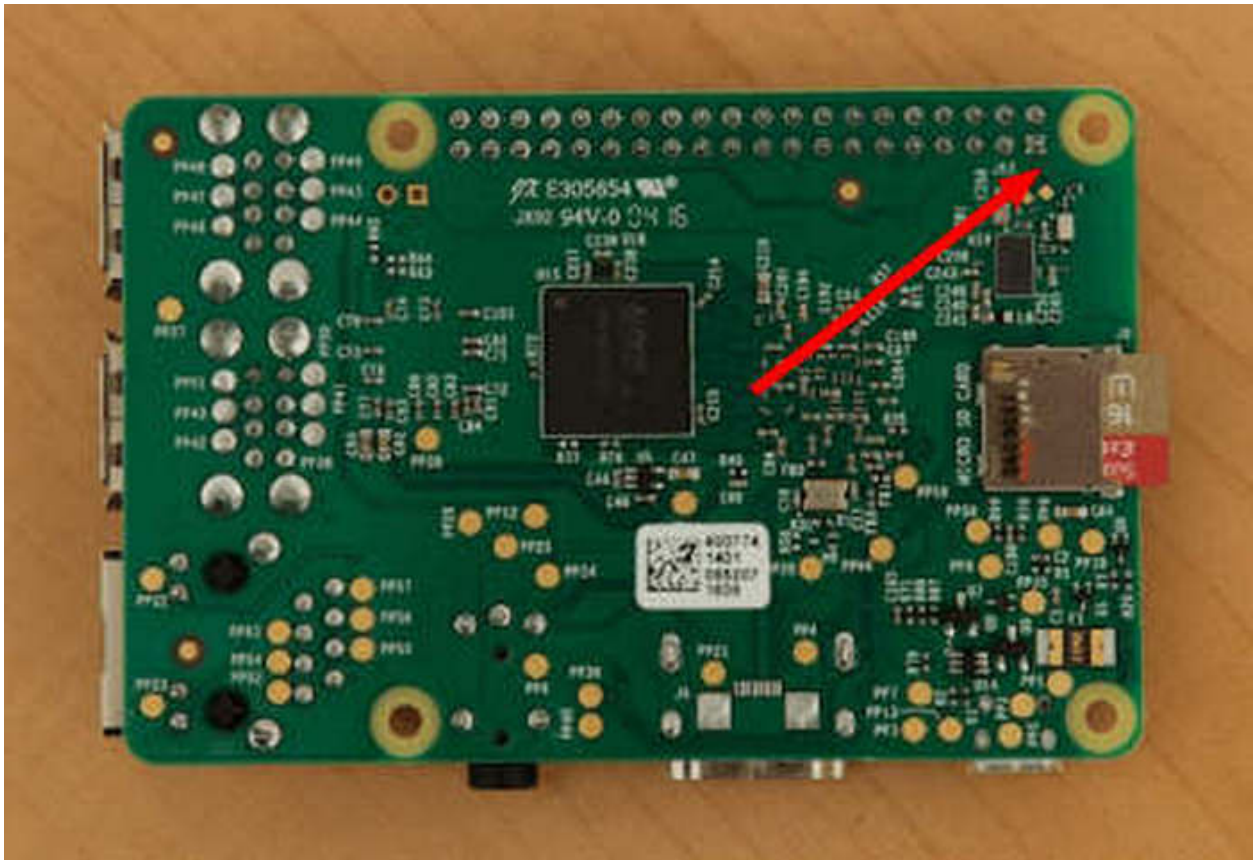
GPIO pins are divided into BCM GPIO number, physics number and WiringPi GPIO number.

We usually use WiringPi GPIO number when using C language and BCM GPIO and physics number are used to Python, as shown below:

In these lessons, we use C language, so WiringPi GPIO number is adopted.

Note: pin(3.3 V) on the left hand is square, but other pins are round. Turn Raspberry Pi over, there is a square GPIO on the back.(you could tell from pin(3.3V)).



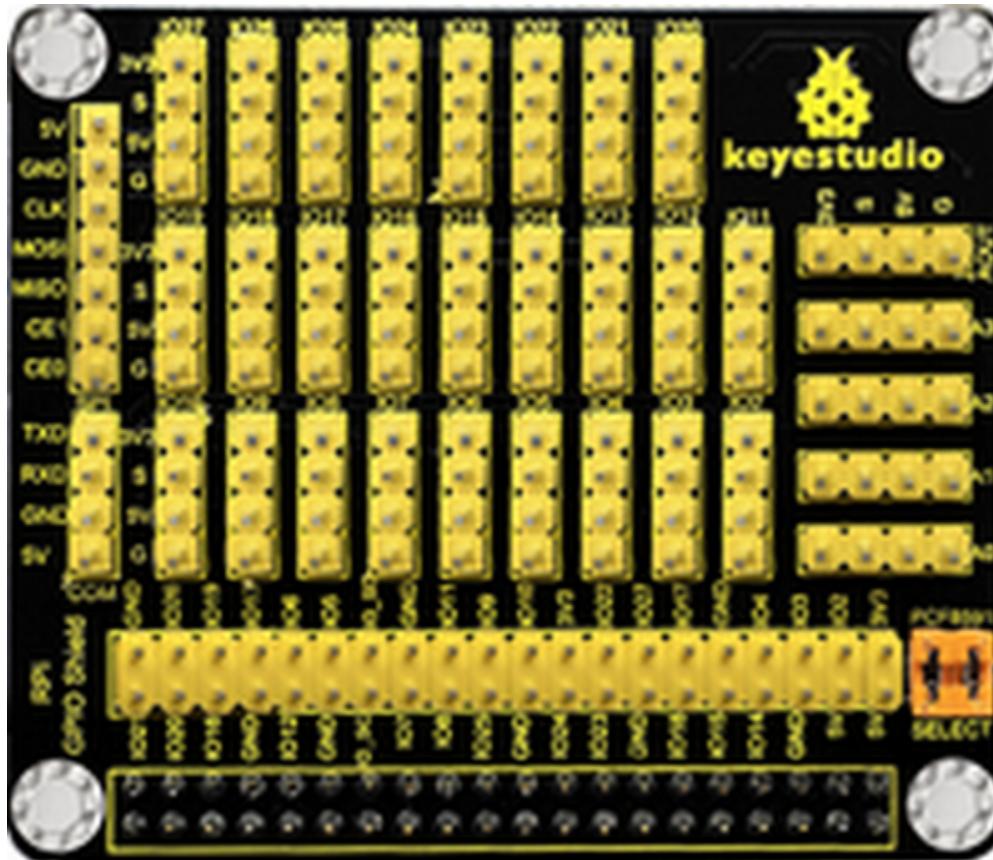


Note: the largest current of each pin on Raspberry Pi 4B is 16mA and the aggregate current of all pins is not less than 51mA.

### 4.3.2 GPIO Extension Board

This extension board is led out by 40-pin headers of Raspberry Pi for convenient connection.

Note: the silk mark is also printed according to BCM GPIO number.

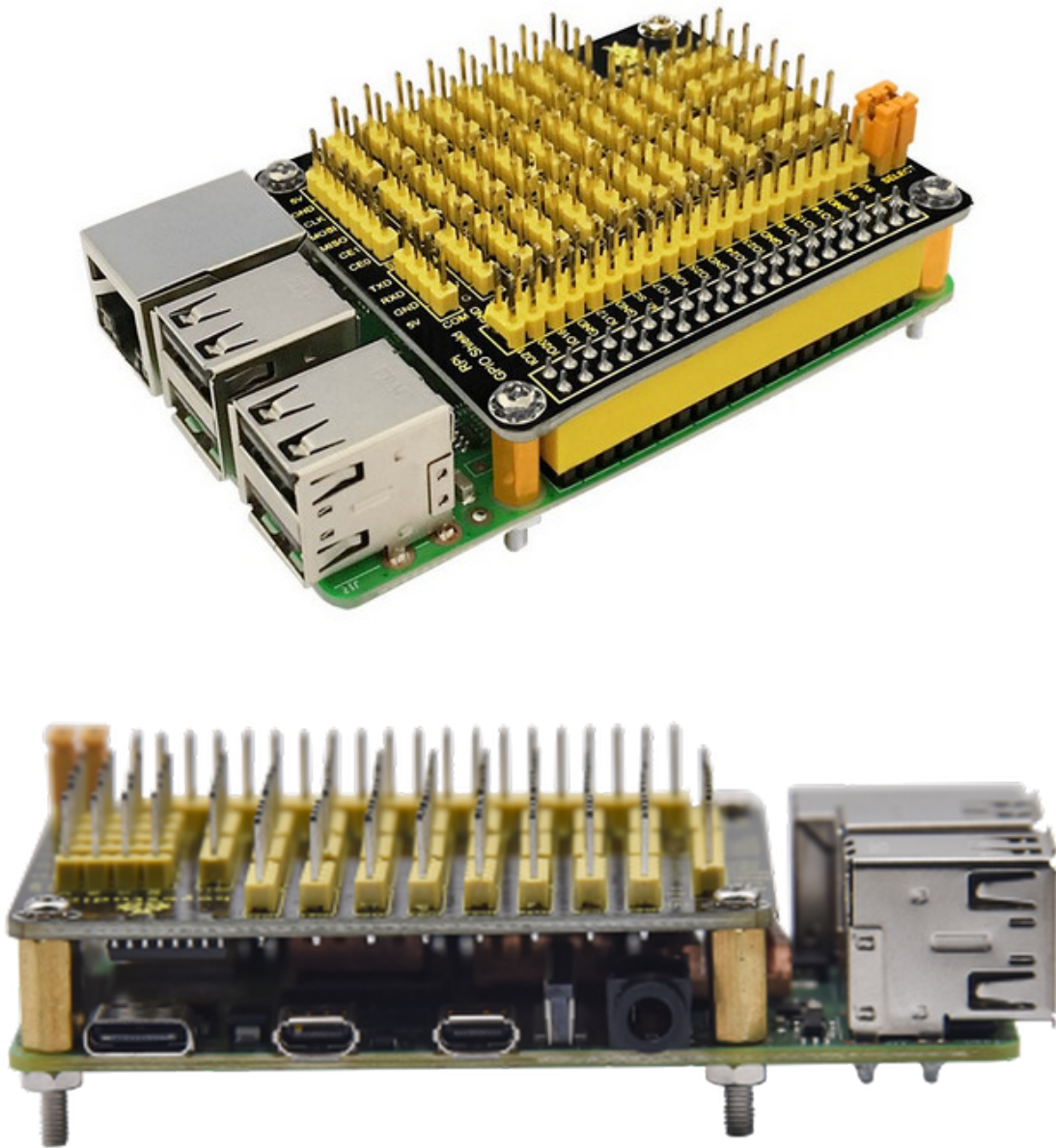


Since the Raspberry Pi itself does not have AD/DA function, an expansion board with this function is required when it is connected to external analog sensors. And the PCF8591 chip, welded behind RPI GPIO-PCF8591 shield , has four AD pins and one DA pin which can be connected to the Raspberry Pi via the I2C interface on the Pi.

There are two ways to fix Raspberry Pi with the RPI GPIO-PCF8591 shield available below:

Fix it with screws,nuts and pillar copper;





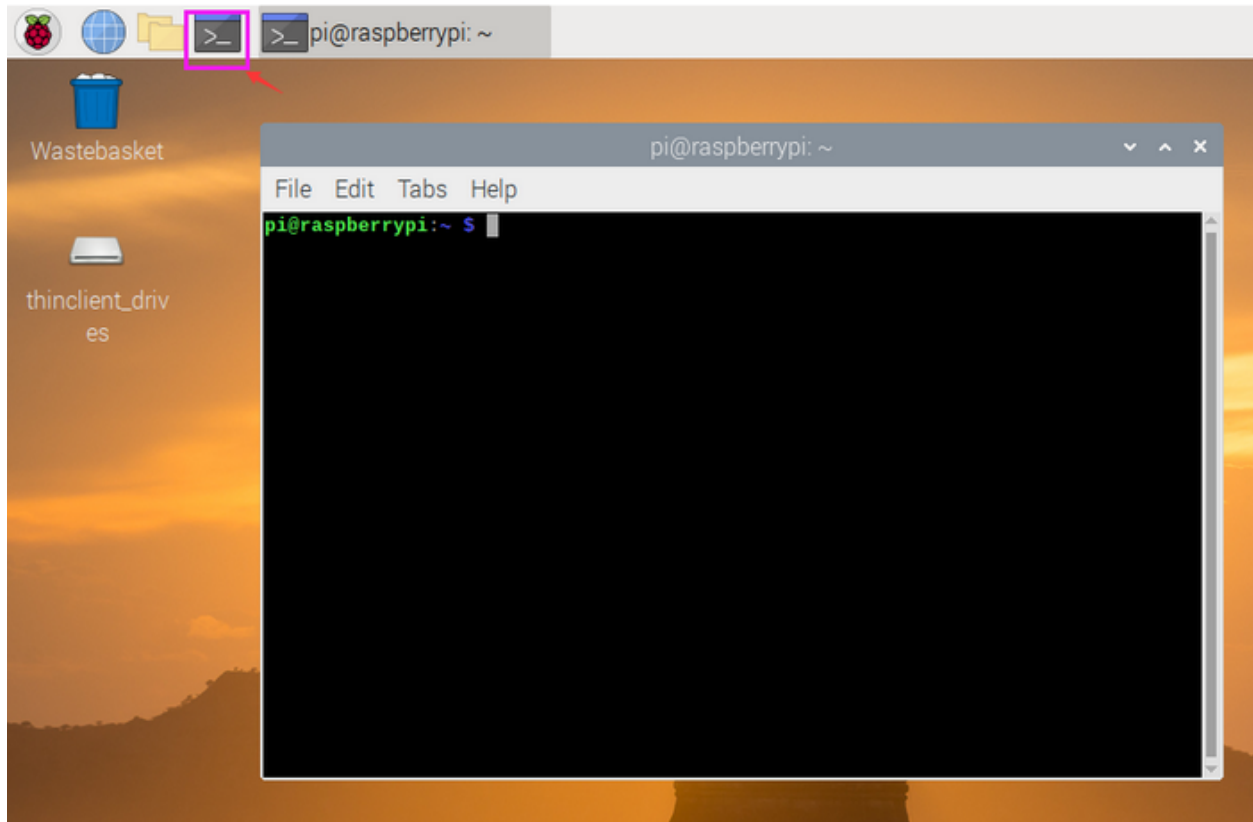
Fix without screws,nuts and pillar copper;



### 4.3.3 Install WiringPi GPIO Library

We will control IO ports of Raspberry Pi by WiringPi GPIO library, let's install WiringPi GPIO library.

Click the terminal icon of Raspberry Pi and open the terminal, as shown below:



Enter the following commands in the terminal and tap“Enter”

```
cd /tmp
```

```
wget https://project-downloads.drogon.net/wiringpi-latest.deb
```

```
sudo dpkg -i wiringpi-latest.deb
```

As shown below

```

pi@raspberrypi: /tmp
File Edit Tabs Help
pi@raspberrypi:~ $ cd /tmp
pi@raspberrypi:/tmp $
pi@raspberrypi:/tmp $ wget https://project-downloads.drogon.net/wiringpi-latest.deb
--2020-12-11 01:28:31-- https://project-downloads.drogon.net/wiringpi-latest.deb
Resolving project-downloads.drogon.net (project-downloads.drogon.net)... 188.246.205.22, 2a03:980
0:10:7b::2
Connecting to project-downloads.drogon.net (project-downloads.drogon.net)|188.246.205.22|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 52260 (51K) [application/x-debian-package]
Saving to: 'wiringpi-latest.deb'

wiringpi-latest.deb 100%[=====] 51.04K 78.8KB/s in 0.6s

2020-12-11 01:28:33 (78.8 KB/s) - 'wiringpi-latest.deb' saved [52260/52260]

pi@raspberrypi:/tmp $
pi@raspberrypi:/tmp $ sudo dpkg -i wiringpi-latest.deb
(Reading database ... 154938 files and directories currently installed.)
Preparing to unpack wiringpi-latest.deb ...
Unpacking wiringpi (2.52) over (2.52) ...
Setting up wiringpi (2.52) ...
Processing triggers for man-db (2.8.5-2) ...
pi@raspberrypi:/tmp $

```

Check the version of WiringPi GPIO library and corresponding definition of 40-pin headers

Input the following commands and press“Enter”

gpio -v

gpio readall

The version of WiringPi GPIO library is 2.52

```

pi@raspberrypi:/tmp $ gpio -v
gpio version: 2.52
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
Type: Pi 4B, Revision: 01, Memory: 2048MB, Maker: Sony
* Device tree is enabled.
*--> Raspberry Pi 4 Model B Rev 1.1
* This Raspberry Pi supports user-level GPIO access.

```

Pins definition of WiringPi GPIO Library

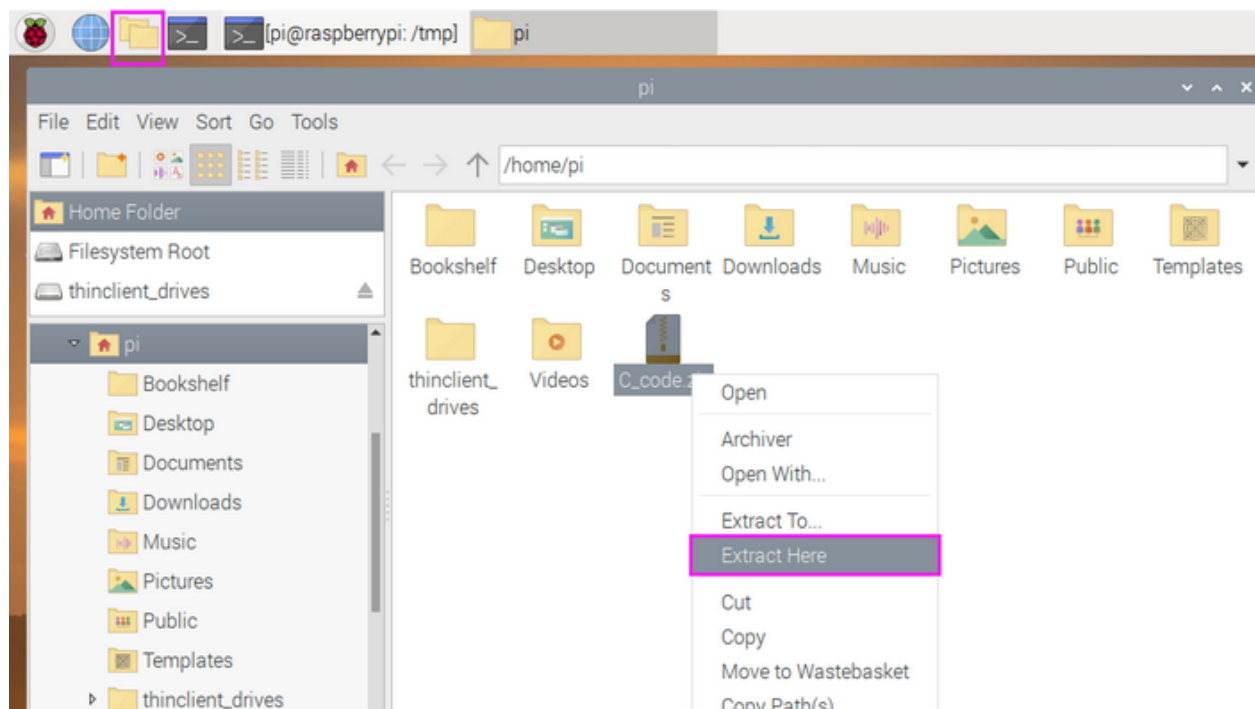
(note: the silk mark of our extension board is defined by pins of BCM GPIO as well)

```
pi@raspberrypi:/tmp $ gpio readAll
```

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1	2		5v		
2	8	SDA.1	ALT0	1	3	4		5v		
3	9	SCL.1	ALT0	1	5	6		0v		
4	7	GPIO. 7	IN	1	7	8	0	IN	15	14
		0v			9	10	0	IN	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	1	18
27	2	GPIO. 2	IN	0	13	14		0v		
22	3	GPIO. 3	IN	0	15	16	0	IN	4	23
		3.3v			17	18	0	IN	5	24
10	12	MOSI	ALT0	0	19	20		0v		
9	13	MISO	ALT0	0	21	22	1	OUT	6	25
11	14	SCLK	ALT0	0	23	24	1	OUT	10	8
		0v			25	26	1	OUT	11	7
0	30	SDA.0	IN	1	27	28	1	IN	31	1
5	21	GPIO.21	IN	1	29	30		0v		
6	22	GPIO.22	IN	1	31	32	0	IN	26	12
13	23	GPIO.23	IN	0	33	34		0v		
19	24	GPIO.24	IN	0	35	36	0	IN	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	28	20
		0v			39	40	0	IN	29	21

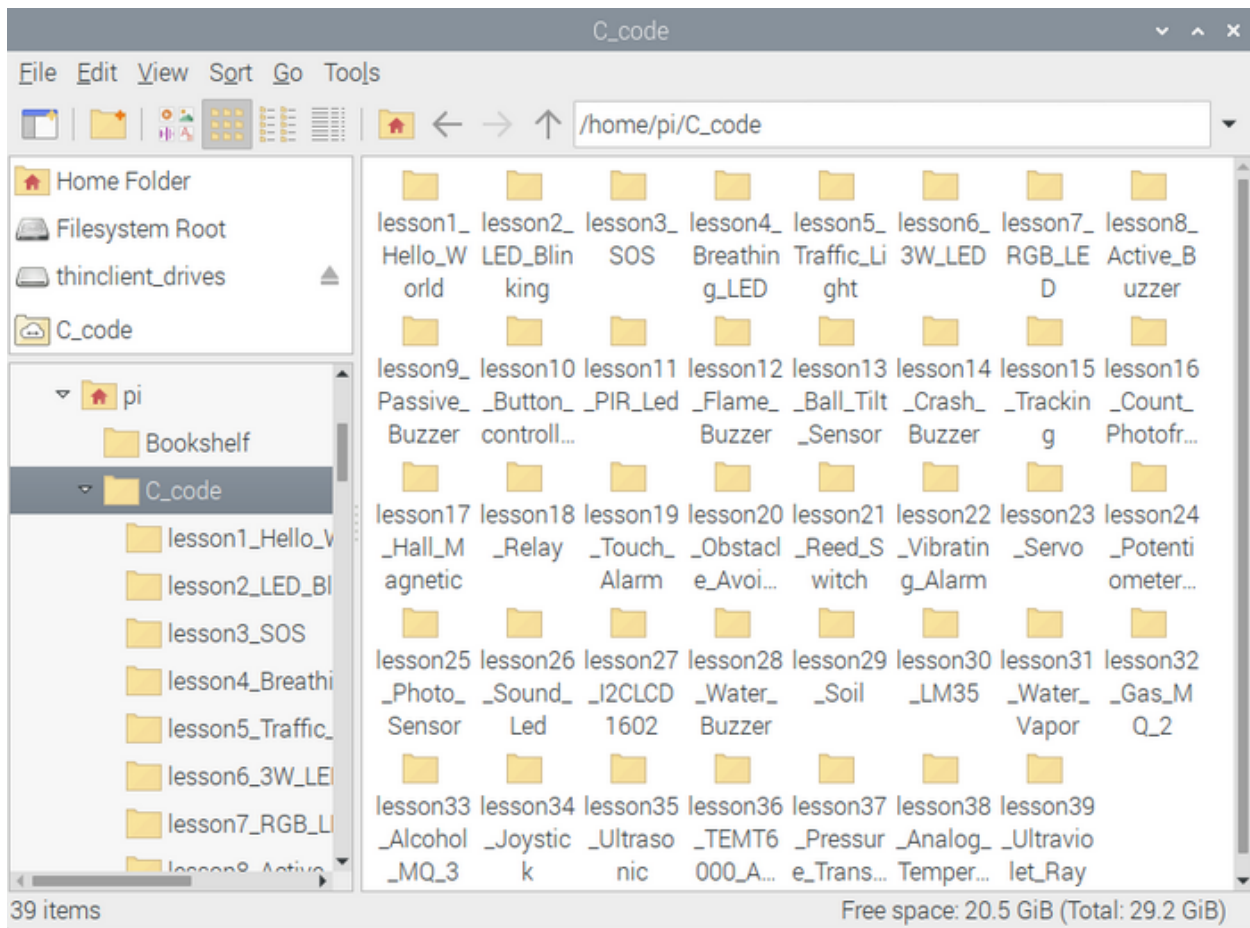
#### 4.3.4 Run Example Code1

Copy the C\_code.zip we provide to pi folder, and extract the example code from zip file, as shown below:



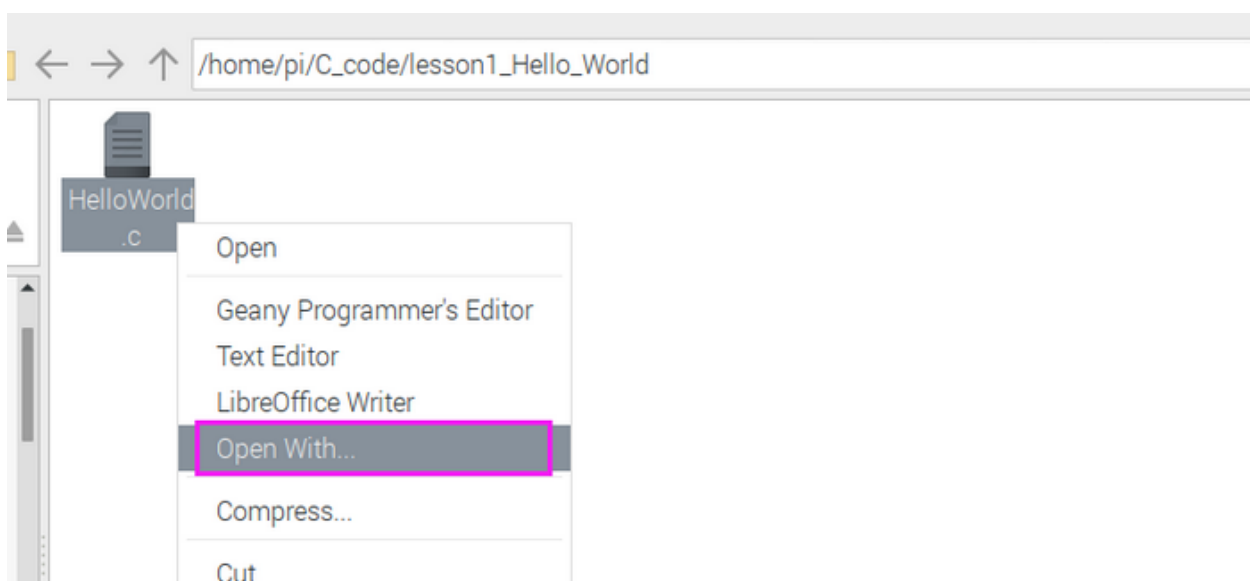


Double-click C\_code folder to look through example code, as shown below:

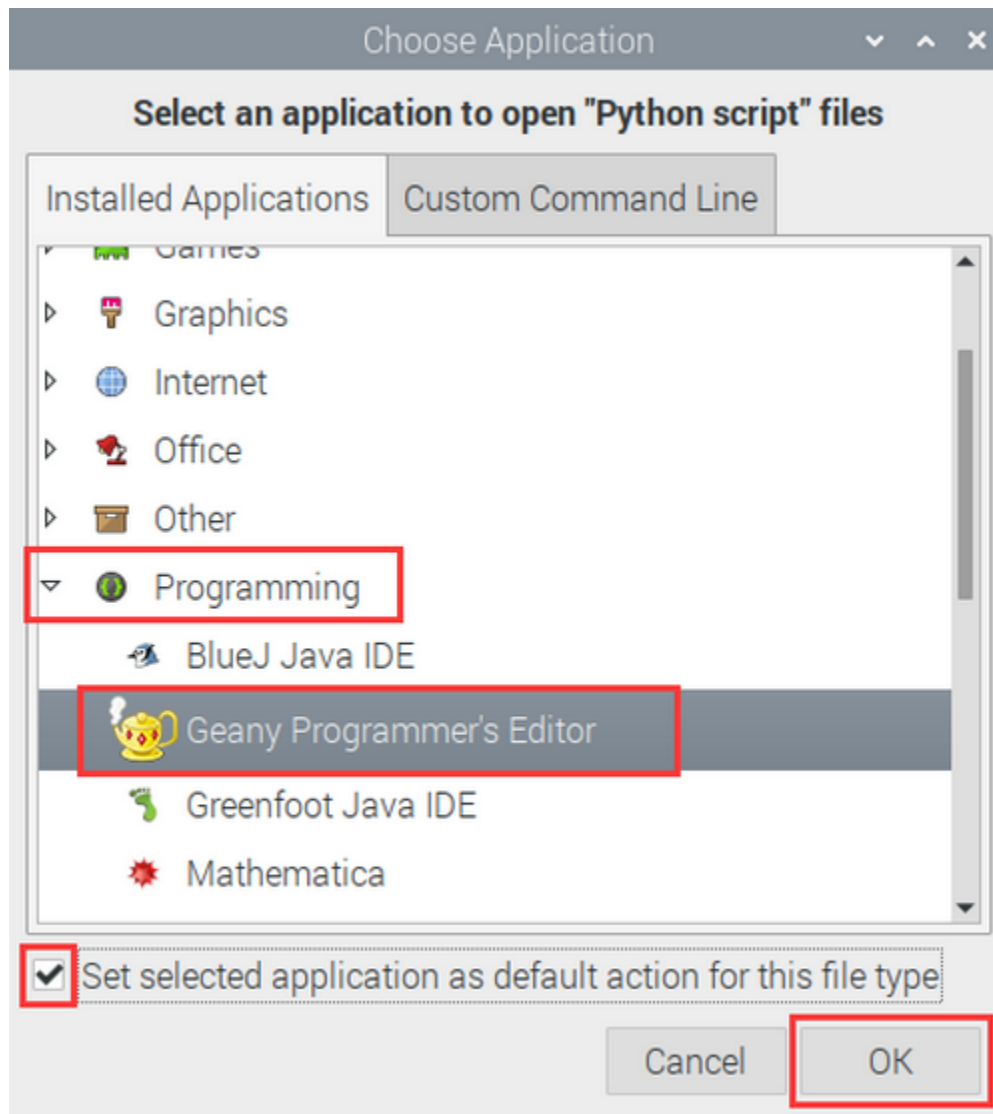


Set the default editor of file with .c

Enter lesson1\_Hello\_World and right-click Open with...



Click Programming to select Geany Programmer's Editor :

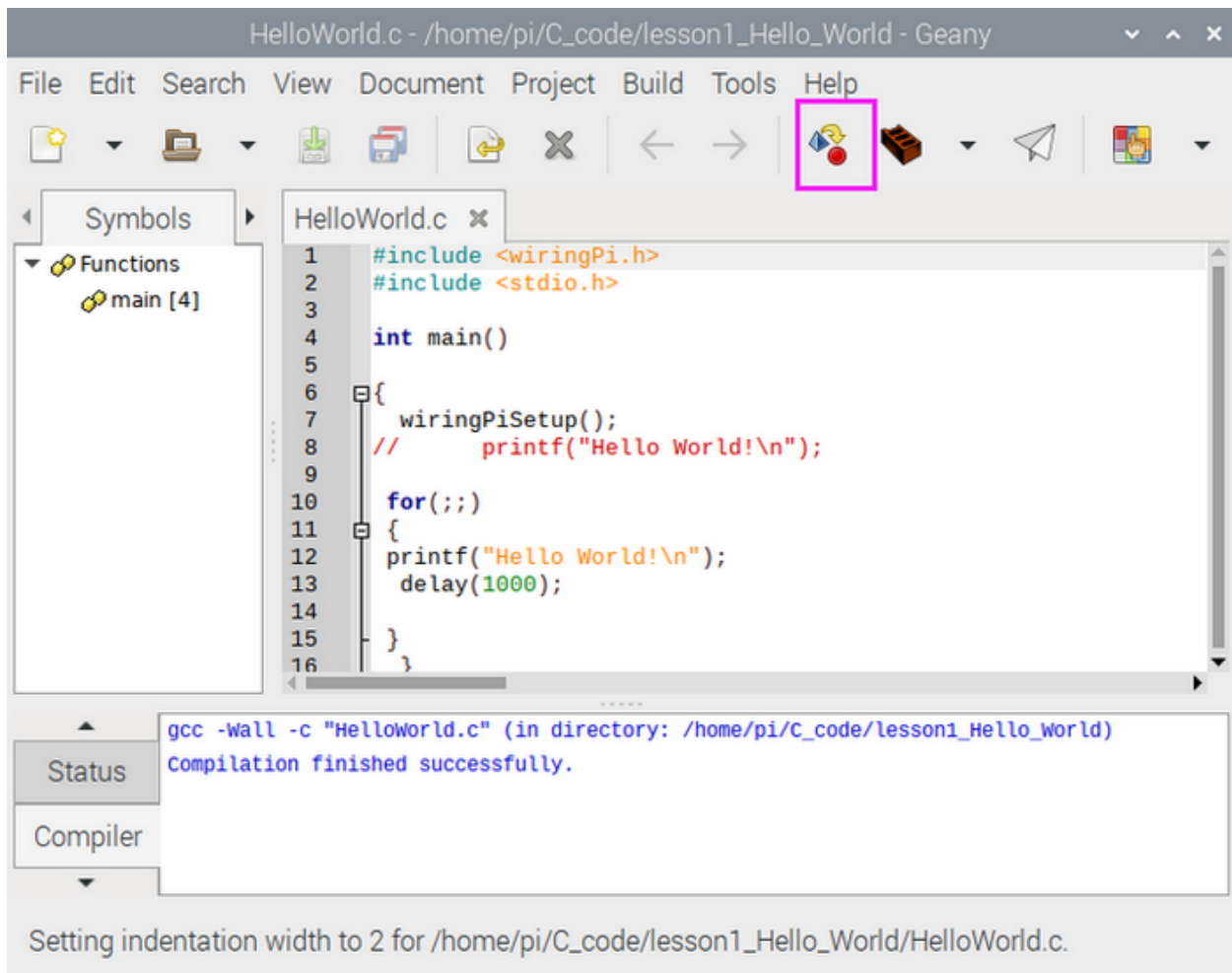


Then, we can directly open file by double-click Geany Programmer's Editor

#### The Use of Geany Programmer's Editor



Open "HelloWorld.c" via it, click to compile code to check grammar errors.



### Run Example Code

Terminal enters the corresponding coursesfor example, enter lesson1\_Hello\_World.

Enter the route with terminal command cd

cd /home/pi/C\_code/lesson1\_Hello\_World

Input the compilation command

gcc HelloWorld.c -o HelloWorld -lwiringPi

Input ls to check file of the current folder:

The compilation file:HelloWorld

Run a compilation file: HelloWorld

Input commandsudo ./HelloWorld(as shown below)

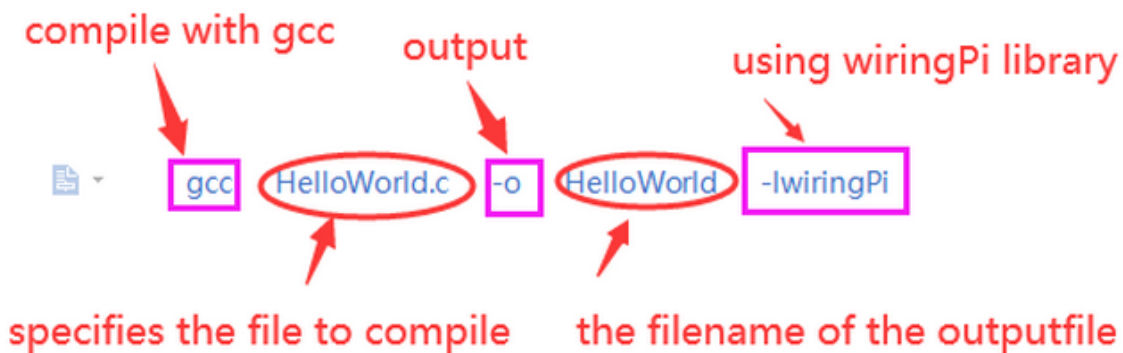


```

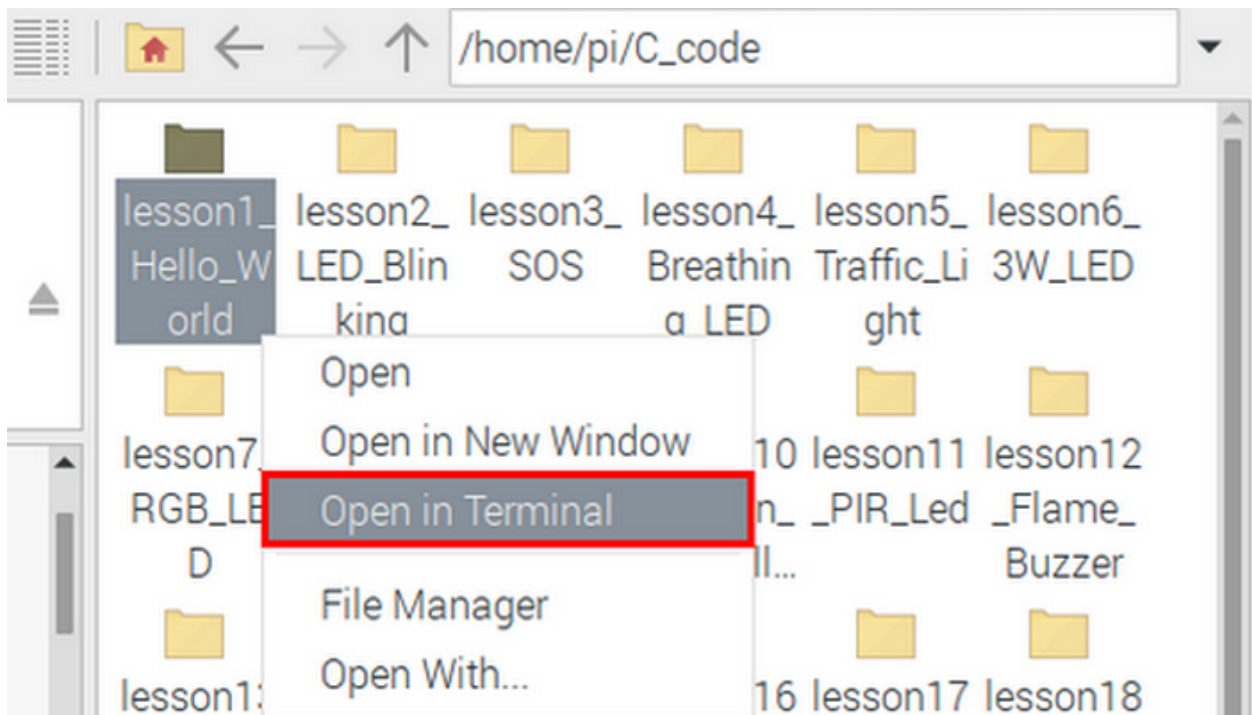
pi@raspberrypi: ~/C_code/lesson1_Hello_World
File Edit Tabs Help
pi@raspberrypi:~ $ cd /home/pi/C_code/lesson1_Hello_World
pi@raspberrypi:~/C_code/lesson1_Hello_World $ gcc HelloWorld.c -o HelloWorld -lwiringPi
pi@raspberrypi:~/C_code/lesson1_Hello_World $ ls
HelloWorld HelloWorld.c HelloWorld.o
pi@raspberrypi:~/C_code/lesson1_Hello_World $ sudo ./HelloWorld
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!

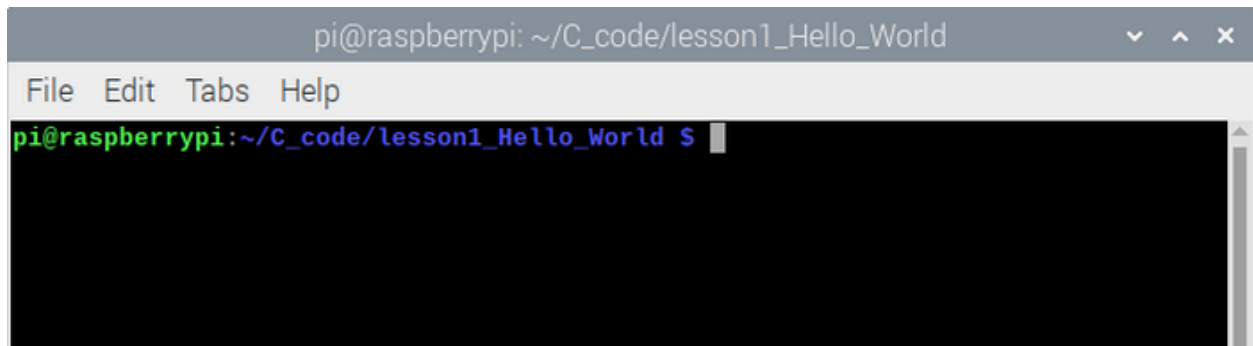
```

Command Explanation



You could select the folder and right-click to choose Open in the terminal as shown below, if you feel it complicated to enter the route.





## 4.4 4. Projects

**Note:** G, - and GND marked on sensors and modules are so-called positive, which are connected to GND of RPI GPIO-PCF8591 Shield ; V and VCC are known as positive, which are interfaced 3V3 or 5V on extension board.

### 4.4.1 Project 1Hello World

(1)Run Example Code

Input the following commands in the terminal, and press“Enter”:

```
cd /home/pi/C_code/lesson1_Hello_World
```

```
gcc HelloWorld.c -o HelloWorld -lwiringPi
```

```
sudo ./HelloWorld
```

(2)Test Results

Terminal prints Hello World ! , as shown below:

```
pi@raspberrypi:~ $ cd /home/pi/C_code/lesson1_Hello_World
pi@raspberrypi:~/C_code/lesson1_Hello_World $ gcc HelloWorld.c -o HelloWorld -lwiringPi
pi@raspberrypi:~/C_code/lesson1_Hello_World $ sudo ./HelloWorld
Hello World!
Hello World!
Hello World!
Hello World!
```

(3)Example Code:

```
#include <wiringPi.h> //wiringPi GPIO library
#include <stdio.h> //standard input & output library

int main() //Main function, the entry of the program
{
    wiringPiSetup(); //Initializes the wiringPi GPIO library
    while(1) //An infinite loop
    {
```

(continues on next page)

(continued from previous page)

```

printf("Hello World!\n"); //\n is a newline print
delay(1000); //delay 1000ms
}
}

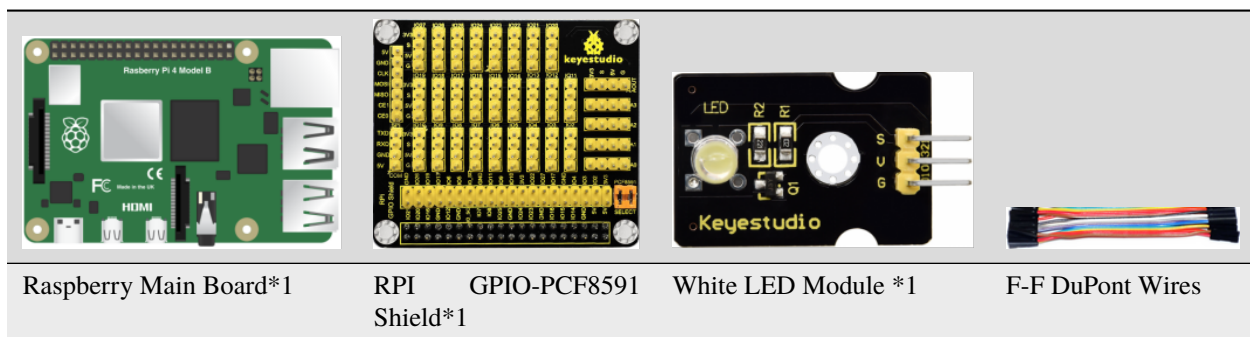
```

## 4.4.2 Project 2LED Blinks

### (1)Description

Let's start from a rather basic and simple experiment—LED Blinks.

### (2)Components Needed:



### (3)Knowledge about Component :

**The white LED module** is a commonly used LED module. It is a F5 LED with white appearance and white light display. During experiments, when the G and V on the module are powered up and the signal end S is at high level, the white LED is on while when the S is at low level, the LED is off.

The module is compatible with various microcontroller control boards, such as arduino series microcontrollers. The white LED module can open and close the S8050 NPN transistor by controlling the high and low level of the IO port of the single-chip microcomputer so as to turn on and off the LED.

### (4)Connection Diagram

White LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



### (5)Working Principle

According to the diagram above we can find out that the positive pole(V) is connected to 5V, negative pole(G) to GND and signal terminal(S) to the pin of GPIO18. When GPIO18 outputs high level, LED is on; when it outputs low level, LED is off.

### (6)Run Example Code

Input the following commands in the terminal and press“Enter”

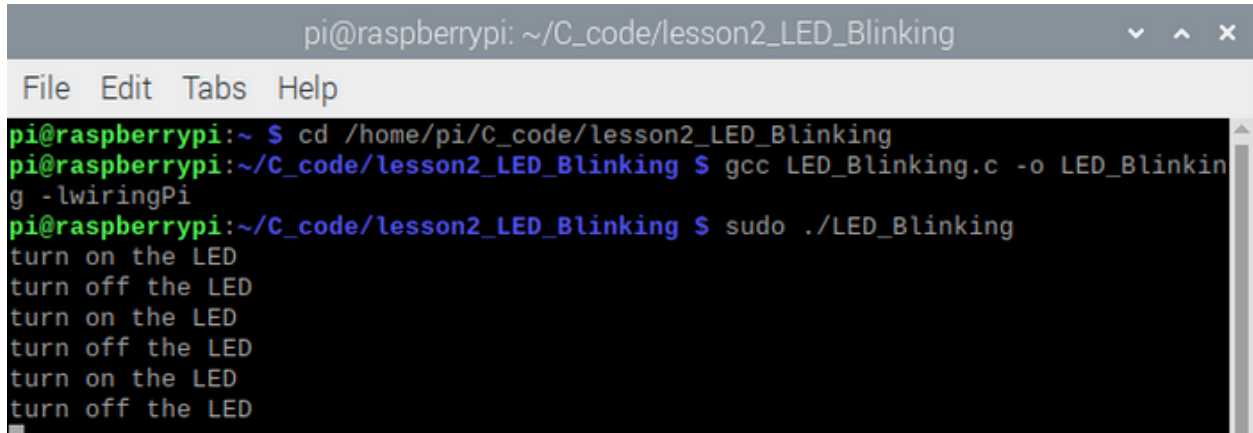
```
cd /home/pi/C_code/lesson2_LED_Blinking
```

```
gcc LED_Blinking.c -o LED_Blinking -lwiringPi
```

```
sudo ./LED_Blinking
```

(7)Test Results

Terminal prints and LED flashes.



Note: Press Ctrl + C on keyboard and exit code running.

(8)Example Code

```
#include <wiringPi.h>
#include <stdio.h>

#define ledPin 1 //define led pin, BCM GPIO 18

int main()
{
    wiringPiSetup(); //Initialize wiringPi

    pinMode(ledPin,OUTPUT); //set the ledPin OUTPUT mode

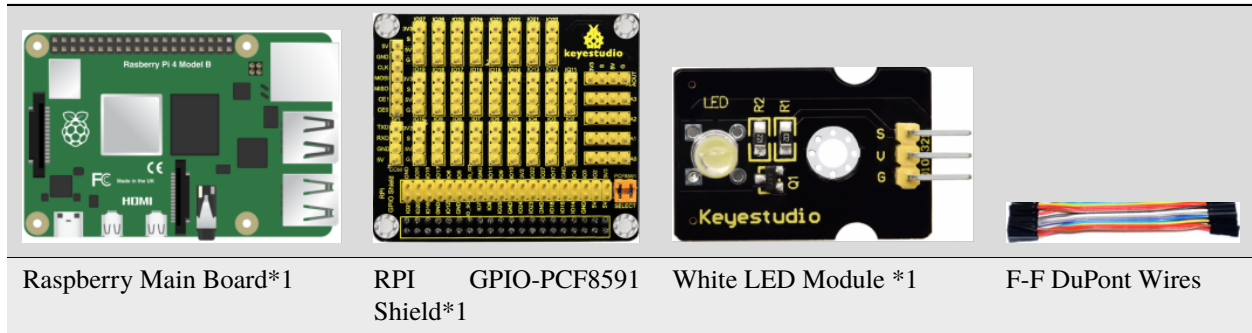
    while(1)
    {
        digitalWrite(ledPin,HIGH); //turn on led
        printf("turn on the LED\n");
        delay(500); //delay 500ms
        digitalWrite(ledPin,LOW); //turn off led
        printf("turn off the LED\n");
        delay(500);
    }
}
```

### 4.4.3 Project 3SOS Light

#### (1)Description

S.O.S is a Morse code distress signal , used internationally, that was originally established for maritime use. We will present it with flashing LED.

#### (2)Components Needed:



#### (3)Connection Diagram

White LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### (4)Run Example Code

Input the following commands and press “Enter”

```
cd /home/pi/C_code/lesson3_SOS
```

```
gcc SOS.c -o SOS -lwiringPi
```

```
sudo ./SOS
```

#### (5)Test Results

LED flashes quickly for three times, three times slowly and quickly three times, the terminal prints ... \_ \_ \_ ...

```

pi@raspberrypi:~/C_code/lesson3_S0S $ sudo ./S0S^C
pi@raspberrypi:~/C_code/lesson3_S0S $ cd /home/pi/C_code/lesson3_S0S
pi@raspberrypi:~/C_code/lesson3_S0S $
pi@raspberrypi:~/C_code/lesson3_S0S $ gcc S0S.c -o S0S -lwiringPi
pi@raspberrypi:~/C_code/lesson3_S0S $
pi@raspberrypi:~/C_code/lesson3_S0S $ sudo ./S0S

```

Note: Press Ctrl + C on keyboard and exit code running.

(6)Example Code:

```

#include <wiringPi.h>
#include <stdio.h> //The stdio.h header file defines three variable types,
                  //some macros, and various functions to perform input and output.

#define ledPin 1 //define led pin
int i1,i2,i3;

int main()
{
    wiringPiSetup(); //Initialize wiringPi

    pinMode(ledPin,OUTPUT); //set the ledPin OUTPUT mode

    while(1)
    {
        while(i1<3)
        {
            digitalWrite(ledPin,HIGH); //turn on led
            delay(100); //delay 100ms
            digitalWrite(ledPin,LOW); //turn off led
            delay(100);
            i1 = i1 + 1;
            printf(".\n");
        }
        while(i2<3)
        {
            digitalWrite(ledPin,HIGH); //turn on led
            delay(1000); //delay 1000ms
            digitalWrite(ledPin,LOW); //turn off led
            delay(1000);
            i2 = i2 + 1;
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

        printf("-\n");
    }
    while(i3<3)
    {
        digitalWrite(ledPin,HIGH); //turn on led
        delay(100);                //delay 100ms
        digitalWrite(ledPin,LOW);  //turn off led
        delay(100);
        i3 = i3 + 1;
        printf(".\n");
    }
    //clean
    i1 = 0;
    i2 = 0;
    i3 = 0;
    printf(" \n");
    delay(500);
}
}

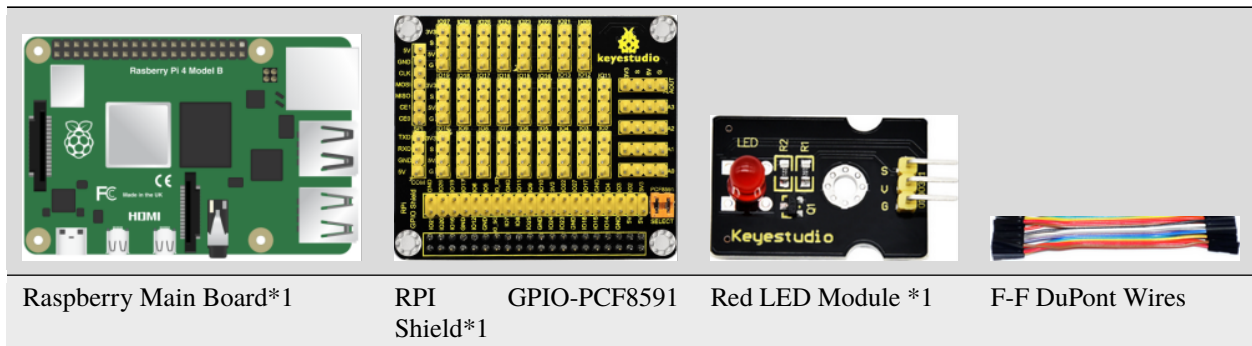
```

#### 4.4.4 Project 4Breathing LED

##### (1)Description

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.

##### (2)Components Needed



##### (3)Working Principle

We use the PWM output of GPIO, PWM outputs analog signals and output value is 0~100 which is equivalent to output voltage 0~3.3V from GPIO port.

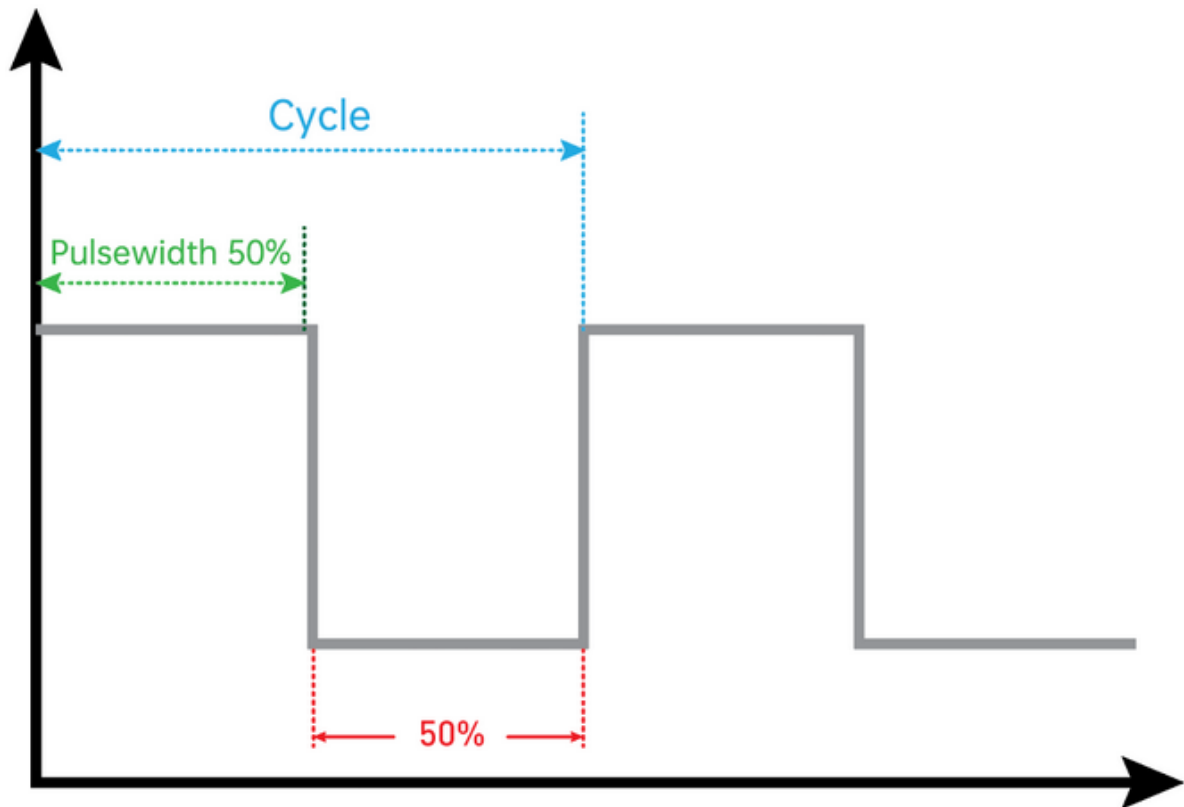
According to Ohm’s law:  $U/R = I$ , the resistance is 220, and the value of voltage  $U$  changes, so does the value of current  $I$ , which can control the brightness of the LED lamp.

PWM (Pulse Width Modulation) is the control of the analog circuit through the digital output of microcomputer and a method that makes digital coding on analog signal levels.

It sends square waves with certain frequency through digital pins, that is, high level and low level output alternately for a period of time. Total time of each group high and low level is fixed, which is called cycle.

The time of high level output is pulse width whose percentage is called Duty Cycle. The longer that high level lasts, the larger the duty cycle of analog signals is, and the corresponding voltage as well.

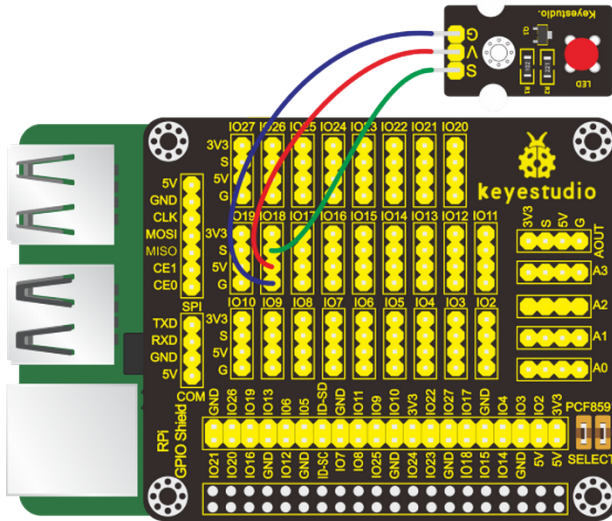
Below chart is pulse width 50%, then the output voltage is  $3.3 \times 50\% = 1.65\text{V}$  and the brightness of LED is medium.



(4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G





#### (5) Run Example Code 1

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson4_Breathing_LED
```

```
gcc Breathing_LED1.c -o Breathing_LED1 -lwiringPi
```

```
sudo ./Breathing_LED1
```

#### (6) Test Result 1

LED gradually brightens then darkens and changes in a loop.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code 1

```
#include <stdio.h>
#include <wiringPi.h>

#define LED 1 //define led pin

int main(void)
{
    int bright;
    printf("Raspberry Pi wiringPi PWM test program\n");
    wiringPiSetup(); //Initialize wiringPi
    pinMode(LED,PWM_OUTPUT); //set the ledPin OUTPUT mode

    while(1)
    {
        for (bright = 0; bright < 1024; ++bright) // pwm 0~1024
        {
            pwmWrite(LED,bright);
            printf("bright:%d\n",bright); // %d is the integer output, bright is the
            ↪ variable to output
            delay(10);
        }
        for (bright = 1023; bright >= 0; --bright)
```

(continues on next page)

(continued from previous page)

```
    {  
        pwmWrite(LED,bright);  
        printf("bright:%d\n",bright);  
        delay(10);  
    }  
}  
return 0;  
}
```

#### (8)Run Example Code 2

Software simulates PWM output:

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson4_Breathing_LED
```

```
gcc Breathing_LED2.c -o Breathing_LED2 -lwiringPi
```

```
sudo ./Breathing_LED2
```

#### (9)Test Results2

LED gradually brightens then darkens and repeats this pattern.

Note: Press Ctrl + C on keyboard and exit code running

#### (10)Example Code2

```
#include <stdio.h>  
#include <wiringPi.h>  
#include <softPwm.h> //Software PWM library  
  
#define LED 1  
  
int main(void)  
{  
    int i = 0;  
    wiringPiSetup(); //Initialize wiringPi  
    softPwmCreate(LED, 0, 100); //Create pin LED as the PWM output(0~100)  
    while (1)  
    {  
        for(i=0; i<100; i++)  
        {  
            softPwmWrite(LED, i); //pwm write  
            delay(20);  
            printf("PWM = %d\n",i);  
        }  
        for(i=99; i>0; i--)  
        {  
            softPwmWrite(LED, i);  
            delay(20);  
            printf("PWM = %d\n",i);  
        }  
    }  
    return 0;  
}
```

## 4.4.5 Project 5Traffic Lights

### (1)Description

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

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

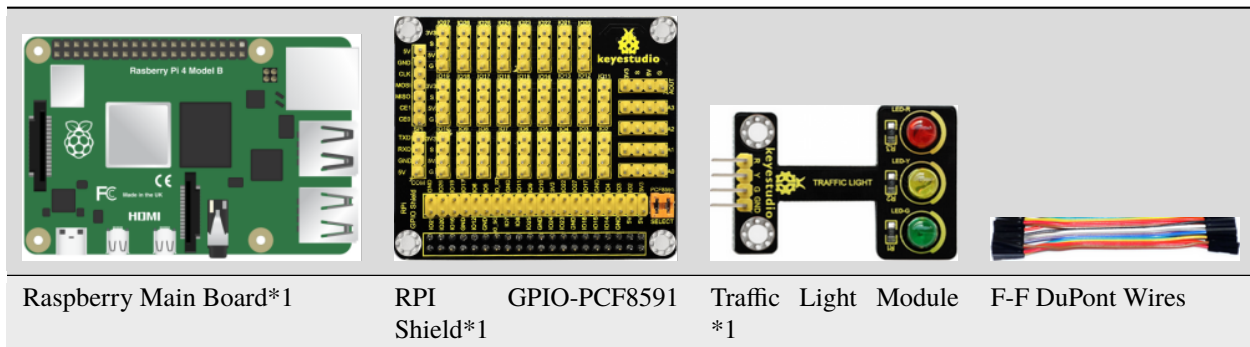
Green light allows traffic to proceed in the direction denoted if it is safe to do so and there is room on the other side of the intersection.

Red light prohibits any traffic from proceeding. A flashing red indication requires traffic to stop and then proceed when it is safe (equivalent to a stop sign).

Amber light (also known as ‘orange light’ or ‘yellow light’):

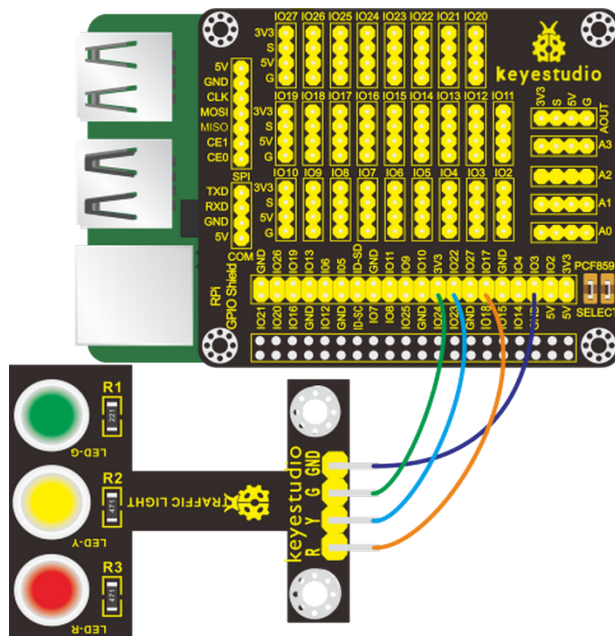
Warns that the signal is about to change to red, with some jurisdictions requiring drivers to stop if it is safe to do so, and others allowing drivers to go through the intersection if safe to do so.

### (2)Components Needed



### (3)Connection Diagram

Traffic Light Module	RPI GPIO-PCF8591 Shield
R	IO18
Y	IO23
G	IO24
GND	GND



#### (4) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson5_Traffic_Light
gcc Traffic_Light.c -o Traffic_Light -lwiringPi
sudo ./Traffic_Light
```

#### (5) Test Results

Note: Press Ctrl + C on keyboard and exit code running.

Red light is on 5s and off, yellow light flashes 3s and turn off, green light is lit for 5s and off, in loop way.

#### (6) Example Code

```
#include <wiringPi.h>

#define R_pin 1 //BCM GPIO 18
#define G_pin 5 //BCM GPIO 24
#define Y_pin 4 //BCM GPIO 23

int main()
{
    wiringPiSetup();
    char j;
    pinMode(R_pin, OUTPUT);
    pinMode(G_pin, OUTPUT);
    pinMode(Y_pin, OUTPUT);

    digitalWrite(R_pin, LOW);
    digitalWrite(G_pin, LOW);
    digitalWrite(Y_pin, LOW);

    while(1)
```

(continues on next page)

(continued from previous page)

```

{
  digitalWrite(R_pin, HIGH);//// turn on red LED
  delay(5000);// wait 5 seconds
  digitalWrite(R_pin, LOW); // turn off red LED
  for(j=0;j<3;j++) // blinks for 3 times
  {
    digitalWrite(Y_pin, HIGH);// turn on yellow LED
    delay(500);// wait 0.5 second
    digitalWrite(Y_pin, LOW);// turn off yellow LED
    delay(500);// wait 0.5 second
  }

  digitalWrite(G_pin, HIGH);// turn on green LED
  delay(5000);// wait 5 second
  digitalWrite(G_pin, LOW);// turn off green LED
}
}

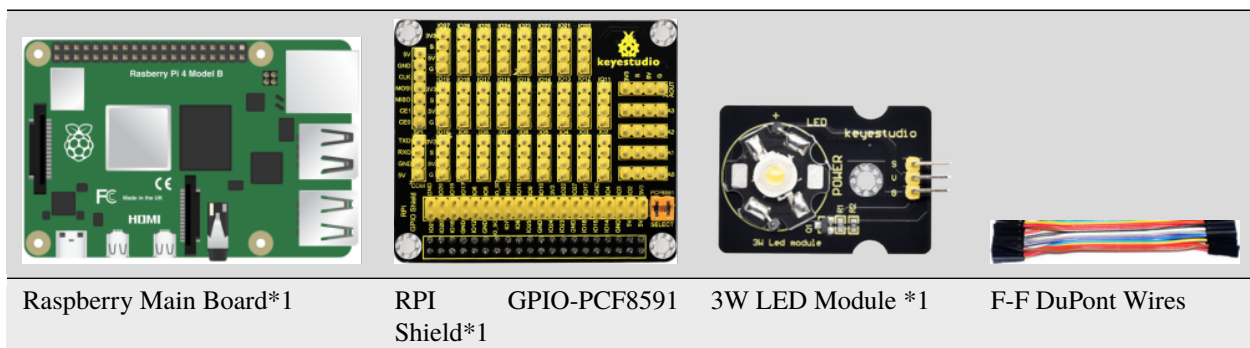
```

#### 4.4.6 Project 6Illuminating Lamp

##### (1)Description

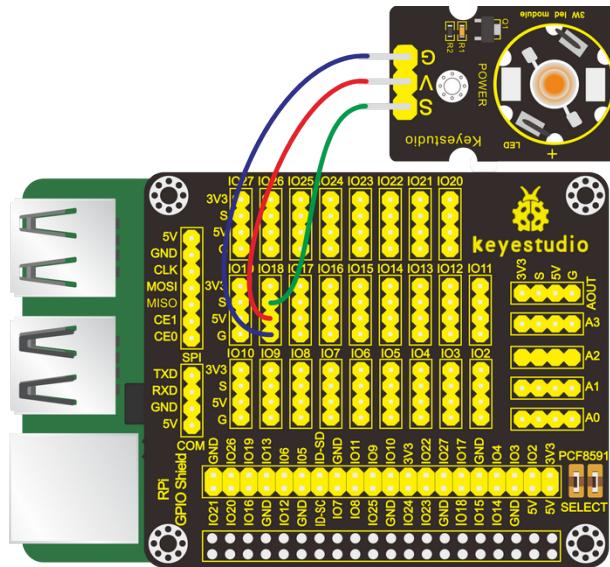
Nowadays, illuminating lamps are indispensable in our lives for we need them to light the surroundings for us, especially at night. In this experiment, we will use a LED of 3W . This LED is of high brightness because the lamp beads it carries are 3W, that is, the luminous power is 3W. We can apply this module to Arduino, Raspberry Pi and other projects. For example, smart robots can use the module for lighting purposes. However, please note that for safety reasons, do not touch your eyes with this LED directly. Otherwise your eyes maybe damaged.

##### (2)Components Needed



##### Connection Diagram

3W LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



### (3)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson6_3W_LED
```

```
gcc 3W_LED.c -o 3W_LED -lwiringPi
```

```
sudo ./3W_LED
```

### (4)Test Results

When the program runs, the 3W LED lights and it is shown on the terminal.

Note: Press Ctrl + C on keyboard and exit code running.

### Example Code

```
#include <wiringPi.h>
#include <stdio.h>

#define ledPin 1 //define led pin, BCM GPIO 18

int main()
{
    wiringPiSetup(); //Initialize wiringPi

    pinMode(ledPin,OUTPUT); //set the ledPin OUTPUT mode

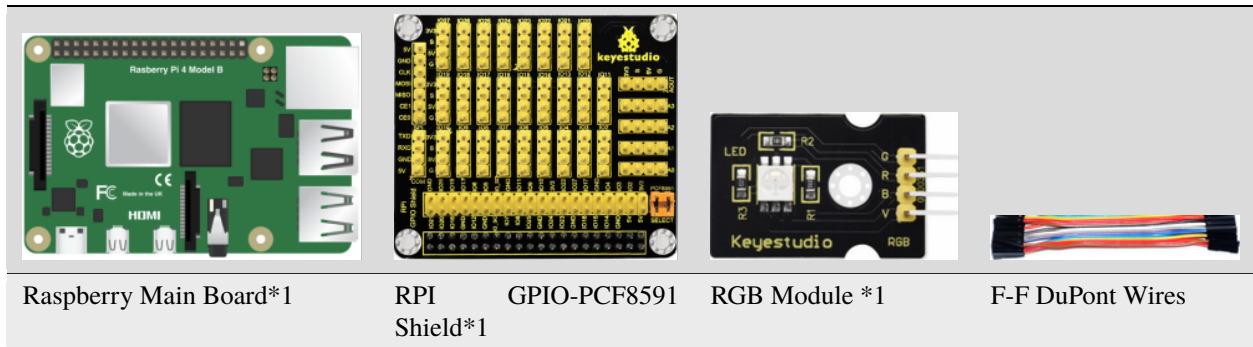
    while(1)
    {
        digitalWrite(ledPin,HIGH); //turn on led
        printf("turn on the LED\n");
    }
}
```

### 4.4.7 Project 7RGB Light

#### (1)Description

In this chapter, we will demonstrate how RGB lights show different colors via programming

#### (2)Components Needed

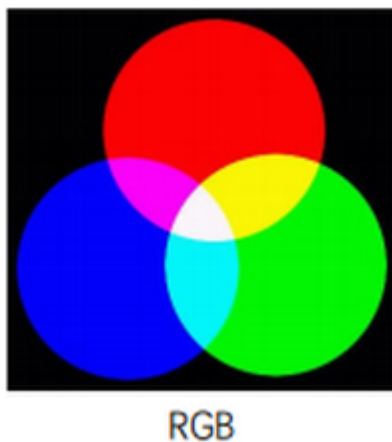


#### (3)Knowledge about Component:

##### RGB Module

The RGB module integrates with three LEDs in red, green and blue respectively. These three LEDs also share the same anode. The combinations of these three colors can form almost all other colors visible to human eyes. Thus, it has found wide applications in terms of colors.

Red, green and blue are three primary colors. They could produce all kinds of visible lights when mixing them up. Computer screen, single pixel mobile phone screen, neon light work under this principle.

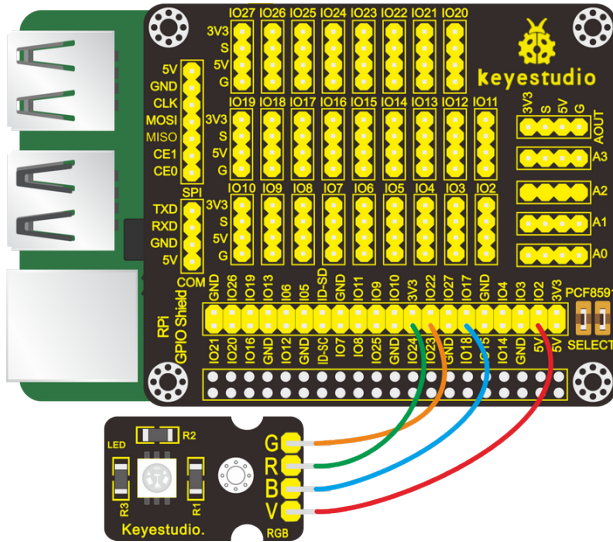


Theoretically, if we use three 8-bit PWM (Pulse Width Modulation) signals to control a RGB LED, we can create  $2^8 * 2^8 * 2^8 = 16777216$  (about 16 million) different combinations.

Now, let's make a RGB LED display all kinds of colors.

#### (4)Connection Diagram

RGB Module	RPI GPIO-PCF8591 Shield
R	IO24
G	IO23
B	IO18
V	5V



#### (5)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson7_RGB_LED
```

```
gcc RGB_LED.c -o RGB_LED -lwiringPi
```

```
sudo ./RGB_LED
```

#### (6)Test Results

RGB lights show colors randomly

Note: Press Ctrl + C on keyboard and exit code running

RGB light shows the all kinds of colors randomly.

#### (7)Example Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <wiringPi.h>
#include <softPwm.h>
#include <time.h>

#define pin_R 5 //BCM GPIO 24
#define pin_G 4 //BCM GPIO 23
#define pin_B 1 //BCM GPIO 18

int main(void){
```

(continues on next page)



(continued from previous page)

```

int red,green,blue;
if (wiringPiSetup() == -1){
    printf("Setup GPIO error!\n");
    return -1;
}
softPwmCreate(pin_R, 0, 100);
softPwmCreate(pin_G, 0, 100);
softPwmCreate(pin_B, 0, 100);

while (1){
    srand((unsigned)time(NULL));
    red = rand()%101 + 0;
    green = rand()%101 + 0;
    blue = rand()%101 + 0;
    softPwmWrite(pin_R, red);
    softPwmWrite(pin_G, green);
    softPwmWrite(pin_B, blue);
    delay(100);
}
return 0;
}

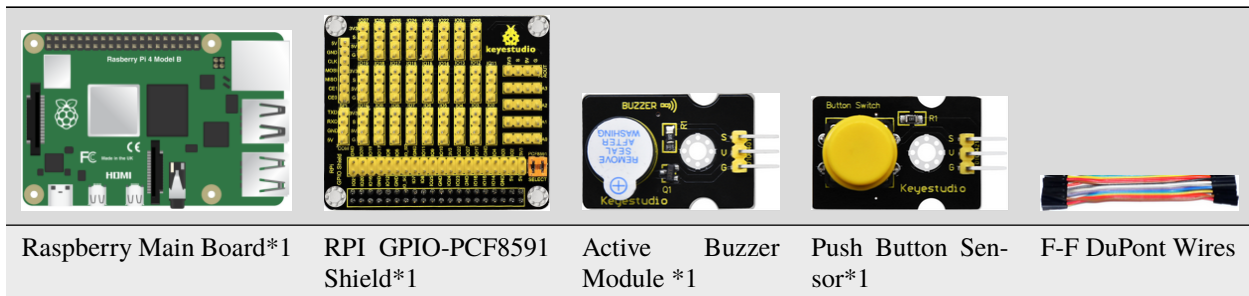
```

## 4.4.8 Project 8Doorbell

### (1)Description

Doorbells have made our daily life more convenient. When a guest arrives, we will get this information when he/she rings the bell. In this project, we will learn to make a doorbell by ourselves.

### (2)Components Needed

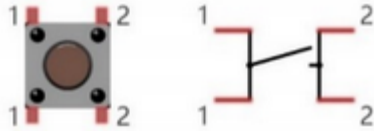


### (3)Knowledge about Component:

**\*\*Active Buzzer Module\*\***The active buzzer is equipped with an internal oscillator, which makes it possible to automatically generate a tone as long as current flows through. It is very easy and convenient. But it also has its shortcoming that the fixed frequency means it can only makes a monotone.

#### Push Button Sensor

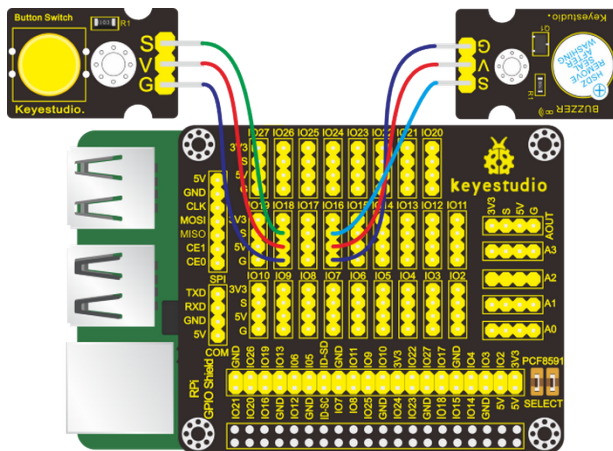
It can control circuits. Before pressed, the current can't pass from one end to the other end. Both ends are like two mountains. There is a river in between. We can't cross this mountain to another mountain. When pressed, the internal metal piece is connecting the two sides to let the current pass, just like building a bridge to connect the two mountains.



Inner structure: , 1 and 1 , 2 and 2 are connected. However, 1 and 2 are disconnected when the button is not pressed; 1 and 2 are connected when pressing the button.

#### (4)Connection Diagram

Active Buzzer Module	RPI Shield	GPIO-PCF8591	Push Button Sensor	RPI Shield	GPIO-PCF8591
S	SIO16		S	SIO18	
V	5V		V	5V	
G	G		G	G	



#### (5)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson8_Active_Buzzer
gcc Active_Buzzer.c -o Active_Buzzer -lwiringPi
sudo ./Active_Buzzer
```

#### (6)Test Results

After running the program and pressing the button, the buzzer makes a sound and the terminal prints 0 (low level); otherwise, the buzzer makes no sounds and the terminal prints 1 (high level).

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define button 1 //button pin BCM GPIO 18
#define buzzer 27 //buzzer pin BCM GPIO 16
int main()
```

(continues on next page)

(continued from previous page)

```

{
  wiringPiSetup();
  char val;
  {
    pinMode(button,INPUT); //set the button pin INPUT mode
    pinMode(buzzer,OUTPUT);
  }

  while(1)
  {
    val=digitalRead(button); // digital read
    printf("val = %d\n", val);
    if(val==0)//check if the button is pressed, if yes, turn on the Buzzer
      digitalWrite(buzzer,HIGH); //The buzzer made a sound
    else
      digitalWrite(buzzer,LOW);
  }
}

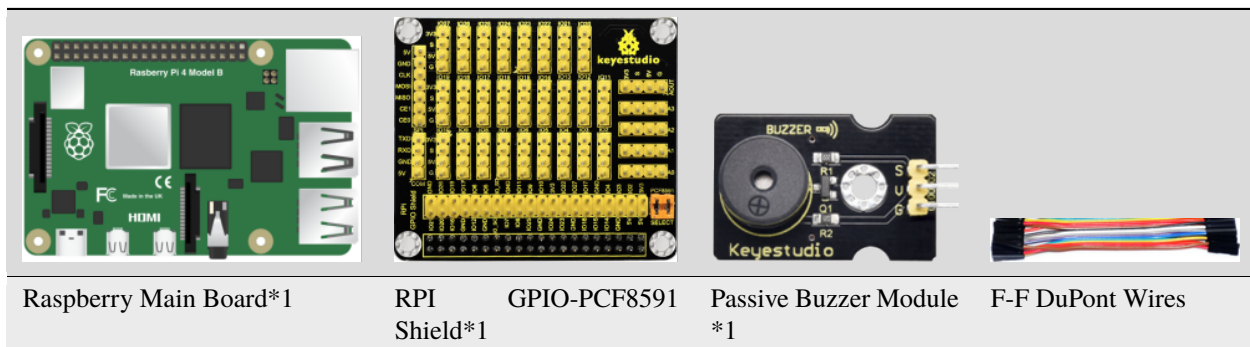
```

#### 4.4.9 Project 9 Passive Buzzer

##### (1)Description

We will conduct an interesting experiment——control passive buzzer to compose a song.

##### (2)Components Needed



##### (3)Knowledge about Component

###### Passive buzzer

Passive buzzer is a type of electronic buzzer with integrated structure.

Buzzers can be categorized as active and passive ones (see the following picture).

An active buzzer has a built-in oscillating source, so it will make sounds when electrified. But a passive buzzer does not have such source, so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

Turn the pins of two buzzers face up, and the one with a green circuit board is a passive buzzer, while the other enclosed



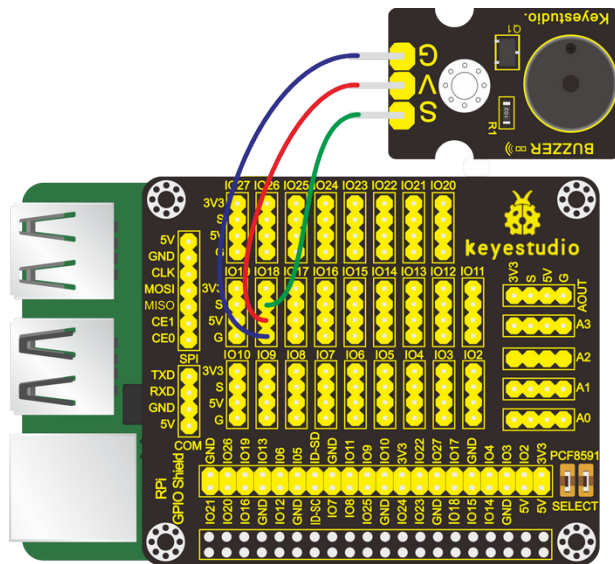
with a black tape is an active one, as shown

Passive buzzer provides alternating current to sound coils to make electronic magnet and permanent magnet attraction or repulsion so as to push vibration film to emit sound, according to electromagnetic induction.

Only certain frequency with high and low levels can make passive buzzer emit sound, since DC current only makes vibration film vibrated continuously rather than producing sound.

#### (4) Connection Diagram

Passive Buzzer Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### (5) Run Example Code1

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson9_Passive_Buzzer
```

```
gcc Passive_Buzzer1.c -o Passive_Buzzer1 -lwiringPi
```

```
sudo ./Passive_Buzzer1
```

#### (6) Test Results1

After running the program, the passive buzzer makes the sound “didi”.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code1

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

(continues on next page)

(continued from previous page)

```

#include <wiringPi.h>

#define buzPin 1 //BCM GPIO 18

void init()
{
    if (wiringPiSetup () == -1)
        exit (1) ;
    pinMode(buzPin, PWM_OUTPUT); //Set the pin to PWM output mode
    pwmSetMode(PWM_MODE_MS); // Set PWM signal mode to MS mode
    pwmSetClock(32); // Set the clock base frequency to 19.2m /32=600KHZ
}

void beep(int freq,int t_ms)
{
    int range;
    if(freq<100||freq>1000)
    {
        printf("invalid freq");
        return;
    }
    // Set the range to 600KHZ/ Freq. That is,
    //the freQ frequency period is composed of the range of 1/600khz.
    range=600000/freq;
    pwmSetRange(range);
    pwmWrite(buzPin,range/2); // Set the duty cycle to 50%.
    if(t_ms>0)
    {
        delay(t_ms);
    }
}

int main()
{
    wiringPiSetup();
    init();

    while(1)
    {
        beep(262,300); //Frequency and time
        printf("do\n");
        beep(294,300);
        printf("re\n");
        beep(330,300);
        printf("mi\n");
        beep(349,300);
        printf("fa\n");
        beep(392,300);
        printf("so\n");
        beep(440,300);
        printf("la\n");
        beep(494,300);
    }
}

```

(continues on next page)

(continued from previous page)

```

    printf("si\n");
    beep(523,300);
    printf("Do\n");
    pwmWrite(buzPin,0);    //turn off the buzzer
    delay(2000);
}
}

```

## (8)Run Example Code2

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson9_Passive_Buzzer
```

```
gcc Passive_Buzzer2.c -o Passive_Buzzer2 -lwiringPi
```

```
sudo ./Passive_Buzzer2
```

## (9)Test Results2

The passive buzzer plays a “Happy Birthday” song.

Note: Press Ctrl + C on keyboard and exit code running.

## (10)Example Code2

```

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <wiringPi.h>
#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Do_h 532
#define Re_h 587
#define Mi_h 659
#define Fa_h 698
#define Sol_h 784
#define La_h 880
#define Si_h 988

#define buzPin 1    //buzzer pin BCM GPIO 18

//The tones
int song_1[]=
{
    Sol,Sol,La,Sol,Do_h,Si,
    Sol,Sol,La,Sol,Re_h,Do_h,
    Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,
    Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_h
};

```

(continues on next page)

(continued from previous page)

```

//To the beat
float beat_1[]=
{
    0.5,0.5,1,1,1,1+1,
    0.5,0.5,1,1,1,1+1,
    0.5,0.5,1,1,1,1,1,
    0.5,0.5,1,1,1,1+1
};

int length;
int x;

void init()
{
    if (wiringPiSetup () == -1)
        exit (1) ;
    pinMode(buzPin, PWM_OUTPUT); //Set the pin to PWM output mode
    pwmSetMode(PWM_MODE_MS); // Set PWM signal mode to MS mode
    pwmSetClock(32); // Set the clock base frequency to 19.2m /32=600KHZ
}

void beep(int freq,int t_ms)
{
    int range;
    if(freq<100 || freq>1000)
    {
        printf("invalid freq");
        return;
    }
    // Set the range to 600KHZ/ Freq. That is,
    //the freq frequency period is composed of the range of 1/600khz.
    range=600000/freq;
    pwmSetRange(range);
    pwmWrite(buzPin,range/2); // Set the duty cycle to 50%.
    if(t_ms>0)
    {
        delay(t_ms);
    }
}

int main()
{
    wiringPiSetup();
    init();
    length=sizeof(song_1)/sizeof(song_1[0]); //Number of tones

    while(1)
    {
        for(x=0;x<length;x++) //play
        {
            beep(song_1[x],500*beat_1[x]);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    pwmWrite(buzPin,0);    //turn off buzzer
    delay(2000);
  }
}

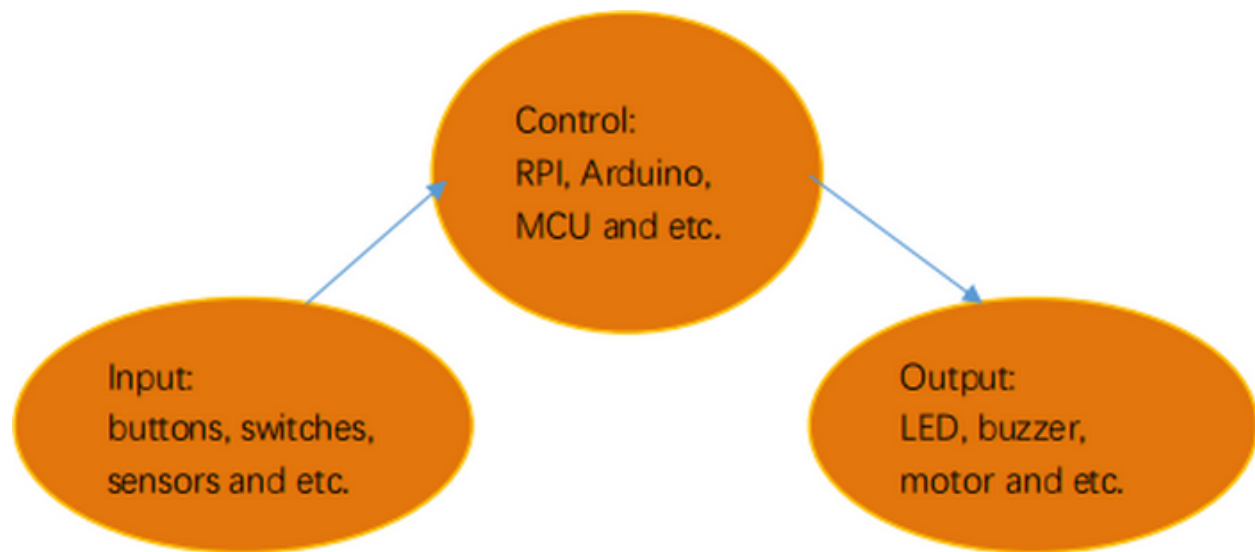
```

#### 4.4.10 Project 10Button-controlled LED

##### (1)Description

Usually a complete open loop control is made of external information input, controller and actuator.

The external information is input into controller which can analyze the input data and send to control signals to make actuator to react.



A button-controlled LED is decided by an open loop control. Next, we will make a desk lamp with a button, an LED and RPi. LED is on when button is pressed, on the contrary, it will be off.

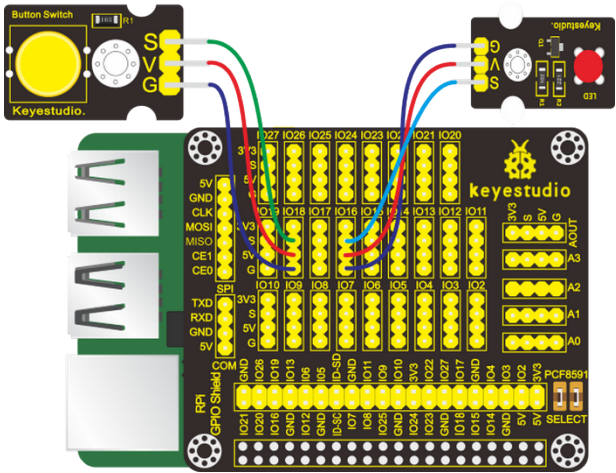
##### (2)Components Needed



##### (3)Connection Diagram

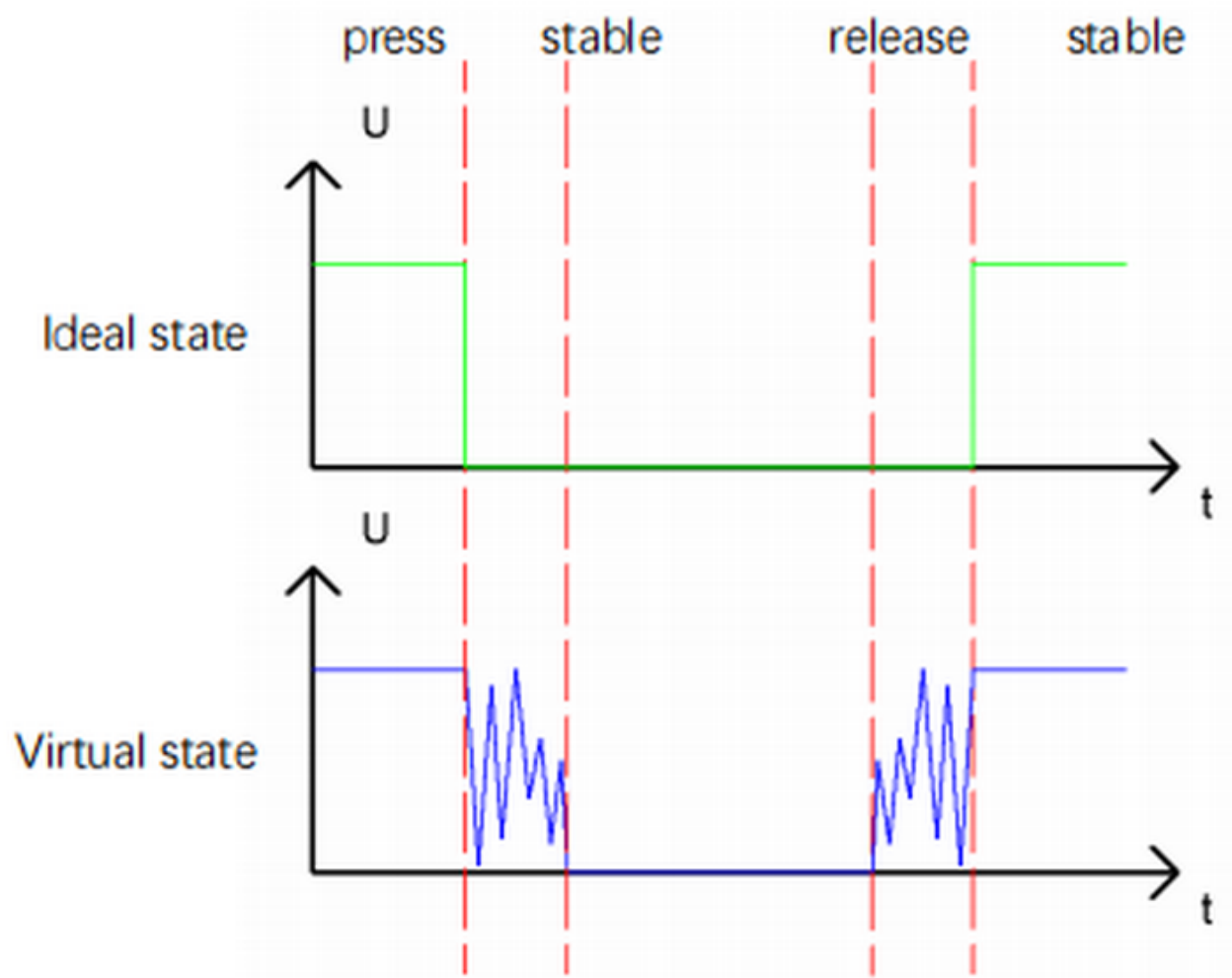


Red LED Module	RPI GPIO-PCF8591 Shield	Push Button Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	S	SIO18
V	5V	V	5V
G	G	G	G



(4)Eliminate button jitters

When the button is pressed, its state does not change immediately because it is a mechanical vibration and continuous jitters exists before entering another state. It is similar to release the button.



Therefore, if we directly detect the state of the button, there will be multiple presses and releases. Jitters will mislead the high-speed operation of the MCU to cause considerable misjudgments. To eliminate the button jitters, here we put forward a solution that we design to detect the button state for more than once. After a while, the stable button state is adopted to represent that the button is pressed.

#### (5)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson10_Button_controlled_LED
gcc Button_controlled_LED.c -o Button_controlled_LED -lwiringPi
sudo ./Button_controlled_LED
```

#### (6)Test Results

Press button, LED turns on, press again, LED is off and then repeats this pattern.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define btnPin 1 // button Pin BCM GPIO 18
```

(continues on next page)

(continued from previous page)

```
#define ledPin 27 // LED pin BCM GPIO 16

int main()
{
    wiringPiSetup();
    int val; //Button variables
    int count = 0; //Record the number of button presses
    int flag = 0; //Odd even variable
    pinMode(btnPin, INPUT);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW); //turn off led

    while(1)
    {
        val=digitalRead(btnPin); //Receive button value
        if(val == 0)
        {
            delay(10);
            val=digitalRead(btnPin); //Receive button value
            if(val == 1)
            {
                count = count + 1;
                printf("count = %d", count);
            }
        }
        flag = count % 2; //Remainder 2 ,Even is 0, odd is 1
        if(flag == 1)
            digitalWrite(ledPin, HIGH); //turn on led
        else
            digitalWrite(ledPin, LOW); //turn off led
    }
}
```

#### 4.4.11 Project 11 PIR Motion Sensor

##### (1)Description

Lamps only light up when people pass by installed in some places, which are conducive to energy and cost saving. Have you ever thought about the principle behind these lamps? It is because of PIR motion sensors. In this lesson, we will learn about PIR motion sensor.

##### (2)Components Needed



## (3) Knowledge about Component

**PIR Motion Sensor**

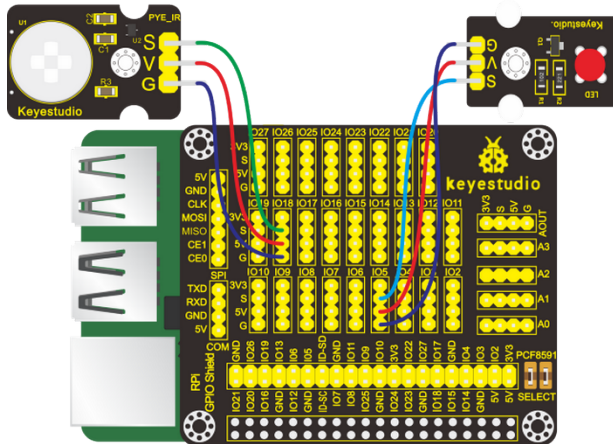
The principle of human infrared sensor is that when certain crystals, such as lithium tantalate and triglyceride sulfate, are heated, the two ends of the crystal will generate an equal number of charges, with opposite signs, which can be converted into voltage output by an amplifier.

Human body will emit IR ray, although weak but can be detected. This sensor outputs 1 (high level ) when human being is detected; otherwise, it outputs 0(low level).

Note: Nothing but moving person can be detected, with the detection distance up to 3m.

## (4) Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	PIR Motion Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



## (5) Working Principle

When the PIR motion sensor detects movements around, the LED lights and the terminal prints somebody; while when there is no movements sensed, the LED reminds off and the terminal prints nobody.

## (6) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson11_PIR_Led
```

```
gcc PIR_Led.c -o PIR_Led -lwiringPi
```

```
sudo ./PIR_Led
```

## (7) Test Results

LED will turn on and terminal prints somebody if PIR motion sensor detects people; if not, LED will be off and terminal will print nobody.

Note: Press Ctrl + C on keyboard and exit code running.

## (8)Example Code

```

#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define PIR_pin 1 //PIR pin BCM GPIO 18
#define led_pin 21 //LED pin BCM GPIO 5

int main(void)
{
    int val = 0;
    wiringPiSetup();
    pinMode(PIR_pin, INPUT);
    pinMode(led_pin, OUTPUT);

    while(1)
    {
        val=digitalRead(PIR_pin);
        if(val==1)
        {
            printf("somebody\n");
            digitalWrite(led_pin,HIGH);
        }
        else
        {
            printf("nobody\n");
            digitalWrite(led_pin,LOW);
        }
    }
}

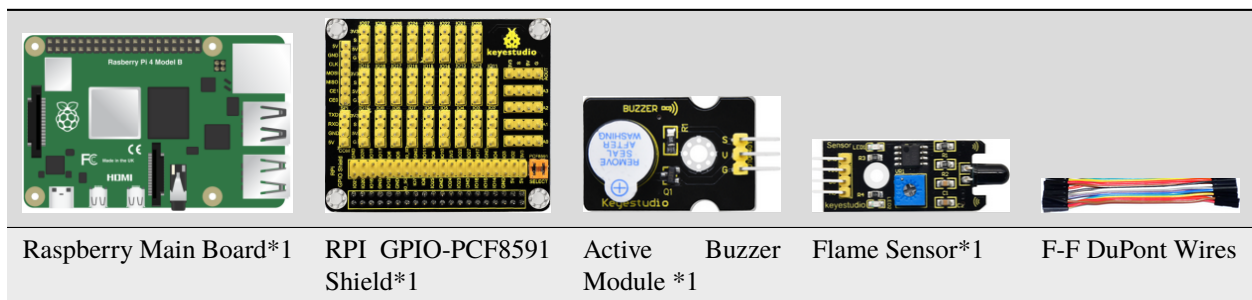
```

## 4.4.12 Project 12Fire Alarm

## (1)Description

A flame detector is a sensor designed to detect and respond to the presence of flames or fire, allowing flame detection.

## (2)Components Needed



## (3)Knowledge about Component

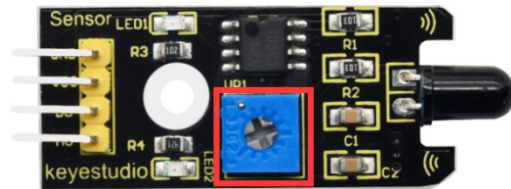
## Flame Sensor

Flame sensor is made based on the principle that infrared ray is highly sensitive to flame. It has an infrared receiving tube specially designed to detect fire, and then convert the flame brightness to fluctuating level signal. The signals are then input into the central processor and be dealt with accordingly.

Flame sensor is used to detect fire source with wavelength in 760nm-1100nm, detection angle is 60°. When its IR waves length is close to 940nm, and its sensitivity is the highest.

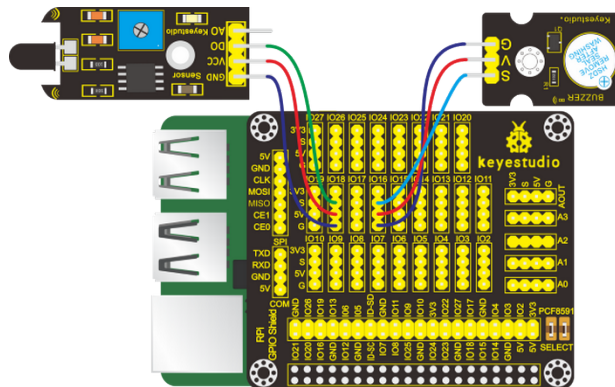
Notice that keep flame sensor away from fire source to defend its damage for its working temperature is between -25°-85°

Note: a potentiometer is built in the sensor so its sensitivity can be adjusted by rotating it.



(4)Connection Diagram

Active Buzzer Module	RPI GPIO-PCF8591 Shield	Flame Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	D0	SIO18
V	5V	VCC	5V
G	G	GND	G



(5)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson12_Flame_Buzzer
gcc Flame_Buzzer.c -o Flame_Buzzer -lwiringPi
sudo ./Flame_Buzzer
```

(6)Test Results

After running the program, when the sensor detects flame, the buzzer makes noises, the LED1 lights and the terminal prints 0 (low level); otherwise, the buzzer makes no sounds, the LED1 reminds off and the terminal prints 1 (high level).

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define flamePin 1 //BCM GPIO 18
#define buzPin 27 //define buzzer pin BCM GPIO 16

int main()
{
    wiringPiSetup();
    char val;
    {
        pinMode(flamePin, INPUT);
        pinMode(buzPin, OUTPUT);
    }

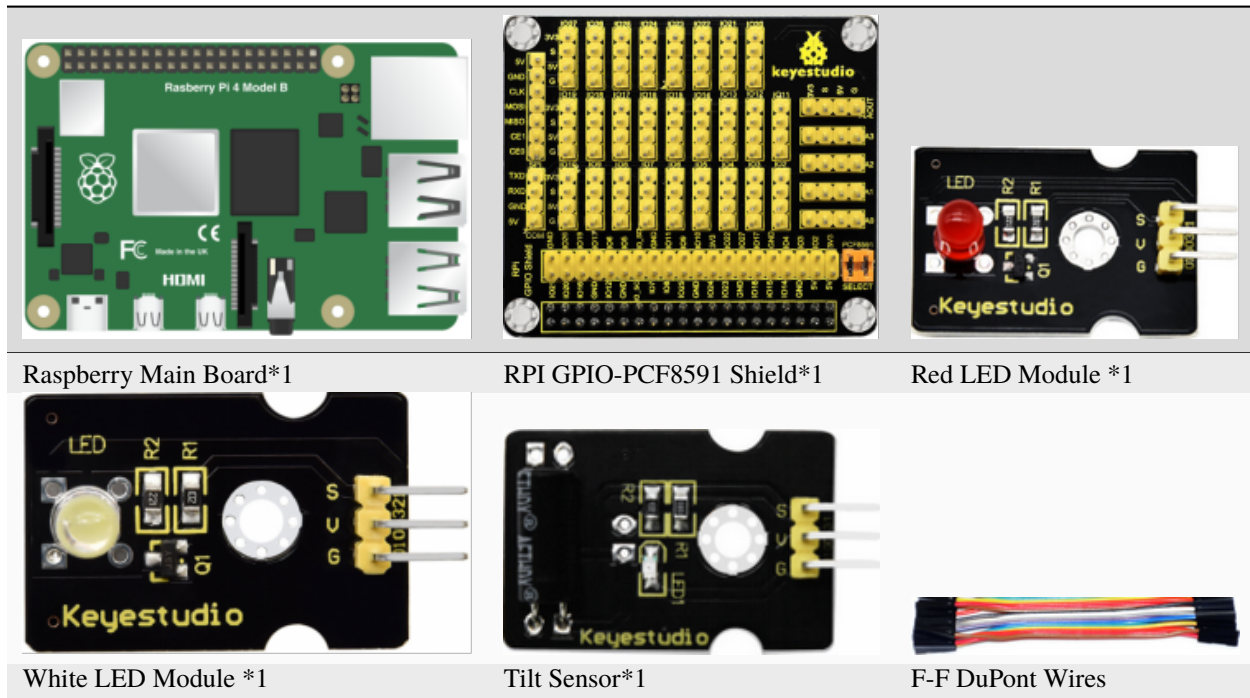
    while(1)
    {
        val=digitalRead(flamePin);
        printf("val = %d\n",val);
        if(val==0) //When flame is detected
            digitalWrite(buzPin,HIGH); //Buzzer turn on
        else
            digitalWrite(buzPin,LOW); //Buzzer turn off
    }
}
```

### 4.4.13 Project 13Electronic Hourglass

#### (1)Description

An hourglass (or sand glass, sand timer, sand clock or egg timer) is a device used to measure the passage of time. It comprises two glass bulbs connected vertically by a narrow neck that allows a regulated flow of a substance(historically sand) from the upper bulb to the lower one. Typically the upper and lower bulbs are symmetric so that the hourglass will measure the same duration regardless of orientation. The specific duration of time a given hourglass measures is determined by factors including the quantity and coarseness of the particulate matter, the bulb size, and the neck width.

#### (2)Components Needed



### (3) Knowledge about Component

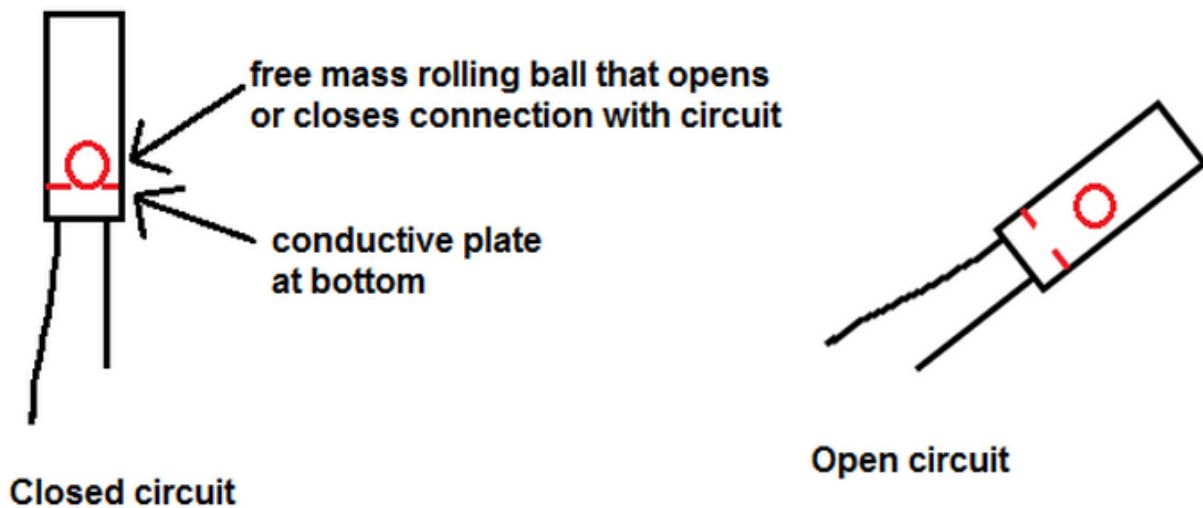
#### Tilt Sensor

Tilt sensors (tilt ball switch) allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out.

The tilt-switch twig is the equivalent of a button, and is used as a digital input. Inside the tilt switch is a ball that make contact with the pins when the case is upright. Tilt the case over and the balls don't touch, thus not making a connection. When the switch is level it is open, and when tilted, the switch closes.

It can be used for orientation detection, alarm device or others.

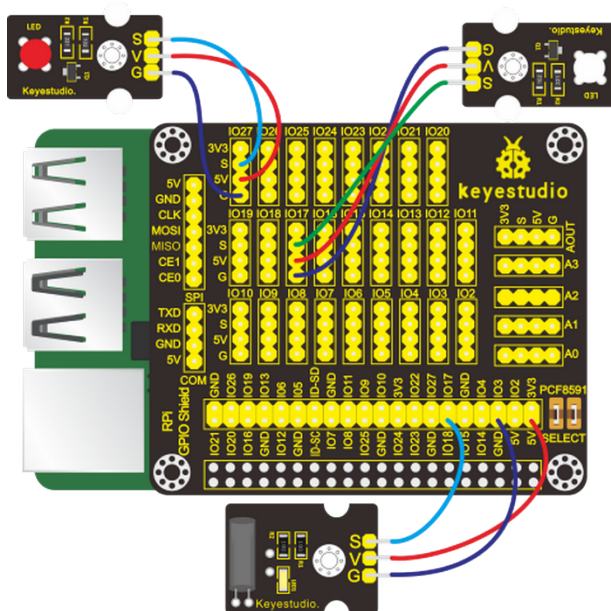
Here is the principle of tilt sensor to illustrate how it works:



### (4) Connection Diagram



Red LED Module	RPI GPIO-PCF8591 Shield	Ball Tilt Sensor	RPI GPIO-PCF8591 Shield
S	SIO27	S	SIO18
V	5V	V	5V
G	G	G	GND
White LED Module	RPI GPIO-PCF8591 Shield		
S	SIO17		
V	5V		
G	G		



#### (5)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson13_Ball_Tilt_Sensor
gcc Ball_Tilt_Sensor.c -o Ball_Tilt_Sensor -lwiringPi
sudo ./Ball_Tilt_Sensor
```

#### (6)Test Results

LED1 gradually brightens and LED2 gradually darkens when placing electronic hourglass. However, when you make it upside down, LED1 gradually darkens and LED2 gets bright.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <wiringPi.h>
#include <softPwm.h>

//define led pin
#define LED1 0 //BCM GPIO 17
#define LED2 2 //BCM GPIO 27
//define Ball Tilt Sensor Pin
#define tiltPin 1 //BCM GPIO 18

int main(void){
    int val;
    int val1 = 50; //Initial value of LED brightness
    int val2 = 50;
    if (wiringPiSetup() == -1)
    {
        printf("Setup GPIO error!\n");
        return -1;
    }
    softPwmCreate(LED1, 0, 100); //Define the pin as PWM output
    softPwmCreate(LED2, 0, 100);
    while (1)
    {
        val=digitalRead(tiltPin); //Read the value of the tilt sensor
        if(val==0) //upright
        {
            val1++; //The value of LED1 increases
            val2--; //Led2 value reduced
            if(val1>=100) //The size of the limit
            {
                val1 = 100;
            }
            if(val2<=0) //The size of the limit
            {
                val2 = 0;
            }
            softPwmWrite(LED1, val1); //The value after PWM output changes
            softPwmWrite(LED2, val2);
            delay(50); //Delay, adjust the speed
        }
        else
        {
            val1--;
            val2++;
            if(val1<=0)
            {
                val1 = 0;
            }
            if(val2>=100)
            {
                val2 = 100;
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
    softPwmWrite(LED1, val1);
    softPwmWrite(LED2, val2);
    delay(50);
  }
}
return 0;
}

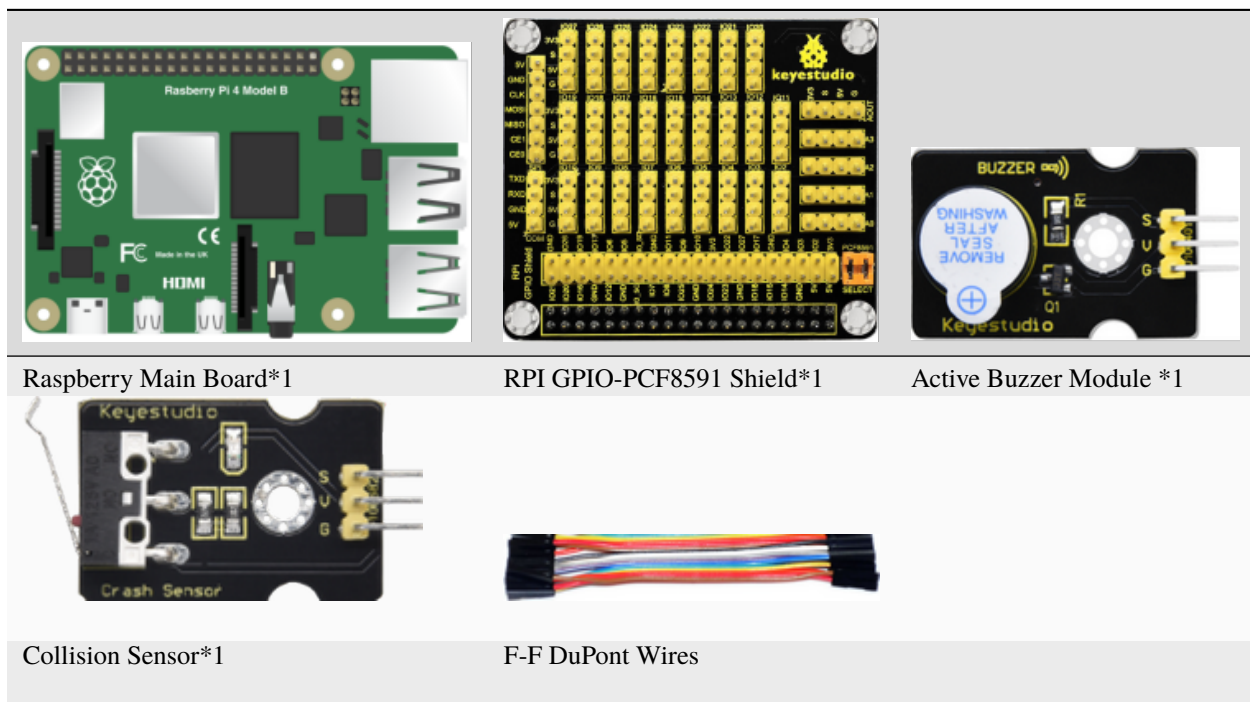
```

#### 4.4.14 Project 14 Collision Alarm

##### (1)Description

We can use the collision sensor to detect whether crash happens. When the metal plate above the push button switch of the sensor is knocked, it outputs low level signals; and when the button is open, it remind in high level. In this project, collision sensor will be applied to control the active buzzer.

##### (2)Components Needed



##### (3)Knowledge about Component

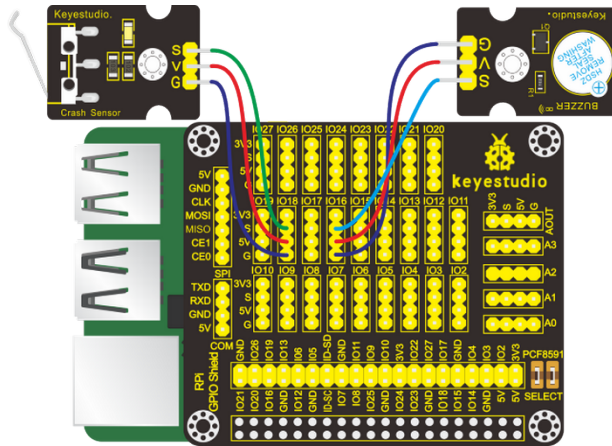
###### Collision Sensor:

It is a widely used collision sensor that has a push button switch covered by a mental plate. When the plate is pushed, the button is pressed, the sensor outputs low level and the LED on it lights; or it outputs high level and the LED reminds off.

This sensor is often used as a limit switch in a 3D printer.

##### (4)Connection Diagram

Active Buzzer Module	RPI GPIO-PCF8591 Shield	Collision Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	S	SIO18
V	5V	V	5V
G	G	G	G



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson14_Crash_Buzzer
```

```
gcc Crash_Buzzer.c -o Crash_Buzzer -lwiringPi
```

```
sudo ./Crash_Buzzer
```

#### (6) Test Results

After running the program, when the metal plate of the push button switch is pressed, the buzzer makes sound and the terminal prints 0 ( low level) or it keeps silent and the terminal prints 1 (high level).

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define crash 1 //crash pin BCM GPIO 18
#define buzzer 27 //buzzer pin BCM GPIO 16
int main()
{
    wiringPiSetup();
    char val;
    {
        pinMode(crash, INPUT); //set the crash pin INPUT mode
        pinMode(buzzer, OUTPUT);
    }
}
```

(continues on next page)

(continued from previous page)

```

while(1)
{
    val=digitalRead(crash); // digital read
    printf("val = %d\n", val);
    if(val==0)//check if the metal shrapnel is pressed, if yes, turn on the Buzzer
        digitalWrite(buzzer,HIGH); //The buzzer made a sound
    else
        digitalWrite(buzzer,LOW);
}
}

```

#### 4.4.15 Project 15 Line-tracking Sensor

##### (1)Description

You may have seen that in an experiment a smart car moved along a black line and it didn't overstep this boundary. How did it make it? The credit goes to a line-tracking sensor. And in this project, we intend to learn about the line-tracking sensor.

##### (2)Components Needed



##### (3)Knowledge about Component

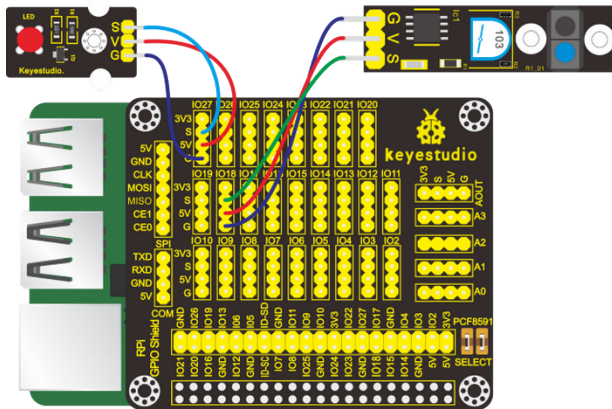
##### Line-tracking Sensor

It is an infrared sensor in nature which can detect white and black objects. The working principle of the TCRT5000 pair tube on the sensor is based on the different reflectivity of infrared to colors so as to convert this different strengths of reflected signals to electric signals. When the sensor detects black objects, it is in high level while when it sensors white items it is in low level. And the detection altitude is from 0 to 3cm. You can rotate the potentiometer in a bid to adjust the sensitivity of the line-tracking sensor.



##### (4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Line-tracking Sensor	RPI GPIO-PCF8591 Shield
S	SIO27	S	SIO18
V	5V	V	5V
G	G	G	G



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson15_Tracking
```

```
gcc Tracking.c -o Tracking -lwiringPi
```

```
sudo ./Tracking
```

#### (6) Test Results

After running the program, when the line-tracking sensor detects black objects or no objects, then LED reminds off and the terminal prints 1 (high level); otherwise the LED lights up and the terminal prints 0 (low level).

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define tracking 1 //tracking pin BCM GPIO 18
#define led 2 //led pin BCM GPIO 27
int main()
{
    wiringPiSetup();
    char val;
    {
        pinMode(tracking, INPUT); //set the tracking pin INPUT mode
        pinMode(led, OUTPUT);
    }

    while(1)
```

(continues on next page)

(continued from previous page)

```

{
    val=digitalRead(tracking); // digital read
    printf("val = %d\n", val);
    if(val==0)//check if the the white line is detected if yes, turn on the led
        digitalWrite(led,HIGH); //The led made a sound
    else
        digitalWrite(led,LOW);
}
}

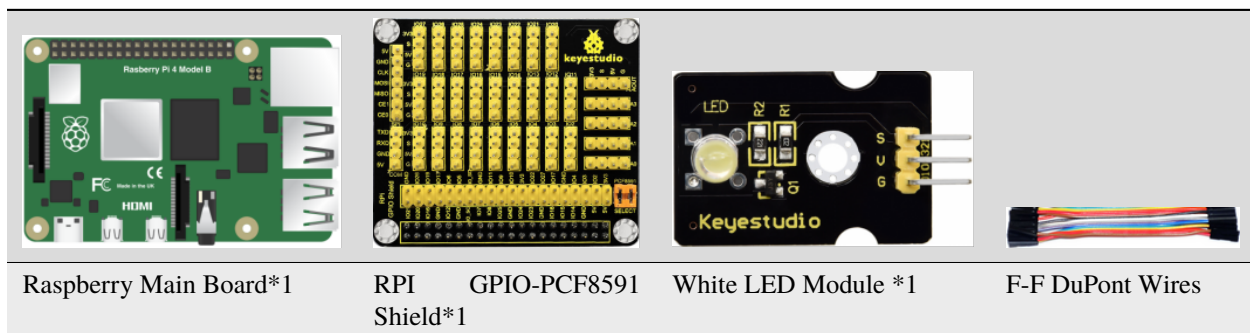
```

#### 4.4.16 Project 16Photo Interrupter Module

##### (1)Description

In our daily life, we often need to count and take measurements. But how? The combination of light interrupter module and Raspberry Pi can do the trick. In the project, we will count with the photo interrupter module.

##### (2)Components Needed



##### (3)Knowledge about Component

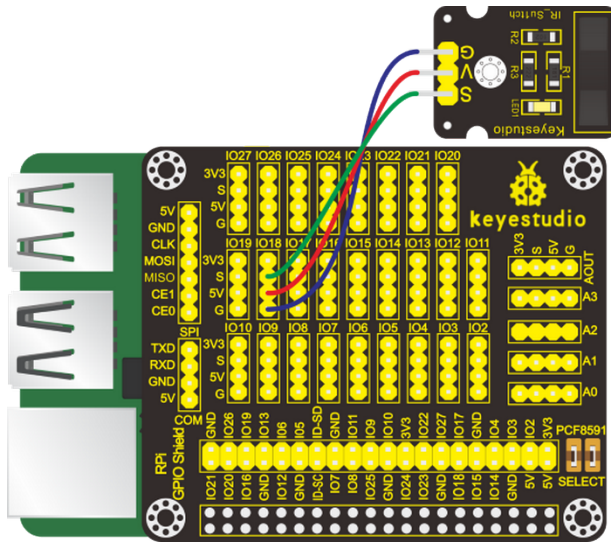
###### Photo Interrupter Module

It is a module which is equipped with a light emitting elements and light receiving elements aligned facing each other in a single package. It is based on the principle that the light passing through the U-shaped area will encounter blockage. Therefore, it is widely used in speed measurements, positioning count, small household appliances, optical limit switches, target detection and other fields.

If an object constantly passes through the U-shaped area of the photo interrupter module, the signal it outputs will shows constant changes between high and low levels. Therefore, we can count and measure speed by calculating the amount of high level and low level occurring.

##### (4)Connection Diagram

Photo Interrupter Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd/home/pi/C_code/lesson16_Count_Photofracture
```

```
gcc Count_Photofracture.c -o Count_Photofracture -lwiringPi
```

```
sudo ./Count_Photofracture
```

#### (6) Test Results

After running the program, when an object constantly passes through the U-shaped area on the sensor, the terminal prints numbers and these numbers gradually increase by 1.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define photofracturePin 1 // photofracture Pin BCM GPIO 18
int main()
{
    wiringPiSetup();
    int val; //Photofracture variables
    int count = 0; //Record the number of photofracture
    int flag = 0; //Odd even variable
    pinMode(photofracturePin, INPUT);

    while(1)
    {
        val=digitalRead(photofracturePin); //Receive photofracture value
        if(val == 0)
        {
            delay(10);
            val=digitalRead(photofracturePin); //Receive photofracture value
            if(val == 1)
            {
                count = count + 1;
            }
        }
    }
}
```

(continues on next page)



(continued from previous page)

```

        printf("count = %d\n",count);
        delay(50);
    }
}
flag = count % 2; //Remainder 2 ,Even is 0, odd is 1
}
}

```

#### 4.4.17 Project 17Magnetic Detection

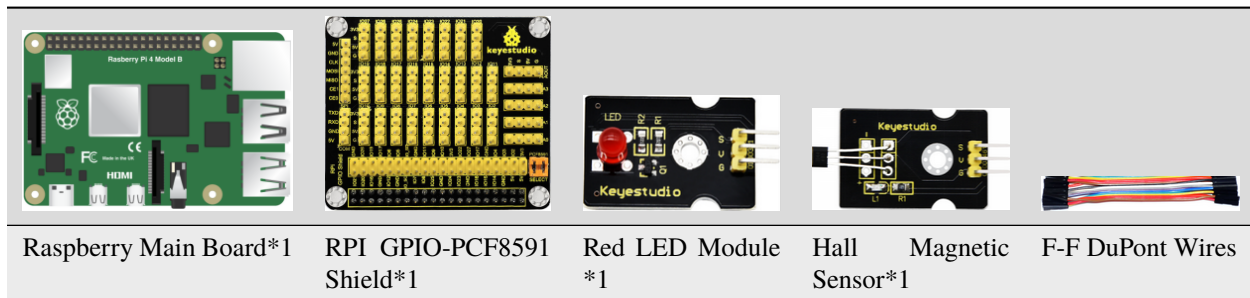
##### (1)Description

What is the best way to detect a magnet? Use another magnet? Yeah , it can but it is not sensitive enough. You still need to feel it by yourselves.

Perhaps you can try a hall magnetic sensor which features high sensitivity, quick response, nice temperature performance, and high reliability.

In this project, we will try to turn a LED on and off through a hall magnetic sensor.

##### (2)Components Needed



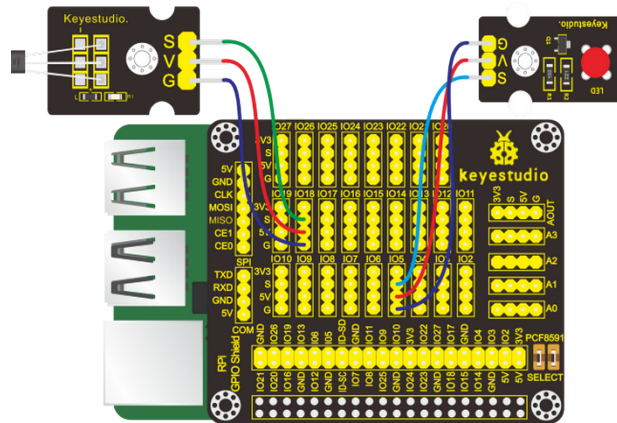
##### (3)Knowledge about Component:

##### Hall Magnetic Sensor

The main component built in the sensor is A3144E, which is an electronic magnetic device and an active one. It uses magnetic field and Hall effects to achieve the purpose of non-contact control. Since the Hall element itself is a chip in nature, its working life is theoretically unlimited. The sensor can be used to detect magnetic fields and output digital signals. It can sense magnetic materials within a detection range of about 3cm. Note that it can only detect the presence of a magnetic field nearby, but not the strength of the magnetic field.

##### (4)Connection Diagram

Red LED Module	RPi GPIO-PCF8591 Shield	Hall Magnetic Sensor	RPi GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson17_Hall_Magnetic
gcc Hall_Magnetic.c -o Hall_Magnetic -lwiringPi
sudo ./Hall_Magnetic
```

#### (6) Test Results

After running the program and placing a magnetic ball around the Hall magnetic sensor, when the sensor detects magnetic field nearby, the terminal prints “magnetic” and the LED lights up; otherwise, the terminal prints “nonmagnetic” and the LED stays dark.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Hall_pin 1 //hall pin BCM GPIO 18
#define led_pin 21 //LED pin BCM GPIO 5

int main(void)
{
    int val = 0;
    wiringPiSetup();
    pinMode(Hall_pin, INPUT);
    pinMode(led_pin, OUTPUT);

    while(1)
    {
        val=digitalRead(Hall_pin);
        if(val==1)
        {
            printf("nonmagnetic\n");
            digitalWrite(led_pin, LOW);
        }
        else
```

(continues on next page)

(continued from previous page)

```

{
    printf("magnetic\n");
    digitalWrite(led_pin,HIGH);
}
}

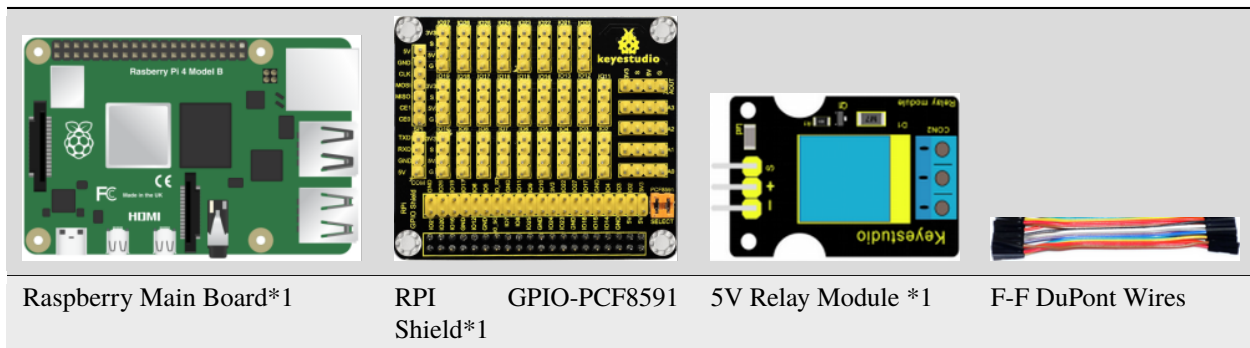
```

#### 4.4.18 Project 185V Relay

##### (1)Description

In daily life, electronic devices are driven by 220V AC and controlled by switches. When connecting switch to 220V AC directly, people will be in danger once electricity leakage happens. From a safety perspective, we specially designed this relay module with NO (normally open) and NC (normally closed) terminals. In this lesson, we will learn a special and easy-to-use switch, which is the relay module. Let's get started.

##### (2)Components Needed



##### (3)Knowledge about Component

###### Relay:

It is an “automatic switch” that uses a small current to control the operation of a large current.

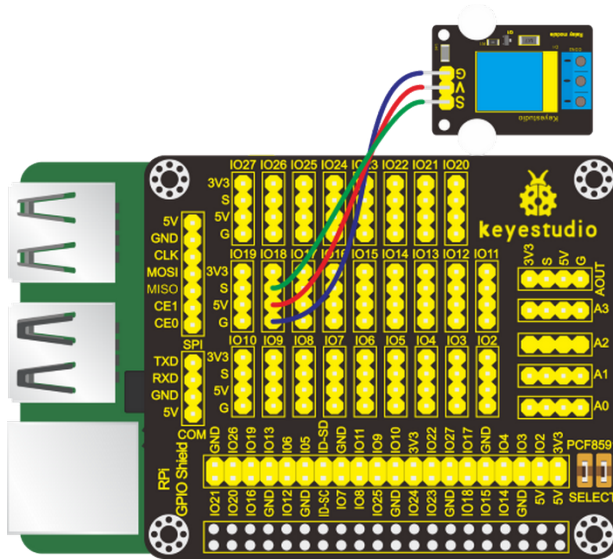
Control input voltage: 5V

Rated load: 5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

Rated load: You can use the 5V voltage of the Raspberry Pi to control a device with a DC voltage of 24V or an AC voltage of 250V.

##### (4)Connection Diagram

Relay Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson18_Relay
```

```
gcc Relay.c -o Relay -lwiringPi
```

```
sudo ./Relay
```

#### (6) Test Results

After running the program, the LED on the relay lights and then dims and repeats this pattern.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>

#define relayPin 1 //BCM GPIO 18

int main()
{
    wiringPiSetup();
    pinMode(relayPin, OUTPUT);

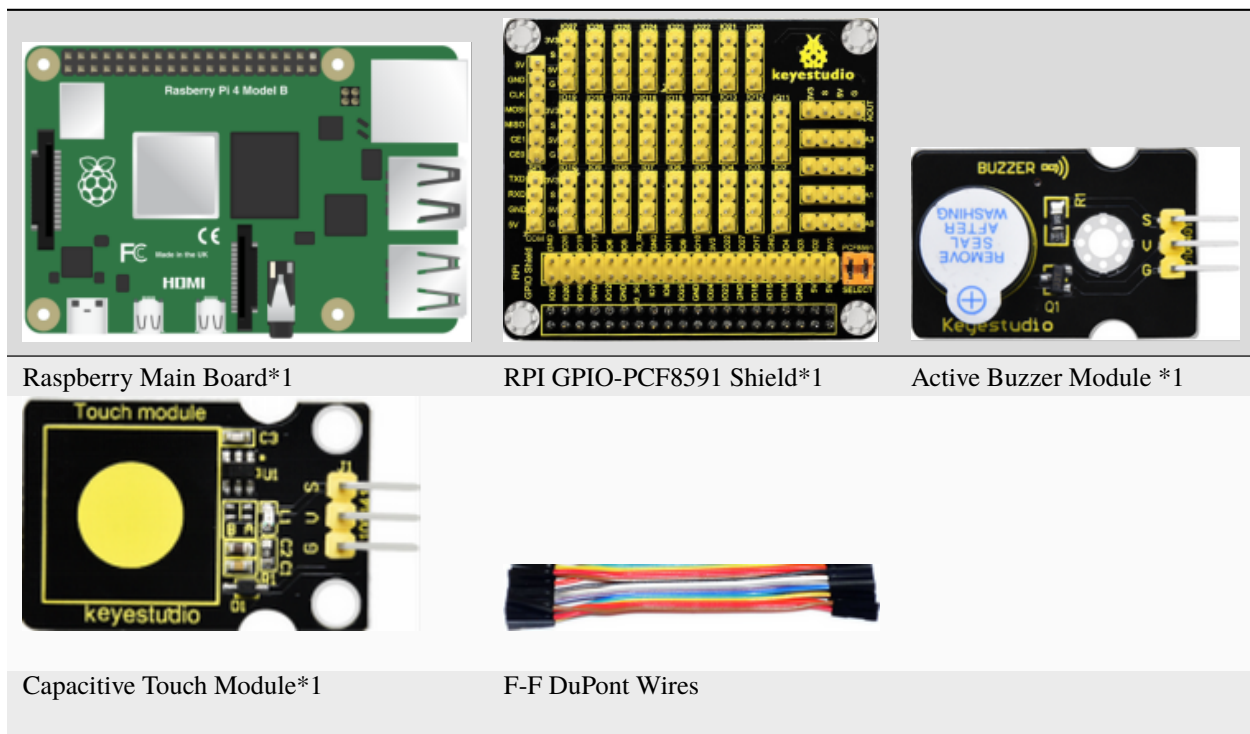
    while(1)
    {
        digitalWrite(relayPin, HIGH);
        printf("turn on\n");
        delay(5000);
        digitalWrite(relayPin, LOW);
        printf("turn off\n");
        delay(1000);
    }
}
```

### 4.4.19 Project19Touch-sensitive Alarm

#### (1)Description

Touch-sensitive alarm is very commonplace in daily life, especially found in home anti-theft and car anti-theft systems. When someone touches the alarming mental material, the device alarms to warn people. And it is of high sensitivity and high reliability evidenced by issuing alarm the moment it is touched.

#### (2)Components Needed



#### (3)Knowledge about Component:

##### Capacitive Touch Module

It mainly uses touch detection IC and can be found in many electronic devices. It uses the most popular capacitive sensing technology, just like the smart buttons on your phone. The touching area of this small sensor can feel the touch of humans and metals by responding with high or low level. It can still detect the touch though covered by a piece of paper and cloth. The sensitivity reduces with the increase of items between the touch-sensitive area and the object performing the touch.

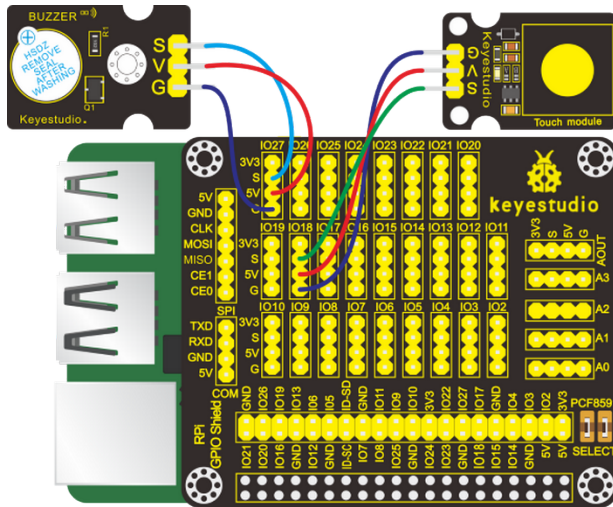
The touch detection IC is designed to replace the traditional button with a variable area key, featuring low power consumption and wide operating voltage.

When the module is powered up, it needs a stabilization time of about 0.5 sec. During this time period, do not touch the keypad. At this time, all functions are disabled, and self-calibration is always performed. No touching the key, the recalibration period is about 4.0sec.

Capacitive touch sensors are used in many devices such as digital audio players, computer displays, mobile phones, mobile devices, tablets and others.

#### (4)Connection Diagram

Active Buzzer Module	RPI Shield	GPIO-PCF8591	Capacitive Touch Sensor	RPI Shield	GPIO-PCF8591
S	SIO27		S	SIO18	
V	5V		V	5V	
G	G		G	G	



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson19_Touch_Alarm
```

```
gcc Touch_Alarm.c -o Touch_Alarm -lwiringPi
```

```
sudo ./Touch_Alarm
```

#### (6) Test Results

After running the program, when the sensing area on the capacitive touch sensor is touched, the terminal outputs 1 and the buzzer makes sounds; otherwise, the terminal outputs 0 and the buzzer is in silence.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#define touchPin 1 //BCM GPIO 18
#define buzPin 2 //define buzzer pin BCM GPIO 27

int main()
{
    wiringPiSetup();
    char val;
    {
        pinMode(touchPin, INPUT);
```

(continues on next page)

(continued from previous page)

```

    pinMode(buzPin,OUTPUT);
}

while(1)
{
    val=digitalRead(touchPin);
    printf("val = %d\n",val);
    if(val==1) //When the touch area is touched
        digitalWrite(buzPin,HIGH); //Buzzer turn on
    else
        digitalWrite(buzPin,LOW); //Buzzer turn off
}
}

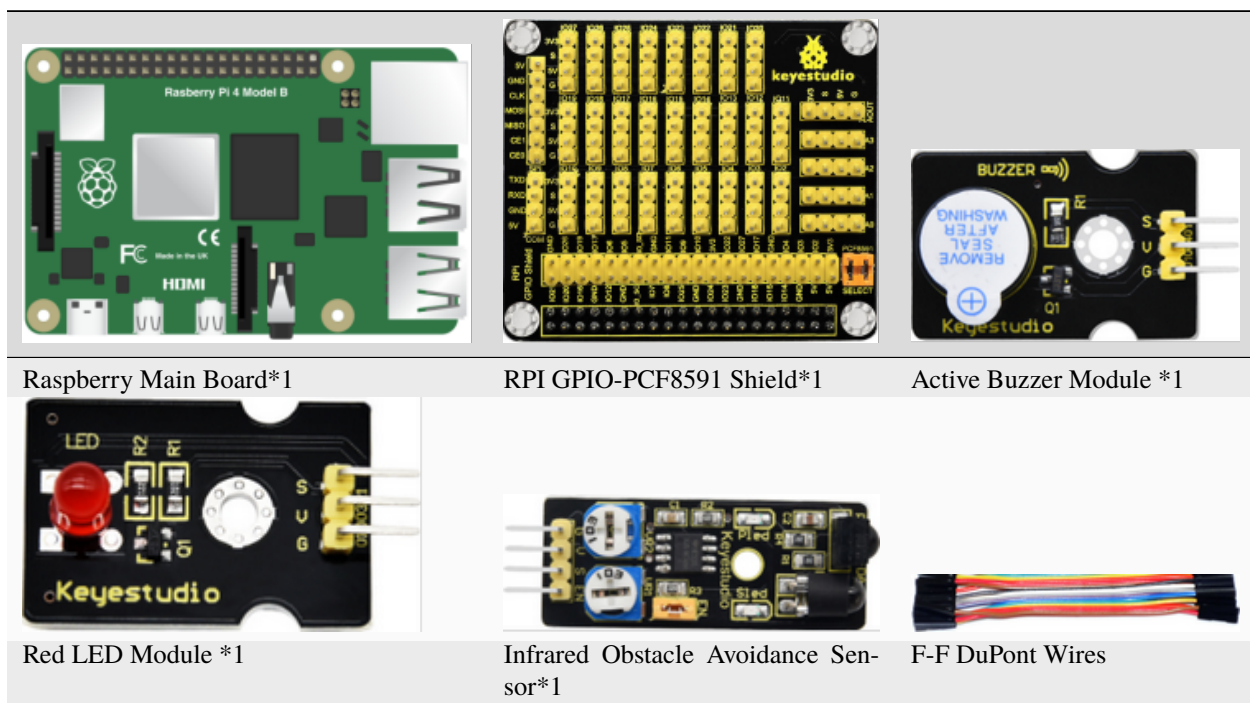
```

#### 4.4.20 Project 20Obstacle Avoidance Sensor

##### (1)Description

You may have seen that a smart car automatically avoid the obstacles around it. How did it make it? The credit comes to an infrared obstacle avoidance sensor. And in this project, we intend to learn about this infrared obstacle avoidance sensor.

##### (2)Components Needed



Raspberry Main Board\*1

RPI GPIO-PCF8591 Shield\*1

Active Buzzer Module \*1

Red LED Module \*1

Infrared Obstacle Avoidance Sensor\*1

F-F DuPont Wires

##### (3)Knowledge about Component

##### Infrared Obstacle Avoidance Sensor

It is equipped with distance adjustment function and is especially designed for wheeled robots. This sensor has strong adaptability to ambient light and is of high precision. It has a pair of infrared transmitting and receiving tube.

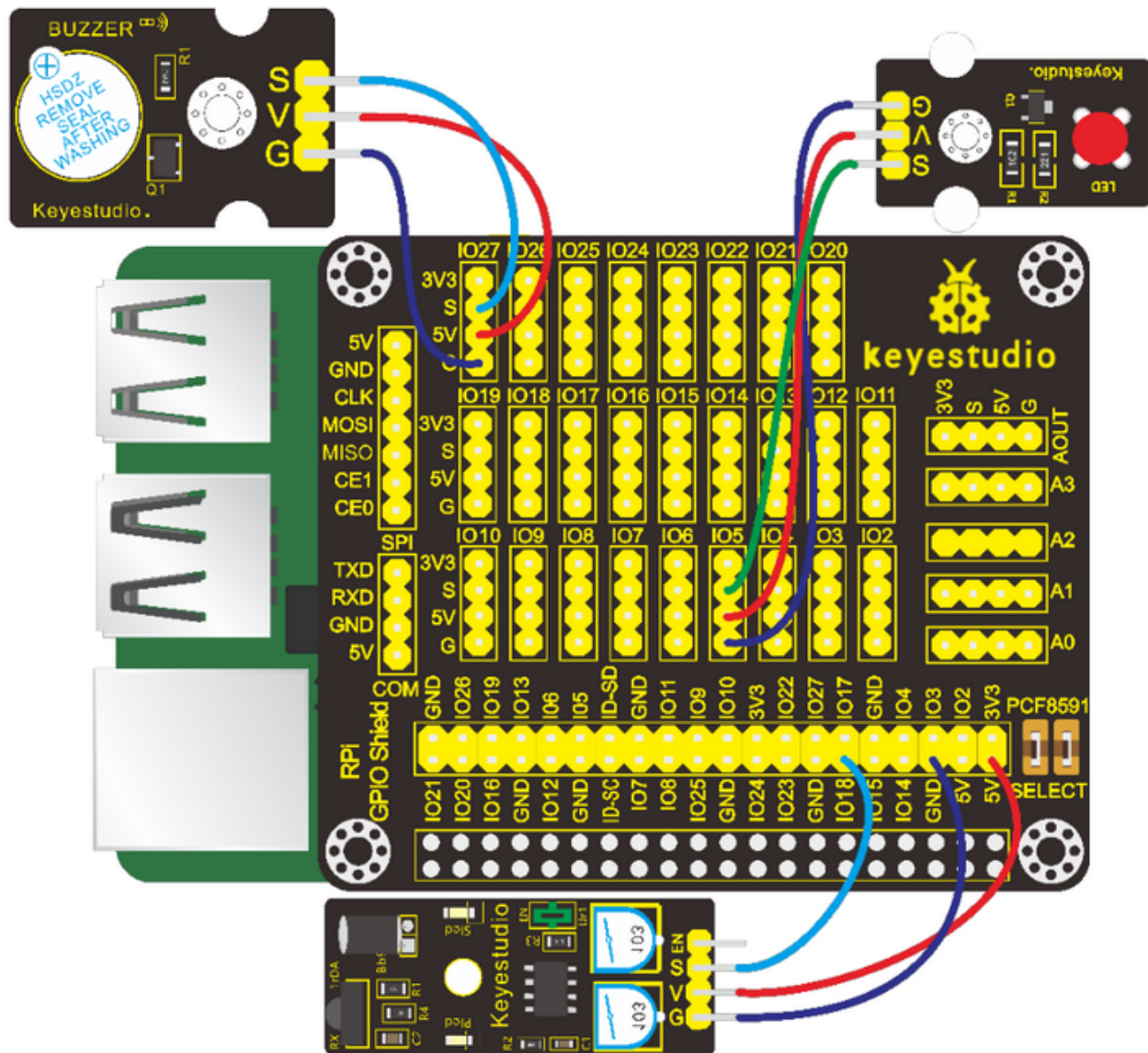
When infrared ray launched by the transmitting tube encounters an obstacle (its reflector), the infrared ray is reflected to the receiving tube, and the signal terminal outputs 0 (low level); if no objects is detected, the infrared signal decreases with the increase of distance and finally dies out so the receiving tube receives no signals and the the terminal outputs 1(high level). That's how it determines whether there are obstacles around.

We can adjust the detection distance through the potentiometer knob (effective distance: 240cm, working Voltage: 3.3V-5V ).

(4)Connection Diagram

Active Module	Buzzer	RPI Shield	GPIO-PCF8591	Infrared Sensor	Obstacle Avoidance	RPI Shield	GPIO-PCF8591
S		SIO27		S		SIO18	
V		5V		V		5V	
G		G		G		G	
Red LED Module		RPI Shield	GPIO-PCF8591				
S		SIO5					
V		5V					
G		G					





#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson20_Obstacle_Avoidance
```

```
gcc Obstacle_Avoidance.c -o Obstacle_Avoidance -lwiringPi
```

```
sudo ./Obstacle_Avoidance
```

#### (6) Test Results

After running the program, when the sensor detects any obstacles, the terminal outputs 0, the buzzer makes sounds and the LED keeps flashing; otherwise, the terminal outputs 1, the buzzer utters no sounds and the LED reminds off.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>
```

(continues on next page)

(continued from previous page)

```
#define obstaclePin 1 //BCM GPIO 18
#define buzPin 2 //define buzzer pin BCM GPIO 27
#define ledPin 21 //define led pin BCM GPIO 5

int main()
{
    wiringPiSetup();
    char val;
    {
        pinMode(obstaclePin, INPUT);
        pinMode(buzPin, OUTPUT);
        pinMode(ledPin, OUTPUT);
        digitalWrite(buzPin, LOW);
        digitalWrite(ledPin, LOW);
    }

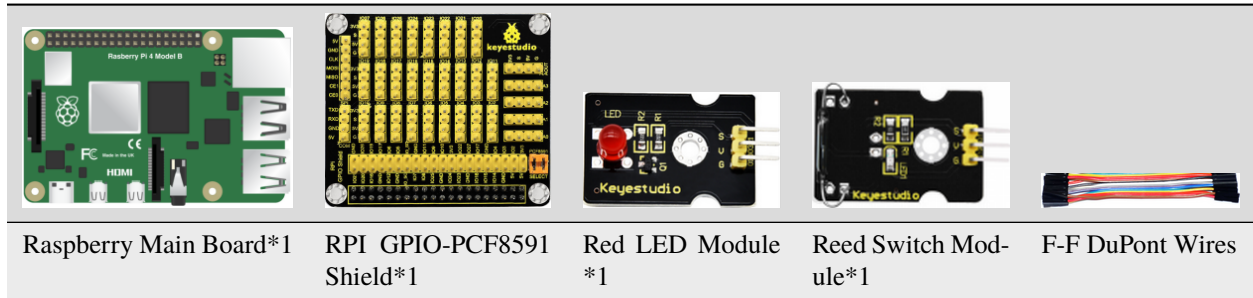
    while(1)
    {
        val=digitalRead(obstaclePin);
        printf("val = %d\n",val);
        if(val==0) //When the obstacle avoidance is detected
        {
            digitalWrite(buzPin,HIGH); //Buzzer turn on
            digitalWrite(ledPin,HIGH); //LED turn on
            delay(100);
            digitalWrite(ledPin,LOW); //LED turn off
            delay(100);
        }
        else
        {
            digitalWrite(buzPin,LOW); //Buzzer turn off
            digitalWrite(ledPin,LOW); //LED turn off
        }
    }
}
```

#### 4.4.21 Project 21 Reed Switch Module

##### (1)Description

In this project, we will learn to detect whether there is magnetic force around with reed switch sensor and Raspberry Pi. Actually, we have known how to detect magnetic force with Hall magnetic sensor. Then what's the difference between these two sensors? You will have the answer after learning this lesson.

##### (2)Components Needed



### (3)Knowledge about Component:

#### Reed Switch Module

The magnetic reed sensor is mainly composed of a magnetic reed switch, which is a mechanical magnetic switch and also a contact switch, and a passive device.

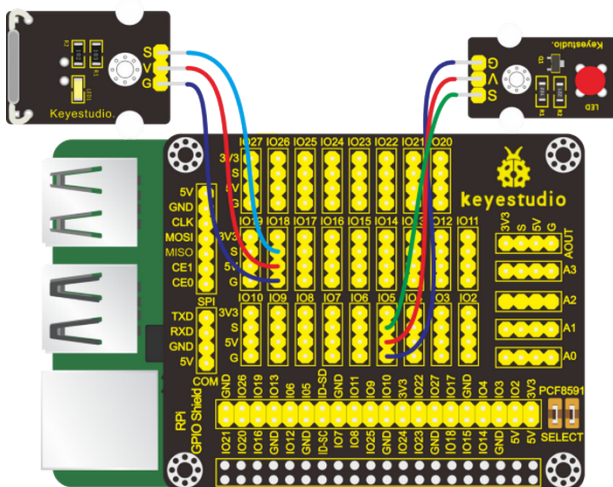
Its working principle is to use a magnetic field to magnetize the reed so as to control the on and off status of the switch. When the sensor is exposed to a magnetic field, the two ferromagnetic blades inside the switch pull together and the switch closes. When the magnetic field is removed, the two blades separate and the switch opens.

However, since the reed switch is a contact switch, it has a limited service life and is easily damaged during transportation and installation.

The reed switch is applied widely in household appliances, automobile, communication, industrial manufacturing, health care and security, as well as other electronic devices like door magnet, reed relay and level gauge.

#### (4)Connection Diagram

Red LED Module	RPi GPIO-PCF8591 Shield	Reed Switch Module	RPi GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



#### (5)Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson21_Reed_Switch
gcc Reed_Switch.c -o Reed_Switch -lwiringPi
sudo ./Reed_Switch
```

#### (6)Test Results

After running the program, when the reed switch sensor detects magnetic field nearby, the terminal prints the “0” detected by the sensor and the LED lights; otherwise, the terminal prints the number 1 and the LED reminds off.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define reed_pin 1 //reed pin BCM GPIO 18
#define led_pin 21 //LED pin BCM GPIO 5

int main(void)
{
    int val = 0;
    wiringPiSetup();
    pinMode(reed_pin,INPUT); //set reed reedPin INPUT mode
    pinMode(led_pin,OUTPUT); //set ledPin OUTPUT mode

    while(1)
    {
        val=digitalRead(reed_pin);
        printf("val = %d\n",val);
        if(val==0) //when magnetism is detected
            digitalWrite(led_pin,HIGH); //led on
        else
            digitalWrite(led_pin,LOW); //led off
    }
}
```

## 4.4.22 Project 22Vibration Alarm

#### (1)Description

In this project, we will make a simply equipped vibration alarm with a vibration sensor and a buzzer.

#### (2)Components Needed



sudo ./Vibrating\_Alarm

#### (6)Test Results

After running the program, when the vibration sensor is triggered, the terminal keeps printing “buzzer ring.....buzzer off”and the buzzer rings constantly; otherwise, the terminal prints “...buzzer off”and the buzzer becomes silent.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <stdio.h>

#define vibPin 1 //vibration pin BCM GPIO 18
#define buzPin 21 //buzzer pin BCM GPIO 5
int buz_status = 0;

void swbuz(void)
{
    buz_status = ~buz_status;
    digitalWrite(buzPin, buz_status);
    if(buz_status == 1)
    {
        printf("buzzer ring ...");
    }
    else
    {
        printf("...buzzer off");
    }
}

int main()
{
    wiringPiSetup();
    pinMode(buzPin, OUTPUT);
    pinMode(vibPin, INPUT);
    pullUpDnControl(vibPin, PUD_UP);
    wiringPiISR(vibPin, INT_EDGE_FALLING, swbuz); //interrupt

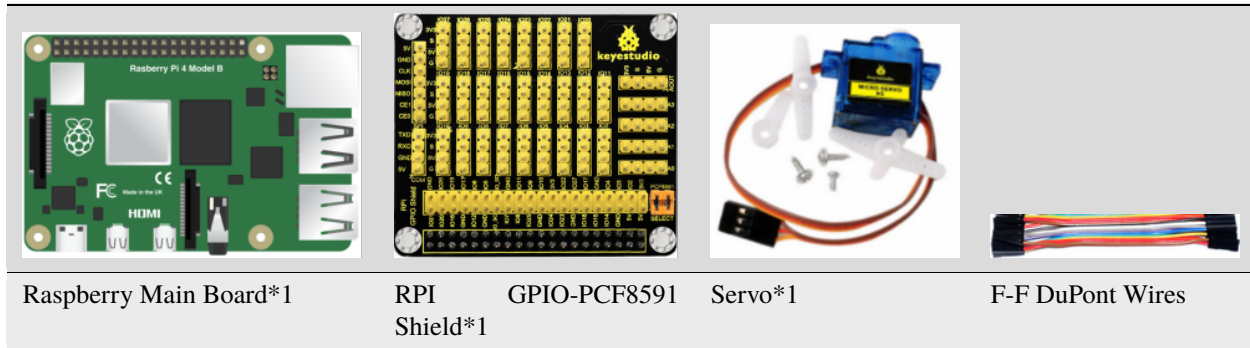
    while(1)
    {
        //val=digitalRead(vibPin); //Receive
        //printf("value = %d\n", val);
    }
}
```

### 4.4.23 Project 23Servo

#### (1)Description

Relay Module is applied widely, especially for robot like human robots and moving robots. In this lesson, we will learn how it works.

#### (2)Components Needed

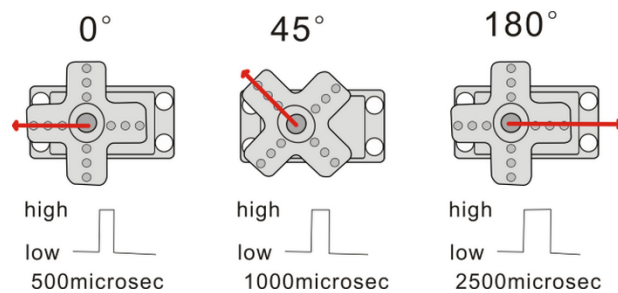


#### (3)Knowledge about Component

##### Servo:

A location(angle) driver which can rotate a certain angle with high accuracy. It has three external wires which are brown, red and orange. Brown one is grounded, red one is positive pole of power and orange one is signal wire.

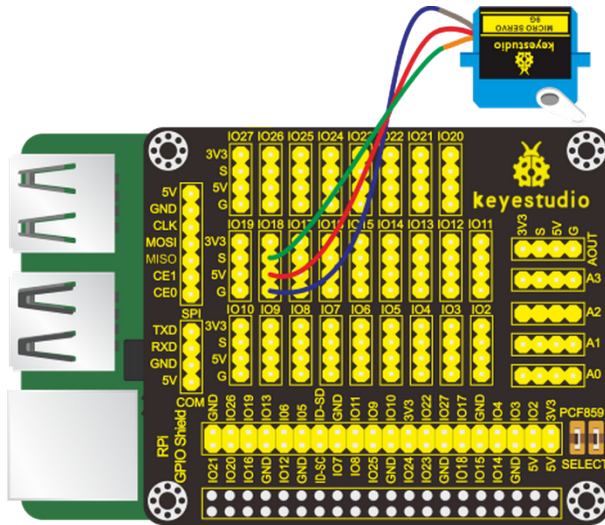
The rotation angle of the servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms(50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle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



#### (4)Connection Diagram

Servo	RPI GPIO-PCF8591 Shield
Orange Wire	SIO18
Red Wire	5V
Brown Wire	G





#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson23_Relay Module
```

```
gcc Relay Module.c -o Relay Module -lwiringPi
```

```
sudo ./Relay Module
```

#### (6) Test Results

Servo rotates in the range of 0°-180°.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#define serPin 1 //servo pin BCM GPIO 18

int main()
{
    wiringPiSetup();
    pinMode(serPin,OUTPUT);
    int i;
    for(;;)
    {
        for(i=0;i<50;i++)
        {
            digitalWrite(serPin,HIGH);
            delayMicroseconds(500); //Pulse width 0.5ms, Angle 0
            digitalWrite(serPin,LOW);
            delay(20-0.5);          //Cycle 20 ms
        }

        delay(1000);
        for(i=0;i<50;i++)
        {
            digitalWrite(serPin,HIGH);
```

(continues on next page)



(continued from previous page)

```

        delayMicroseconds(2500);
        digitalWrite(serPin,LOW);
        delay(20-2.5);
    }
    delay(1000);
}
return 0;
}

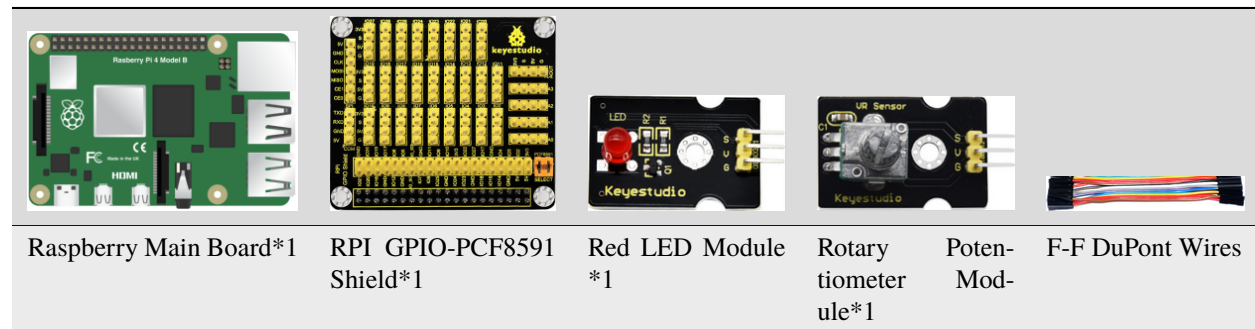
```

#### 4.4.24 Project 24 Adjust the Brightness of LED

##### (1)Description

Some of the lamps on market can be adjusted to display different brightness, which gives us better shopping experiences. And in this project, we will learn how to make this happen.

##### (2)Components Needed



##### (3)Knowledge about Components

###### PCF8591 A/D converter chip:

It is installed behind the RPI GPIO-PCF8591 shield with voltage resolution of 5V/255 0.01961.

Since the Raspberry Pi itself does not have AD/DA function, an expansion board with this function is required when it is connected to external analog sensors. And here we use PCF8591 A/D converter with I2C communication.

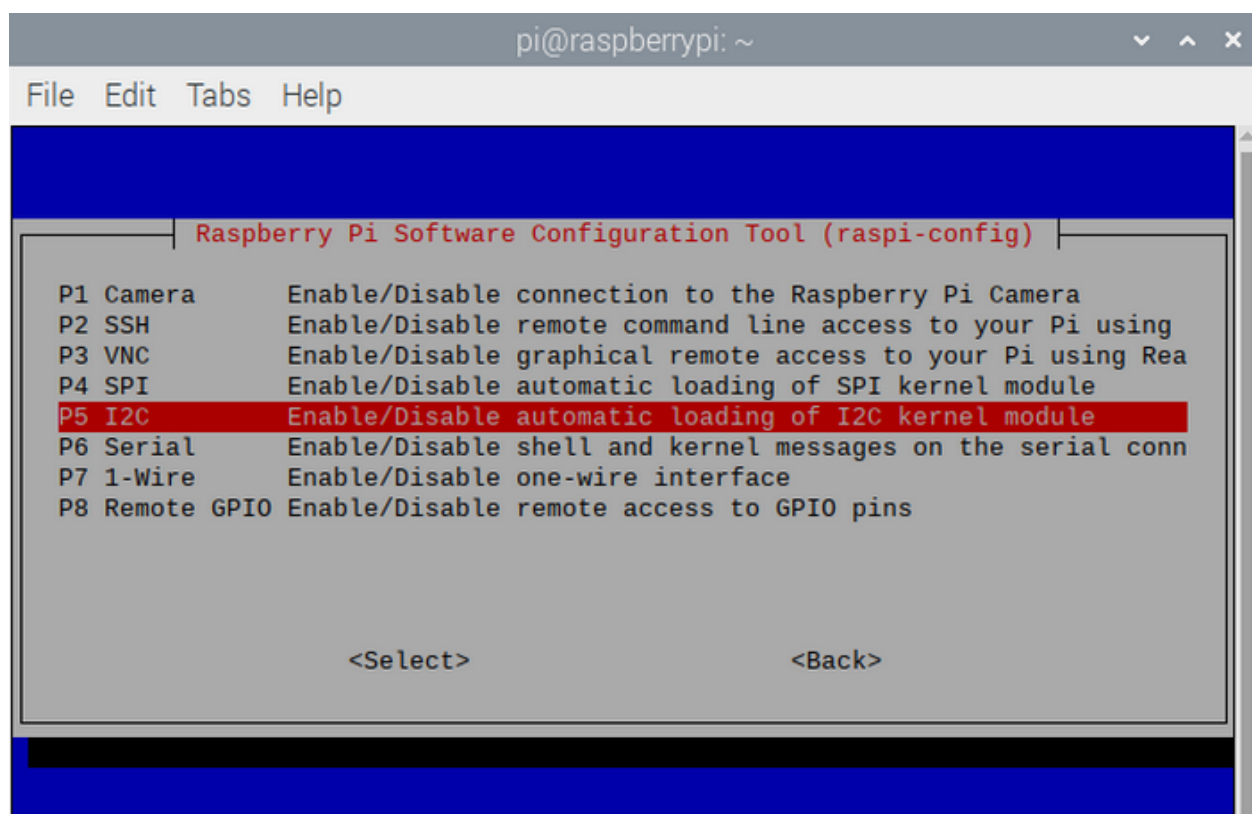
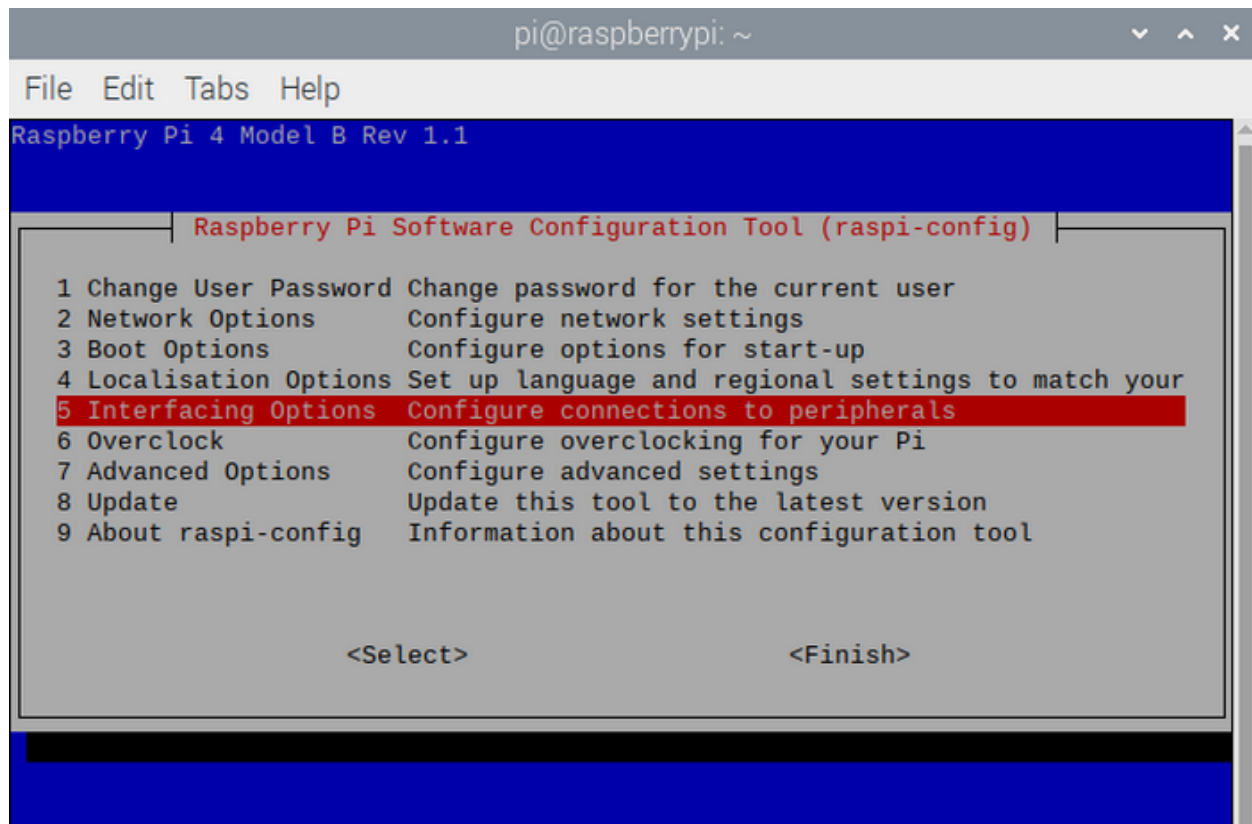
1. Enable the I2C communication function of the Raspberry Pi as follows:
2. Raspberry Pi does not enable the I2C function by default. Enter `sudo raspi-config` in the terminal to enter the Raspberry Pi configuration interface.

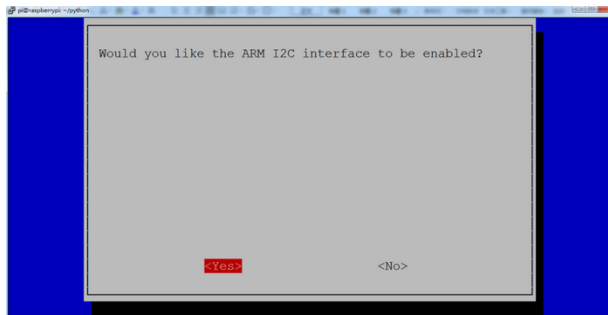
```

pi@raspberrypi:~/python $ sudo raspi-config

```

Enable the I2C function of Raspberry Pi as follows(Use the up (↑), down (↓), left (←), and right (→) keys on the keyboard to select the corresponding option, and then press “Enter”):





Find more about I2C:

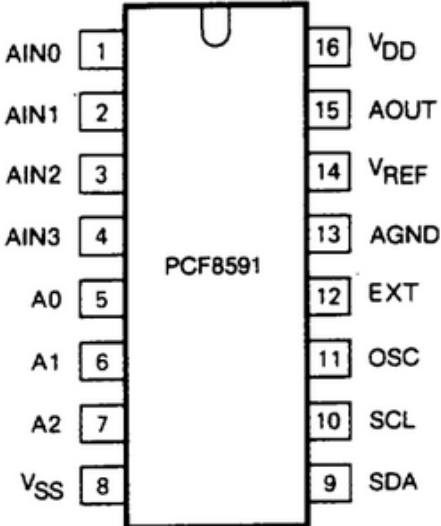
<https://www.nxp.com/docs/en/user-guide/UM10204.pdf>

#### Pin description:

You can find more information, such as the specification of this chip, in the resources link:

<https://fs.keyestudio.com/KS3016>

From the picture below, it is obvious that the PCF8591 converter is equipped with a AOUT pin and 4 analog inputs pins A0~A3

SYMBOL	PIN	DESCRIPTION	TOP VIEW
AIN0	1	Analog inputs (A/D converter)	
AIN1	2		
AIN2	3		
AIN3	4		
A0	5	Hardware address	
A1	6		
A2	7		
Vss	8	Negative supply voltage	
SDA	9	I2C-bus data input/output	
SCL	10	I2C-bus clock input	
OSC	11	Oscillator input/output	
EXT	12	external/internal switch for oscillator input	
AGND	13	Analog ground	
Vref	14	Voltage reference input	
AOUT	15	Analog output(D/A converter)	
Vdd	16	Positive supply voltage	

Check the address of the I2C module (PCF8591) connected to the Raspberry Pi, enter the command: `i2cdetect -y 1`, and then press "Enter".

From below picture, it is known that the I2C address is 0x48 .

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~ $

```

The address for reading pins A0~A3 is:

A0 = 0x40 ##A0 —> port address

A1 = 0x41

A2 = 0x42

A3 = 0x43

The address for analog output pin AOUT is: 0x40, which is 64 when hexadecimal is converted to decimal.

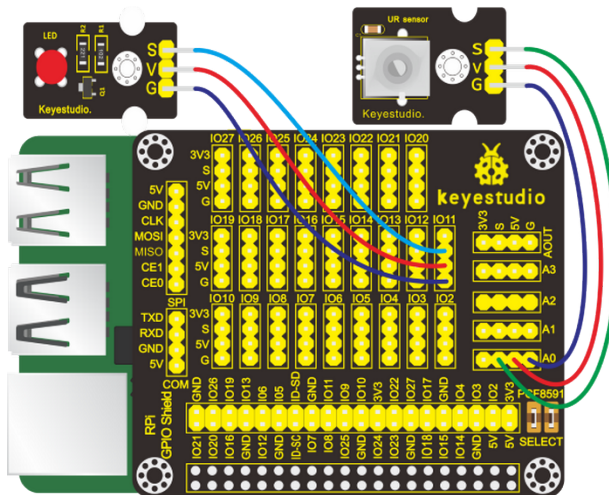
### Rotary Potentiometer

It can be viewed as an adjustable resistor with the range from 0~10K.

Therefore when we rotate the potentiometer, we actually change its resistance. We can build a circuit to convert the changes in the resistance to the changes in voltage. Then input the voltage changes to the GPIO analog input port for detection through the signal terminal of the module.

(4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Rotary Potentiometer	RPI GPIO-PCF8591 Shield
S	SIO11	S	SA0
V	5V	V	5V
G	G	G	G



#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson24_Potentiometer_LED
```

```
gcc Potentiometer_LED.c -o Potentiometer_LED -lwiringPi
```

```
sudo ./Potentiometer_LED
```

#### (6) Test Results

After running the program, the terminal prints the analog value of the rotary potentiometer and the brightness of the LED changes with the adjustments of the potentiometer.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <pcf8591.h> //pcf8591 library
#include <softPwm.h>
#include <stdio.h>

#define Address 0x48 //iic address
#define BASE 64 //DAC write address
#define A0 BASE+0 //A0 analogRead address
#define A1 BASE+1 //A1 analogRead address
#define A2 BASE+2
#define A3 BASE+3

#define ledPin 14 //led pin BCM GPIO 11

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address); //Initialize the pcf8591
    softPwmCreate(ledPin,0,100);

    while(1)
```

(continues on next page)

(continued from previous page)

```

{
    value=analogRead(A0);    //read the ADC value of channel 0
    softPwmWrite(ledPin,value*100/255); // Mapping to PWM duty cycle
    printf("A0:%d\n",value);
    analogWrite(BASE,value);    //write the DAC value
    printf("AOUT:%d\n",value);
    delay(100);
}
}

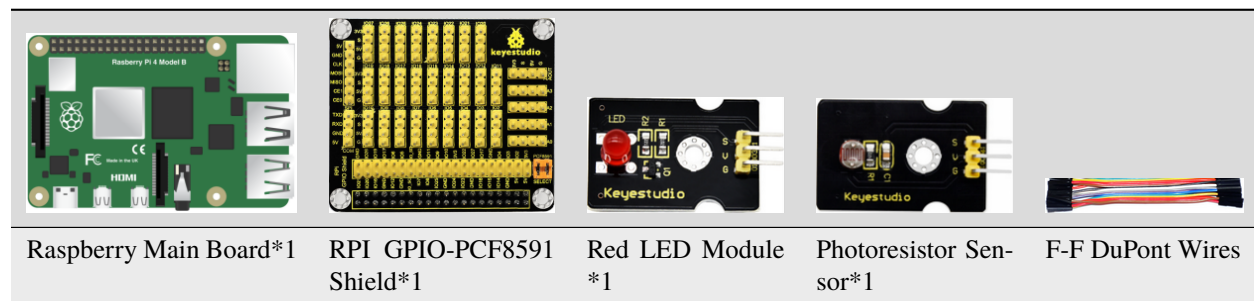
```

#### 4.4.25 Project 25Photoresistor

##### (1)Description

Sensors or modules have found wide applications in daily life. For example, some street lights automatically light up when it is dark and are blocked out when it is bright. But why? Actually it is because of an element called photoresistor which can make changes with the light intensity. And in this project, we will learn how to make this happen.

##### (2)Components Needed



##### (3)Knowledge about Component

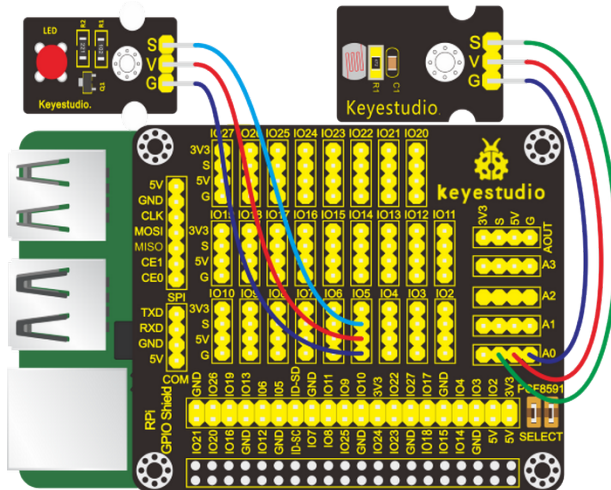
###### Photoresistor

Photoresistor (Photovaristor) is a resistor whose resistance varies according to different incident light strengths. It's made based on the photoelectric effect of semiconductor. If the incident light is intense, its resistance reduces; if the incident light is weak, the resistance increases.

If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance.

##### (4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Photoresistor Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SA0
V	5V	V	5V
G	G	G	G



#### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson25_Photo_Sensor
```

```
gcc Photo_Sensor.c -o Photo_Sensor -lwiringPi
```

```
sudo ./Photo_Sensor
```

#### (6)Test Results

Terminal prints the value tested by photoresistor. LED will lights up if the ambient environment is dim; otherwise, LED will be off.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

#define ledPin 21 //led pin BCM GPIO 5

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);
    pinMode(ledPin,OUTPUT);
```

(continues on next page)



(continued from previous page)

```

while(1)
{
    value=analogRead(A0);
    printf("A0:%d\n",value);
    delay(100);
    if(value>100)
        digitalWrite(ledPin,HIGH);
    else
        digitalWrite(ledPin,LOW);
}
}

```

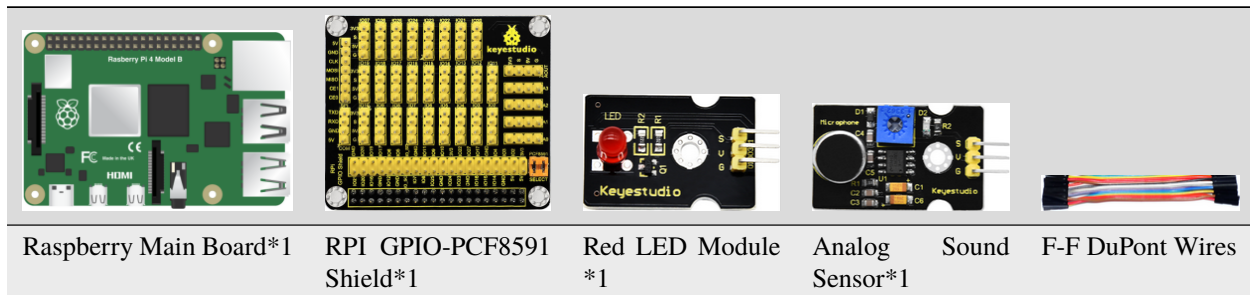
#### 4.4.26 Project 26Sound-activated Light

##### (1)Description

You might find the lights automatically light up when you pass them. And they will be off if the surrounding is quiet. Do you know why?

Actually, it is sound sensor that controls them on and off.

##### (2)Components Needed:



##### (3)Knowledge about Component

###### Sound Sensor

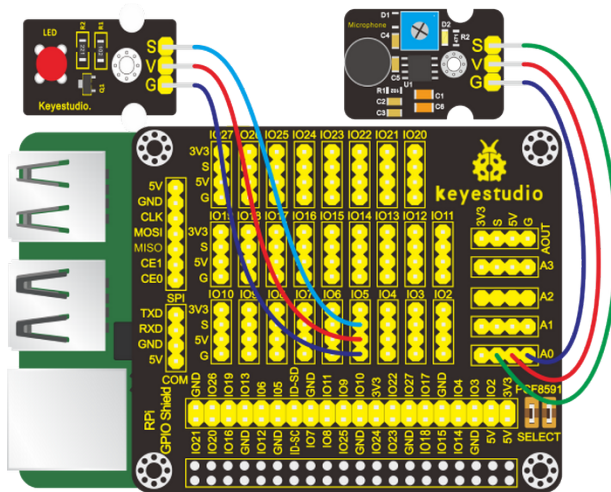
A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals.

The sound sensor has a built-in capacitive electret microphone which is highly sensitive to sound. Sound waves cause the thin film of the electret to vibrate and then the capacitance changes, thus producing the corresponding changed voltage. Since the voltage change is extremely weak, it needs to be amplified. So it is converted into a voltage ranging from 0 to 5V, which is received by data acquisition unit after A/D adapter conversion and then sent to an MCU.

##### (4)Connection Diagram



Red LED Module	RPI GPIO-PCF8591 Shield	Analog Sound Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SA0
V	5V	V	5V
G	G	G	G



#### (5) Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press "Enter". If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/C_code/lesson26_Sound_Led
gcc Sound_Led.c -o Sound_Led -lwiringPi
sudo ./Sound_Led
```

#### (6) Test Results

When you clap your hands suddenly, LED lights up and clap again, LED is off.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
```

(continues on next page)

(continued from previous page)

```
#define A3 BASE+3

#define ledPin 21 //led pin //BCM GPIO 5

int count = 0;
int flag = 0;

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);
    pinMode(ledPin,OUTPUT);

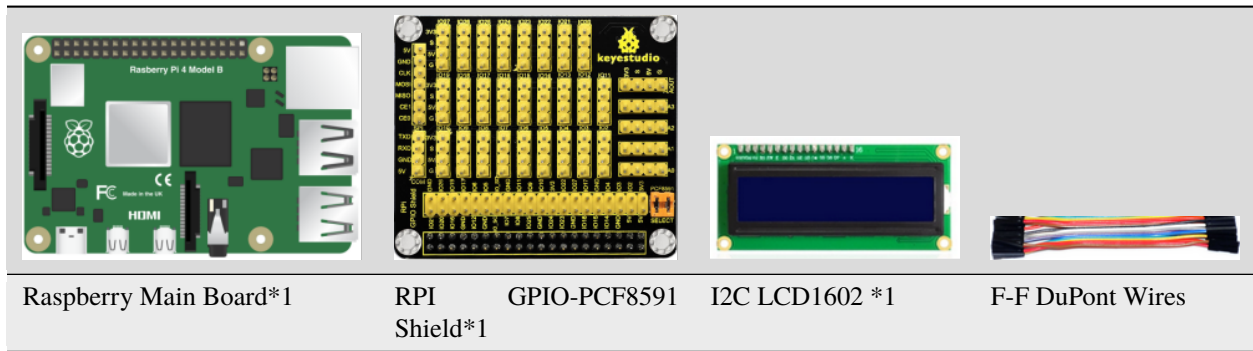
    while(1)
    {
        value=analogRead(A0); //Read the value of the sound sensor
        printf("A0:%d\n",value);
        delay(100);
        if(value>80)
        {
            count = count + 1;
            flag = count % 2;
        }
        if(flag == 1)
        {
            digitalWrite(ledPin,HIGH);
        }
        else
        {
            digitalWrite(ledPin,LOW);
        }
    }
}
```

#### 4.4.27 Project 27I2C LCD1602

##### (1)Description

Liquid crystal display can be used to conduct all kind of experiments and make various DIY items. For example, you can make a temperature detection device out of a temperature sensor and a LCD and distance measurement equipment with an ultrasonic module and a LCD. In this project, we will connect a LCD 1602 with Raspberry Pi and use it to show characters and numbers.

##### (2)Components Needed

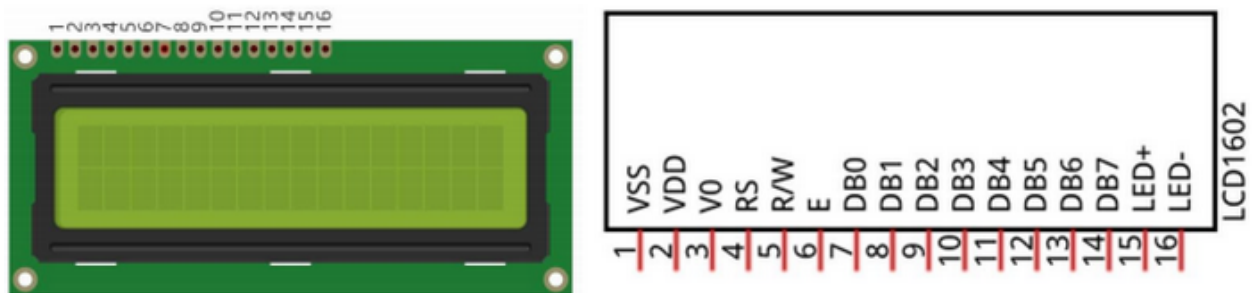


### (3) Knowledge about Component

#### LCD1602 LED Display

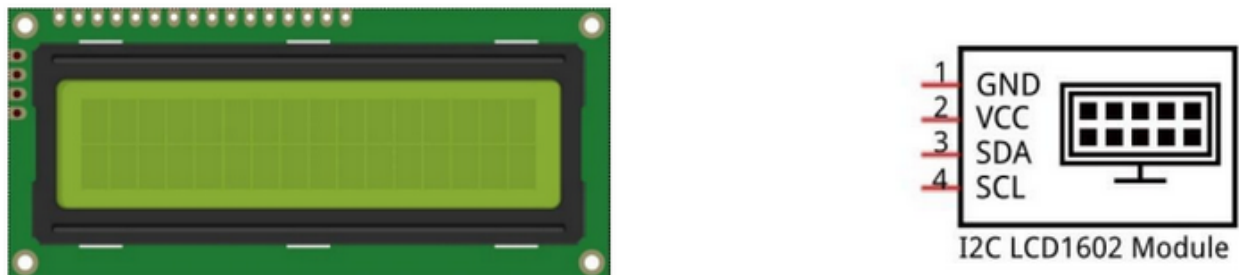
This I2C LCD 1602 could show the characters or numbers in 16 rows and 2 columns

The following is a monochrome LCD1602 display screen and its pin diagram:



The I2C LCD1602 display integrates I2C interface which can be connected with serial input and parallel output pins so as to transmit data to the display.

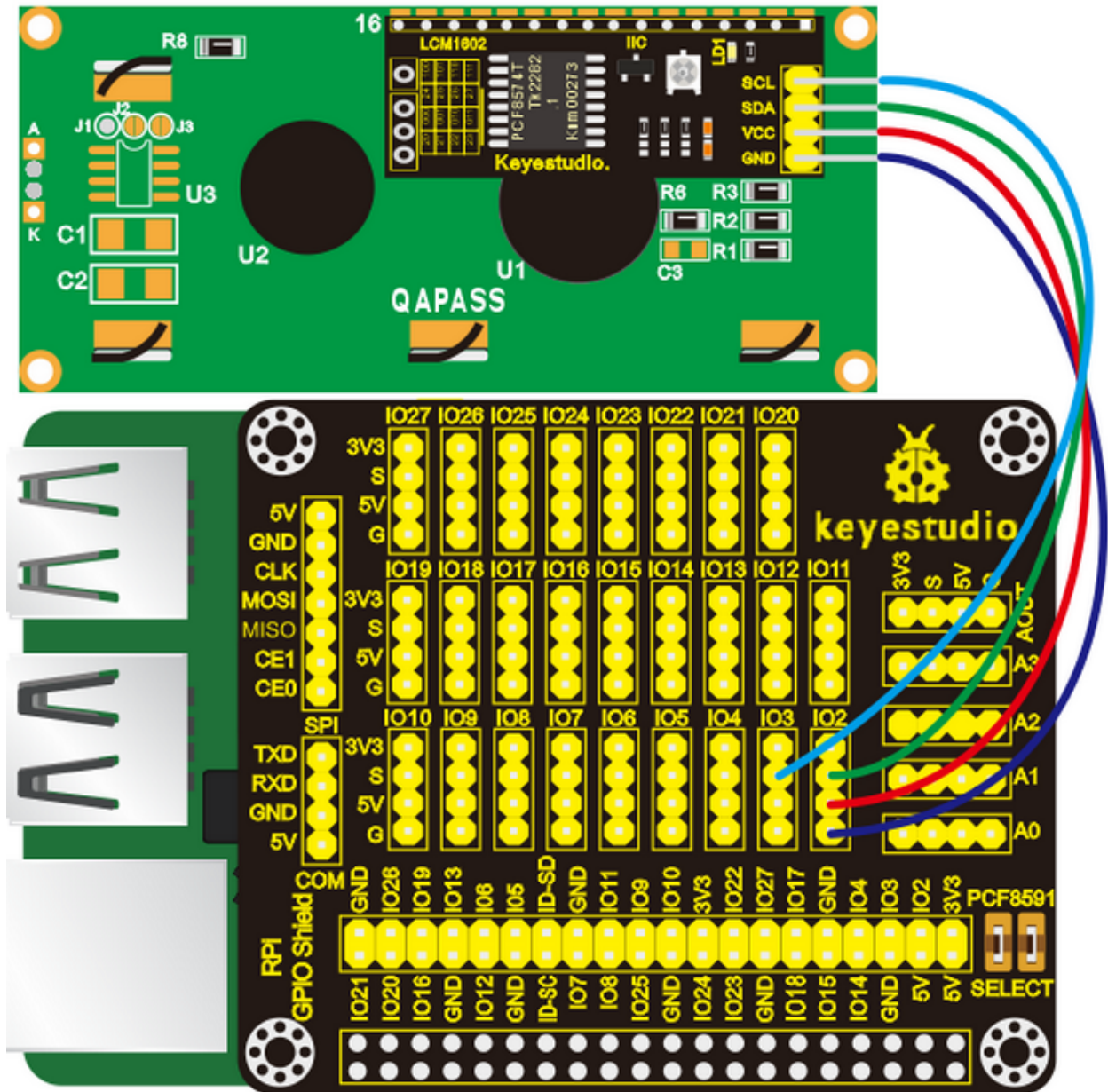
This allows us to operate the LCD1602 with 4 lines.



The IC chip used in this module is PCF8574T (PCF8574AT) and its default IC address 0x27(0x3F). You can also check the RPI bus on your I2C device address with the command “i2cdetect -y 1”.

### (4) Connection Diagram

I2C LCD1602 Module	RPI GPIO-PCF8591 Shield
GND	GND
VCC	5V
SDA	IO2
SCL	IO3



#### (5) Run Example Code

**Note:** in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson27_I2CLCD1602
gcc I2CLCD1602.c -o I2CLCD1602 -lwiringPiDev -lwiringPi
sudo ./I2CLCD1602
```

#### (6) Test Results

After running the program, the LCD 1602 shows the CPU temperature and the system time of your Raspberry Pi.

**Note:** After the program is executed, if you can't see anything on the display or the display is not clear, try to slowly turn the blue knob on the back of the LCD1602 to adjust the contrast until the screen can clearly display the time and temperature.

Note: Press Ctrl + C on keyboard and exit code running.

(7)Example Code

```
#include <stdlib.h>
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <pcf8574.h>
#include <lcd.h>
#include <time.h>

int pcf8574_address = 0x27;          // PCF8574T:0x27, PCF8574AT:0x3F
#define BASE 64                      // BASE any number above 64
//Define the output pins of the PCF8574, which are directly connected to the LCD1602 pin.
#define RS      BASE+0
#define RW      BASE+1
#define EN      BASE+2
#define LED     BASE+3
#define D4      BASE+4
#define D5      BASE+5
#define D6      BASE+6
#define D7      BASE+7

int lcdhd; // used to handle LCD
void printCPUtemperature(){// sub function used to print CPU temperature
    FILE *fp;
    char str_temp[15];
    float CPU_temp;
    // CPU temperature data is stored in this directory.
    fp=fopen("/sys/class/thermal/thermal_zone0/temp","r");
    fgets(str_temp,15,fp);          // read file temp
    CPU_temp = atof(str_temp)/1000.0; // convert to Celsius degrees
    printf("CPU's temperature : %.2f \n",CPU_temp);
    lcdPosition(lcdhd,0,0);         // set the LCD cursor position to (0,0)
    lcdPrintf(lcdhd,"CPU:%.2fC",CPU_temp);// Display CPU temperature on LCD
    fclose(fp);
}

void printDateTime(){//used to print system time
    time_t rawtime;
    struct tm *timeinfo;
    time(&rawtime);// get system time
    timeinfo = localtime(&rawtime);//convert to local time
    printf("%s \n",asctime(timeinfo));
    lcdPosition(lcdhd,0,1);// set the LCD cursor position to (0,1)
    lcdPrintf(lcdhd,"Time:%02d:%02d:%02d",timeinfo->tm_hour,timeinfo->tm_min,timeinfo->
    ↪tm_sec); //Display system time on LCD
}

int detectI2C(int addr){
    int _fd = wiringPiI2CSetup (addr);
```

(continues on next page)

(continued from previous page)

```

    if (_fd < 0){
        printf("Error address : 0x%x \n",addr);
        return 0 ;
    }
    else{
        if(wiringPiI2CWrite(_fd,0) < 0){
            printf("Not found device in address 0x%x \n",addr);
            return 0;
        }
        else{
            printf("Found device in address 0x%x \n",addr);
            return 1 ;
        }
    }
}
}
int main(void){
    int i;

    printf("Program is starting ...\n");

    wiringPiSetup();
    if(detectI2C(0x27)){
        pcf8574_address = 0x27;
    }else if(detectI2C(0x3F)){
        pcf8574_address = 0x3F;
    }else{
        printf("No correct I2C address found, \n"
            "Please use command 'i2cdetect -y 1' to check the I2C address! \n"
            "Program Exit. \n");
        return -1;
    }
    pcf8574Setup(BASE,pcf8574_address);//initialize PCF8574
    for(i=0;i<8;i++){
        pinMode(BASE+i,OUTPUT);    //set PCF8574 port to output mode
    }
    digitalWrite(LED,HIGH);    //turn on LCD backlight
    digitalWrite(RW,LOW);    //allow writing to LCD
    lcdhd = lcdInit(2,16,4,RS,EN,D4,D5,D6,D7,0,0,0,0);//
    ↪ initialize LCD and return "handle" used to handle LCD
    if(lcdhd == -1){
        printf("lcdInit failed !");
        return 1;
    }
    while(1){
        printCPUTemperature();//print CPU temperature
        printDateTime();    // print system time
        delay(1000);
    }
    return 0;
}

```

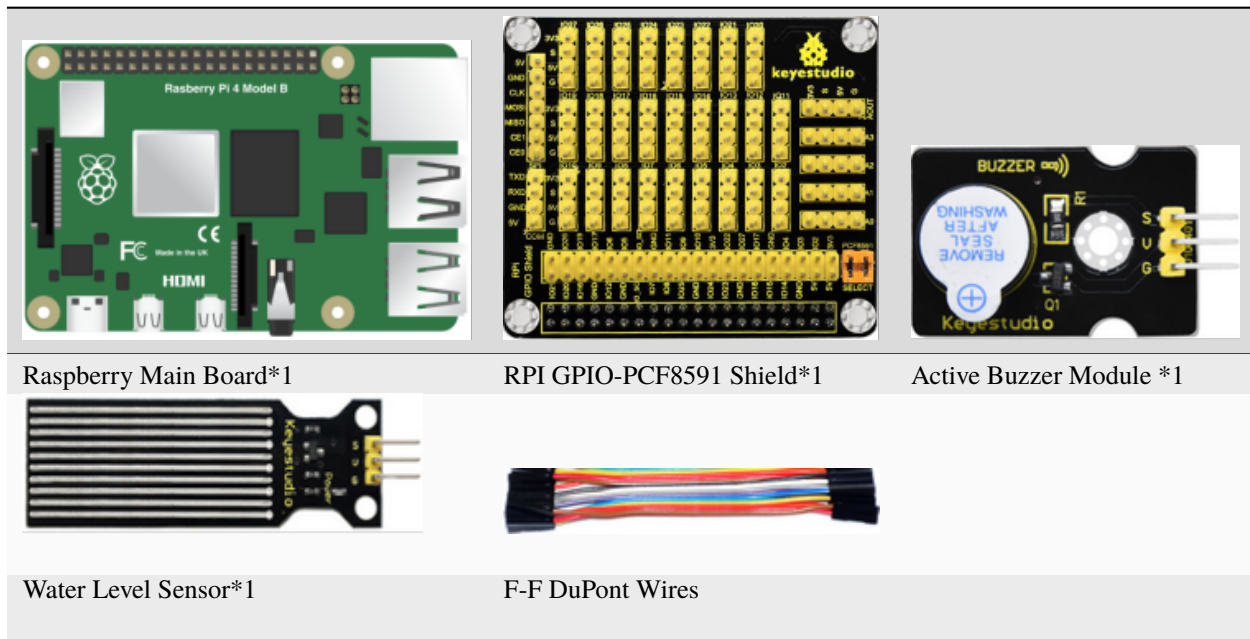


## 4.4.28 Project 28 Water Level Monitor

### (1) Description

In daily life, when there is heavy or even torrential rain, the water level in rivers or reservoirs soars. And when it reaches a certain water level, it is necessary to open the gates to discharge the flood to solve the hidden safety hazards. But how to detect the water level in a river or a reservoir? The answer lies in the water level sensor. In this lesson, we will learn to use this sensor to issue alarms when the water bucket is almost full.

### (2) Components Needed:



Raspberry Main Board\*1

RPI GPIO-PCF8591 Shield\*1

Active Buzzer Module \*1

Water Level Sensor\*1

F-F DuPont Wires

### (3) Knowledge about Component

#### Water Level Sensor

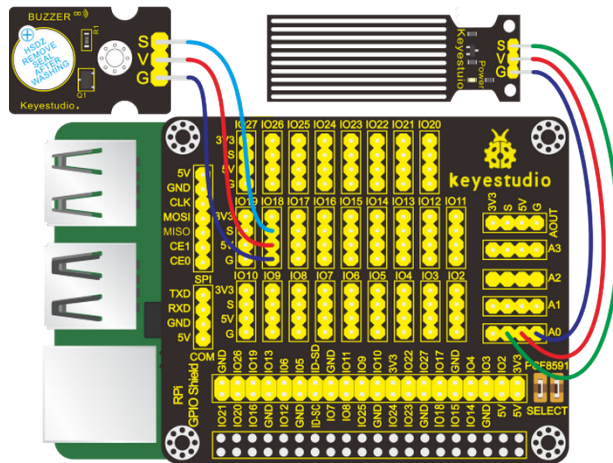
Our water sensor is easy- to-use, portable and cost-effective, designed to identify and detect water level and water drop.

This sensor measures the volume of water drop and water quantity through an array of traces of exposed parallel wires.

It could convert water content to analog signals, and output analog value could be used by function of application. It has the features of low consumption as well.

### (4) Connection Diagram

Active Buzzer Module	RPI Shield	GPIO-PCF8591	Water Level Sensor	RPI Shield	GPIO-PCF8591
S	SIO18		S	SA0	
V	5V		V	5V	
G	G		G	G	



### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson28_Water_Buzzer
```

```
gcc Water_Buzzer.c -o Water_Buzzer -lwiringPi
```

```
sudo ./Water_Buzzer
```

### (6)Test Results

Buzzer makes a sound when water covering the exposed detection part.

Note: Press Ctrl + C on keyboard and exit code running.

### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

#define buzPin 1 //buzzer pin BCM GPIO 18

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);
    pinMode(buzPin,OUTPUT);

    while(1)
```

(continues on next page)



(continued from previous page)

```

{
  value=analogRead(A0); //Read the value of the water sensor
  printf("A0:%d\n",value);
  delay(100);
  if(value>30)
  {
    digitalWrite(buzPin,HIGH);
  }
  else
  {
    digitalWrite(buzPin,LOW);
  }
}
}

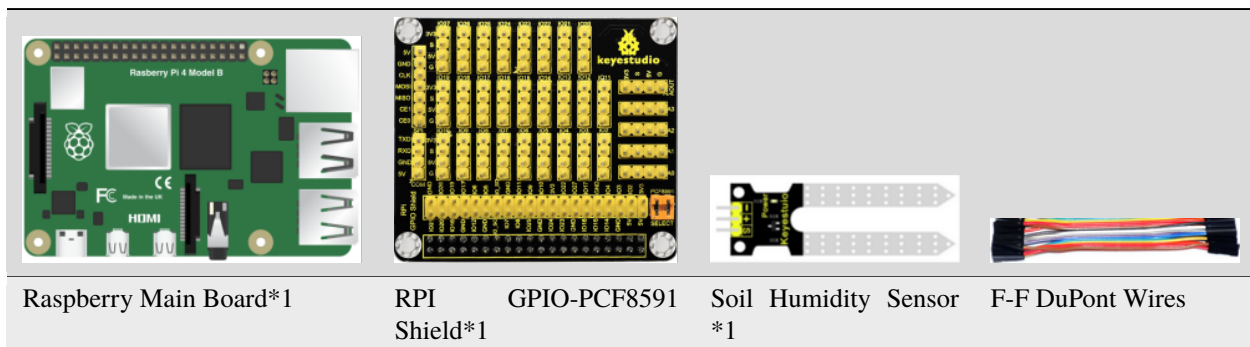
```

#### 4.4.29 Project 29 Flower-watering Device

##### (1)Description

The household plants are popular in many communities. But they will die if you forget to water them, how about making an automatic watering device? In this project, we will learn to detect the soil humidity of your plants with soil humidity sensor and Raspberry Pi.

##### (2)Components Needed



##### (3)Knowledge about Component

###### Soil Humidity Sensor

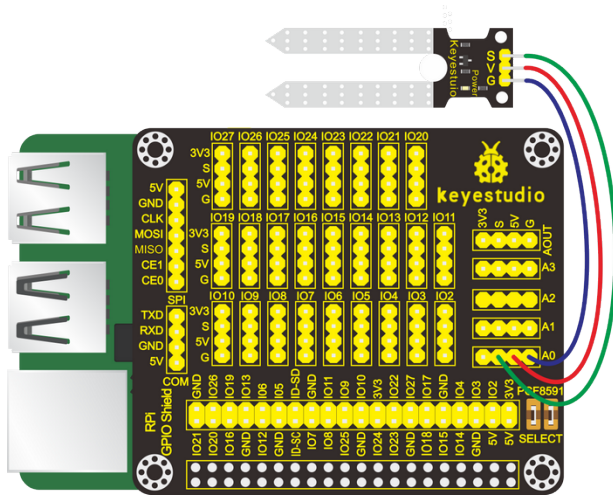
This is a simple soil humidity sensor aims to detect the soil humidity.

If the soil is in lack of water, the analog value output by the sensor will decrease; otherwise, it will increase. If you use this sensor to make an automatic watering device, it can detect whether your botany is thirsty to prevent it from withering when you go out.

Using the sensor with controller makes your plant more comfortable and your garden smarter. The soil humidity sensor module is not as complicated as you might think, and if you need to detect the soil in your project, it will be your best choice.

##### (4)Connection Diagram

Soil Humidity Sensor	RPI GPIO-PCF8591 Shield
S	SA0
V	5V
G	G



#### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press“Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson29_Soil
```

```
gcc Soil.c -o Soil -lwiringPi
```

```
sudo ./Soil
```

#### (6)Test Results

After running the program, when the soil humidity sensor is inserted into the land, the terminal prints the analog value of the soil humidity.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48 //address ---> device address
#define BASE 64 //DA converter command
#define A0 BASE+0 //A0 ----> port address
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

int main(void)
```

(continues on next page)

(continued from previous page)

```

{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);  //which port of the device you want to access

    while(1)
    {
        value=analogRead(A0);
        printf("A0:%d\n",value);
        delay(100);
    }
}

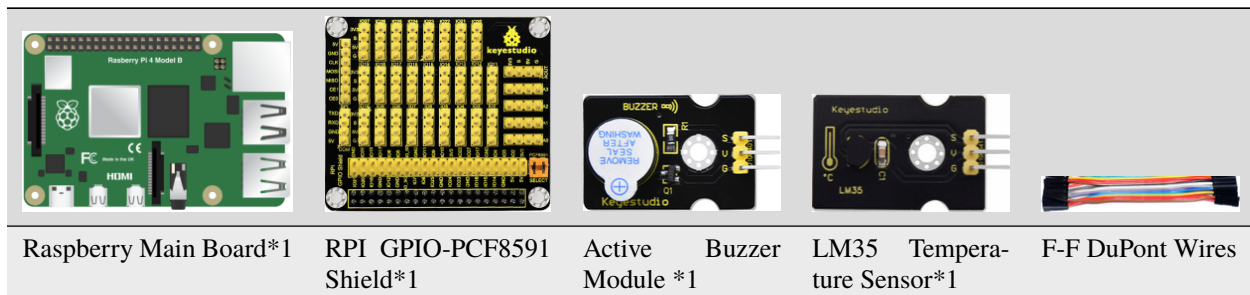
```

### 4.4.30 Project 30Temperature Alarm

#### (1)Description

In the frozen winter, farmers tend to heat the greenhouse to make the temperature suitable for vegetables to live so as to prevent them from freezing to death. And a temperature alarm device is required to avoid overheating. In the project, we will learn to make such a device.

#### (2)Components Needed



#### (3)Knowledge about Component

##### LM35 Temperature Sensor:

It is widely used temperature sensor whose output voltage proportional to temperature. It outputs 0° at the beginning since it adopts internal compensation. Its sensitivity is 10mV/°C and output temperature in the range of 0°C-100°C.

Transfer formula: output 0V when 0°, plus 1° each time, output voltage increases 10mV.

Working Voltage is 4-30V;

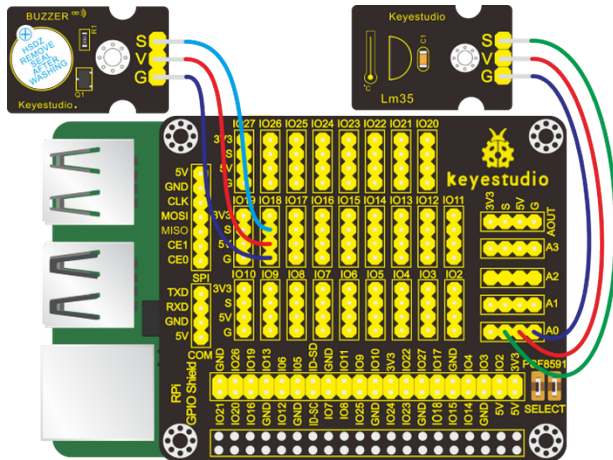
Accuracy:  $\pm 1^{\circ}\text{C}$ .

Maximum linear error:  $\pm 0.5^{\circ}\text{C}$ ;

Quiescent current: 80uA.

#### (4)Connection Diagram

Active Buzzer Mod- ule	RPI Shield	GPIO-PCF8591	LM35 Temperature Sen- sor	RPI Shield	GPIO-PCF8591
S	SIO18		S	SA0	
V	5V		V	5V	
G	G		G	G	



#### (5) Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson30_LM35
```

```
gcc LM35.c -o LM35 -lwiringPi
```

```
sudo ./LM35
```

#### (6) Test Results

When the programs, the terminal prints the value and temperature; and when the temperature detected is bigger than 20°C, the buzzer rings; otherwise, it makes no sounds.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

The temperature threshold set in the program is 20 and can be adjusted according to your needs.

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
```

(continues on next page)

(continued from previous page)

```

#define A2 BASE+2
#define A3 BASE+3

#define buzPin 1 //buzzer pin BCM GPIO 18

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);
    pinMode(buzPin,OUTPUT);

    while(1)
    {
        value=analogRead(A0); //Read the value of the LM35 temperture sensor
        printf("Temp:%d\n",value);
        delay(100);
        if(value>20)
        {
            digitalWrite(buzPin,HIGH);
        }
        else
        {
            digitalWrite(buzPin,LOW);
        }
    }
}

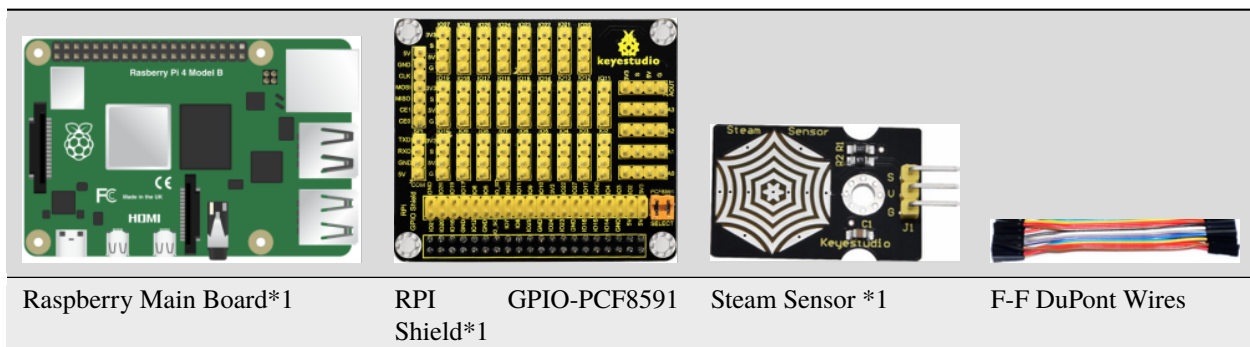
```

#### 4.4.31 Project 31 Steam in the Air

##### (1)Description

The world is infused with air and there are many elements in the air, some of which is useful and some of them is harmful to physical health. And in this project, we will learn to detect steam in the air with a steam sensor.

##### (2)Components Needed



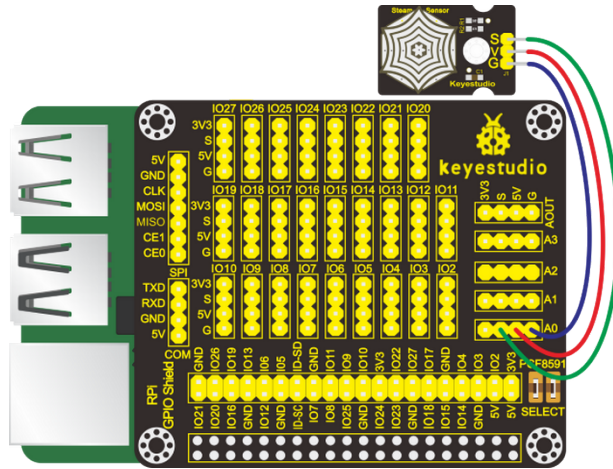
##### (3)Knowledge about Component

This is a commonly used steam sensor. Its principle is to detect the amount of water by bare printed parallel lines on the circuit board. The more the water is, the more wires will be connected. As the conductive contact area increases,

the output voltage will gradually rise. It can detect water vapor in the air as well. The steam sensor can be used as a rain water detector and level switch. When the humidity on the sensor surface surges, the output voltage will increase.

#### (4)Connection Diagram

Steam Sensor	RPI GPIO-PCF8591 Shield
S	SA0
V	5V
G	G



#### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson31_Water_Vapor
```

```
gcc Water_Vapor.c -o Water_Vapor -lwiringPi
```

```
sudo ./Water_Vapor
```

#### (6)Test Results

After running the program, the terminal displays the steam amount detected by the sensor.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3
```

(continues on next page)

(continued from previous page)

```

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);

    while(1)
    {
        value=analogRead(A0); //Read the value of the water_vapor sensor
        printf("water vapor value:%d\n",value); //print data
        delay(100);
    }
}

```

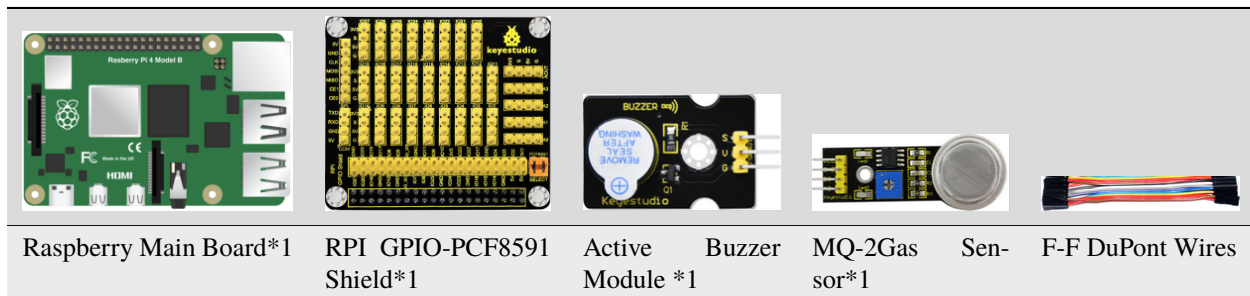
#### 4.4.32 Project 32MQ-2 Gas Leakage Alarm

##### (1)Description

Some families have access to gas, which is composed of CO, CO<sub>2</sub>, N<sub>2</sub>, H<sub>2</sub> and CH<sub>4</sub>. CO is one of toxic gases. People will be in danger if absorbing too much CO. However, we could tackle with this problem over a gas leakage alarm.

Gas MQ-2 leakage alarm detects the presence of a combustible or toxic gas and react by displaying a reading, setting off an audible or visual alarm.

##### (2)Components Needed



##### (3)Knowledge about Component

This gas sensor - MQ-2 adapts a gas-sensitive material called tin dioxide (SnO<sub>2</sub>) which is of low conductivity in clean air. Therefore, when combustible gases are detected in the air, it becomes more conductive. And the analog value increases with the increase of the concentration of flammable gases.

Meanwhile, it has high sensitivity to natural gas, liquefied petroleum gas and other smoke, especially to alkanes smoke.

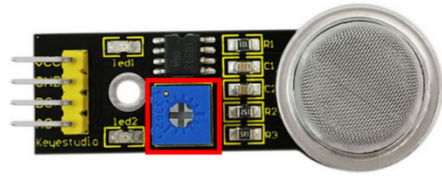
In final analysis, this gas sensor can find application in a wide range with low cost. For example, it can be applied to gas leak detection devices in homes and factories.

##### Note

1The sensitivity of the alcohol sensor can be adjusted by rotating the potentiometer on it.

Turning the knob clockwise, the threshold value increases while turning it counterclockwise, the threshold value decreases.

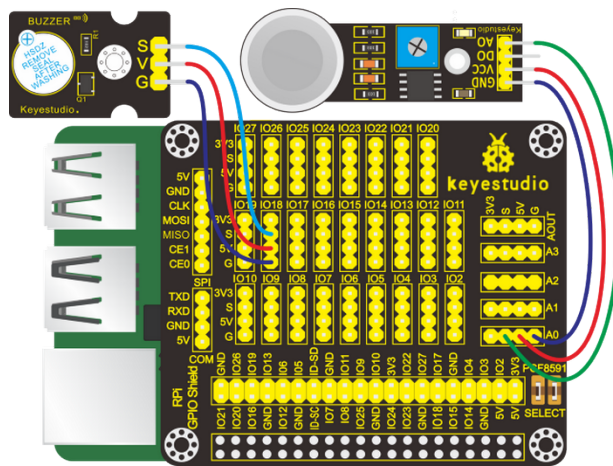




(2)The sensor may not be able to output stable and accurate data immediately, and it needs to be warmed up for about 1 minute to collect stable data.

(3)Connection Diagram

Active Buzzer Module	RPI GPIO-PCF8591 Shield	MQ-2Gas sor	Sen- sor	RPI GPIO-PCF8591 Shield
S	SIO18	S		SA0
V	5V	V		5V
G	G	G		G



(5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson32_Gas_MQ_2
gcc Gas_MQ_2.c -o Gas_MQ_2 -lwiringPi
sudo ./Gas_MQ_2
```

(6)Test Results

After running the program, the terminal shows the analog gas value detected by the MQ-2 gas sensor. And when the analog value of noxious gases is bigger than 60 the buzzer issues alarms.

Note: Press Ctrl + C on keyboard and exit code running.

(7)Example Code



```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>
#include <pcf8591.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

#define buzPin 1 //buzzer pin BCM GPIO 18

int main(void)
{
    unsigned char dat;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);
    if (wiringPiSetup() == -1){
        exit(1);
    }
    {
        pinMode(buzPin,OUTPUT);
    }
    while(1){

        dat=analogRead(A0);
        if(dat>80)
            digitalWrite(buzPin,HIGH);
        else
            digitalWrite(buzPin,LOW);
        printf("MQ-2:%d\n",dat);
        delay(100);
    }
    return 0;
}

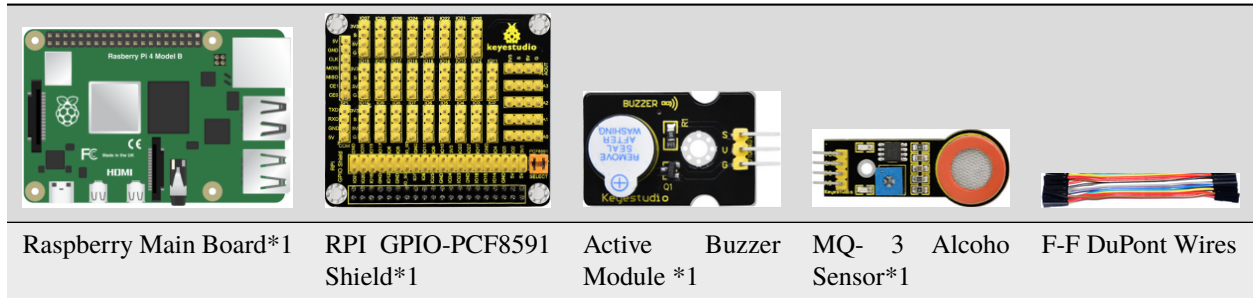
```

### 4.4.33 Project 33Alcohol Tester

#### (1)Description

The alcohol tester is an instrument that can be used to detect the content of alcohol left in bodies. It can assist traffic policemen to determine whether a driver drives after consuming alcohol or not or how much alcohol left in his/her body so as to prevent major traffic accidents. It can also be used in other occasions to detect the alcohol content in exhaled breath to avoid personal injuries, deaths and major property losses. For example, it can be applied to high-risk positions prohibiting work after drinking. In this lesson, we will simulate an alcohol tester.

#### (2)Components Needed

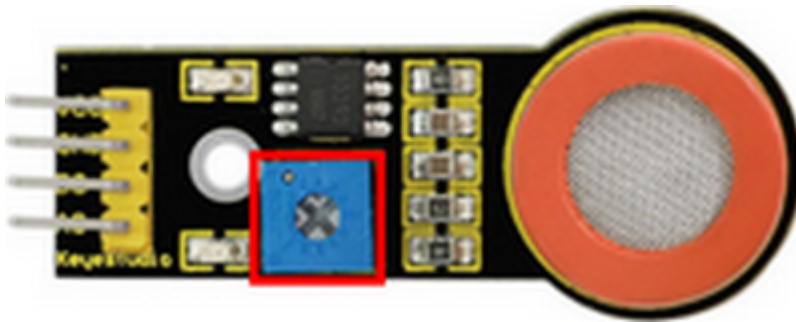


### (3) Knowledge of Component

#### MQ-3 Alcohol Sensor

This analog gas sensor - MQ3 adapts a gas-sensitive material called tin dioxide( $\text{SnO}_2$ ) which is of low conductivity in clean air. Therefore, when there is alcohol vapor detected, its conductivity increases with the increase of the alcohol vapor concentration and it outputs signals (digital and analog signals). The higher the alcohol concentration it senses, the greater the analog value the terminal outputs.

**Note:** the sensitivity of the alcohol sensor can be adjusted by rotating the potentiometer on it.



**Please note that the sensor may not be able to output stable and accurate data immediately, and it needs to be warmed up for about 1 minute to collect stable data.**

### (4) Connection Diagram

Active Buzzer Module	RPI Shield	GPIO-PCF8591	MQ-3 Alcohol Sensor	RPI Shield	GPIO-PCF8591
S	SIO18		S	SA0	
V	5V		V	5V	
G	G		G	G	



(continued from previous page)

```

pcf8591Setup(BASE,Address);
if (wiringPiSetup() == -1){
    exit(1);
}
{
    pinMode(buzPin,OUTPUT);
}
while(1){

    dat=analogRead(A0);
    if(dat>80)
        digitalWrite(buzPin,HIGH);
    else
        digitalWrite(buzPin,LOW);
    printf("MQ-3:%d\n",dat);
    delay(100);
}
return 0;
}

```

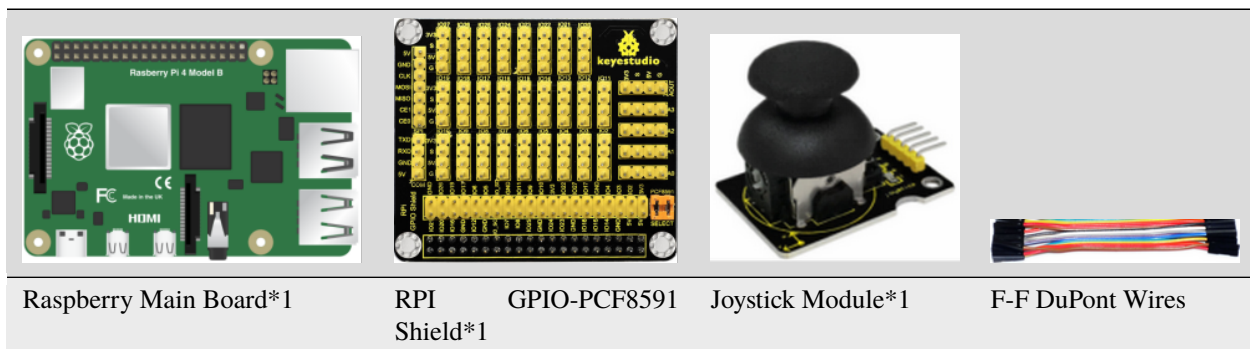
#### 4.4.34 Project 34 Joystick Module

##### (1)Description

Many a people play games with gamepad. But do you know who it work?

Let's learn about it.

##### (2)Components Needed



##### (3)Knowledge about Component

##### Joystick Module

This is a joystick very similar to the 'analog' joysticks on PS2 (PlayStation 2) controllers. It is a self-centering spring loaded joystick, meaning when you release the joystick it will center itself. It also contains a comfortable cup-type knob/cap which gives the feel of a thumb-stick.

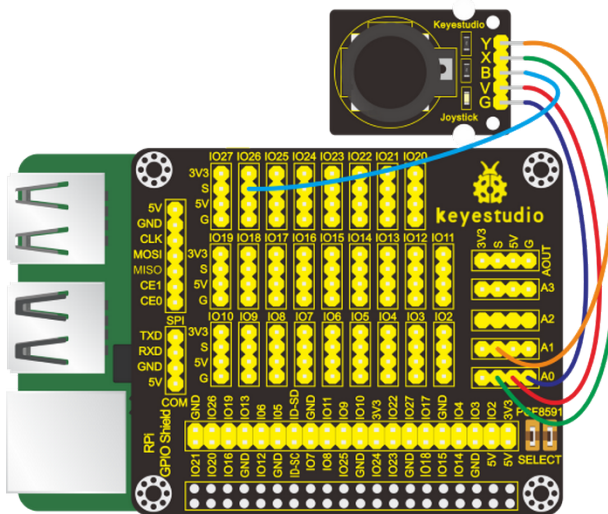
It has three signal pins which are connected GND, VCC and signal endB, X, Y). The X pin is **X-axis** (left to right), the Y pin is **Y-axis** (front and back) and signal B end is Z-axis(usually used as digital port and pushbutton).

VCC is connected to V/VCC3.3/5V of MCU, GND to G/GND of MCU and the voltage is around 1.65V/2.5V in initial status.

X axis gives readout of the joystick in the horizontal direction (X-coordinate) i.e. how far left and right the joystick is pushed.

#### (4)Connection Diagram

Joystick Module	RPI GPIO-PCF8591 Shield
Y	SA1
X	SA0
B	S(IO26)
V	5V
G	G



#### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson34_Joystick
```

```
gcc Joystick.c -o Joystick -lwiringPi
```

```
sudo ./Joystick
```

#### (6)Test Results

Rotate Joystick , terminal will show the responding data change and press it, “The key is pressed” is displayed in the terminal.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
```

(continues on next page)

(continued from previous page)

```

#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

#define btnPin 25 //GPIO 26

int main(void)
{
    unsigned char x_val;
    unsigned char y_val;
    unsigned char z_val;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);
    pinMode(25,INPUT);

    while(1)
    {
        x_val=analogRead(A0); //read x
        y_val=analogRead(A1); //read y
        z_val=digitalRead(25); //read z, button
        printf(" x:%d y:%d z:%d\n", x_val,y_val,z_val);
        if(z_val==1)
            printf("The key is presed!\n");
        delay(100);
    }
}

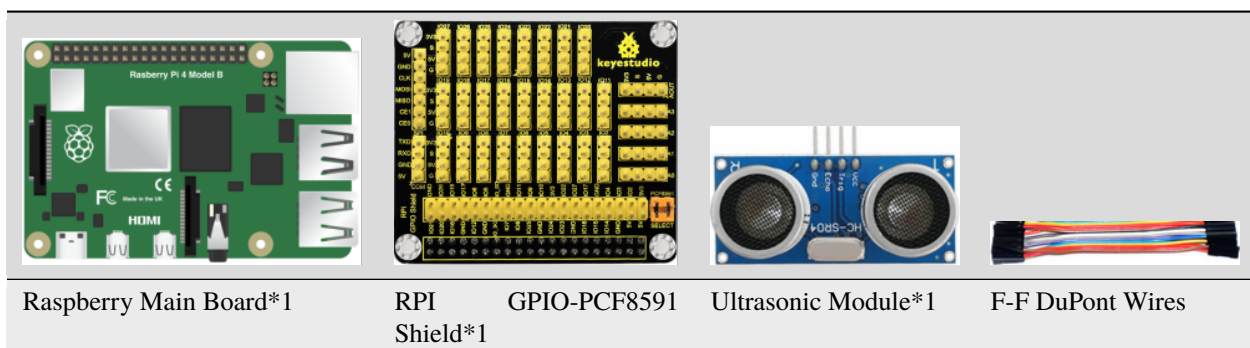
```

#### 4.4.35 Project 35Ultrasonic Sensor

##### (1)Description

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal.

##### (2)Components Needed



##### (3)Knowledge about Component

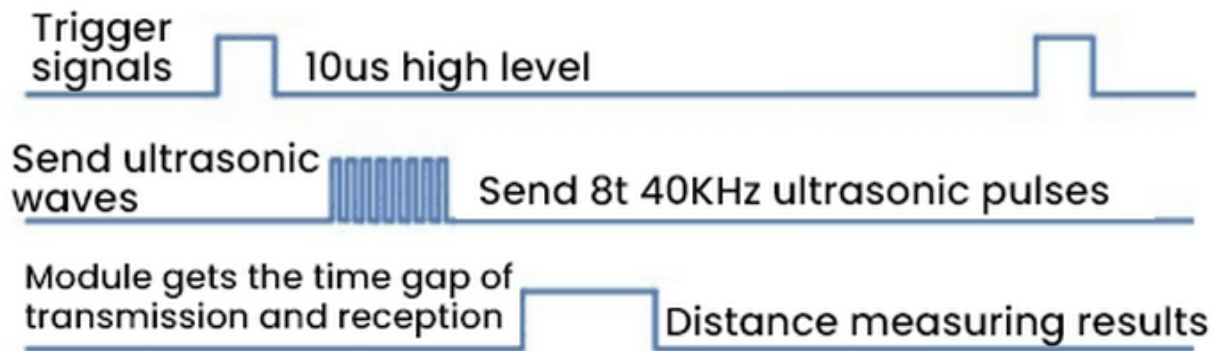
The ultrasonic module will emit the ultrasonic waves after trigger signal. When the ultrasonic waves encounter the object and are reflected back, the module outputs an echo signal, so it can determine the distance of object from the time difference between trigger signal and echo signal.

The  $t$  is the time that emitting signal meets obstacle and returns.

and the propagation speed of sound in the air is about 343m/s, therefore, distance = speed \* time, because the ultrasonic wave emits and comes back, which is 2 times of distance, so it needs to be divided by 2, the distance measured by ultrasonic wave = (speed \* time)/2

Use method and timing chart of ultrasonic module:

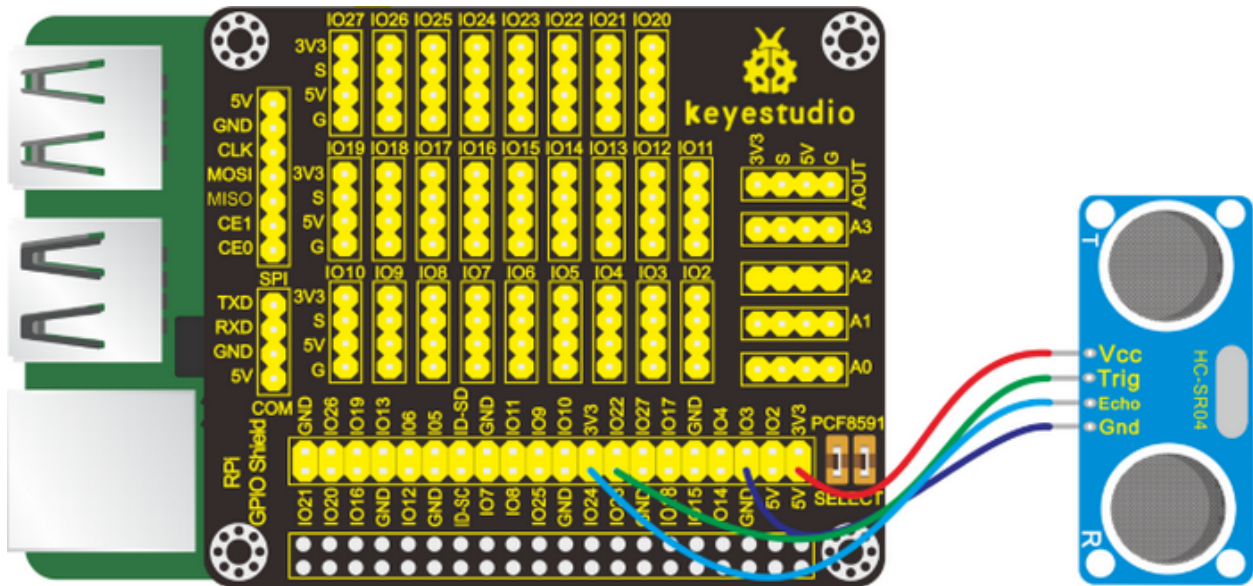
1. Setting the delay time of Trig pin of SR04 to 10s at least, which can trigger it to detect distance.
2. After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step will be completed automatically by the module.
3. If the signal returns, the Echo pin will output a high level, and the duration of the high level is the time from the transmission of the ultrasonic wave to the return.



(4)Connection Diagram

Ultrasonic Module	RPI GPIO-PCF8591 Shield
Vcc	5V
Trig	S(IO23)
Echo	S(IO24)
Gnd	GND





#### (5) Run Example Code

Input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson35_Ultrasonic
```

```
gcc Ultrasonic.c -o Ultrasonic -lwiringPi
```

```
sudo ./Ultrasonic
```

#### (6) Test Results

Terminal prints the detected distance, unit is cm.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#include <sys/time.h> //Import the time system header file

//define the pin
#define Trig 4 //BCM GPIO 23
#define Echo 5 //BCM GPIO 24

//set pin mode
void ultraInit(void)
{
    pinMode(Echo, INPUT);
    pinMode(Trig, OUTPUT);
}

//Write programs based on sequence diagrams
float disMeasure(void)
{
    struct timeval tv1; //Create the Timeval structure tv1
    struct timeval tv2; //Create the Timeval structure tv2
```

(continues on next page)



(continued from previous page)

```

    long start, stop;
    float dis;

    digitalWrite(Trig, LOW);
    delayMicroseconds(2);

    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);

    while(!(digitalRead(Echo) == 1)); //Wait for the low level received by the Echo
    ↪pin to pass
    gettimeofday(&tv1, NULL); //function gettimeofday, The time it took the system
    ↪to get here

    while(!(digitalRead(Echo) == 0)); //Wait for the high level received by the
    ↪Echo pin to pass
    gettimeofday(&tv2, NULL); //function gettimeofday, The time it took the system
    ↪to get here

    //Tv1.tv_sec is the seconds obtained, tv1.tv_usec is the subtlety obtained
    //Calculate the first time
    start = tv1.tv_sec * 1000000 + tv1.tv_usec;

    //Calculate the second time
    stop = tv2.tv_sec * 1000000 + tv2.tv_usec;

    //stop - start , the time difference is the high level time acquired by the echo pin
    //34000cm/s, speed of sound
    //Calculate the distance measured(cm)
    dis = (float)(stop - start) / 1000000 * 34000 / 2;

    return dis;
}

int main(void)
{
    float dis;

    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
        printf("setup wiringPi failed !");
        return 1;
    }

    ultraInit();

    while(1){
        dis = disMeasure();
        printf("distance = %0.2f cm\n",dis);
        delay(100);
    }
}

```

(continues on next page)

(continued from previous page)

```

    return 0;
}

```

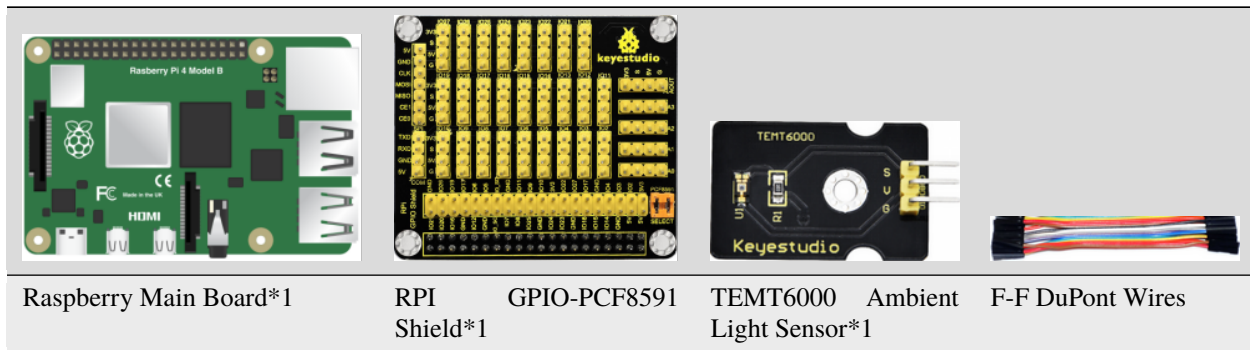
link[https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/sys\\_time.h.html](https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/sys_time.h.html)

## 4.4.36 Project 36 Light Intensity Detection

### (1)Description

This project is a little bit similar to the one involved the photoresistor sensor. But this time we will use a TEMT6000 ambient light sensor which has better sensitivity. Now, let's learn how to use this sensor to detect light intensity with Raspberry Pi.

### (2)Components Needed



### (3)Knowledge about Component

#### TEMT6000 Ambient Light Sensor

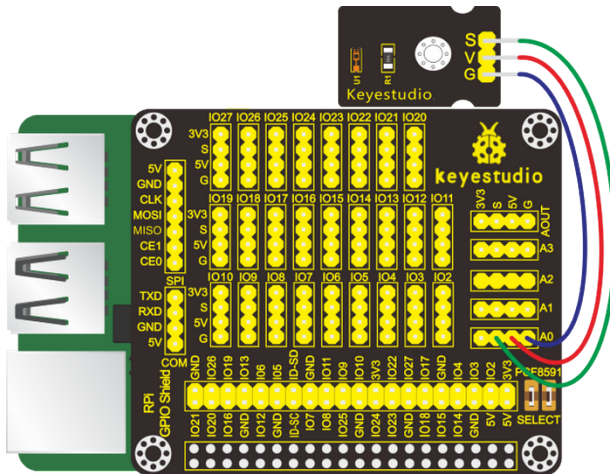
This module is mainly composed of a highly sensitive visible photocell (NPN type) triode, which can magnify the captured tiny light illumination changes by about 100 times, and is easily recognized by the microcontroller for AD conversion.

And the light intensity is directly proportional to current flowing through. Therefore, it is easy to figure out the light intensity as long as its voltage is known.

Its response to visible light illumination is similar to that of the human eye, so that can detect the intensity of ambient light.

### (4)Connection Diagram

TEMT6000 Ambient Light Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



#### (5) Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press “Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson36_TEMT6000_Ambient_Light
gcc TEMT6000_Ambient_Light.c -o TEMT6000_Ambient_Light -lwiringPi
sudo ./TEMT6000_Ambient_Light
```

#### (6) Test Results

After running the program, the terminal displays the light intensity value detected by the sensor; and the stronger the light, the bigger the analog value.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7) Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);

    while(1)
```

(continues on next page)

(continued from previous page)



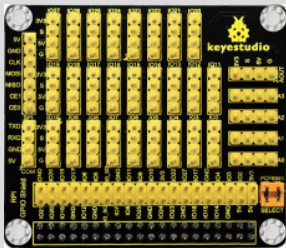
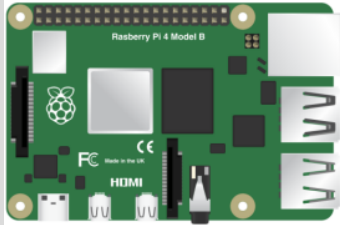
```
{
  value=analogRead(A0); //Read the value of the TEMT6000 Ambient Light sensor
  printf("Ambient Light:%d\n",value); //print data
  delay(100);
}
```

4.4.37 Project 37Pressure Measurement

(1)Description

In previous projects, we have learned how to use different sensors to obtain external information about temperature, light, sound ,gas and others. Now, let’s move to detect pressure with a thin-film pressure sensor and Raspberry Pi.

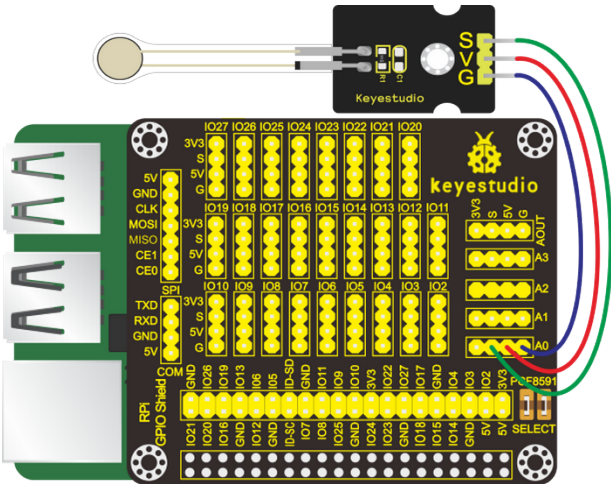
(2)Components Needed:



Raspberry Main Board\*1RPI GPIO-PCF8591 Shield\*1Thin-film Pressure Sensor\*1PressureF-F DuPont Wires

(3)Connection Diagram

Thin-film Pressure Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



## (4) Knowledge about Component

**Thin-film Pressure Sensor**

This sensor adopts the flexible Nano pressure-sensitive material with an ultra-thin film pad. It has the functions of water-proof and pressure detection.

When the sensor detects the external pressure, the resistance of sensor will make a change. So, we can design a circuit to convert the pressure signal that senses pressure changes into the corresponding electric signal outputs.

In this way, we can know the conditions of pressure changes by detecting the signal changes.

## (5) Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press "Enter". If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/C_code/lesson37_Pressure_Transducer
gcc Pressure_Transducer.c -o Pressure_Transducer -lwiringPi
sudo ./Pressure_Transducer
```

## (6) Test Results

After running the program, the terminal prints the value of the external pressure detected by the thin-film pressure sensor and the value increases with the increase of the pressure detected and reduces with the decrease of the pressure.

Note: Press Ctrl + C on keyboard and exit code running.

## (7) Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address);

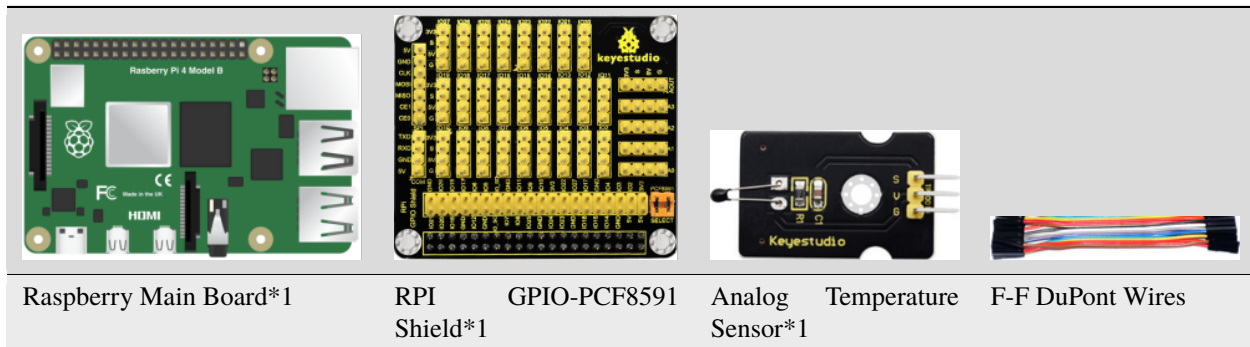
    while(1)
    {
        value=analogRead(A0); //Read the value of the pressure sensor
        printf("pressure value:%d\n",value); //print data
        delay(100);
    }
}
```

## 4.4.38 Project 38 Temperature Detection

### (1)Description

Thermistor is a kind of resistor whose resistance varies with temperature. We can use this characteristics to make thermometers.

### (2)Components Needed

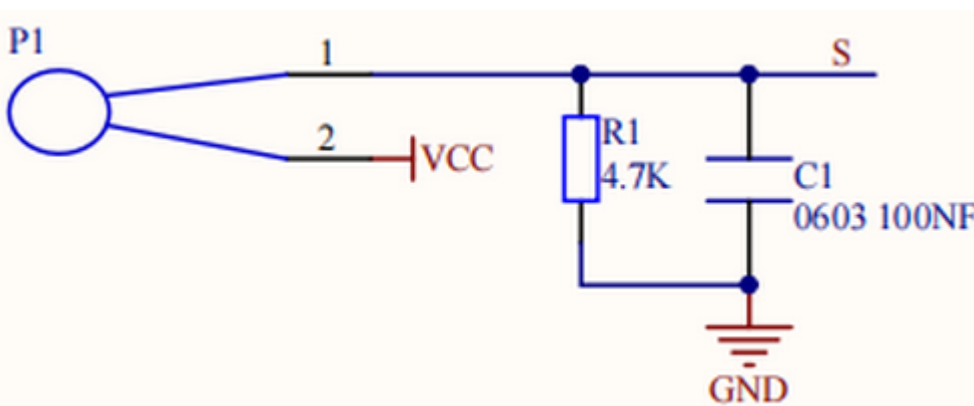


### (3)Knowledge about Component

#### Analog Temperature Sensor

The main part of this sensor is a thermistor which is quite sensitive to temperature. When it senses the changes of temperature, it makes changes in its resistance. This function of it can be used to detect temperature. Therefore, it has found applications in gardening, home alarm systems and other devices.

The NTC-MF52AT thermistor of 10K (P1) S and resistor R1 of 4.7K are connected in series. The resistance value of the thermistor alters with temperature changes.



Calculation of NTC thermistor:

The calculation formula of the forNTC thermistor is:

$$R_t = R \cdot \text{EXP}[B \cdot (1/T_1 - 1/T_2)]$$

Among them, T1 and T2 refer to degrees, which is the temperature in Kelvin;

Rt is the resistance of the thermistor at temperature T1;

R is the nominal resistance of the thermistor at normal temperature T2, and the resistance of the 10K thermistor at 25°C is 10K (that is, R=10K); T2 = (273.15+ 25);

EXP[n] represents en ( e to the nth power );

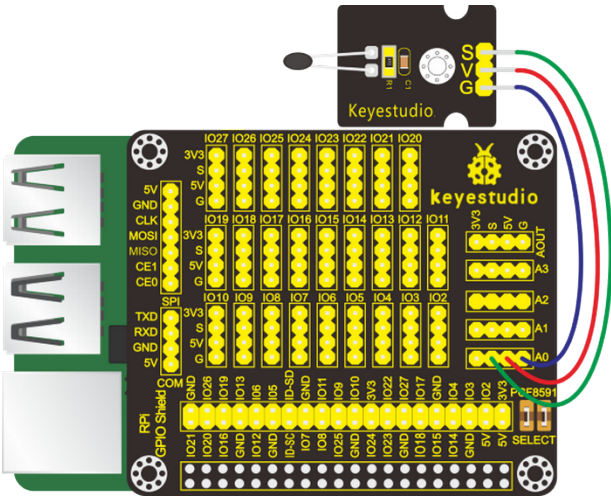
The value of B is an important parameter of thermistor and B=3950.

We can use the value measured by the ADC converter to get the resistance value of the thermistor, and then use the formula to get the temperature value. Therefore, the temperature formula can be derived as  $T1=1/(\ln(Rt/R)/B+1/T2)$ , where ln can be converted to log, that is,  $T1=1/(\log(Rt/R)/B+1/T2)$ .

The corresponding Celsius temperature is  $t=T1-273.15$ , and the deviation is  $\pm 0.5$ .

(4)Connection Diagram

Analog Temperature Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



(5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter commandi2cdetect -y 1 and press“Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communicationinput the following commands and press “Enter”:

```
cd /home/pi/C_code/lesson38_Analog_Temperature
gcc Analog_Temperature.c -o Analog_Temperature -lwiringPi -lm
sudo ./Analog_Temperature
```

(6)Test Results

After running the program, the terminal prints the ADC value of the analog temperature sensor, voltage and temperature.

Note: Press Ctrl + C on keyboard and exit code running.

(7)Example Code

```
#include <wiringPi.h>
#include <stdio.h>
#include <pcf8591.h>
#include <math.h>
```

(continues on next page)



(continued from previous page)

```

#define Address 0x48
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

int main(void){
    wiringPiSetup();
    pcf8591Setup(BASE,Address);

    while(1){
        int value = analogRead(A0); //read analog value A0 pin
        float voltage = (float)value / 255.0 * 5.0; // calculate voltage
        float Rt = 4.7 * (5.0 / voltage) - 4.7 ; //calculate resistance value of
        ↪ thermistor, 5.0 * (R / (Rt + R)) = voltage,>>>Rt = R * (5.0 / voltage) - R
        float tempK = 1/(1/(273.15 + 25) + log(Rt/4.7)/3950.0); //calculate temperature
        ↪ (Kelvin)
        float tempC = tempK - 273.15; //calculate temperature (Celsius)
        printf("ADC value : %d ,\tVoltage : %.2fV, \tTemperature : %.2fC\n",value,
        ↪ voltage,tempC);
        delay(100);
    }
    return(0);
}

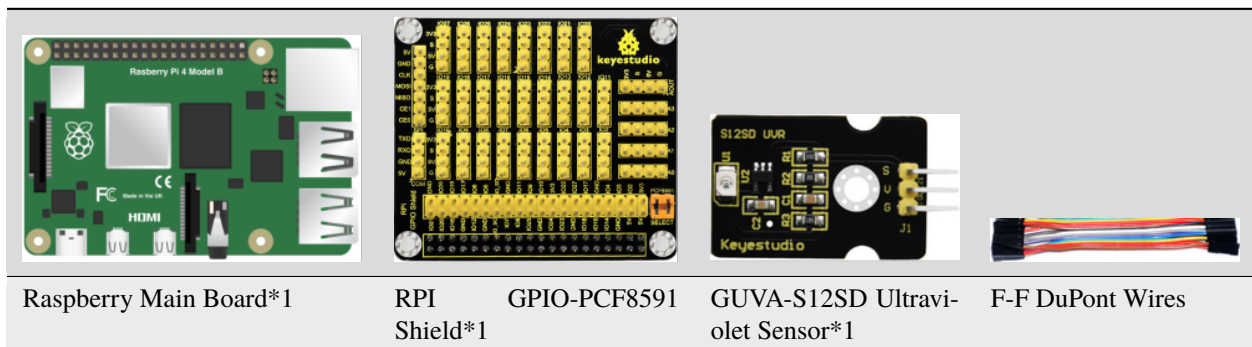
```

#### 4.4.39 Project 39: Ultraviolet Light Detection

##### (1)Description

Ultraviolet light is a kind of physical optics. The main source of ultraviolet light in nature is the sun. Most of the ultraviolet rays emitted by the sun are absorbed by the ozone in the atmosphere, and a very small part will be irradiated on the earth. We can use an ultraviolet sensor to detect the amount of ultraviolet rays in the sun.

##### (2)Components Needed



##### (3)Knowledge about Component:

##### GUVVA-S12SD Ultraviolet Sensor

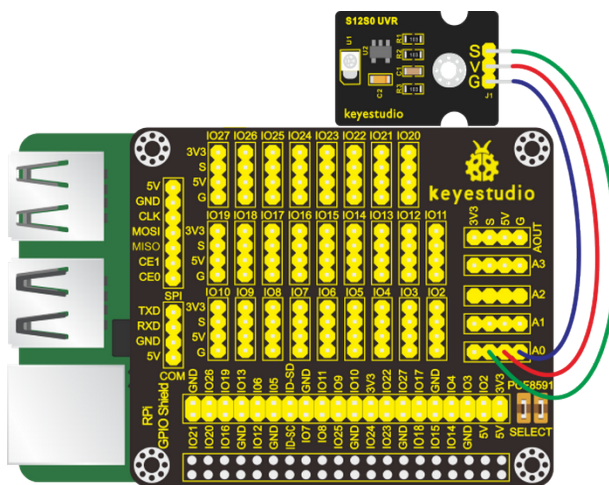


It is used for ultraviolet light detection. For example, it can be applied to detect the UV index of some smart wearable devices including watches, smart phones and others and of some outdoor equipment. It also can find applications in detecting the intensity of ultraviolet light and serve as an ultraviolet flame detector when disinfecting things.

The output current of the GUVA-S12SD ultraviolet sensor is proportional to the light intensity, and the product output has a very high consistency. The sensor has a specific spectral response. It mainly for the measurement of ultraviolet rays in the sun and the intensity of UVA lamps, and is particularly suitable for UVI detection.

#### (4)Connection Diagram

GUVA-S12SD Ultraviolet Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



#### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press "Enter". If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/C_code/lesson39_Ultraviolet_Ray
gcc Ultraviolet_Ray.c -o Ultraviolet_Ray -lwiringPi
sudo ./Ultraviolet_Ray
```

#### (5)Test Results

After running the program, pointing an ultraviolet pen (we don't provide it) emitting ultraviolet rays at a sensor or putting it under the sun, the terminal prints the ultraviolet intensity value.

Note: Press Ctrl + C on keyboard and exit code running.

#### (7)Example Code

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
```

(continues on next page)

(continued from previous page)

```
#define Address 0x48 //address ---> device address
#define BASE 64 //DA converter command
#define A0 BASE+0 //A0 ----> port address
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3

int main(void)
{
    unsigned char value;
    wiringPiSetup();
    pcf8591Setup(BASE,Address); //which port of the device you want to access

    while(1)
    {
        value=analogRead(A0);
        printf("ultraviolet intensity:%d\n",value);
        delay(100);
    }
}
```

## PROCESSING JAVA TUTORIAL

We recommend you to learn the Python and C language tutorials about this kit firstly. And we have demonstrated how to install Raspberry Pi OS, fix IP address and conduct remote login.

### 5.1 1.Preparations

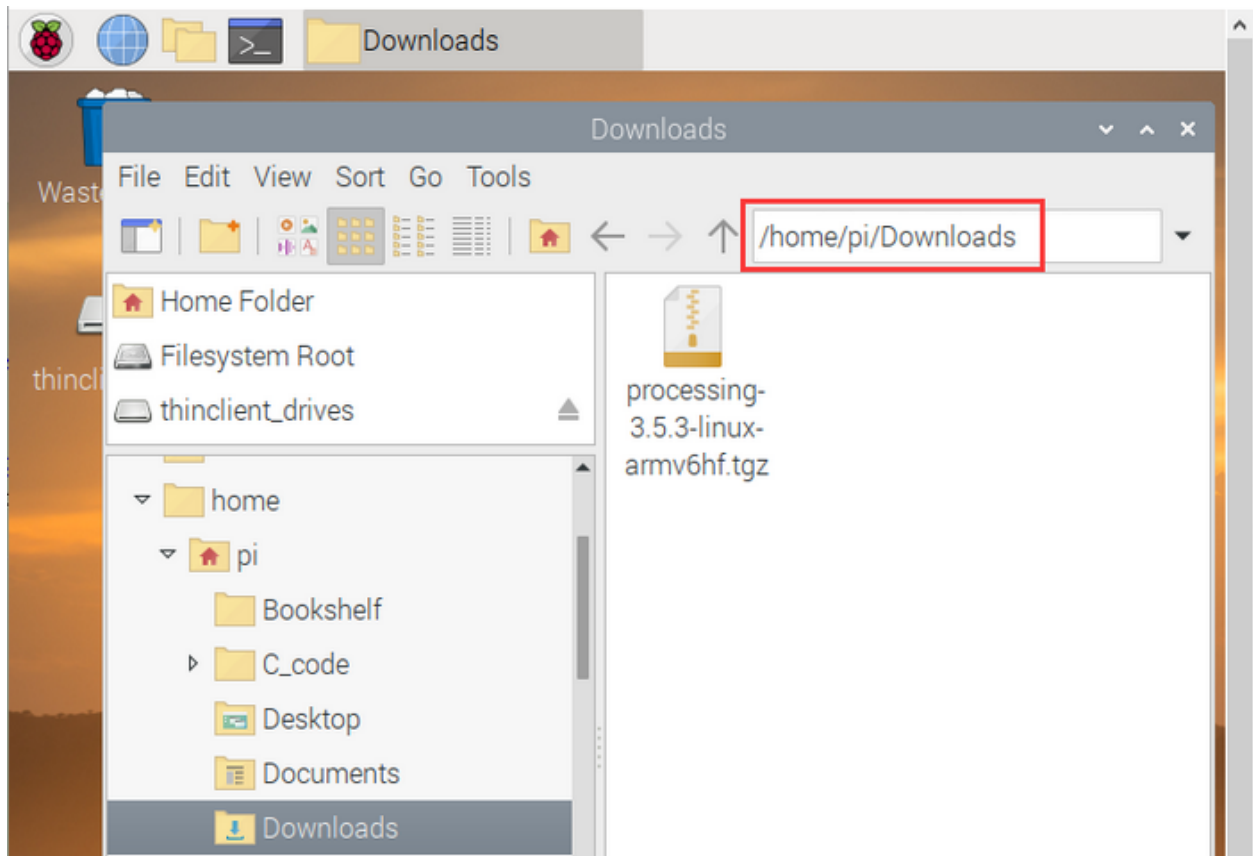
#### 5.1.1 (1)Install processing IDE

1.Processing Website<https://pi.processing.org/get-started/>

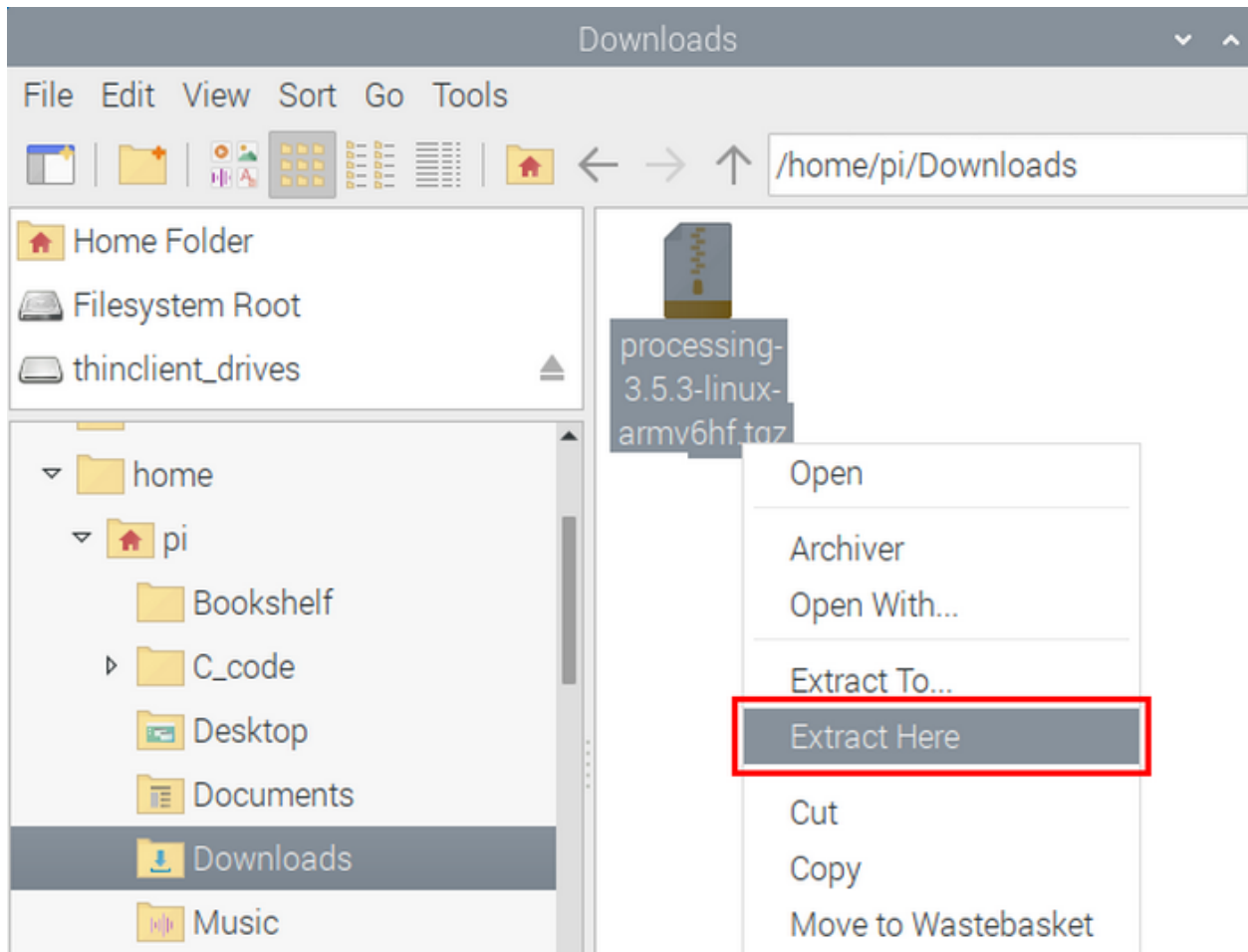
2. Download processing IDE installation package:

<https://github.com/processing/processing/releases/download/processing-0269-3.5.3/processing-3.5.3-linux-armv6hf.tgz>

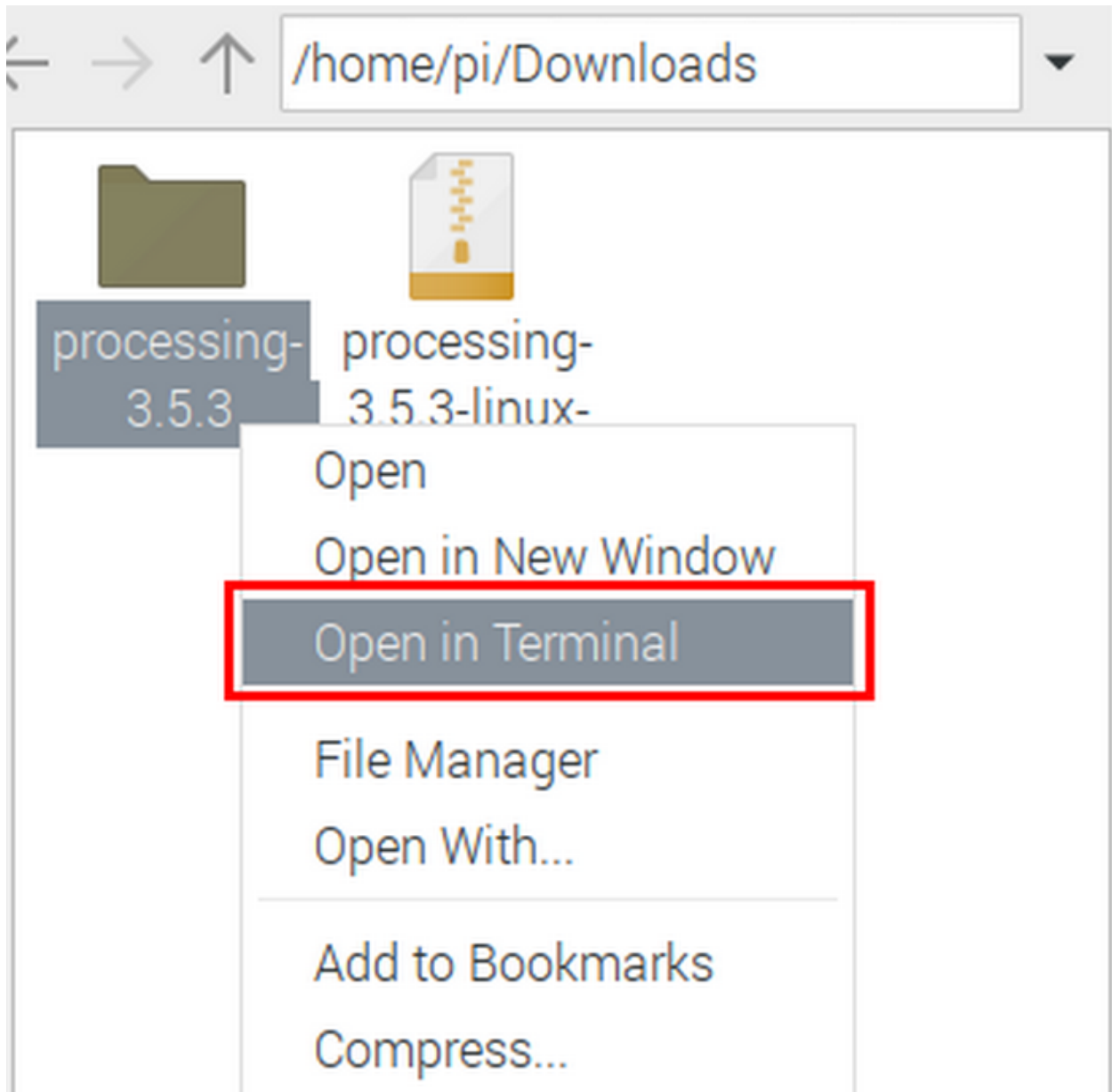
You could download zip file to the download folder of Raspberry Pi. Equally, you could save it into Downloads folder in the windows system, as shown below:



Unzip installation package and click it and select Extract Here



Then right-click to unzip folder and choose Open in Terminal;

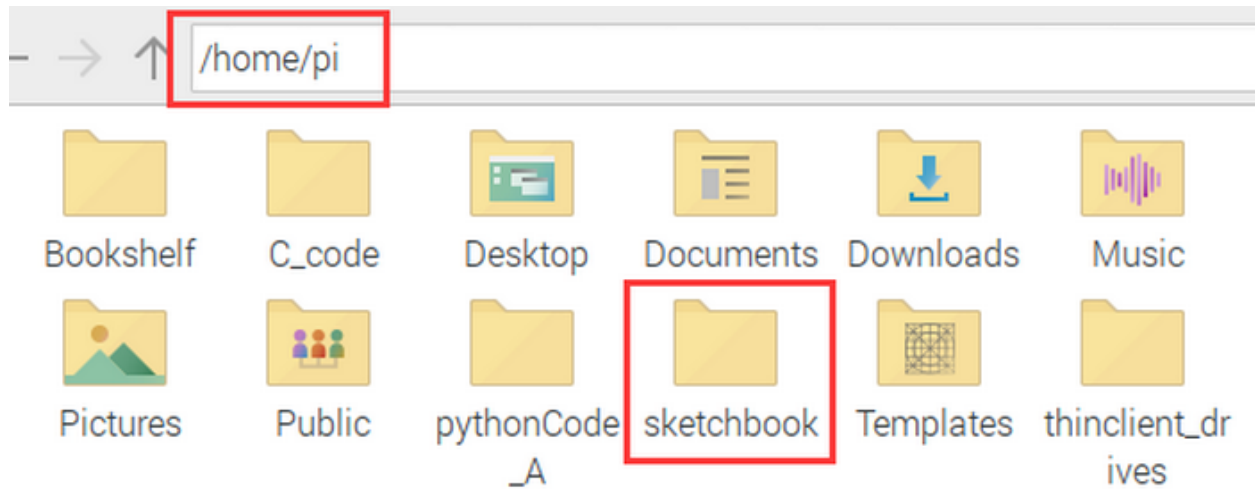


Input installation command: `sudo ./install.sh` and press “Enter”;

A screenshot of a terminal window titled `pi@raspberrypi: ~/Downloads/processing-3.5.3`. The terminal shows the command `sudo ./install.sh` being executed, which is highlighted with a red rectangular box. The output of the command is as follows:

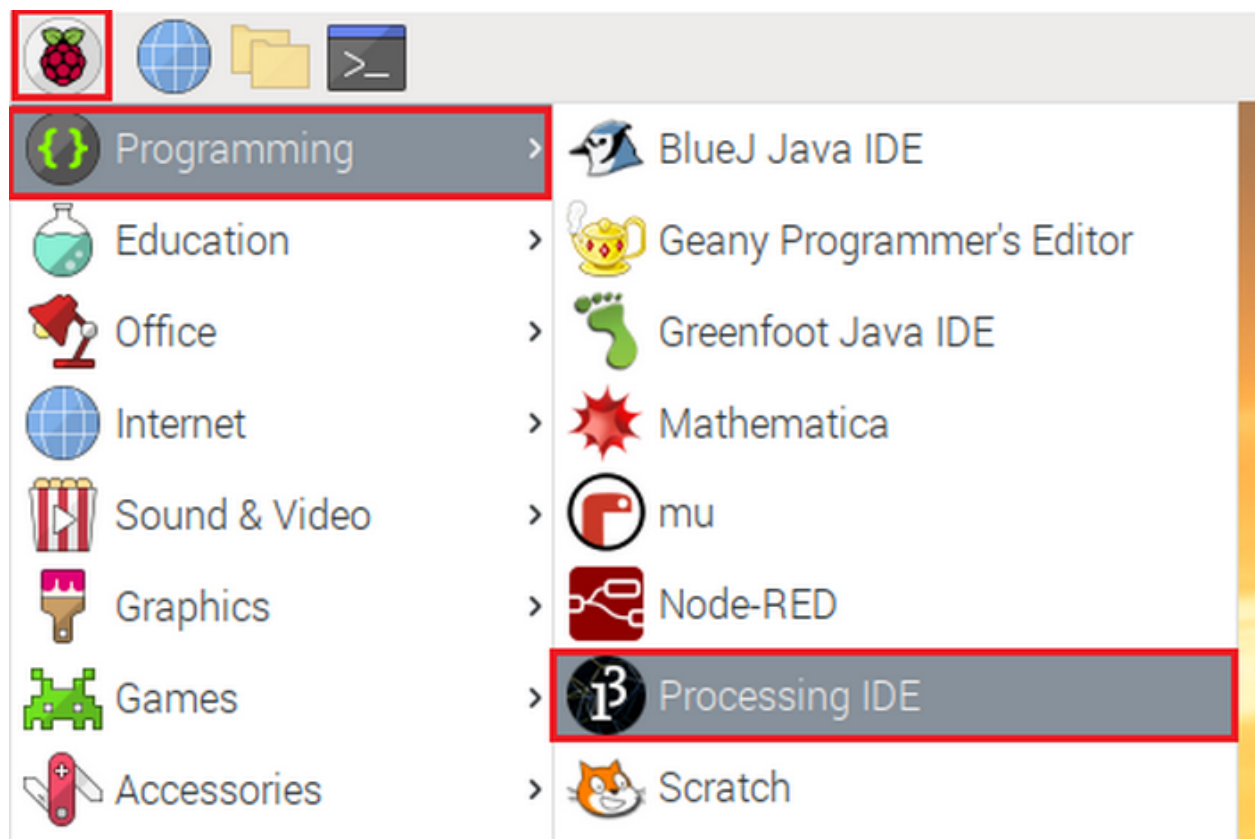
```
pi@raspberrypi:~/Downloads/processing-3.5.3 $ sudo ./install.sh
Adding desktop shortcut, menu item and file associations for Processing...touch:
cannot touch '/root/.config/mimeapps.list': No such file or directory
/usr/bin/xdg-mime: 848: /usr/bin/xdg-mime: cannot create /root/.config/mimeapps.
list.new: Directory nonexistent
done!
pi@raspberrypi:~/Downloads/processing-3.5.3 $
```

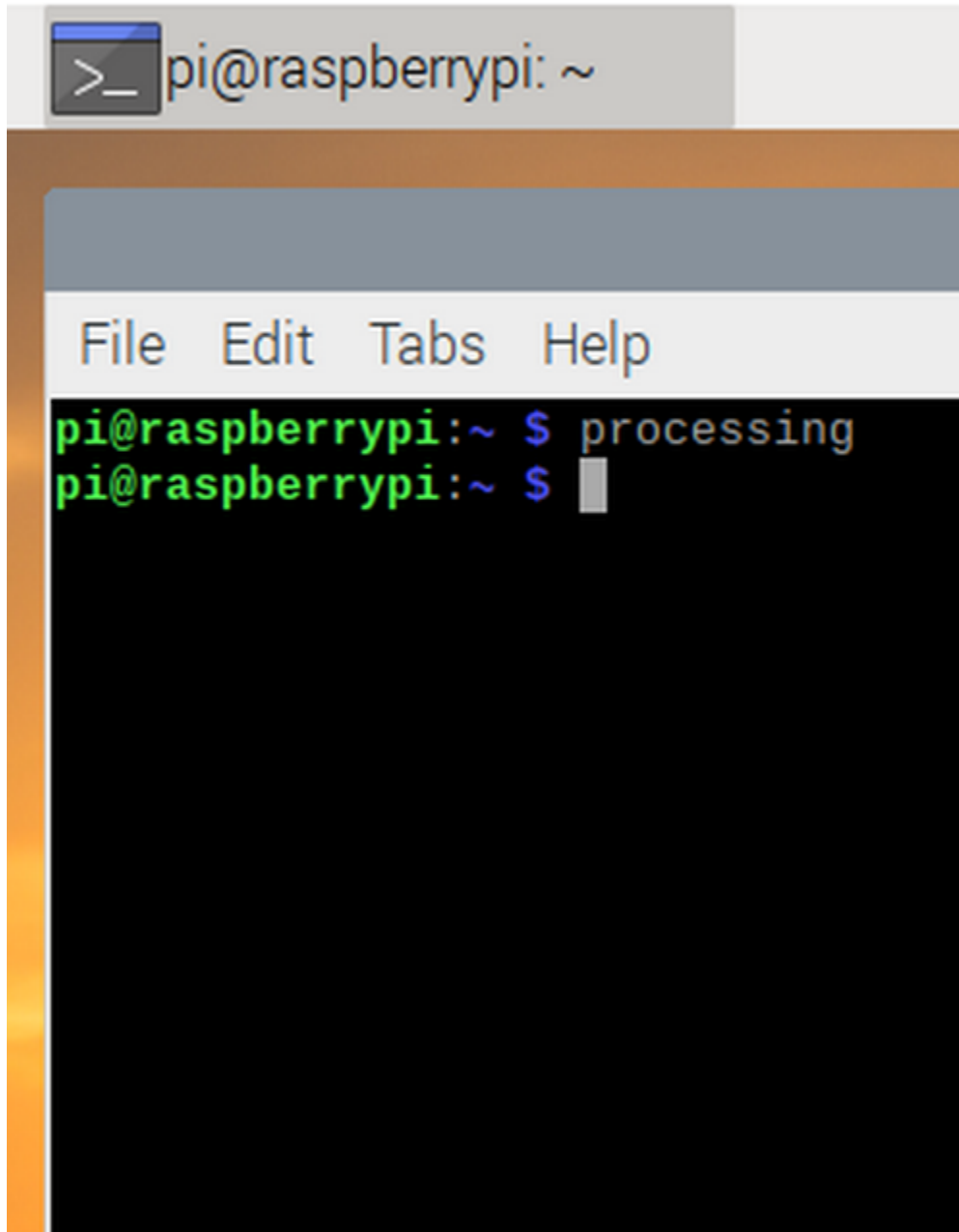
After the download, a sketchbook folder generates in the pi folder, which is default route of saving code;



Then click Programming→Processing IDE

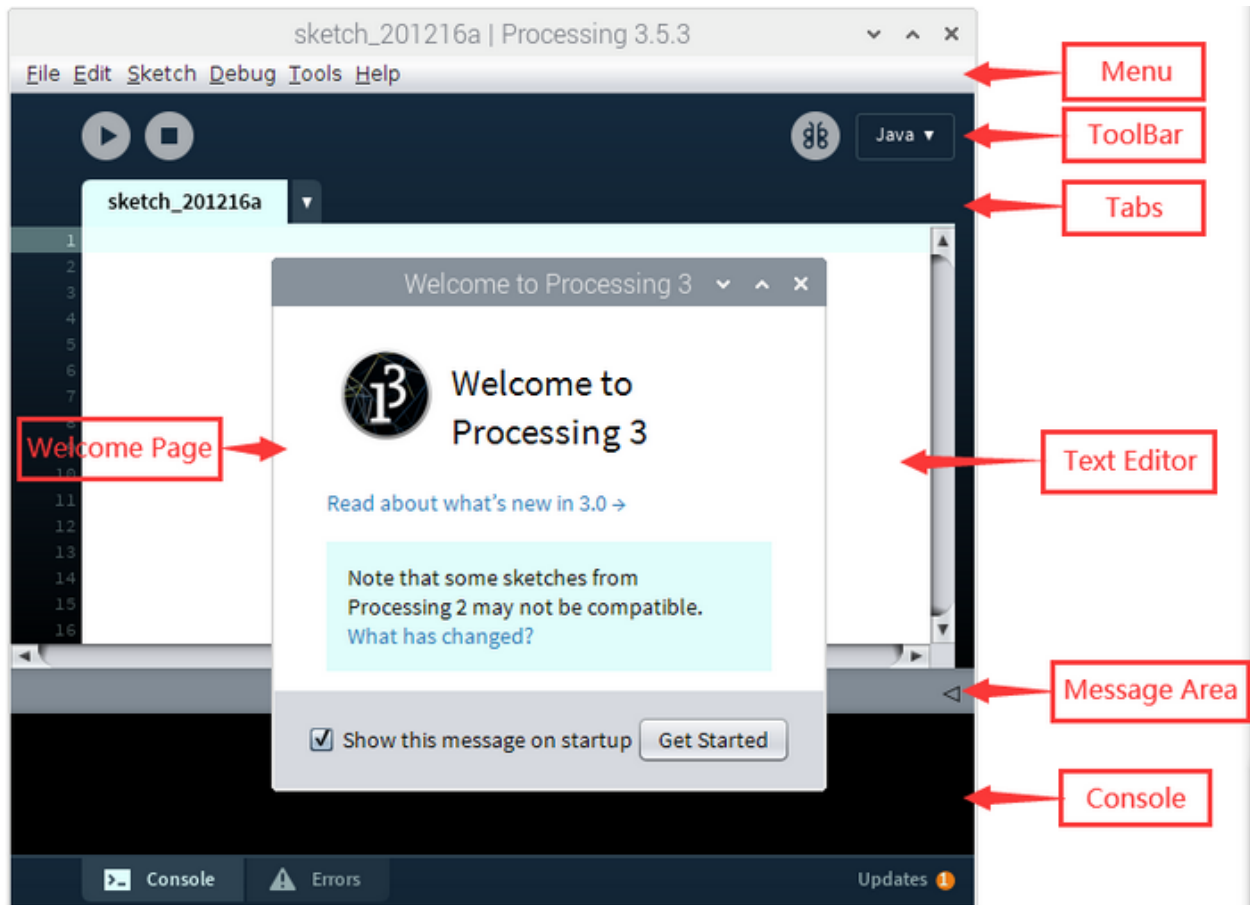
Input processing in terminal to open processing IDE, as shown below:





Its interface is shown below:

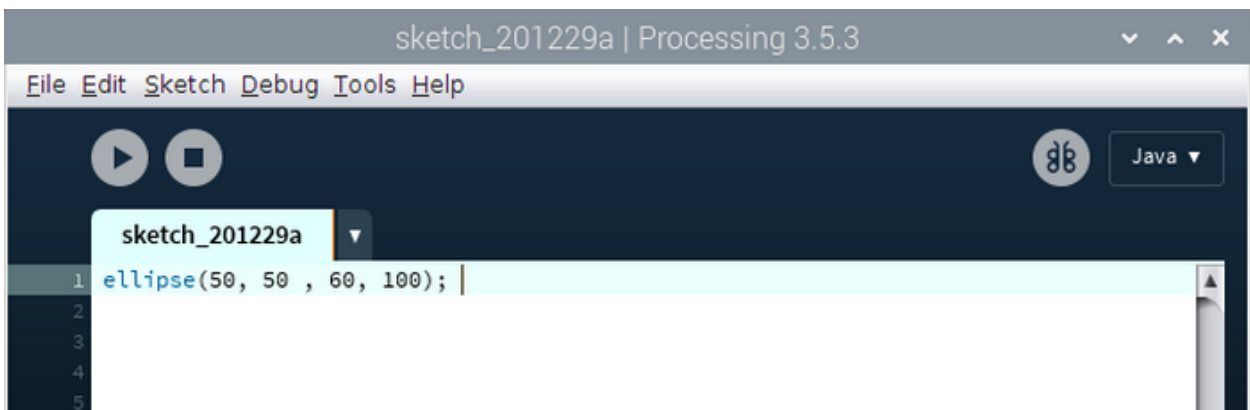




### 5.1.2 (2)Use Processing IDE

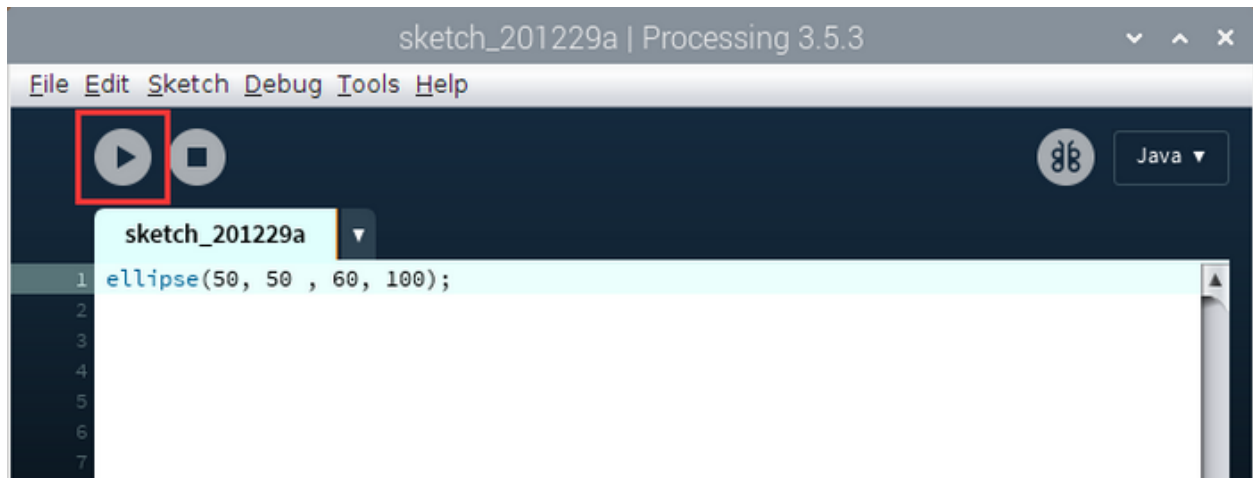
Enter the code in the editor

```
ellipse(50, 50 , 60, 100);
```

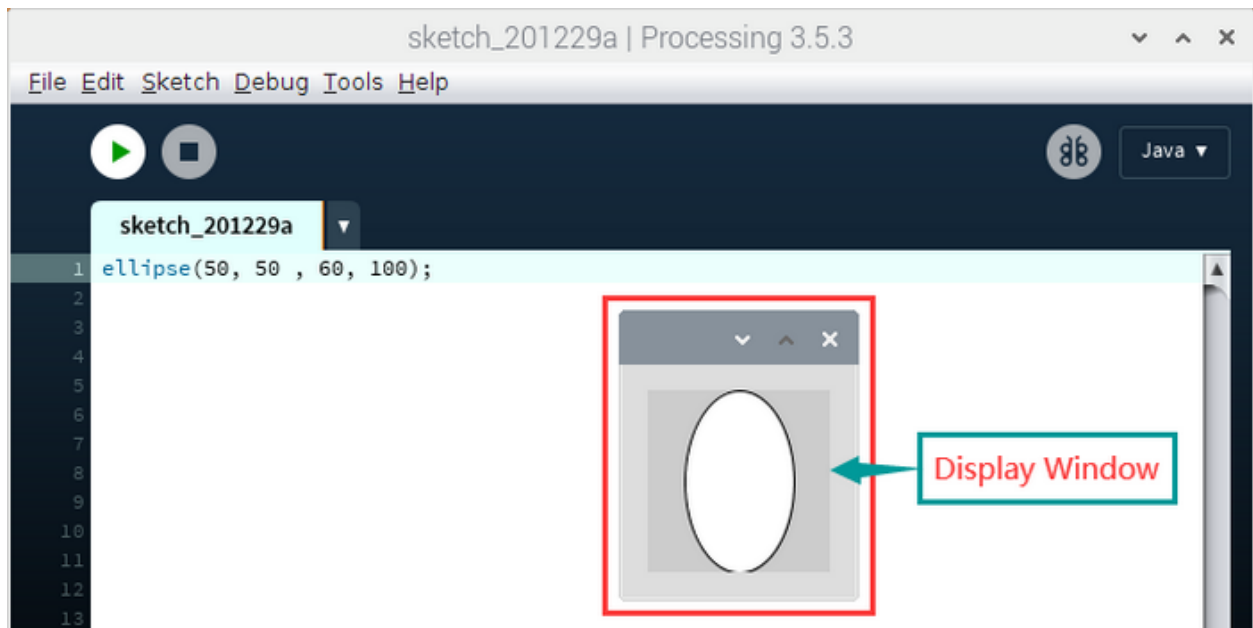



This code refers to “drawing an oval with a center 50 pixels from left to bottom, 50 pixels from top to bottom, width 60 and height 100 pixels.”

Click the “Run” button (the triangle button in the toolbar) .

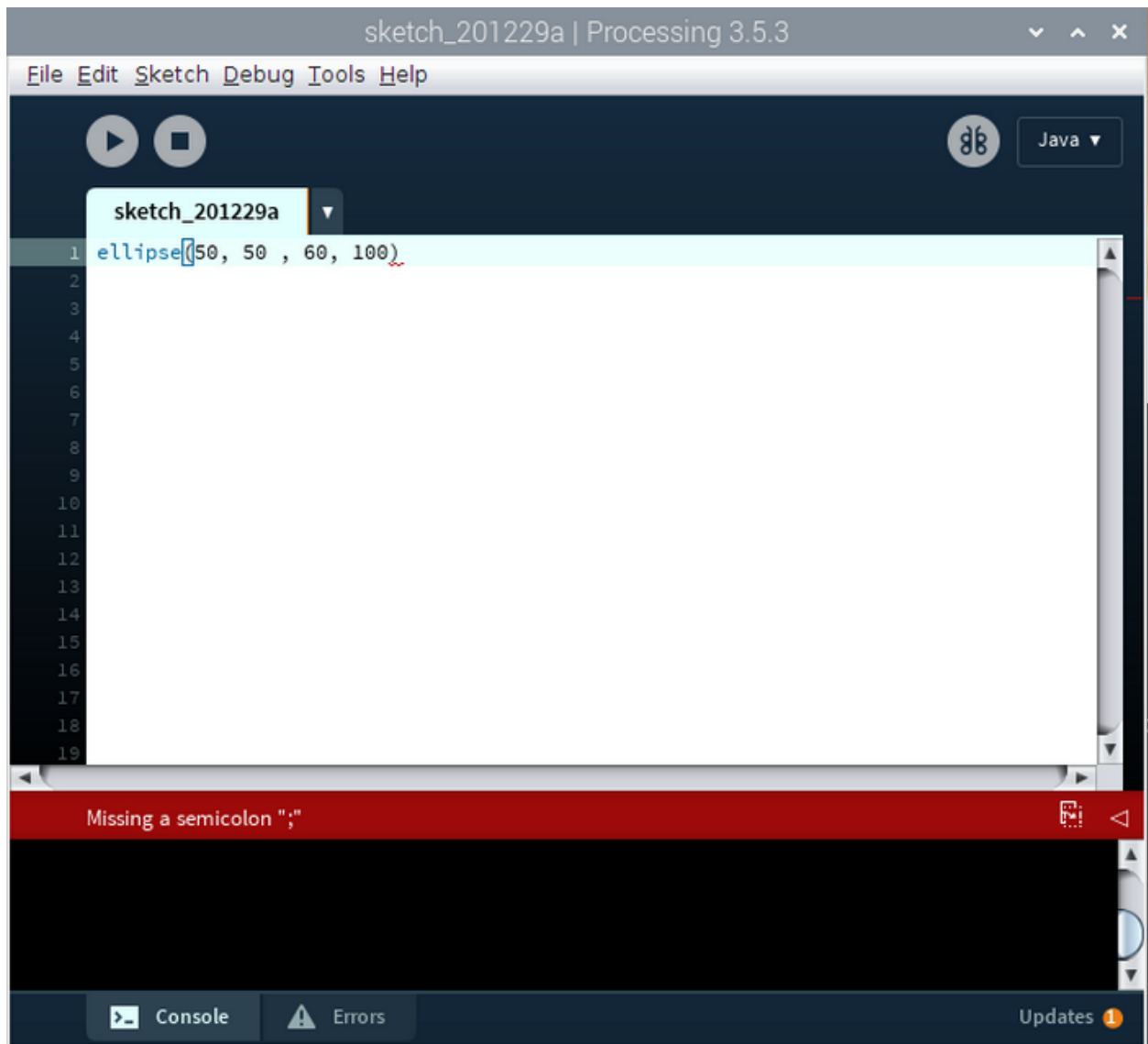


You will view an oval if all content is correct, as shown below:

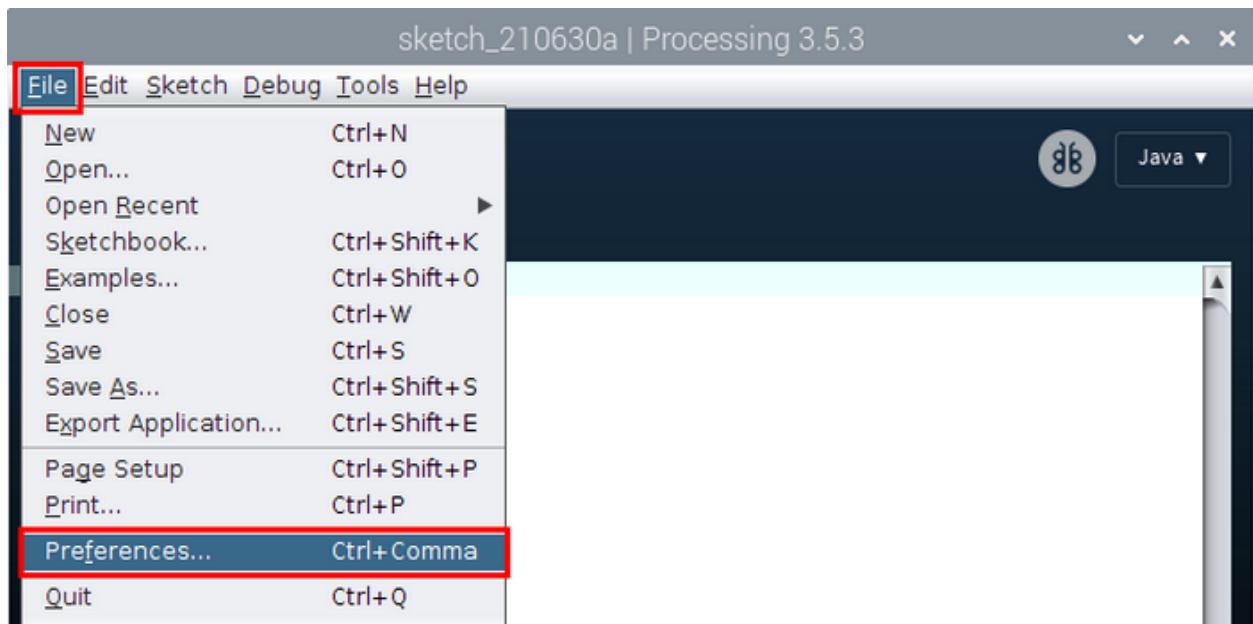


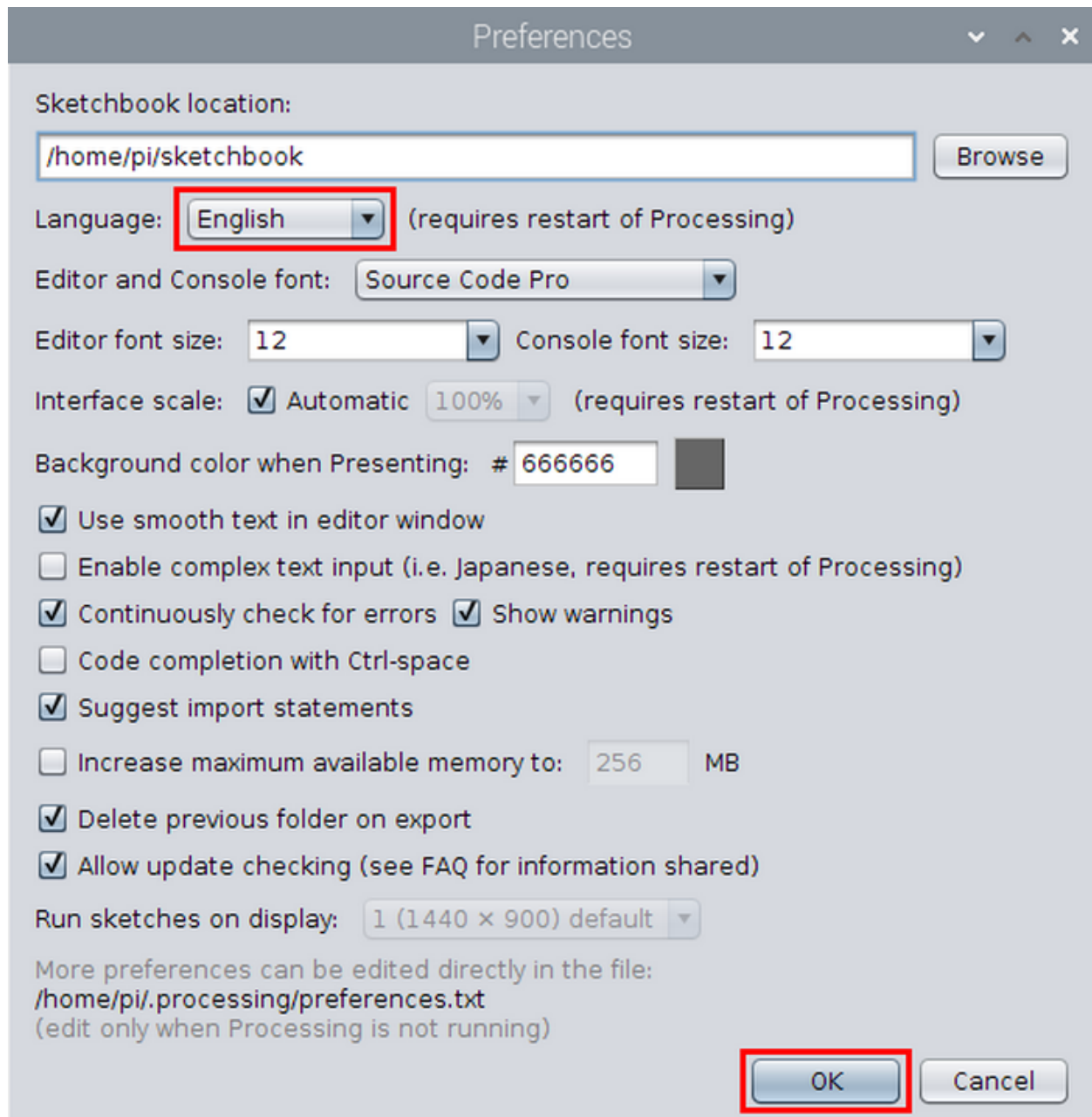
Click  to stop or close, the program will stop running.

A notification will appear if the input code is wrong. Don't worry about it. Check the code immediately. numbers are separated with a comma and enclosed in parentheses and each line should end with a semicolon. The wrong code is as follows:



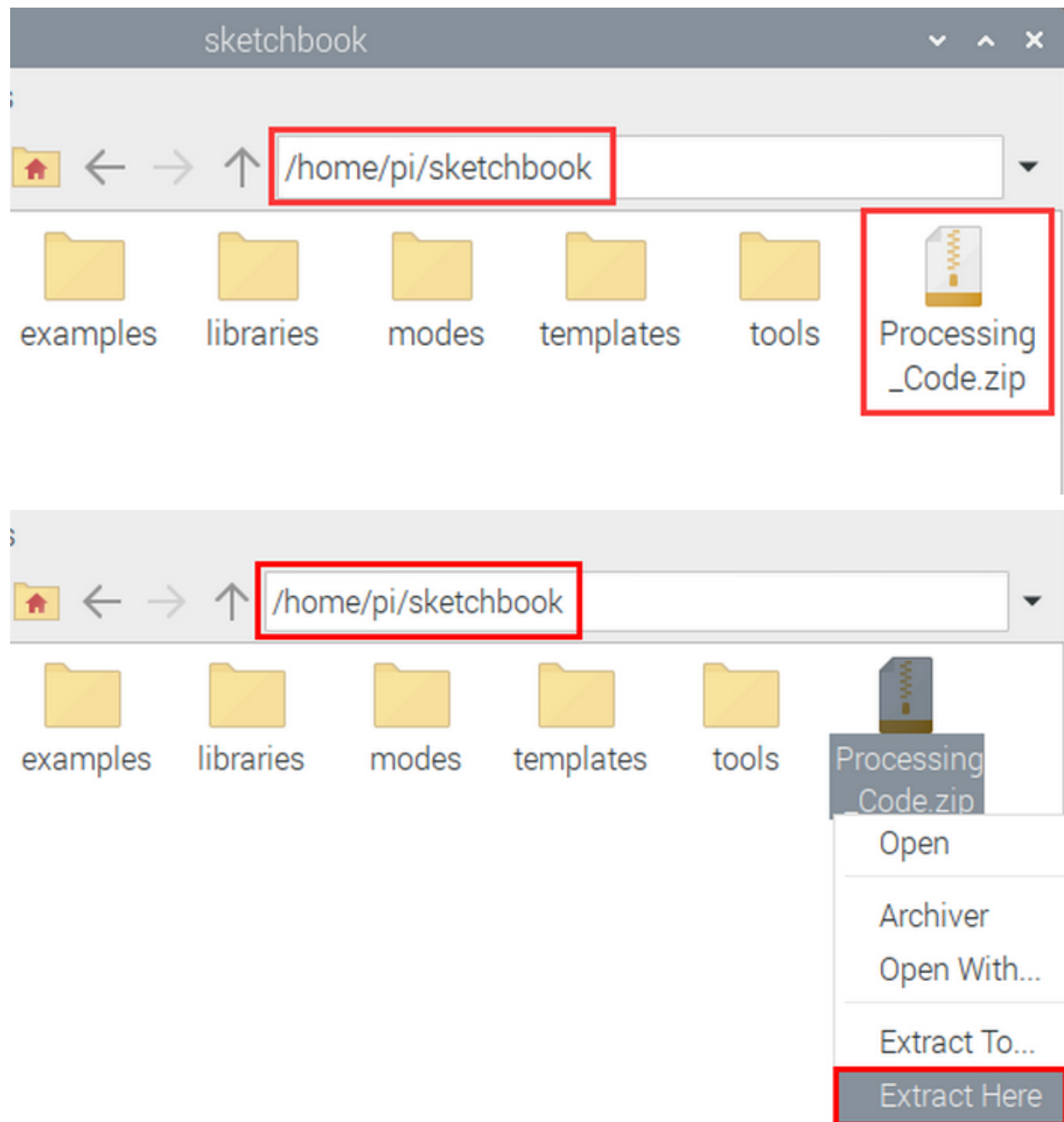
You could set up language mode and others in below page. Except language, others are all default settings.

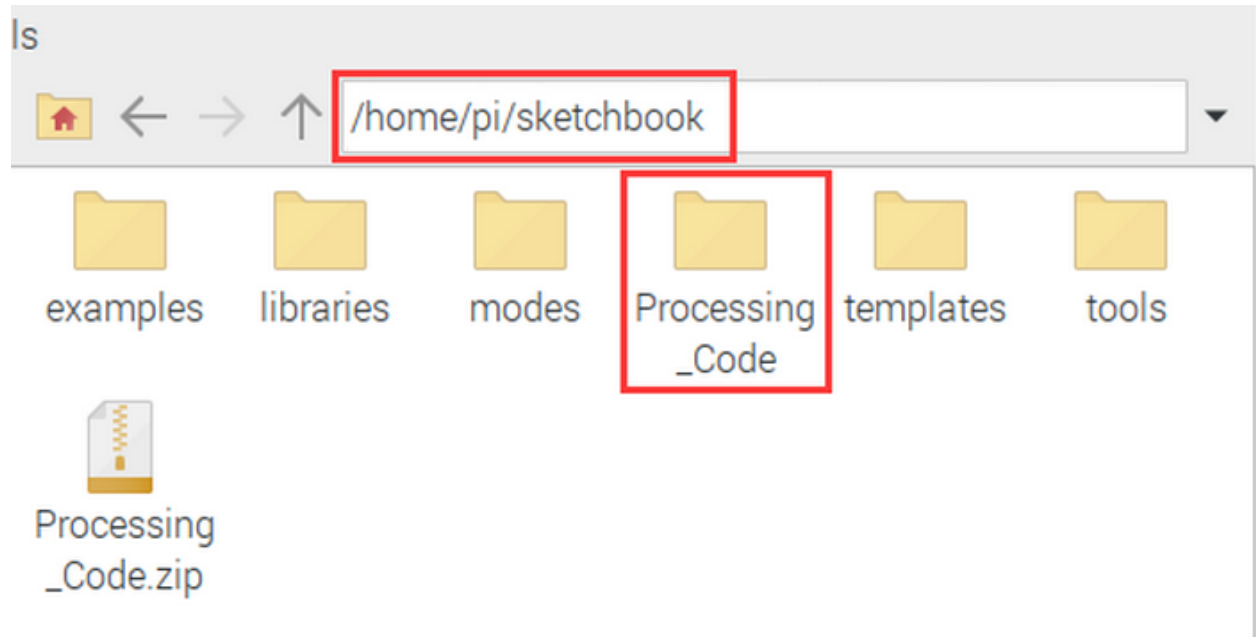




### 5.1.3 (3)Copy Example Code to Raspberry Pi

Copy the Processing-Code.zip to sketchbook folder and unzip it, as shown below:





## 5.2 2.Projects

### 5.2.1 Project 1Print Hello World

#### (1)Run Example Code

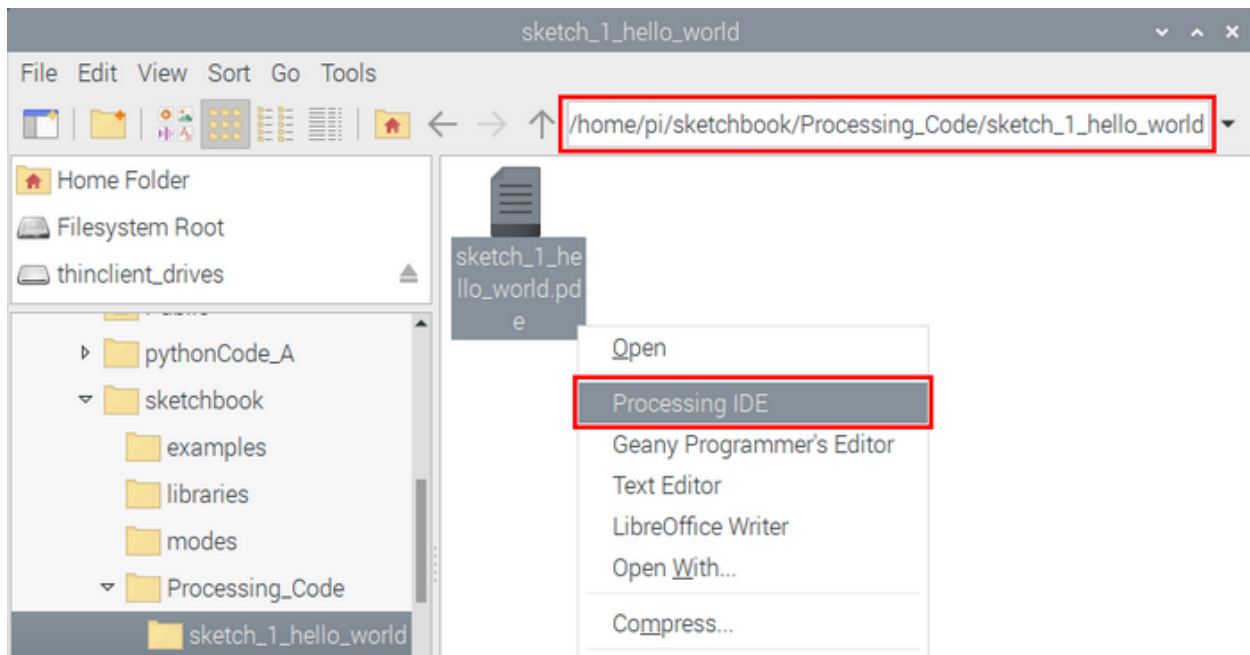
Input the following command and press“Enter”. Then Processing IDE will boot, click“RUN”:

```
processing /home/pi/sketchbook/Processing_Code/sketch_1_hello_world/sketch_1_hello_world.pde
```

Another method for your reference as below:

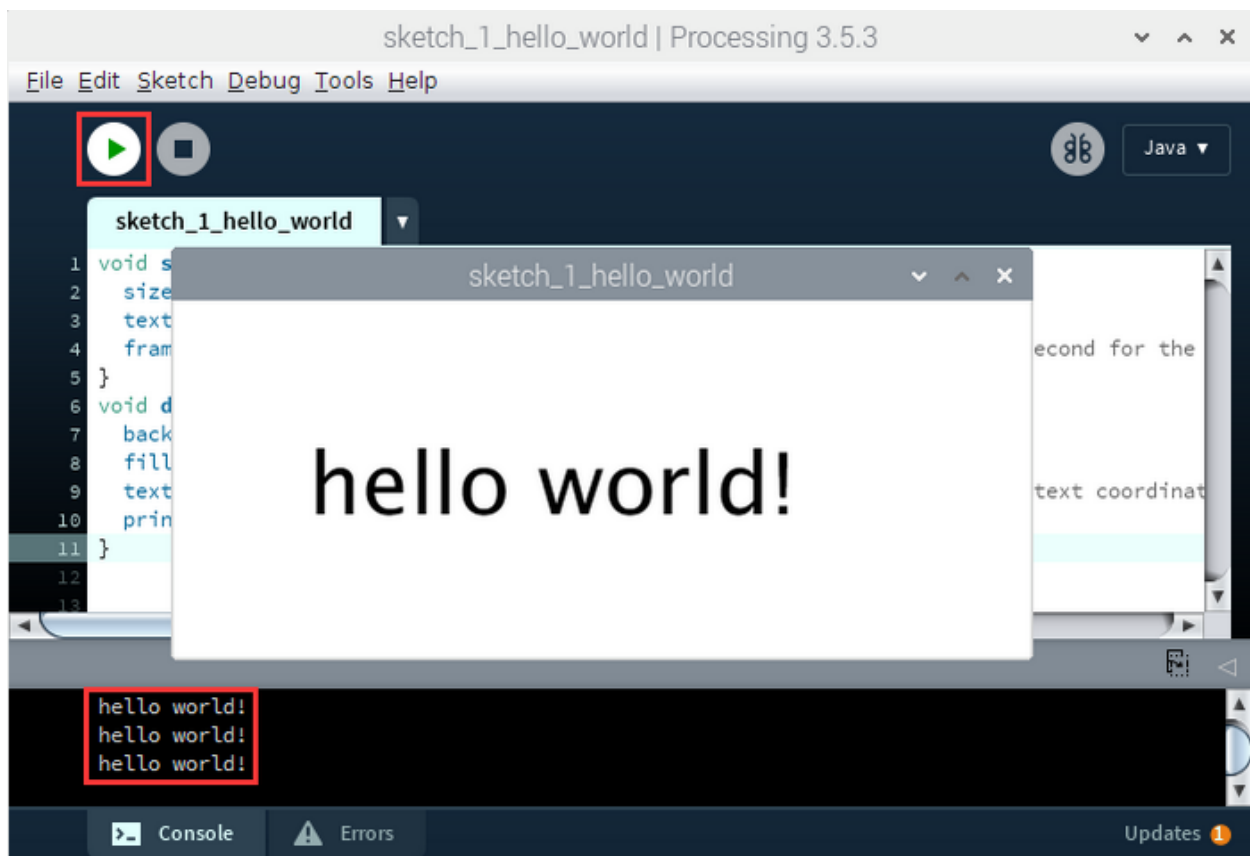
Click     to find out the route

/home/pi/sketchbook/Processing\_Code/sketch\_1\_hello\_worldthen right-click sketch\_1\_hello\_world.pde to select Processing IDE.



## (2)Test Results

Input "hello world" in the control window and hello world" appears in graphical display window, as shown below:



## (3)Example Code



```

void setup() { //execute only once when the program starts running
  size(480, 200); //set window size
  textSize(48); //set the font size
  frameRate(1); //To set the refresh rate,
  ↪ set the number of flushes per second for the draw () function
}
void draw() { //every frame is called once
  background(255); //full screen filled with solid color
  fill(1); //set the fill color of the words
  text("hello world!", 75, 120); //draw text,
  ↪ the lower left corner of the text coordinates are (75,120)
  println("hello world!"); //Output in the console window
}

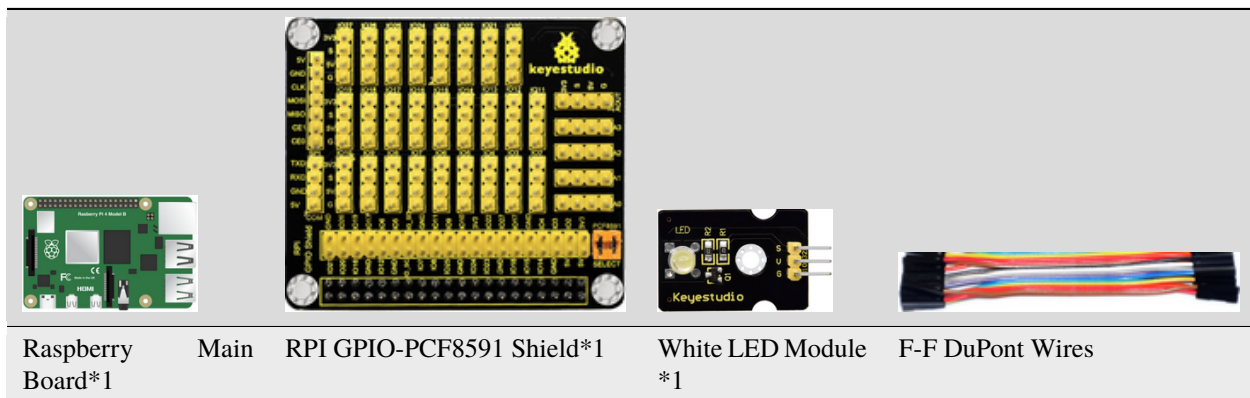
```

## 5.2.2 Project 2LED Blinks

### (1)Description

Let's start from a rather basic and simple experiment—LED Blinks

### (2)Components Needed:



### (3) Knowledge about Component :

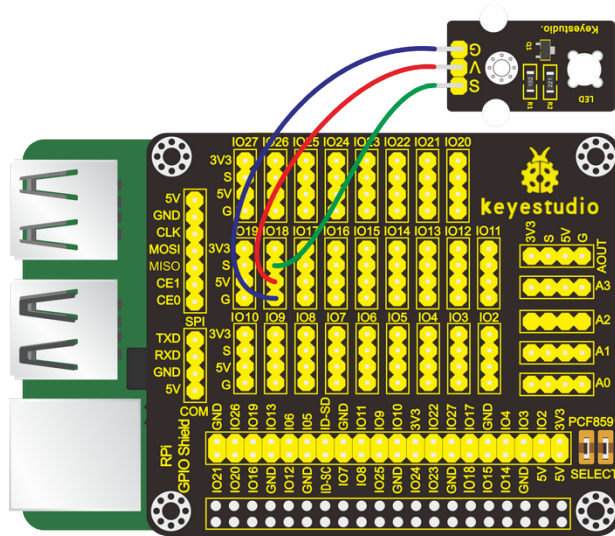
#### The white LED module:

It is a commonly used LED module. It is a F5 LED with white appearance and white light display. During experiments, when the GND and VCC on the module are powered up and the signal end S is at high level ,the white LED is on while when the S is at low level, the LED is off.

This module is compatible with various microcontrollers, including the Arduino series.

### (4)Connection Diagram

White LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



### (5)Working Principle

According to the diagram above we can find out that the positive pole(V) is connected to 5V, negative pole(G) to GND and signal terminal(S) to the pin of GPIO18. When GPIO18 outputs high level, LED is on; when it outputs low level, LED is off.

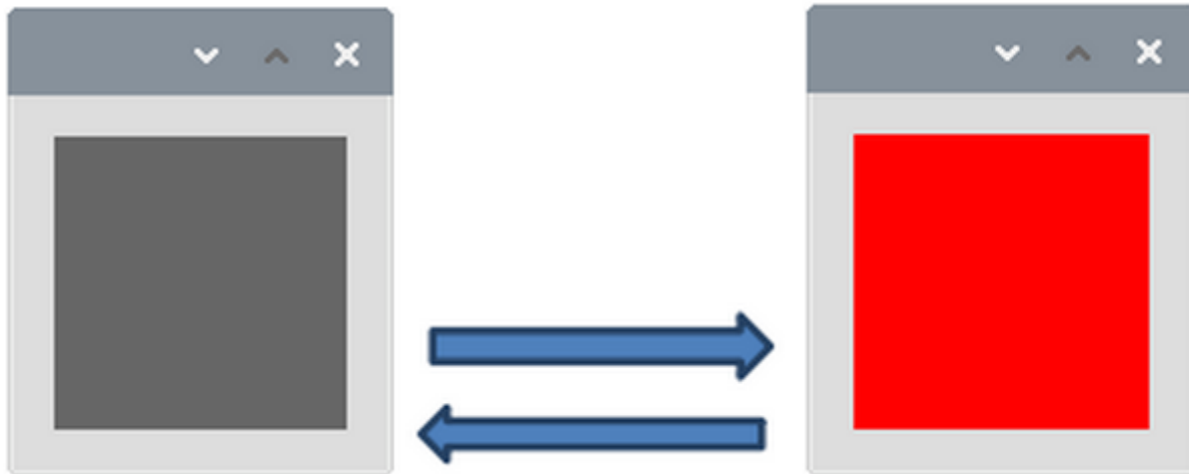
### (6)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_2_LED_Blinking/sketch_2_LED_Blinking.pde
```

### (7)Test Results

LED starts blinking and the background of display window varies with the state of LED, as shown below:



### (8)Example Code

```
import processing.io.*;

int ledPin = 18;    //define ledPin
boolean ledState = false;    //define ledState

void setup() {
```

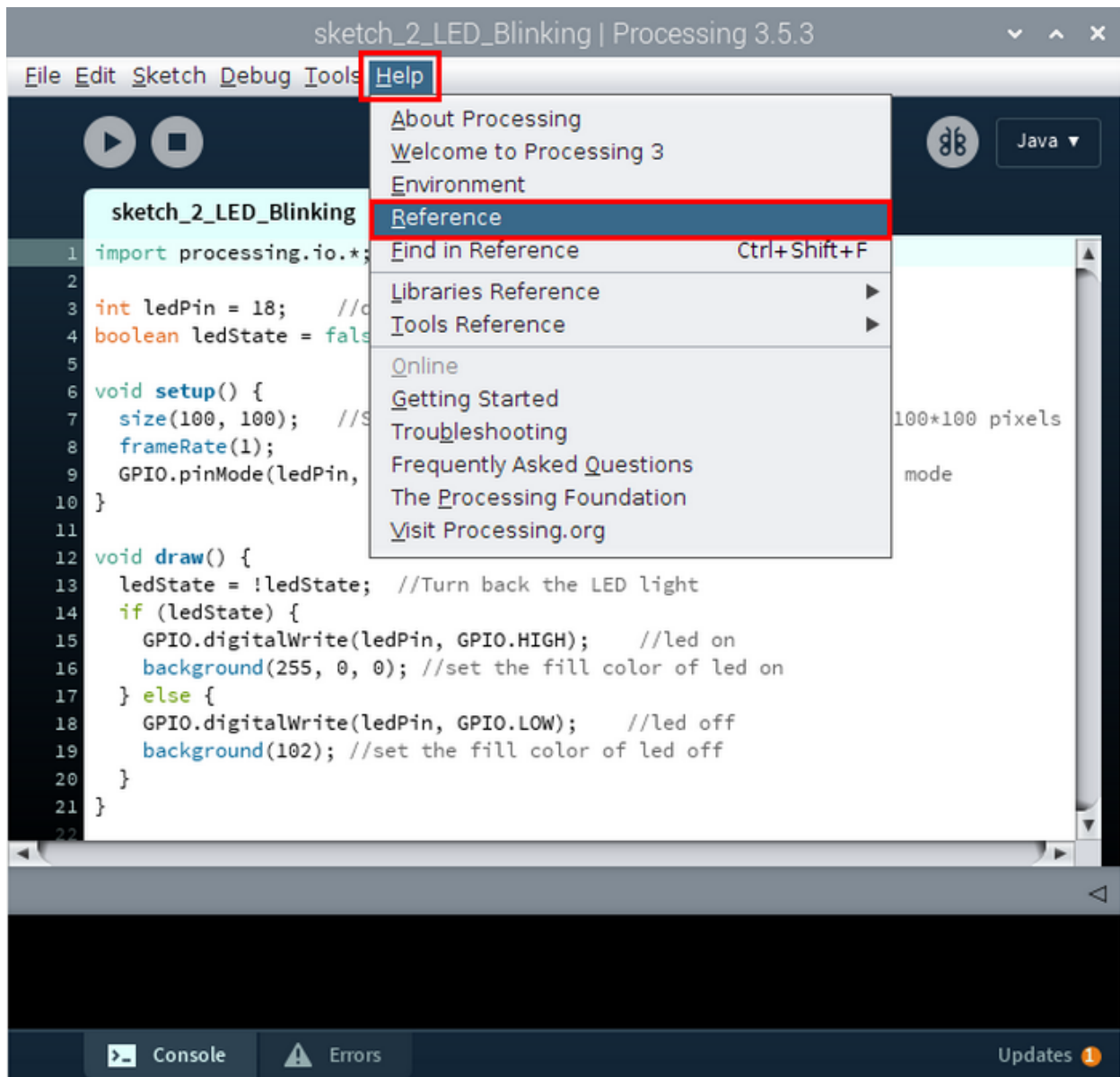
(continues on next page)

(continued from previous page)

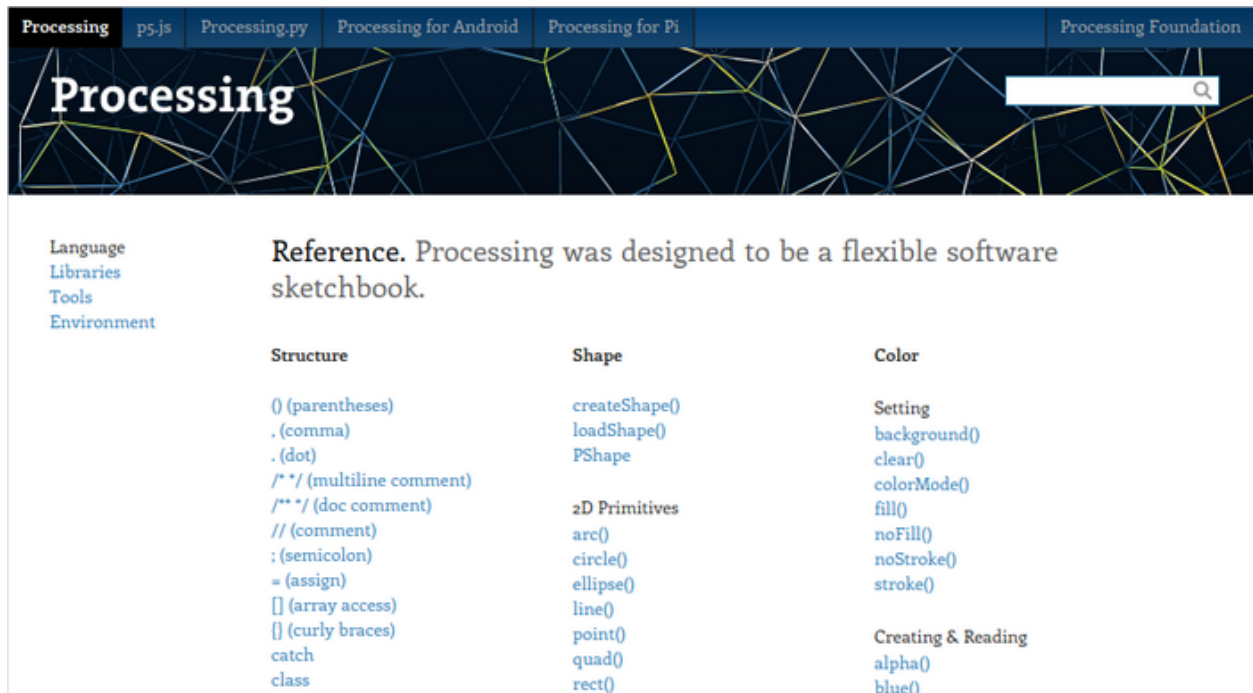
```
size(100, 100); //Set the size of the graphics display box to 100*100 pixels
frameRate(1); //set frame rate
GPIO.pinMode(ledPin, GPIO.OUTPUT); //set the ledPin to output mode
}

void draw() {
  ledState = !ledState; //Turn back the LED light
  if (ledState) {
    GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
    background(255, 0, 0); //set the fill color of led on
  } else {
    GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
    background(102); //set the fill color of led off
  }
}
```

The function of the above code is included in Processing Software. You could look through detailed information and reference by clicking “Help”→“Reference”, as shown below:



Then the following page shows;



Equally, you could navigate the official website <http://processing.org/reference>

## 5.2.3 Project 3 Mouse-controlled LED

### (1) Description

In this program, we will control the status of LED by mouse. The components, connection and schematic diagrams are same as the lesson 2.

### (2) Working Principle:

Left-click the display window, LED is on; right-click the display window, LED is off.

### (3) Run Example Code

Input the following command, press "Enter" and click "RUN" on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_3_mouse_led/sketch_3_mouse_led.pde
```

### (4) Test Results

After running example code, LED is off and display window is in gray color. Left-click the gray area, LED is on and window turns into red; right-click display window, LED is off and its background color is gray-black as shown below:



### (5)Example Code

```
import processing.io.*;

int ledPin = 18;    //define ledPin

void setup()
{
  size(100, 100);
  GPIO.pinMode(ledPin, GPIO.OUTPUT);
}

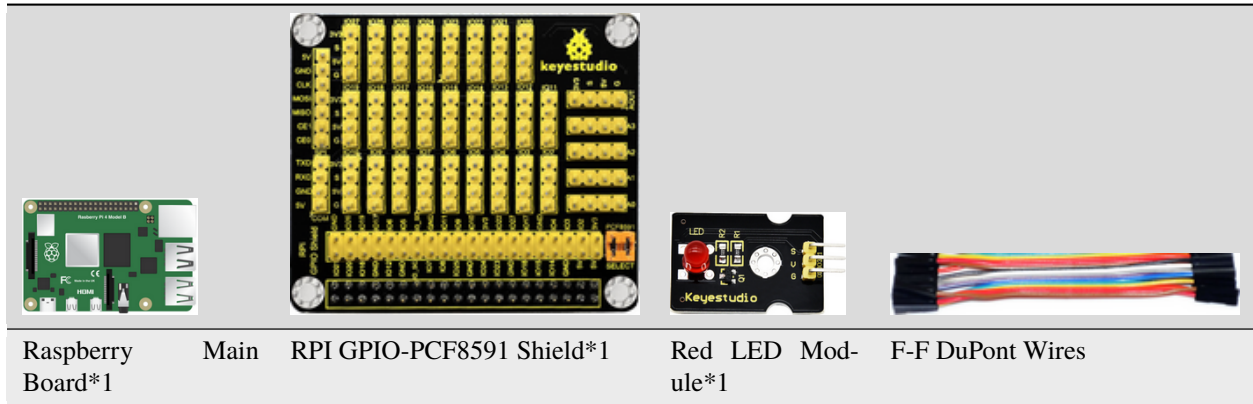
void draw() {
  if (mousePressed && (mouseButton == LEFT)) {
    background(255, 0, 0); //set the fill color of led on
    GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
  }
  if (mousePressed && (mouseButton == RIGHT)) {
    background(102); //set the fill color of led off
    GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
  }
}
```

## 5.2.4 Project 4Breathing LED

### (1)Description

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.

### (2)Components Needed



### (3)Working Principle

We use the PWM output of GPIO, PWM outputs analog signals and output value is 0~100 which is equivalent to output voltage 0~3.3V from GPIO port.

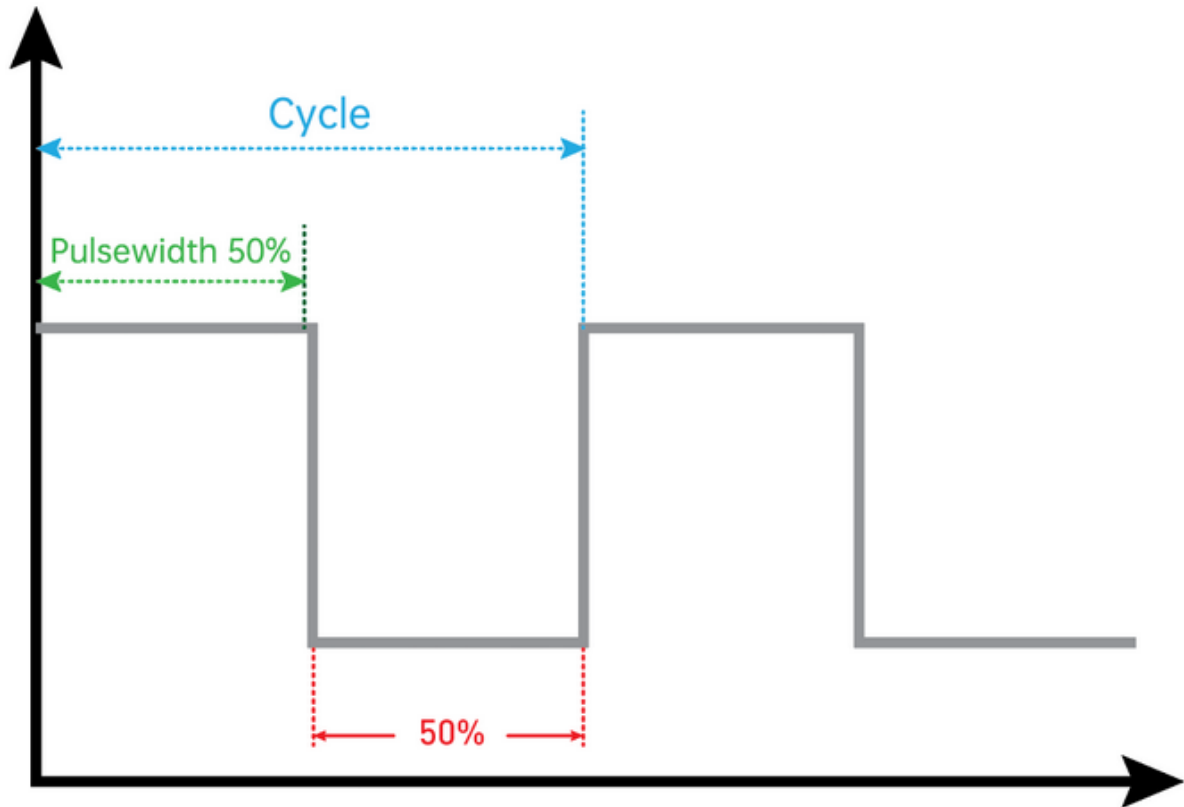
According to Ohm's law:  $U/R = I$ , the resistance is 220, and the value of voltage  $U$  changes, so does the value of current  $I$ , which can control the brightness of the LED lamp.

PWM (Pulse Width Modulation) is the control of the analog circuit through the digital output of microcomputer and a method that makes digital coding on analog signal levels.

It sends square waves with certain frequency through digital pins, that is, high level and low level output alternately for a period of time. Total time of each group high and low level is fixed, which is called cycle.

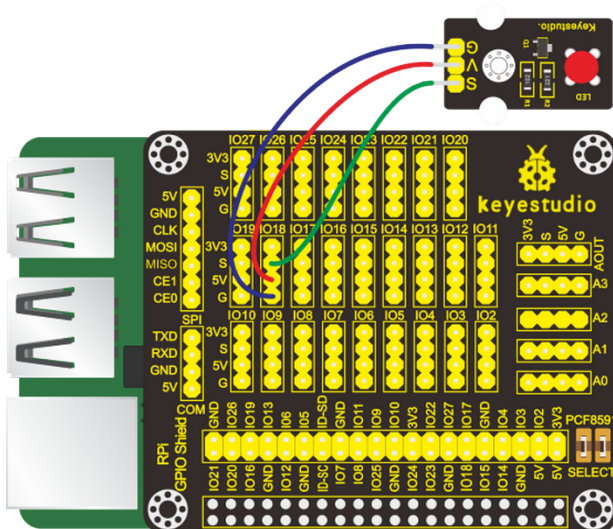
The time of high level output is pulse width whose percentage is called Duty Cycle. The longer that high level lasts, the larger the duty cycle of analog signals is, and the corresponding voltage as well.

Below chart is pulse width 50%, then the output voltage is  $3.3 * 50\% = 1.65V$  and the brightness of LED is medium.



#### (4) Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### (5) Run Example Code



Input the following the command and press“Enter”, open Processing IDE and click“RUN”

processing /home/pi/sketchbook/Processing\_Code/sketch\_4\_Breathing\_LED/sketch\_4\_Breathing\_LED.pde

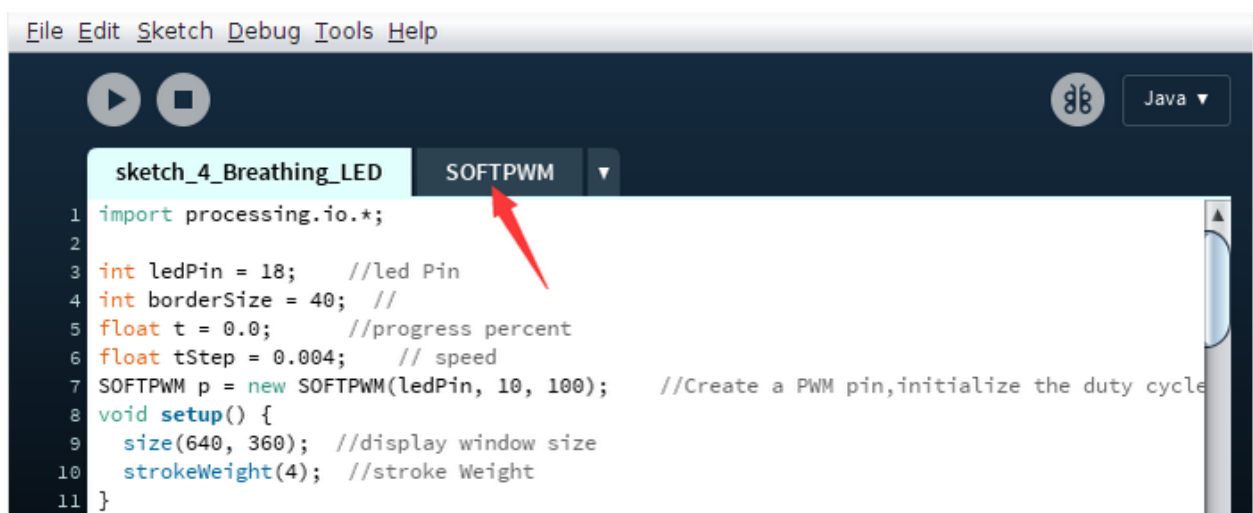
#### (6)Test Results

LED gradually brightens, and the color of red dot in the display window gets darker as well. Progress bar can adjust the LED's brightness, as shown below:



#### (7)Example Code

Except main program code, there is a“SOFTPWM”custom page in processing software, as shown below:



The code of the main program:

```

import processing.io.*;

int ledPin = 18;    //led Pin
int borderSize = 40; //
float t = 0.0;      //progress percent
float tStep = 0.004; // speed
SOFTPWM p = new SOFTPWM(ledPin, 10, 100);    //Create a PWM pin,
↳ initialize the duty cycle and period
void setup() {
    size(640, 360); //display window size
    strokeWeight(4); //stroke Weight
}

void draw() {
    // Show static value when mouse is pressed, animate otherwise
    if (mousePressed) {
        //Gets the value of the X-axis coordinate when the mouse is pressed,
        //within the (borderSize, width-borderSize) range
        int a = constrain(mouseX, borderSize, width - borderSize);
        t = map(a, borderSize, width - borderSize, 0.0, 1.0); //
        ↳ Gets the value after the mapping
    } else {
        t += tStep; //The value of the variable T increases automatically
        if (t > 1.0) t = 0.0;
    }
    p.softPwmWrite((int)(t*100)); //wirte the duty cycle according to t
    background(255); //A white background
    titleAndSiteInfo(); //title and Site infomation

    //The brightness of the red circle varies with the value of T
    fill(255, 255-t*255, 255-t*255);
    //The center of the display box is a circle with a diameter of 100px
    ellipse(width/2, height/2, 100, 100);

    pushMatrix();
    translate(borderSize, height - 45); //Set the new origin of coordinates
    int barLength = width - 2*borderSize; //Define the length of the line

    barBgStyle(); //progressbar bg
    line(0, 0, barLength, 0); //A horizontal line
    //Draw a 10px vertical line at the end of the horizontal line
    line(barLength, -5, barLength, 5);
    barStyle(); //progressbar
    //Draw a 10px vertical line at the beginning of the horizontal line
    line(0, -5, 0, 5);
    //Draw the length of the black line according to the value of the variable t
    line(0, 0, t*barLength, 0);

    barLabelStyle(); //progressbar label
    text("progress : "+nf(t*100,2,2),barLength/2,-25);
    popMatrix();
}

```

(continues on next page)

(continued from previous page)

```

void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);         //set text size
  text("Breathing Light", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}

void barBgStyle() {
  stroke(220);
  noFill();
}

void barStyle() {
  stroke(50);
  noFill();
}

void barLabelStyle() {
  noStroke();
  fill(120);
}

```

#### (8)Reference

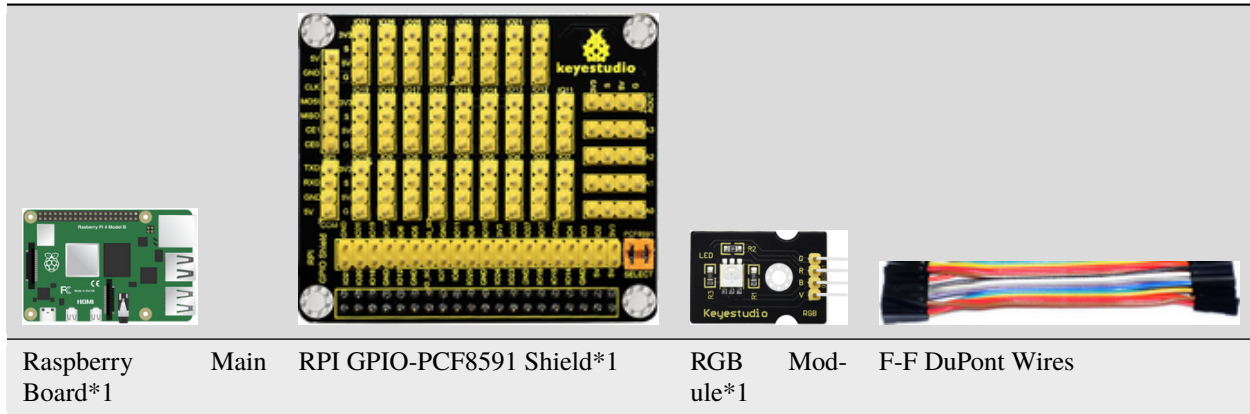
class SOFTPWM		
public SOFTPWM**( <b>int</b> iPin,** int dc**,** int pwmRange**)**	Construct functionused to create PWM pinset pwmRange and initial duty cycle The time of minimum duty cycle of pwmRange is 0.1ms pwmRange=100 means than PWM duty cycle is 0.1ms*100=10ms	
public void softPwmWrite**( <b>int value</b> )**	Set PMW Duty Cycle	
public void softPwmStop**()**	Stop outputting PWM	

## 5.2.5 Project 5RGB

### (1)Description

In this chapter, we will demonstrate how RGB lights show different colors via programming.

### (2)Components Needed

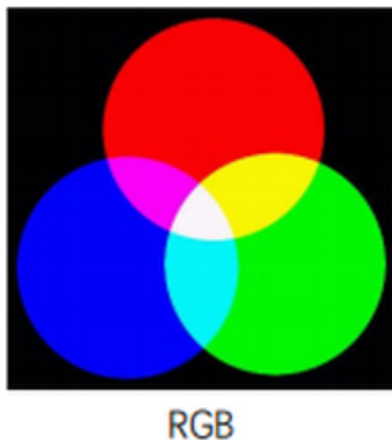


### (3) Knowledge about Component:

#### RGB Module

The RGB module integrates with three LEDs in red, green and blue respectively. These three LEDs also share the same anode. The combinations of these three colors can form almost all other colors visible to human eyes. Thus, it has found wide applications in terms of colors.

Red, green and blue are three primary colors. They could produce all kinds of visible lights when mixing them up. Computer screen, single pixel mobile phone screen, neon light work under this principle.

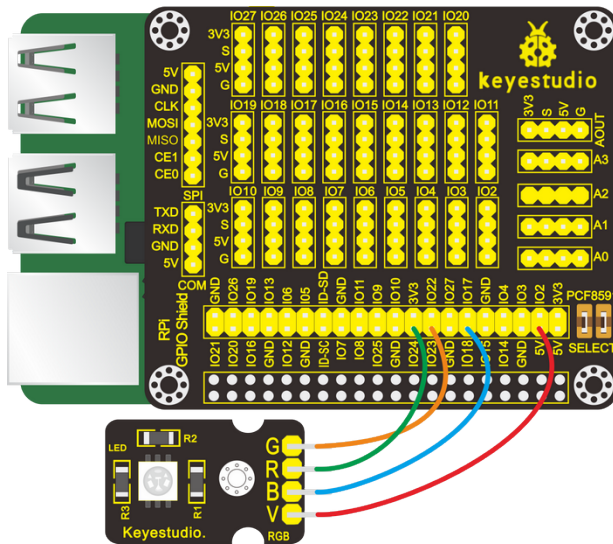


Theoretically, if we use three 8-bit PWM (Pulse Width Modulation) signals to control a RGB LED, we can create  $28 * 28 * 28 = 16777216$  (about 16 million) different combinations.

Now, let's make a RGB LED display all kinds of colors.

#### (4) Connection Diagram

RGB Module	RPI GPIO-PCF8591 Shield
R	IO24
G	IO23
B	IO18
V	5V



### (5)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

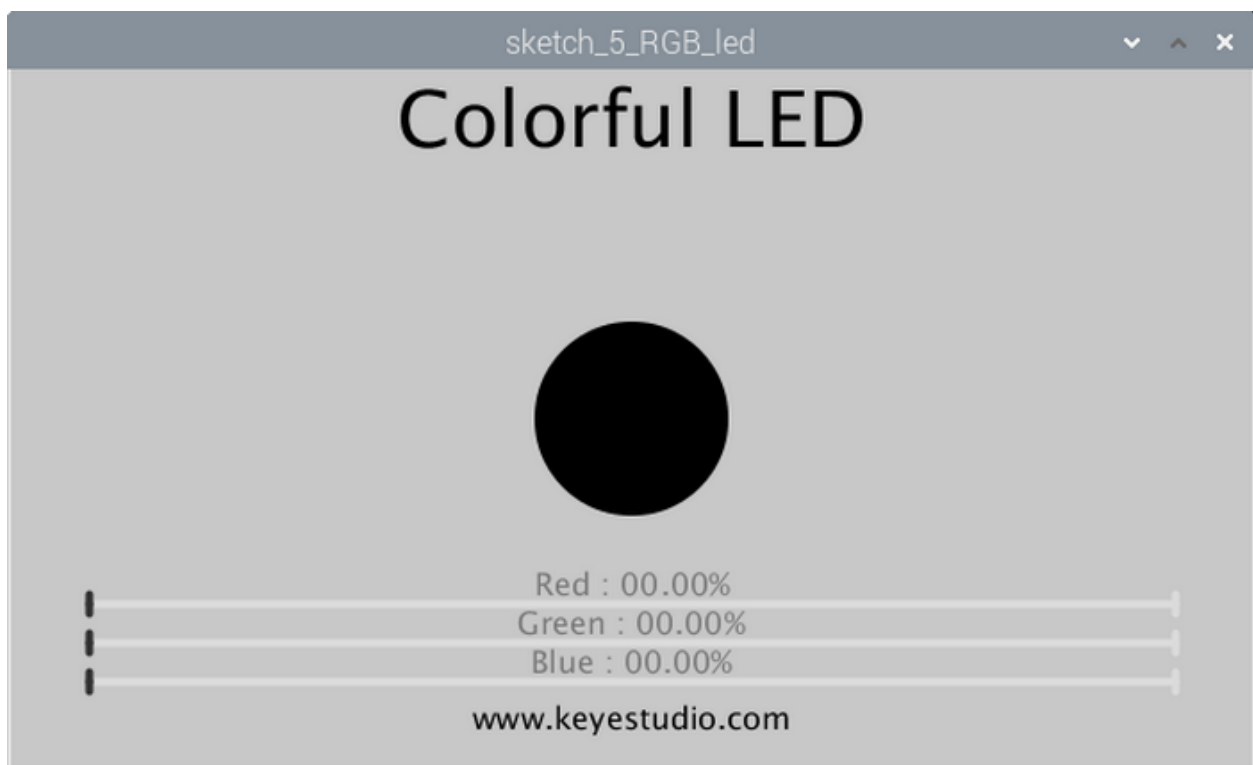
```
processing /home/pi/sketchbook/Processing_Code/sketch_5_RGB_led/sketch_5_RGB_led.pde
```

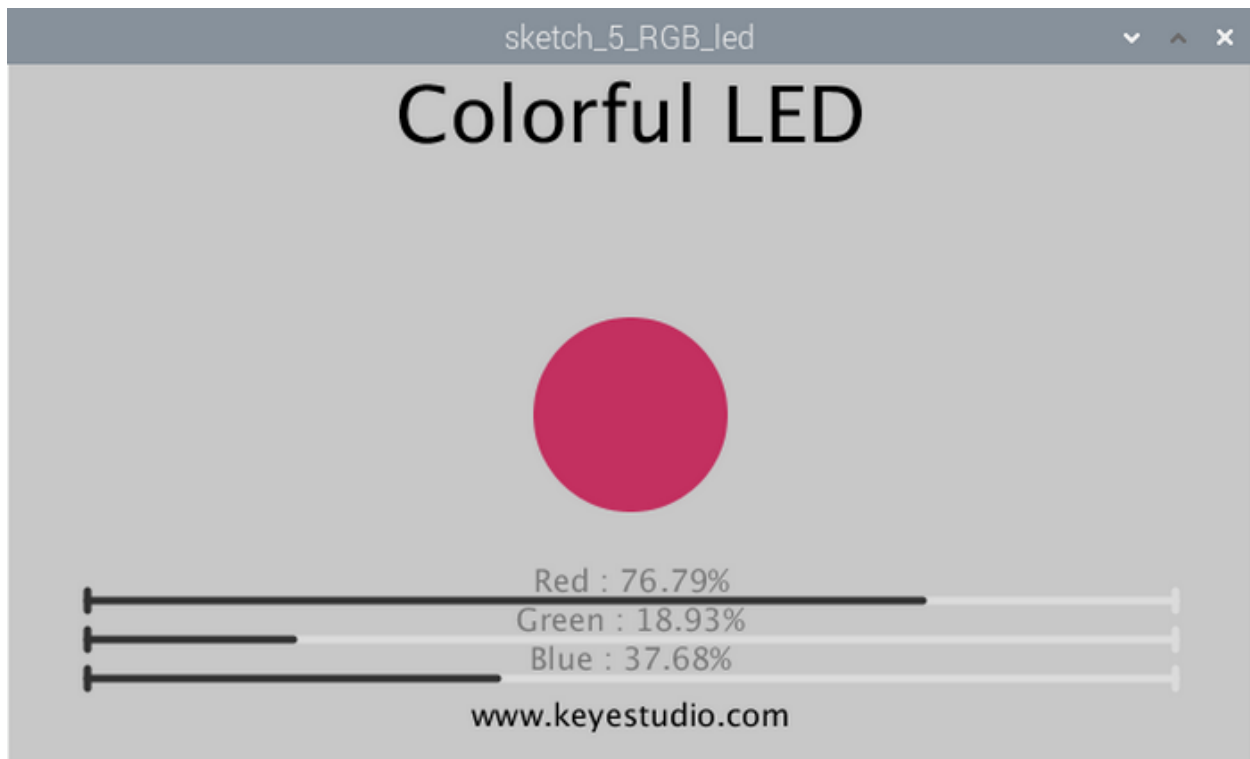
### (6)Test Results

After running the program, RGB LED is off, the window displays the round dot in black and the progress bars for red, green and blue are 0%.

The round dot will change colors when dragging the progress bar to set PWM duty cycle for each color channel.

The color of RGB is as same as that of the round dot.

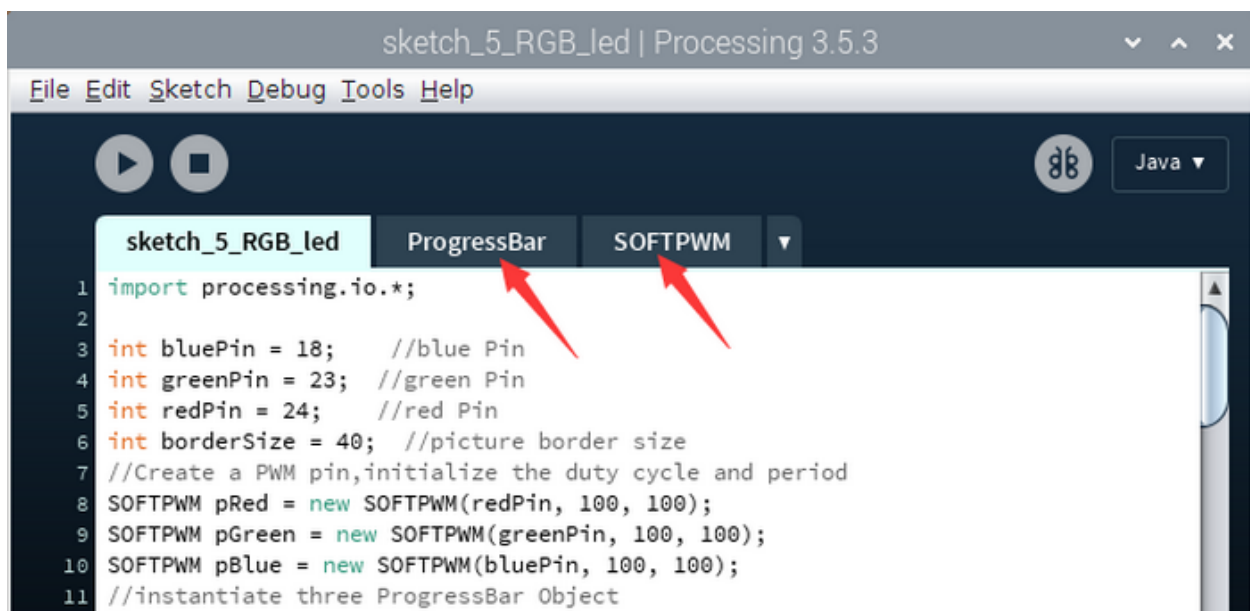




### (7) Example Code

This project contains a lot of code files, the core code is contained in the file sketch\_5\_RGB\_led.pde.

Other files are customized. As shown below:



Code:

```

import processing.io.*;

int bluePin = 18;    //blue Pin
  
```

(continues on next page)

(continued from previous page)

```

int greenPin = 23; //green Pin
int redPin = 24;   //red Pin
int borderSize = 40; //picture border size
//Create a PWM pin, initialize the duty cycle and period
SOFTPWM pRed = new SOFTPWM(redPin, 100, 100);
SOFTPWM pGreen = new SOFTPWM(greenPin, 100, 100);
SOFTPWM pBlue = new SOFTPWM(bluePin, 100, 100);
//instantiate three ProgressBar Object
ProgressBar rBar, gBar, bBar;
boolean rMouse = false, gMouse = false, bMouse = false;
void setup() {
    size(640, 360); //display window size
    strokeWeight(4); //stroke Weight
    //define the ProgressBar length
    int barLength = width - 2*borderSize;
    //Create ProgressBar Object
    rBar = new ProgressBar(borderSize, height - 85, barLength);
    gBar = new ProgressBar(borderSize, height - 65, barLength);
    bBar = new ProgressBar(borderSize, height - 45, barLength);
    //Set ProgressBar's title
    rBar.setTitle("Red"); gBar.setTitle("Green"); bBar.setTitle("Blue");
}

void draw() {
    background(200); //A white background
    titleAndSiteInfo(); //title and Site infomation

    fill(rBar.progress*255, gBar.progress*255, bBar.progress*255); //cycle color
    ellipse(width/2, height/2, 100, 100); //show cycle

    rBar.create(); //Show progressBar
    gBar.create();
    bBar.create();
}

void mousePressed() {
    if ( (mouseY < rBar.y+5) && (mouseY > rBar.y-5) ) {
        rMouse = true;
    } else if ( (mouseY < gBar.y+5) && (mouseY > gBar.y-5) ) {
        gMouse = true;
    } else if ( (mouseY < bBar.y+5) && (mouseY > bBar.y-5) ) {
        bMouse = true;
    }
}

void mouseReleased() {
    rMouse = false;
    bMouse = false;
    gMouse = false;
}

void mouseDragged() {
    int a = constrain(mouseX, borderSize, width - borderSize);
    float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
}

```

(continues on next page)

(continued from previous page)

```

if (rMouse) {
  pRed.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
  rBar.setProgress(t);
} else if (gMouse) {
  pGreen.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
  gBar.setProgress(t);
} else if (bMouse) {
  pBlue.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
  bBar.setProgress(t);
}
}

void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);         //set text size
  text("Colorful LED", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}

```

## (8)Reference

class ProgressBar used to create progress bar	
public ProgressBar**(int ix,** int iy,** int barlen**)**	Constructed function, used to create ProgressBar, coordinates X, Y of ProgressBar and length
public void setTitle**(String str)**	Used to set the name of progress bar and display it in the middle of progress bar
public void setProgress**(float pgress)**	Used to set the process of progress bar parameter 0<pgress<1.0.
public void create**() &** public void create**(float pgress)**	Used to draw the progress bar

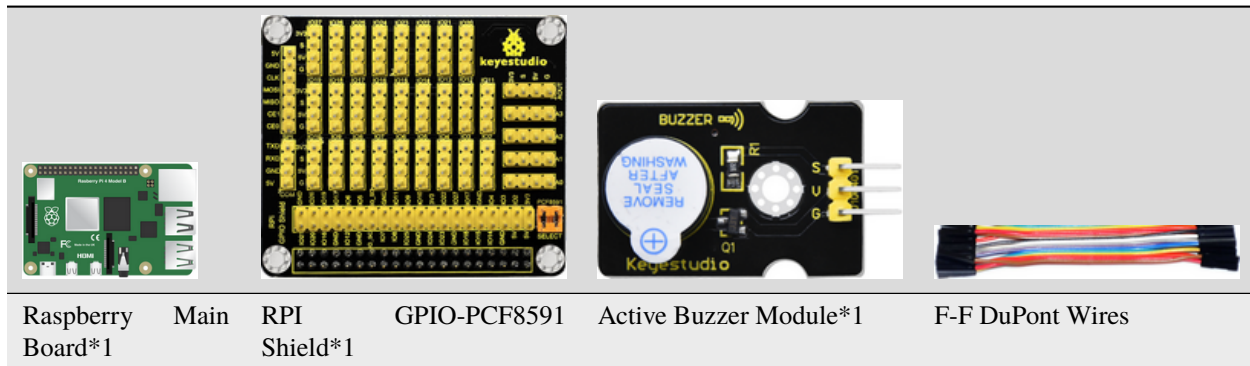
## 5.2.6 Project 6Active Buzzer

### (1)Description

In this project, we will control a active buzzer via a mousse.

### (2)Components Needed



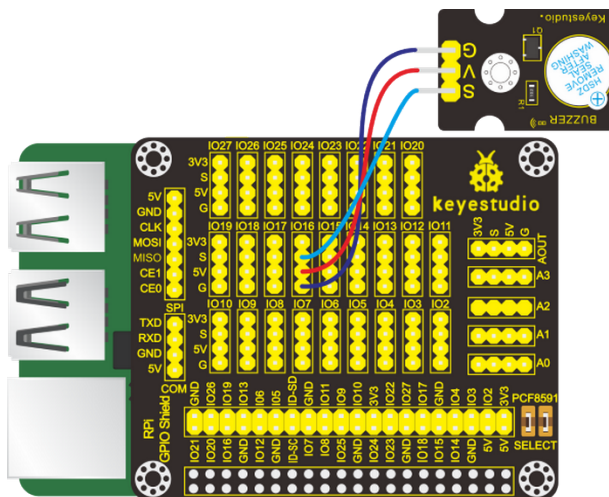


### (3)Knowledge about Component:

**\*\*Active Buzzer Module\*\***The active buzzer is equipped with an internal oscillator, which makes it possible to automatically generate a tone as long as current flows through. It is very easy and convenient. But it also has its shortcoming that the fixed frequency means it can only makes a monotone.

### (4)Connection Diagram

Active Buzzer Module	RPI GPIO-PCF8591 Shield
S	SIO16
V	5V
G	G



### (5)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_6_active_buzzer/sketch_6_active_buzzer.pde
```

### (6)Test Results

Click any area of the display window, active buzzer emits sound and the icon on display window below varies with the status of active buzzer. When it stops the icon disappears.



#### (7)Example Code

```
import processing.io.*;  
int buzzerPin = 16;
```

(continues on next page)

(continued from previous page)

```

boolean buzzerState = false;
void setup() {
  size(640, 360);
  GPIO.pinMode(buzzerPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo();    //title and site infomation
  drawBuzzer();          //buzzer img
  if (buzzerState) {
    GPIO.digitalWrite(buzzerPin, GPIO.HIGH); // buzzer on
    drawArc();           //Sounds waves img
  } else {
    GPIO.digitalWrite(buzzerPin, GPIO.LOW);  // buzzer off
  }
}

void mouseClicked() { //if the mouse Clicked
  buzzerState = !buzzerState; //Change the buzzer State
}

void drawBuzzer() {
  strokeWeight(1);
  fill(0);
  ellipse(width/2, height/2, 50, 50);
  fill(255);
  ellipse(width/2, height/2, 10, 10);
}

void drawArc() {
  noFill();
  strokeWeight(8);
  for (int i=0; i<3; i++) {
    arc(width/2, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
  }
}

void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);          //set text size
  text("Active Buzzer", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}

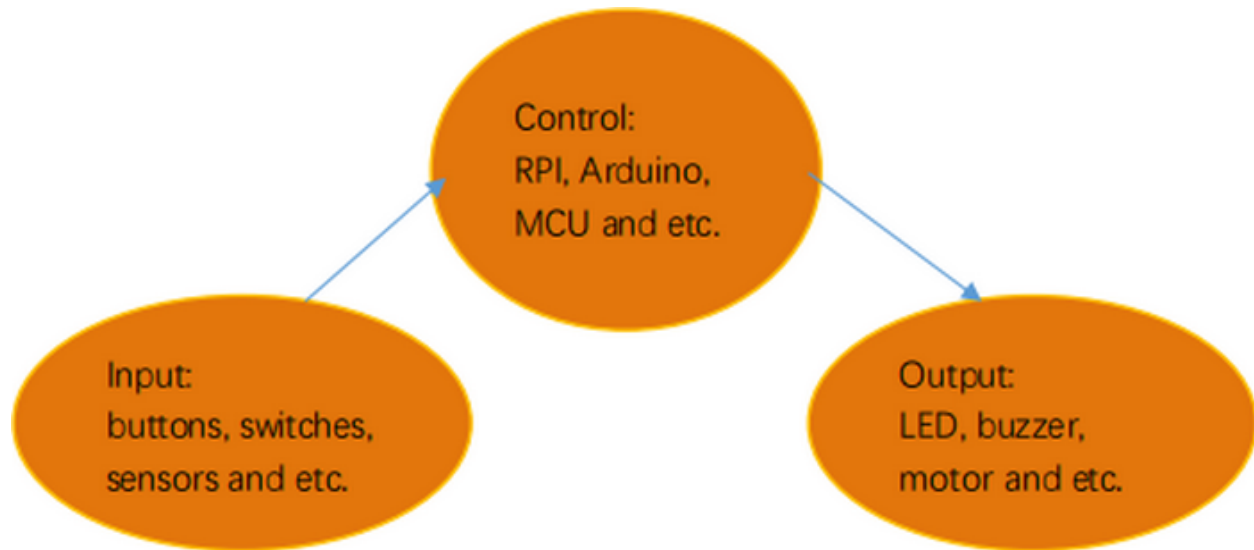
```

## 5.2.7 Project 7 Button-controlled LED

### (1)Description

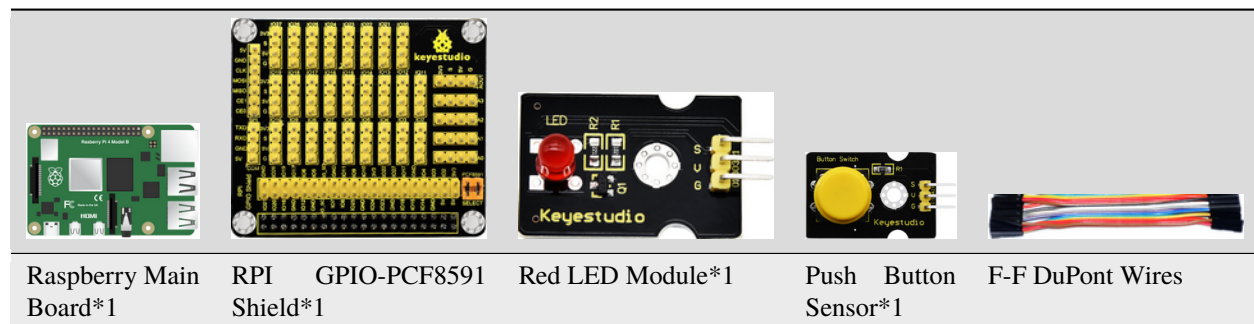
Usually a complete open loop control is made of external information input, controller and actuator.

The external information is input into controller which can analyze the input data and send to control signals to make actuator to react.



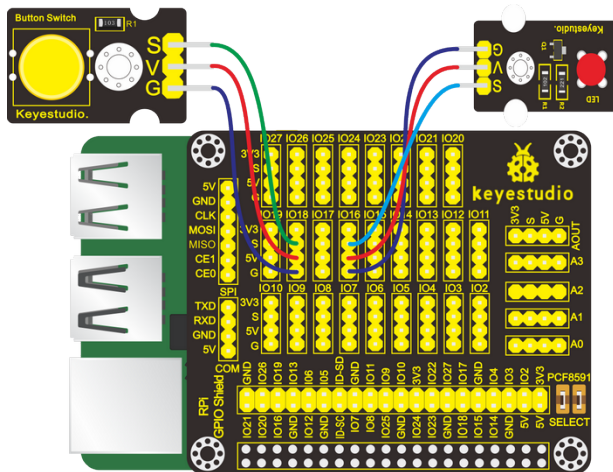
A button-controlled LED is decided by an open loop control. Next, we will make a desk lamp with a button, an LED and RPi. LED is on when button is pressed, on the contrary, it will be off.

### (2)Components Needed



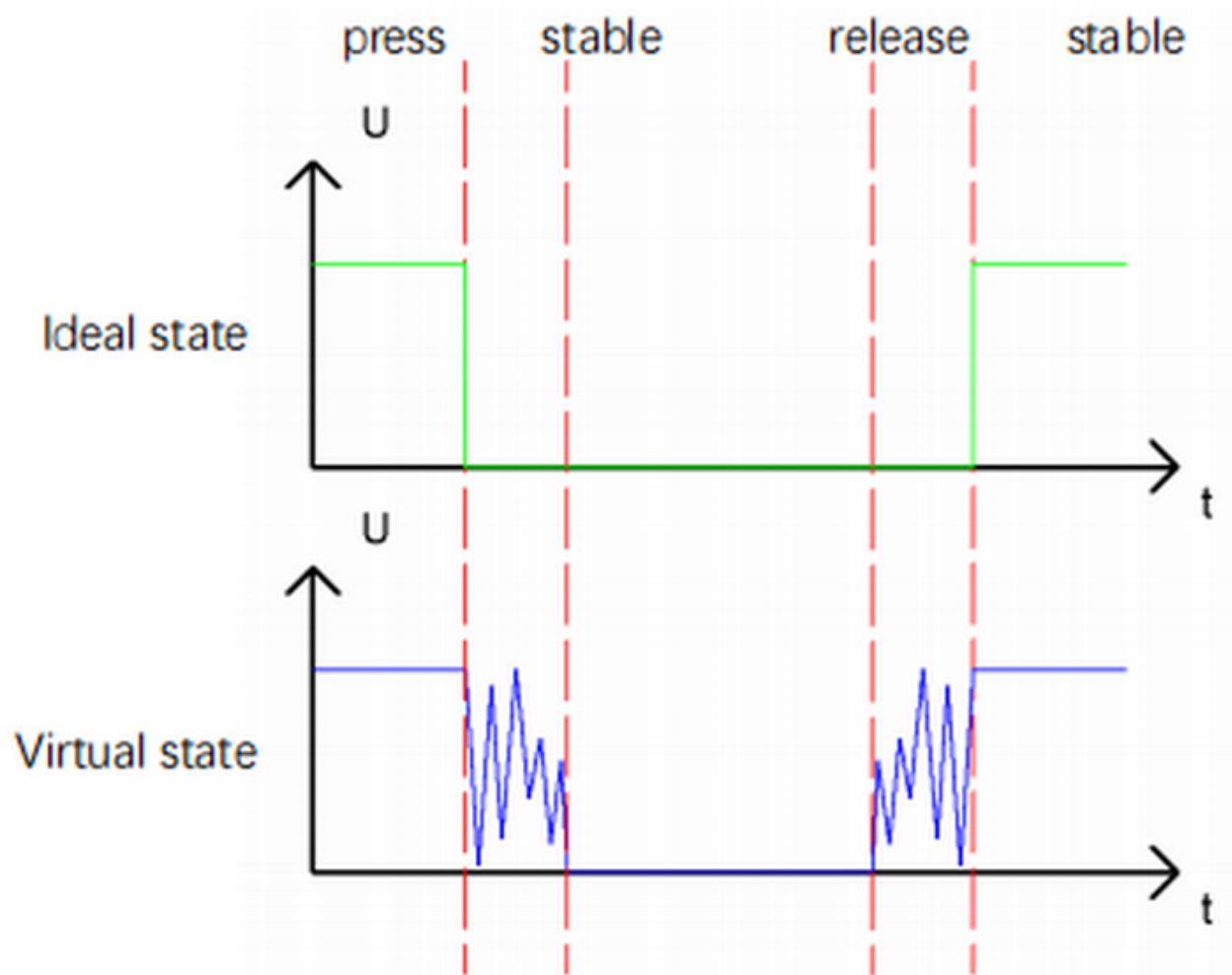
### (3)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Push Button Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	S	SIO18
V	5V	V	5V
G	G	G	G



#### (4) Eliminate Button Shaking

The LED status won't jump into new state immediately when button is pressed. There will be a short continuous shaking before into new status, which is similar with release status.



Therefore, there will be many pressing and releasing actions. The shaking will misleads the high speed movement of MCU, causing wrong judgement. That requires us to judge the button' status frequently. And only when its status is

stable can we be sure that the button is pressed.

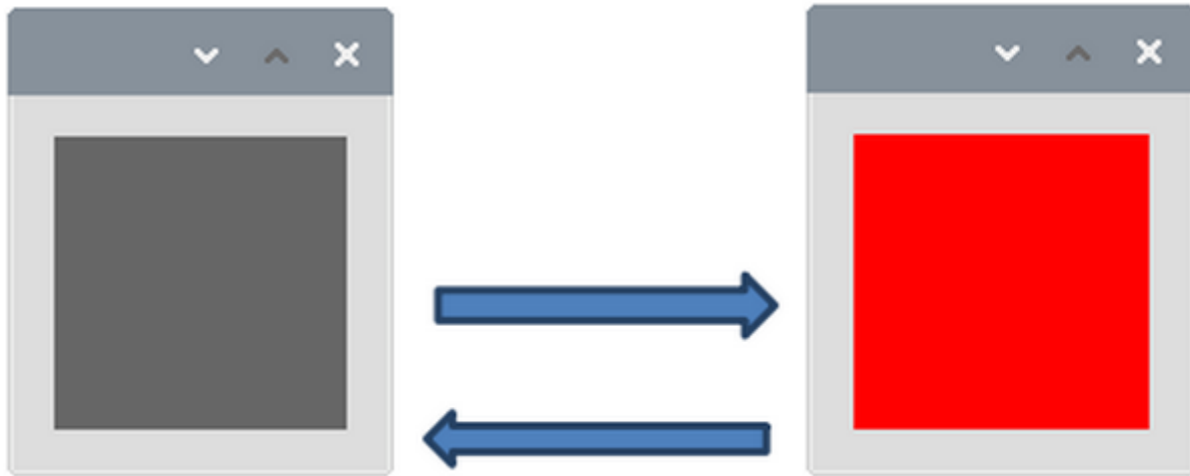
#### (5)Run Test Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

processing /home/pi/sketchbook/Processing\_Code/sketch\_7\_button\_led/sketch\_7\_button\_led.pde

#### (6)Test Results

After running example code, the display window is in dark gray. Press button, LED is on and window turns into red color; Press button again, LED is off and its background color is dark gray color, as shown below:



#### (7)Example Code

```
import processing.io.*;

int ledPin = 16; //define ledPin
int btnPin = 18; //define btnPin
int count = 0;
int flag = 0;
int ledState = 0;

void setup() {
  size(100, 100);
  GPIO.pinMode(btnPin, GPIO.INPUT_PULLUP);
  GPIO.pinMode(ledPin, GPIO.OUTPUT);
}

void draw() {

  if (GPIO.digitalRead(btnPin) == GPIO.LOW) { // button is pressed
    delay(10);
    flag = 1;
    if(flag == 1)
    {
      delay(10);
      if (GPIO.digitalRead(btnPin) == GPIO.HIGH)
      {
        count = count + 1;
      }
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

        println(count);
        flag = 0;
    }
}
}
ledState = count % 2;
if(ledState == 1)
{
    GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
    background(255, 0, 0);
}
else
{
    GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
    background(102);
}
}

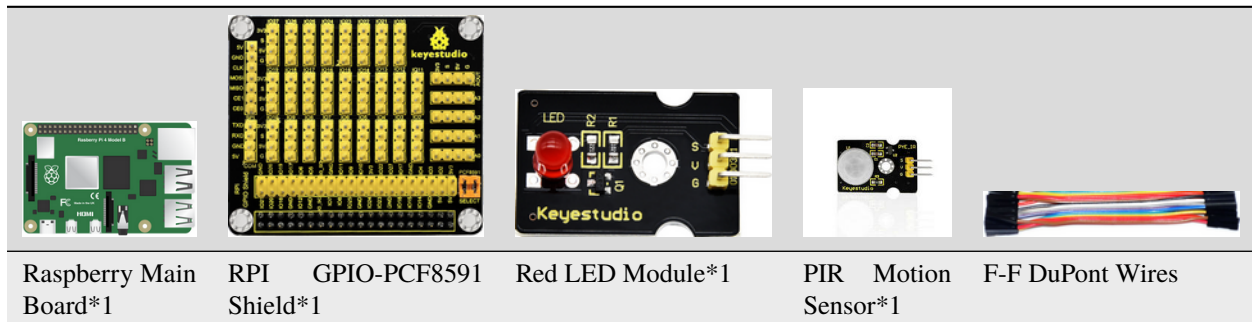
```

## 5.2.8 Project 8PIR Motion Sensor

### (1)Description

Lamps only light up when people pass by installed in some places, which are conducive to energy and cost saving. Have you ever thought about the principle behind these lamps? It is because of PIR motion sensors. In this lesson, we will learn about PIR motion sensor.

### (2)Components Needed



### (3)Knowledge about Component

#### PIR Motion Sensor

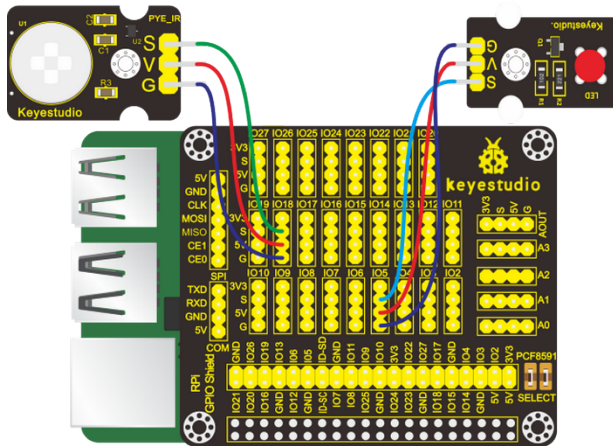
The principle of human infrared sensor is that when certain crystals, such as lithium tantalate and triglyceride sulfate, are heated, the two ends of the crystal will generate an equal number of charges, with opposite signs, which can be converted into voltage output by an amplifier.

Human body will emit IR ray, although weak but can be detected. This sensor outputs 1 (high level ) when human being is detected; otherwise, it outputs 0 (low level).

Note: Nothing but moving person can be detected, with the detection distance up to 3m.

### (4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	PIR Motion Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



### (5)Run Example Code

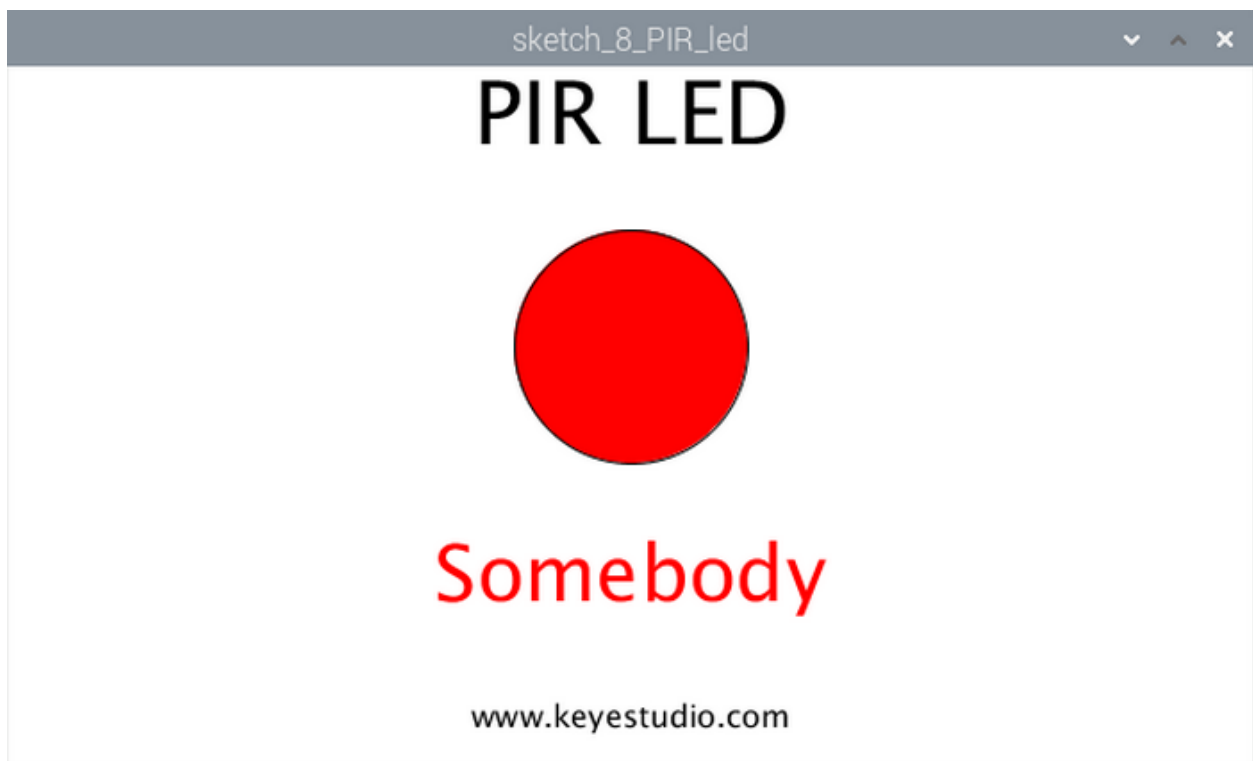
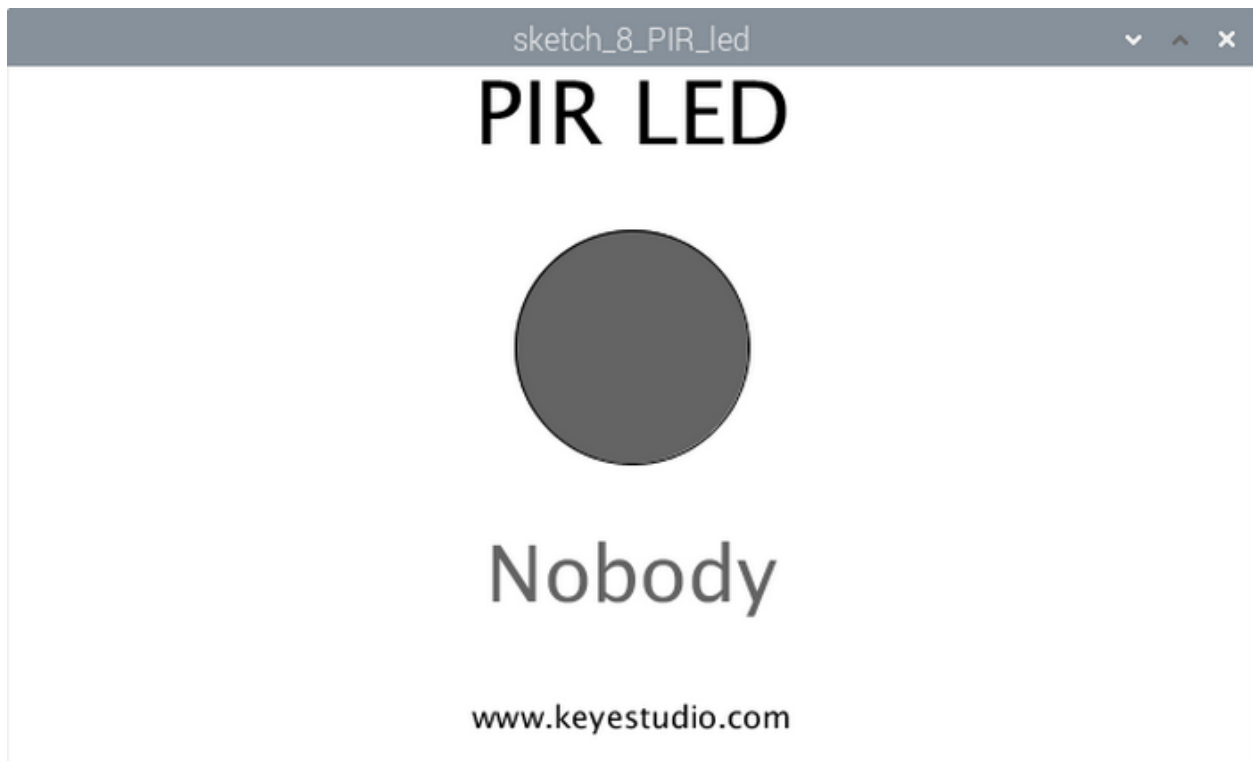
Input the following command, press “Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_8_PIR_led/sketch_8_PIR_led.pde
```

### (6)Test Results

If PIR motion sensor doesn’t detect moving person, LED will be off and display window will show black dot and“Nobody”; on the contrary, LED will be on, and window will show red dot and “Somebody”, as shown below:





#### (7)Example Code

```
import processing.io.*;
```

(continues on next page)

(continued from previous page)

```
final int sensorPin = 18; //connect to sensor pin
final int ledPin = 5;    //connect to led pin
void setup() {
  size(640,360);    //window size
  GPIO.pinMode(sensorPin, GPIO.INPUT);
  GPIO.pinMode(ledPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo();
  //if read sensor for high level
  if (GPIO.digitalRead(sensorPin) == GPIO.HIGH) {
    GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
    fill(255,0,0);    //fill in red
    textAlign(CENTER);    //set the text centered
    textSize(40);    //set text size
    text("Somebody", width / 2, 275);    //title
  } else {
    GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
    fill(100);    //fill in white
    textAlign(CENTER);    //set the text centered
    textSize(40);    //set text size
    text("Nobody", width / 2, 275);    //title
  }
  ellipse(width/2,height/2.5,height/3,height/3);
}

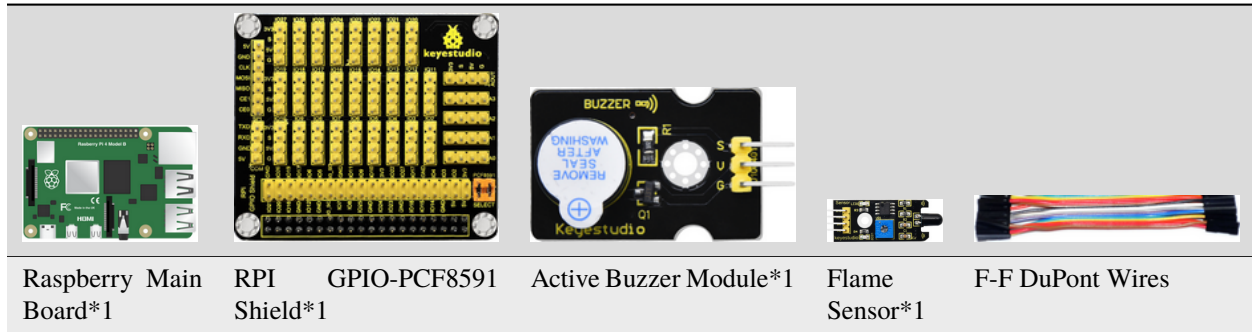
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(45);    //set text size
  text("PIR LED", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}
```

## 5.2.9 Project 9Fire Alarm

### (1)Description

A flame detector is a sensor designed to detect and respond to the presence of flames or fire, allowing flame detection.

### (2)Components Needed



### (3) Knowledge about Component

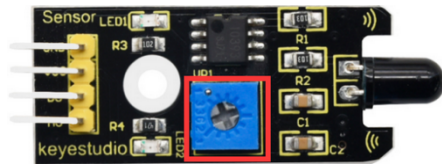
#### Flame Sensor

Flame sensor is made based on the principle that infrared ray is highly sensitive to flame. It has an infrared receiving tube specially designed to detect fire, and then convert the flame brightness to fluctuating level signal. The signals are then input into the central processor and be dealt with accordingly.

Flame sensor is used to detect fire source with wavelength in 760nm-1100nm, detection angle is 60°. When its IR waves length is close to 940nm, and its sensitivity is the highest.

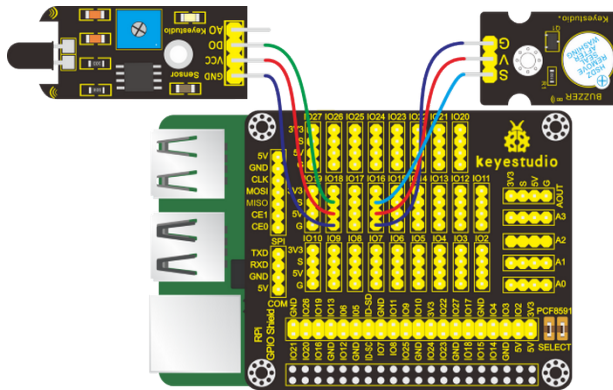
Notice that keep flame sensor away from fire source to defend its damage for its working temperature is between -25°-85°

Note: a potentiometer is built in the sensor so its sensitivity can be adjusted by rotating it.



### (4) Connection Diagram

Active Buzzer Module	RPI GPIO-PCF8591 Shield	Flame Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	D0	SIO18
V	5V	VCC	5V
G	G	GND	G



### (5)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_9_flame_buzzer/sketch_9_flame_buzzer.pde
```

### (6)Test Results

Active buzzer will emit sound and window display will show arc lines and“Fire”as follows, when flame is detected; otherwise, active buzzer won’t emit sound, arc lines and“Fire”will disappear.





#### (7)Example Code

```
import processing.io.*;

int flamePin = 18; //connect to flame pin
int buzzerPin = 16; //connect to buzzer pin
boolean buzzerState = false;
void setup() {
  size(640,360); //window size
  GPIO.pinMode(flamePin, GPIO.INPUT);
  GPIO.pinMode(buzzerPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo(); //title and site infomation
  drawBuzzer(); //buzzer img
  //if read sensor for high level
  if (GPIO.digitalRead(flamePin) == GPIO.LOW) {
    GPIO.digitalWrite(buzzerPin, GPIO.HIGH); //buzzer on
    drawArc(); //Sounds waves img
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("Fire", width / 3, 250); //title
  } else {
    GPIO.digitalWrite(buzzerPin, GPIO.LOW); //buzzer off
  }
}
```

(continues on next page)

(continued from previous page)

```

void drawBuzzer() {
  strokeWeight(1);
  fill(0);
  ellipse(width/2, height/2, 50, 50);
  fill(255);
  ellipse(width/2, height/2, 10, 10);
}
void drawArc() {
  noFill();
  strokeWeight(8);
  for (int i=0; i<3; i++) {
    arc(width/2, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
  }
}
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);         //set text size
  text("Fire Alarm", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}

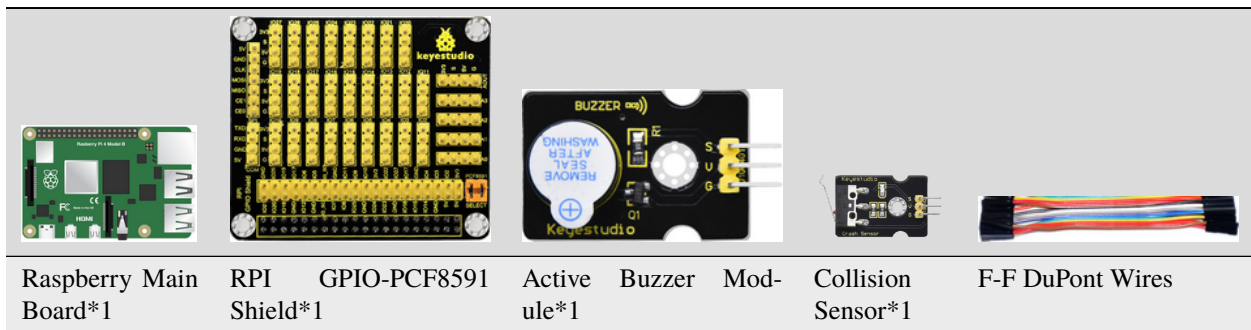
```

## 5.2.10 Project 10 Collision Alarm

### (1)Description

We can use the collision sensor to detect whether crash happens. When the metal plate above the push button switch of the sensor is knocked, it outputs low level signals; and when the button is open, it remind in high level. In this project, collision sensor will be applied to control the active buzzer.

### (2)Components Needed



### (3)Knowledge about Component

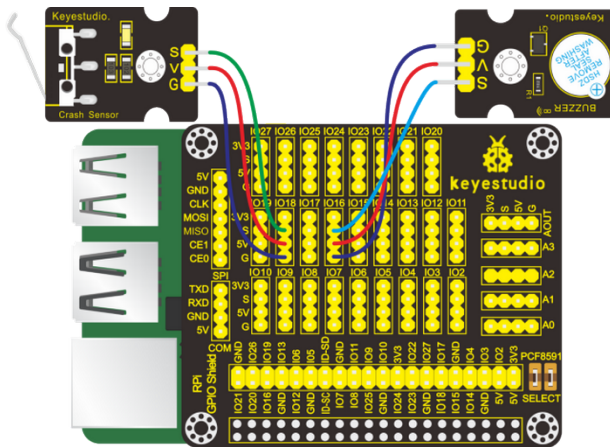
#### Collision Sensor:

It is a widely used collision sensor that has a push button switch covered by a mental plate. When the plate is pushed, the button is pressed, the sensor outputs low level and the LED on it lights; or it outputs high level and the LED reminds off.

This sensor is often used as a limit switch in a 3D printer.

#### (4)Connection Diagram

Active Buzzer Module	RPI GPIO-PCF8591 Shield	Collision Senso	RPI GPIO-PCF8591 Shield
S	SIO16	S	SIO18
V	5V	V	5V
G	G	G	G



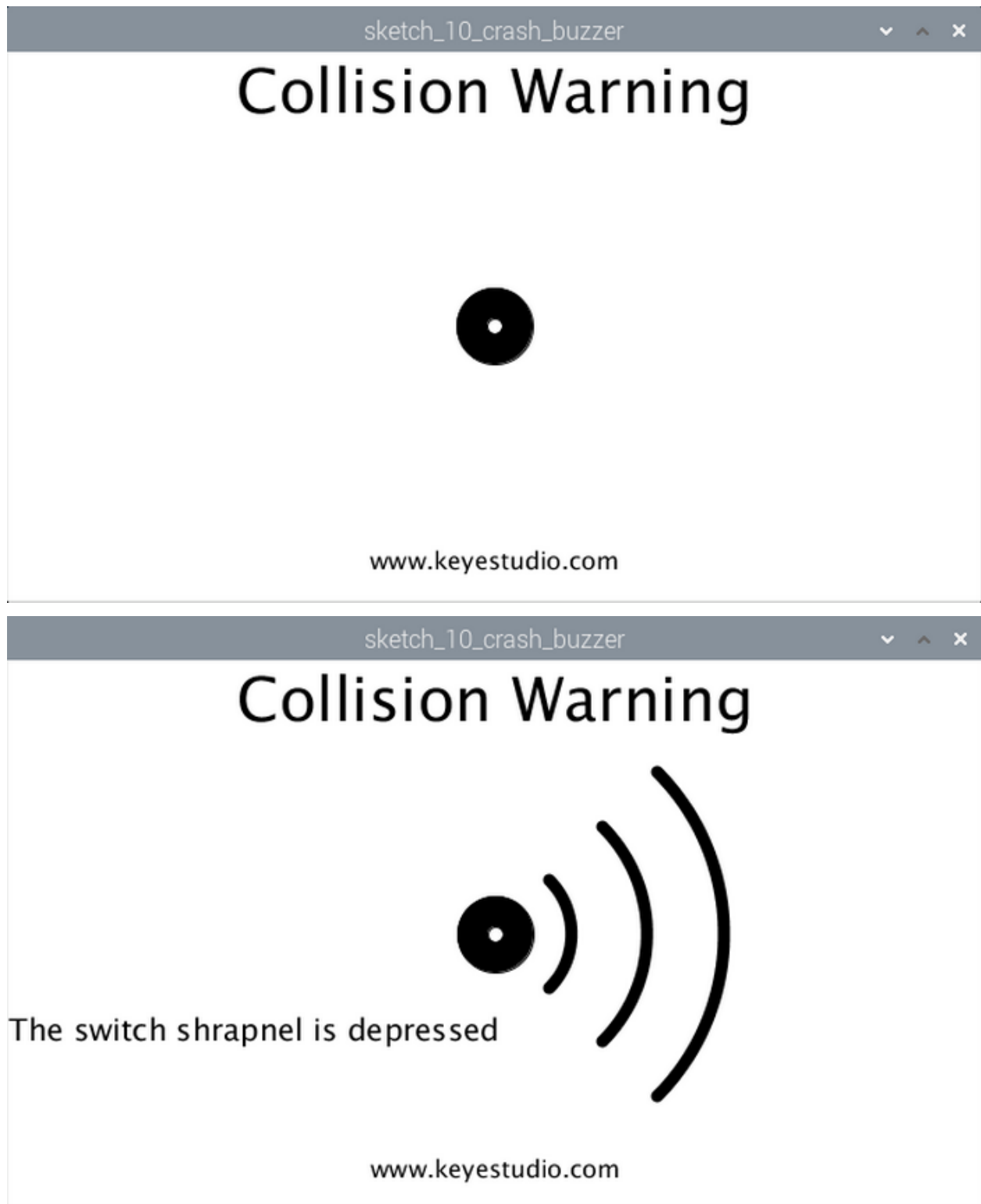
#### (5)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_10_crash_buzzer/sketch_10_crash_buzzer.pde
```

#### (6)Test Results

After running the program, when the metal plate of the push button switch is pressed, the buzzer makes sound and the display window shows the arc-sharped pattern and sentence““The switch shrapnel is depressed””; it keeps silent and the arc-sharped pattern and sentence““The switch shrapnel is depressed”disappear as shown below.



#### (7)Example Code

```
import processing.io.*;
```

(continues on next page)



(continued from previous page)

```

int crashPin = 18; //connect to crash pin
int buzzerPin = 16; //connect to buzzer pin
boolean buzzerState = false;
void setup() {
  size(640,360); //window size
  GPIO.pinMode(crashPin, GPIO.INPUT);
  GPIO.pinMode(buzzerPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo(); //title and site infomation
  drawBuzzer(); //buzzer img
  //if read sensor for high level
  if (GPIO.digitalRead(crashPin) == GPIO.LOW) {
    GPIO.digitalWrite(buzzerPin, GPIO.HIGH); //buzzer on
    drawArc(); //Sounds waves img
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(20); //set text size
    text("The switch shrapnel is depressed", width / 4, 250); //title
  } else {
    GPIO.digitalWrite(buzzerPin, GPIO.LOW); //buzzer off
  }
}

void drawBuzzer() {
  strokeWeight(1);
  fill(0);
  ellipse(width/2, height/2, 50, 50);
  fill(255);
  ellipse(width/2, height/2, 10, 10);
}

void drawArc() {
  noFill();
  strokeWeight(8);
  for (int i=0; i<3; i++) {
    arc(width/2, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
  }
}

void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER); //set the text centered
  textSize(40); //set text size
  text("Collision Warning", width / 2, 40); //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20); //site
}

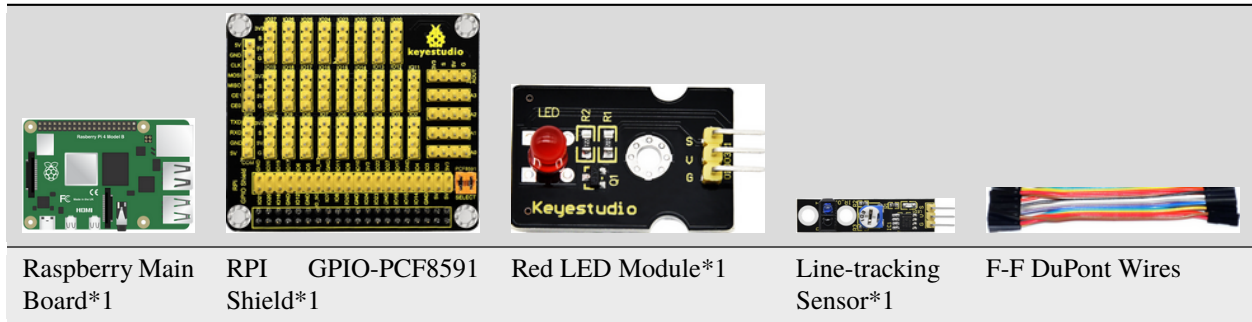
```

## 5.2.11 Project 11 Line-tracking Sensor

### (1)Description

You may have seen that in an experiment a smart car moved along a black line and it didn't overstep this boundary. How did it make it? The credit goes to a line-tracking sensor. And in this project, we intend to learn about the line-tracking sensor.

### (2)Components Needed



### (3)Knowledge about Component

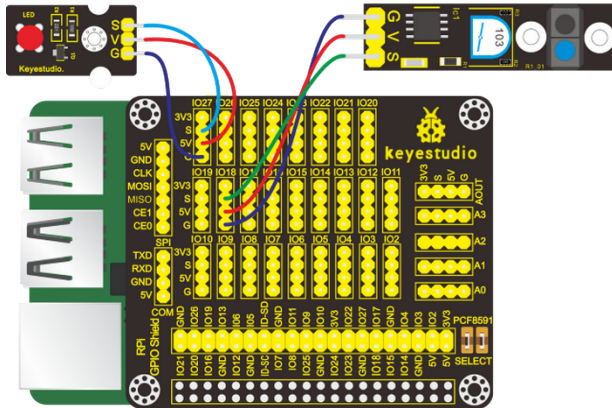
#### Line-tracking Sensor

It is an infrared sensor in nature which can detect white and black objects. The working principle of the TCRT5000 pair tube on the sensor is based on the different reflectivity of infrared to colors so as to convert this different strengths of reflected signals to electric signals. When the sensor detects black objects, it is in high level while when it sensors white items it is in low level. And the detection altitude is from 0 to 3cm. You can rotate the potentiometer in a bid to adjust the sensitivity of the line-tracking sensor.



### (4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Line-tracking Sensor	RPI GPIO-PCF8591 Shield
S	SIO27	S	SIO18
V	5V	V	5V
G	G	G	G



### (5)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDEprocessing /home/pi/sketchbook/Processing\_Code/sketch\_11\_tracking/sketch\_11\_tracking.pde

### (6)Test Results

After running the program, when the line-tracking sensor detects blacks obstacles or senses nothing,the LED reminds off the pattern in the display representing the LED is in black and the phrase “Black object”appears; while when it detects white items, the LED is on, the pattern on the window gets red and the phrase “White object” displays as shown below:



**(7)Example Code**

```
import processing.io.*;

final int trackingPin = 18; //connect to tracking pin
final int ledPin = 27;    //connect to led pin
void setup() {
  size(640,360);    //window size
  GPIO.pinMode(trackingPin, GPIO.INPUT);
  GPIO.pinMode(ledPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo();
  //if read sensor for high level
  if (GPIO.digitalRead(trackingPin) == GPIO.LOW) {
    GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
    fill(255,0,0);    //fill in red
    textAlign(CENTER);    //set the text centered
    textSize(40);    //set text size
    text("White object", width / 2, 275);    //title
  } else {
    GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
    fill(100);    //fill in white
    textAlign(CENTER);    //set the text centered
    textSize(40);    //set text size
    text("Black object", width / 2, 275);    //title
  }
}
```

(continues on next page)

(continued from previous page)

```

    ellipse(width/2,height/2.5,height/3,height/3);
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);    //set the text centered
    textSize(45);         //set text size
    text("Tracking", width / 2, 40);    //title
    textSize(16);
    text("www.keyestudio.com", width / 2, height - 20);    //site
}

```

## 5.2.12 Project 12 Magnetic Detection

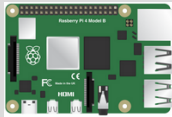
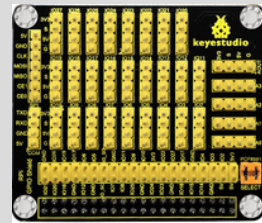



### (1)Description

What is the best way to detect a magnet? Use another magnet? Yeah , it can but it is not sensitive enough. You still need to feel it by yourselves.

Perhaps you can try a hall magnetic sensor which features high sensitivity, quick response, nice temperature performance, and high reliability.

In this project, we will try to turn a LED on and off through a hall magnetic sensor.

### (2)Components Needed

				
Raspberry Main Board*1	RPI    GPIO-PCF8591 Shield*1	Red LED Module*1	Hall    Magnetic Sensor*1	F-F DuPont Wires

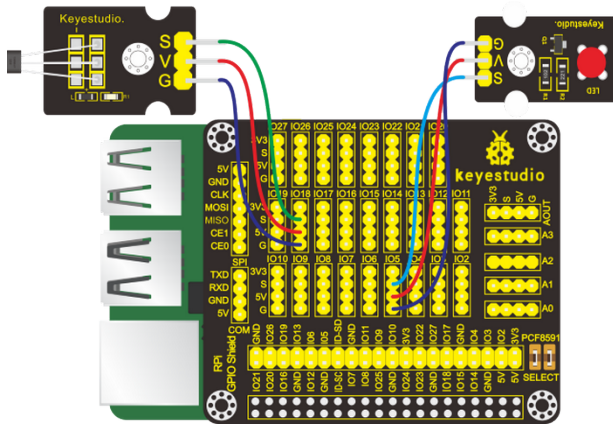
### (3)Knowledge about Component:

#### Hall Magnetic Sensor

The main component built in the sensor is A3144E, which is an electronic magnetic device and an active one. It uses magnetic field and Hall effects to achieve the purpose of non-contact control. Since the Hall element itself is a chip in nature, its working life is theoretically unlimited. The sensor can be used to detect magnetic fields and output digital signals. It can sense magnetic materials within a detection range of about 3cm. Note that it can only detect the presence of a magnetic field nearby, but not the strength of the magnetic field.

### (4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Hall Magnetic Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



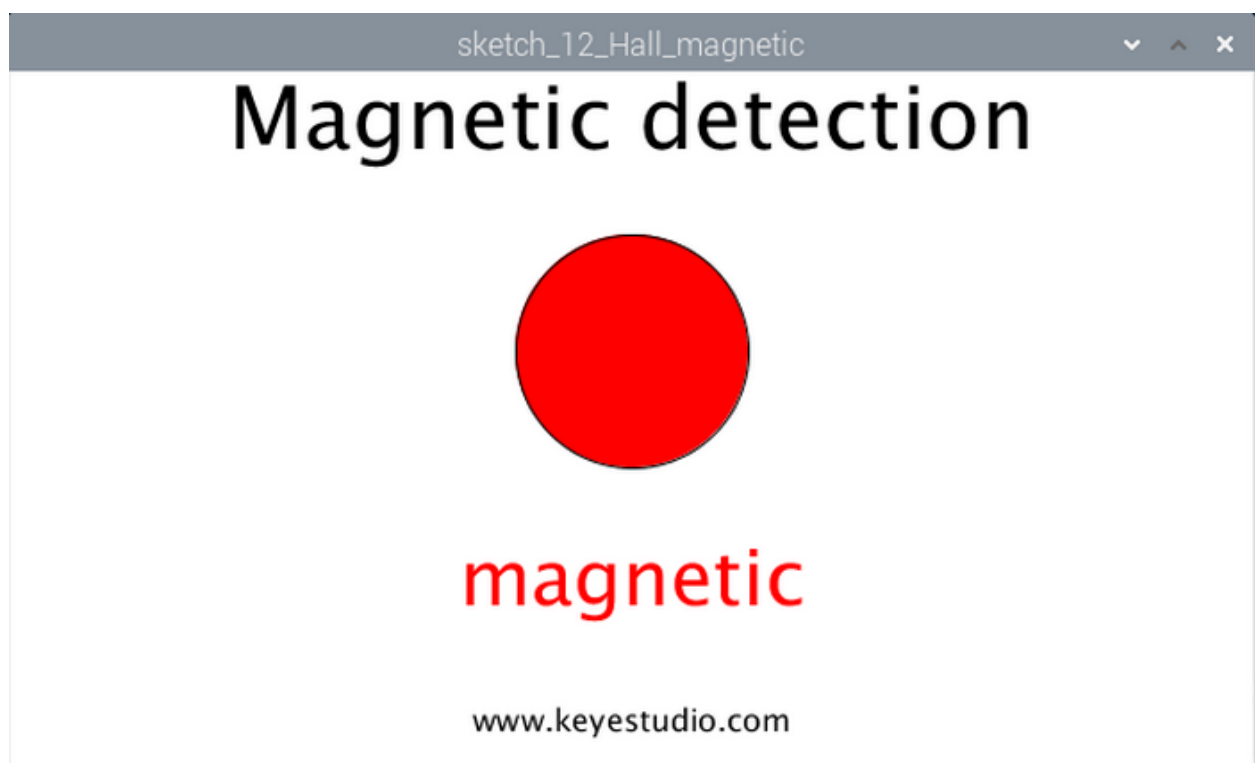
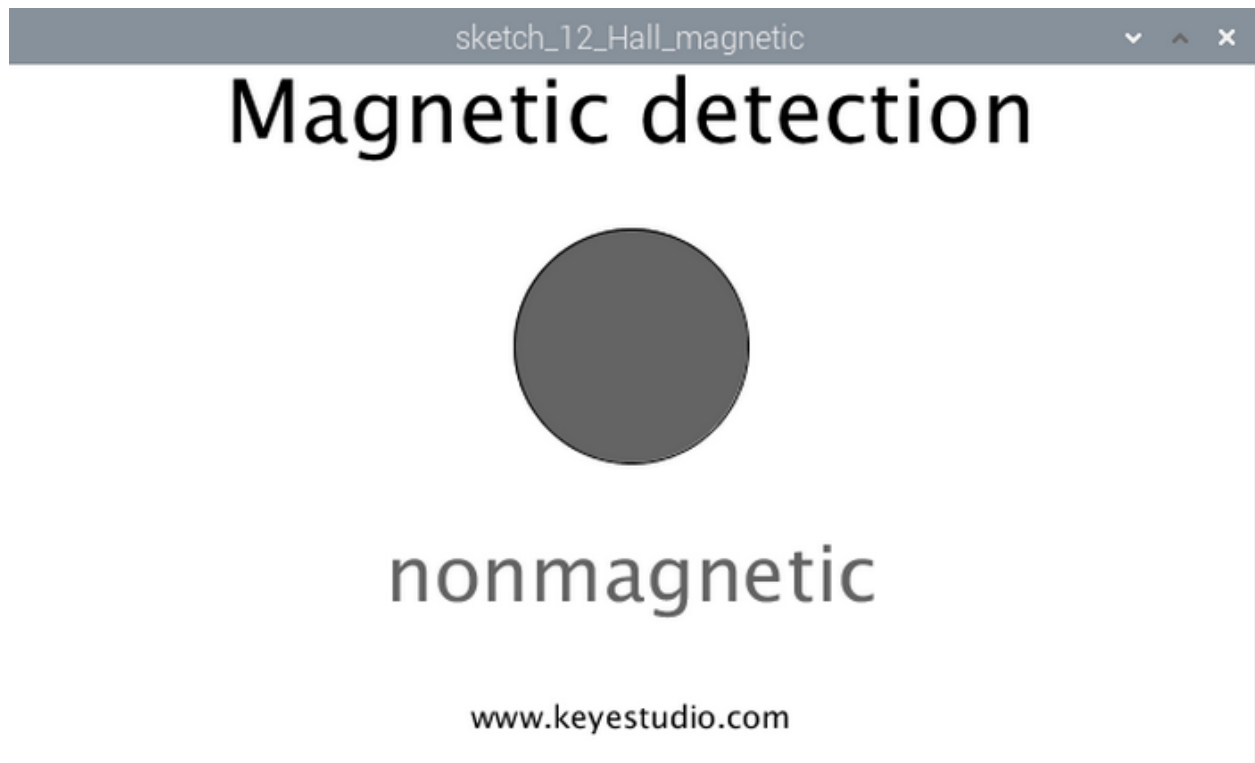
### (5)Run Example Code

Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_12_Hall_magnetic/sketch_12_Hall_magnetic.pde
```

### (6)Test Results

After running the program and placing a magnetic ball around the Hall magnetic sensor, when the sensor detects magnetic field nearby, the window shows “magnetic” and the LED lights up; otherwise, it displays“nonmagnetic”and the LED stays dark.



#### (7)Example Code

```
import processing.io.*;
```

(continues on next page)

```
final int hallPin = 18; //connect to hall pin
final int ledPin = 5;   //connect to led pin
void setup() {
  size(640,360);        //window size
  GPIO.pinMode(hallPin, GPIO.INPUT);
  GPIO.pinMode(ledPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo();
  //if read sensor for high level
  if (GPIO.digitalRead(hallPin) == GPIO.LOW) {
    GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
    fill(255,0,0); //fill in red
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("magnetic", width / 2, 275); //title
  } else {
    GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
    fill(100); //fill in white
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("nonmagnetic", width / 2, 275); //title
  }
  ellipse(width/2,height/2.5,height/3,height/3);
}

void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER); //set the text centered
  textSize(45); //set text size
  text("Tracking", width / 2, 40); //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20); //site
}
```

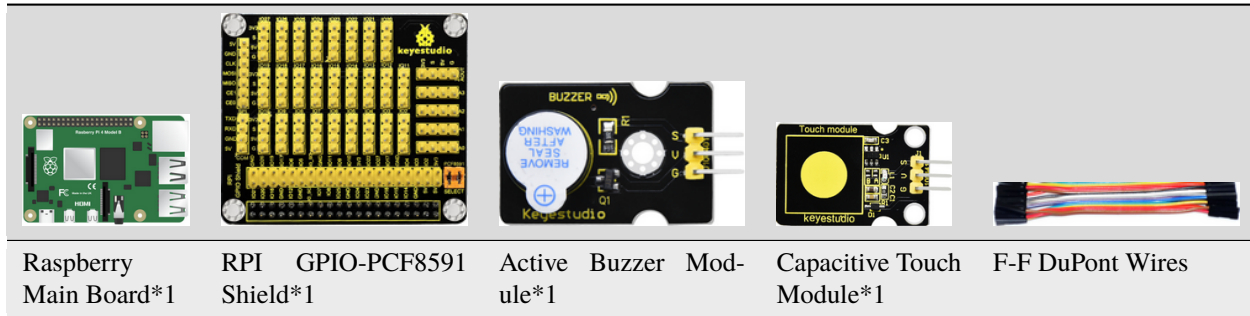
## Project 13 Touch-sensitive Alarm

### (1)Description

Touch-sensitive alarm is very commonplace in daily life, especially found in home anti-theft and car anti-theft systems. When someone touches the alarming mental material, the device alarms to warn people. And it is of high sensitivity and high reliability evidenced by issuing alarm the moment it is touched.

### (2)Components Needed





### (3)Knowledge about Component:

#### Capacitive Touch Module

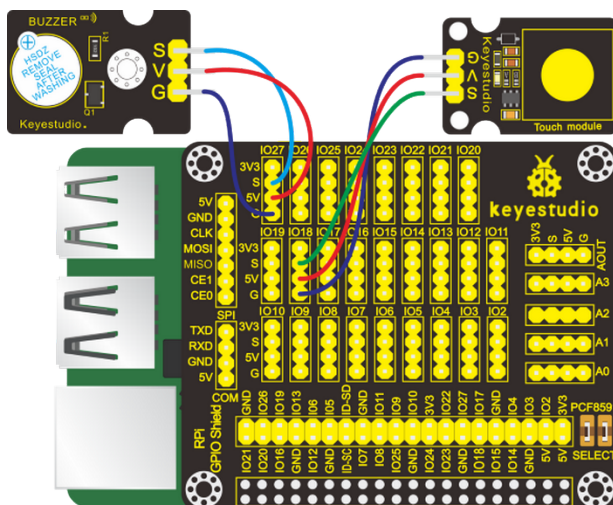
It mainly uses touch detection IC and can be found in many electronic devices. It uses the most popular capacitive sensing technology, just like the smart buttons on your phone. The touching area of this small sensor can feel the touch of humans and metals by responding with high or low level. It can still detect the touch though covered by a piece of paper and cloth. The sensitivity reduces with the increase of items between the touch-sensitive area and the object performing the touch.

The touch detection IC is designed to replace the traditional button with a variable area key, featuring low power consumption and wide operating voltage.

When the module is powered up, it needs a stabilization time of about 0.5 sec. During this time period, do not touch the keypad. At this time, all functions are disabled, and self-calibration is always performed. No touching the key, the recalibration period is about 4.0sec.

#### (4)Connection Diagram

Active Buzzer Mod- ule	RPI Shield	GPIO-PCF8591	Capacitive Touch Sen- sor	RPI Shield	GPIO-PCF8591
S	SIO27		S	SIO18	
V	5V		V	5V	
G	G		G	G	



### (5)Run Example Code

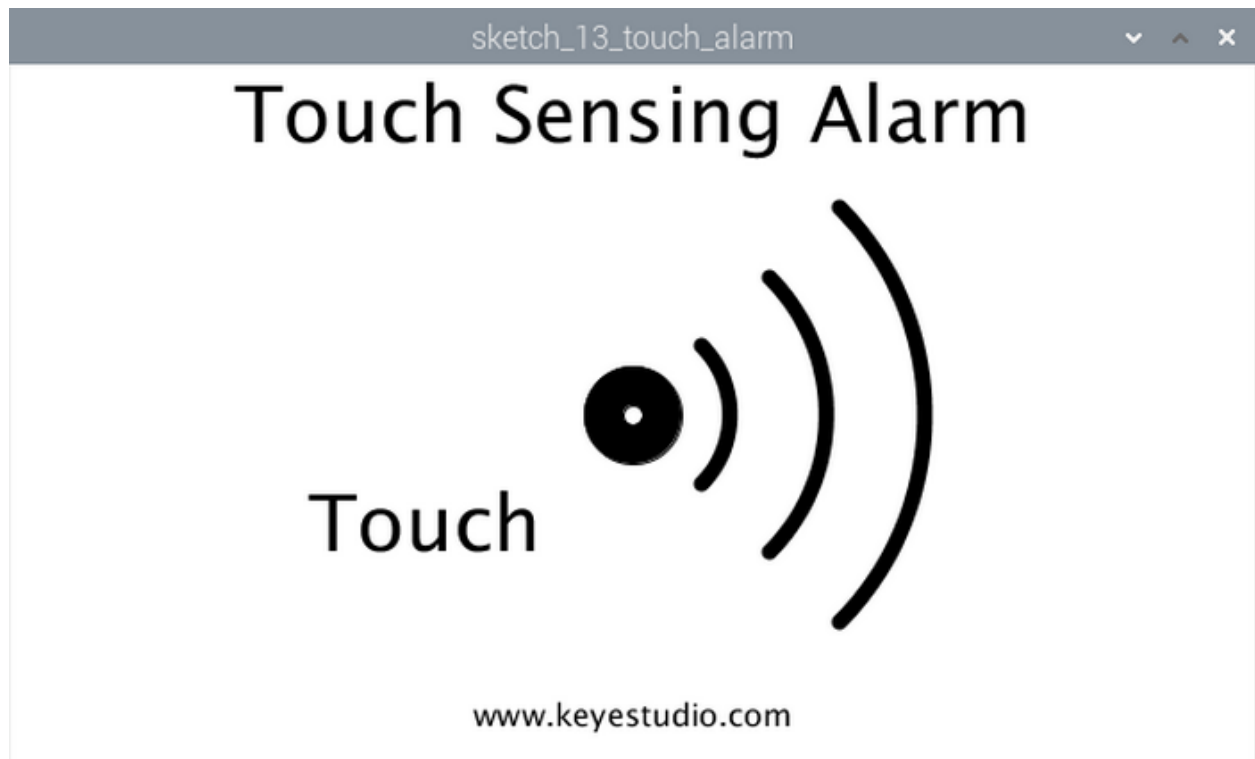
Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_13_touch_alarm/sketch_13_touch_alarm.pde
```

### (6)Test Results

After running the program, when the sensing area on the capacitive touch sensor is touched, the window shows arc-shaped pattern and “Touch” and the buzzer makes sounds; otherwise, shows no arc-shaped pattern and “Touch” and the buzzer is in silence.





#### (7)Example Code

```
import processing.io.*;

int touchPin = 18; //connect to touch pin
int buzzerPin = 27; //connect to buzzer pin
boolean buzzerState = false;
void setup() {
  size(640,360); //window size
  GPIO.pinMode(touchPin, GPIO.INPUT);
  GPIO.pinMode(buzzerPin, GPIO.OUTPUT);
}

void draw() {
  background(255);
  titleAndSiteInfo(); //title and site infomation
  drawBuzzer(); //buzzer img
  //if read sensor for high level
  if (GPIO.digitalRead(touchPin) == GPIO.HIGH) {
    GPIO.digitalWrite(buzzerPin, GPIO.HIGH); //buzzer on
    drawArc(); //Sounds waves img
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("Touch", width / 3, 250); //title
  } else {
    GPIO.digitalWrite(buzzerPin, GPIO.LOW); //buzzer off
  }
}
```

(continues on next page)

(continued from previous page)

```

void drawBuzzer() {
  strokeWeight(1);
  fill(0);
  ellipse(width/2, height/2, 50, 50);
  fill(255);
  ellipse(width/2, height/2, 10, 10);
}
void drawArc() {
  noFill();
  strokeWeight(8);
  for (int i=0; i<3; i++) {
    arc(width/2, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
  }
}
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);         //set text size
  text("Touch Sensing Alarm", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}

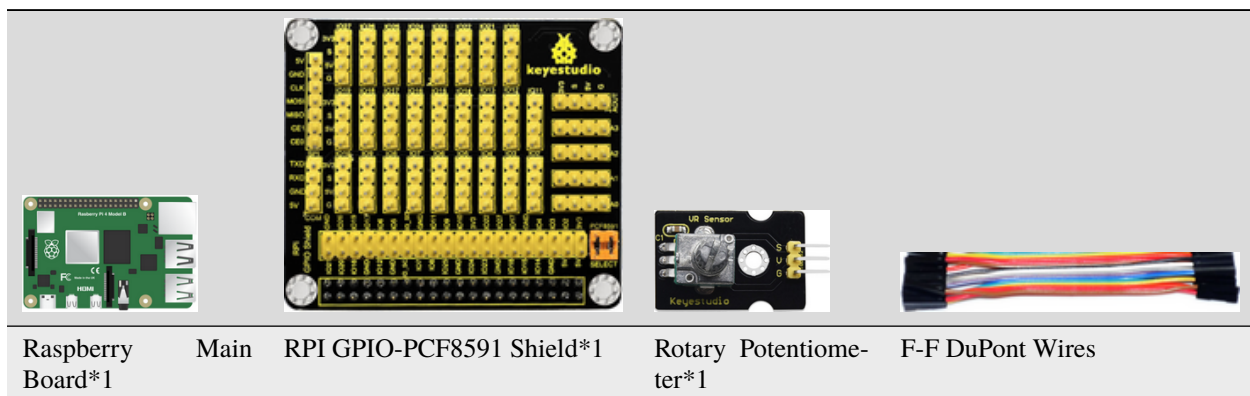
```

## 5.2.13 Project 14 Rotary Potentiometer

### (1)Description

In this project, we intend to use the PCF8591 A/D converter chip behind RPI GPIO-PCF8591 shield to read the voltage value of the potentiometer and make the display window show it.

### (2)Components Needed



### (3)Knowledge about Components

#### PCF8591 A/D converter chip:

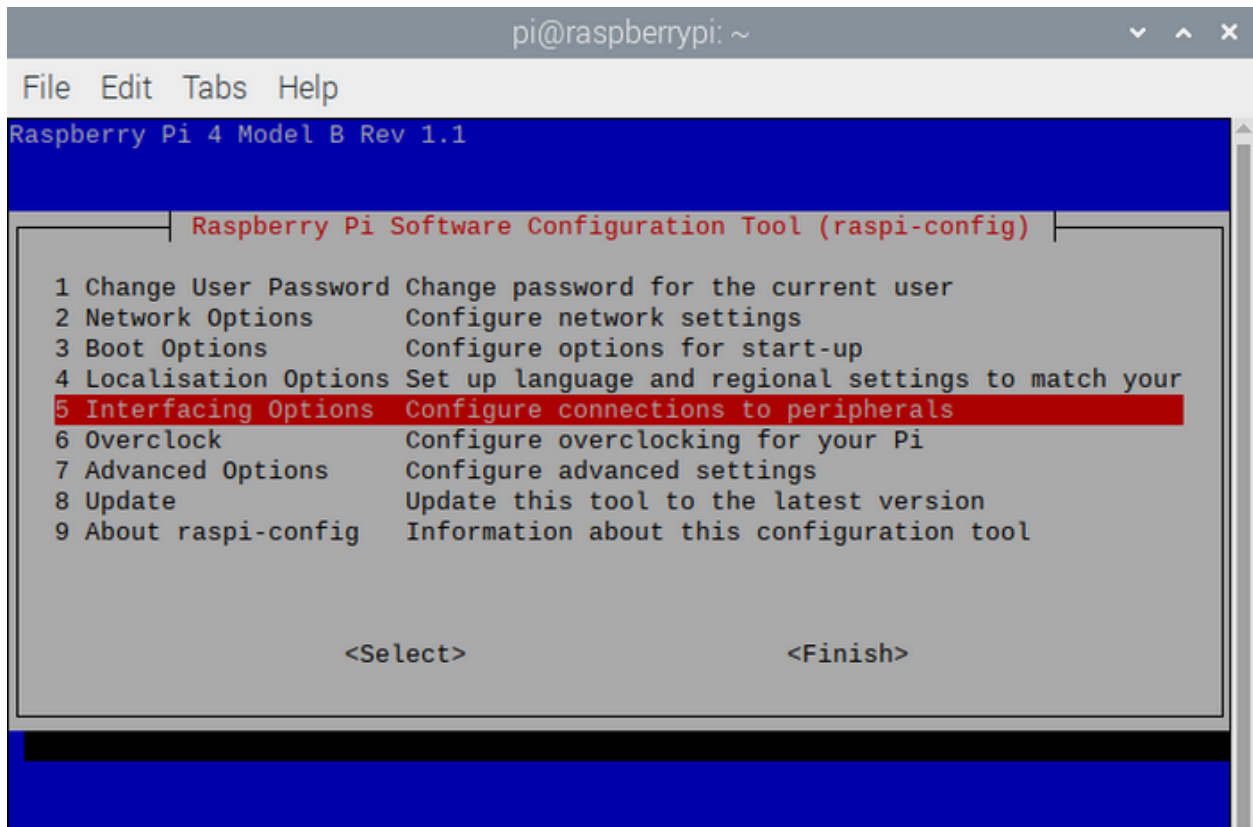
It is installed behind the RPI GPIO-PCF8591 shield with voltage resolution of 5V/255 0.01961.

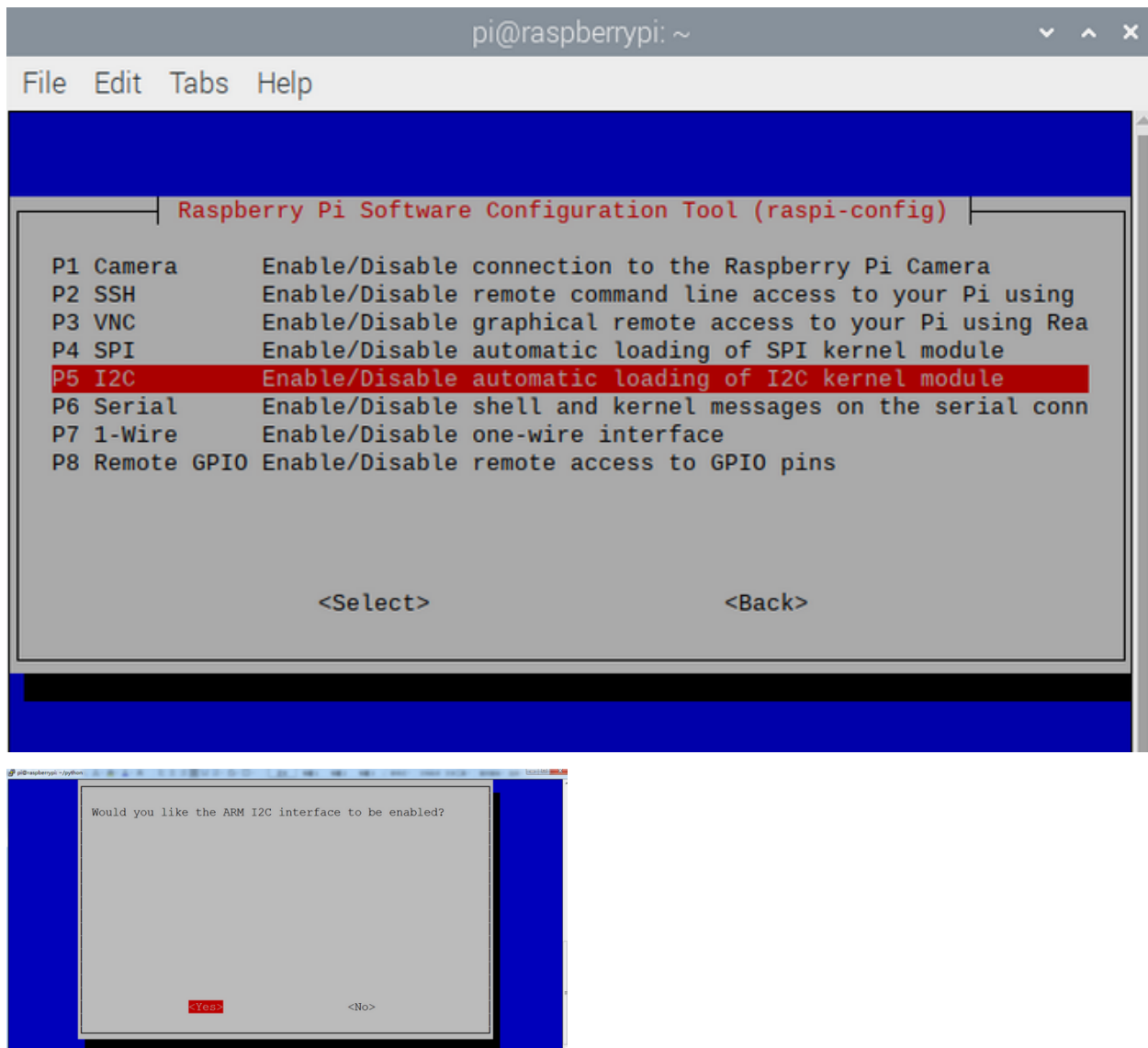
Since the Raspberry Pi itself does not have AD/DA function, an expansion board with this function is required when it is connected to external analog sensors. And here we use PCF8591 A/D converter with I2C communication.

Enable the I2C communication function of the Raspberry Pi as follows: Raspberry Pi does not enable the I2C function by default. Enter `sudo raspi-config` in the terminal to enter the Raspberry Pi configuration interface.

```
pi@raspberrypi:~/python $ sudo raspi-config
```

Enable the I2C function of Raspberry Pi as follows:





Find more about I2C:

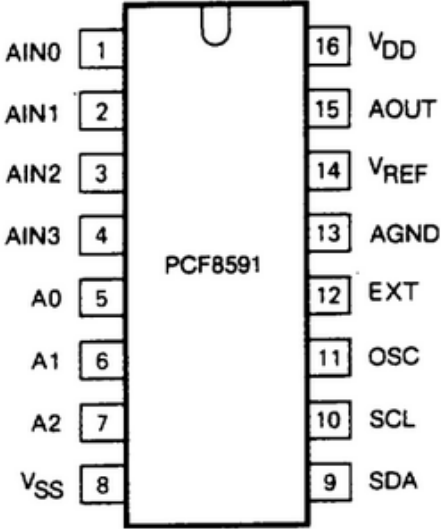
<https://www.nxp.com/docs/en/user-guide/UM10204.pdf>

**Pin description:**

You can find more information, such as the specification of this chip, in the resources link:

<https://fs.keyestudio.com/KS3016>

From the picture below, it is obvious that the PCF8591 converter is equipped with a AOUT pin and 4 analog inputs pins A0~A3

SYMBOL	PIN	DESCRIPTION	TOP VIEW
AIN0	1	Analog inputs (A/D converter)	
AIN1	2		
AIN2	3		
AIN3	4		
A0	5	Hardware address	
A1	6		
A2	7		
Vss	8	Negative supply voltage	
SDA	9	I2C-bus data input/output	
SCL	10	I2C-bus clock input	
OSC	11	Oscillator input/output	
EXT	12	external/internal switch for oscillator input	
AGND	13	Analog ground	
Vref	14	Voltage reference input	
AOUT	15	Analog output(D/A converter)	
Vdd	16	Positive supply voltage	

Check the address of the I2C module (PCF8591) connected to the Raspberry Pi, enter the command: `i2cdetect -y 1`, and then press Enter.

From below picture, it is known that the I2C address is 0x48 .

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~ $

```

The address for reading pins A0~A3 is:

A0 = 0x40 ##A0 —> port address

A1 = 0x41

A2 = 0x42

A3 = 0x43

The address for analog output pin AOUT is: 0x40, which is 64 when hexadecimal is converted to decimal.

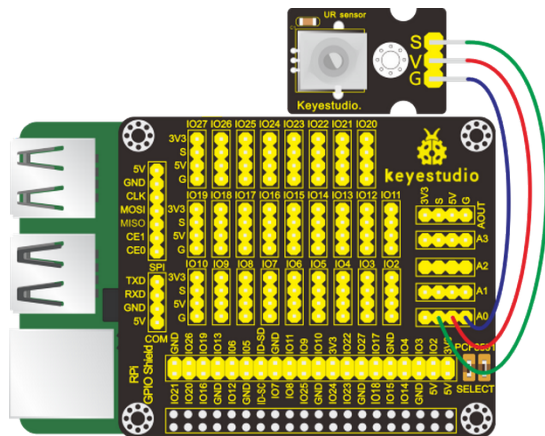
### Rotary Potentiometer

It can be viewed as an adjustable resistor with the range from 0~10K.

Therefore when we rotate the potentiometer, we actually change its resistance. We can build a circuit to convert the changes in the resistance to the changes in voltage. Then input the voltage changes to the GPIO analog input port for detection through the signal terminal of the module.

### (4)Connection Diagram

Rotary Potentiometer	RPI GPIO-PCF8591 Shield
S	SA0
V	5V
G	G



### (5)Run Example Code

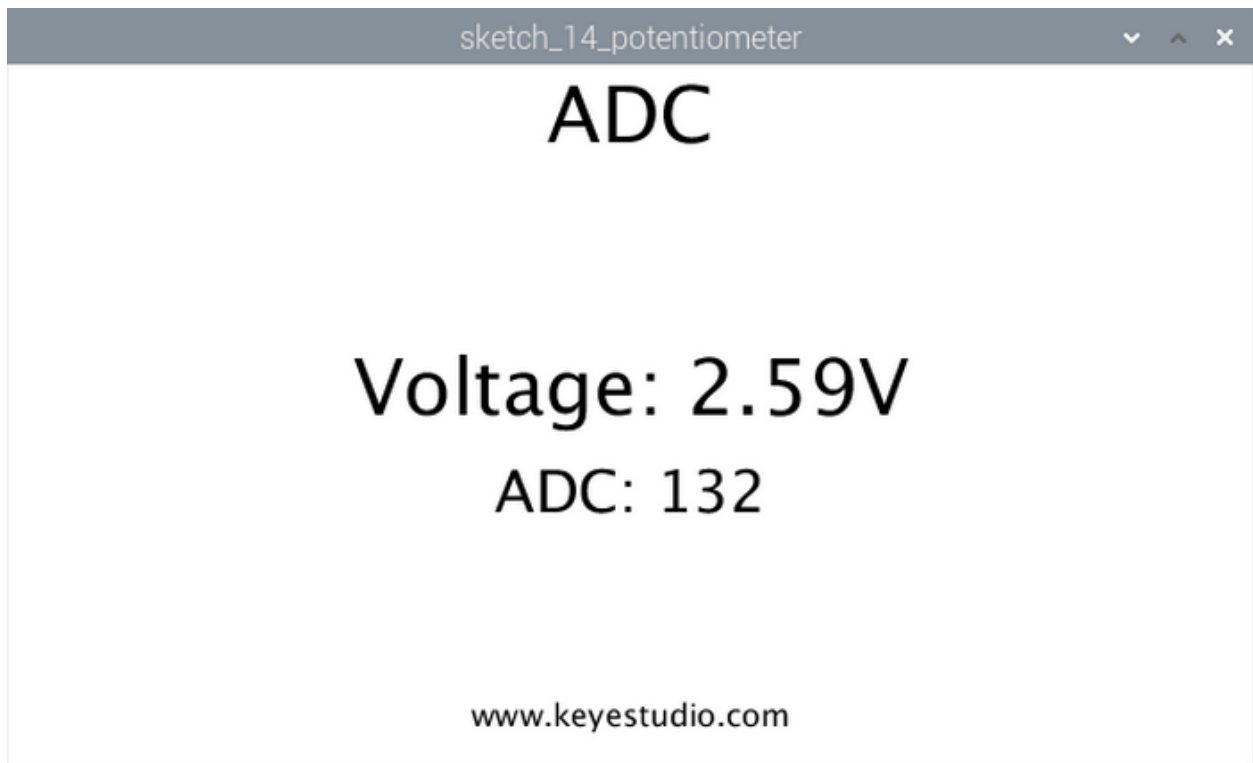
Input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_14_potentiometer/sketch_14_potentiometer.pde
```

### (6)Test Results

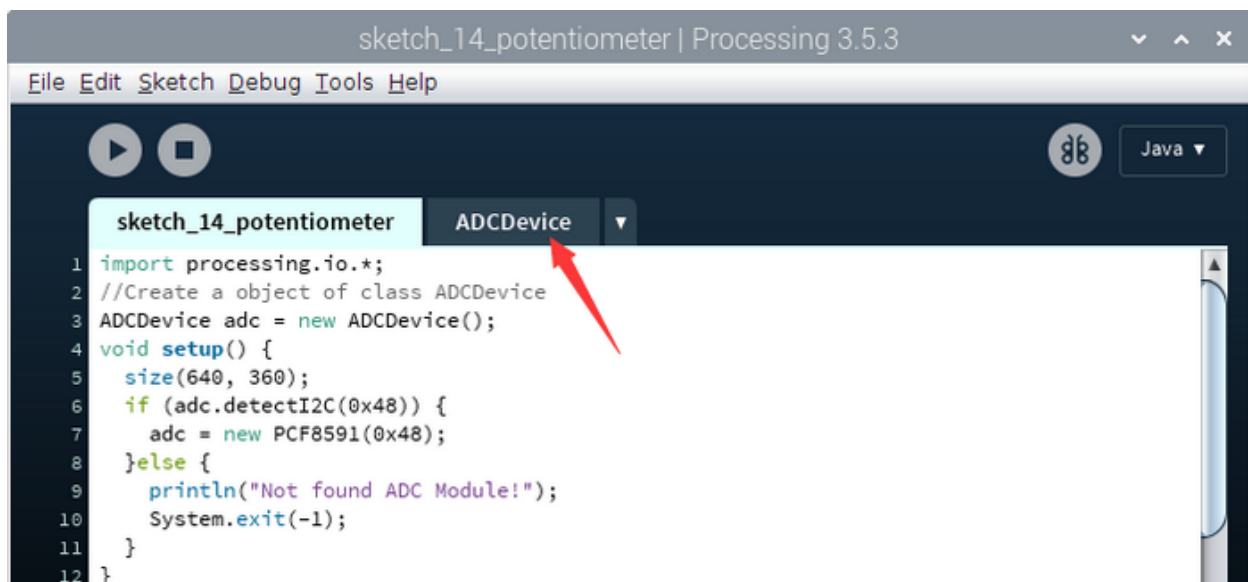
Window shows voltage value and ADC value. You could change the output voltage by rotating the potentiometer, as shown below





### (7)Example Code

This program includes many code files, the core code is included in sketch\_14\_potentiometer.pde file, others are customized, as shown below:



Code:

```

import processing.io.*;
//Create a object of class ADCDevice
ADCDevice adc = new ADCDevice();
void setup() {
  
```

(continues on next page)

(continued from previous page)

```

size(640, 360);
if (adc.detectI2C(0x48)) {
    adc = new PCF8591(0x48);
}else {
    println("Not found ADC Module!");
    System.exit(-1);
}
}
void draw() {
    int adcValue = adc.analogRead(0);    //Read the ADC value of channel 0
    float volt = adcValue*5.0/255.0;    //calculate the voltage
    background(255);
    titleAndSiteInfo();

    fill(0);
    textAlign(CENTER);    //set the text centered
    textSize(30);
    text("ADC: "+nf(adcValue, 3, 0), width / 2, height/2+50);
    textSize(40);    //set text size
    text("Voltage: "+nf(volt, 0, 2)+"V", width / 2, height/2);    //
}
void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);    //set the text centered
    textSize(40);    //set text size
    text("ADC", width / 2, 40);    //title
    textSize(16);
    text("www.keyestudio.com", width / 2, height - 20);    //site
}

```

## (8)Reference

**class ADCDevice**This is a base class which means all ADC module class is its subclass . And it provides two basic member functions.

public int analogRead\*\*(int chn)\*\*

This is a uniform function name. Different chips have different implementation methods. Therefore, specific methods are implemented in subclasses.

public boolean detectI2C\*\*(int addr)\*\*

Used to check the I2C device with address If it exists, return true otherwise, return false

**class PCF8591 extends ADCDevice** (This is a custom class used to operate ADC and DAC of PCF8591)

public PCF8591\*\*(int addr)\*\*

Constructed function used to create PCF8591 class Parameter is the device address of I2C PCF8591

public int analogRead\*\*(int chn)\*\*

Used to read ADC value of one channel of PCF8591 Parameter CHN implies channel number 0,1,2,3

public byte\*\*[ ]\*\* analogRead\*\*( )\*\*

Read the value of ADC on all channels on PCF8591

public void analogWrite\*\*(int data)\*\*

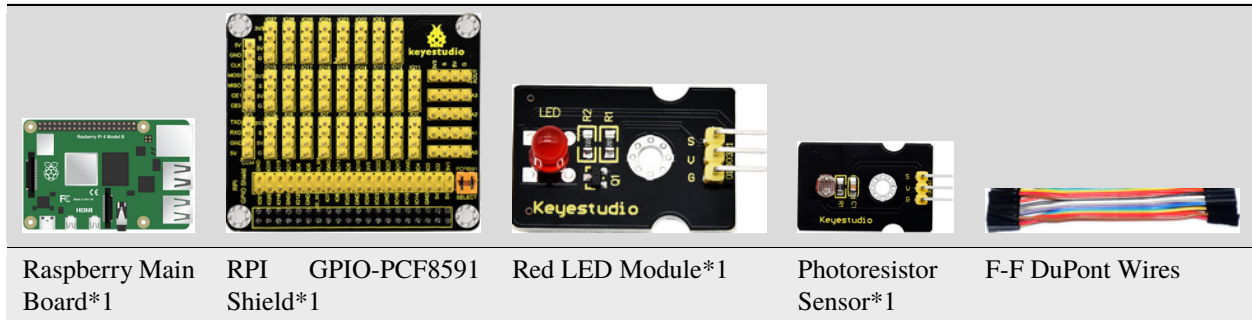
Input DAC value to PCF8591

## 5.2.14 Project 15 Photoresistor

### (1)Description

Photoresistor (Photoresistor) is a resistor whose resistance varies according to different incident light strength. It's made based on the photoelectric effect of semiconductor. In this lesson, let's explain how it works.

### (2)Components Needed



### (3)Knowledge about Component

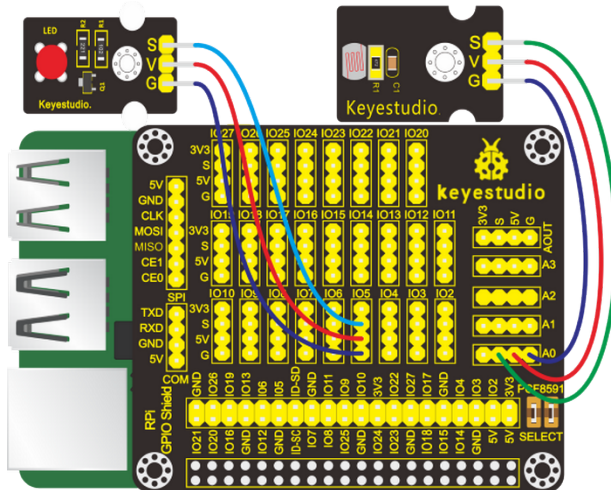
#### Photoresistor

Photoresistor (Photoresistor) is a resistor whose resistance varies according to different incident light strengths. It's made based on the photoelectric effect of semiconductor. If the incident light is intense, its resistance reduces; if the incident light is weak, the resistance increases.

If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance.

### (4)Connection Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Photoresistor Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SA0
V	5V	V	5V
G	G	G	G



### (5)Run Example Code

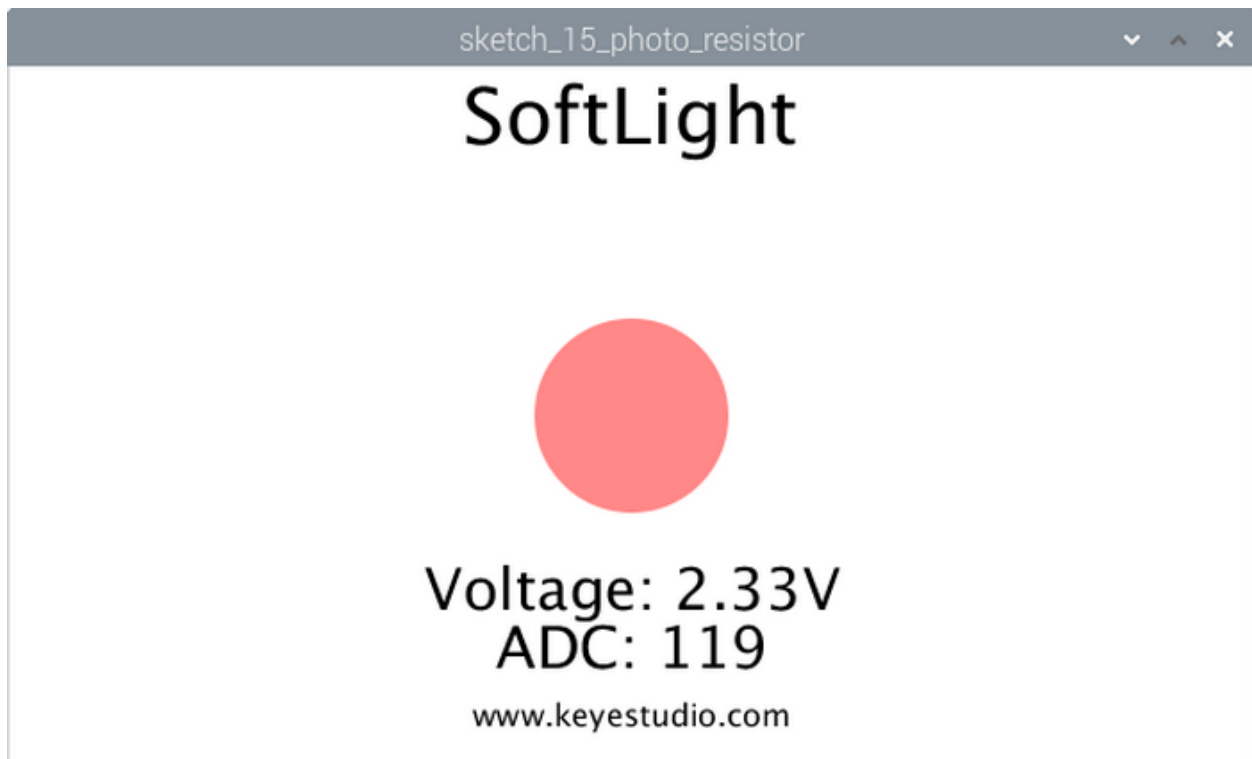
Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press“Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 14 is for your reference.

After enabling the I2C communication, input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_15_photo_resistor/sketch_15_photo_resistor.pde
```

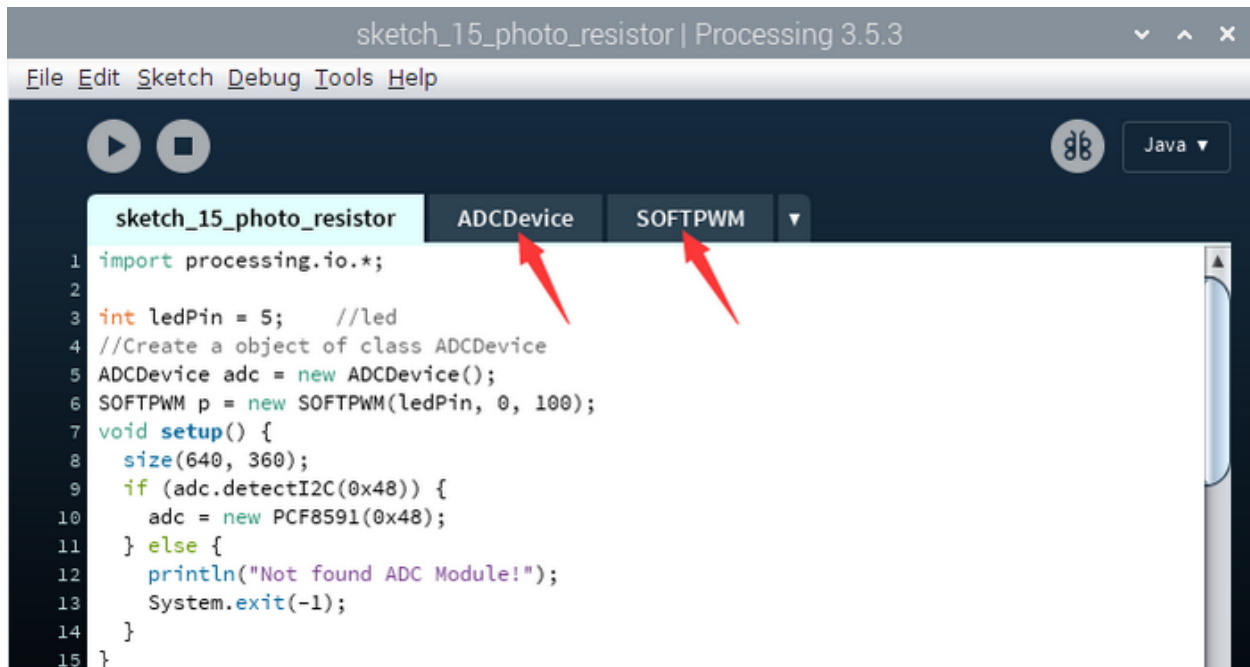
### (6)Test Results

Display window will show the voltage value, ADC value and LED icon. You could change voltage value and its brightness by changing the light intensity of photoresistor,as shown below:



### (7)Example Code

This program includes many code files, the core code is included in sketch\_15\_photo\_resistor.pde file, others are customized, as shown below:



Code:

```
import processing.io.*;

int ledPin = 5;    //led
//Create a object of class ADCDevice
ADCDevice adc = new ADCDevice();
SOFTPWM p = new SOFTPWM(ledPin, 0, 100);
void setup() {
  size(640, 360);
  if (adc.detectI2C(0x48)) {
    adc = new PCF8591(0x48);
  } else {
    println("Not found ADC Module!");
    System.exit(-1);
  }
}

void draw() {
  int adcValue = adc.analogRead(0);    //Read the ADC value of channel 0
  float volt = adcValue*5.0/255.0;    //calculate the voltage
  float dt = adcValue/255.0;
  p.softPwmWrite((int)(dt*100));    //output the pwm
  background(255);
  titleAndSiteInfo();

  fill(255, 255-dt*255, 255-dt*255);    //cycle
  noStroke();    //no border
  ellipse(width/2, height/2, 100, 100);
}
```

(continues on next page)

(continued from previous page)

```

fill(0);
textAlign(CENTER);    //set the text centered
textSize(30);
text("ADC: "+nf(adcValue, 3, 0), width / 2, height/2+130);
text("Voltage: "+nf(volt, 0, 2)+"V", width / 2, height/2+100);
}
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);          //set text size
  text("SoftLight", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}

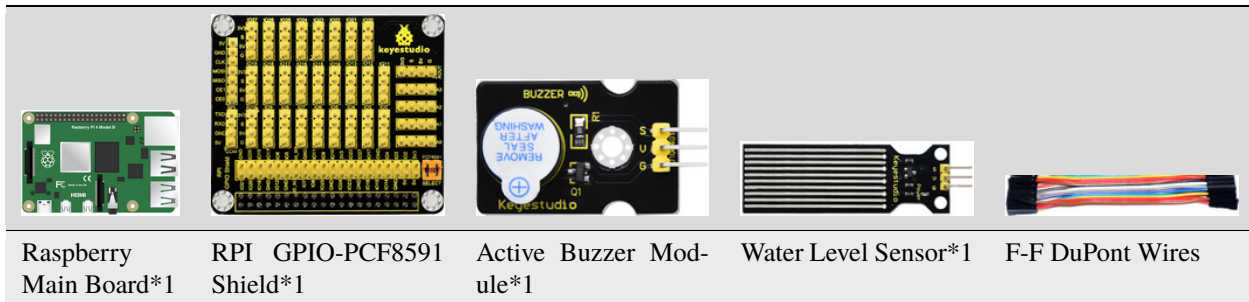
```

## 5.2.15 Project 16 Water Level Monitor

### (1)Description

In daily life, when there is heavy or even torrential rain, the water level in rivers or reservoirs soars. And when it reaches a certain water level, it is necessary to open the gates to discharge the flood to solve the hidden safety hazards. But how to detect the water level in a river or a reservoir? The answer lies in the water level sensor. In this lesson, we will learn to use this sensor to issue alarms when the water bucket is almost full.

### (2)Components Needed:



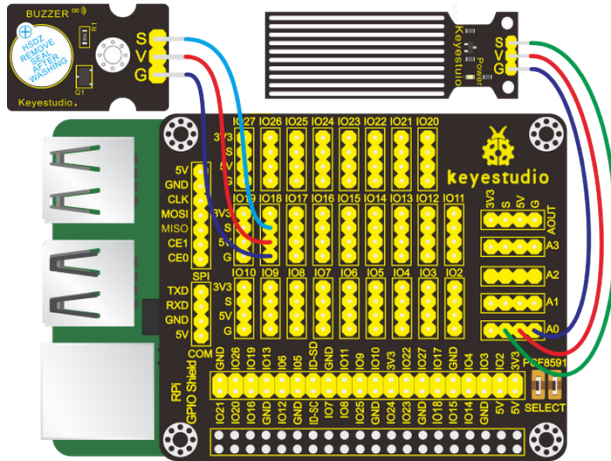
### (3)Knowledge about Component

#### Water Level Sensor

Our water sensor is easy- to-use, portable and cost-effective, designed to identify and detect water level and water drop. This sensor measures the volume of water drop and water quantity through an array of traces of exposed parallel wires. It could convert water content to analog signals, and output analog value could be used by function of application. It has the features of low consumption as well.

### (4)Connection Diagram

Active Buzzer Module	RPI Shield	GPIO-PCF8591	Water Level Sensor	RPI Shield	GPIO-PCF8591
S	SIO18		S	SA0	
V	5V		V	5V	
G	G		G	G	



### (5)Run Example Code

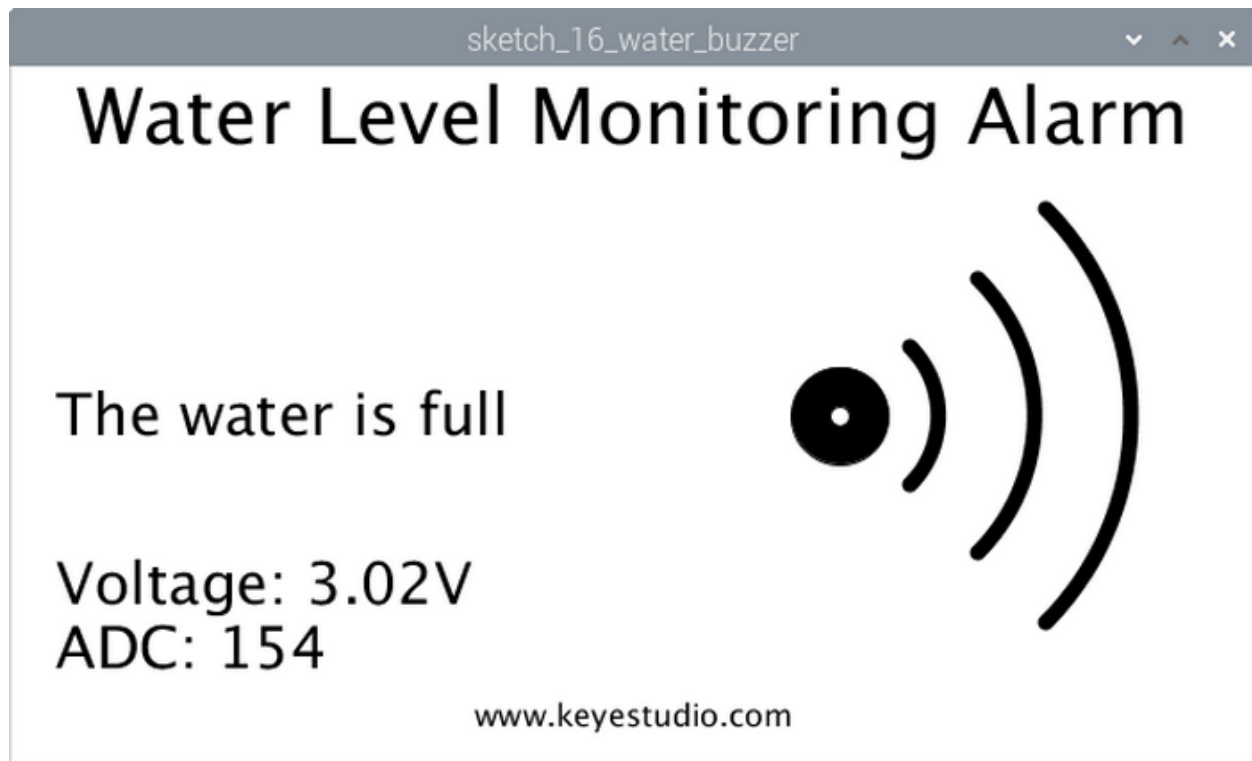
Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press“Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 14 is for your reference.

After enabling the I2C communication, input the following command, press“Enter”and click“RUN”on Processing IDE:  
`processing /home/pi/sketchbook/Processing_Code/sketch_16_water_buzzer/sketch_16_water_buzzer.pde`

### (6)Test Results

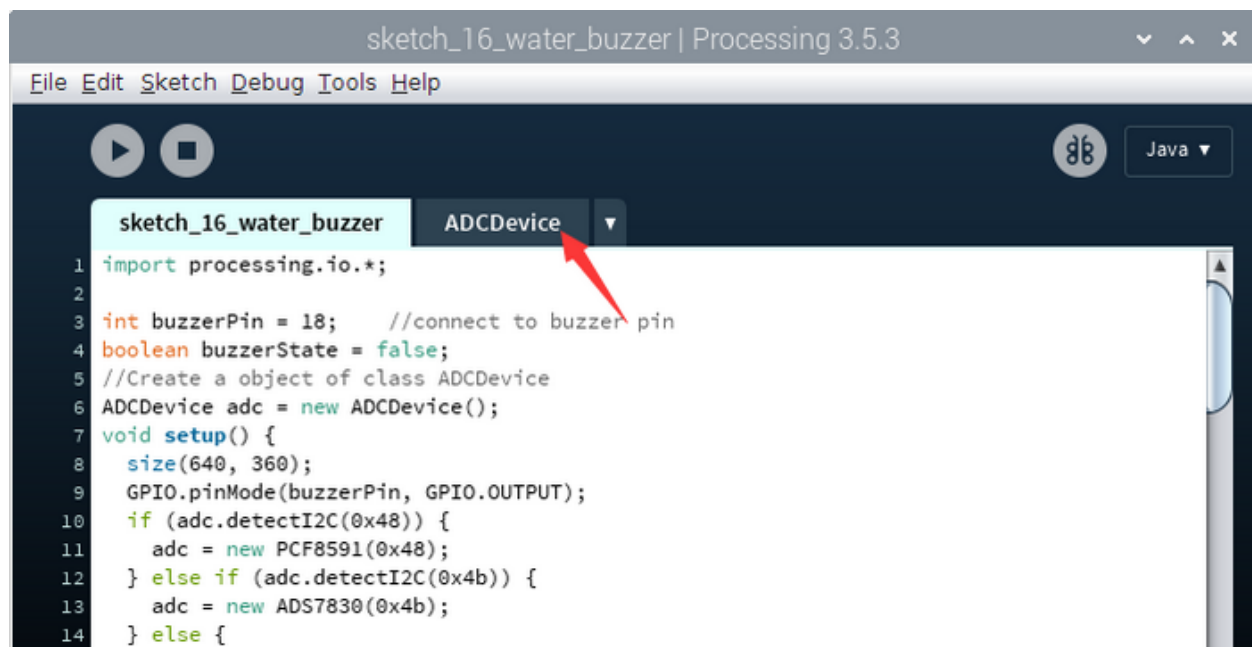
The window will show the voltage value, ADC value and buzzer icon after executing code.

The voltage value and ADC value vary with the depth of water level sensor in the water. Buzzer will emit , arc lines next to buzzer icon will be shown and“`The water is full`”appear, when the water level reaches the certain value. A shown below:



### (7)Example Code

This program includes a few code files, the core code is in the sketch\_16\_water\_buzzer.pde other files consist of some customized ones, as shown below:



Code:

```
import processing.io.*;
```

(continues on next page)



(continued from previous page)

```

int buzzerPin = 18;    //connect to buzzer pin
boolean buzzerState = false;
//Create a object of class ADCDevice
ADCDevice adc = new ADCDevice();
void setup() {
    size(640, 360);
    GPIO.pinMode(buzzerPin, GPIO.OUTPUT);
    if (adc.detectI2C(0x48)) {
        adc = new PCF8591(0x48);
    } else if (adc.detectI2C(0x4b)) {
        adc = new ADS7830(0x4b);
    } else {
        println("Not found ADC Module!");
        System.exit(-1);
    }
}
void draw() {
    int adcValue = adc.analogRead(0);    //Read the ADC value of channel 0
    float volt = adcValue*5.0/255.0;    //calculate the voltage
    background(255);
    titleAndSiteInfo();

    fill(0);
    textAlign(LEFT);    //set the text lefted
    textSize(30);
    text("ADC: "+nf(adcValue, 3, 0), width / 28, height/1.5+70);
    textSize(30);    //set text size
    text("Voltage: "+nf(volt, 0, 2)+"V", width / 28, height/1.30);
    drawBuzzer();    //buzzer img
    if (adcValue > 150) {
        GPIO.digitalWrite(buzzerPin, GPIO.HIGH);    //buzzer on
        drawArc();    //Sounds waves img
        fill(0);
        textAlign(LEFT);    //set the text lefted
        textSize(30);
        text("The water is full", width / 28, height/3+70);
    } else {
        GPIO.digitalWrite(buzzerPin, GPIO.LOW);    //buzzer off
    }
}
void drawBuzzer() {
    strokeWeight(1);
    fill(0);
    ellipse(width/1.5, height/2, 50, 50);
    fill(255);
    ellipse(width/1.5, height/2, 10, 10);
}
void drawArc() {
    noFill();
    strokeWeight(8);
    for (int i=0; i<3; i++) {
        arc(width/1.5, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
}
void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);    //set the text centered
    textSize(40);         //set text size
    text("Water Level Monitoring Alarm", width / 2, 40);    //title
    textSize(16);
    text("www.keyestudio.com", width / 2, height - 20);    //site
}

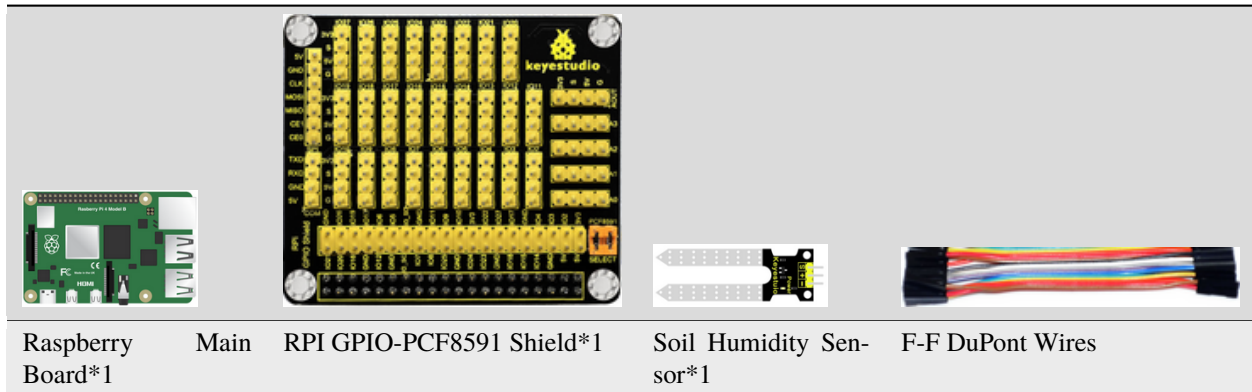
```

## 5.2.16 Project 17 Flower-watering Device

### (1)Description

The household plants are popular in many communities. But they will die if you forget to water them, how about making an automatic watering device? In this project, we will learn to detect the soil humidity of your plants with soil humidity sensor and Raspberry Pi.

### (2)Components Needed



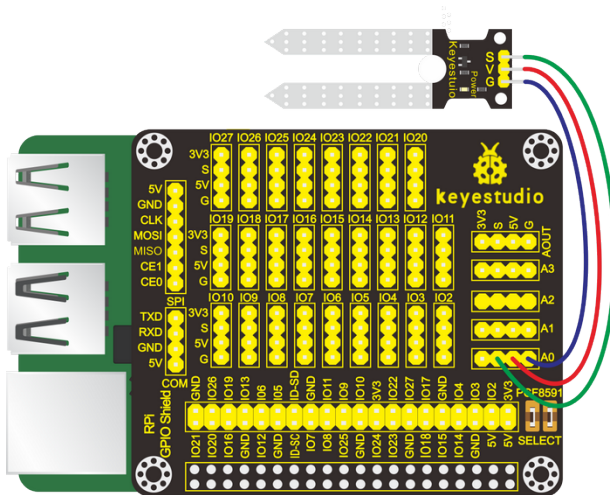
### (3)Knowledge about Component

#### Soil Humidity Sensor

This is a simple soil humidity sensor aims to detect the soil humidity. If the soil is in lack of water, the analog value output by the sensor will decrease; otherwise, it will increase. If you use this sensor to make an automatic watering device, it can detect whether your botany is thirsty to prevent it from withering when you go out.

### (4)Connection Diagram

Soil Humidity Sensor	RPI GPIO-PCF8591 Shield
S	SA0
V	5V
G	G



### (5)Run Example Code

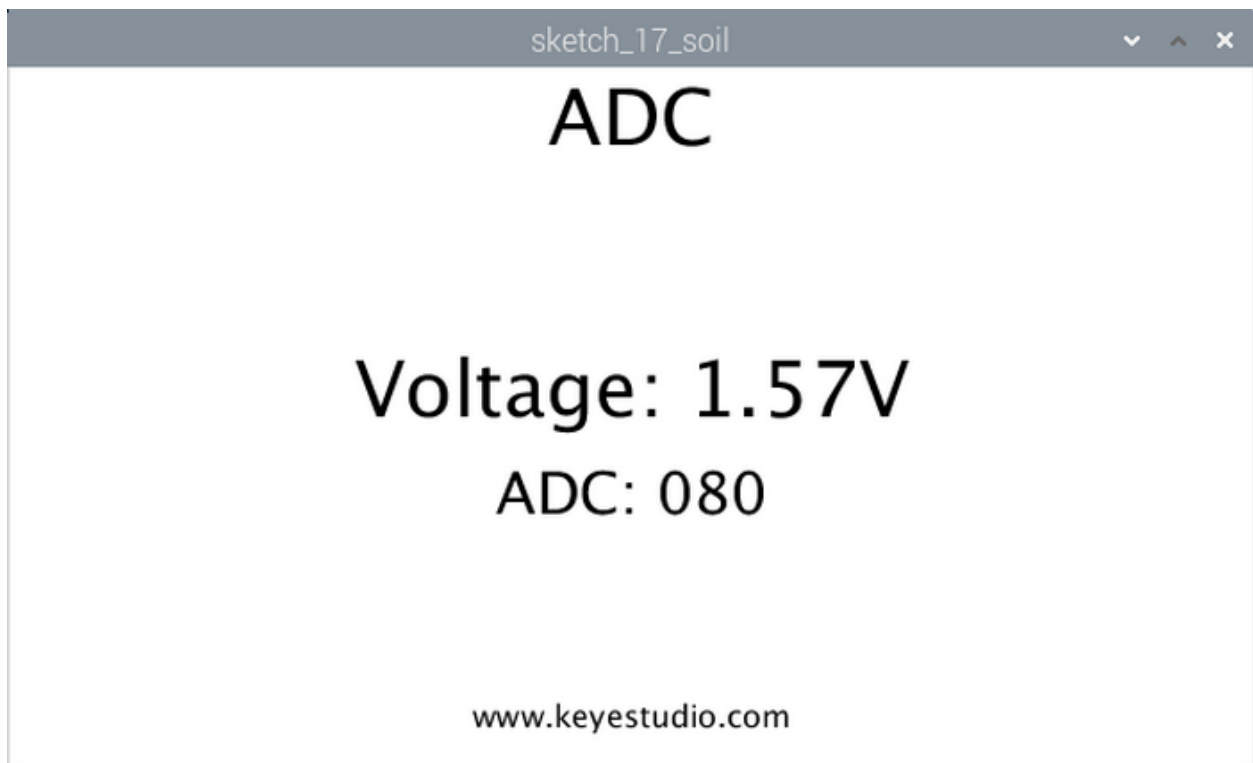
Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press“Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 14 is for your reference.

After enabling the I2C communication, input the following command, press“Enter”and click“RUN”on Processing IDE:

```
processing /home/pi/sketchbook/Processing_Code/sketch_17_soil/sketch_17_soil.pde
```

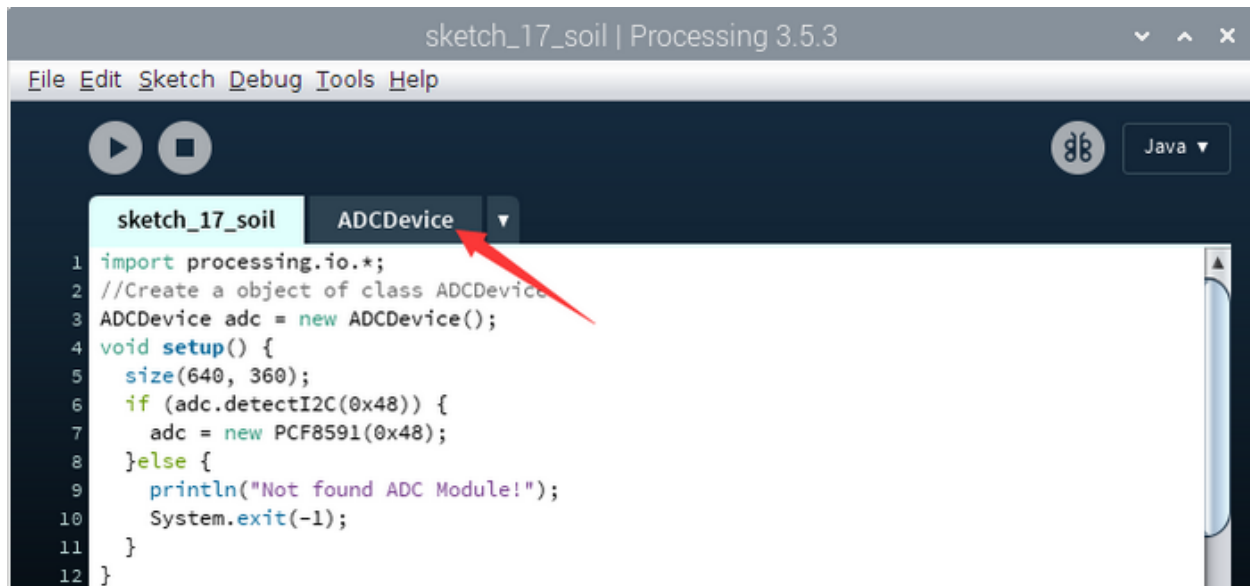
### (6)Test Results

After running the program, when the soil humidity sensor is inserted into the land, the display window shows the analog value of the soil humidity and the value of ADC. The voltage value changes with the humidity of the land.



### (7)Example Code

This program includes a few code files, the core code is in the sketch\_17\_soil.pde and other files consist of some customized ones, as shown below:



Code:

```

import processing.io.*;
//Create a object of class ADCDevice
ADCDevice adc = new ADCDevice();
void setup() {
    size(640, 360);
    if (adc.detectI2C(0x48)) {
        adc = new PCF8591(0x48);
    }else {
        println("Not found ADC Module!");
        System.exit(-1);
    }
}

void draw() {
    int adcValue = adc.analogRead(0);    //Read the ADC value of channel 0
    float volt = adcValue*5.0/255.0;    //calculate the voltage
    background(255);
    titleAndSiteInfo();

    fill(0);
    textAlign(CENTER);    //set the text centered
    textSize(30);
    text("ADC: "+nf(adcValue, 3, 0), width / 2, height/2+50);
    textSize(40);    //set text size
    text("Voltage: "+nf(volt, 0, 2)+"V", width / 2, height/2);    //
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);    //set the text centered
    textSize(40);    //set text size
    text("ADC", width / 2, 40);    //title

```

(continues on next page)

(continued from previous page)

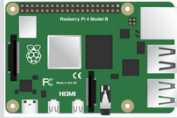
```
    textSize(16);
    text("www.keyestudio.com", width / 2, height - 20);    //site
}
```

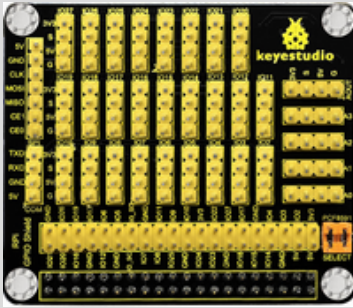
5.2.17 Project 18Joystick


(1)Description


Many a people play games with gamepad. But do you know who it work?  
Let’s learn about it.

(2)Components Needed









Raspberry Board*1	Main	RPI GPIO-PCF8591 Shield*1	Joystick Module*1	F-F DuPont Wires
-------------------	------	---------------------------	-------------------	------------------

(3)Knowledge about Component

Joystick Module

This is a joystick very similar to the ‘analog’ joysticks on PS2 (PlayStation 2) controllers. It is a self-centering spring loaded joystick, meaning when you release the joystick it will center itself. It also contains a comfortable cup-type knob/cap which gives the feel of a thumb-stick.

It has three signal pins which are connected GND, VCC and signal endB, X, Y). The X pin is **X-axis** (left to right), the Y pin is **Y-axis** (front and back) and signal B end is Z-axis(usually used as digital port and pushbutton)

VCC is connected to V/VCC3.3/5Vof MCU, GND to G/GND of MCU and the voltage is around 1.65V/2.5V in initial status

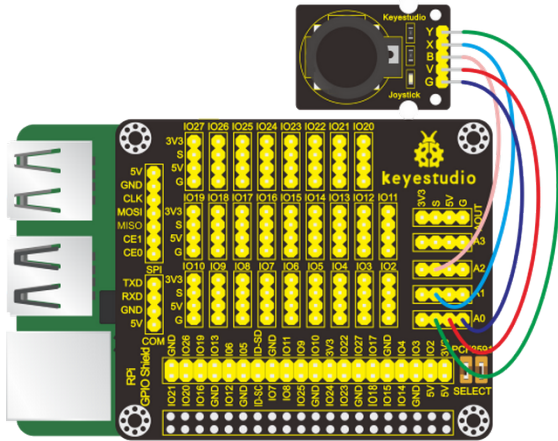
X axis gives readout of the joystick in the horizontal direction (X-coordinate) i.e. how far left and right the joystick is pushed.

Y axis gives readout of the joystick in the vertical direction (Y-coordinate) i.e. how far up and down the joystick is pushed.

Z axis is the output from the pushbutton. It’s normally open, meaning the digital readout from the SW pin will be HIGH. When the button is pushed, it will connect to GND, giving output LOW.

(4)Connection Diagram

Joystick Module	RPI GPIO-PCF8591 Shield
Y	S(A0)
X	S(A1)
B	S(A2)
V	5V
G	G



### (5)Run Example Code

Note: in the experiment, I2C communication is used. We need to check the iic address first( enter command `i2cdetect -y 1` and press“Enter”. If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 14 is for your reference.

After enabling the I2C communication, input the following command, press“Enter”and click“RUN”on Processing IDE:

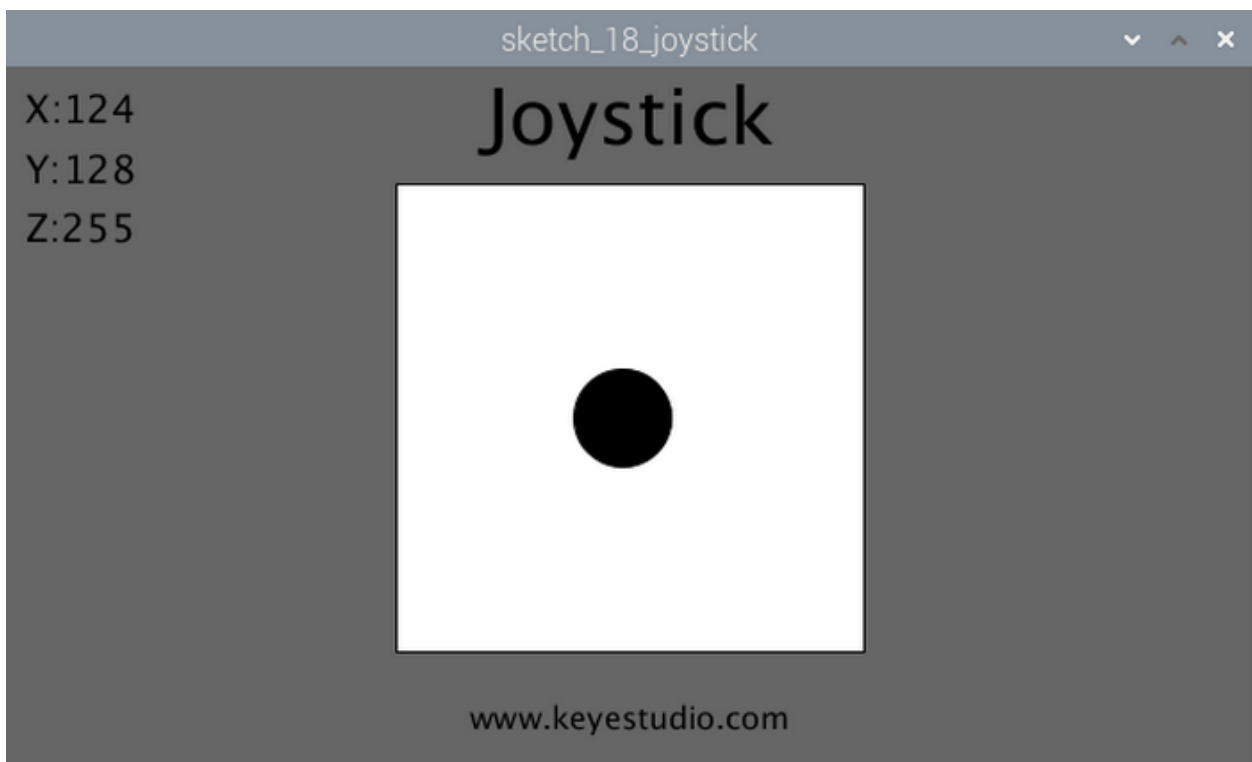
`processing /home/pi/sketchbook/Processing_Code/sketch_18_joystick/sketch_18_joystick.pde`

### (6)Test Results

This window shows the position of joystick and values of X, Y and Z axis, as shown below:

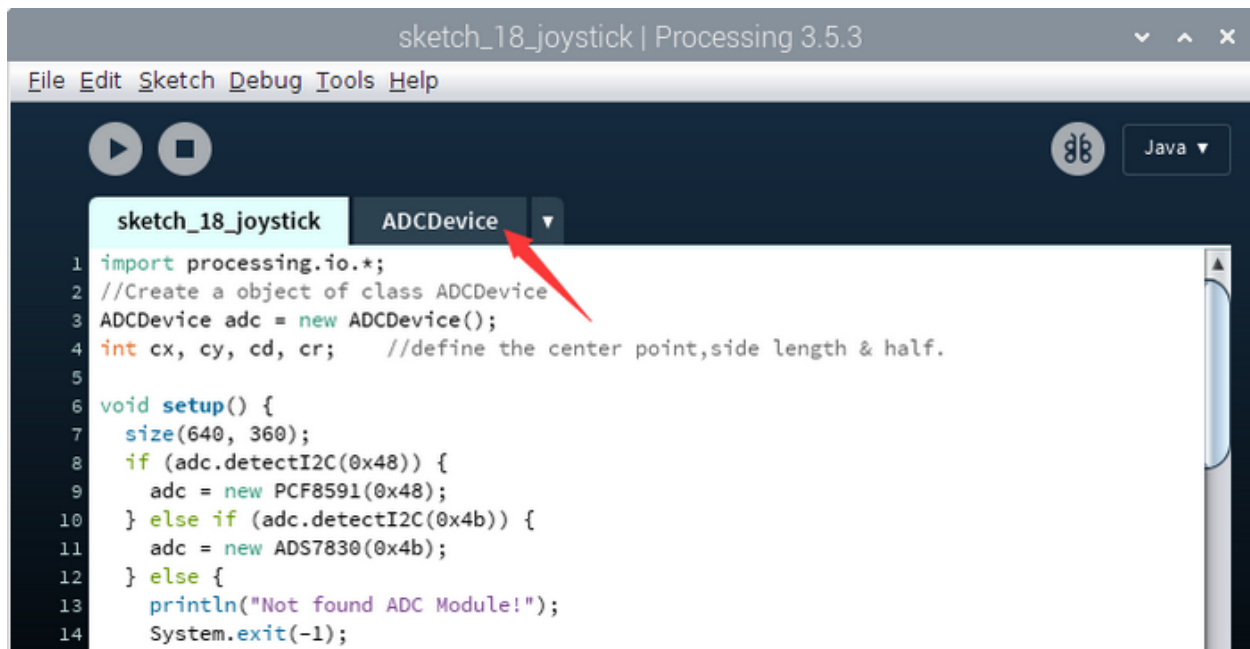


The color of circle will change if button (Z axis/B) is pressed.



#### (7)Example Code

A few code files are included, as shown below:



Code:

```

import processing.io.*;
//Create a object of class ADCDevice
ADCDevice adc = new ADCDevice();
int cx, cy, cd, cr;    //define the center point,side length & half.

void setup() {
    size(640, 360);
    if (adc.detectI2C(0x48)) {
        adc = new PCF8591(0x48);
    } else if (adc.detectI2C(0x4b)) {
        adc = new ADS7830(0x4b);
    } else {
        println("Not found ADC Module!");
        System.exit(-1);
    }
    cx = width/2;    //center of the display window
    cy = height/2;    //
    cd = (int)(height/1.5);
    cr = cd /2;
}

void draw() {
    int x=0, y=0, z=0;
    x = adc.analogRead(1); //read the ADC of joystick
    y = adc.analogRead(0); //
    z = adc.analogRead(2);
    background(102);
    titleAndSiteInfo();
    fill(0);
    textSize(20);
    textAlign(LEFT, TOP);
    text("X:"+x+"\nY:"+y+"\nZ:"+z, 10, 10);
}

```

(continues on next page)



(continued from previous page)

```
fill(255);    //wall color
rect(cx-cr, cy-cr, cd, cd);
fill(constrain(z, 255, 0));    //joystick color
ellipse(map(x, 0, 255, cx-cr, cx+cr), map(y, 0, 255, cy-cr, cy+cr), 50, 50);
}
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);    //set text size
  text("Joystick", width / 2, 40);    //title
  textSize(16);
  text("www.keyestudio.com", width / 2, height - 20);    //site
}
```



## **PYTHON TUTORIAL**

Raspberry Pi and electronic components are controlled via Python.

### **6.1 1. Install Raspberry Pi OS System**

#### **6.1.1 1.1Hardware Tool**

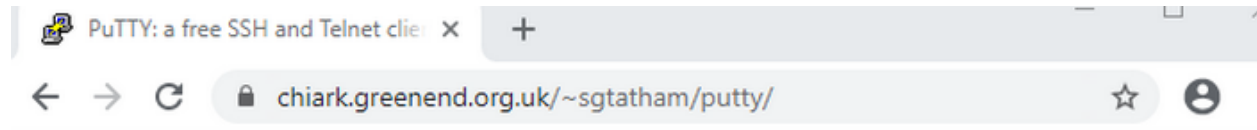
- Raspberry Pi 4B/3B/2B
- Above 8G TFT SD Card
- Card Reader
- Computer and other parts

#### **6.1.2 1.2Software 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 [Simon Tatham](#).

The latest version is 0.74 [Download it here.](#)

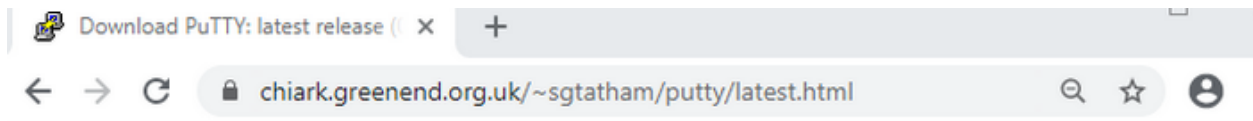
**LEGAL WARNING:** Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. We believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but we are not lawyers, and so if in doubt you should seek legal advice before downloading it. You may find useful information at [cryptolaw.org](http://cryptolaw.org), 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 PuTTY: latest release (0.74)

[Home](#) | [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 [permanent link to the 0.74 release](#).

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 [development snapshots](#), to see if the problem has already been fixed in those versions.

**Package files**

You probably want one of these. They include versions of all the PuTTY utilities.


(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

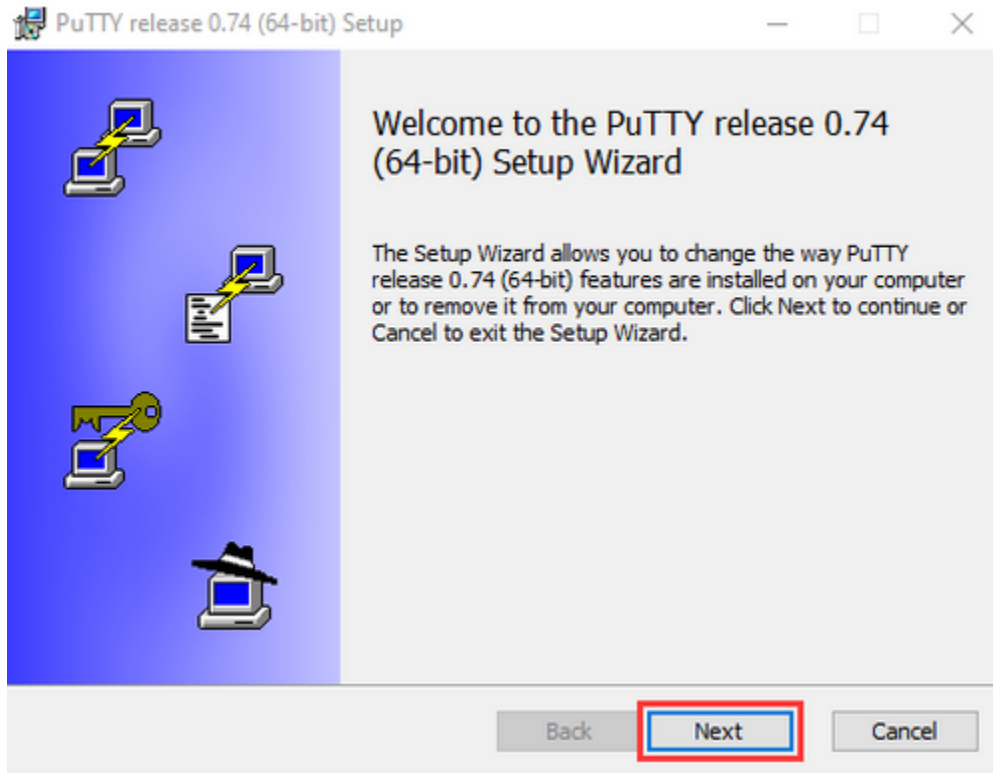
**MSI ('Windows Installer')**

32-bit:	<a href="#">putty-0.74-installer.msi</a>	<a href="#">(or by FTP)</a>	<a href="#">(signature)</a>
64-bit:	<a href="#">putty-64bit-0.74-installer.msi</a>	<a href="#">(or by FTP)</a>	<a href="#">(signature)</a>

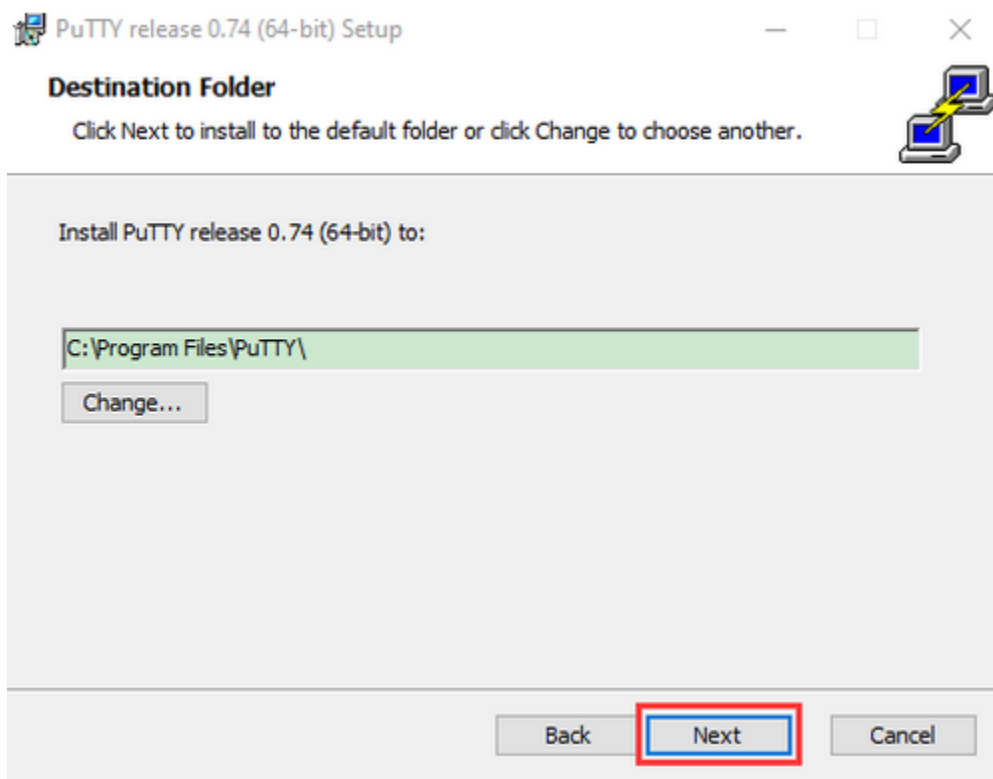
**Unix source archive**

.tar.gz:	<a href="#">putty-0.74.tar.gz</a>	<a href="#">(or by FTP)</a>	<a href="#">(signature)</a>
----------	-----------------------------------	-----------------------------	-----------------------------

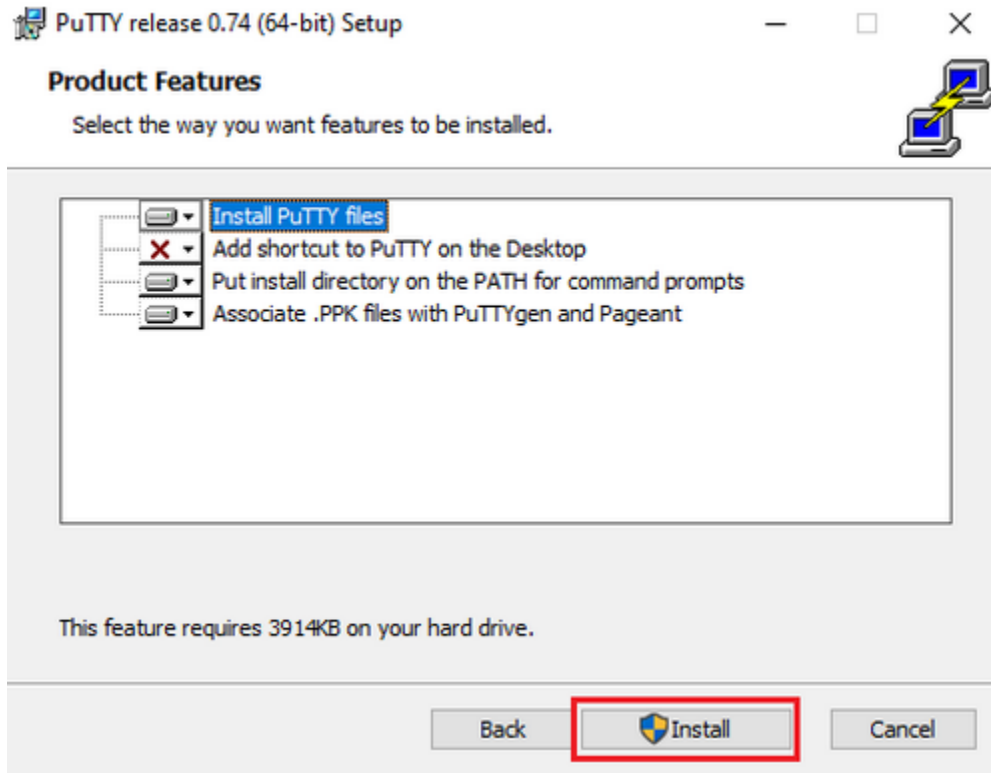
After downloading the driver file  **putty-64bit-0.74-installer** double-click it and tap "Next".



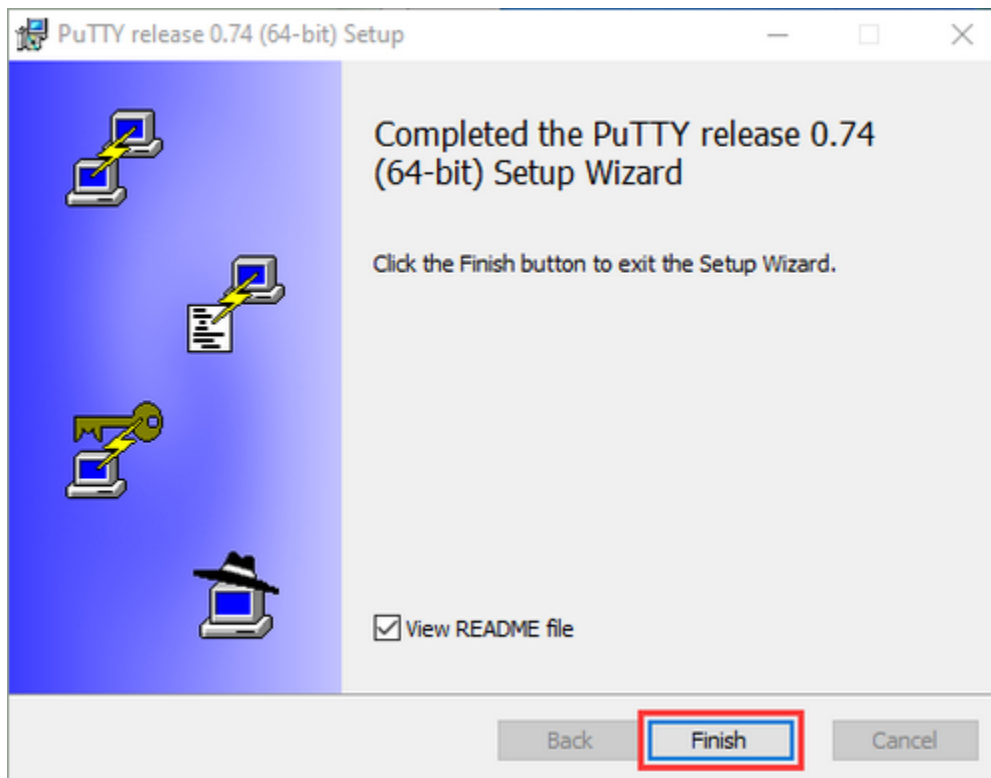
Click "Next".



Select "Install Putty files" and click "Install".


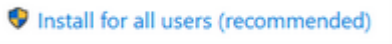


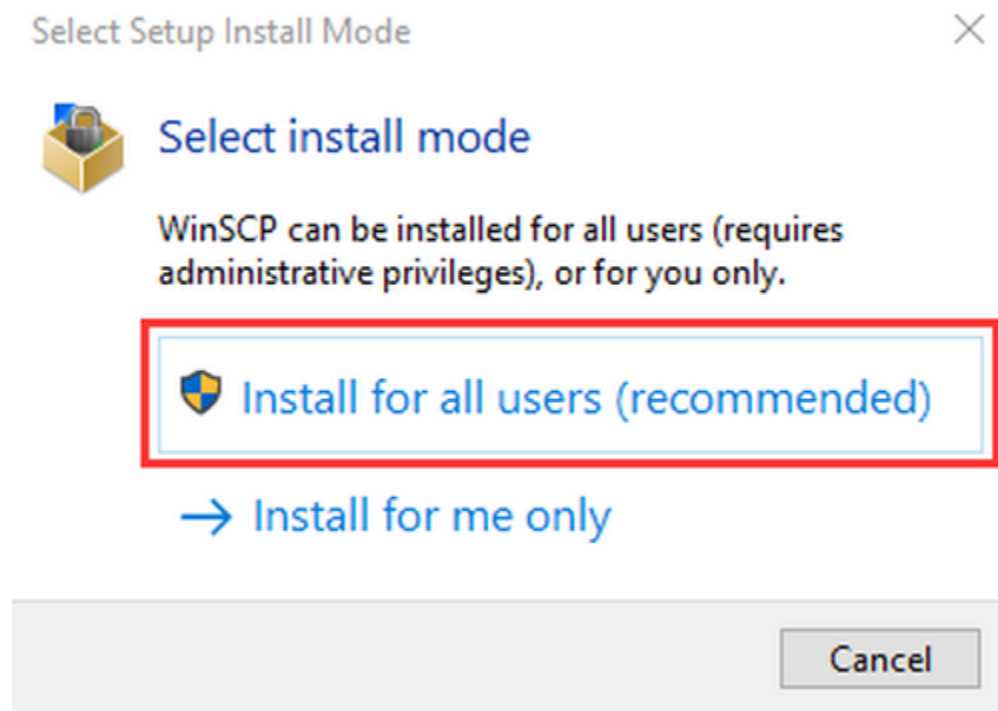
After a few seconds, click "Finish".



## (2) SSH Remote Login software -WinSCP

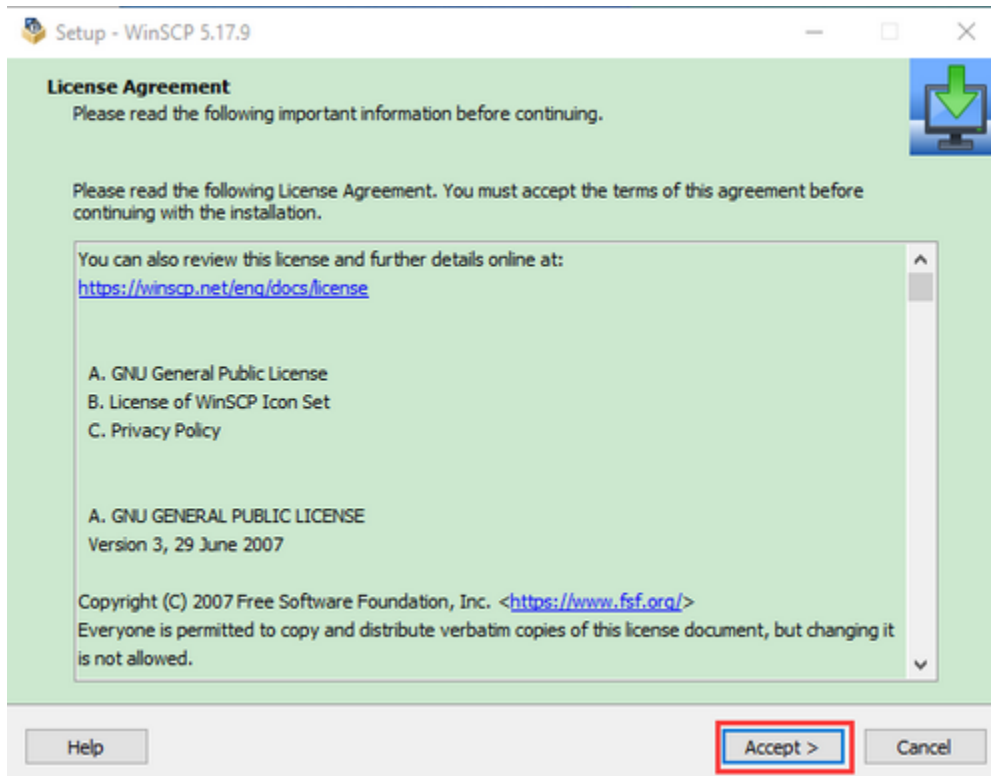
Download WinSCP: <https://winscp.net/eng/download.php>

After the download, click  **WinSCP-5.17.9-Setup.exe** and .

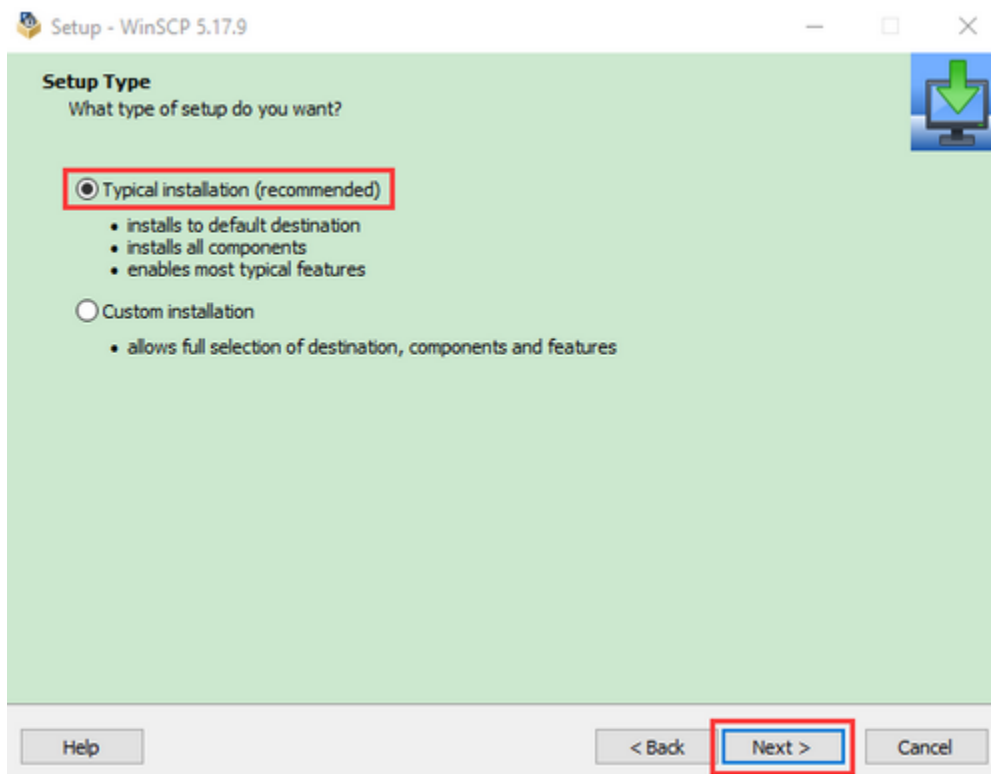


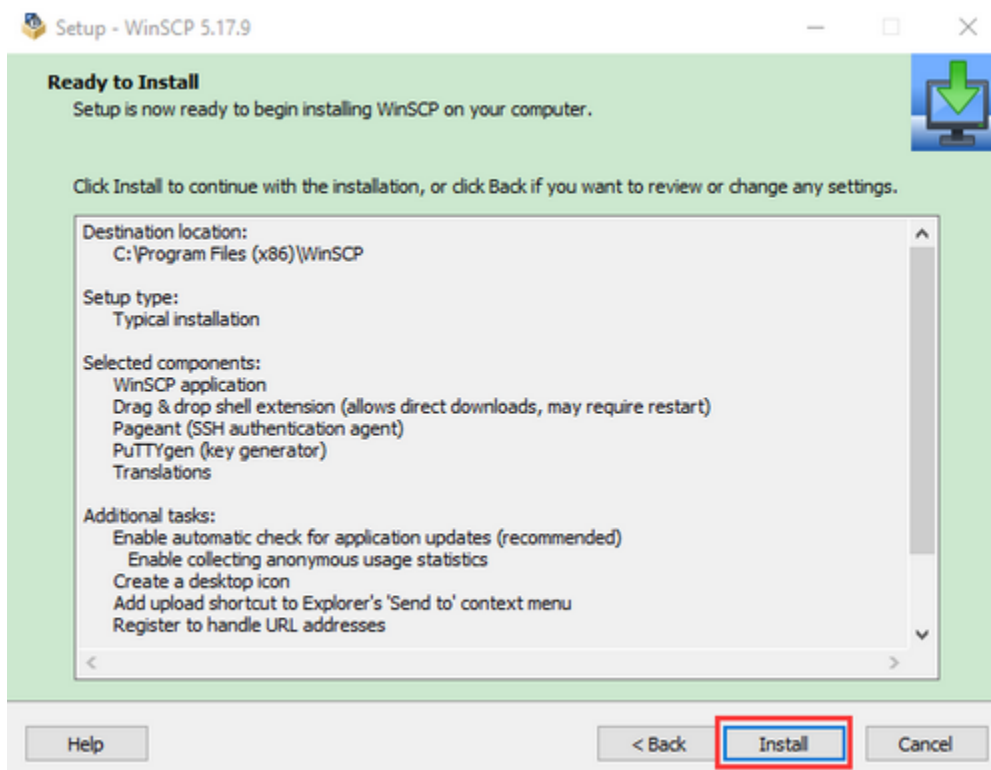
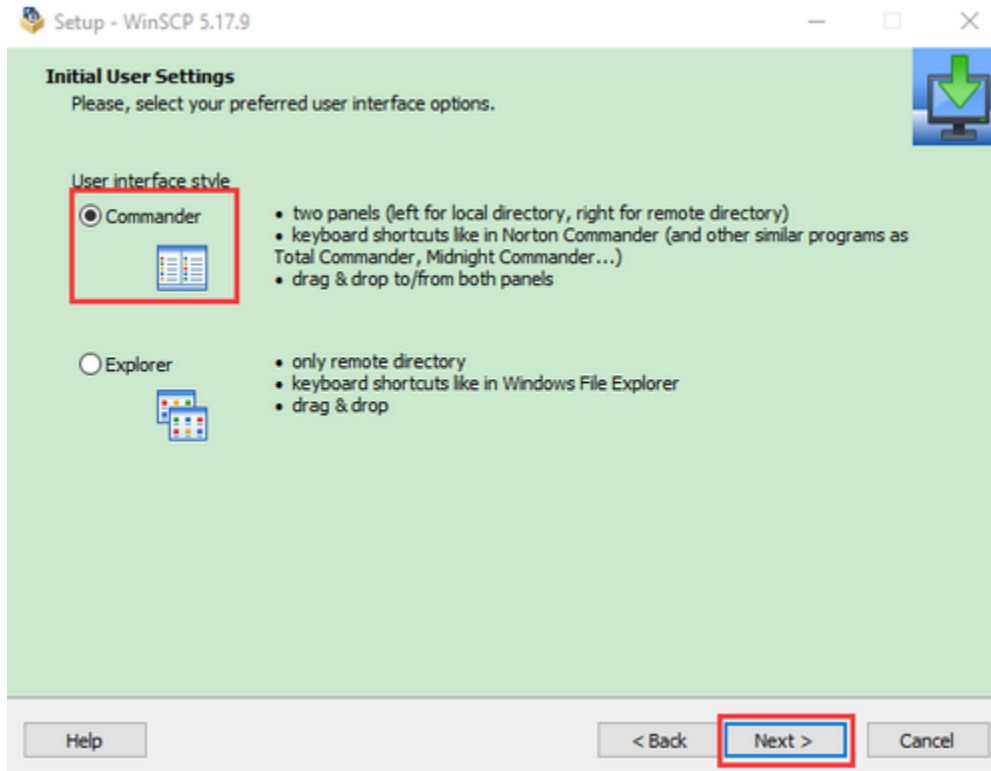
Click "Accept".





Follow the below steps to finish the installation.







### (3) SD Card Formatter

Format TFT card tool


Download SD Card Formatter

[http://www.canadiancontent.net/tech/download/SD\\_Card\\_Formatter.html](http://www.canadiancontent.net/tech/download/SD_Card_Formatter.html)

## SD Card Formatter

Free Formatter Download

Software Downloads ▸ Hardware Software ▸ Hard Drive Software ▸ Hard Drive Formatters ▸



**SD Card Formatter 5.0.1**  
Update Submitted 12 May 2019

★★★★☆

**Software Review:**


SD Card Formatter is a simple and basic formatted which is designed to be used with SD, SDHC and SDXC memory cards.

The application itself isn't too different from the format utility included with Windows and includes two modes: Quick format and Overwrite format. CHS format size adjustment is the only other option.

Once the appropriate card and volume label has been selected, the format can begin after hitting "Format".

Version 5.0.1 is a freeware program which does not have restrictions and it's free so it doesn't cost anything.


### Download File



**Download SD Card Formatter**  
6 MB - Filesize

### Details

**Publisher:** Tuxera  
**License:** Freeware  
**OS/Platform:** Windows 7, Windows 8 (64-bit, 32-bit) / Vista / XP  
**Filesize:** 6 MB  
**Filename:** SDCardFormatterv5\_WinEN.z...  
**Cost (Full Version):** Free  
**Rating:** 3 out of 5 based on 1 rating.  
**Notes** ▸ This file download is licensed as freeware for Windows 7, Windows 8 (64-bit, 32-bit) / Vista / XP.  
**TrustRank** Based on many factors, we give this program a Trust rating of 5 / 10.



SD Card Formatter screenshot

## CanadianContent

Register Account

Software Downloads ▸ Hardware Software ▸ Hard Drive Software ▸ Hard Drive Formatters ▸ SD Card Formatter ▸

Download SD Card Formatter

**Download SD Card Formatter 5.0.1 (x64 & x32) Free**

Have you tried the SD Card Formatter before? If yes, please consider recommending it by clicking the Facebook "Recommend" button!

Download SD Card Formatter 5.0.1 from **Hosted by Sdcard.org**


**SD Card Formatter has been tested for viruses and malware**

This download is 100% clean of viruses. It was tested with 24 different antivirus and anti-malware programs and was clean **100%** of the time. View the full SD Card Formatter homepage for virus test results.


The file that was tested: SDCardFormatterv5\_WinEN.zip.

Tip: If you're experiencing trouble downloading this file, please disable any download managers to SD Card Formatter you may be using.

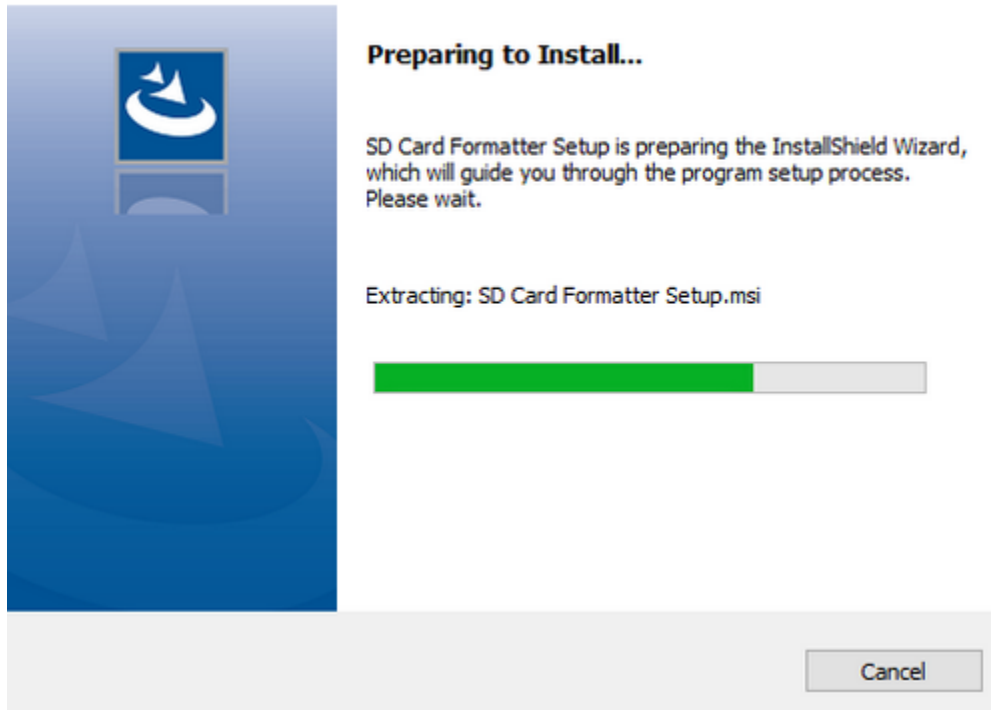
If you're receiving a 404 File Not Found error, this means the publisher has taken the file offline and has not updated their links with us for SD Card Formatter. Please do drop us a note in the event of a missing file.



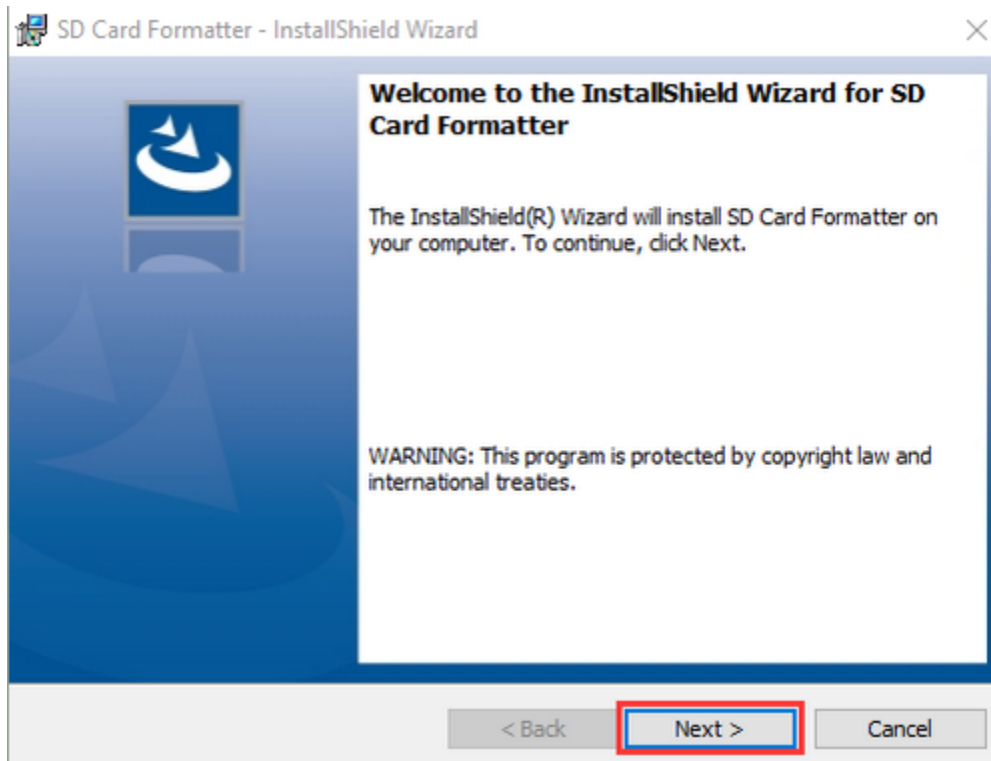
SD Card Formatter 5.0.1 Screenshot

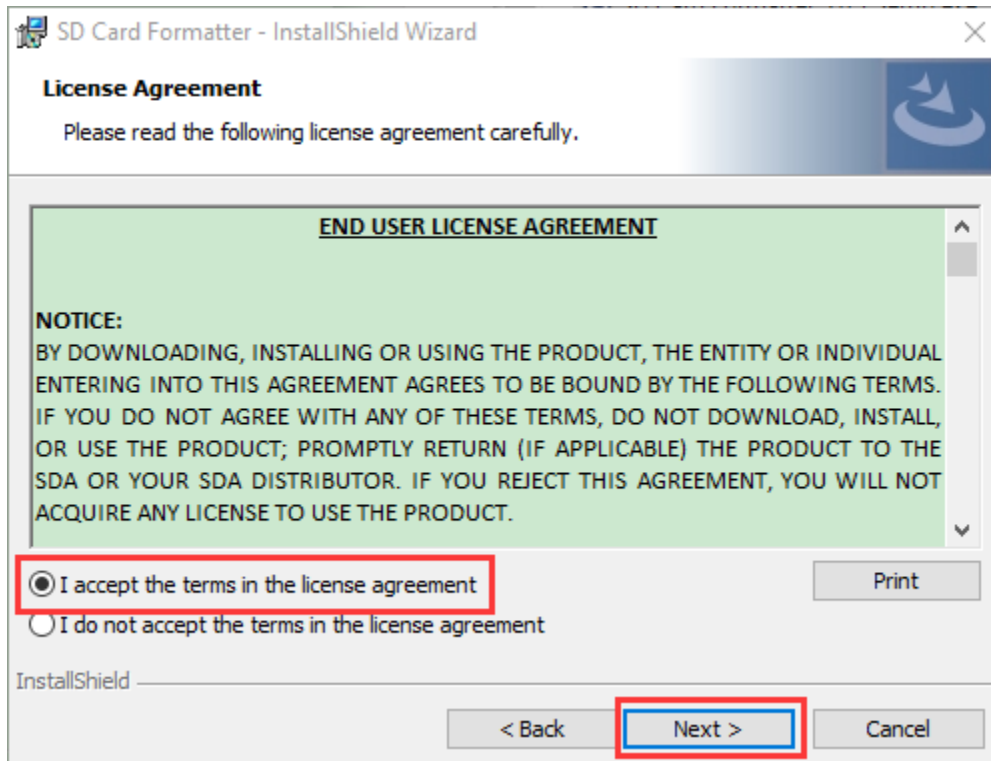
Unzip the SDCardFormatterv5\_WinEN package, double-click  SD Card Formatter 5.0.1 Setup.exe to run it.

SD Card Formatter - InstallShield Wizard

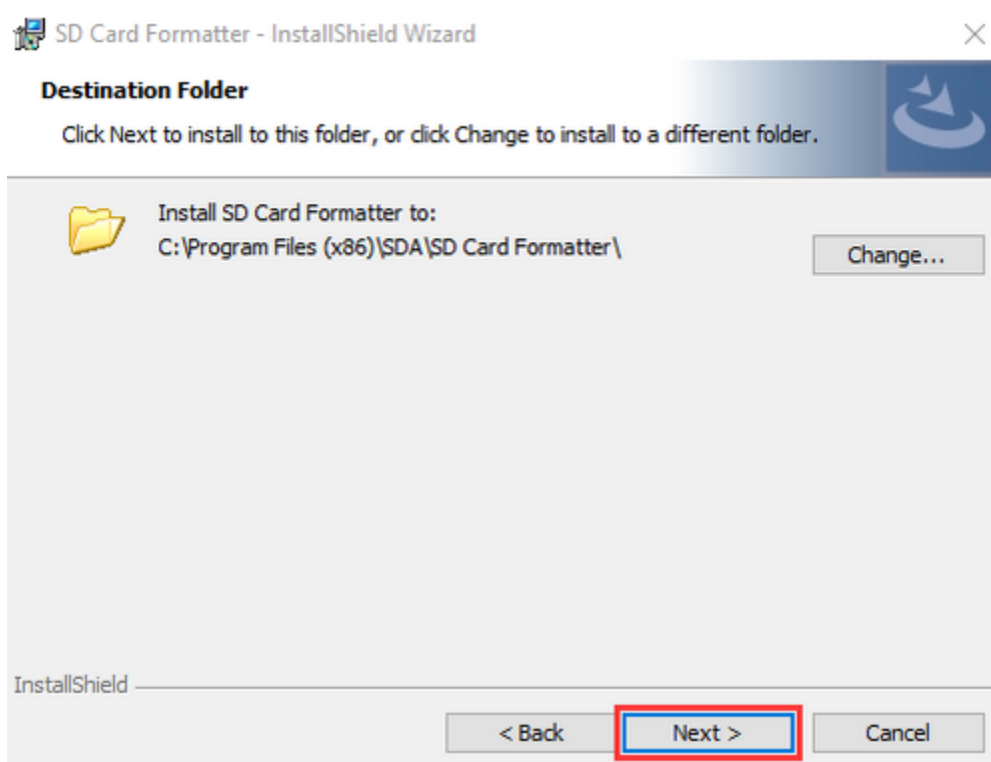


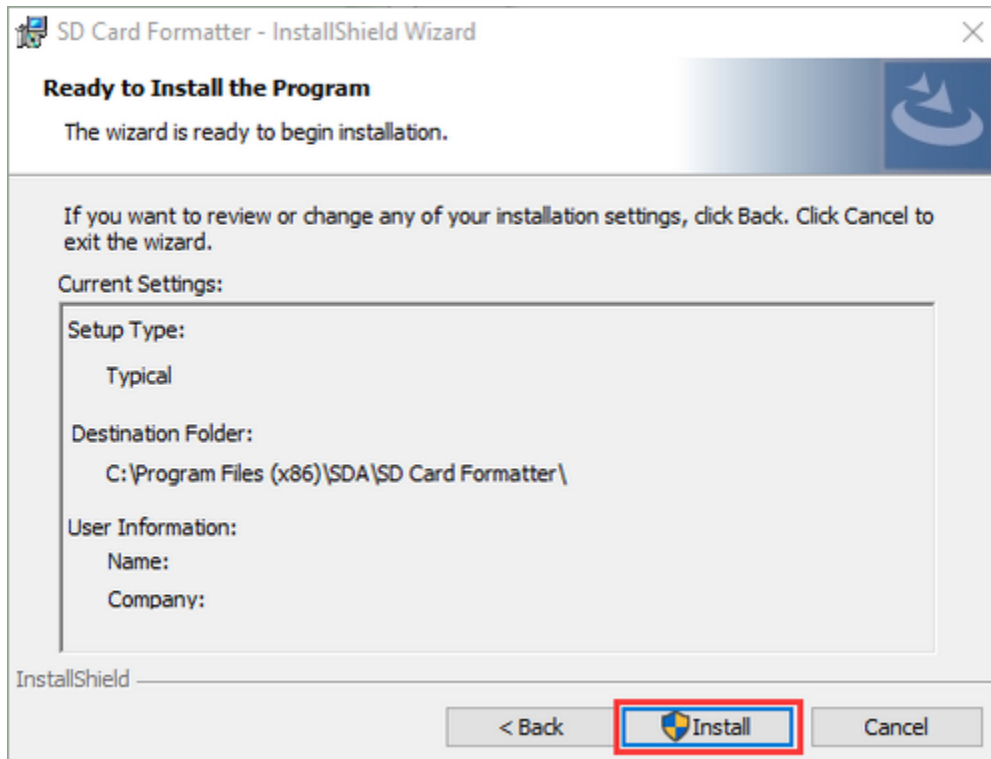
Click "Next" and choose ☒ I accept the terms in the license agreement, then tap "Next".



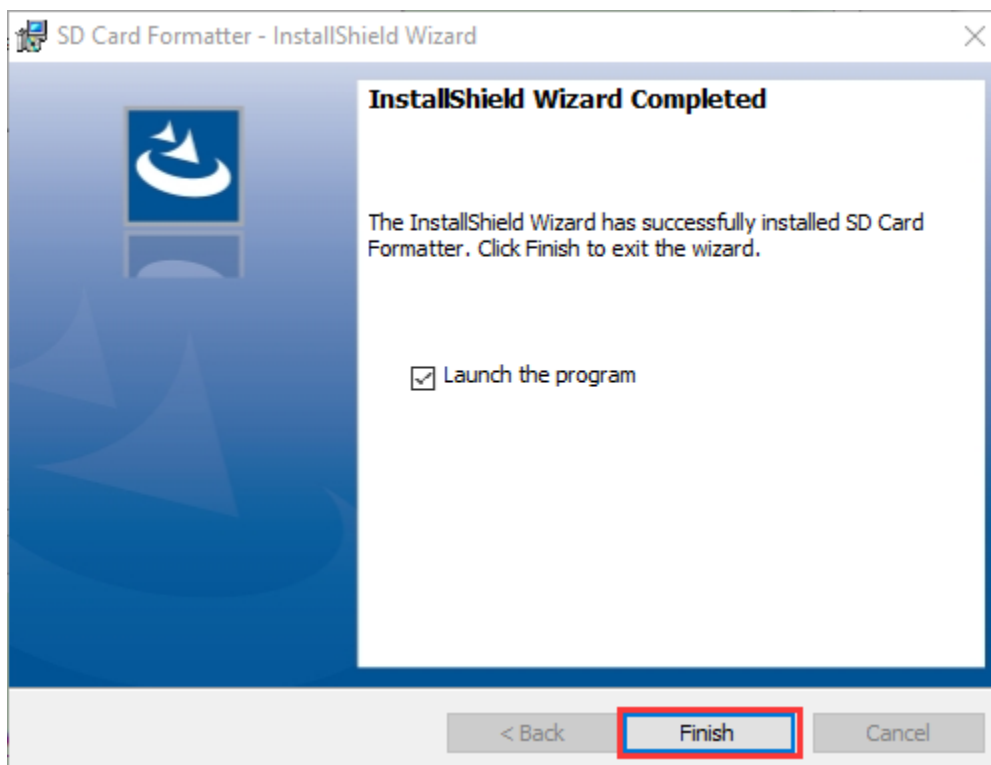


Click “Next” and “Install”.





After a few seconds, click“Finish”.



#### (4) Burn Win32DiskImager

Download Link <https://sourceforge.net/projects/win32diskimager/>



The screenshot shows the SourceForge project page for Win32 Disk Imager. At the top, the breadcrumb navigation reads "Home / Browse / System Administration / Storage / Win32 Disk Imager". The project title "Win32 Disk Imager" is prominently displayed, followed by the description "A Windows tool for writing images to USB sticks or SD/CF cards" and the credit "Brought to you by: gruemaster, tuxinator2009". A "PROJECT OF THE MONTH MAR 2014" badge is in the top right. Below the title, there are four yellow stars, "112 Reviews", "Downloads: 42,251 This Week", and "Last Update: 2018-06-07". A green "Download" button with a red border is highlighted, along with "Get Updates" and "Share This" buttons. A navigation bar at the bottom includes links for Summary, Files, Reviews, Support, Wiki, Feature Requests, Bugs, Code, Mailing Lists, and Blog. The "Summary" tab is active, showing a paragraph about the program's purpose and another paragraph about system compatibility.

Home / Browse / System Administration / Storage / Win32 Disk Imager

## Win32 Disk Imager

A Windows tool for writing images to USB sticks or SD/CF cards

Brought to you by: gruemaster, tuxinator2009


★★★★★ 112 Reviews Downloads: 42,251 This Week Last Update: 2018-06-07

**Download** Get Updates Share This

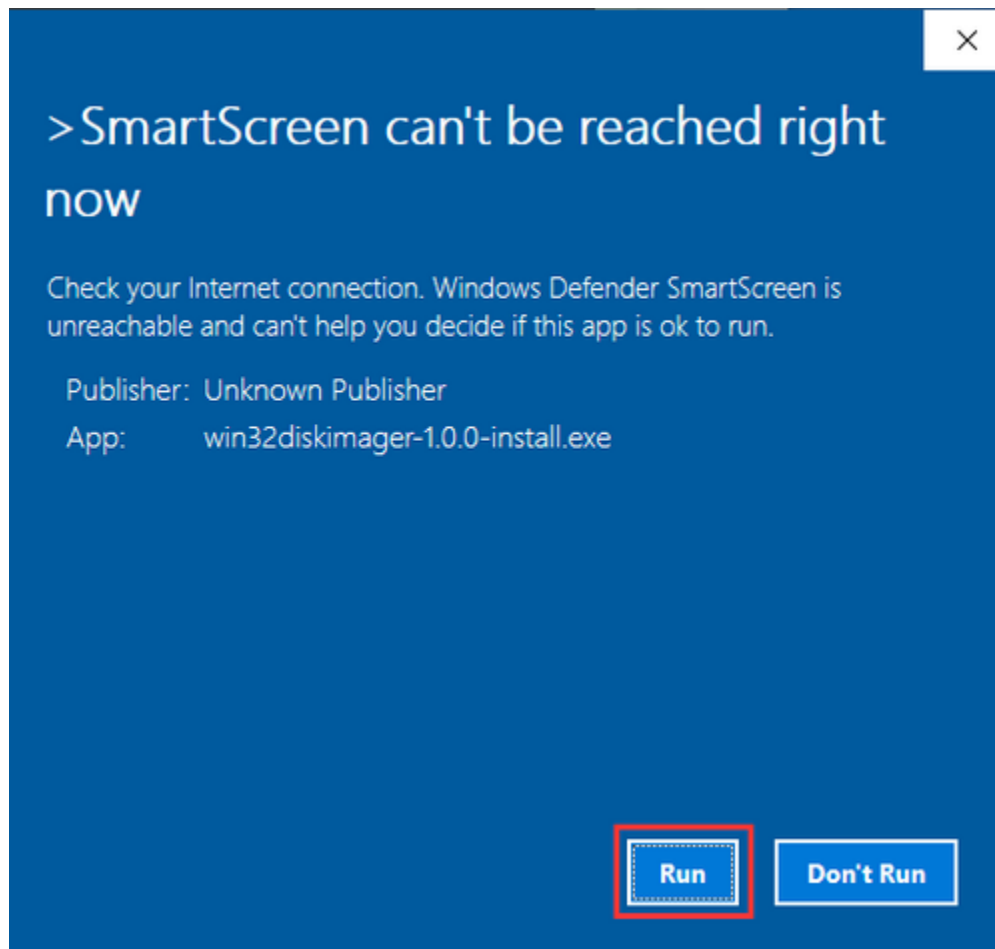
[Summary](#) [Files](#) [Reviews](#) [Support](#) [Wiki](#) [Feature Requests](#) [Bugs](#) [Code](#) [Mailing Lists](#) [Blog](#)

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.

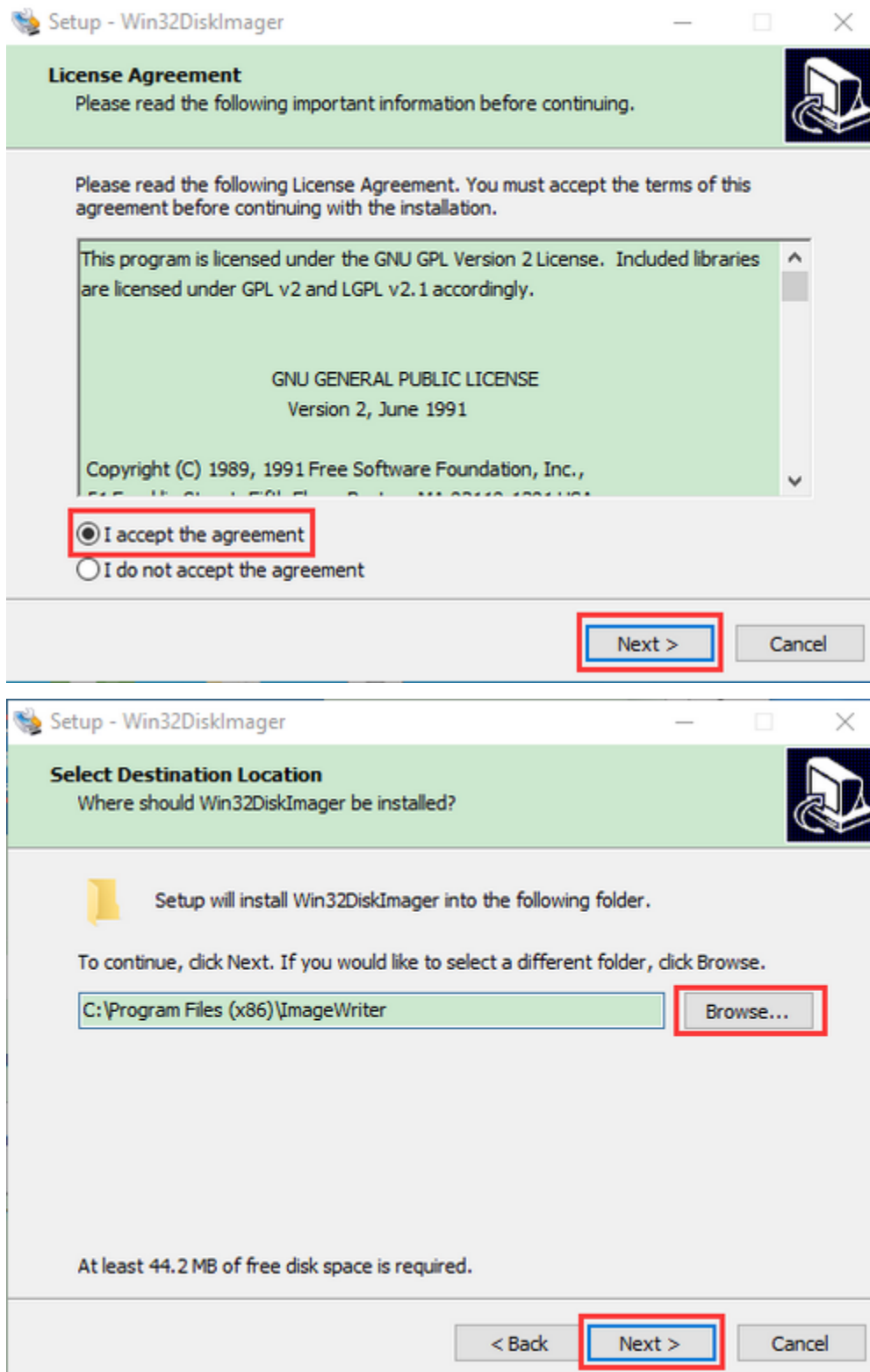
This release is for Windows 7/8.1/10. It will should also work on Windows Server 2008/2012/2016 (although not tested by the developers). For Windows XP/Vista, please use v0.9 (in the files archive).

After the download, double-click  `win32diskimager-1.0.0-install.exe` and tap “Run”.

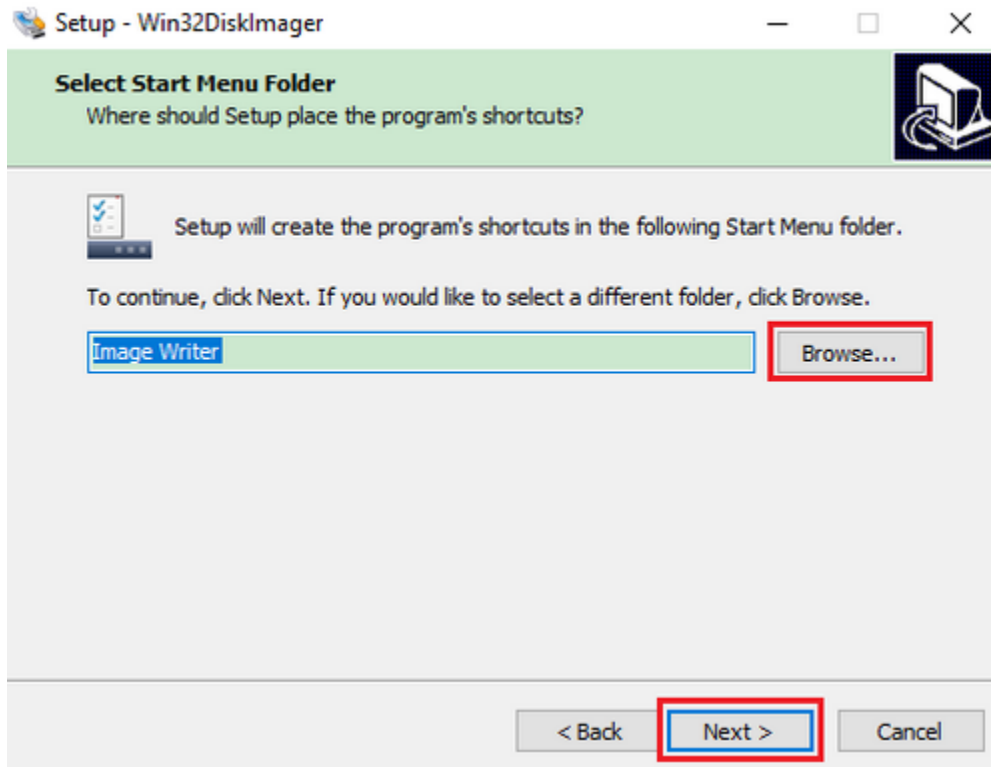




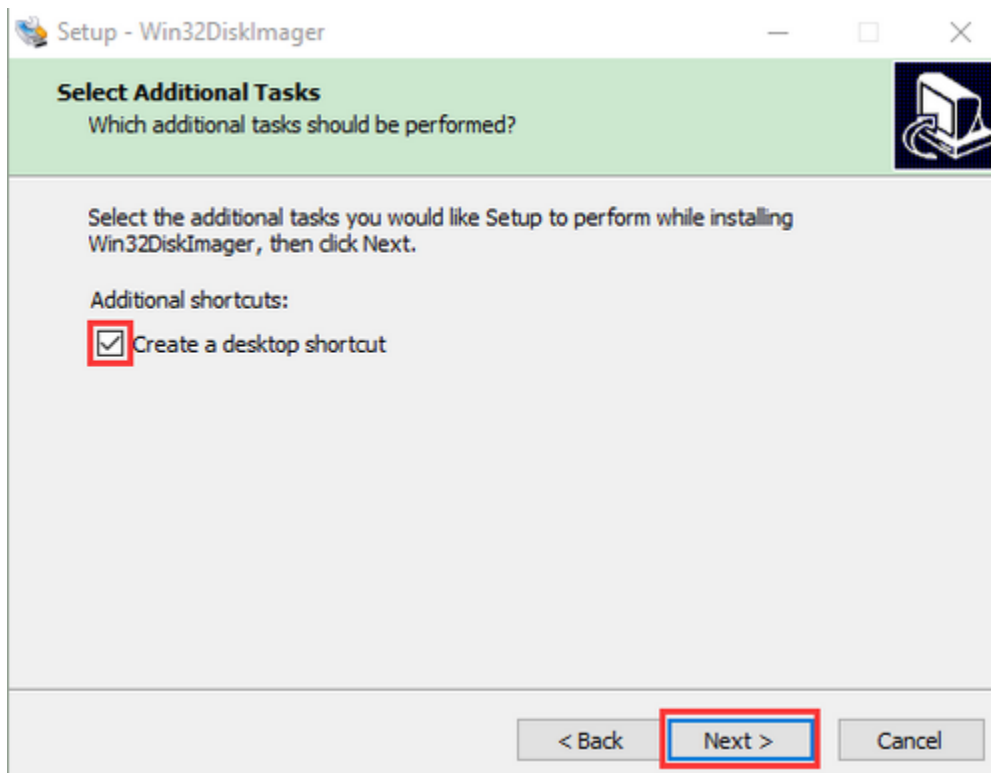
Select ☒ I accept the agreement and tap "Next".

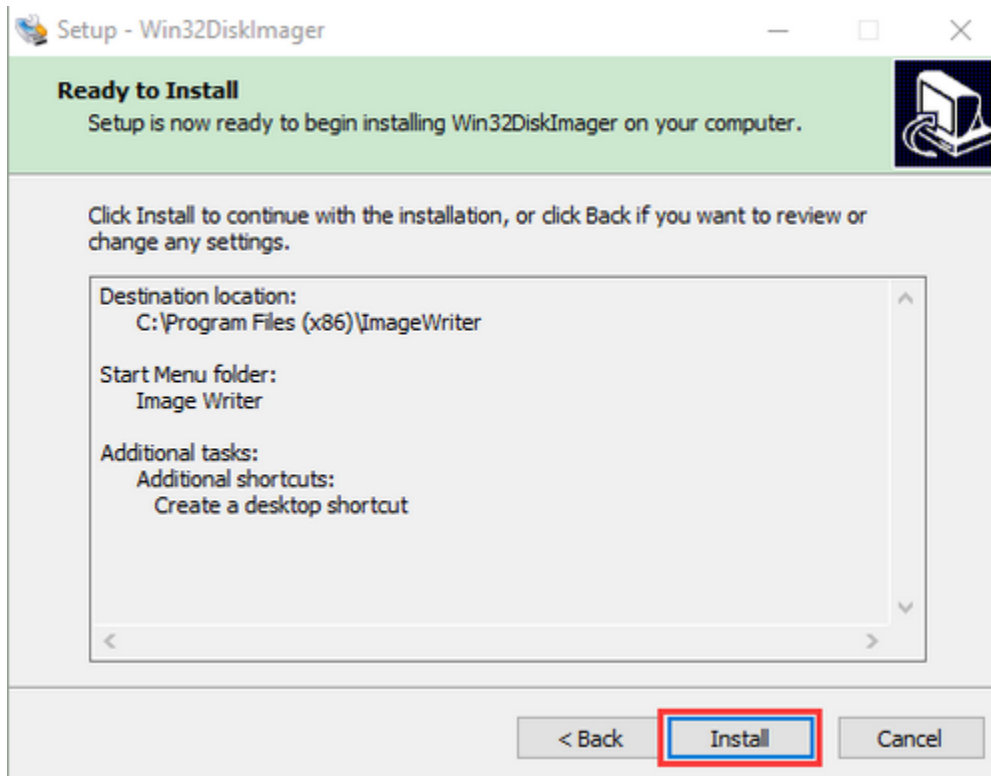


Click "Browse..." and find out the folder where the Win32DiskImager is located, tap "Next".

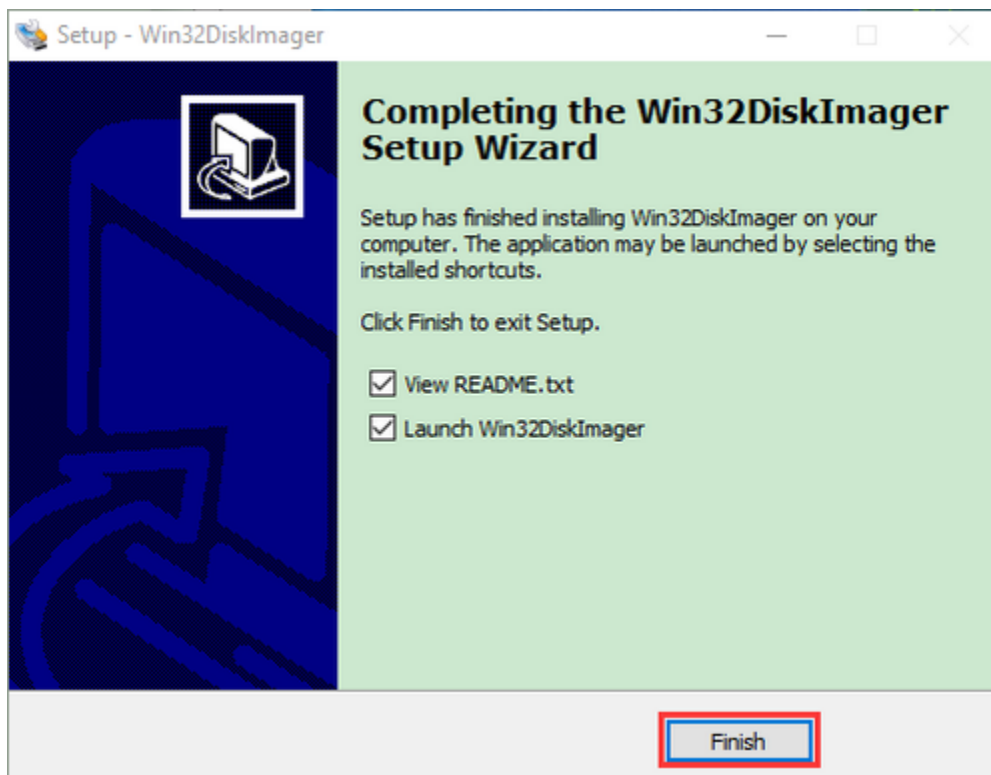


Tick **Create a desktop shortcut**, click "Next" and "Install".





After a few seconds, click“Finish”.



The installation is finished.

## (5) Scan to search ip address software tool—WNetWatcher

Download Link <http://www.nirsoft.net/utils/wnetwatcher.zip>

## (6) Raspberry Pi Imager

Download Address

<https://www.raspberrypi.org/downloads/raspberry-pi-os/>

(recommend downloading the version with desktop and commonly used software)

# Operating system images

Many operating systems are available for Raspberry Pi, including Raspberry Pi OS, our official supported operating system, and operating systems from other organisations.

[Raspberry Pi Imager](#) is the quick and easy way to install an operating system to a microSD card ready to use with your Raspberry Pi. Alternatively, choose from the operating systems below, available to download and install manually.

Download:  
[Raspberry Pi OS \(32-bit\)](#)  
[Raspberry Pi Desktop](#)  
[Third-Party operating systems](#)

## Raspberry Pi OS

Compatible with:  
[All Raspberry Pi models](#)



### Raspberry Pi OS with desktop and recommended software

Release date: December 2nd 2020  
Kernel version: 5.4  
Size: 2,949MB  
[Show SHA256 file integrity hash:](#)  
[Release notes](#)

[Download](#)

[Download torrent](#)

### Raspberry Pi OS with desktop

Release date: December 2nd 2020  
Kernel version: 5.4  
Size: 1,177MB  
[Show SHA256 file integrity hash:](#)  
[Release notes](#)

[Download](#)

[Download torrent](#)

### Raspberry Pi OS Lite

Release date: December 2nd 2020  
Kernel version: 5.4  
Size: 438MB  
[Show SHA256 file integrity hash:](#)  
[Release notes](#)

[Download](#)

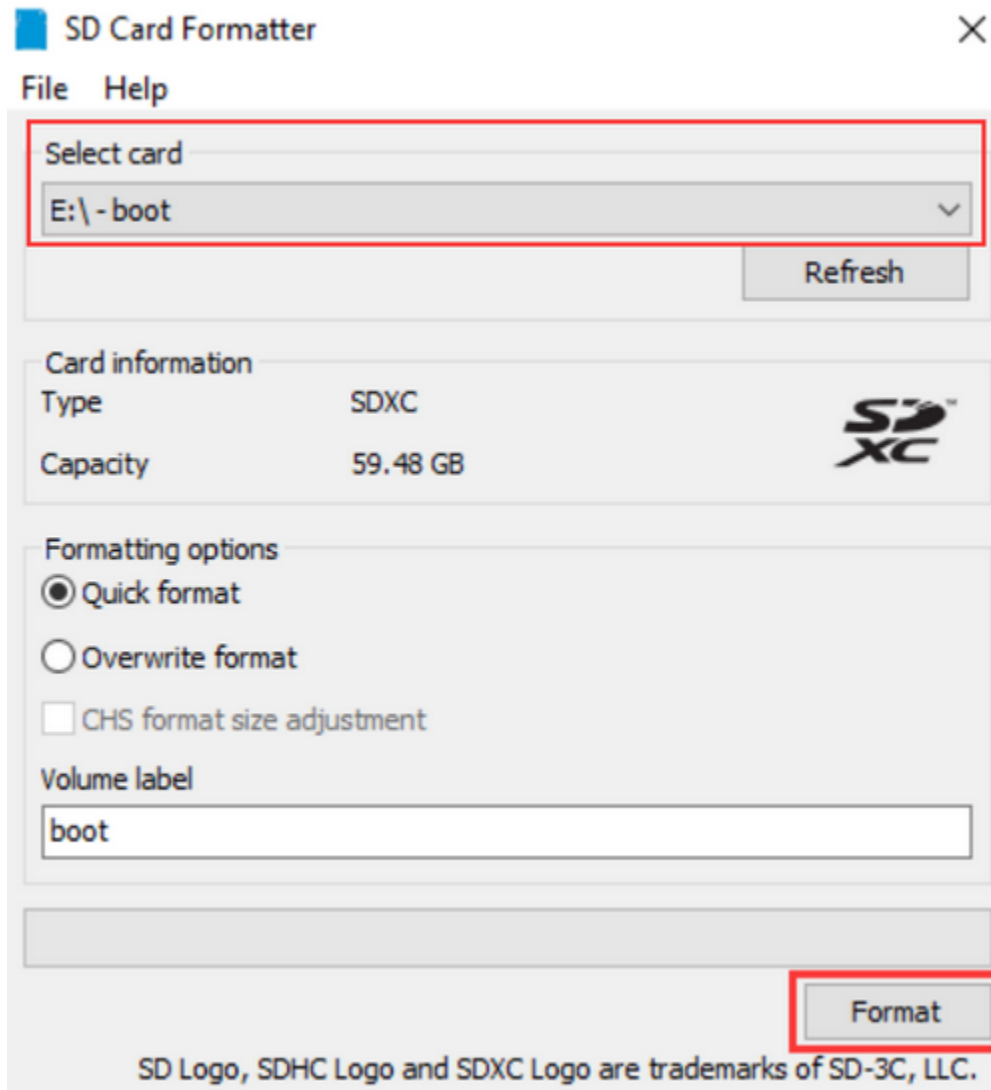
[Download torrent](#)

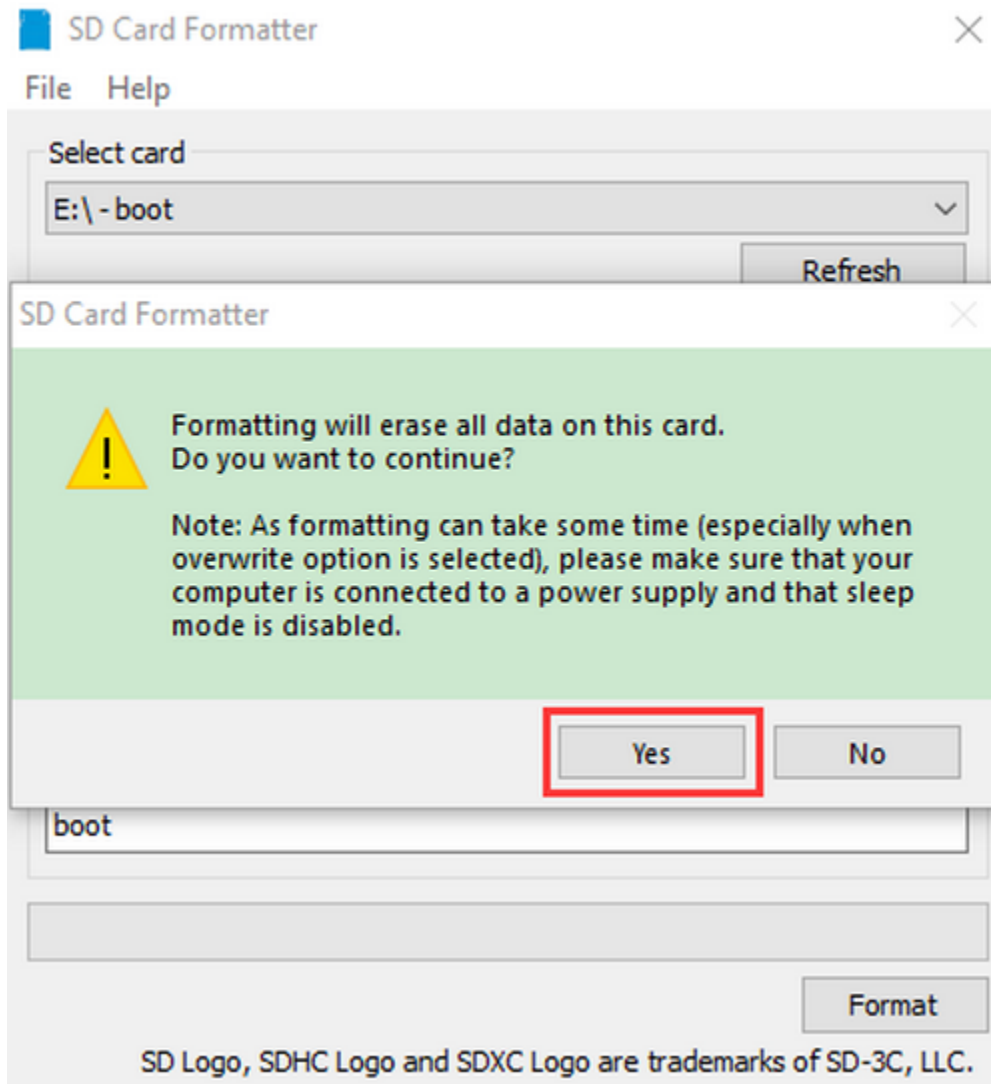
## 6.2 2.Install Raspberry Pi OS on Raspberry Pi 4B

Insert TFT RAM card to card reader, then interface card reader to USB port of computer.

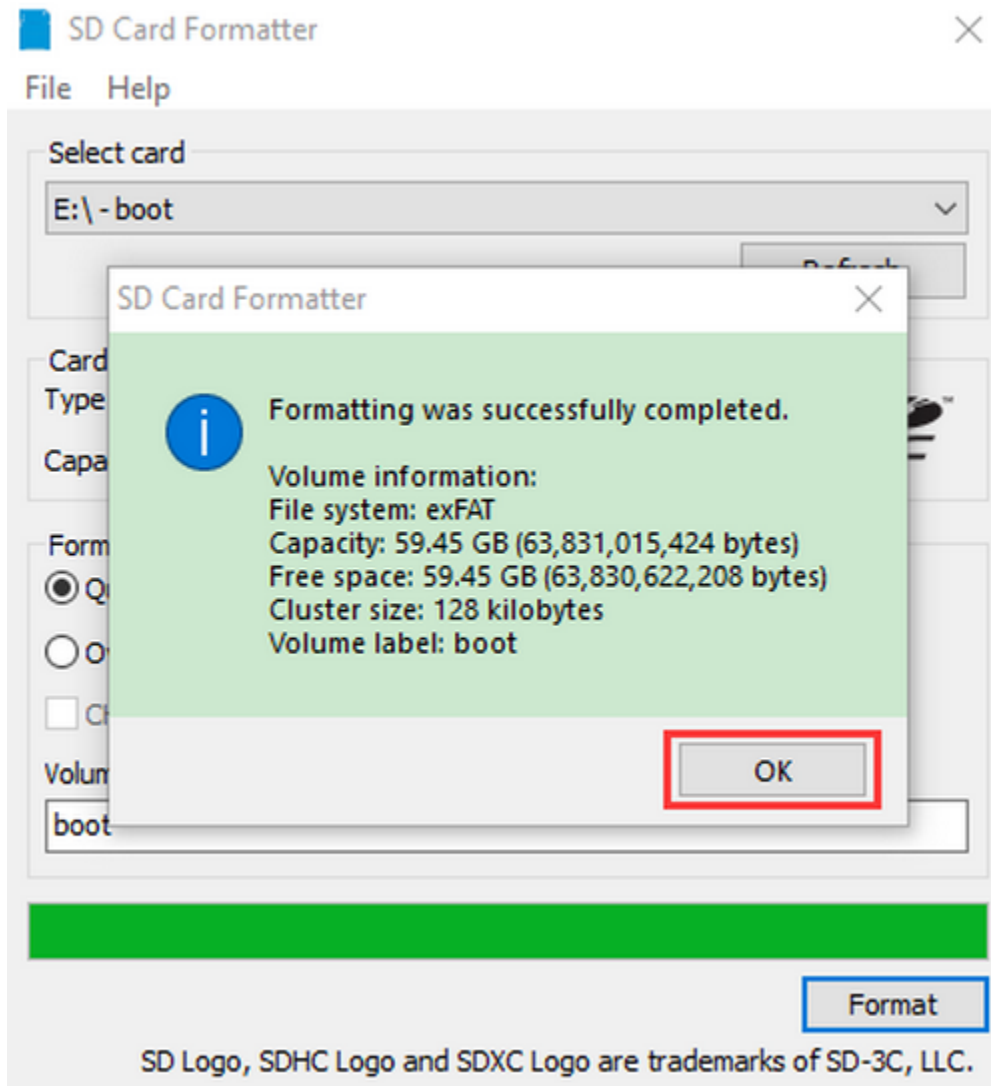


Format TFT RAM card with SD Card Formatter software, as shown below:



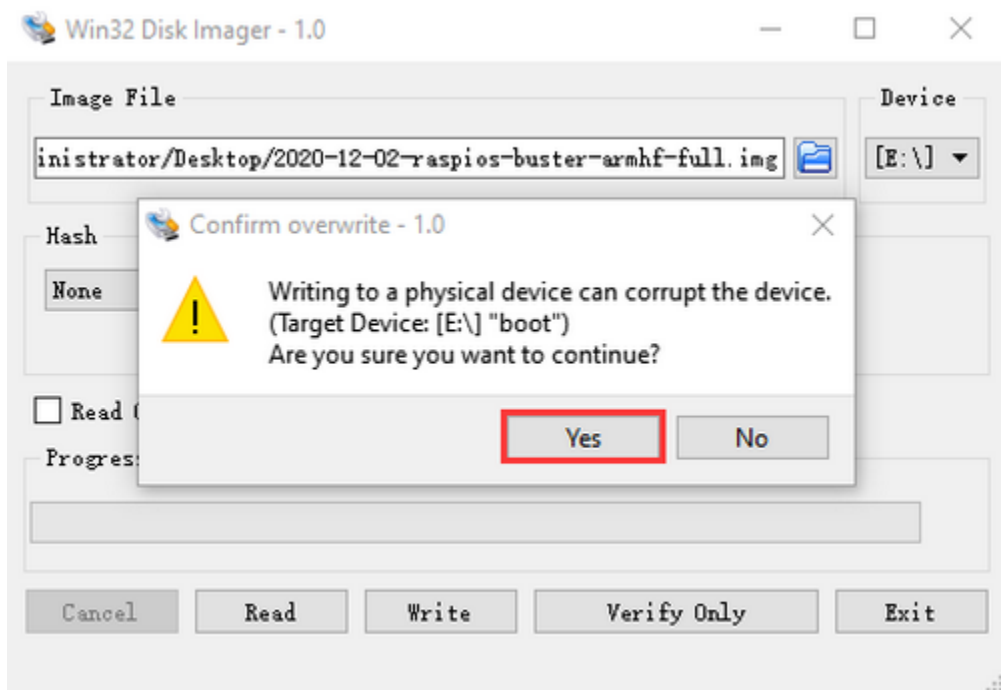
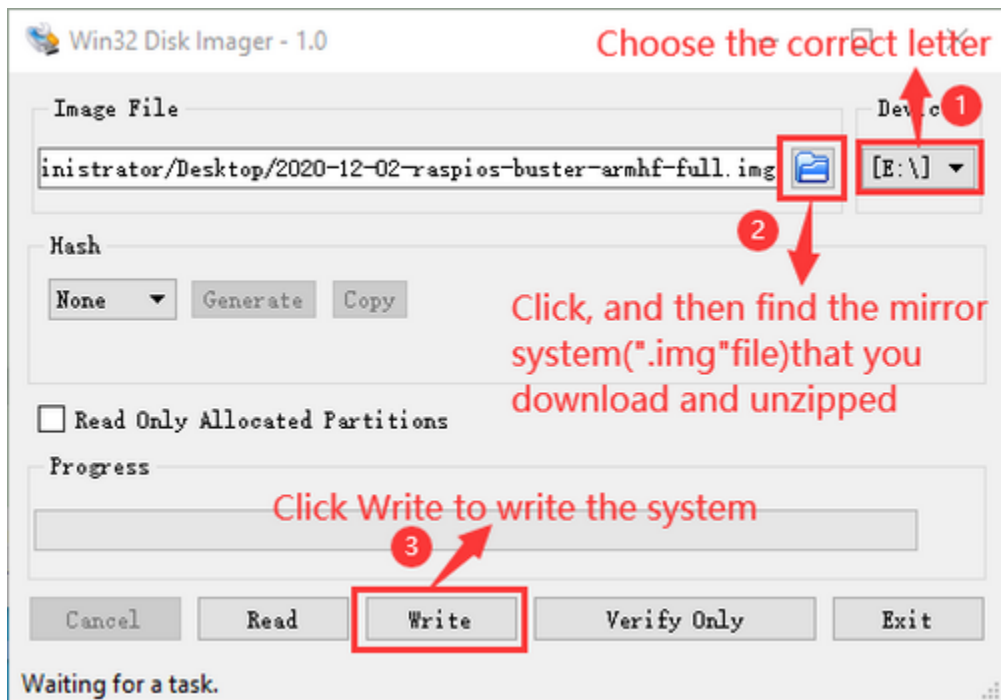


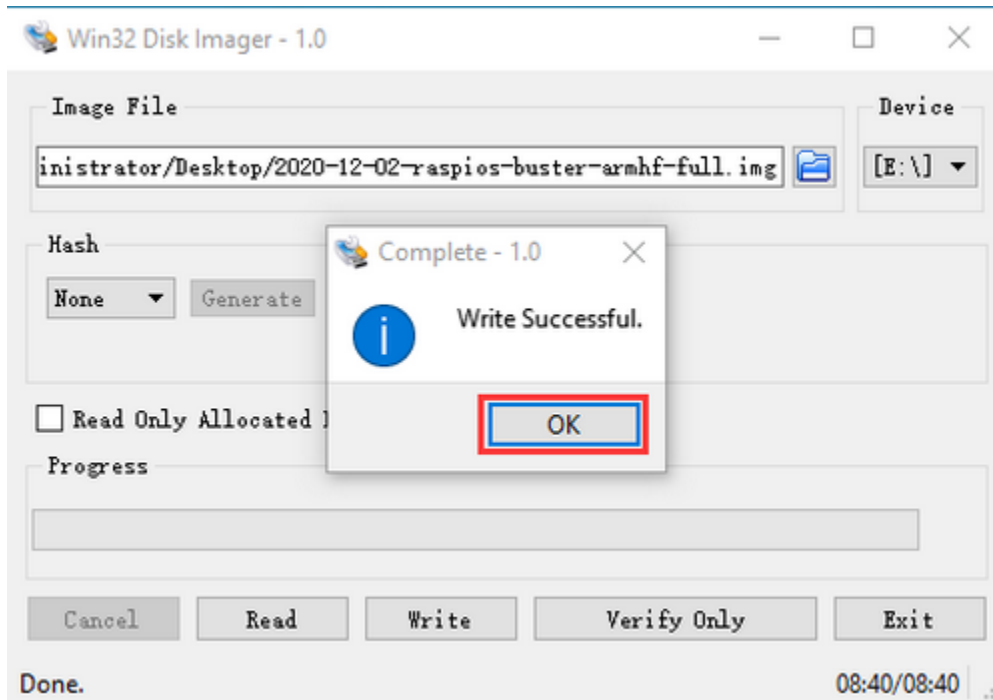




### 6.2.1 (1) Burn System

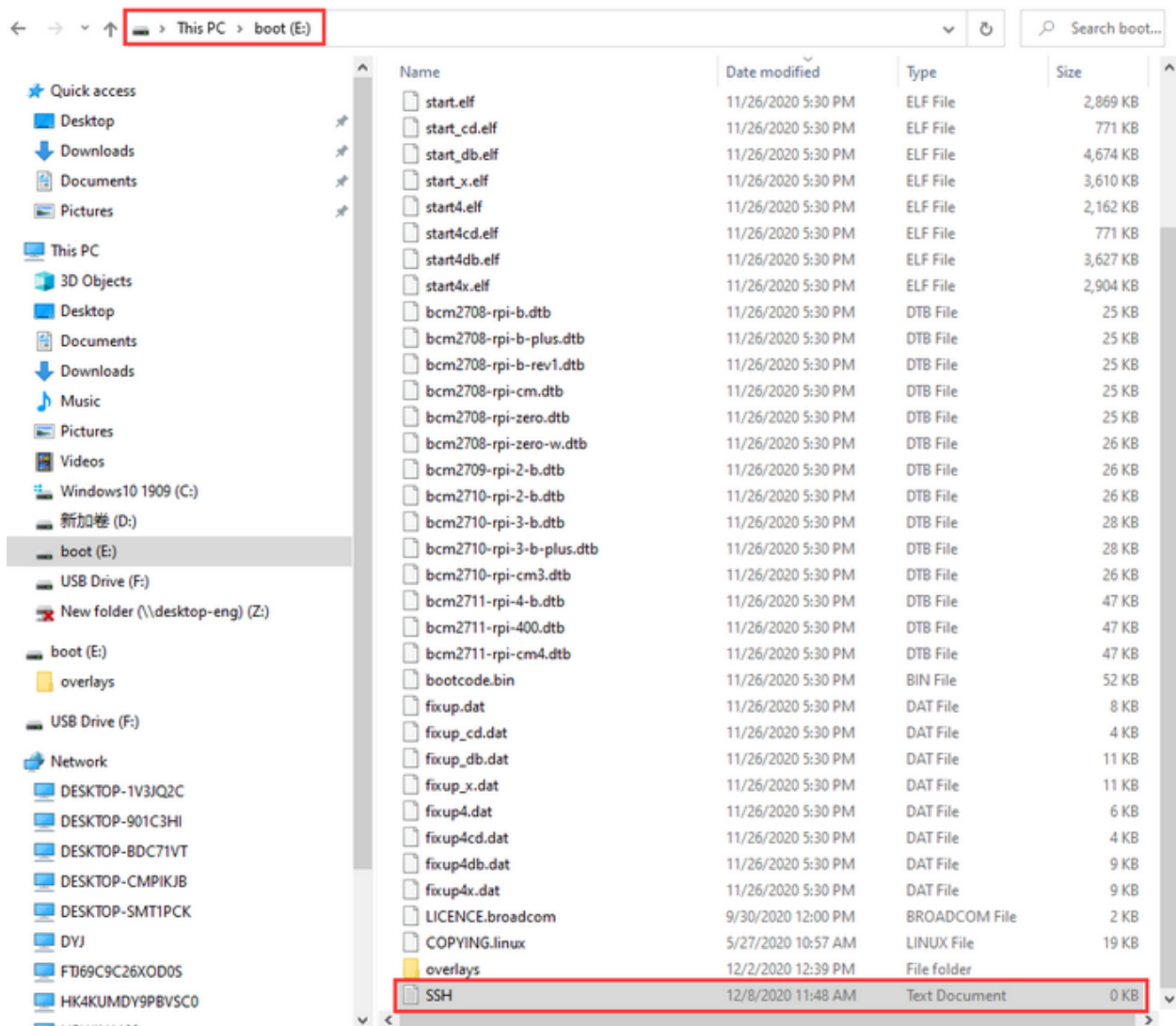
Burn the Raspberry Pi OS system to TFT card using Win32DiskImager software





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.

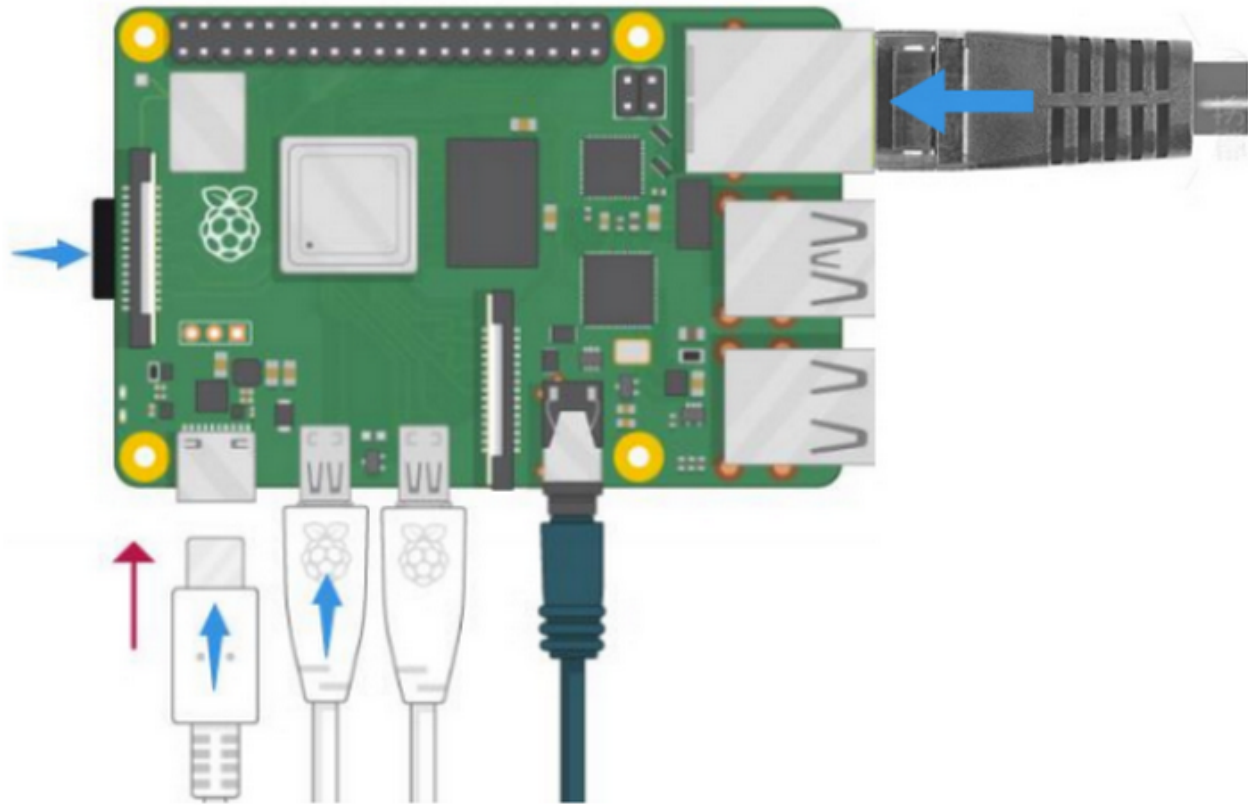


Eject Card Reader

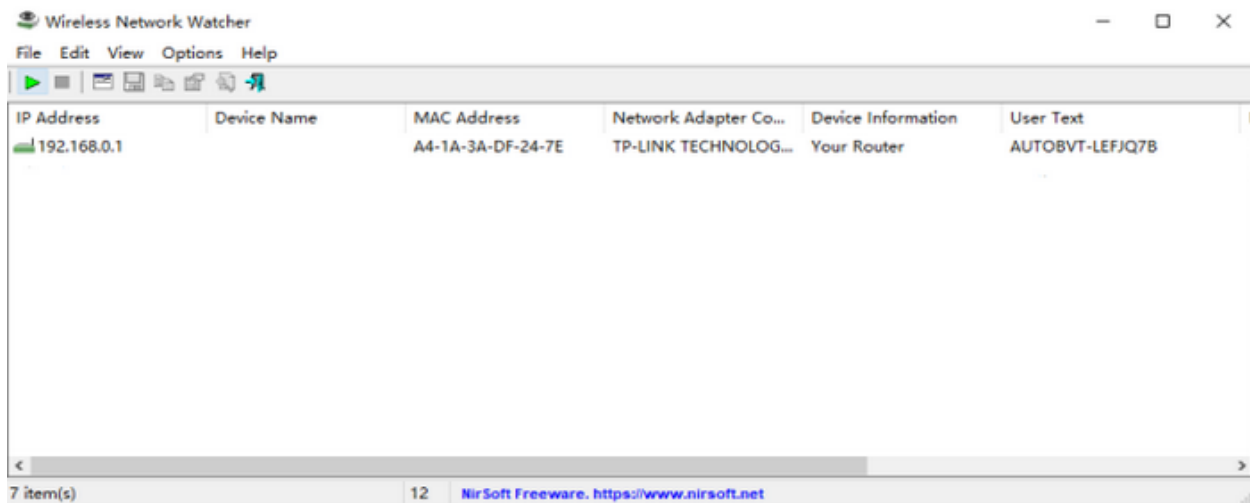
## 6.2.2 (2)Log in system

(Raspberry and PC should be in the same local area network.)

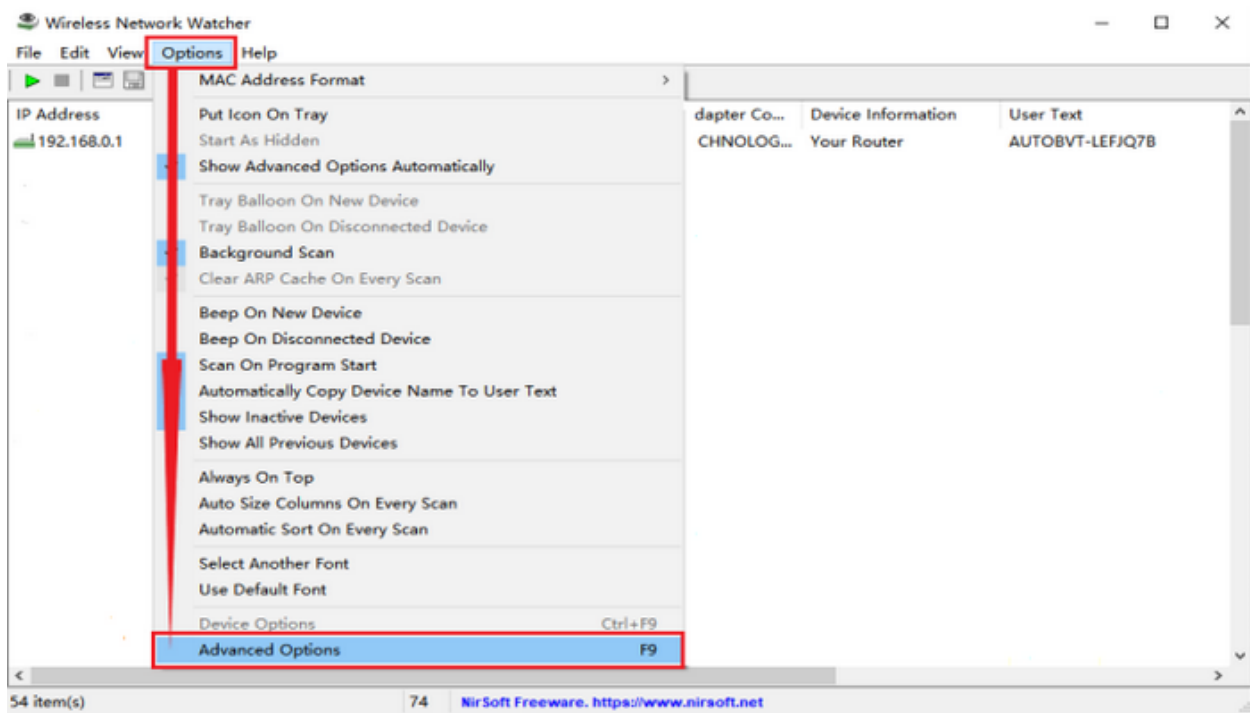
1.Insert TFT memory card into Raspberry Pi, connect internet cable and plug in power. If you have screen and HDMI cable of Raspberry Pi, you could view Raspberry Pi OS activating. If not, you can enter the desktop of Raspberry Pi via SSH remote login software—WinSCP and xrdp.

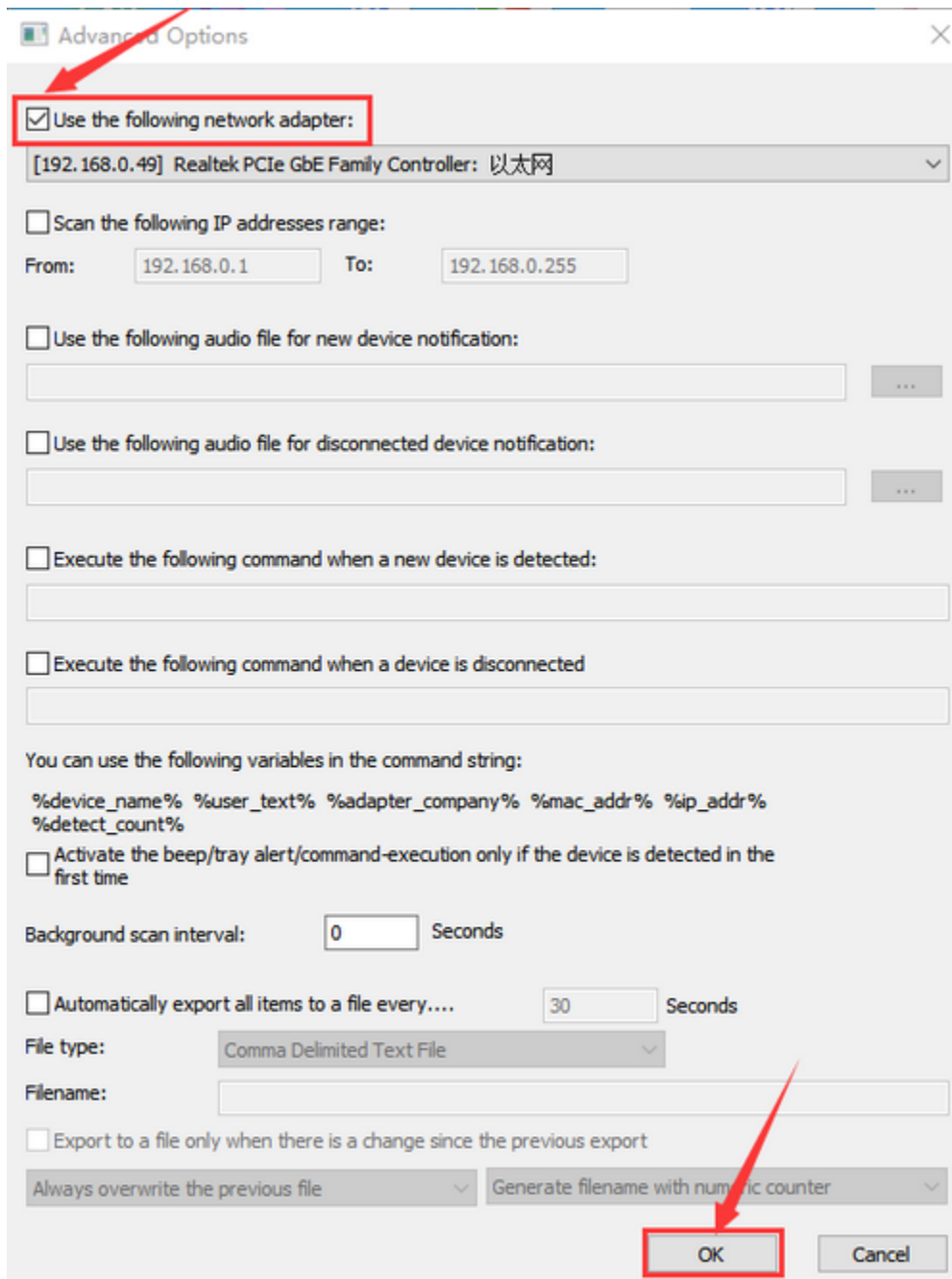


2. Use the WNetWatcher software to find the IP address of the Raspberry Pi.

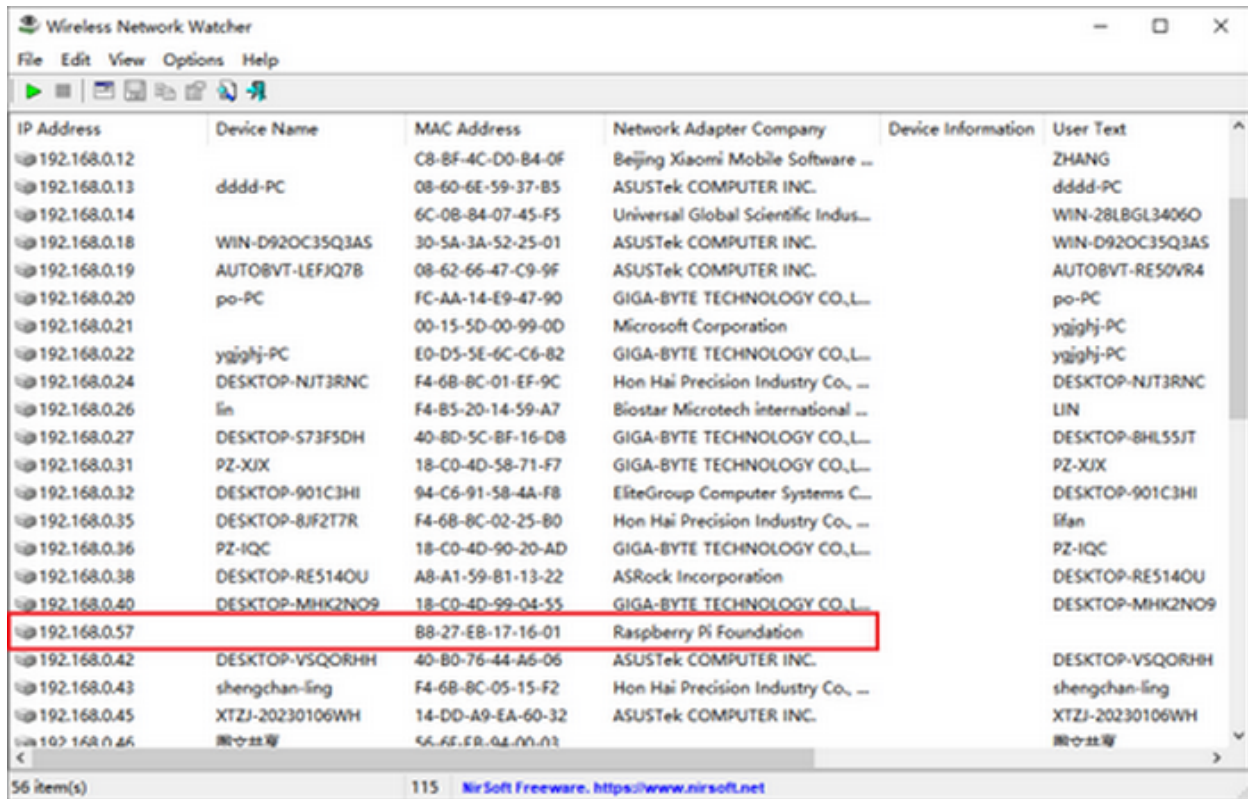


If there is no IP address as shown in the figure above, follow the following steps to set it.





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.



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-28LBGL3406O
192.168.0.18	WIN-D92OC35Q3AS	30-5A-3A-52-25-01	ASUSTek COMPUTER INC.		WIN-D92OC35Q3AS
192.168.0.19	AUTOBVT-LEFJQ7B	08-62-66-47-C9-9F	ASUSTek COMPUTER INC.		AUTOBVT-RES0VR4
192.168.0.20	po-PC	FC-AA-14-E9-47-9D	GIGA-BYTE TECHNOLOGY CO.,L...		po-PC
192.168.0.21		00-15-5D-00-99-0D	Microsoft Corporation		ygqghj-PC
192.168.0.22	ygqghj-PC	E0-D5-5E-6C-C6-B2	GIGA-BYTE TECHNOLOGY CO.,L...		ygqghj-PC
192.168.0.24	DESKTOP-NJT3RNC	F4-6B-8C-01-EF-9C	Hon Hai Precision Industry Co., ...		DESKTOP-NJT3RNC
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-BHL55JT
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-901C3HI
192.168.0.35	DESKTOP-8JF2T7R	F4-6B-8C-02-25-B0	Hon Hai Precision Industry Co., ...		lilan
192.168.0.36	PZ-IQC	18-C0-4D-90-20-AD	GIGA-BYTE TECHNOLOGY CO.,L...		PZ-IQC
192.168.0.38	DESKTOP-RES14OU	A8-A1-59-B1-13-22	ASRock Incorporation		DESKTOP-RES14OU
192.168.0.40	DESKTOP-MHK2NO9	18-C0-4D-99-04-55	GIGA-BYTE TECHNOLOGY CO.,L...		DESKTOP-MHK2NO9
192.168.0.57		B8-27-EB-17-16-01	Raspberry Pi Foundation		
192.168.0.42	DESKTOP-VSQQRHH	40-B0-76-44-A6-06	ASUSTek COMPUTER INC.		DESKTOP-VSQQRHH
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-20230106WH

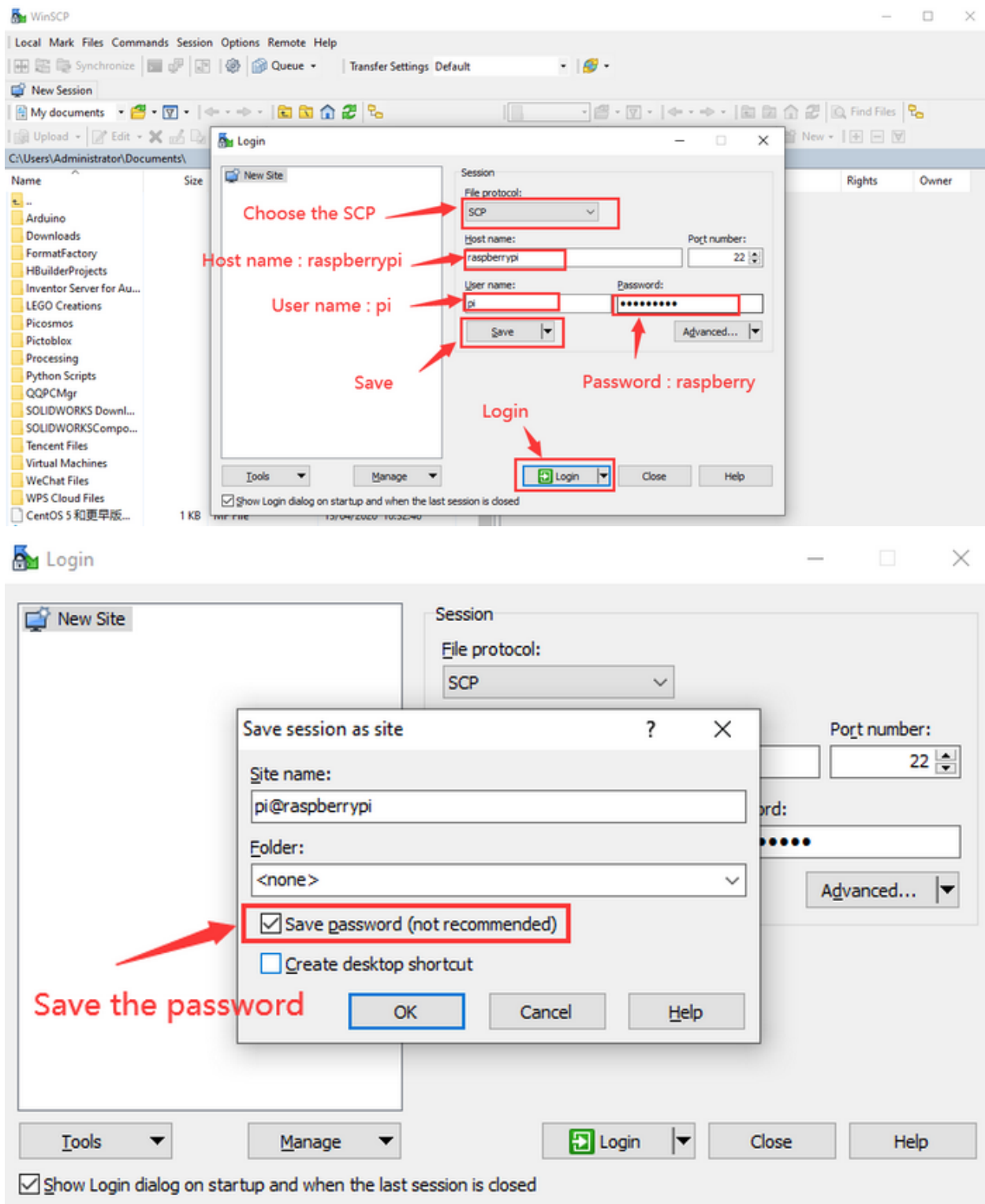
If you do not know the mac address and the ip address of the Raspberry PI, then unplug the network cable of the Raspberry PI first, open the **WNetWatcher** query, and the detection times will be displayed on the right side of the interface. Connect the Raspberry PI cable and query it once using WNetWatcher, and the Raspberry PI address is detected one less time than the other addresses. Then write down the ip and mac addresses.

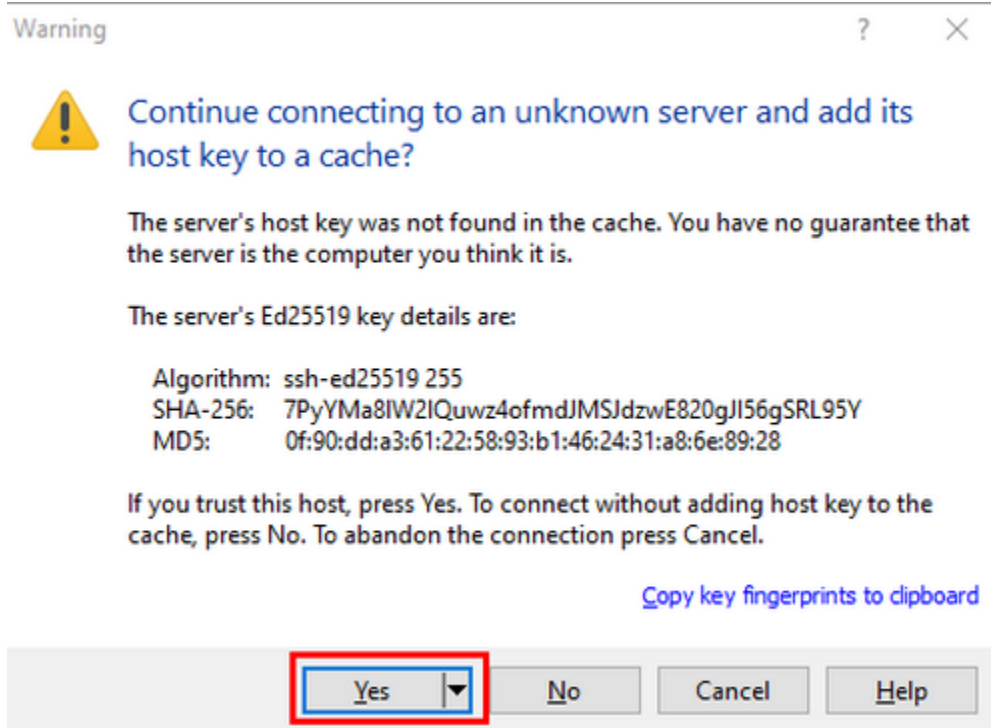
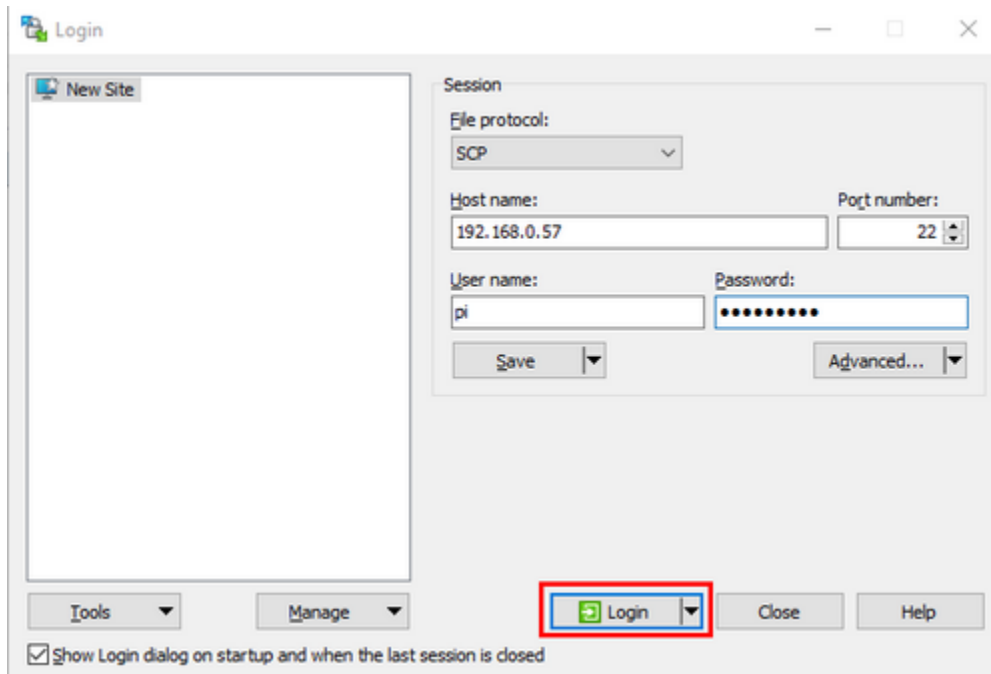
### 6.2.3 (3) Remote Login

Enter default user name, password and host name on WinSCP to log in.

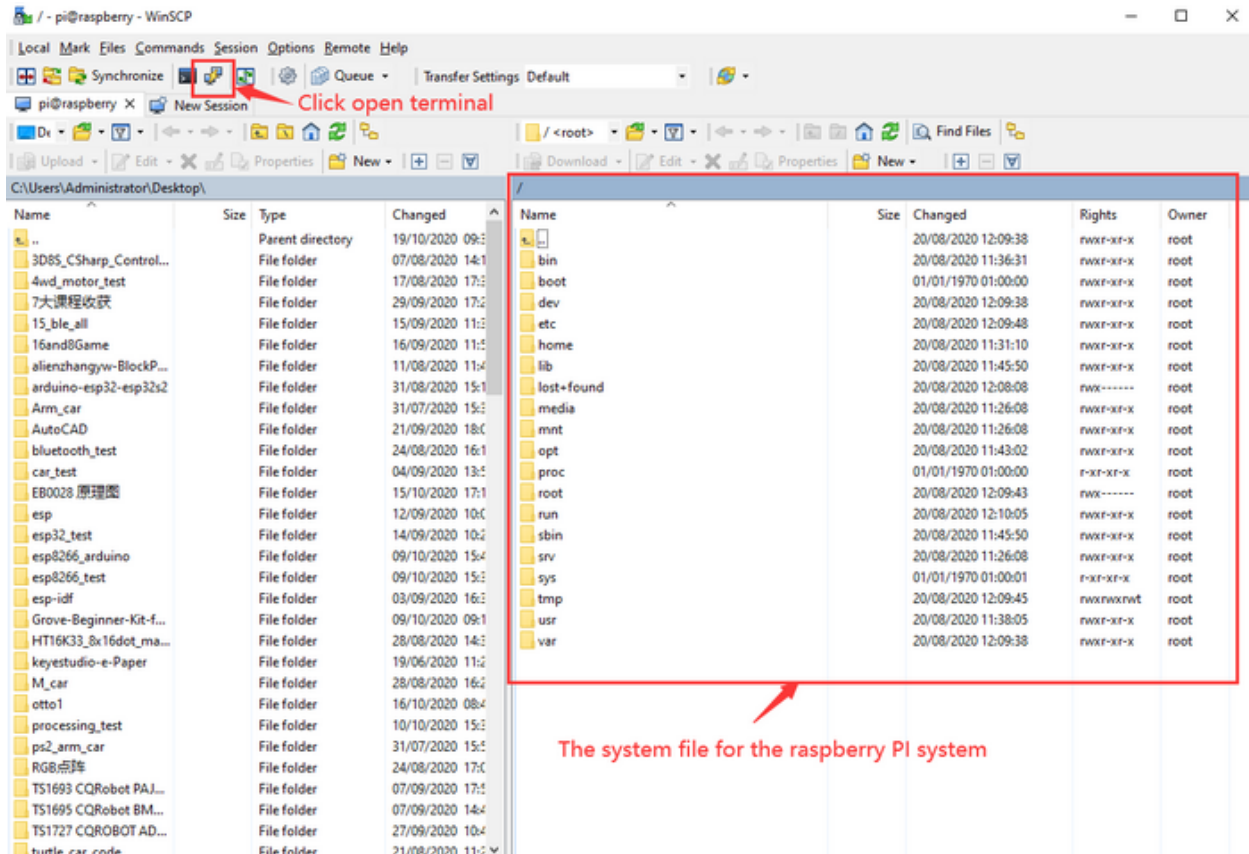
The same network only receives one Raspberry Pi.



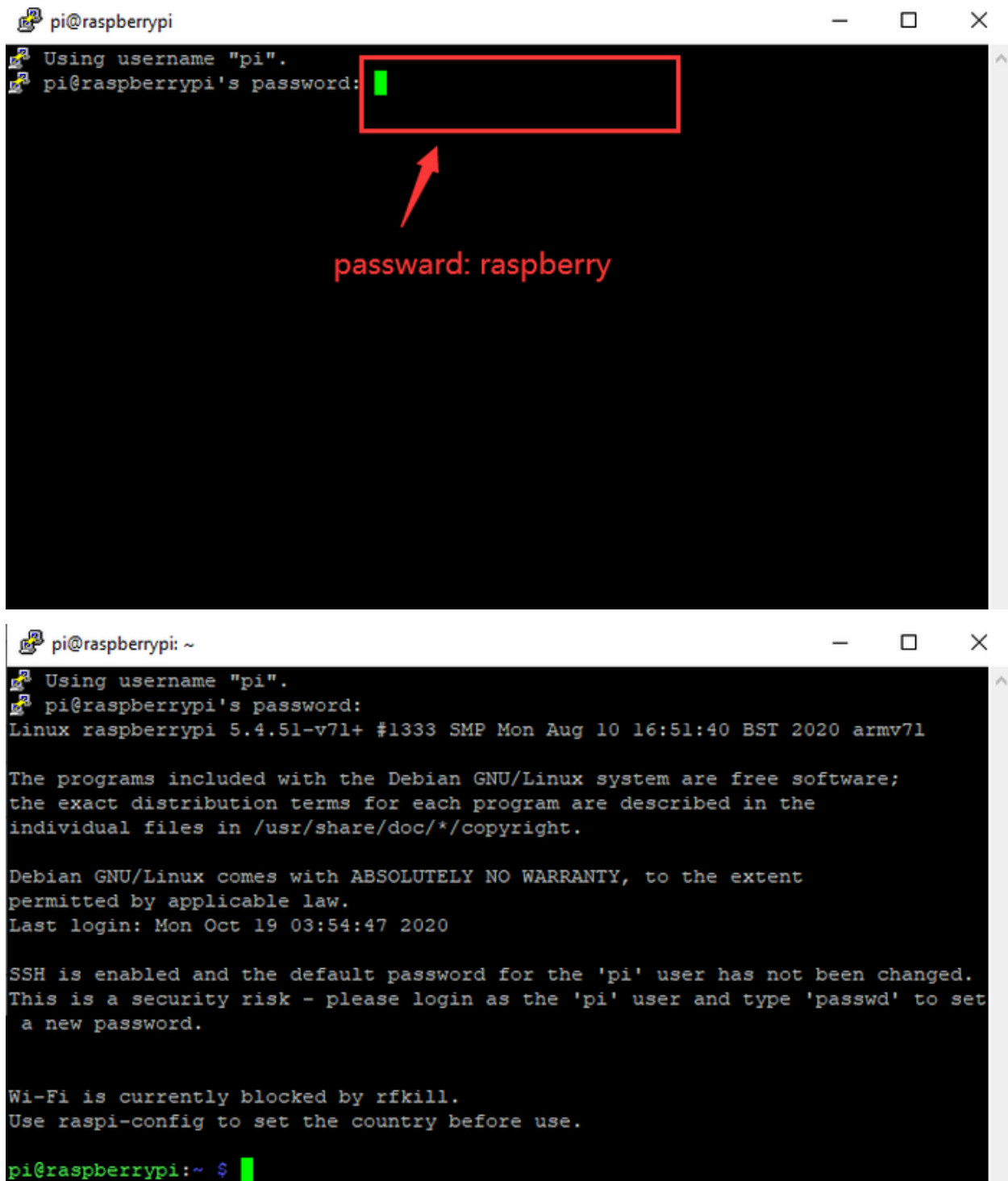




## 6.2.4 (4) Check ip and mac address



Click to open terminal input the password raspberry, and press “Enter” on keyboard.



```
pi@raspberrypi
Using username "pi".
pi@raspberrypi's password: 
password: raspberrypi

pi@raspberrypi: ~
Using username "pi".
pi@raspberrypi's password:
Linux raspberrypi 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv7l

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: ~
Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~ $ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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 default qlen 1000
    link/ether b8:27:eb:17:16:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.57 /24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 1357sec preferred_lft 1132sec
    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 qlen 1000
    link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff
pi@raspberrypi:~ $

```

## 6.2.5 (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 password

Input password in the terminal `sudo 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 address `address 192.168.0.57`

Copy the above new address to terminal and press "Enter".

Configuration File

```

echo -e '
auto eth0
iface eth0 inet static

```

(continues on next page)

(continued from previous page)

```
#Change 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 0
mtu 1492

N>/etc/network/interfaces.d/eth0
```

As shown below:

```
pi@raspberrypi:~$ su root
Password:
root@raspberrypi:/home/pi# echo -e '
> auto eth0
> iface eth0 inet static
>     #Change 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 0
>     mtu 1492
> '>/etc/network/interfaces.d/eth0
root@raspberrypi:/home/pi#
```

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: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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 1492 qdisc mq state UP group default qlen 1000
    link/ether b8:27:eb:17:16:01 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 qlen 1000
    link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff
pi@raspberrypi:~ $

```

### 6.2.6 (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:

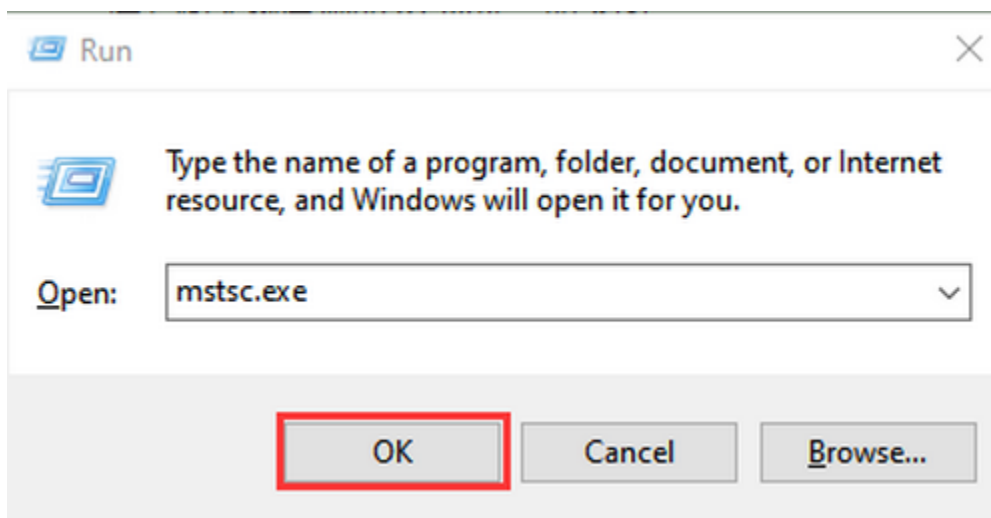


```
pi@raspberrypi: ~  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
  
SSH is enabled and the default password for the 'pi' user has not been changed.  
This is a security risk - please login as the 'pi' user and type 'passwd' to set  
a new password.  
  
pi@raspberrypi:~ $ sudo apt-get install xrdp  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  ssl-cert  
Suggested packages:  
  openssl-blacklist guacamole xrdp-pulseaudio-installer  
The following NEW packages will be installed:  
  ssl-cert xrdp  
0 upgraded, 2 newly installed, 0 to remove and 373 not upgraded.  
Need to get 415 kB of archives.  
After this operation, 2,722 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

### 6.2.7 (7) Open the remote desktop connection on Windows

Press **WIN+R** on keyboard and enter mstsc.exe .

As shown below

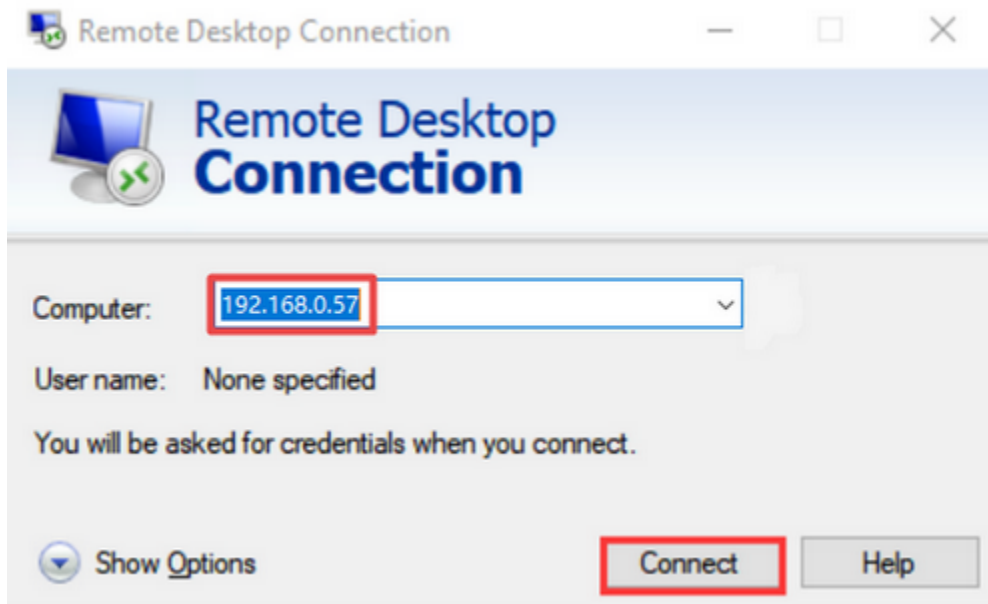


Input ip address of Raspberry Pi, as shown below.

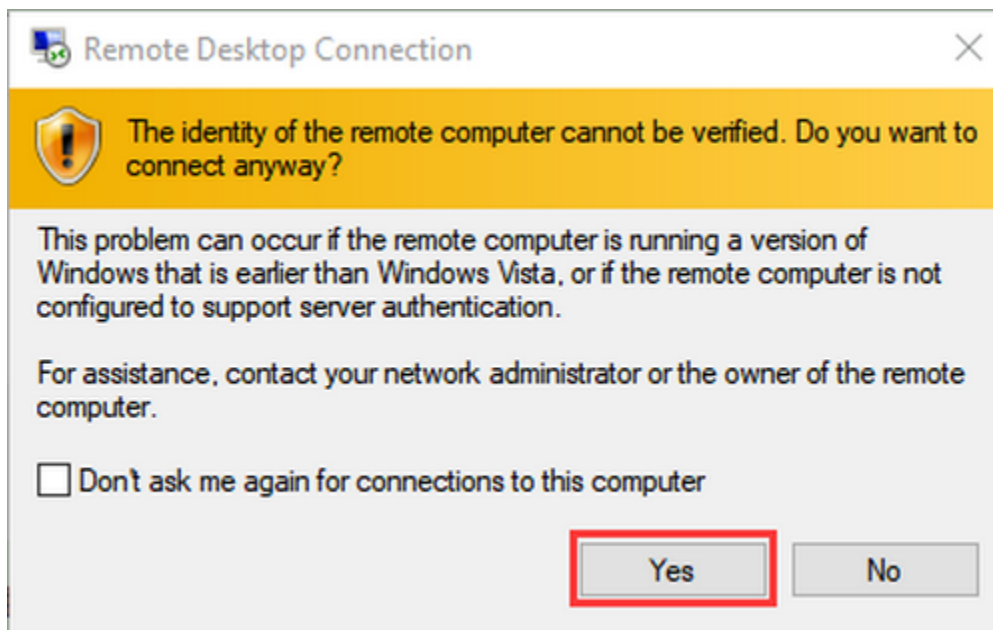
Click“**Connect**”and tap“**Connect**”.

192.168.0.57 is ip address we use, you could change into yours ip address.

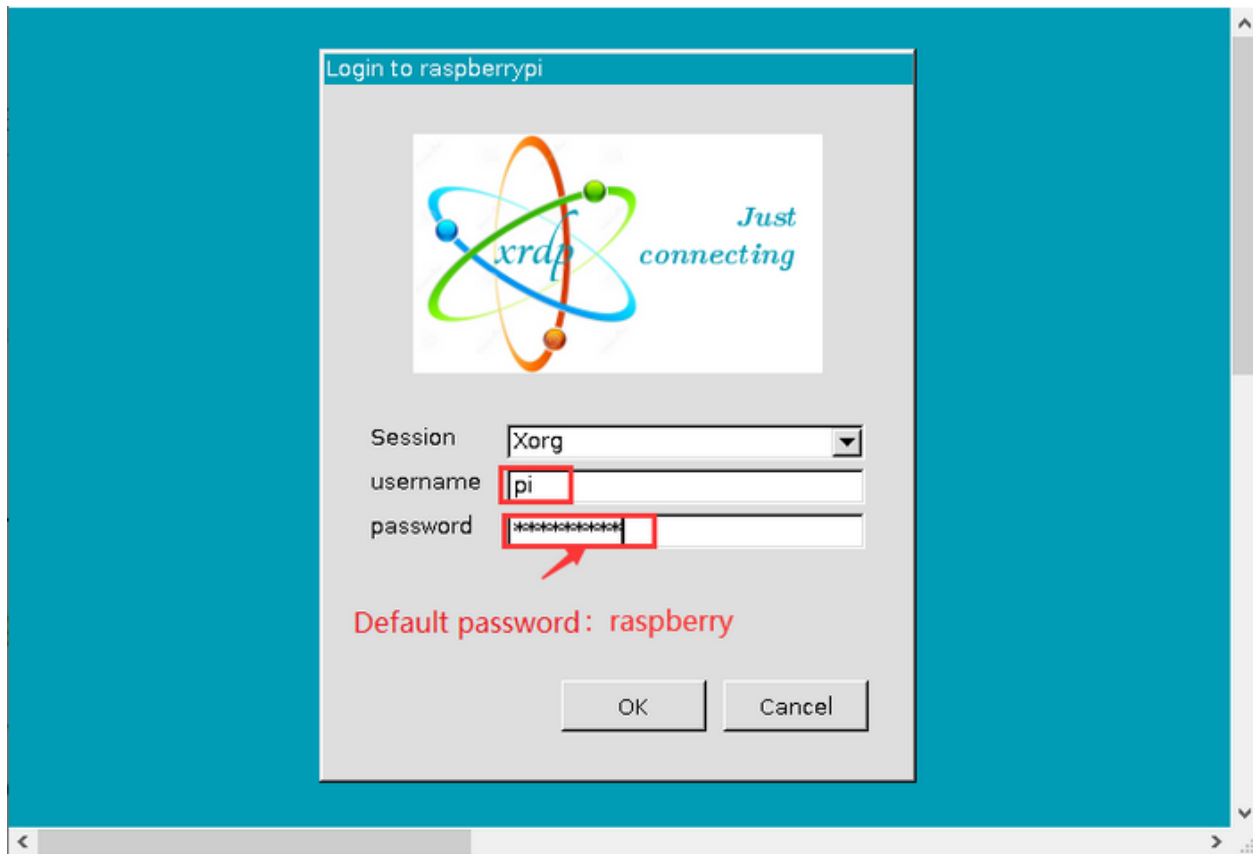




Click "Yes".



Input user name: pi, default password: raspberry, as shown below:



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.

## 6.3 3. Preparations for Python

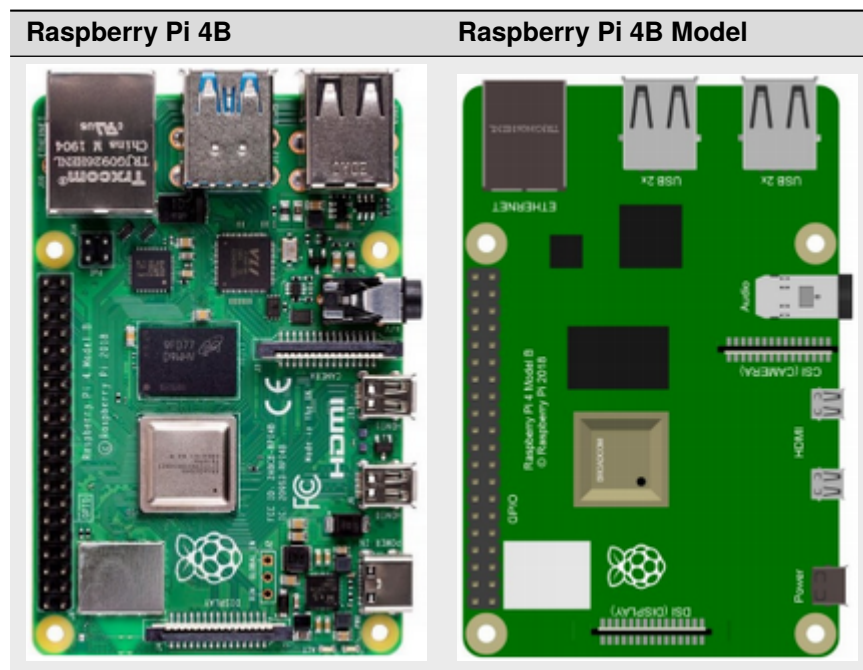
Python is a programming language that lets you work more quickly and integrate your systems more effectively.

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

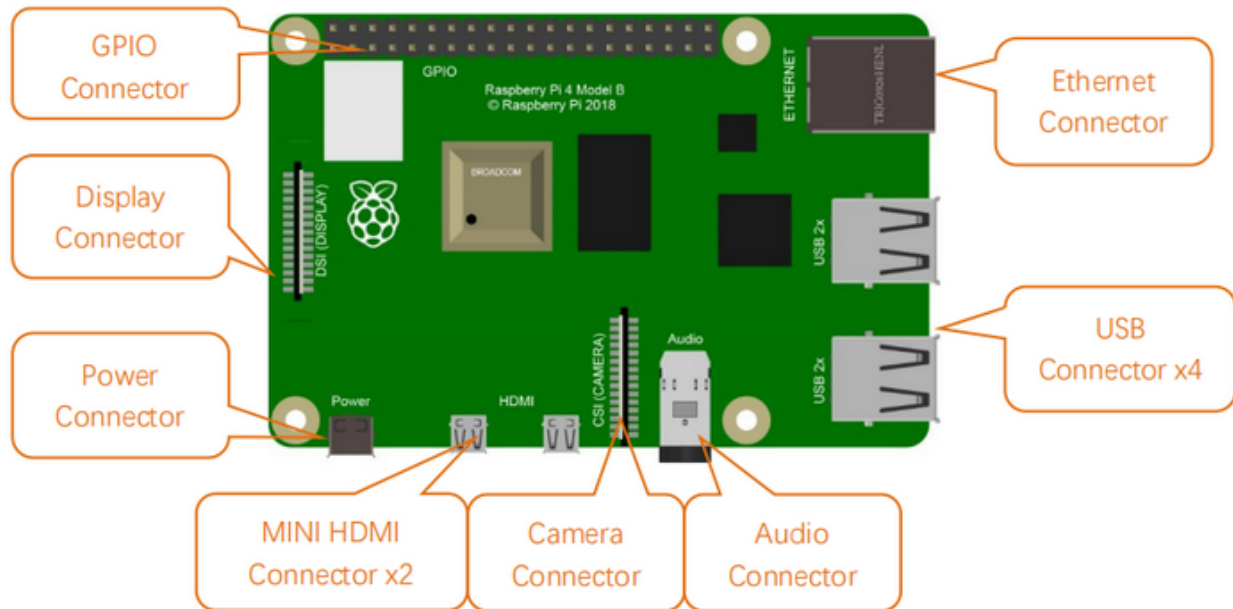
Next to pick up Python to control 40 pin of Raspberry Pi.

### 6.3.1 3.1Hardware

#### Raspberry Pi 4B



## Hardware Interfaces

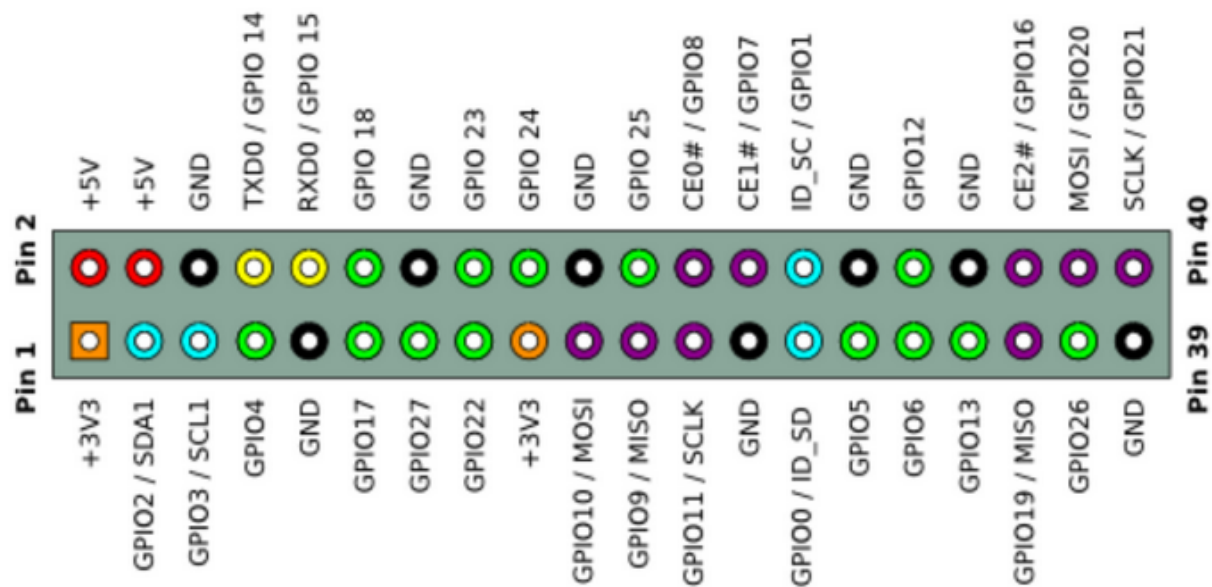


## 40-Pin GPIO Header Description

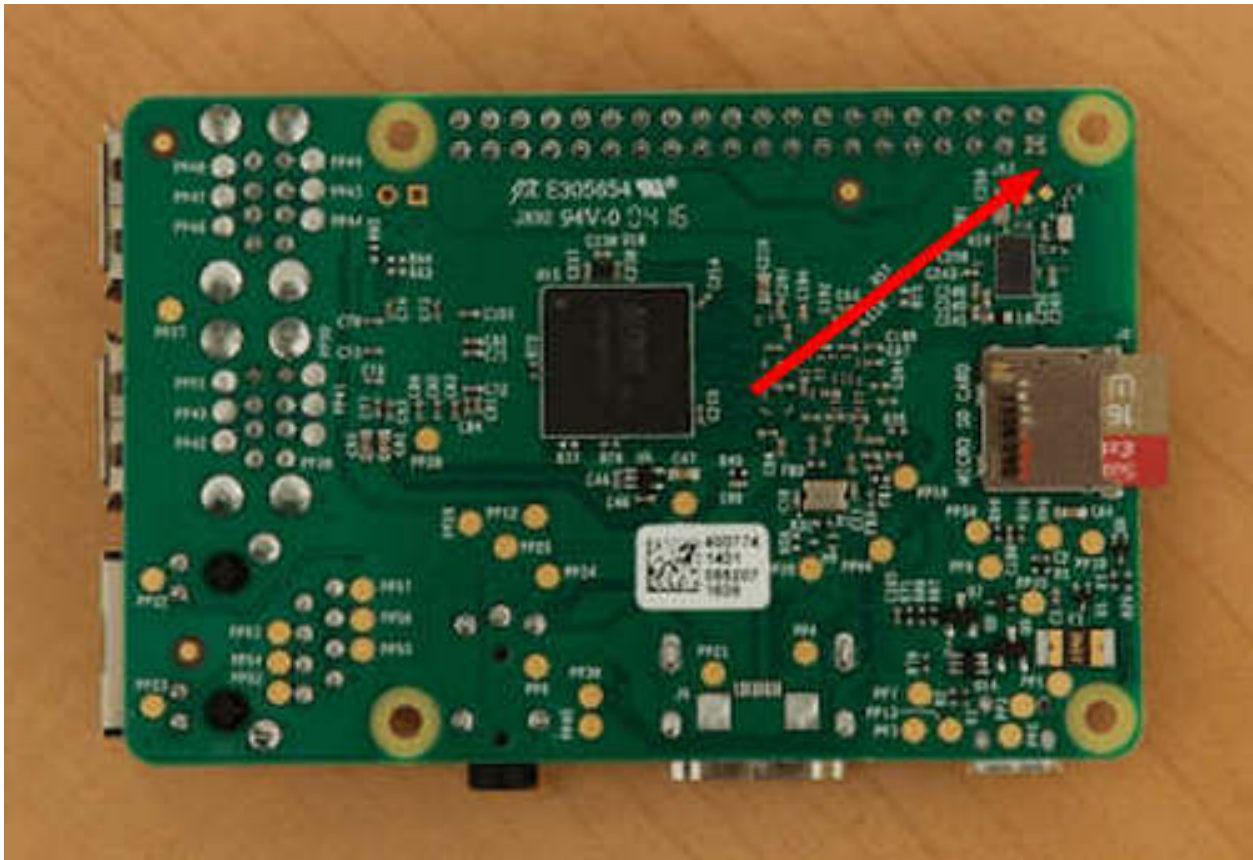
GPIO pins are divided into BCM GPIO number, physics number and WiringPi GPIO number.

We usually use WiringPi GPIO when using C language and BCM GPIO and physics number are used to Python, as shown below;

In these lessons, we use Python, so BCM GPIO number is adopted.



Note: pin(3.3V) on the left hand is square, but other pins are round. Turn Raspberry Pi over, there is a square GPIO on the back.(you could tell from pin(3.3V)).

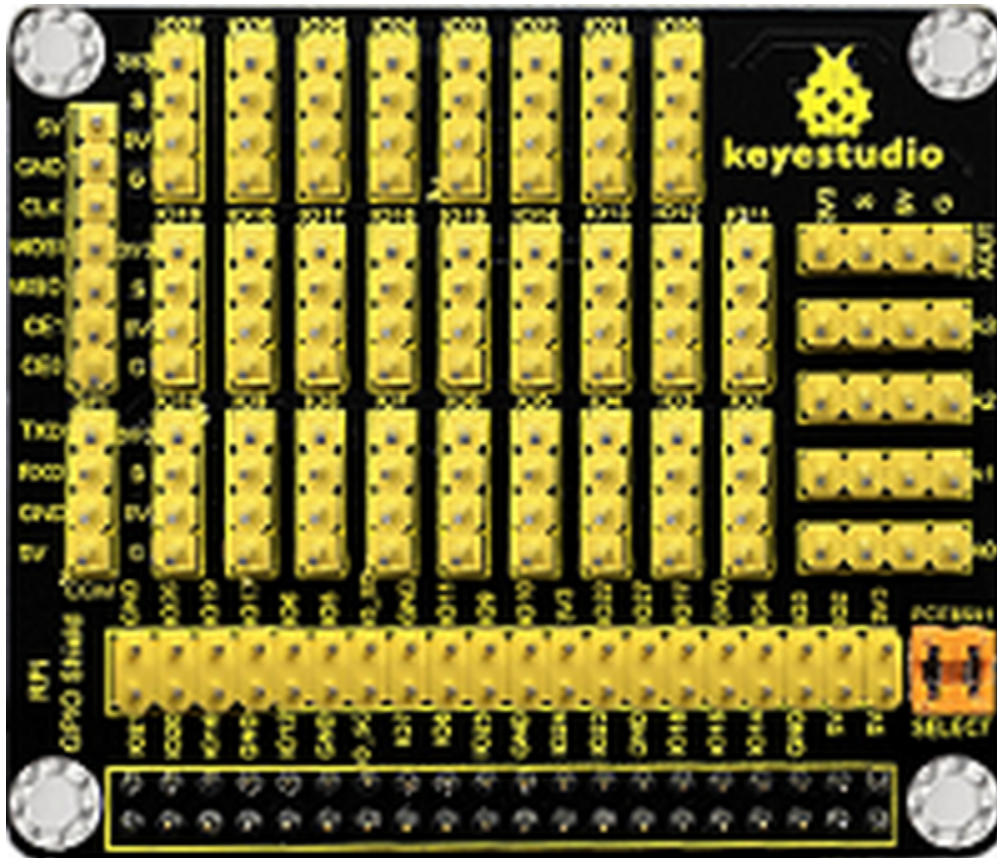


Note: the largest current of each pin on Raspberry Pi 4B is 16mA and the aggregate current of all pins is not less than 51mA.

### RPI GPIO-PCF8591 Shield

This shield extend 40 pins of Raspberry Pi, which can connect a number of sensors and modules.

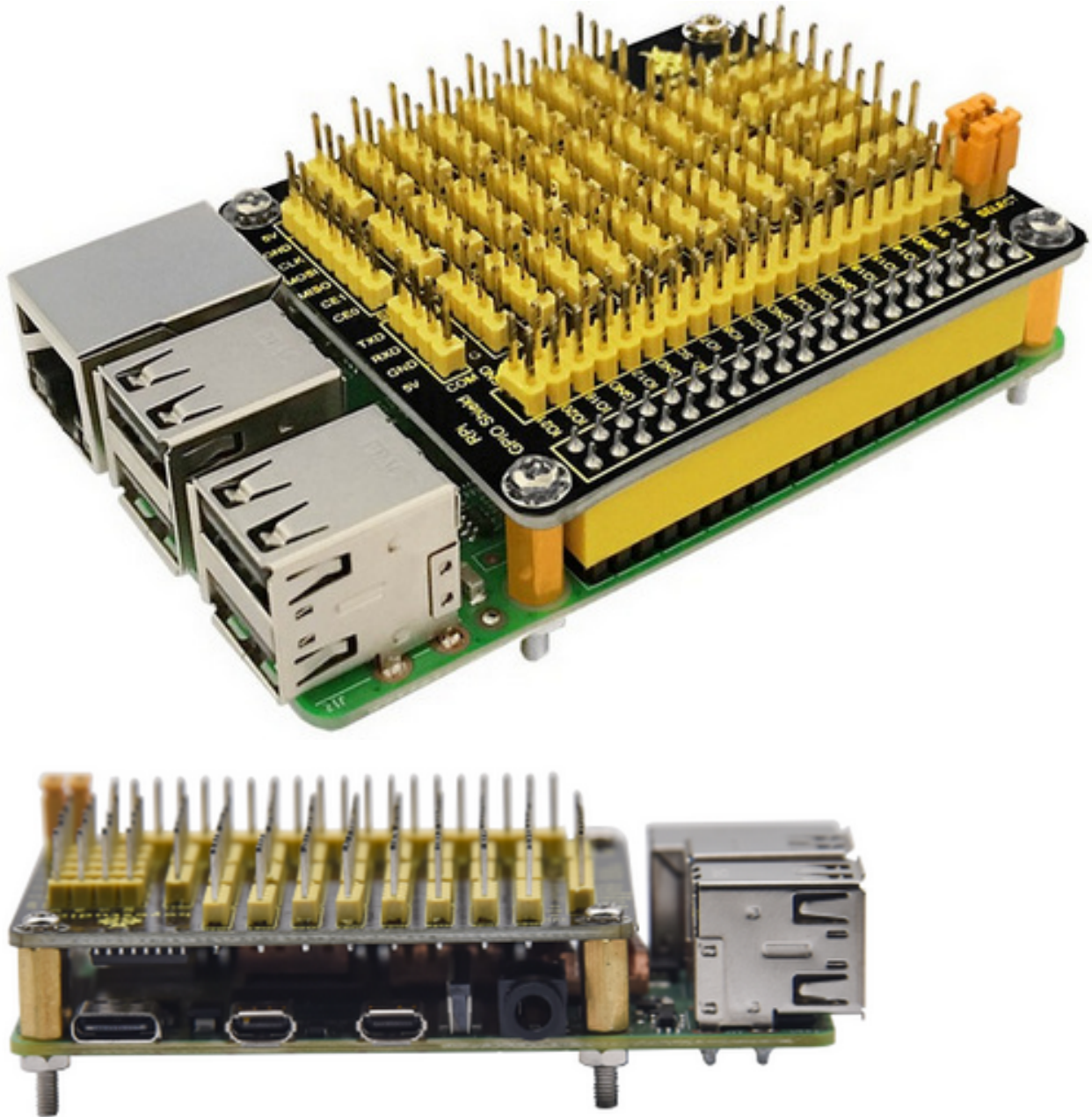




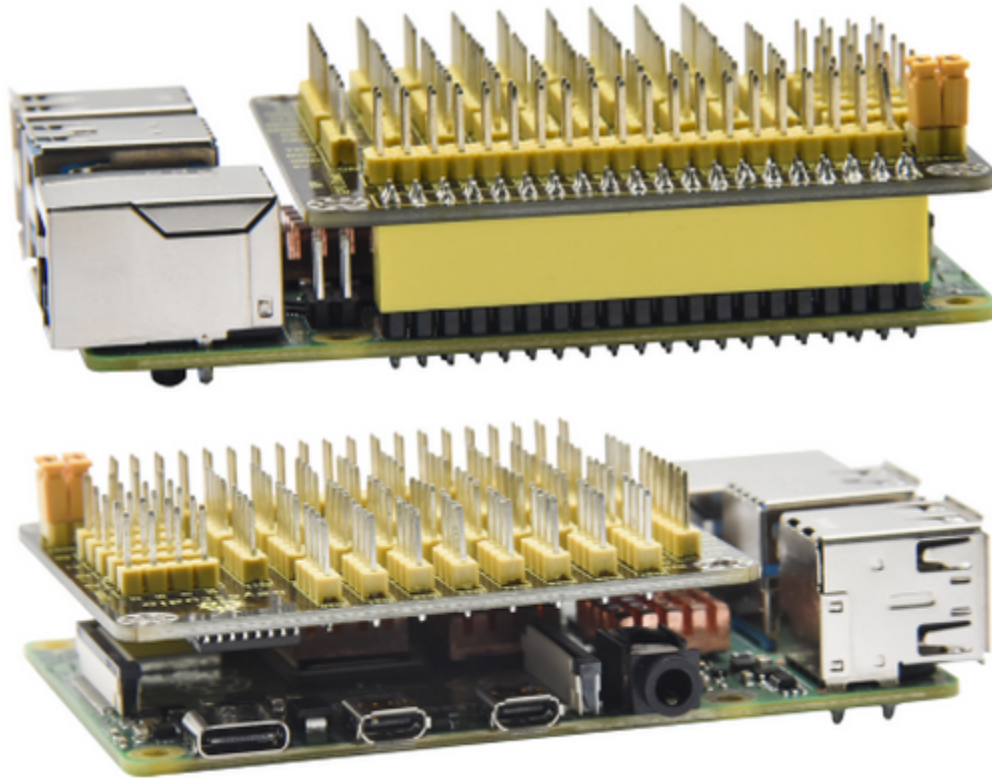
The Raspberry Pi doesn't have an AD / DA function. If it has to be interfaced with a shield with AD / DA function when connected to an analog sensor. The RPI GPIO-PCF8591 shield has a PCF8591 chip which can be applied to 4-channel AD and 1-channel DA of I2C port.

The connection methods of Raspberry Pi and RPI GPIO-PCF8591 shield are shown below:

Fix them with screws, nuts and copper pillars



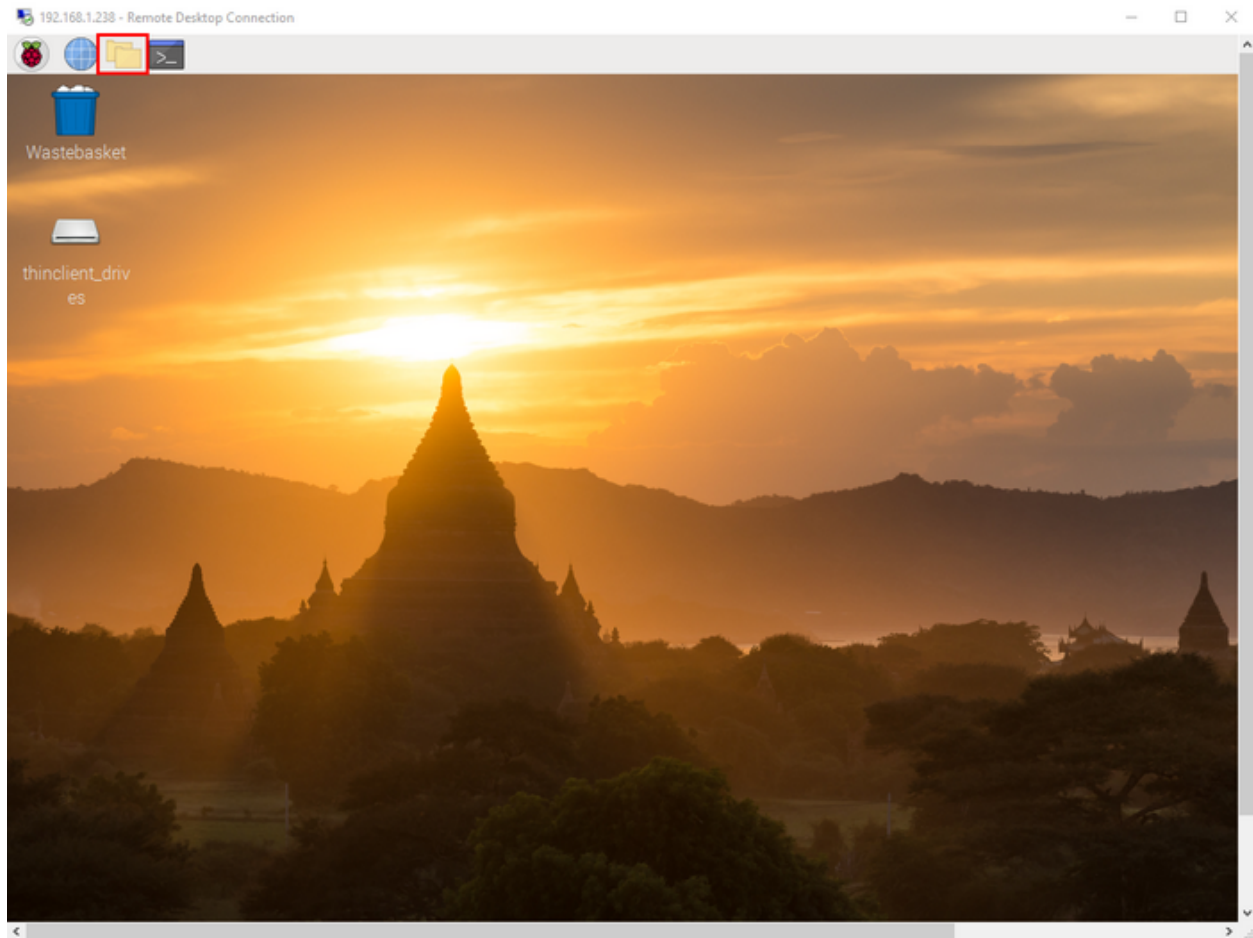
Without screws, nuts and copper pillars

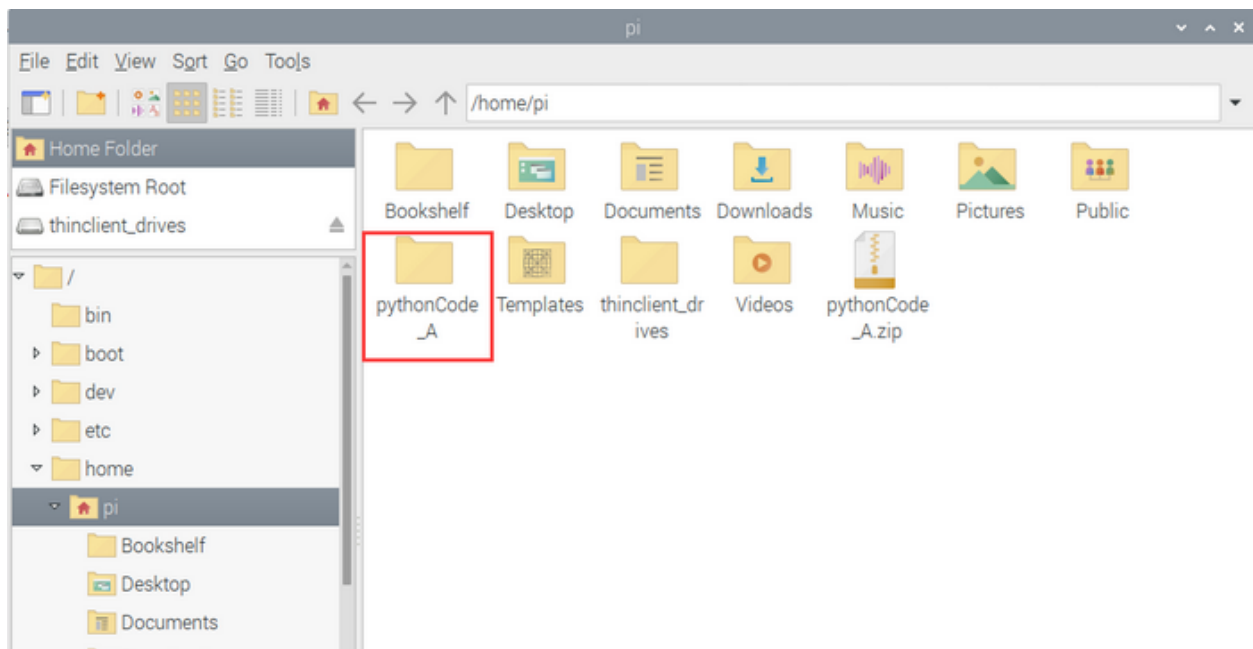
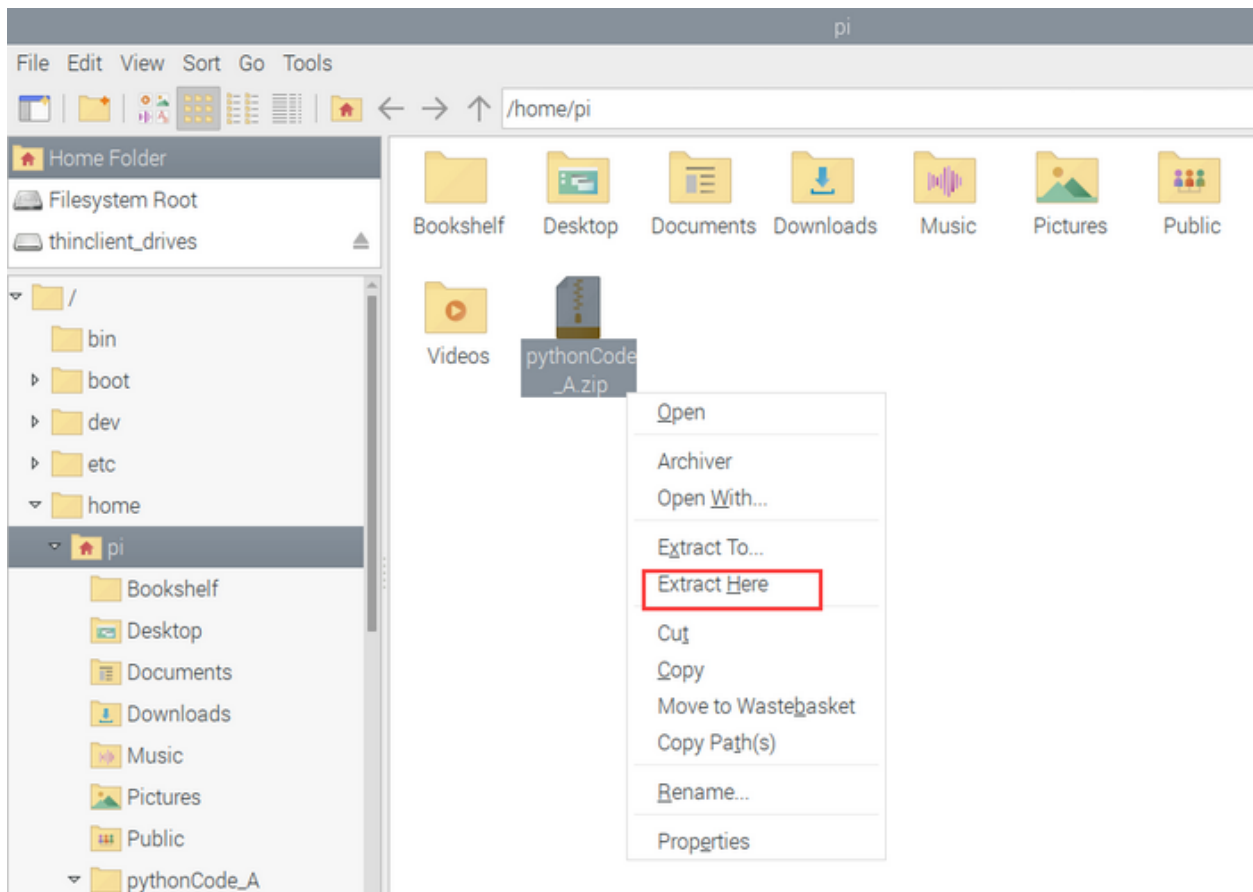


### 6.3.2 3.2Copy Example Code Folder to Raspberry Pi

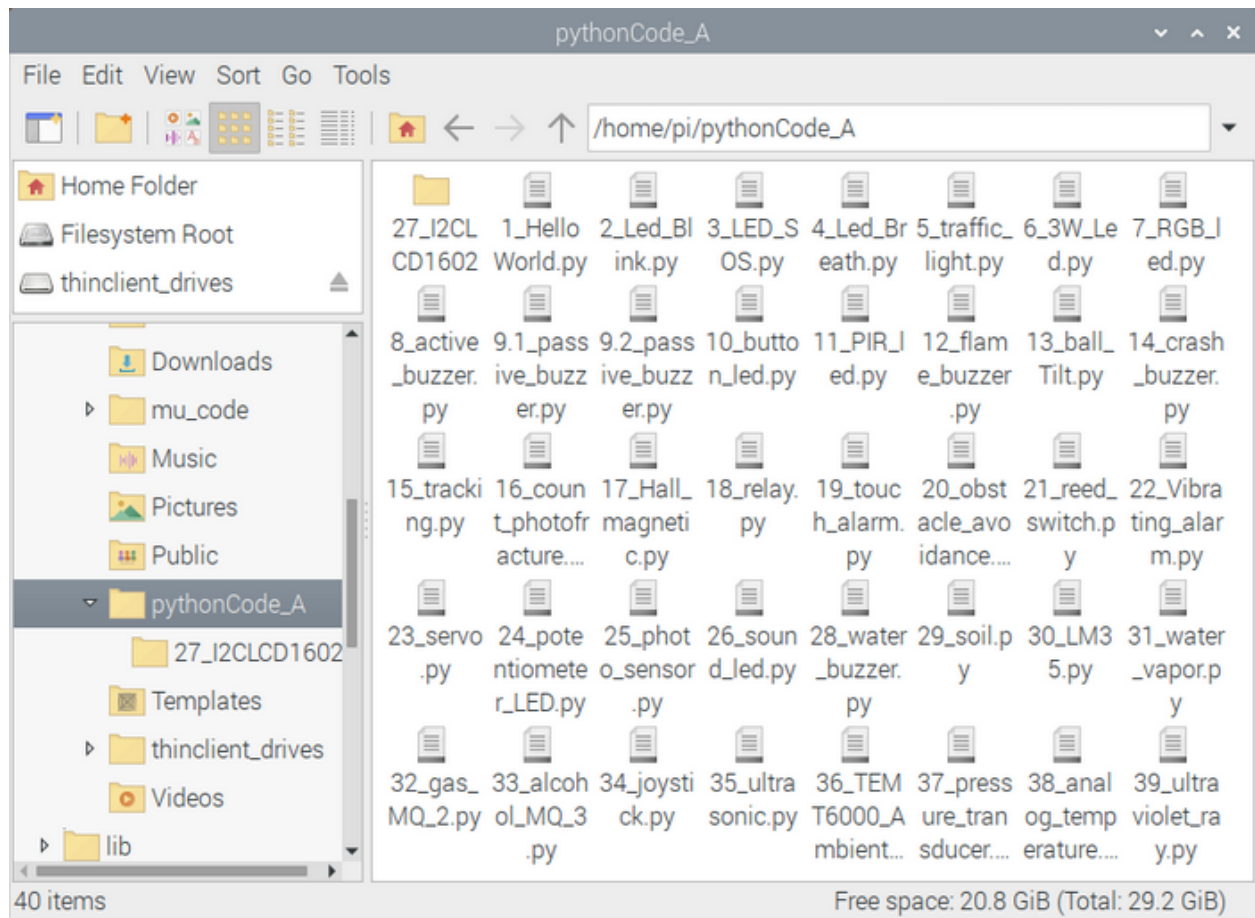
Place the pythonCode\_A.zip folder to the pi folder of Raspberry Pi. And extract the example code from pythonCode\_A folder, as shown below:





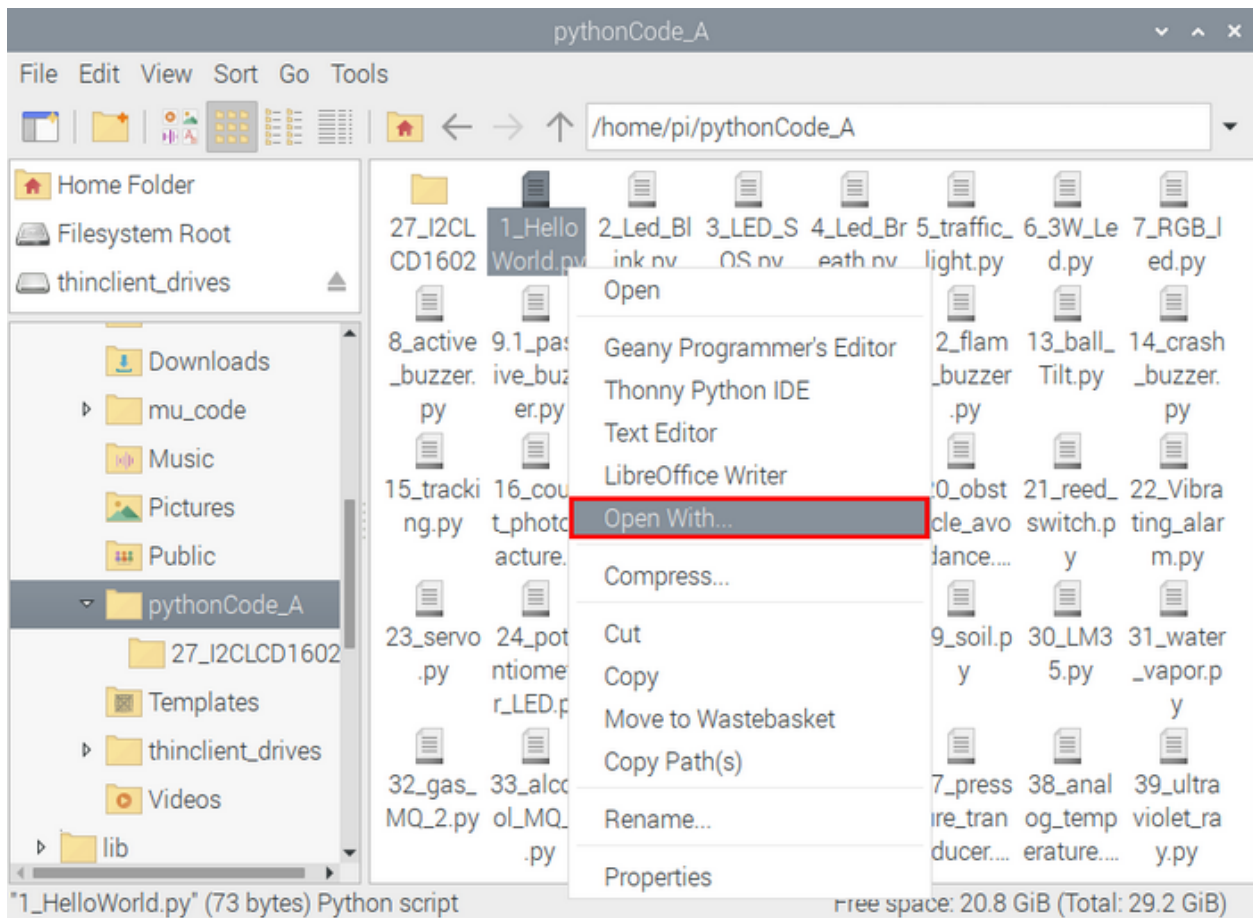


Double-click the pythonCode\_A folder to look through compiled files, as shown below:

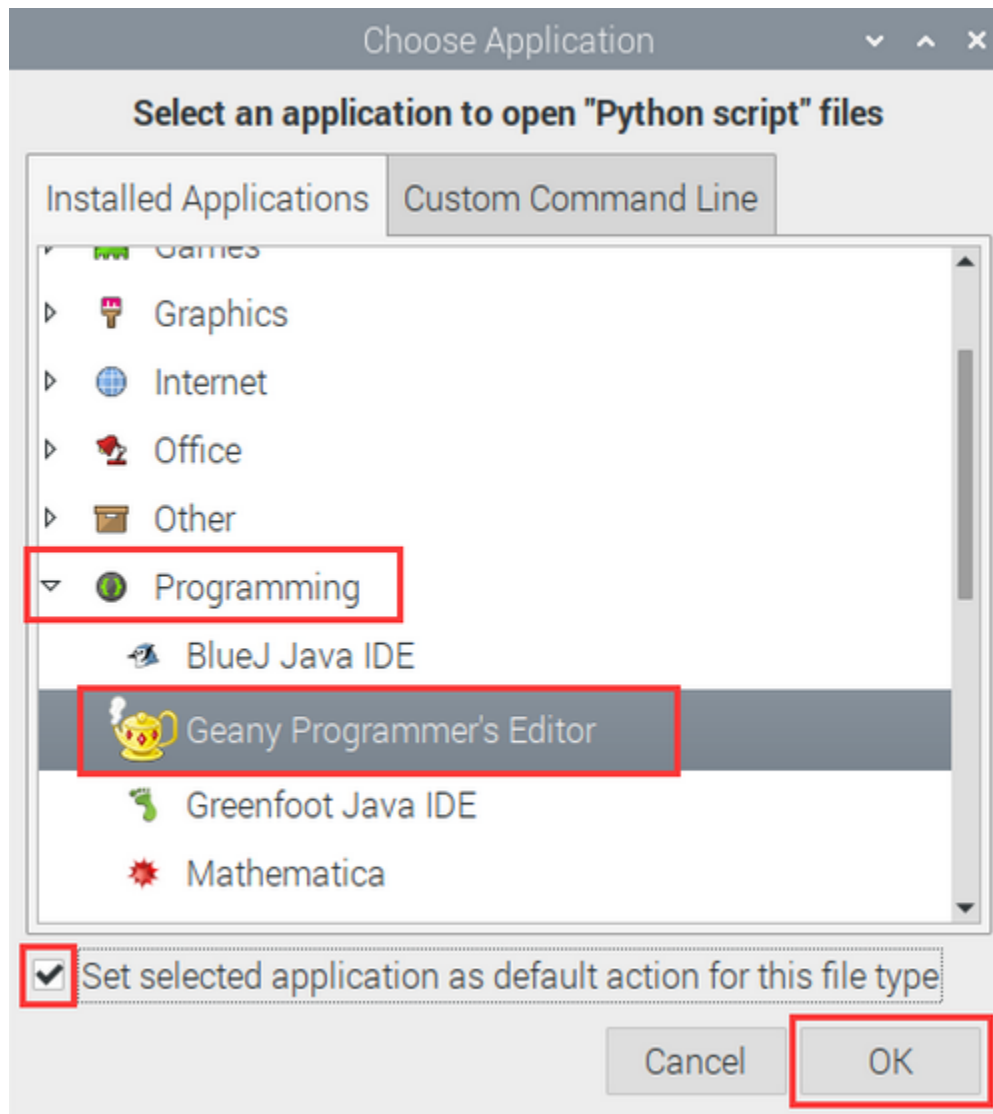


Set the default editor of file with .py

Right-click“Open with...”





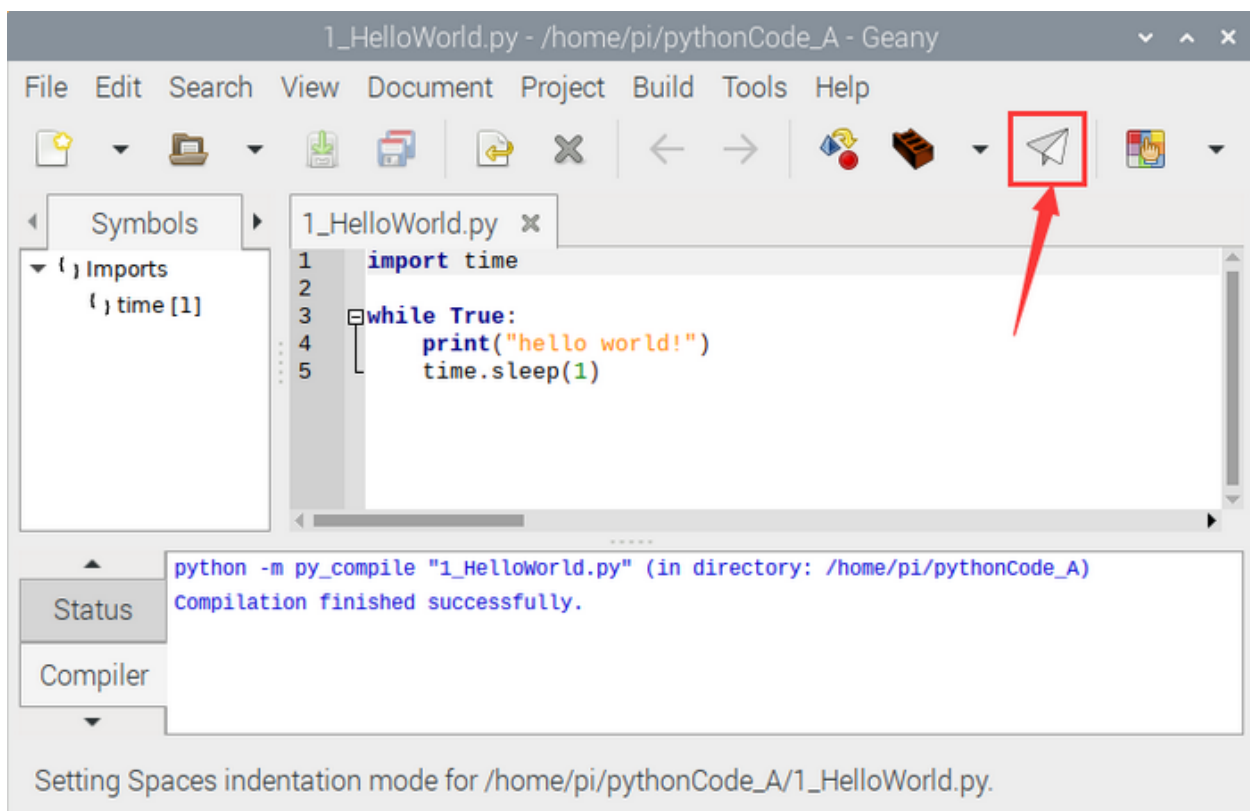
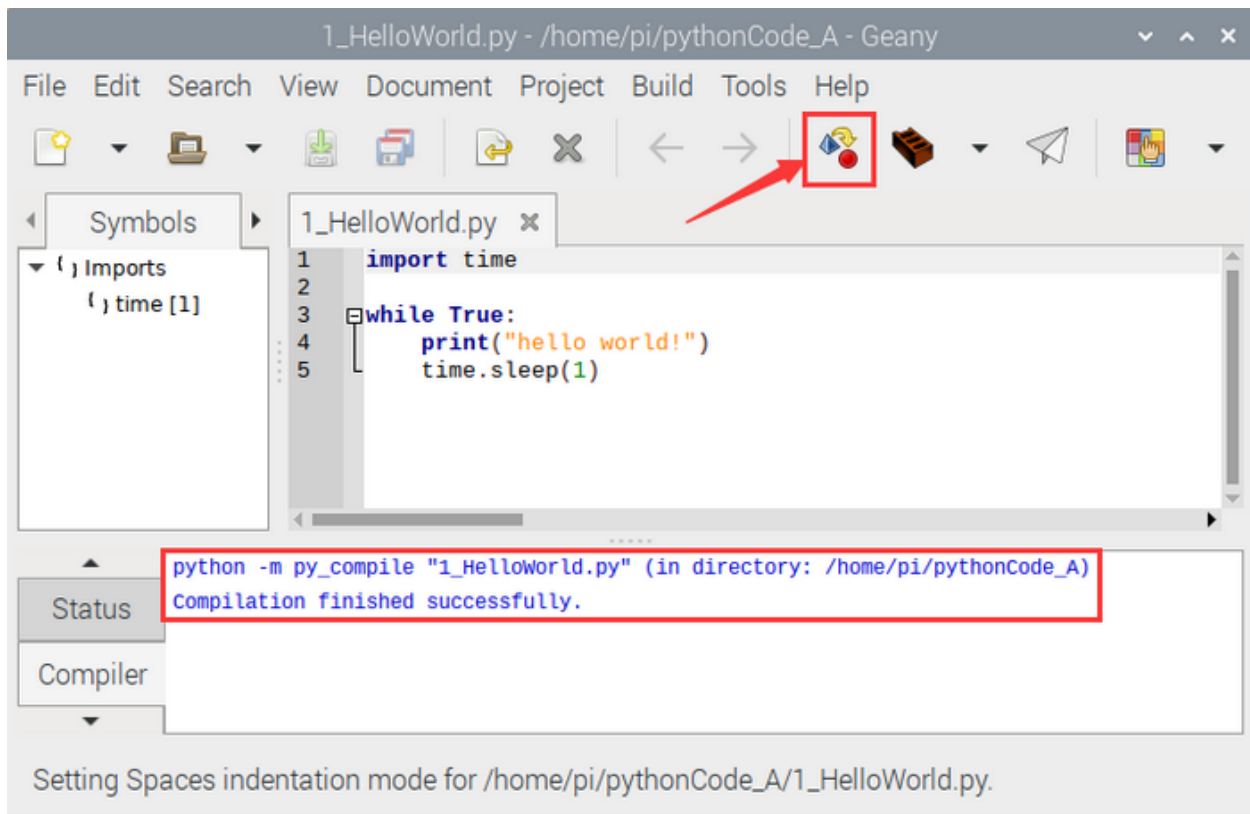
Click Programming to select Geany Programmer's Editor.

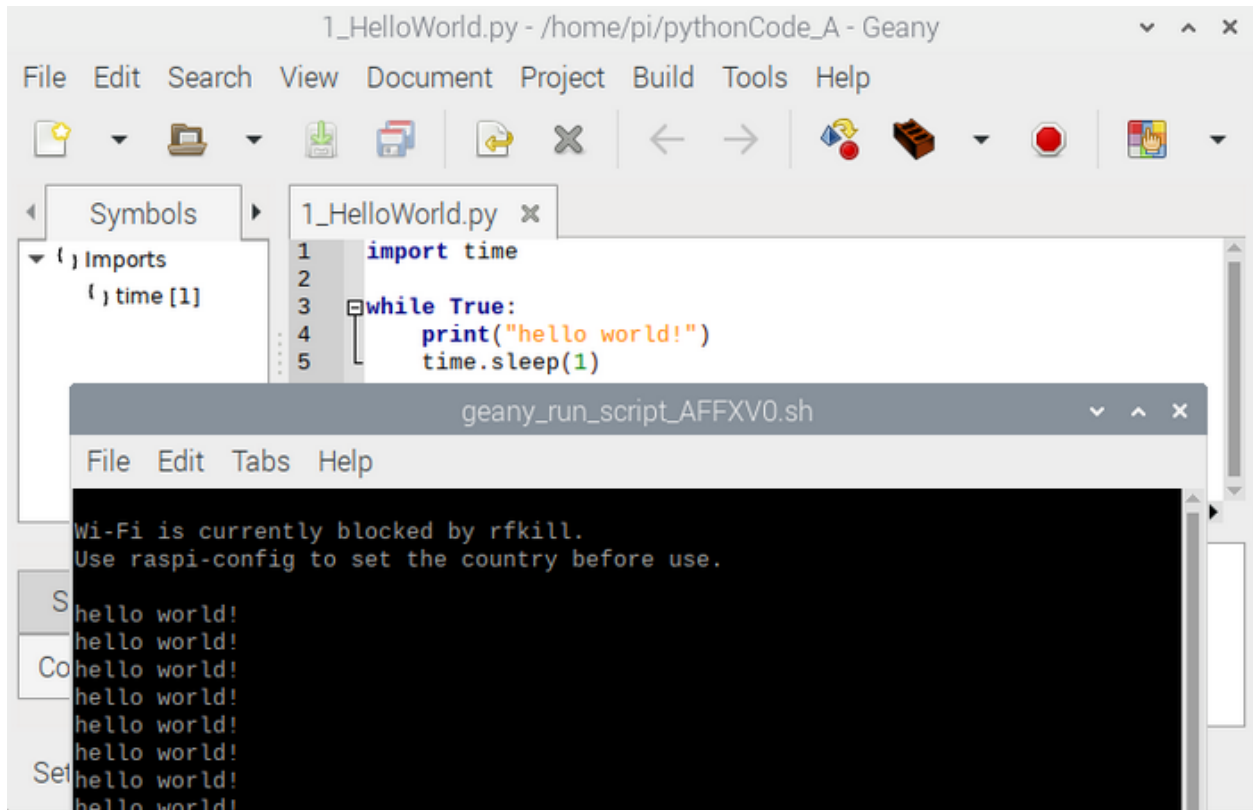


Then, we can directly double-click **Geany Programmer's Editor** to open .py files.

**Run \_HelloWorld.py file to print "Hello World"**

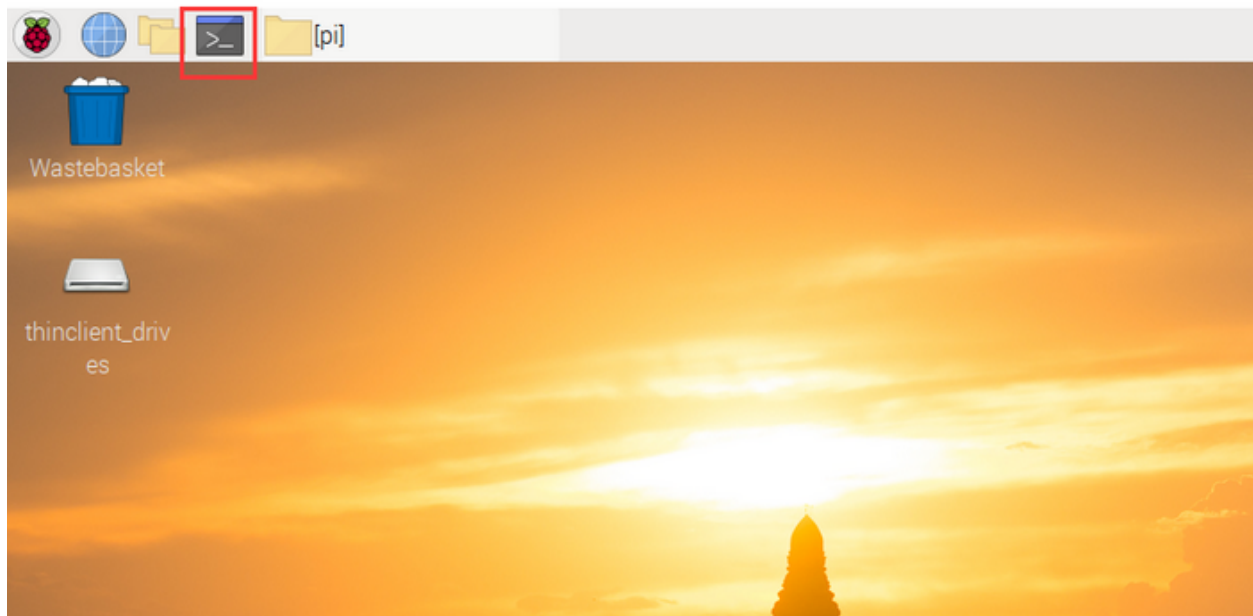
One is to double-click 1\_HelloWorld.py and tap  to compile code and check grammar errors. After successful compilation, tap  to run the code. At same time, terminal appears and prints "hello world"



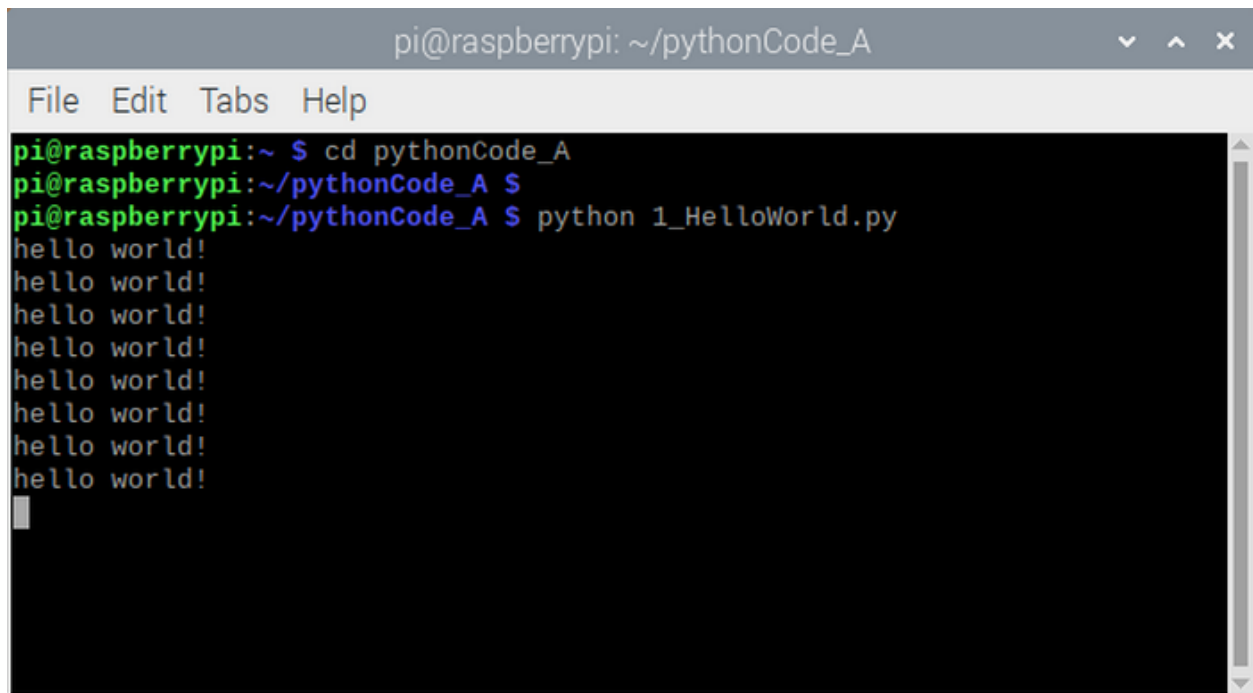


The other way is to open terminal directly, input the following commands and press "Enter" to print "hello world"

```
cd pythonCode_A
python 1_HelloWorld.py
```





A screenshot of a terminal window titled 'pi@raspberrypi: ~/pythonCode\_A'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the following commands and output:

```
pi@raspberrypi:~ $ cd pythonCode_A
pi@raspberrypi:~/pythonCode_A $
pi@raspberrypi:~/pythonCode_A $ python 1_HelloWorld.py
hello world!
hello world!
hello world!
hello world!
hello world!
hello world!
hello world!
hello world!
```

The output consists of eight lines of 'hello world!' printed one after another. A vertical scrollbar is visible on the right side of the terminal window.

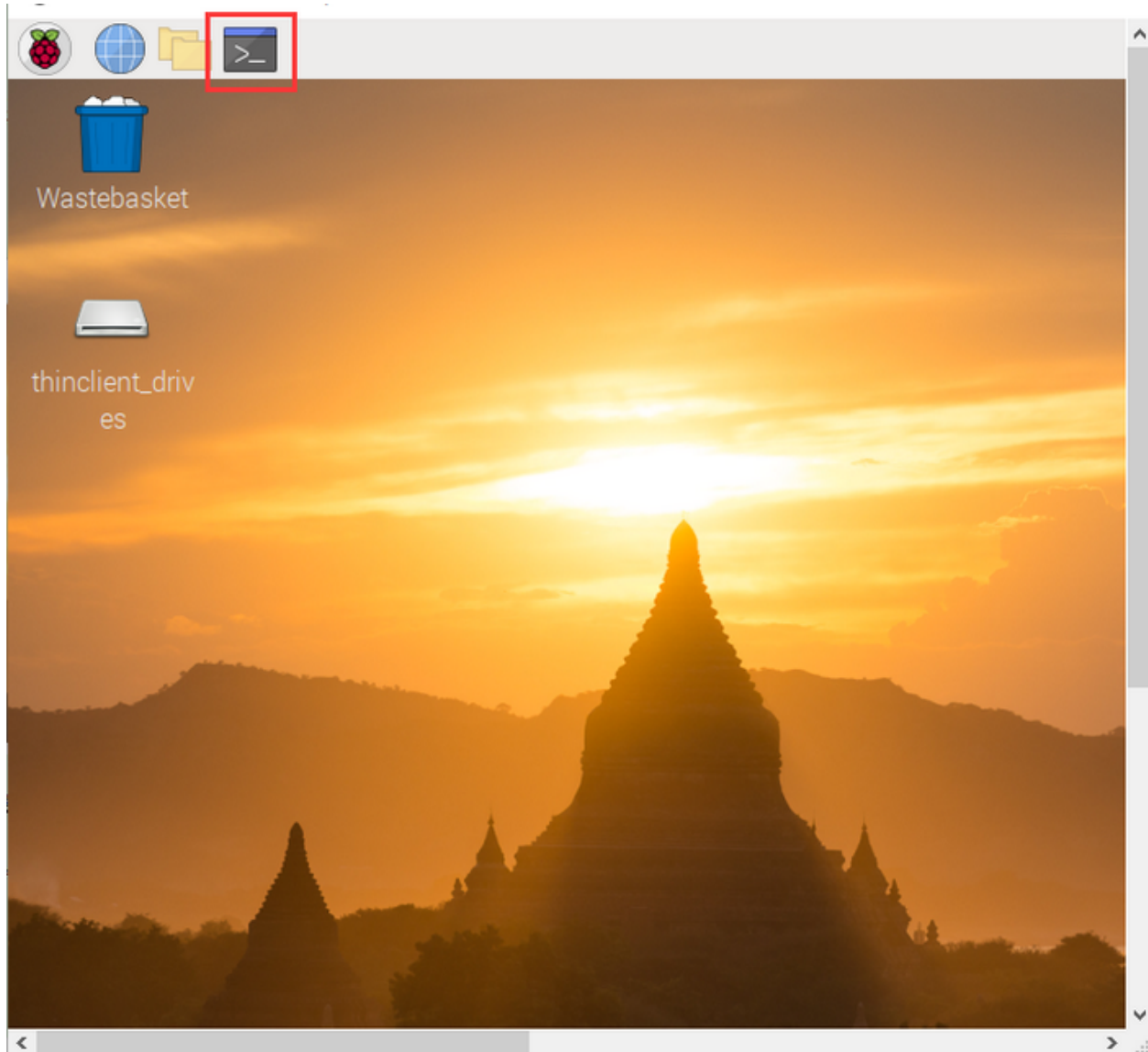
## 6.4 4. Projects

**Note:** G, - and GND marked on sensors and modules are so-called negative, which are connected to GND or G of GPIO board or; V and VCC are known as positive, which are interfaced with 3V3 or 5V on GPIO-PCF8591 shield.

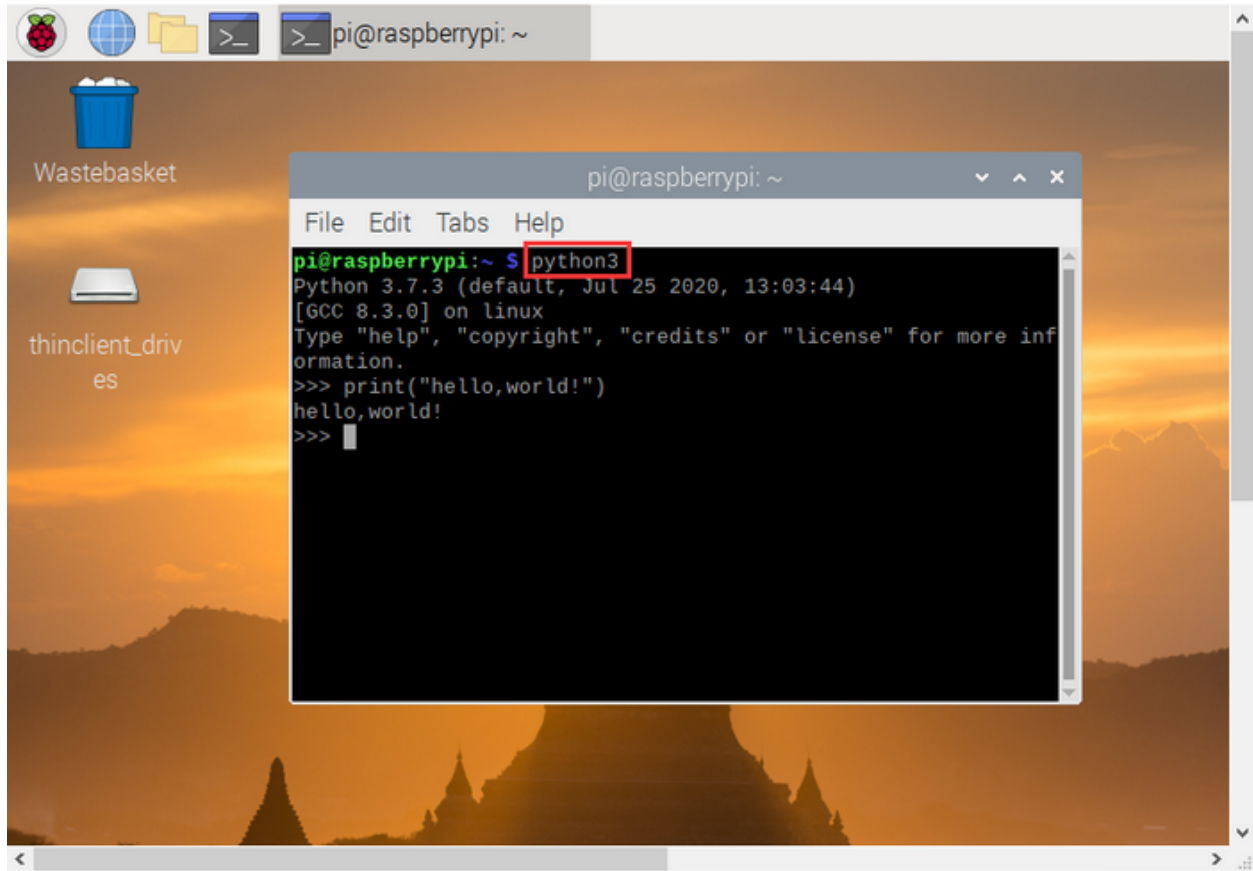
### 6.4.1 Project 1Python3 Shell

Use Windows remote desktop connection to enter the page of Raspberry Pi, then open its terminal.





Input `python3` in the terminal and press “Enter” to enter the `python3` shell’s editing interface, then enter `print(“hello,world!”)` and press “Enter”. The “hello,world!” will be output.



You may find function `print()` is used to print data.

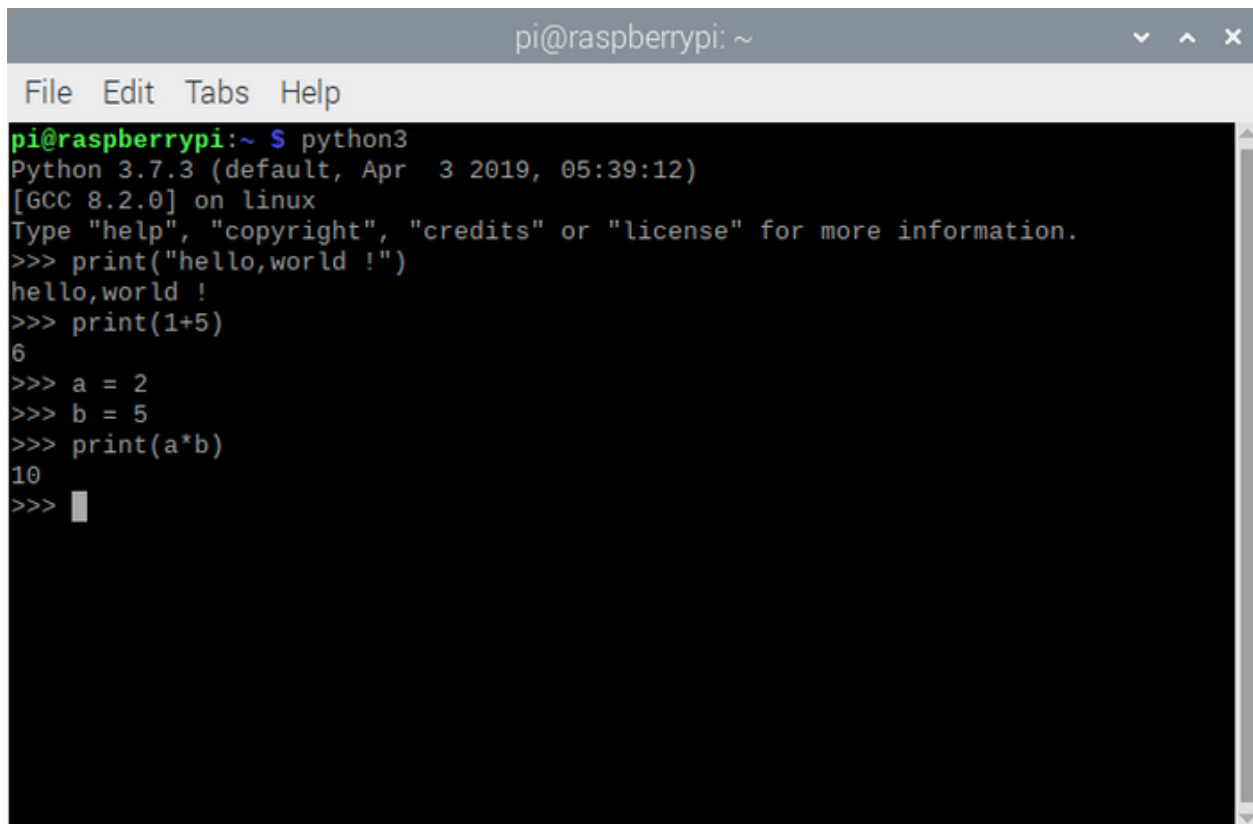
You can print data with other type, like Mathematical formula:

```
print(1+5)
```

```
Variable a = 2 b = 5
```

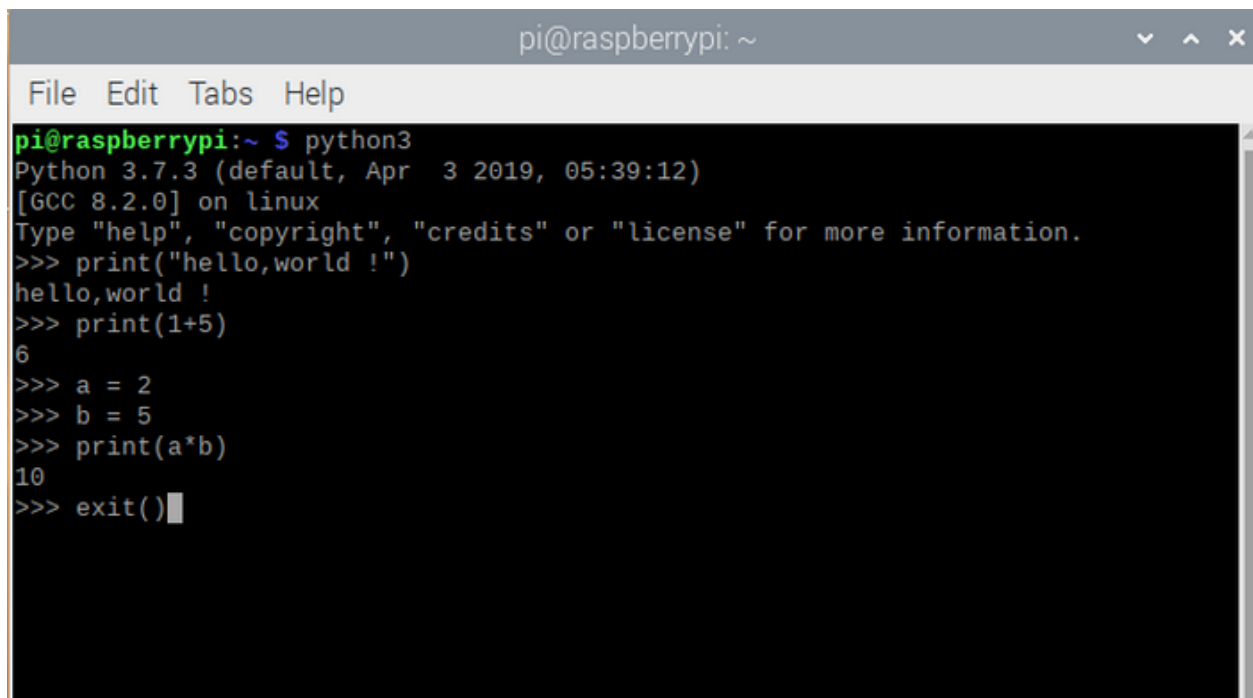
```
print(a*b)
```

As shown below:



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ python3  
Python 3.7.3 (default, Apr  3 2019, 05:39:12)  
[GCC 8.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("hello,world !")  
hello,world !  
>>> print(1+5)  
6  
>>> a = 2  
>>> b = 5  
>>> print(a*b)  
10  
>>> 
```

Input `exit()` to exit python3 shell.



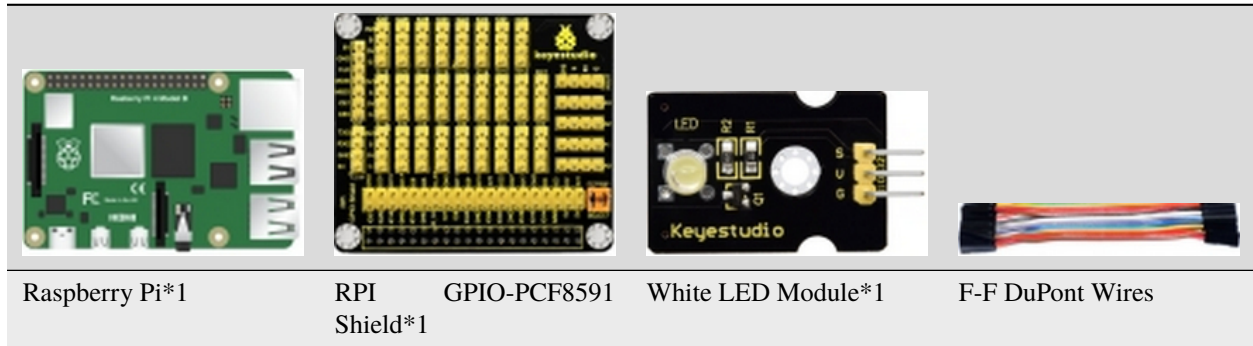
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ python3  
Python 3.7.3 (default, Apr  3 2019, 05:39:12)  
[GCC 8.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("hello,world !")  
hello,world !  
>>> print(1+5)  
6  
>>> a = 2  
>>> b = 5  
>>> print(a*b)  
10  
>>> exit() 
```

## 6.4.2 Project 2LED Blinks

### 1. Description

Let's start from a rather basic and simple experiment—LED Blinks.

### 2. Components



### 3. Component Description

**The white LED module** is a commonly used LED module. It is a F5 LED with white appearance and white light display. During experiments, when the G and V on the module are powered up and the signal end S is at high level, the white LED is on while when the S is at low level, the LED is off.

Modules are compatible with a variety of microcontroller control boards, such as Arduino microcontrollers and white LED module.

### 4. Schematic Diagram

White LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



### 5. The principle to control the LED

According to the connection diagram, the positive (V) of the white LED module is connected to 5V, the negative polar (g) is interfaced with the GND, and the signal terminal (S) is connected to the GPIO18 pin. When the GPIO18 pin outputs high levels, the LED light will be on. When the GPIO18 outputs low levels, the LED lamp is off.

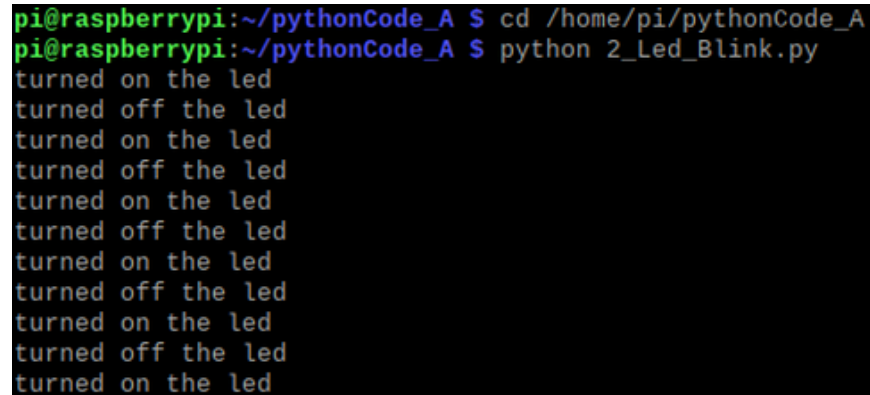
## 6. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 2_Led_Blink.py
```

## 7. Test Results

LED is flashing and the terminal is printing

A terminal window on a Raspberry Pi showing the execution of a Python script. The prompt is 'pi@raspberrypi:~/pythonCode\_A'. The user enters 'cd /home/pi/pythonCode\_A' and then 'python 2\_Led\_Blink.py'. The script outputs a series of messages: 'turned on the led' followed by 'turned off the led', repeating this sequence ten times. The text is displayed in a monospaced font with some color coding (green for prompts, blue for comments in the code).

```
pi@raspberrypi:~/pythonCode_A $ cd /home/pi/pythonCode_A
pi@raspberrypi:~/pythonCode_A $ python 2_Led_Blink.py
turned on the led
turned off the led
turned on the led
turned off the led
turned on the led
turned off the led
turned on the led
turned off the led
turned on the led
turned off the led
turned on the led
```

Note: Press Ctrl + C on keyboard to exit code running

## 8. Example Code

```
import RPi.GPIO as GPIO
import time

ledPin = 18  #define led pin

GPIO.setmode(GPIO.BCM)      # use BCM numbers
GPIO.setup(ledPin,GPIO.OUT)  #set the ledPin OUTPUT mode
GPIO.output(ledPin,GPIO.LOW) # make ledPin output LOW level

while True:    #loop
    GPIO.output(ledPin,GPIO.HIGH) #turn on led
    print("turned on the led")    #Print in the terminal
    time.sleep(1)                 #wait for 1 second
    GPIO.output(ledPin,GPIO.LOW)  #turn off led
    print("turned off the led")
    time.sleep(1)

GPIO.cleanup()    #release all GPIO
```

## 9. Explanation


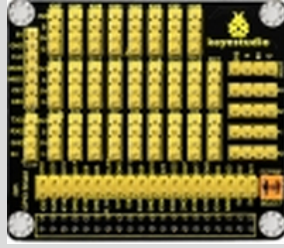

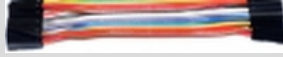
CODE	EXPLANATION
While	While is the loop statement of python, when the condition is true, the program will be executed always be executed.
import RPi.GPIO as GPIO	Import RPi.GPIO library, which can be used to control the digital output of Raspberry Pi and PWM output. GPIO.setmode(GPIO.BCM) There are many definitions about pins of Raspberry Pi, on this condition, we definite pin as BCM digital pin. More resource <a href="https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/">https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/</a>
import time	Import time library, time.sleep(1) means waiting for a second, more resource <a href="https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/">https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/</a>

### 6.4.3 Project 3: SOS Light

#### 1. Description

S.O.S is a Morse code distress signal, used internationally, that was originally established for maritime use. We will present it with flashing LED.

#### 2. Components:

			
Raspberry Main Board*1	RPI GPIO-PCF8591 Shield*1	White LED Module *1	F-F DuPont Wires

#### 3. Schematic Diagram

White LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### 4. Run Example Code

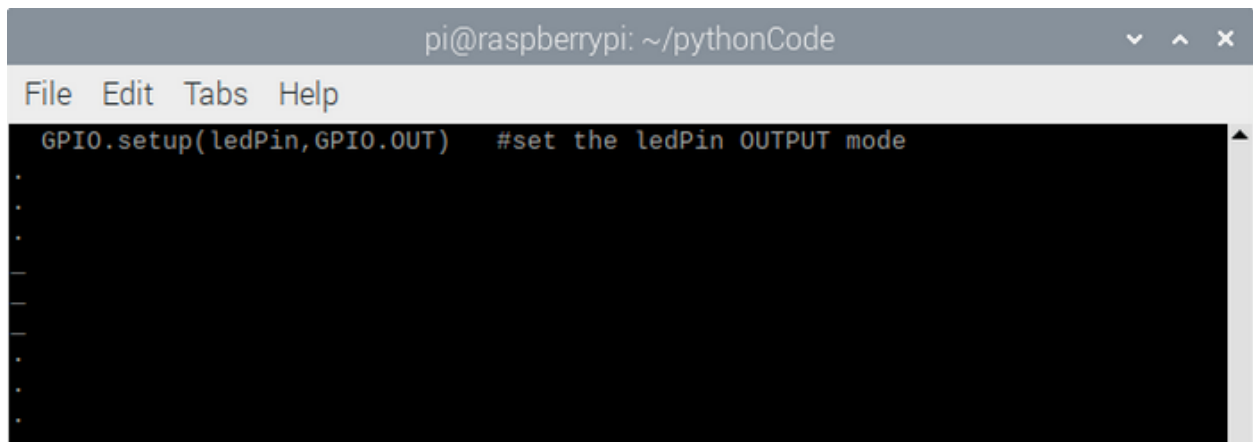
Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 3_LED_SOS.py
```

#### 5. Test Results

LED flashes for three times at once then flashes three times slowly, alternately.

And terminal prints ... \_ \_ \_ ..., as shown below:



Note: Press Ctrl + C on keyboard to exit code running

#### 6. Example Code:

```
import RPi.GPIO as GPIO
import time

ledPin = 18 #define led pin
i1 = 0
i2 = 0
i3 = 0

GPIO.setmode(GPIO.BCM)      # use BCM numbers
GPIO.setup(ledPin,GPIO.OUT)  #set the ledPin OUTPUT mode
GPIO.output(ledPin,GPIO.LOW) # make ledPin output LOW level

while True:    #loop
    while(i1<3):
        GPIO.output(ledPin,GPIO.HIGH) #turn on led
        time.sleep(0.1)                #wait for 1 second
        GPIO.output(ledPin,GPIO.LOW)   #turn off led
        time.sleep(0.1)
        print(".")
        i1 += 1
```

(continues on next page)

(continued from previous page)

```

while(i2<3):
    GPIO.output(ledPin,GPIO.HIGH)  #turn on led
    time.sleep(1)                  #wait for 1 second
    GPIO.output(ledPin,GPIO.LOW)   #turn off led
    time.sleep(1)
    print("_")
    i2 += 1

while(i3<3):
    GPIO.output(ledPin,GPIO.HIGH)  #turn on led
    time.sleep(0.1)                #wait for 1 second
    GPIO.output(ledPin,GPIO.LOW)   #turn off led
    time.sleep(0.1)
    print(".")
    i3 += 1
time.sleep(3)
i1 = 0
i2 = 0
i3 = 0

GPIO.cleanup()    #release all GPIO

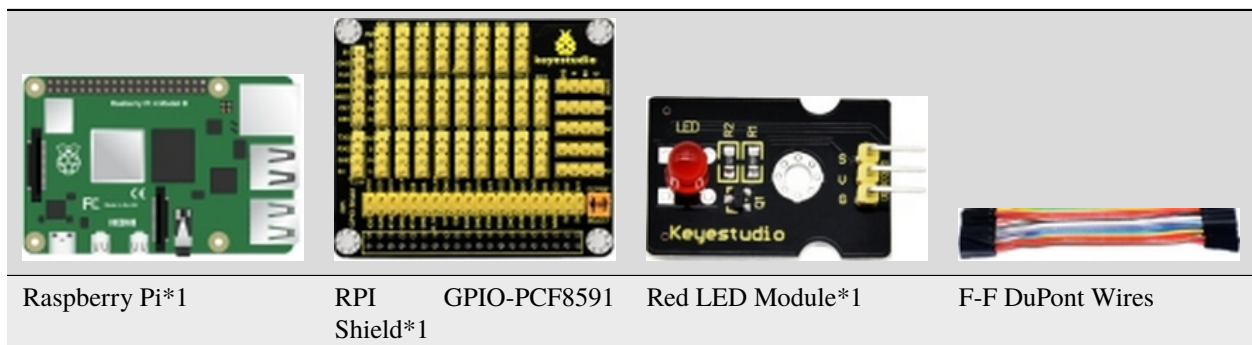
```

## 6.4.4 Project 4: Breathing LED

### 1. Description

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.

### 2. Components





### 3. Working Principle

We use the PWM output of GPIO, PWM outputs analog signals and output value is 0~100 which is equivalent to output voltage 0~3.3V from GPIO port.

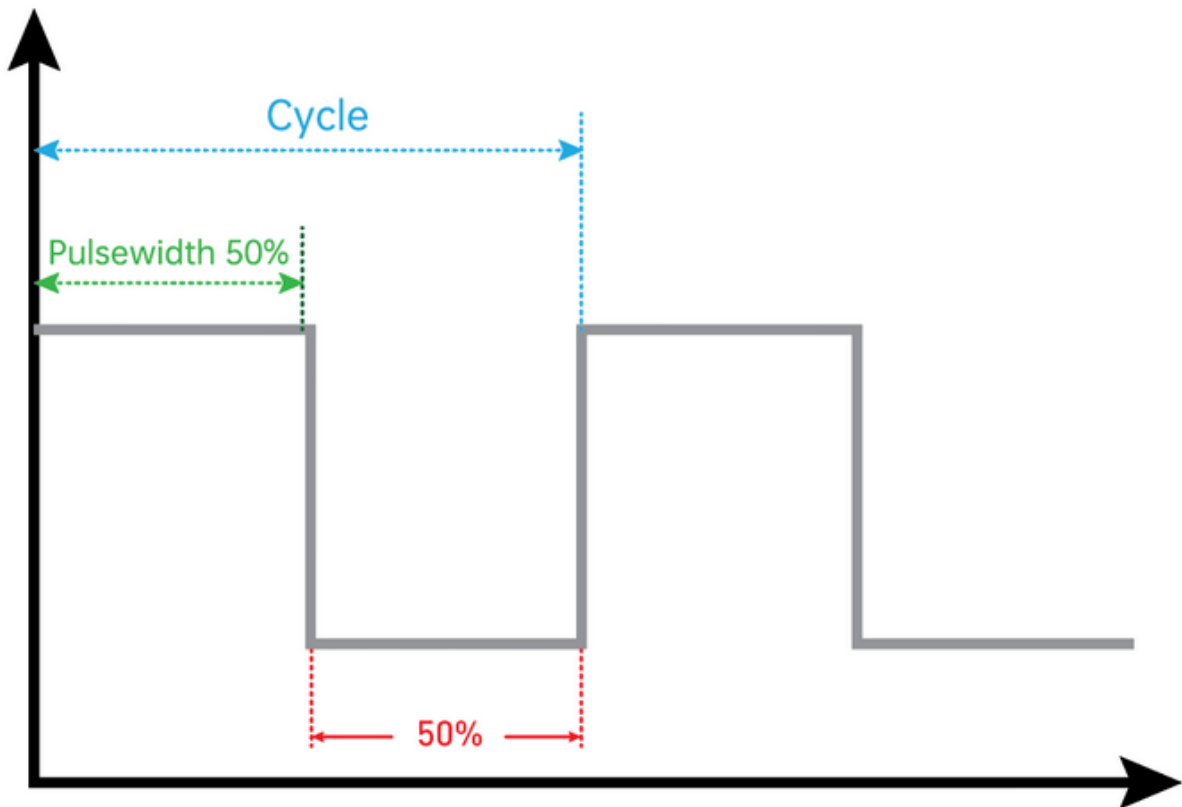
According to Ohm's law:  $U/R = I$ , the resistance is 220, and the value of voltage  $U$  changes, so does the value of current  $I$ , which can control the brightness of the LED lamp.

PWM (Pulse Width Modulation) is the control of the analog circuit through the digital output of microcomputer and a method that making digital coding on analog signal levels.

It sends square waves with certain frequency through digital pins, that is, high level and low level are output alternately for a period of time. Total time of each group high and low level is fixed, which is called cycle.

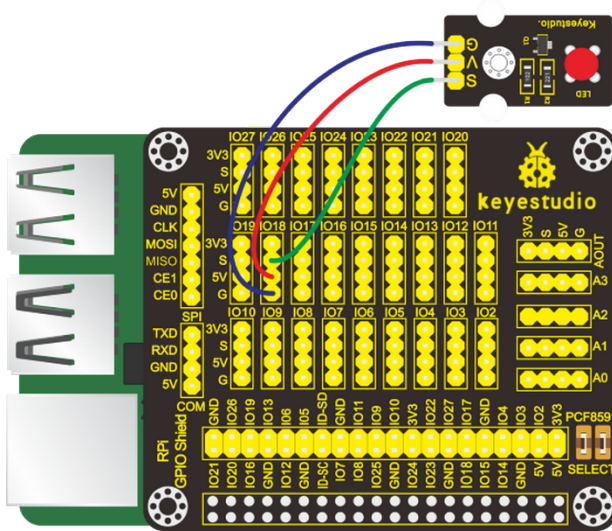
The time of high level output is pulse width whose percentage is called Duty Cycle. The longer that high level lasts, the larger the duty cycle of analog signals is, the corresponding voltage as well.

Below chart is pulse width 50%, then the output voltage is  $3.3 * 50\% = 1.65V$  the brightness of LED is medium.



#### 4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 4_Led_Breath.py
```

#### 6. Test Results

LED gradually brightens then darkens.

Note: Press Ctrl + C on keyboard to exit code running

#### 7. Example Code

```
import RPi.GPIO as GPIO
import time

ledPin = 18 #define led pin

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(ledPin,GPIO.OUT) #set the ledPin OUTPUT mode
GPIO.output(ledPin,GPIO.LOW) # make ledPin output LOW level
pwm = GPIO.PWM(18,100) #create a PWM instance
pwm.start(0) #start PWM
```

(continues on next page)

(continued from previous page)

```

def brighten(): #define function
    for i in range(0,100,+1):
        pwm.ChangeDutyCycle(i) #change the frequency,To lighten gradually
        time.sleep(0.01)

def darken():
    for i in range(100,0,-1):
        pwm.ChangeDutyCycle(i) #To darken gradually
        time.sleep(0.01)

while True: #loop
    brighten() #call function
    darken()

pwm.stop() #stop PWM

GPIO.cleanup() #release all GPIO

```

## 6.4.5 Project 5: Traffic Lights

### 1. Description

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

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

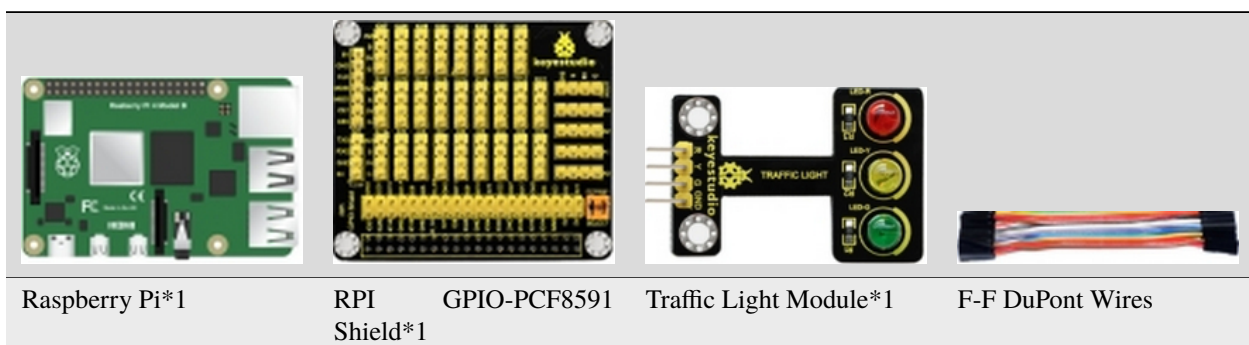
Green light on: Allows traffic to proceed in the direction denoted, if it is safe to do so and there is room on the other side of the intersection.

Red light: Prohibits any traffic from proceeding. A flashing red indication requires traffic to stop and then proceed when safe (equivalent to a stop sign).

Amber light (also known as ‘orange light’ or ‘yellow light’):

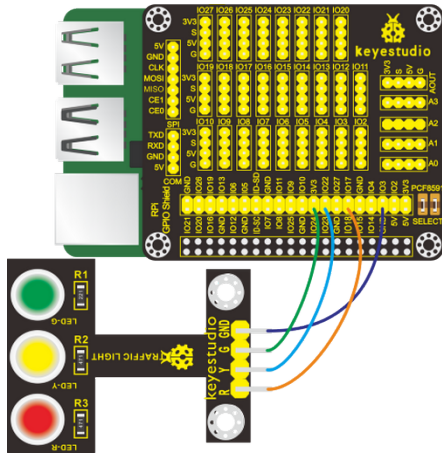
Warns that the signal is about to change to red, with some jurisdictions requiring drivers to stop if it is safe to do so, and others allowing drivers to go through the intersection if safe to do so.

### 2. Components



### 3. Schematic Diagram

Traffic Light Module	RPI GPIO-PCF8591 Shield
R	IO18
Y	IO23
G	IO24
GND	GND



### 4. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 5_traffic_light.py
```

### 5. Test Results

Red light is on 5s and off, yellow light flashes 3s and off, green light is lit for 5s and off, in loop way.

Note: Press Ctrl + C on keyboard to exit code running.

### 6. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#LED pin
R = 18
Y = 23
G = 24

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(R,GPIO.OUT) #set the ledPin OUTPUT mode
GPIO.setup(Y,GPIO.OUT)
```

(continues on next page)

(continued from previous page)

```

GPIO.setup(G,GPIO.OUT)

GPIO.output(R,GPIO.LOW)
GPIO.output(Y,GPIO.LOW)
GPIO.output(G,GPIO.LOW)

while True:
    GPIO.output(R,GPIO.HIGH)
    sleep(5)
    GPIO.output(R,GPIO.LOW)

    GPIO.output(Y,GPIO.HIGH) #turn on yellow_led
    sleep(0.5)
    GPIO.output(Y,GPIO.LOW) #turn off yellow_led
    sleep(0.5)
    GPIO.output(Y,GPIO.HIGH)
    sleep(0.5)
    GPIO.output(Y,GPIO.LOW)
    sleep(0.5)
    GPIO.output(Y,GPIO.HIGH)
    sleep(0.5)
    GPIO.output(Y,GPIO.LOW)
    sleep(0.5)

    GPIO.output(G,GPIO.HIGH) #turn on green_led
    sleep(5) #delay 5s
    GPIO.output(G,GPIO.LOW) #turn off green_led

GPIO.cleanup() #release all GPIO

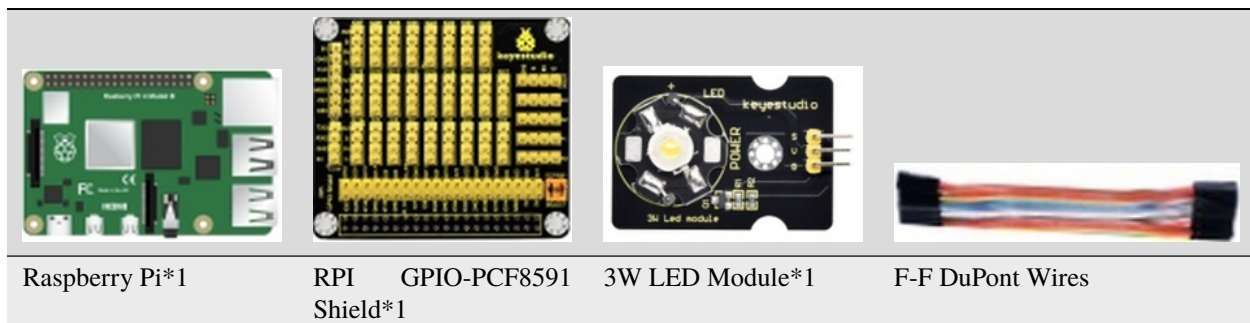
```

## 6.4.6 Project 6Illuminating Lamp

### 1. Description

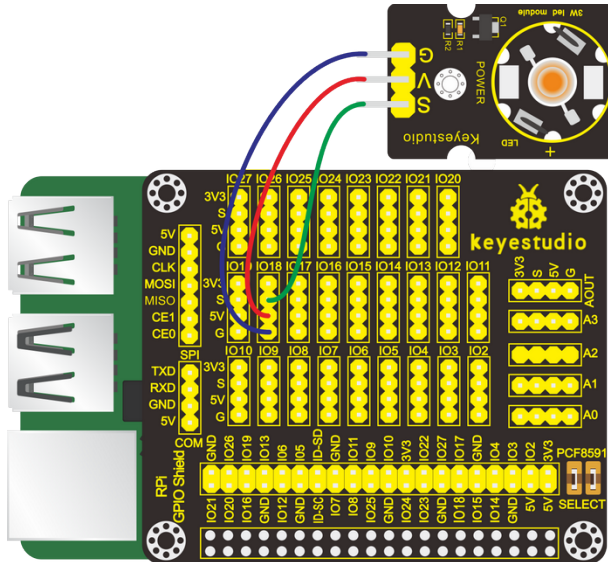
Lights are everywhere in our life. In this chapter, we use a 3W LED module with high brightness. What's more, we apply it into smart robots.

### 2. Components



### 3. Schematic Diagram

3W LED Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



### 4. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 6_3W_Led.py
```

### 5. Test Results

Upload the code, then this 3w LED is on.

Note: Press Ctrl + C on keyboard to exit code running

### 6. Example Code

```
import RPi.GPIO as GPIO
import time

ledPin = 18 #define led pin

GPIO.setmode(GPIO.BCM)      # use BCM numbers
GPIO.setup(ledPin,GPIO.OUT)  #set the ledPin OUTPUT mode
GPIO.output(ledPin,GPIO.LOW) # make ledPin output LOW level
```

(continues on next page)

(continued from previous page)

```
while True:    #loop
    GPIO.output(ledPin,GPIO.HIGH)  #turn on led
    print("turned on the led")  #Print in the terminal

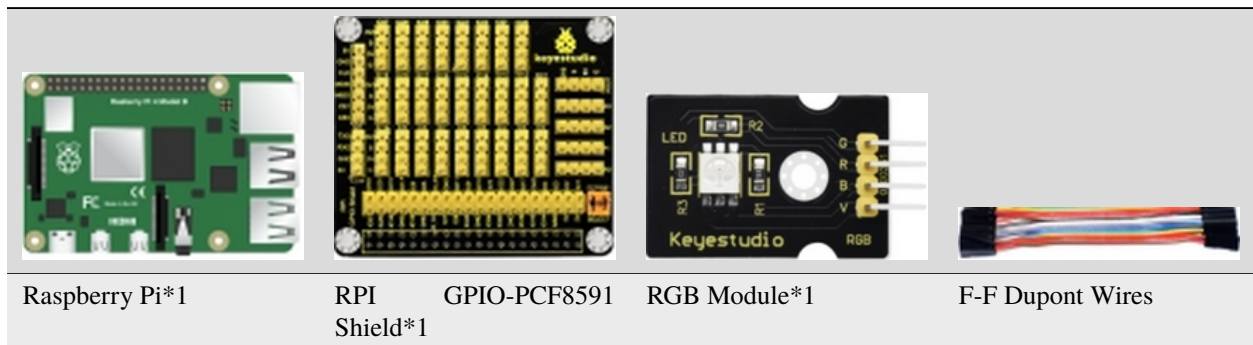
GPIO.cleanup()    #release all GPIO
```

## 6.4.7 Project 7RGB Light

### 1. Description

In this chapter, we will demonstrate how RGB lights show different colors via programming.

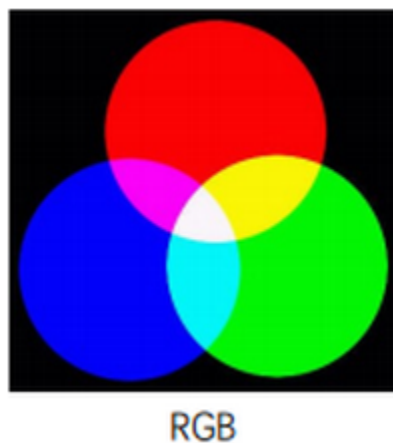
### 2. Components



### 3. Component Knowledge

#### Working Principle

The RGB module integrates with three LEDs in red, green and blue respectively. These three LEDs also share the same anode. The combinations of these three colors can form almost all other colors visible to human eyes. Thus, it has found wide applications in terms of colors.

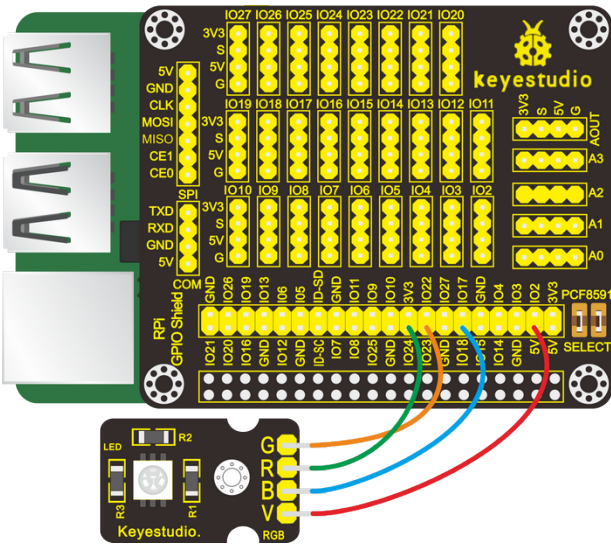


Red, green and blue are three primary colors. They could produce all kinds of visible lights when mixing them up. Computer screen, single pixel mobile phone screen, neon light work under this principle.

Next, we will make a RGB LED display all kinds of colors.

4. Schematic Diagram

RGB Module	RPI GPIO-PCF8591 Shield
R	IO24
G	IO23
B	IO18
V	5V



5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 7_RGB_led.py
```

6. Test Results

RGB light shows colors randomly.

Note: Press Ctrl + C on keyboard to exit code running.



## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep
import random

#define RGB pin
pin_R = 24
pin_G = 23
pin_B = 18
GPIO.setmode(GPIO.BCM) # use BCM numbers
#set the RGB Pin OUTPUT mode
GPIO.setup(pin_R,GPIO.OUT)
GPIO.setup(pin_G,GPIO.OUT)
GPIO.setup(pin_B,GPIO.OUT)

# makeRGB Pin output LOW level
GPIO.output(pin_R,GPIO.LOW)
GPIO.output(pin_G,GPIO.LOW)
GPIO.output(pin_B,GPIO.LOW)

#set pwm frequency to 1000hz
pwm_R = GPIO.PWM(pin_R,100)
pwm_G = GPIO.PWM(pin_G,100)
pwm_B = GPIO.PWM(pin_B,100)
#set initial duty cycle to 0
pwm_R.start(0)
pwm_G.start(0)
pwm_B.start(0)

#function. receive the value to display different colors
def setColor(val_R,val_G,val_B):
    pwm_R.ChangeDutyCycle(val_R)
    pwm_G.ChangeDutyCycle(val_G)
    pwm_B.ChangeDutyCycle(val_B)

while True:
    # get a random in 0~100
    R = random.randint(0,100)
    G = random.randint(0,100)
    B = random.randint(0,100)
    setColor(R,G,B) #set the color value
    print('Red=%d, Green = %d, Blue = %d' %(R, G, B))
    sleep(0.2)

#stop pwm
pwm_R.stop()
pwm_G.stop()
pwm_B.stop()

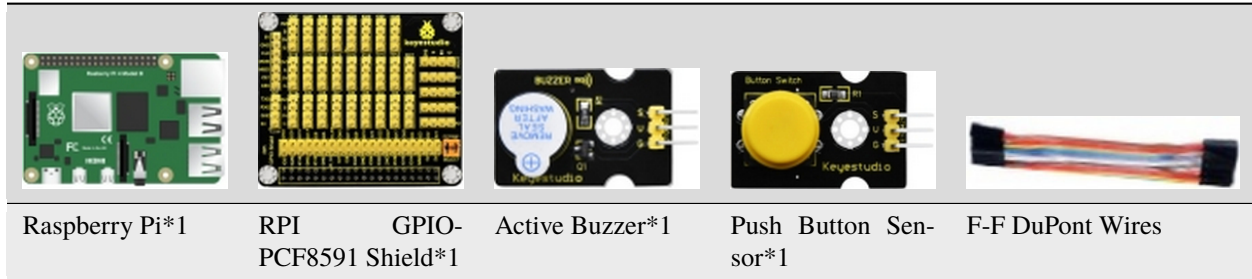
GPIO.cleanup() #release all GPIO
```

## 6.4.8 Project 8Doorbell

### 1. Description

Doorbells have made our daily life more convenient. When a guest arrives, we will get this information when he/she rings the bell. In this project, we will learn to make a doorbell by ourselves.

### 2. Components



### 3. Components Knowledge:

#### Active buzzer

An active buzzer will generate a tone using an internal oscillator, so all that is needed is a DC voltage. A passive buzzer requires an AC signal to make a sound. It is like an electromagnetic speaker, where a changing input signal produces the sound, rather than producing a tone automatically.

As a type of electronic buzzer with integrated structure, buzzers, which are supplied by DC power, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive ones (see the following picture). Turn the pins of two buzzers face up, and the one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.

Button switch: it can control circuit. Before pressed, the current can't pass from one end to the other end. Both ends are like two mountains. There is a river in between. We can't cross this mountain to another mountain. When pressed, my internal metal piece is connecting the two sides to let the current pass, just like building a bridge to connect the two mountains.

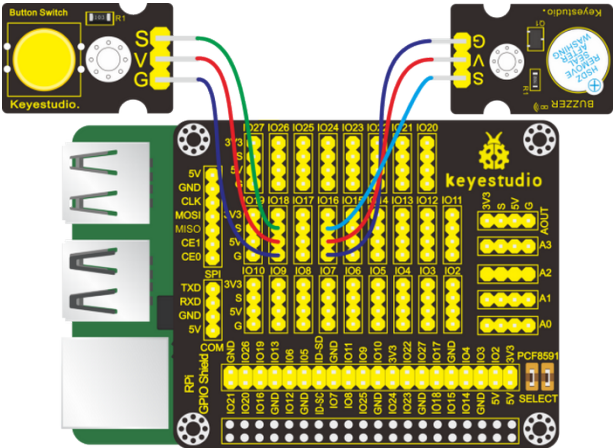


Inner structure:

1 and 1, 2 and 2 are connected, however, 1 and 2 are disconnected when the button is not pressed; 1 and 2 are connected when pressing the button.

4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	Push Button Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	S	SIO18
V	5V	V	5V
G	G	G	G



5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 8_active_buzzer.py
```

6. Test Results

The buzzer will emit sounds and terminal will print 0 if the button is pressed; otherwise, buzzer will keep quiet and terminal will output 1.

Note: Press Ctrl + C on keyboard to exit code running.

7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#active buzzer pin
buzPin = 16
#button pin
btnPin = 18
```

(continues on next page)

(continued from previous page)

```

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(buzPin,GPIO.OUT) #set buzPin OUTPUT mode
GPIO.setup(btnPin,GPIO.IN,GPIO.PUD_UP) # set btnPin INPUT mode and btnPin to PULL UP

while True:
    val = GPIO.input(btnPin)
    print(val)
    if(val == 0): #Judge whether the button is pressed
        GPIO.output(buzPin,GPIO.HIGH) #Buzzer ring
    else:
        GPIO.output(buzPin,GPIO.LOW) #buzzer off

GPIO.cleanup() # Release all GPIO

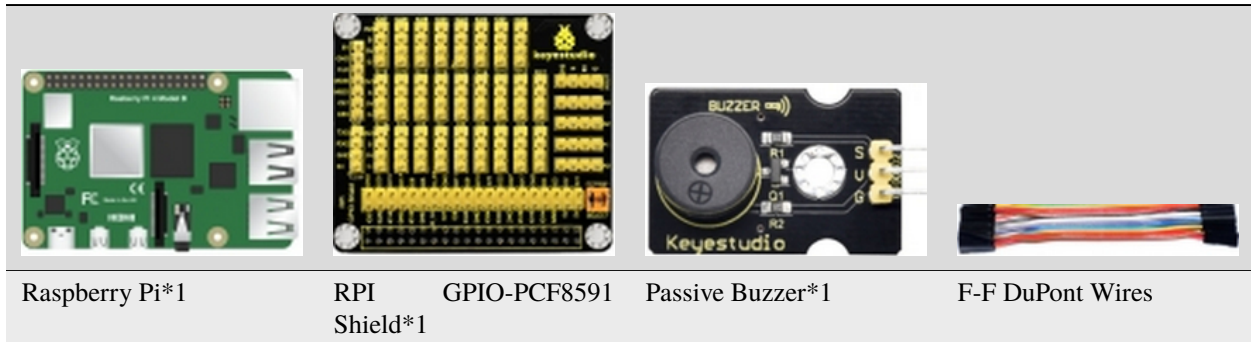
```

## 6.4.9 Project 9: Passive Buzzer

### 1. Description

We will conduct an interesting experiment—control passive buzzer to compose a song.

### 2. Components



### 3. Component Knowledge

#### Passive buzzer



Passive buzzer is a type of electronic buzzer with integrated structure.

Buzzers can be categorized as active and passive ones (see the following picture).

An active buzzer has a built-in oscillating source, so it will make sounds when electrified. But a passive buzzer does not have such source, so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

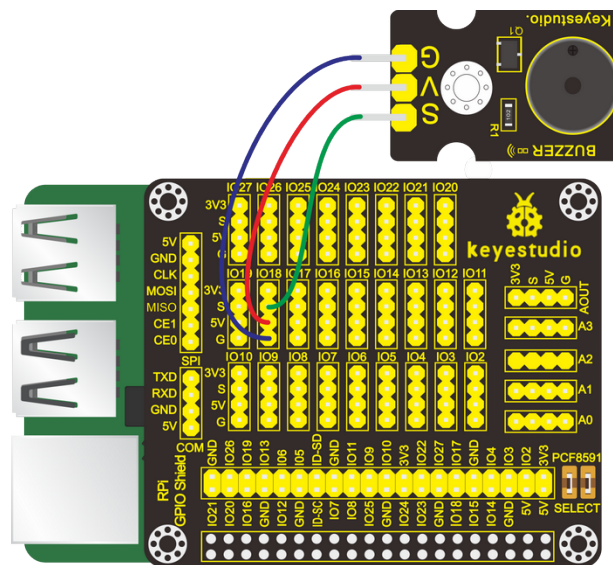
Turn the pins of two buzzers face up, and the one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one, as shown

Passive buzzer provides alternating current to sound coils to make electronic magnet and permanent magnet attraction or repulsion so as to push vibration film to emit sound, according to electromagnetic induction.

Only certain frequency with high and low levels can make passive buzzer emit sound, since DC current only makes vibration film vibrated continuously rather than producing sound.

#### 4. Schematic Diagram

Passive Buzzer	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



#### 5. Run Example Code1

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 9.1_passive_buzzer.py
```

## 6. Test Results1

Passive emits “tick ,tick” sounds.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code1

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import time
import RPi.GPIO as GPIO

buzPin = 18
i1 = 0
i2 = 0
GPIO.setmode(GPIO.BCM)
GPIO.setup(buzPin, GPIO.OUT)

try:
    while 1: #loop
        while(i1<50):
            GPIO.output(buzPin,GPIO.HIGH)
            time.sleep(0.001) #wait for 1 ms
            GPIO.output(buzPin,GPIO.LOW)
            time.sleep(0.001)
            i1 = i1 + 1
        time.sleep(0.3)
        while(i2<50):
            GPIO.output(buzPin,GPIO.HIGH)
            time.sleep(0.001) #wait for 1 ms
            GPIO.output(buzPin,GPIO.LOW)
            time.sleep(0.001)
            i2 = i2 + 1
        time.sleep(1)
        i1 = 0
        i2 = 0
except KeyboardInterrupt:
    pass
p.stop() #stop pwm
GPIO.cleanup() #release all GPIO
```

## 8. Run Example Code 2

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 9.2_passive_buzzer.py
```

## 9. Test Results 2

Passive buzzer plays a“Happy Birthday”song.

Note: Press Ctrl + C on keyboard to exit code running.

## 10. Example Code2

```
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time

Buzzer = 18 # set the Pin

# Happy birthday
Do = 262
Re = 294
Mi = 330
Fa = 349
Sol = 392
La = 440
Si = 494
Do_h = 523
Re_h = 587
Mi_h = 659
Fa_h = 698
Sol_h = 784
La_h = 880
Si_h = 988

# The tune
song_1 = [
    Sol,Sol,La,Sol,Do_h,Si,
    Sol,Sol,La,Sol,Re_h,Do_h,
    Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,
    Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_h
]
# delay
beat_1 = [
    0.5,0.5,1,1,1,1+1,
    0.5,0.5,1,1,1,1+1,
    0.5,0.5,1,1,1,1,1,
    0.5,0.5,1,1,1,1+1,
]
```

(continues on next page)

(continued from previous page)

```

def setup():
    GPIO.setmode(GPIO.BCM)      # Numbers GPIOs by physical location
    GPIO.setup(Buzzer, GPIO.OUT) # Set pins' mode is output
    global Buzz
    Buzz = GPIO.PWM(Buzzer, 440) # 440 is initial frequency.
    Buzz.start(50)              # Start Buzzer pin with 50% duty ration

def loop():
    while True:
        print('\n    Playing song 3...')
        for i in range(0, len(song_1)): # Play song 1
            Buzz.ChangeFrequency(song_1[i]) # Change the frequency along the song note
            time.sleep(beat_1[i] * 0.5)    # delay a note for beat * 0.5s

def destroy():
    Buzz.stop()                  # Stop the buzzer
    GPIO.output(Buzzer, 1)      # Set Buzzer pin to High
    GPIO.cleanup()              # Release resource

if __name__ == '__main__':     # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy()
        ↪will be executed.
        destroy()

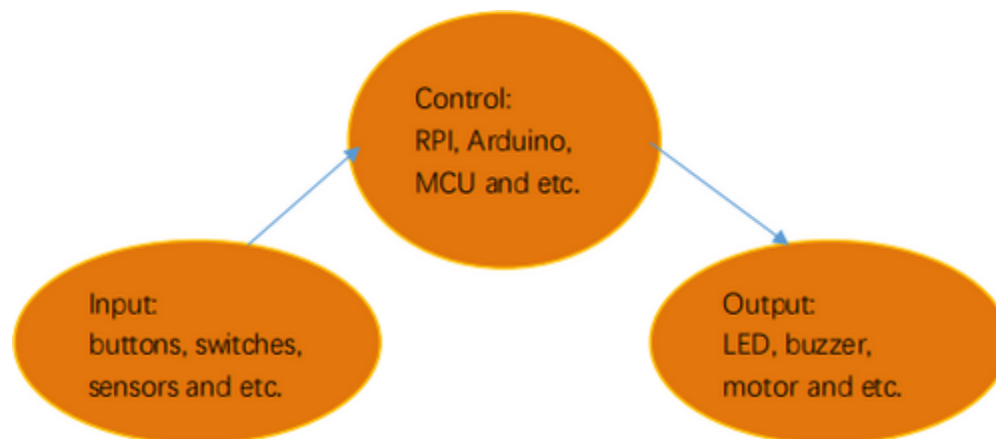
```

## 6.4.10 Project 10Button-controlled LED

### 1. Description

Usually, a complete open loop control is made of external information input. Controller and actuator.

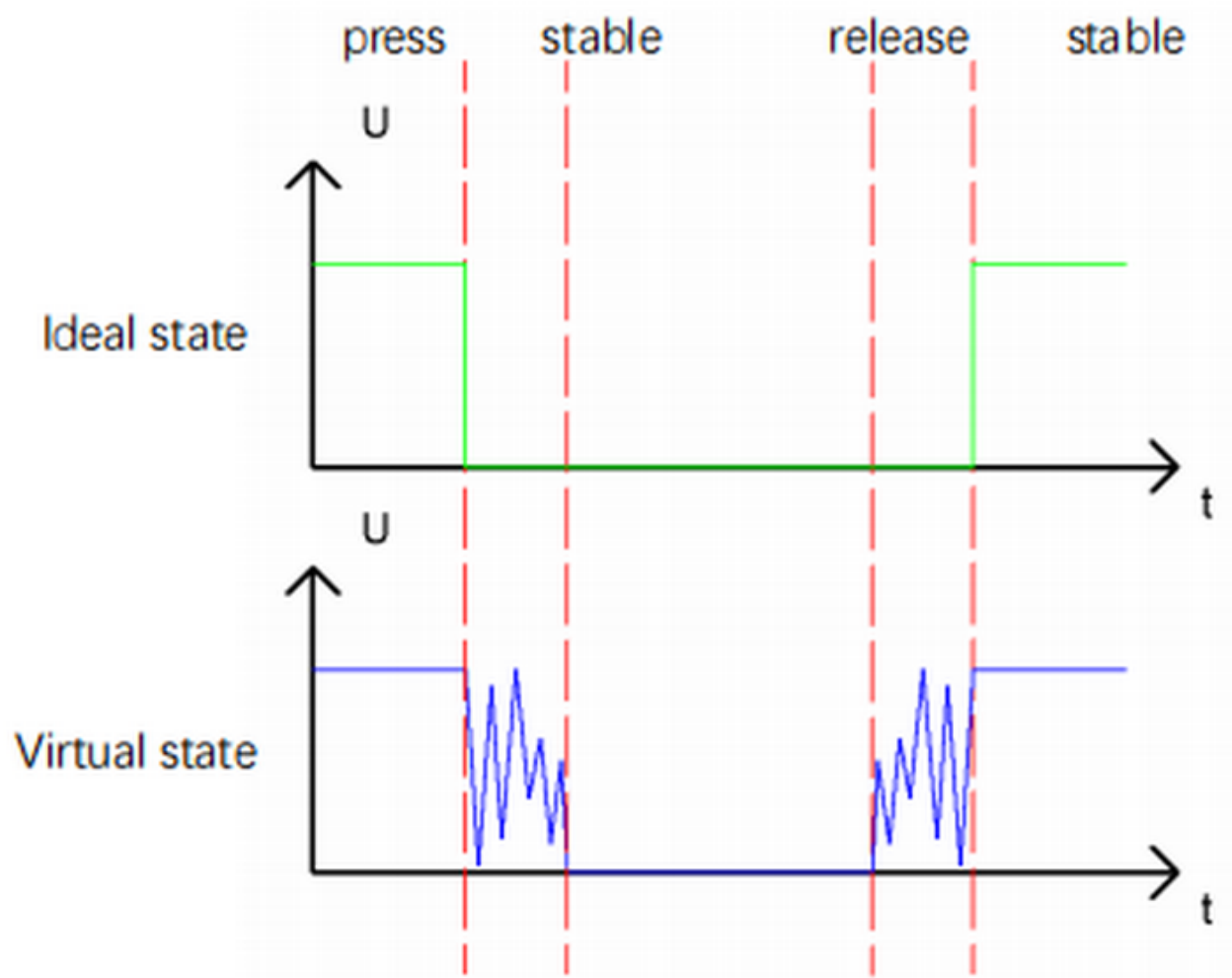
The external information is input into controller which can analyze the input data and send to control signals to make actuator to react.



A button-controlled LED is decided by an open loop control. Next, we will make a desk lamp with a button, an LED and RPi. LED is on when button is pressed, on the contrary, it will be off.







Therefore, there will be many a presses and release actions. The shaking will misleads the high speed movement of MCU, causing wrong judgement. That requires that we need to judge the button' status frequently.

The button means being pressed when its status is stable.

## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 10_button_led.py
```

## 6. Test Results

Press the button, LED turns on, then press it again, LED is off.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

LED = 16 #set ledPin
button = 18 #set buttonPin
val = 0 #Button variables
count = 0 #Record the number of button presses
flag = 0 #Odd even variable
GPIO.setmode(GPIO.BCM) # use BCM numbers

GPIO.setup(LED,GPIO.OUT) #set the ledPin OUTPUT mode
GPIO.setup(button,GPIO.IN,GPIO.PUD_UP) #set the buttonPin INPUT mode and buttonPin to
↪PULL UP

while True:
    val = GPIO.input(button) #Receive button value
    #print("button = %d"%(val))
    if(val == 0): #if button is pressed
        sleep(0.01) #Eliminate button jitter
        val = GPIO.input(button) #Receive button value
        if(val == 1): #Loosen the button
            count = count + 1 #Count the number of clicks on the button
            print("count = %d" %count)

    flag = count % 2 #Remainder 2 ,Even is 0, odd is 1
    if(flag == 1):
        GPIO.output(LED,GPIO.HIGH) #turn on led
    else:
        GPIO.output(LED,GPIO.LOW) #turn off led

GPIO.cleanup() #release all GPIO
```

### 6.4.11 Project 11PIR Motion Sensor

#### 1. Description

In this lesson, we will learn about PIR motion sensor.

## 2. Components



## 3. Component Knowledge

### PIR Motion Sensor

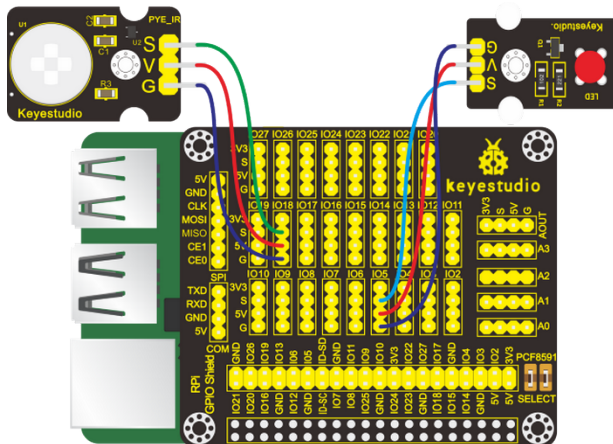
The principle of human infrared sensor is that when certain crystals, such as lithium tantalite and triglyceride sulfate, are heated, the two ends of the crystal will generate an equal number of charges, with opposite signs, which can be converted into voltage output by an amplifier.

Human body will emit IR ray, although weak but can be detected. Sensor will output high level(1) when human being is detected by sensor, otherwise, it will output low level(0).

Note: Nothing but moving person can be detected, with the detection distance is up to 3m.

## 4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	PIR Motion Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 11_PIR_led.py
```

## 6. Test Results

LED will turn on and terminal will print **somebody** if the PIR motion sensor detects people; if not, LED will be off and terminal will print **nobody**.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

ledPin = 5  #set led pin
pirPin = 18 #set PYE-IR pin
GPIO.setup(ledPin,GPIO.OUT)
GPIO.setup(pirPin,GPIO.IN)

while True:                                ##loop
    if GPIO.input(pirPin):                  #When someone is detected
        GPIO.output(ledPin,GPIO.HIGH)      #turn on the led
        print("somebody")
    else:
        GPIO.output(ledPin,GPIO.LOW)       #turn off led
        print("nobody")

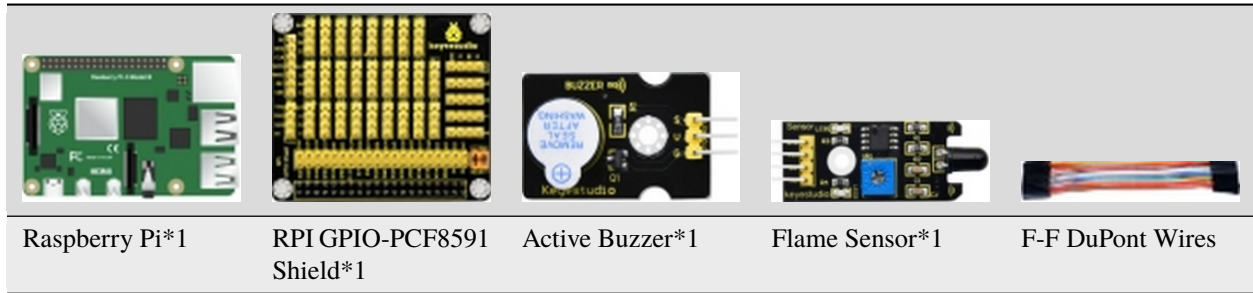
GPIO.cleanup()
```

## 6.4.12 Project 12Fire Alarm

### 1. Description

A flame detector is a sensor designed to detect and respond to the presence of a flame or fire, allowing flame detection.

## 2. Components



## 3. Component Knowledge

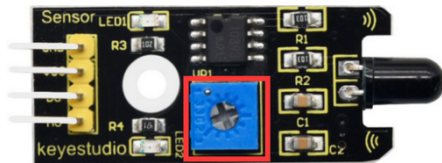
### Flame Sensor

Flame sensor is made based on the principle that infrared ray is highly sensitive to flame. It has an infrared receiving tube specially designed to detect fire, and then convert the flame brightness to fluctuating level signal. The signals are then input into the central processor and be dealt with accordingly.

Flame sensor is used to detect fire source with wavelength in 760nm-1100nm, detection angle is 60°. When its IR waves length is close to 940nm, and its sensitivity is highest.

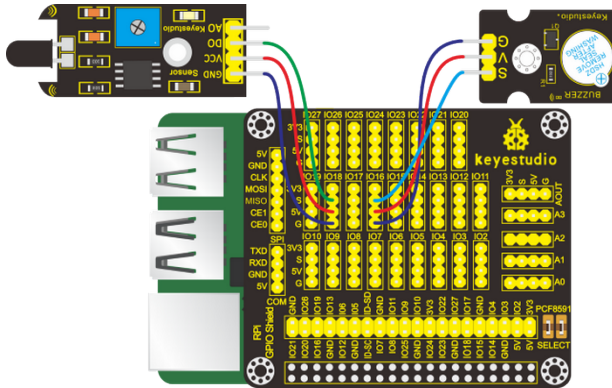
Notice that keep flame sensor away from fire source to defend its damage for its working temperature is between -25°-85°

Note: You can rotate the potentiometer on the module to adjust module's sensitivity



## 4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	Flame Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	D0	SIO18
V	5V	VCC	5V
G	G	GND	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 12_flame_buzzer.py
```

## 6. Test Results

When the flame is detected, the buzzer will make a sound and the terminal will print low level 0, LED1 will be on; otherwise, no sound will be emitted, the terminal will print high level 1 and LED1 will be off.

Buzzer will alarm when detecting fire; otherwise, it will stop emitting sound.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#define buzzer pin
buzPin = 16
#define flame Pin
flamePin = 18

val = 0 #

GPIO.setmode(GPIO.BCM) #use BCM numbers
GPIO.setup(buzPin,GPIO.OUT) #set the buzPin OUTPUT
GPIO.setup(flamePin,GPIO.IN,GPIO.PUD_UP) #set the flamePin INPUT

while True:
    val = GPIO.input(flamePin) #Receives the value of the flame sensor
    print("val = %d" %val)
    if (val == 0): #When flame is detected
        GPIO.output(buzPin,GPIO.HIGH) #Buzzer turn on
    else:
        GPIO.output(buzPin,GPIO.LOW) #buzzer turn off
```

(continues on next page)

(continued from previous page)

```
GPIO.cleanup() # Release all GPIO
```

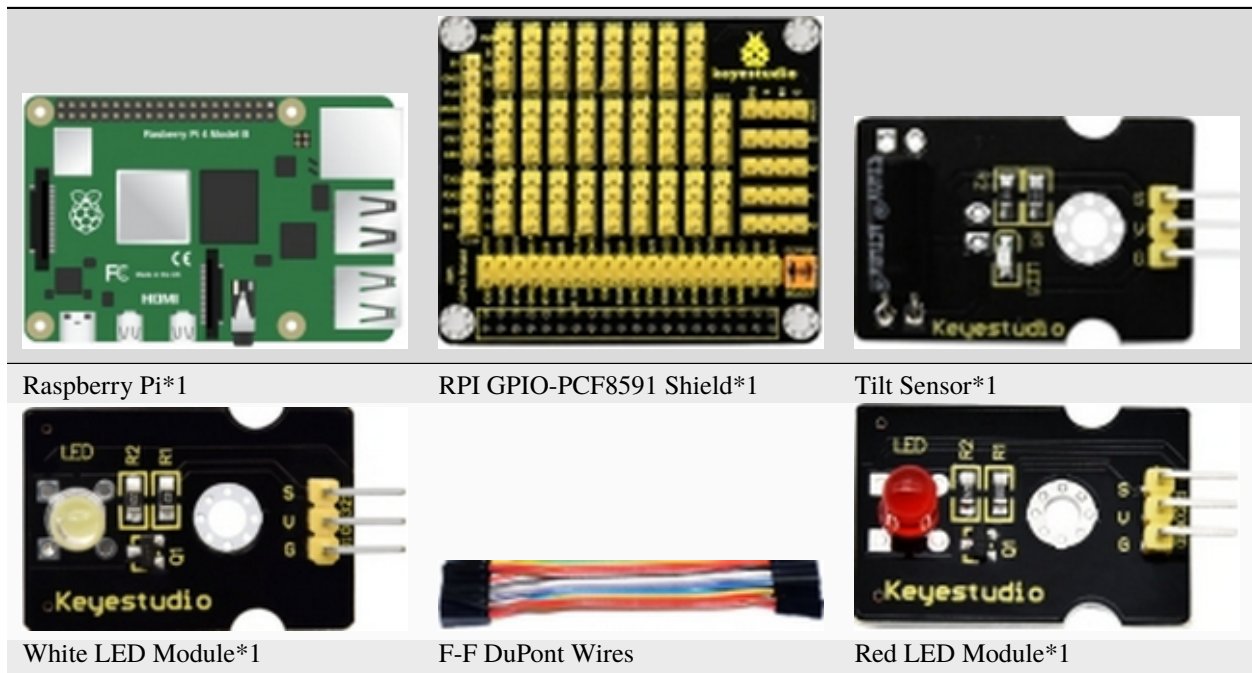
## 6.4.13 Project 13 Electronic Hourglass

### 1. Description

An hourglass (or sand glass, sand timer, sand clock or egg timer) is a device used to measure the passage of time. It comprises two glass bulbs connected vertically by a narrow neck that allows a regulated flow of a substance (historically sand) from the upper bulb to the lower one.

Typically the upper and lower bulbs are symmetric so that the hourglass will measure the same duration regardless of orientation. The specific duration of time a given hourglass measures is determined by factors including the quantity and coarseness of the particulate matter, the bulb size, and the neck width.

### 2. Components



### 3. Component Knowledge

#### Ball Tilt Sensor

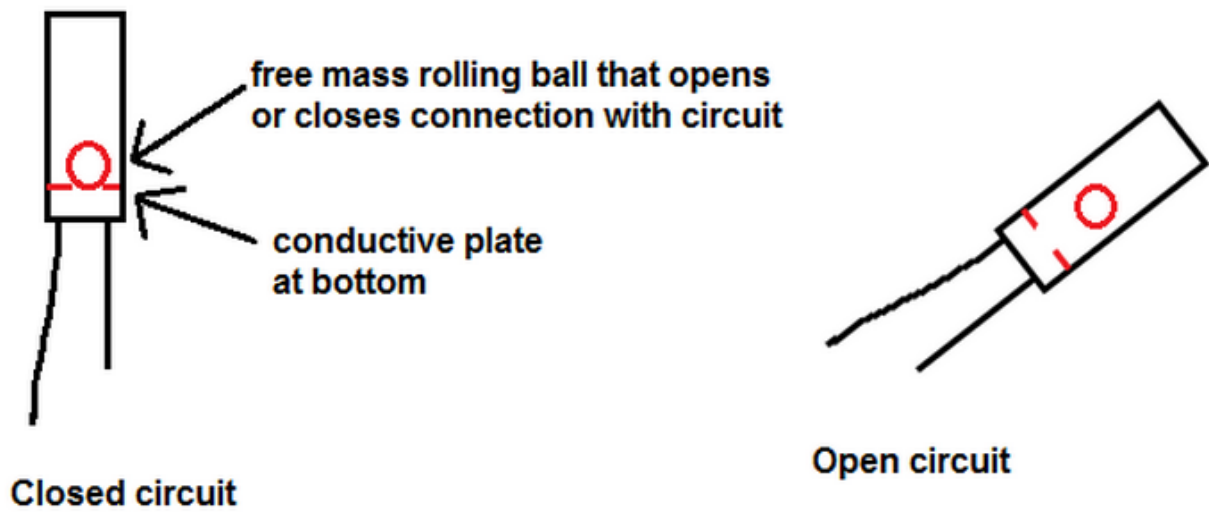
Tilt sensors (tilt ball switch) allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out.

The tilt-switch twig is the equivalent of a button, and is used as a digital input. Inside the tilt switch is a ball that make contact with the pins when the case is upright. Tilt the case over and the balls don't touch, thus not making a connection. When the switch is level it is open, and when tilted, the switch closes.

It can be used for orientation detection, alarm device or others.

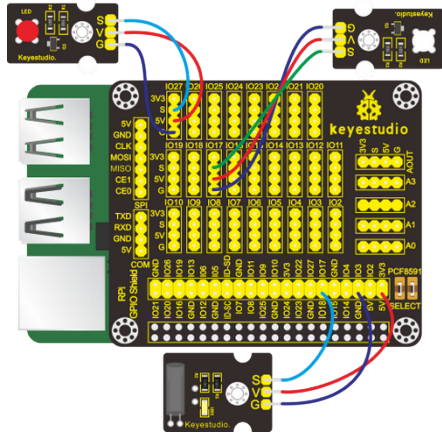


Here is the principle of tilt sensor to illustrate how it works:



4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Tilt Sensor	RPI GPIO-PCF8591 Shield
S	SIO27	S	SIO18
V	5V	V	5V
G	G	G	GND
White LED Module	RPI GPIO-PCF8591 Shield		
S	SIO17		
V	5V		
G	G		



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 13_ball_Tilt.py
```

## 6. Test Results

Led1 will brighten gradually and led2 will gradually darken when the two pins of the tilt sensor tilt; otherwise, when this sensor is tilt to another side or placed horizontally, led1 will get dim and led2 will get bright.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#define led pin
led1Pin = 17
led2Pin = 27
#define Ball Tilt Sensor Pin
tiltPin = 18

GPIO.setmode(GPIO.BCM) #use BCM unmbers
GPIO.setup(led1Pin,GPIO.OUT) #set the ledPin OUTPUT mode
GPIO.setup(led2Pin,GPIO.OUT)
GPIO.output(led1Pin,GPIO.HIGH) # make ledPin output HIGH level
GPIO.output(led2Pin,GPIO.LOW) # make ledPin output LOW level
GPIO.setup(tiltPin,GPIO.IN,GPIO.PUD_UP)
pwm1 = GPIO.PWM(led1Pin,1000) #create a pwm1 instance
pwm1.start(0) #start pwm1
pwm2 = GPIO.PWM(led2Pin,1000) #create a pwm2 instance
pwm2.start(0) #start pwm2
val1 = 50
val2 = 50
```

(continues on next page)

(continued from previous page)

```

while True:
    if not GPIO.input(tiltPin):
        val1 = val1 + 1
        val2 = val2 - 1
        if (val1 >= 100): #Limit PWM value to no more than 100
            val1 = 100
        if (val2 < 0):    #Limit PWM value not less than 0
            val2 = 0
        print("led1 = %1.0f" %(val1))
        pwm1.ChangeDutyCycle(val1) #change the frequency
        pwm2.ChangeDutyCycle(val2)
        sleep(0.1)
    else:
        val1 = val1 - 1
        val2 = val2 + 1
        if (val1 < 0):
            val1 = 0
        if (val2 >= 100):
            val2 = 100
        print("led2 = %1.0f" %(val2))
        pwm1.ChangeDutyCycle(val1)
        pwm2.ChangeDutyCycle(val2)
        sleep(0.1)

pwm1.stop() #stop pwm1

GPIO.cleanup() #release all GPIO


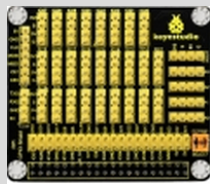



```

## 6.4.14 Project 14 Collision Alarm

### 1. Description

We use collision sensors to detect if there is a collision. When the object hits the metal switch of the sensor, the sensor will output a low level signal. When the metal switch is not touched, it will keep a high level. In this project, the cooling of the collision sensor will be used to control the buzzer.

### 2. Components

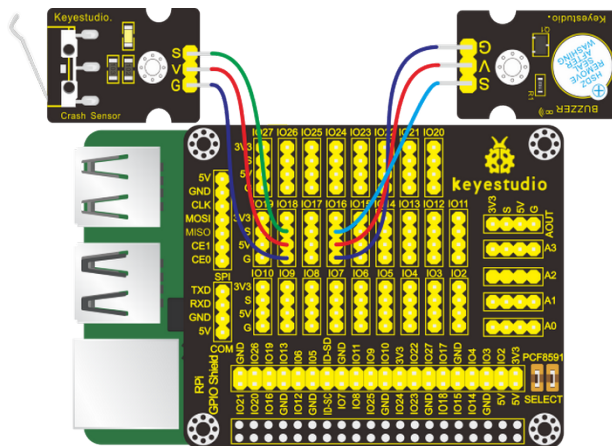
				
Raspberry Pi*1	RPI GPIO-PCF8591 Shield*1	Active Buzzer*1	Collision Sensor*1	F-F DuPont Wires

### 3. Component Knowledge

This is a common collision sensor, which mainly uses a tact switch. When the tact switch is touched by an object and the sensor signal is low, and the LED will be on; otherwise, the sensor signal is high level, and the LED will be off.

### 4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	Collision Sensor	RPI GPIO-PCF8591 Shield
S	SIO16	S	SIO18
V	5V	V	5V
G	G	G	G



### 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 14_crash_buzzer.py
```

### 6. Test Results

When you press the tact switch, the buzzer will emit a sound and the terminal will print low level 0. Otherwise, the buzzer will make no sounds, the terminal will print high level 1.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#active buzzer pin
buzPin = 16
#crash pin
crashPin = 18

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(buzPin,GPIO.OUT) #set buzPin OUTPUT mode
GPIO.setup(crashPin,GPIO.IN,GPIO.PUD_UP) # set crashPin INPUT mode and crashPin to PULL_
↳UP

while True:
    val = GPIO.input(crashPin)
    print(val)
    if(val == 0): #Judge whether the metal shrapnel is pressed
        GPIO.output(buzPin,GPIO.HIGH) #Buzzer ring
    else:
        GPIO.output(buzPin,GPIO.LOW) #buzzer off

GPIO.cleanup() # Release all GPIO
```

## 6.4.15 Project 15Line Tracking Sensor

### 1. Description

The smart car we launch can follow black lines to move. The key component is a line tracking sensor. In this lesson, we will learn it.

### 2. Components

				
Raspberry Pi*1	RPI GPIO-PCF8591 Shield*1	Red LED Module*1	Line Tracking Sensor*1	F-F DuPont Wires

3. Component Knowledge

Line Tracking Sensor

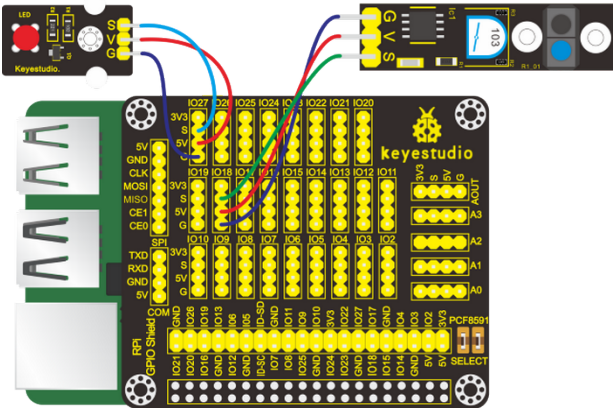
Line tracking sensor is an infrared sensor that can detect black and white objects. Its working principle is that the strength of the reflected signal is converted into a current signal.

It will be high level when detecting the black object; however, it will be low level when detecting the white object. Additionally its detection height is 0 ~ 3cm. In the circuit you can use the knob potentiometer to adjust the sensitivity



4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Line Tracking Sensor	RPI GPIO-PCF8591 Shield
S	SIO27	S	SIO18
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 15_tracking.py
```

## 6. Test Results

When it detects a black line(or no object is detected), LED will be off and high level 1 will be output; otherwise, LED will be on and low level 0 will be output.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#led pin
ledPin = 27
#tracking pin
trackingPin = 18

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(ledPin,GPIO.OUT) #set ledPin OUTPUT mode
GPIO.setup(trackingPin,GPIO.IN) # set trackingPin INPUT mode

while True:
    val = GPIO.input(trackingPin)
    print(val);
    if(val == 0): #Judge whether the white line is detected
        GPIO.output(ledPin,GPIO.HIGH) #led on
    else:
        GPIO.output(ledPin,GPIO.LOW) #led off



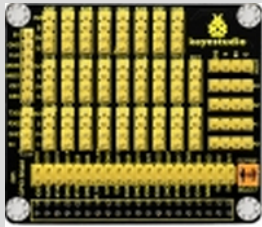

GPIO.cleanup() # Release all GPIO
```

## 6.4.16 Project 16Photo Interrupter Module

### 1. Description

In our daily life, we often need to count and take measurements. But how? The combination of light interrupter module and Raspberry Pi can do the trick. In the project, we will count with the photo interrupter module.

2. Components



Raspberry Pi\*1RPI GPIO-PCF8591 Shield\*1Photo Interrupter Module\*1F-F DuPont Wires

3. Component Knowledge

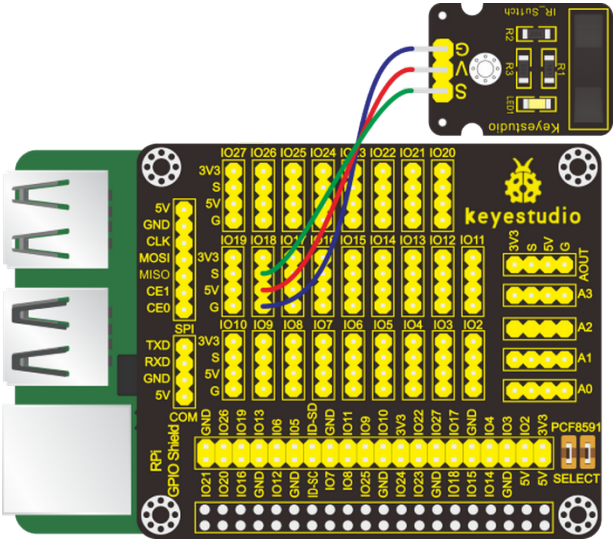
Photo Interrupter Module

It is a module which is equipped with a light emitting elements and light receiving elements aligned facing each other in a single package. It is based on the principle that the light passing through the U-shaped area will encounter blockage. Therefore, it is widely used in speed measurements, positioning count, small household appliances, optical limit switches, target detection and other fields.

If an object constantly passes through the U-shaped area of the photo interrupter module, the signal it outputs will shows constant changes between high and low levels. Therefore, we can count and measure speed by calculating the amount of high level and low level occurring.

4. Schematic Diagram

Light Interrupter	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G





## 5. Run Example Code

Input the following commands in the terminal and press “Enter”:

```
cd /home/pi/pythonCode_A
python 16_count_photofracture.py
```

## 6. Test Results

When the object pass through the U type groove on the light interrupter, the terminal will print numbers.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

photofracture = 18 #set photofracturePin
val = 0 #photofracture variables
count = 0 #Record the number of photofracture
flag = 0 #Odd even variable
GPIO.setmode(GPIO.BCM) # use BCM numbers

GPIO.setup(photofracture,GPIO.IN) #set the photofracturePin INPUT mode

while True:
    val = GPIO.input(photofracture) #Receive photofracture value
    #print("photofracture = %d"%(val))
    if(val == 0): #if light is broken
        sleep(0.01)
        val = GPIO.input(photofracture) #Receive photofracture value
        if(val == 1): #light is not broken
            count = count + 1 #Count the number of light is broken
            print("count = %d" %count)

    flag = count % 2 #Remainder 2 ,Even is 0, odd is 1

GPIO.cleanup() #release all GPIO
```




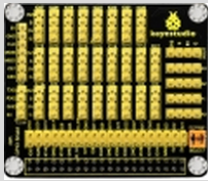

## 6.4.17 Project 17Magnetic Detection

### 1. Description

In this chapter, you can use the Hall sensor featuring high sensitivity, fast response, sound temperature performance and high reliability.

In this project, you will learn how to use a Hall magnetic sensor to control on and off of external LEDs.

2. Components



Raspberry Pi\*1      RPI      GPIO-      Red      LED      Mod-      Hall Magnetic Sen-      F-F DuPont Wires  
PCF8591 Shield\*1      module\*1      sor\*1

3. Component Knowledge

Hall Magnetic Sensor

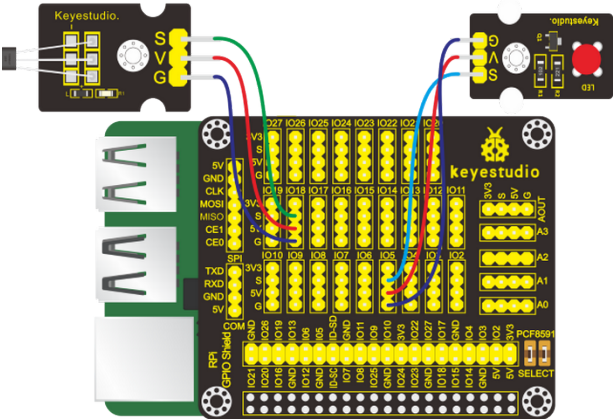
Its main component is A3144E, which is an electronic magnetic device and an active device. It can use the magnetic field and the Hall effect to achieve non-contact control.

Since it is a chip, its lifespan is infinite theoretically. The sensor can be used to detect magnetic fields and output digital signals.

It can sense magnetic materials within a detection range of around 3 cm. Note that it can only detect whether there is a magnetic field nearby rather than the strength of the magnetic field.

4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Hall Magnetic Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 17_Hall_magnetic.py
```

## 6. Test Results

When the magnetic bead is placed near the hall magnetic sensor and detected by the hall sensor, the LED will be on, the terminal will print“magnetic”; on the contrary, the LED will be off and the terminal will “nonmagnetic”.

Note: Press Ctrl + C on keyboard to exit code running

## 7. Example Code

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

ledPin = 5  #set led pin
hallPin = 18 #set hall magnetic pin
GPIO.setup(ledPin,GPIO.OUT)
GPIO.setup(hallPin,GPIO.IN)

while True:                                ##loop
    if GPIO.input(hallPin):                 #When Magnetic is not detected
        GPIO.output(ledPin,GPIO.LOW)      #turn off the led
        print("nonmagnetic")
    else:
        GPIO.output(ledPin,GPIO.HIGH)     #turn on the led
        print("magnetic")
```

## 6.4.18 Project 185V Relay


### 1. Description

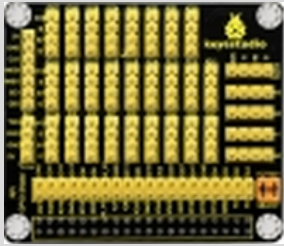
From a safety perspective, we specially designed this relay module with NO (normally open) and NC (normally closed) terminals. In this lesson, we will learn a special and easy-to-use switch, which is the relay module. Use the relay to start the motor.


In daily life, the electronic device is driven by 220V AC and controlled by switch. People will be in danger once the electricity leakage happens, connecting switch to 220V AC directly.

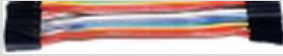
Therefore, we design a relay module with NO and NC ends. Let's get started.

2. Components









Raspberry Pi\*1

RPI GPIO-PCF8591 Shield\*1

5V Relay Module\*1

F-F DuPont Wires

3. Component Knowledge

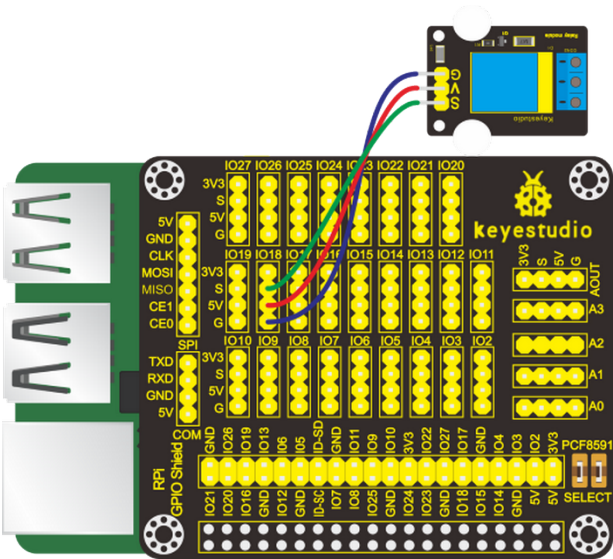
**Relay:** It is an “automatic switch” that uses a small current to control the operation of a large current.

Control input voltage: 5V

Rated load: 5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

4. Schematic Diagram

Relay Module	RPI GPIO-PCF8591 Shield
S	SIO18
V	5V
G	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 18_relay.py
```

## 6. Test Results

The light of the relay module will flash.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

relayPin = 18    #define relay pin

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(relayPin,GPIO.OUT)

while True:
    GPIO.output(relayPin,GPIO.HIGH)    #Starting relay
    print("turn on")
    sleep(2)
    GPIO.output(relayPin,GPIO.LOW)    #Close relay
    print("turn off")
    sleep(1)

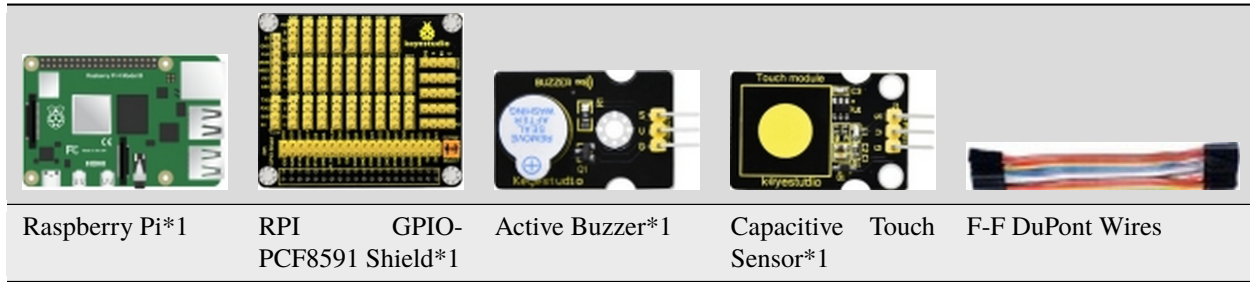
GPIO.cleanup()
```

## 6.4.19 Project 19: Touch capacitive Alarm

### 1. Description

Touch-sensitive alarm is very commonplace in daily life, especially found in home anti-theft and car anti-theft systems. When someone touches the alarming mental material, the device alarms to warn people. And it is of high sensitivity and high reliability evidenced by issuing alarm the moment it is touched.

## 2. Components



## 3. Component Knowledge

### Capacitive Touch Sensor

It mainly uses touch detection IC and can be found in many electronic devices. It uses the most popular capacitive sensing technology, just like the smart buttons on your phone. The touching area of this small sensor can feel the touch of humans and metals by responding with high or low level. It can still detect the touch though covered by a piece of paper and cloth. The sensitivity reduces with the increase of items between the touch-sensitive area and the object performing the touch.

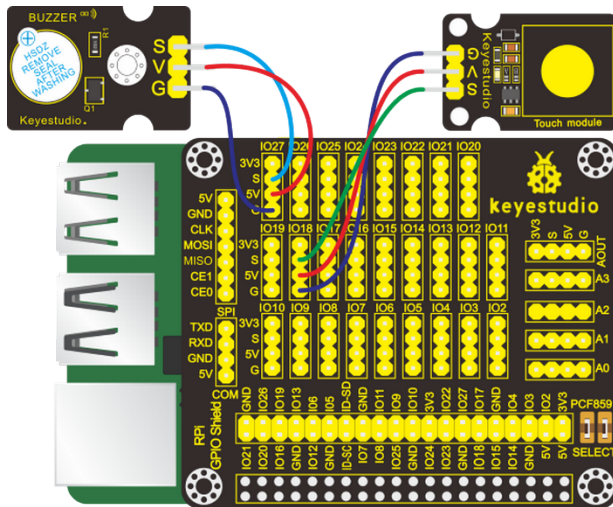
The touch detection IC is designed to replace the traditional button with a variable area key, featuring low power consumption and wide operating voltage.

When the module is powered up, it needs a stabilization time of about 0.5 sec. During this time period, do not touch the keypad. At this time, all functions are disabled, and self-calibration is always performed. No touching the key, the recalibration period is about 4.0sec.

Capacitive touch sensors are used in many devices such as digital audio players, computer displays, mobile phones, mobile devices, tablets and others.

## 4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	Capacitive Touch Sensor	RPI GPIO-PCF8591 Shield
S	SIO27	S	SIO18
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 19_touch_alarm.py
```

## 6. Test Results

When we touch the touch area of the capacitive touch sensor, the terminal will print the digital signal 1 and buzzer will emit sounds; otherwise, the buzzer won't make sounds, the digital signal 0 will be output.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#active buzzer
buzPin = 27
#touch pin
touchPin = 18

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(buzPin,GPIO.OUT) #set buzPin OUTPUT mode
GPIO.setup(touchPin,GPIO.IN) # set touchPin INPUT mode

while True:
    val = GPIO.input(touchPin)
    print(val)
    if(val == 1): #Judge whether the touch area is touched
        GPIO.output(buzPin,GPIO.HIGH) #Buzzer ring
    else:
```

(continues on next page)

(continued from previous page)

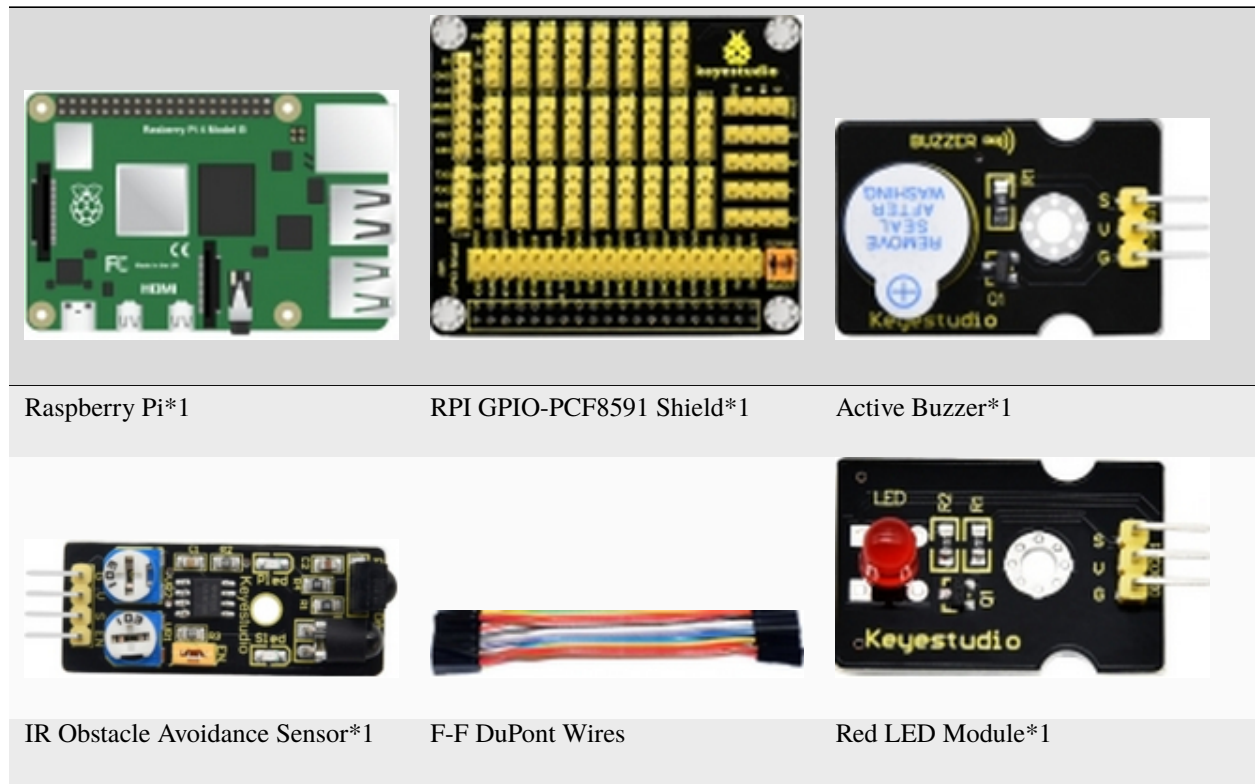
```
GPIO.output(buzPin,GPIO.LOW)    #Buzzer off
GPIO.cleanup() # Release all GPIO
```

## 6.4.20 Project 20Obstacle Avoidance Sensor

### 1. Description

In this chapter, we will introduce the obstacle avoidance.

### 2. Components



### 3. Component Knowledge

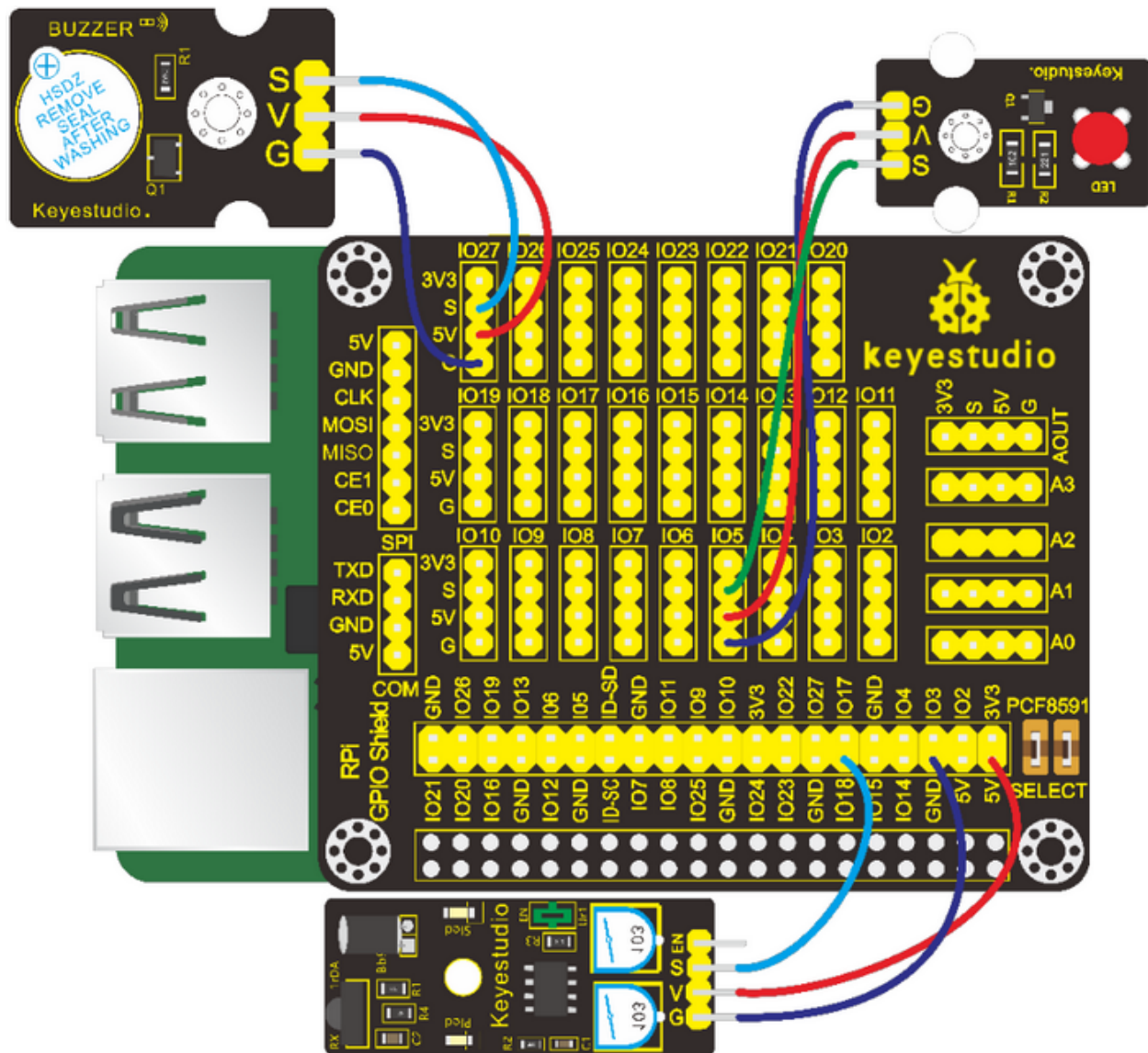
#### IR Obstacle Avoidance Sensor

It has a pair of infrared emissions and receiving tubes. When encountering an obstacle (reflective surface), the infrared light will be reflected back, and the signal terminal will output low level(0). Flat (0). If no obstacle is detected, the emitted infrared rays will weaken as the distance value increases, eventually disappear, the receiving tube cannot receive the infrared ray, and the sensor signal terminal will output high level (1). In this case, this sensor can determine whether there is an obstacle in front. You can rotate the potentiometer knob on the sensor to adjust the detection distance. The effective distance 2-40cm, working voltage is 3.3V-5V.



#### 4. Schematic Diagram

Active Buzzer	RPI Shield	GPIO-PCF8591	IR Obstacle Avoidance Sensor	RPI Shield	GPIO-PCF8591
S	SIO27		S	SIO18	
V	5V		V	5V	
G	G		G	G	
Red LED Module	RPI Shield	GPIO-PCF8591			
S	SIO5				
V	5V				
G	G				



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”

```
cd /home/pi/pythonCode_A
```

```
python20_obstacle_avoidance.py
```

## 6. Test Results

When the obstacle avoidance sensor detects an obstacle, the terminal will print the digital signal 0, and the buzzer will emit sound, and the LED will blink; otherwise, the terminal will print the digital signal 1, LED will be off and no sound will be emitted.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#active buzzer pin
buzPin = 27
#led pin
ledPin = 5
#obstacle avoidance pin
obstaclePin = 18

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(buzPin,GPIO.OUT) #set buzPin OUTPUT mode
GPIO.setup(ledPin,GPIO.OUT) #set ledPin OUTPUT mode
GPIO.setup(obstaclePin,GPIO.IN) # set obstacle avoidance Pin INPUT mode

while True:
    val = GPIO.input(obstaclePin)
    print(val)
    if(val == 0): #Judge whether obstacle avoidance is detected
        GPIO.output(buzPin,GPIO.HIGH) #Buzzer ring
        GPIO.output(ledPin,GPIO.HIGH) #led on
        sleep(0.2)
        GPIO.output(ledPin,GPIO.LOW) #led off
        sleep(0.1)
    else:
        GPIO.output(buzPin,GPIO.LOW) #Buzzer off
        GPIO.output(ledPin,GPIO.LOW) #led off

GPIO.cleanup() # Release all GPIO
```

### 6.4.21 Project 21 Reed Switch Module

#### 1. Description

In this project, you will learn how to use a reed switch module and a Raspberry Pi to detect magnetic fields and control an LED.



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 21_reed_switch.py
```

## 6. Test Results

When the reed sensor detects that the magnetic field, the terminal will print the digital signal 0, while the LED will be on; On the contrary, the digital signal 1 will be output, and the LED will be off.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
from time import sleep

#led pin
ledPin = 5
#reed switch pin
reedPin = 18

GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setup(ledPin,GPIO.OUT) #set ledPin OUTPUT mode
GPIO.setup(reedPin,GPIO.IN) # set reed switch Pin INPUT mode

while True:
    val = GPIO.input(reedPin)
    print(val)
    if(val == 0): #Judge whether magnetism is detected
        GPIO.output(ledPin,GPIO.HIGH) #led on
    else:
        GPIO.output(ledPin,GPIO.LOW) #led off

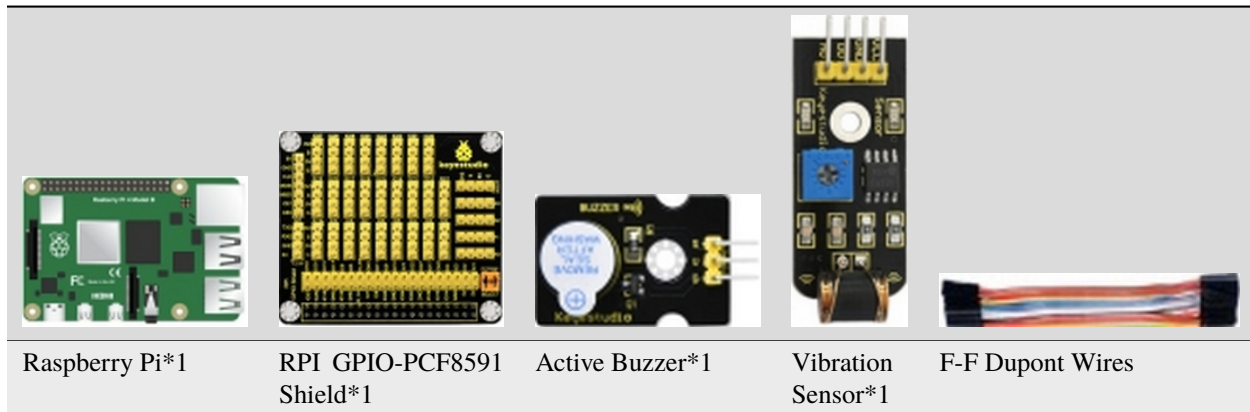
GPIO.cleanup() # Release all GPIO
```

## 6.4.22 Project 22Vibration Sensor

### 1. Description

We often encounter vibration alarms. In this project, let's learn to use a vibration sensor and buzzer to make a simple vibration alarm.

## 2. Components



## 3. Component Knowledge

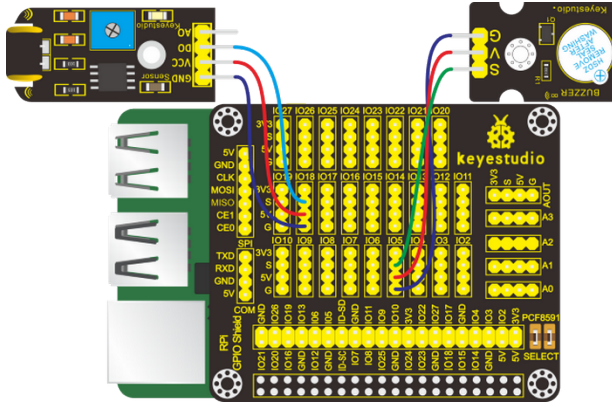
### Vibration Sensor

This is a commonly used vibration module/sensor. It has non-directional operation characteristics, which means it can be triggered to work by forces from any angles. The fully sealed package makes it waterproof and dustproof. And it is suitable for triggering in small current circuits.

After powering up the sensor, when it is not triggered by any forces, the circuit is openOFF, the signal end outputs high level and the LED on it remains off; when it is activated by an external force to reach its vibration threshold, the circuit is closeON, the signal end outputs low level and the LED on it lights up; and when the force exerted dies out, the circuit returns to open (OFF)state. The sensitivity of the sensor can be altered by rotating the potentiometer on it.

## 4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	Vibration Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SIO18
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 22_Vibrating_alarm.py
```

## 6. Test Results

After running the program, when the vibration sensor is triggered, the terminal keeps printing “buzzer ring.....buzzer off”and the buzzer rings constantly; otherwise, the terminal prints “...buzzer off”and the buzzer becomes silent.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO

buzPin = 5      # pin5 --- buzzer
vibPin = 18     # pin18 --- vibration sensor

buz_status = 0
def setup():
    GPIO.setmode(GPIO.BCM) # use BCM numbers
    GPIO.setwarnings(False)
    GPIO.setup(buzPin,GPIO.OUT) # Set buzPin's mode is output
    GPIO.setup(vibPin,GPIO.IN,pull_up_down=GPIO.PUD_UP) # Set vibPin's mode is input,
    ↪ and pull up to high level(3.3V)

def swbuz(ev=None):
    global buz_status
    buz_status = not buz_status
    GPIO.output(buzPin, buz_status) # switch buz status(ring-->off; off-->ring)
    if buz_status == 1:
        print 'buzzer ring...'
    else:
        print '...buzzer off'
```

(continues on next page)

(continued from previous page)

```
def loop():
    GPIO.add_event_detect(vibPin, GPIO.FALLING, callback=swbuz) # wait for falling
    while True:
        pass # Don't do anything

def destroy():
    GPIO.output(buzPin, GPIO.LOW) # buzzer off
    GPIO.cleanup() # Release resource

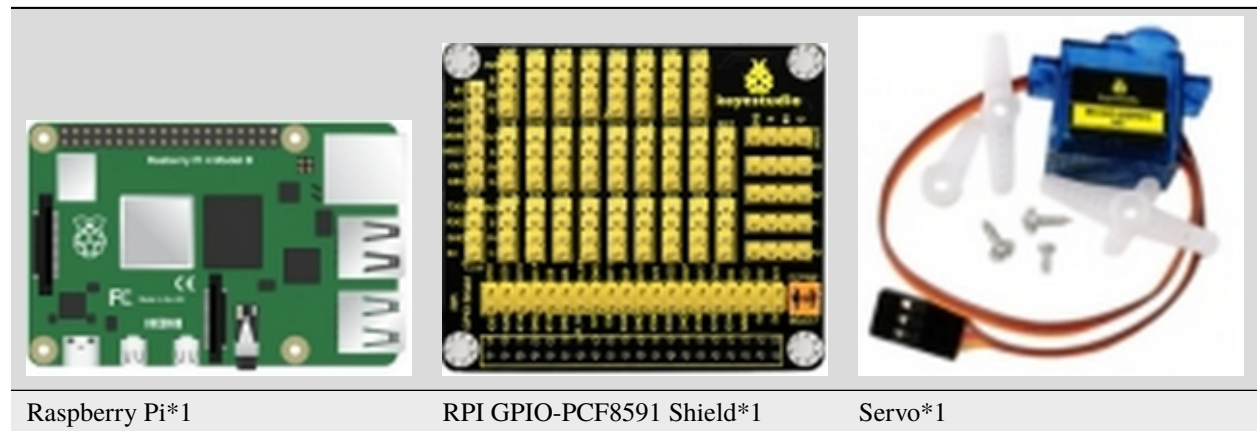
if __name__ == '__main__': # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

## 6.4.23 Project 23Servo

### 1. Description

Servo is applied widely, especially for robot like human robots and moving robots. In this lesson, we will learn how it works.

### 2. Components



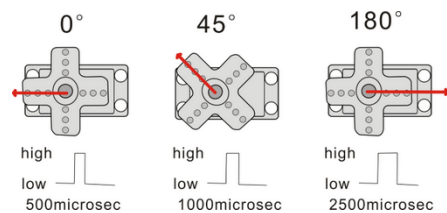


### 3. Component Knowledge

#### Servo:

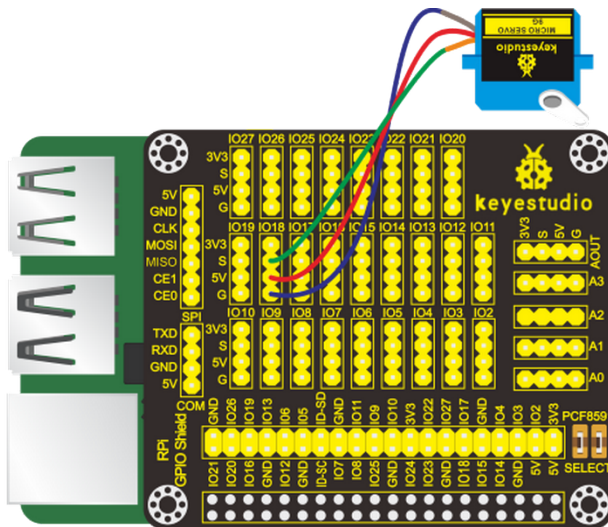
A location(angle) driver which can rotate a certain angle with high accuracy. It has three external wires which are brown, red and orange,. Brown one is grounded, red one is positive pole of power and orange one is signal wire.

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° to 180°. But note that for different brand motor, the same signal may have different rotation angle.



### 4. Schematic Diagram

Servo	RPI GPIO-PCF8591 Shield
Orange Wire	SIO18
Red Wire	5V
Brown Wire	G



## 5. Run Example Code

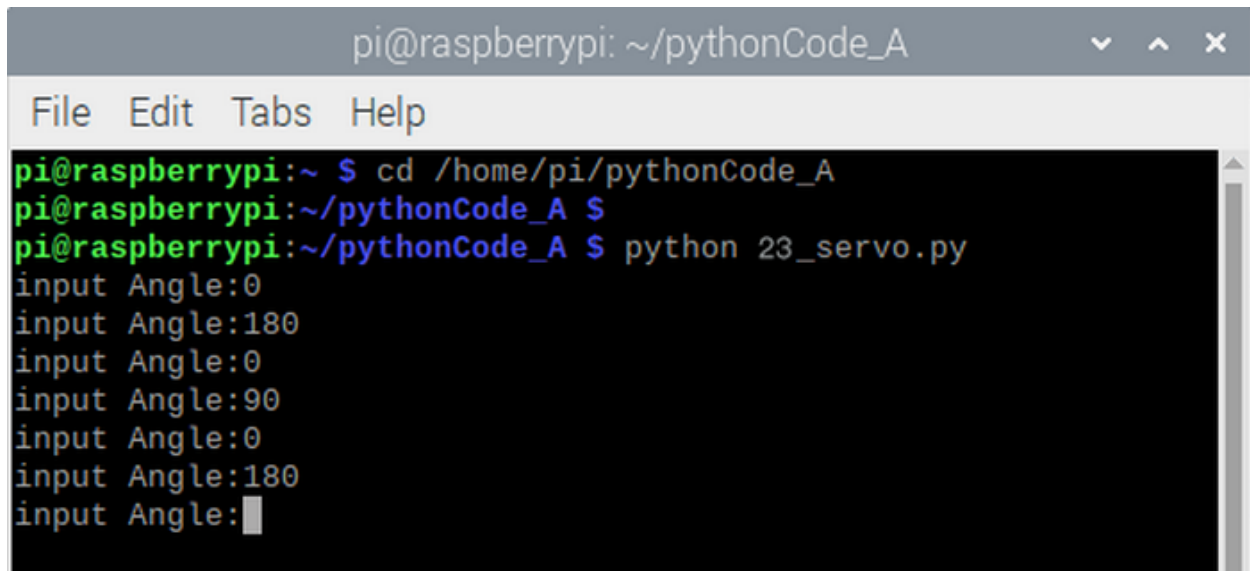
Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 23_servo.py
```

## 6. Test Results

Enter the angle value and servo rotates the corresponding value, as shown below:



The screenshot shows a terminal window titled 'pi@raspberrypi: ~/pythonCode\_A'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows the following commands and responses:

```
pi@raspberrypi:~ $ cd /home/pi/pythonCode_A
pi@raspberrypi:~/pythonCode_A $
pi@raspberrypi:~/pythonCode_A $ python 23_servo.py
input Angle:0
input Angle:180
input Angle:0
input Angle:90
input Angle:0
input Angle:180
input Angle:
```

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import time

servo_min_angle = 2.5  #define pulse duty cycle for minimun angle of servo
servo_max_angle = 12.5 #define pulse duty cycle for maximun angle of servo

servopin = 18  #servo Pin
GPIO.setmode(GPIO.BCM)  #BCM numbers

GPIO.setup(servopin,GPIO.OUT)
p = GPIO.PWM(servopin,50)  #set 50Hz , The working frequency of the steering gear is 50Hz
p.start(0)  # start PWM
time.sleep(2)

#define function, map a value from one range to another range
def map(angle, val1, val2, min_angle, max_angle):
    return (max_angle-min_angle)*(angle-val1)/(val2-val1)+min_angle
```

(continues on next page)

(continued from previous page)

```

while(True): #loop
    p.ChangeDutyCycle(0) #set
    time.sleep(0.4)
    b = input("input Angle:")
    b = int(b)
    c = map(b, 0, 180, servo_min_angle, servo_max_angle) #map angle from 0~180 to 2.5~
    ↪ 12.5
    p.ChangeDutyCycle(c)
    time.sleep(0.4)

p.stop()
GPIO.cleanup()

```

## 6.4.24 Project 24 Adjust the Brightness of LED

### 1. Description

Some of the lamps on market can be adjusted to display different brightness, which gives us better shopping experiences. And in this project, we will learn how to make this happen.

### 2. Components



### 3. Component Knowledge

#### PCF8591 A/D Conversion Chip

PCF8591 A/D conversion chip is installed at the back of the RPI GPIO-PCF8591 shield and its chip's voltage resolution is 5V/2550.01961. However, Raspberry Pi doesn't AD/DA function.

The external shield with AD/DA function is needed if Raspberry Pi is interfaced with analog sensors.

We use a pcf8591 AD/DA converter which requires iic communication.

Then we need to open iic communication of Raspberry Pi, as shown below:

Input `sudo raspi-config` in the terminal and press Enter to open the interface of Raspberry Pi.

```

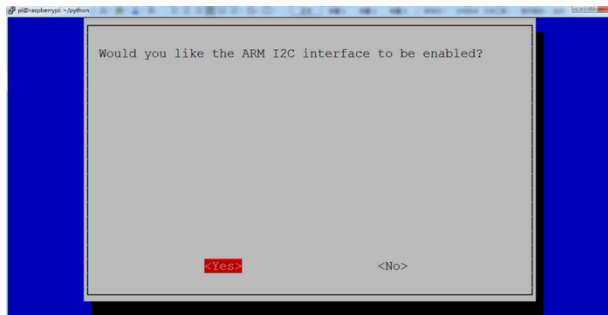
pi@raspberrypi:~/python $ sudo raspi-config

```

Press key ↑, ↓, ← and → and then press Enter

```
pi@raspberrypi: ~  
File Edit Tabs Help  
Raspberry Pi 4 Model B Rev 1.1  
Raspberry Pi Software Configuration Tool (raspi-config)  
1 Change User Password Change password for the current user  
2 Network Options Configure network settings  
3 Boot Options Configure options for start-up  
4 Localisation Options Set up language and regional settings to match your  
5 Interfacing Options Configure connections to peripherals  
6 Overclock Configure overclocking for your Pi  
7 Advanced Options Configure advanced settings  
8 Update Update this tool to the latest version  
9 About raspi-config Information about this configuration tool  
  
<Select> <Finish>
```

```
pi@raspberrypi: ~  
File Edit Tabs Help  
Raspberry Pi Software Configuration Tool (raspi-config)  
P1 Camera Enable/Disable connection to the Raspberry Pi Camera  
P2 SSH Enable/Disable remote command line access to your Pi using  
P3 VNC Enable/Disable graphical remote access to your Pi using Rea  
P4 SPI Enable/Disable automatic loading of SPI kernel module  
P5 I2C Enable/Disable automatic loading of I2C kernel module  
P6 Serial Enable/Disable shell and kernel messages on the serial conn  
P7 1-Wire Enable/Disable one-wire interface  
P8 Remote GPIO Enable/Disable remote access to GPIO pins  
  
<Select> <Back>
```

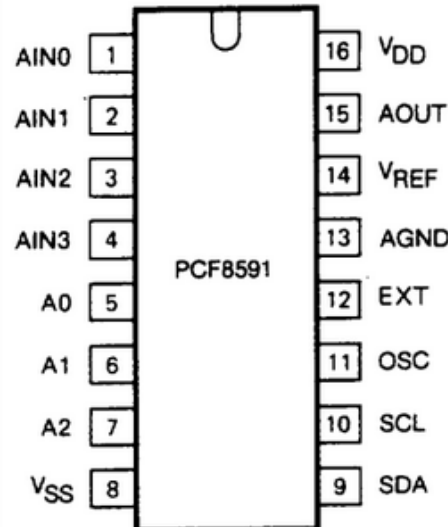


More information about I2C communication, check it in the link: <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>

### PCF8591 Pins:

More details about PCF8591 chip, you could look through chip specification folder .

From the below figure, PCF8591 has an analog output pin Aout and four analog input pin A0-A3.

SYMBOL	PIN	DESCRIPTION	TOP VIEW
AIN0	1	Analog inputs (A/D converter)	
AIN1	2		
AIN2	3		
AIN3	4		
A0	5	Hardware address	
A1	6		
A2	7		
Vss	8	Negative supply voltage	
SDA	9	I2C-bus data input/output	
SCL	10	I2C-bus clock input	
OSC	11	Oscillator input/output	
EXT	12	external/internal switch for oscillator input	
AGND	13	Analog ground	
Vref	14	Voltage reference input	
AOUT	15	Analog output(D/A converter)	
Vdd	16	Positive supply voltage	

Check the address of iic modulePCF8591of Raspberry Pi, enter command `i2cdetect -y 1` and press Enter.

The iic address of PCF8591 is 0x48.

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~ $

```

Used to read the address of pin A0~A3.

The address of analog output pin AOUT: 0x40, that is, 64 converting from hexadecimal to decimal.

A0 = 0x40 ##A0 —> port address

A1 = 0x41

A2 = 0x42

A3 = 0x43

### Adjustable Potentiometer

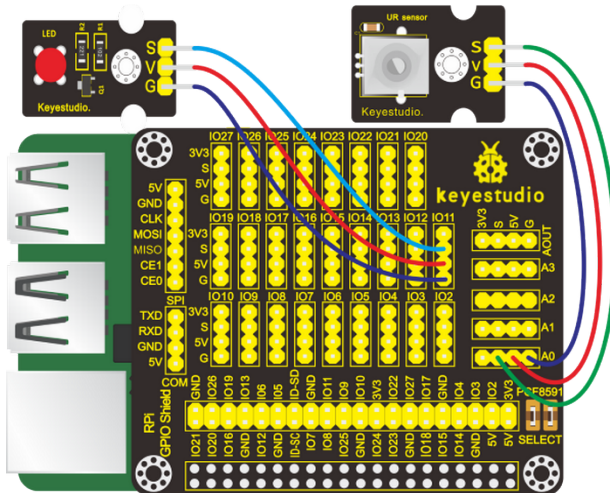
The rotary potentiometer means the change of resistance.

We could convert the resistance's change into the voltage's when setting circuit. Then, voltage changes will be output to GPIO port through module signals.

Wiring according to the below figure and rotate clockwise, resistance value reduces.

## 4. Schematic Diagram

Red LED Mod- ule	RPI Shield	GPIO-PCF8591	Adjustable Potentiome- ter	RPI Shield	GPIO-PCF8591
S	SIO11		S	SA0	
V	5V		V	5V	
G	G		G	G	



## 5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 24_potentiometer_LED.py
```

## 6. Test Results

Terminal prints the analog value read by adjustable potentiometer. The LED brightness will vary with the the rotary of potentiometer.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time
address = 0x48 #default address of PCF8591
bus=smbus.SMBus(1) #Create an instance of smbus
cmd=0x40 #command
# A0 = 0x40      ##A0 ----> port address
# A1 = 0x41
# A2 = 0x42
# A3 = 0x43

ledPin = 11
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin,GPIO.OUT)
GPIO.output(ledPin,GPIO.LOW)
p = GPIO.PWM(ledPin,100)
p.start(0)

def analogRead(chn):    #read ADC value,chn:0,1,2,3
```

(continues on next page)

(continued from previous page)

```

value = bus.read_byte_data(address,cmd+chn)
return value

def analogWrite(value):#write DAC value
    bus.write_byte_data(address,cmd,value)

def loop():
    while True:
        value = analogRead(0) #read the ADC value ofchannel 0
        analogWrite(value) #write the DAC value
        p.ChangeDutyCycle(value*100/255) #Convert ADC value to duty cycle of PWM
        voltage = value / 255.0 * 3.3 #calculate the voltage value
        print ('ADC Value : %d, Voltage : %.2f'%(value,voltage))
        time.sleep(0.01)

def destroy():
    bus.close()

if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

## 8. Explanation

smbus	Smbus is based on iic communication. We treat it as iic communication library.
bus.read_byte_data(address,cmd+chn)	Read the corresponding modules with iic addressaddress is the address of pcf8591 modulecmd+chn correspond to the address of analog port pcf8591: A0 = 0x40A1 = 0x41A2 = 0x42A3 = 0x43
bus.write_byte_data(address,cmd,value)	D/A analog value outputs, address is address of pcf8591 modulecmd outputs the address of pinsvalue: output value
Smbus library file	<a href="https://pypi.org/project/smbus2/0.1.2/">https://pypi.org/project/smbus2/0.1.2/</a>

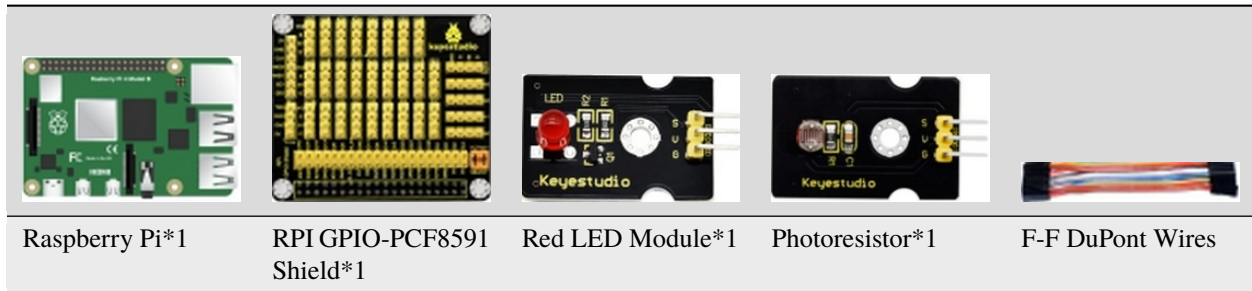


## 6.4.25 Project 25Photoresistor

### 1. Description

Photo resistor (Photoresistor) is a resistor whose resistance varies according to different incident light strength. It's made based on the photoelectric effect of semiconductor. In this lesson, let's explain how it works.

### 2. Components



### 3. Component Knowledge

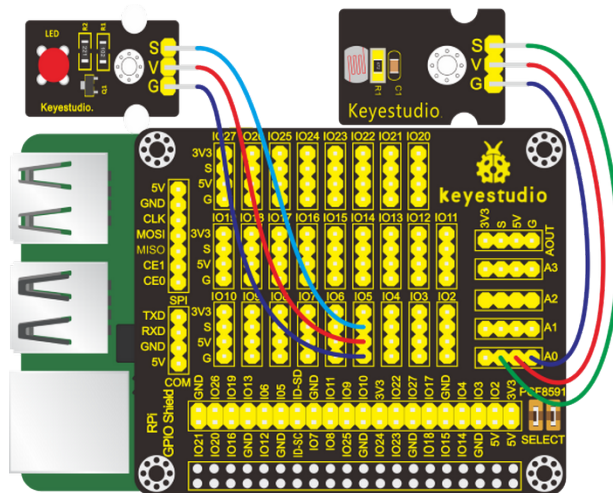
#### Photoresistor

Photo resistor (Photoresistor) is a resistor whose resistance varies according to different incident light strength. It's made based on the photoelectric effect of semiconductor. If the incident light is intense, its resistance reduces; if the incident light is weak, the resistance increases.

If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance.

### 4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Photoresistor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SA0
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

**Special Note:** The I2C communication method is used in the experiment. We need to check the iic address first (enter `command2cdetect -y 1` and press “Enter”). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/pythonCode_A
python 25_photo_sensor.py
```

## 6. Test Results

Terminal prints the value tested by photoresistor. LED will turn on if the ambient environment is dim; otherwise, LED will be off.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

led = 5
GPIO.setup(led, GPIO.OUT)

address = 0x48 ##address ---> device address
cmd = 0x40     ##DA converter command
A0 = 0x40      ##A0 ----> port address
A1 = 0x41
A2 = 0x42
```

(continues on next page)

(continued from previous page)

```

A3 = 0x43
bus = smbus.SMBus(1)          ##start the bus

def analogRead(count):        #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True:                   ##loop
    #Vout = 10                  ##10*0.0196=0.196V
    #bus.write_byte_data(address,cmd,Vout) ##DA converter
    value = analogRead(0)      ##read A0 data
    if(value<100):             #When the ambient brightness is less than 100,
    → the LED light will be on
        GPIO.output(led,GPIO.LOW)
    else:
        GPIO.output(led,GPIO.HIGH)

    print("data:%1.0f" %(value)) ##print data

time.sleep(0.5)                ##delay 0.5 second
GPIO.cleanup()

```

## 6.4.26 Project 26Sound-activated Light

### 1. Description

You might find the lights automatically on when you pass them, nevertheless, they will be off if the surrounding is quiet. Do you know why?

Actually, it is sound sensor that controls them on and off.

### 2. Components:

				
Raspberry Pi*1	RPI GPIO-PCF8591 Shield*1	Red LED Module*1	Analog Sound Sensor*1	F-F DuPont Wires

### 3. Component

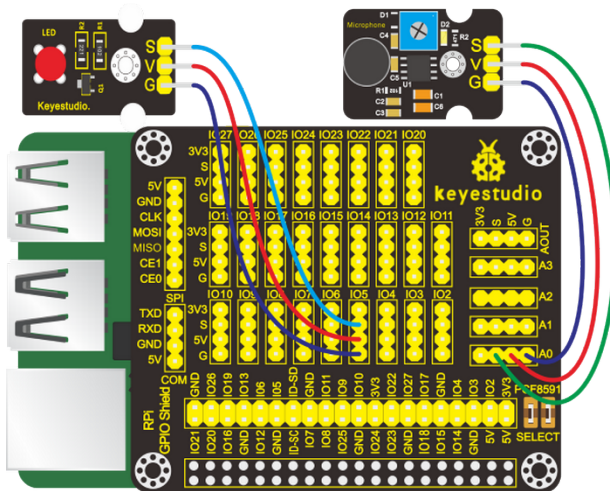
A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals.

It has a built-in capacitive electret microphone which is highly sensitive to sound. Sound waves cause the thin film of the electret to vibrate and then the capacitance changes, thus producing the corresponding changed voltage. Since the voltage change is extremely weak, it needs to be amplified. So it is converted into a voltage ranging from 0 to 5V, which is received by data acquisition unit after A/D adapter conversion and then sent to an MCU.

The module can be applied to noise monitoring in traffic artery, and detection of noises within the boundary of industrial enterprises, factories, and construction sites, detection of noises in urban regions, and noise detection and assessment of living surroundings.

### 4. Schematic Diagram

Red LED Module	RPI GPIO-PCF8591 Shield	Analog Sound Sensor	RPI GPIO-PCF8591 Shield
S	SIO5	S	SA0
V	5V	V	5V
G	G	G	G



### 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter command `i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 26_sound_led.py
```

## 6. Test Results

When you clap your hands suddenly, LED will be on; if you clap again, LED will be off.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

led = 5
GPIO.setup(led,GPIO.OUT)

address = 0x48 ##address ---> device address
cmd = 0x40 ##DA converter command
A0 = 0x40 ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1) ##start the bus

flag = 0
mode = 0

def analogRead(count): #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True: ##loop
    value = analogRead(0) ##read A0 data
    if(value>50):
        flag += 1
        mode = flag % 2
    if(mode == 0):
        GPIO.output(led,GPIO.LOW)
    else:
        GPIO.output(led,GPIO.HIGH)

    print("data:%1.0f" %(value)) ##print data
    time.sleep(0.05) ##delay 0.05 second

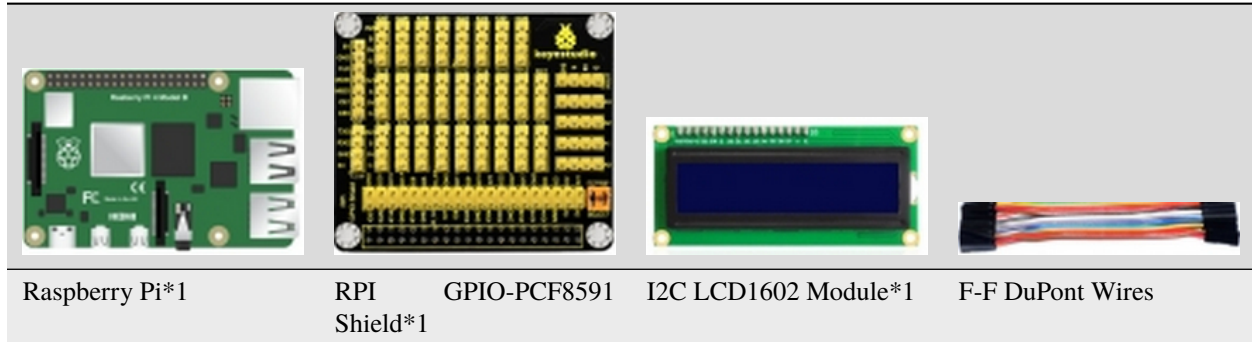
GPIO.cleanup()
```

## 6.4.27 Project 27LCD1602

### 1. Description

In this chapter, we will use a 1602 I2C module as a display and connect it to the Raspberry Pi. And we will show you how to control a 1602 LCD module.

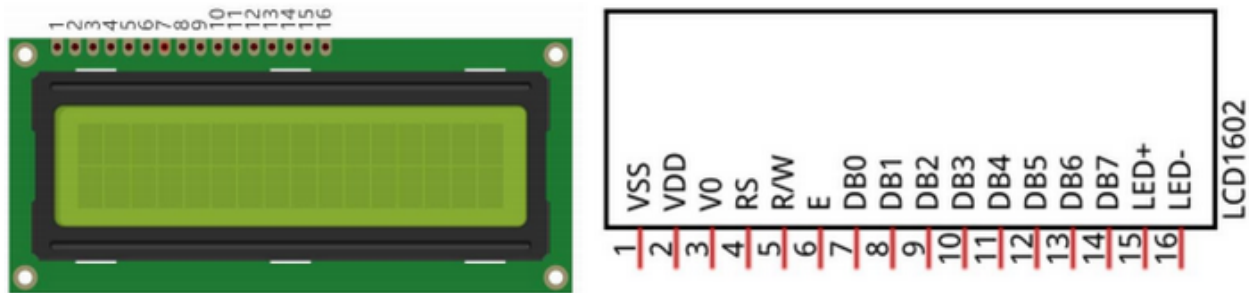
### 2. Components



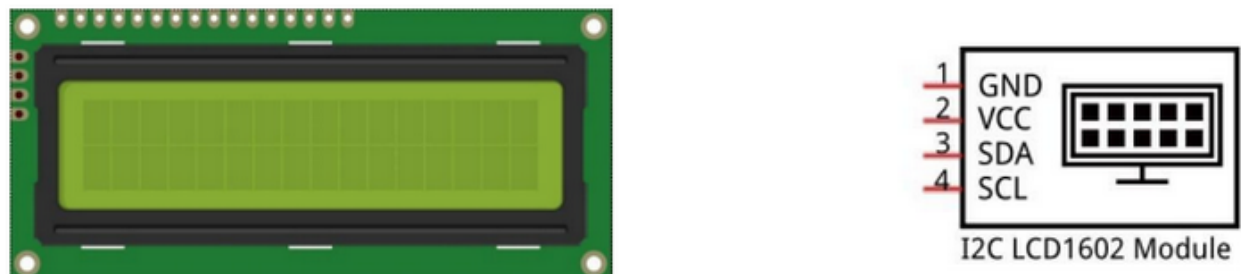
### 3. Component Knowledge

#### LCD1602 Display Module

The display has an LCD 1602 LCD display and I2C LCD. But we use an I2C LCD 1602 in this project. It can display characters of 16 column and 2 row. It also can display numbers, letters, symbols, ASCII codes, etc. As shown below:



The I2C LCD1602 display integrates an I2C interface, a connected serial input & parallel output. This makes us operate the LCD1602 with only 4 lines.

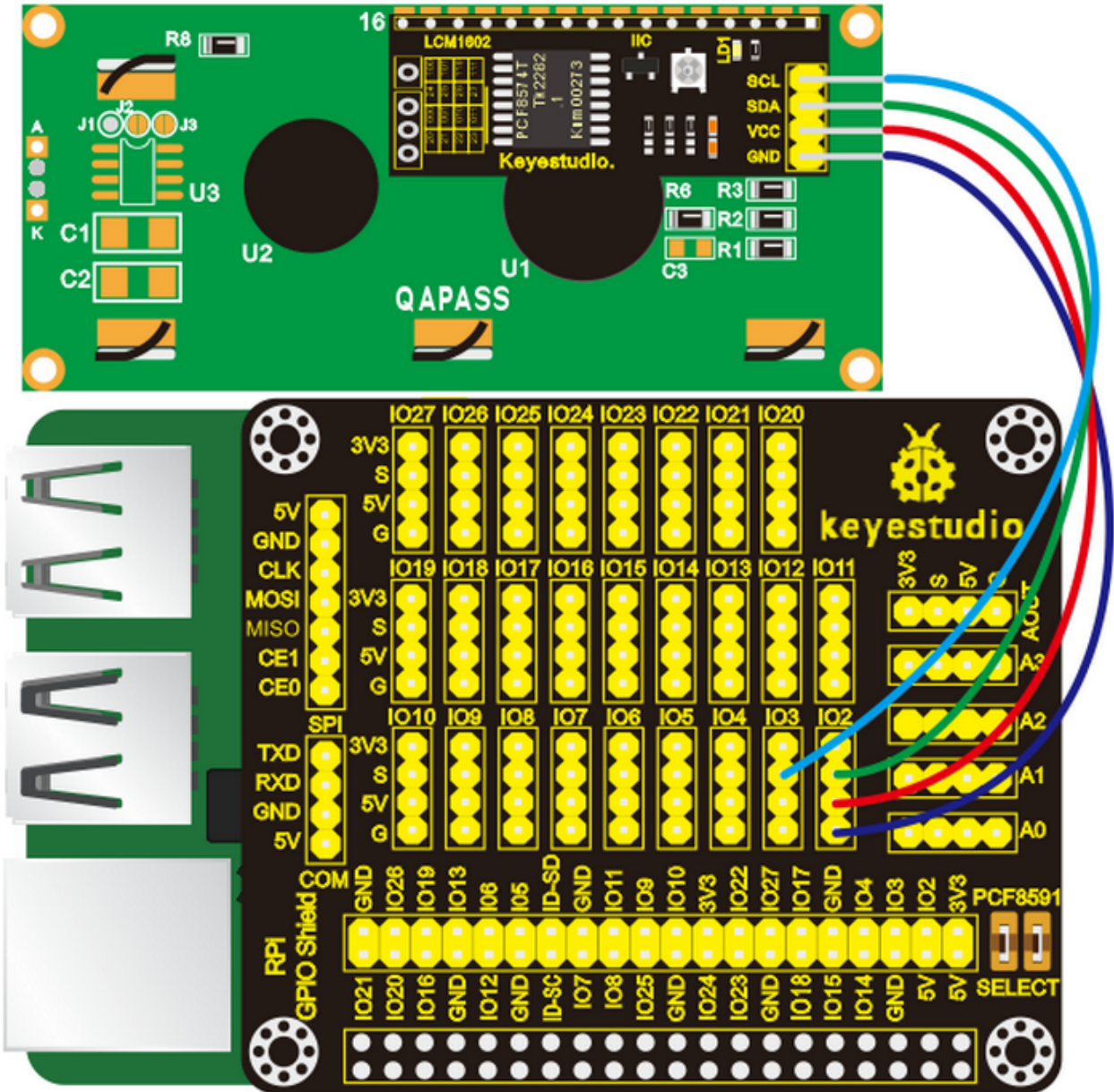


The IC chip used in this module is PCF8574T (PCF8574AT) whose default I2C address is 0x27 (0x3f). You can also view the RPI bus on your I2C device address by command “I2CDetect -y 1”.



4. Schematic Diagram

I2C LCD1602 Module	RPI GPIO-PCF8591 Shield
GND	GND
VCC	5V
SDA	IO2
SCL	IO3



## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first(enter command `i2cdetect -y 1` and press “Enter”). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/pythonCode_A/27_I2CLCD1602
```

```
python 27_I2CLCD1602.py
```

## 6. Test Results

The LCD1602 screen will display the CPU temperature and system time of your Raspberry Pi.

Note: after uploading code, if you can't see anything on the display or display is unclear, try to rotate the blue knob on the back of the LCD1602, adjust the contrast until the screen can clear the time and temperature.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
from PCF8574 import PCF8574_GPIO
from Adafruit_LCD1602 import Adafruit_CharLCD

from time import sleep, strftime
from datetime import datetime

def get_cpu_temp():      # get CPU temperature and store it into file "/sys/class/thermal/
    ↪ thermal_zone0/temp"
    tmp = open('/sys/class/thermal/thermal_zone0/temp')
    cpu = tmp.read()
    tmp.close()
    return '{:.2f}'.format( float(cpu)/1000 ) + ' C'

def get_time_now():      # get system time
    return datetime.now().strftime('%H:%M:%S')

def loop():
    mcp.output(3,1)      # turn on LCD backlight
    lcd.begin(16,2)      # set number of LCD lines and columns
    while(True):
        #lcd.clear()
        lcd.setCursor(0,0) # set cursor position
        lcd.message( 'CPU: ' + get_cpu_temp()+'\n' )# display CPU temperature
        lcd.message( get_time_now() ) # display the time
        sleep(1)

def destroy():
    lcd.clear()

PCF8574_address = 0x27 # I2C address of the PCF8574 chip.
PCF8574A_address = 0x3F # I2C address of the PCF8574A chip.
```

(continues on next page)



(continued from previous page)

```
# Create PCF8574 GPIO adapter.
try:
    mcp = PCF8574_GPIO(PCF8574_address)
except:
    try:
        mcp = PCF8574_GPIO(PCF8574A_address)
    except:
        print ('I2C Address Error !')
        exit(1)
# Create LCD, passing in MCP GPIO adapter.
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)


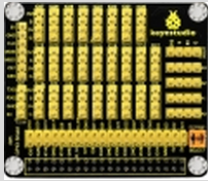



if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

6.4.28 Project 28Water Level Monitor

1. Description

If you have ever had a water heater explode or ever tried to make submersible electronics, then you know how important it is to detect when water is around. Let’s know more about water level sensor.

2. Components:

				
Raspberry Pi*1	RPI GPIO-PCF8591 Shield*1	Active Buzzer*1	Water Level Sensor*1	F-F DuPont Wires

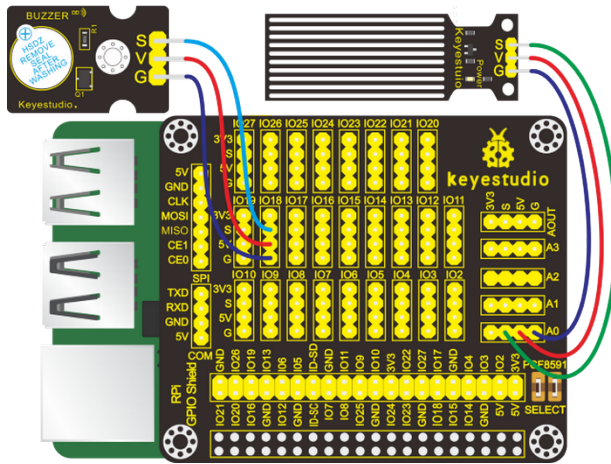
3. Component Knowledge

Water Level Sensor

Our water sensor is easy- to-use, portable and cost-effective, designed to identify and detect water level and water drop. This sensor measures the volume of water drop and water quantity through an array of traces of exposed parallel wires. It could convert water content to analog signals, and output analog value could be used by function of application. It has the features of low consumption as well.

#### 4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	Water Level Sensor	RPI GPIO-PCF8591 Shield
S	SIO18	S	SA0
V	5V	V	5V
G	G	G	G



#### 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter `commandi2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 28_water_buzzer.py
```

#### 6. Test Results

When water covers the detection part of the sensor, the buzzer will emit sounds.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

buz = 18
GPIO.setup(buz,GPIO.OUT)

address = 0x48 ##address ---> device address
cmd = 0x40 ##DA converter command
A0 = 0x40 ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1) ##start the bus

def analogRead(count): #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True: ##loop
    value = analogRead(0) ##read A0 data
    if(value>30):
        GPIO.output(buz,GPIO.HIGH)
    else:
        GPIO.output(buz,GPIO.LOW)

    print("data:%1.0f" %(value)) ##print data
    time.sleep(0.05) ##delay 0.05 second

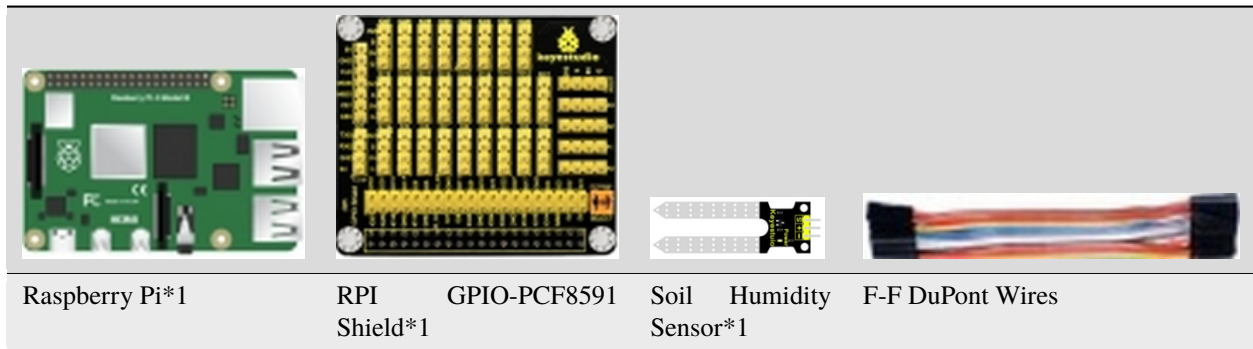
GPIO.cleanup()
```

## 6.4.29 Project 29Flower-watering Device

### 1. Description

The household plants are popular in many a communities. They will die if you forget to water them, how about making an automatic watering device?

## 2. Components



## 3. Component Knowledge

### Soil Humidity Sensor

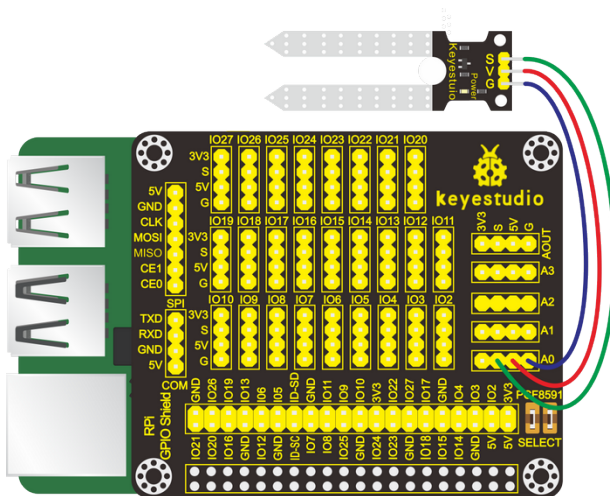
This is a simple soil humidity sensor aims to detect the soil humidity.

If the soil is in lack of water, the analog value output by the sensor will decrease; otherwise, it will increase. If you use this sensor to make an automatic watering device, it can detect whether your botany is thirsty to prevent it from withering when you go out.

Using the sensor with controller makes your plant more comfortable and your garden smarter. The soil humidity sensor module is not as complicated as you might think, and if you need to detect the soil in your project, it will be your best choice.

## 4. Schematic Diagram

Soil Humidity Sensor	RPI GPIO-PCF8591 Shield
S	SA0
V	5V
G	G



## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter command `i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 29_soil.py
```

## 6. Test Results

Insert the soil humidity sensor into the plant pot, then the terminal will print the soil humidity value.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

address = 0x48 ##address--->device address
cmd = 0x40     ##DA converter command
A0 = 0x40      ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1)          ##start the bus
while True:                  ##loop
    #Vout = 10                ##10*0.0196=0.196V
    #bus.write_byte_data(address,cmd,Vout) ##DA converter
    bus.write_byte(address,A0) ##which port of the device you want to access
    value = bus.read_byte(address) ##access the data
    print("data:%1.0f" %(value)) ##print data

time.sleep(0.5)              ##delay 0.5 second
GPIO.cleanup()
```


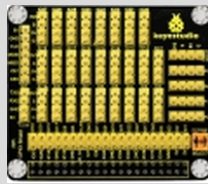



## 6.4.30 Project 30Temperature Alarm

### 1. Description

In the cold winter, in order to prevent the greenhouse vegetables from being frozen, the vegetable farmers will heat the greenhouse vegetables so that the temperature in the greenhouse is suitable.

In this project, we will learn to make a temperature alert model.

### 2. Components

				
Raspberry Pi*1	RPI GPIO-PCF8591 Shield*1	Active Buzzer*1	LM35 Temperature Sensor*1	F-F DuPont Wires

### 3. Component Knowledge

#### LM35 Temperature Sensor

The temperature sensor based on the semiconductor LM35 is a temperature sensor that is proportional to a total proportion of a Celsius temperature, and its output voltage is a temperature standard.

LM35DZ is a widely used temperature sensor. Since it adopts internal compensation, the output can start from 0 °C.

Sensitivity is 10 mV /°C;

The output temperature range is from 0°C100°C,

Conversion formula: output 0V when temperature is 1°C, the output voltage increases by 10 mV.

Operating voltage: 4-30V;

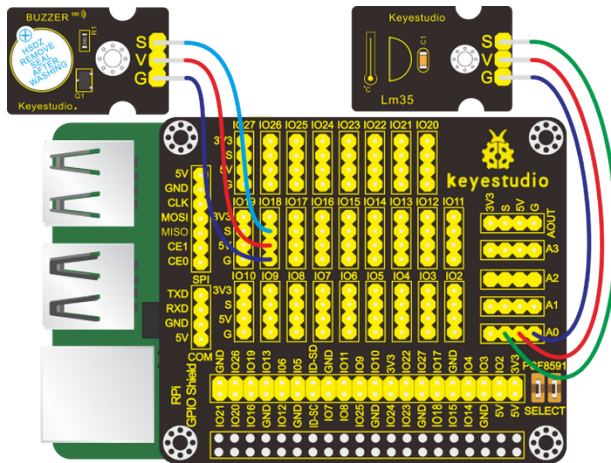
Accuracy:  $\pm 1^{\circ}\text{C}$ .

The maximum linear error:  $\pm 0.5^{\circ}\text{C}$ ;

The static current: 80ua.

### 4. Schematic Diagram

Active Buzzer	RPI Shield	GPIO-PCF8591	LM35Temperature Sensor	RPI Shield	GPIO-PCF8591
S	SIO18		S	SA0	
V	5V		V	5V	
G	G		G	G	



## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter `command2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After the I2C communication function is enabled, then input the following commands in the terminal and press "Enter":

```
cd /home/pi/pythonCode_A
```

```
python 30_LM35.py
```

## 6. Test Results

After the program is activated, the terminal prints the temperature value. When the temperature value is greater than 20°, the buzzer will emit sounds; on the contrary, the buzzer won't emit sounds.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

The temperature value 20 in the code can be adjusted according to the local temperature.

```
import RPi.GPIO as GPIO
import time
import smbus

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)      # Use BCM GPIO numbers

buz = 18
GPIO.setup(buz, GPIO.OUT)

address = 0x48 ##address--->device address
cmd = 0x40
A0 = 0x40    ##A0---->port address
A1 = 0x41
A2 = 0x42
```

(continues on next page)

(continued from previous page)

```

A3 = 0x43
bus = smbus.SMBus(1)

def analogRead(count):    #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True:
    temp = analogRead(0)    ##read A0 data
    if(temp>20):
        GPIO.output(buz,GPIO.HIGH)
    else:
        GPIO.output(buz,GPIO.LOW)

    print("Temp = %s"%(temp))    ##print data
    time.sleep(0.1);    ##delay 0.5 second

GPIO.cleanup()

```

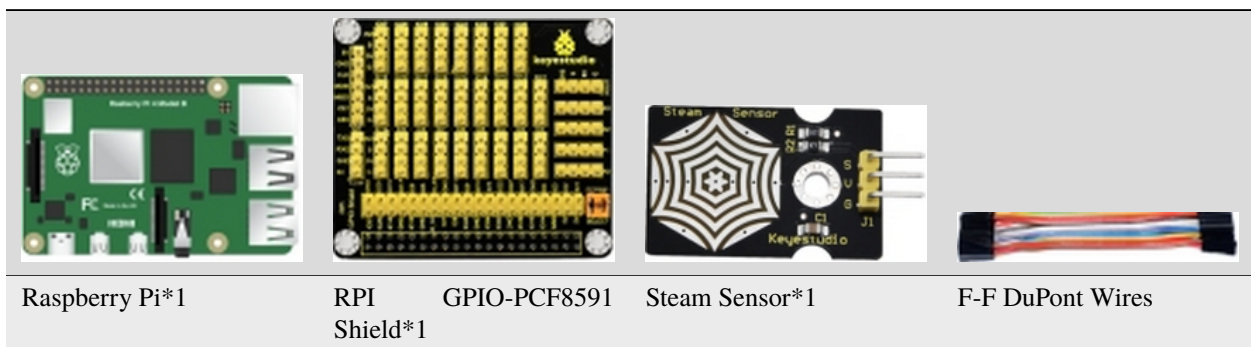
## 6.4.31 Project 31: Steam Sensor

### 1. Description

Our lives are surrounded by air everywhere. The air contains many ingredients, some of which are useful, some are harmful, some of which have a significant impact on the human body, and some of which have a slight effect on the human body.

So in this lesson, you will learn how to use a steam sensor and Raspberry Pi to detect the vapor content in the air.

### 2. Components





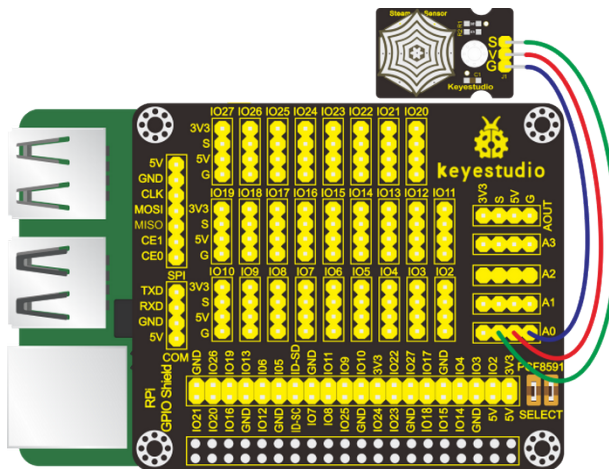
### 3. Component Knowledge

#### Steam Sensor

This is a commonly used steam sensor. Its principle is to detect the amount of water by bare printed parallel lines on the circuit board. The more the water is, the more wires will be connected. As the conductive contact area increases, the output voltage will gradually rise. It can detect water vapor in the air as well. The steam sensor can be used as a rain water detector and level switch. When the humidity on the sensor surface surges, the output voltage will increase.

### 4. Schematic Diagram

Steam Sensor	RPI GPIO-PCF8591 Shield
S	SA0
V	5V
G	G



### 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter `command i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 31_water_vapor.py
```

## 6. Test Results

The terminal shows the steam content in the air.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import time
import smbus

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)      # Use BCM GPIO numbers

address = 0x48 ##address--->device address
cmd = 0x40
A0 = 0x40    ##A0---->port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1)

def analogRead(count):      #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True:
    value = analogRead(0)    ##read A0 data
    print("water content = %s"%(value))    ##print data
    time.sleep(0.1);        ##delay 0.5 second

GPIO.cleanup()
```

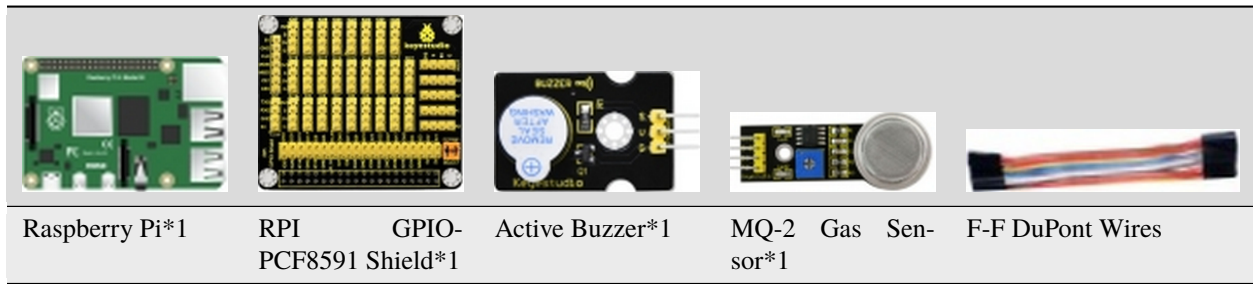
### 6.4.32 Project 32Gas Leakage Alarm

#### 1. Description

Some households have access to gas, which is composed of CO, CO<sub>2</sub>, N<sub>2</sub>, H<sub>2</sub> and CH<sub>4</sub>. CO is one of toxic gases. People will be in danger if absorbing too much CO. However, we could tackle with this problem over a gas leakage alarm.

Gas MQ-2 leakage alarm detects the presence of a combustible or toxic gas and react by displaying a reading, setting off an audible or visual alarm.

## 2. Components



## 3. Component Knowledge

### MQ-2 Gas Sensor

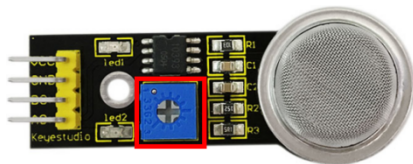
MQ-2 gas sensor adopts the material sensitive to gas——SnO<sub>2</sub> with low electricity conductivity. When beset with combustible gas, its electricity conductivity varies with the of the concentration of flammable gas, however, the simple circuit could convert the change of electricity conductivity into the output signals of the concentration of gas sensor.

MQ-2 gas sensor is a multi-purpose and cost-effective. It can detect the concentration of flammable gas and smoke in the range of 300~10000ppm. Meanwhile, it has high sensitivity to natural gas, liquefied petroleum gas and other smoke, especially to alkanes smoke.

#### Note

(1) The sensitivity of the alcohol sensor can be adjusted by rotating the potentiometer on it.

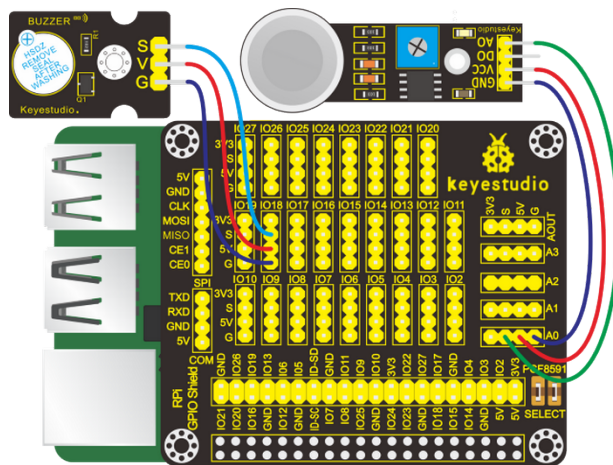
Turning the knob clockwise, the threshold value increases while turning it counterclockwise, the threshold value decreases.



(2) The sensor may not be able to output stable and accurate data immediately, and it needs to be warmed up for about 1 minute to collect stable data.

## 4. Schematic Diagram

Active Buzzer	RPi GPIO-PCF8591 Shield	MQ-2 Gas Sensor	RPi GPIO-PCF8591 Shield
S	SIO18	S	SA0
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter `command2cdetect -y 1` and press “Enter”). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/pythonCode_A
```

```
python 32_gas_MQ_2.py
```

## 6. Test Results

The terminal will show the gas analog value detected by MQ-2 sensor; when the value is more than 60, the buzzer will emit sounds.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import time
import smbus

#pcf8591
address=0x48
cmd=0x40
A0=0x40##A0---->port address
A1=0x41
A2=0x42
A3=0x43
bus=smbus.SMBus(1)

#buzzer
buzPin = 18 #set buzPin to 18
GPIO.setmode(GPIO.BCM) # use BCM numbers
```

(continues on next page)

(continued from previous page)

```

GPIO.setwarnings(False)
GPIO.setup(buzPin,GPIO.OUT)  #set buzPin OUTPUT mode

def main():
    while True:
        Value = analogRead(0)
        print("MQ-2 = %s"%(value))
        time.sleep(0.01)

def analogRead(count):
    read_val=bus.read_byte_data(address,cmd+count)
    if(read_val > 60):
        GPIO.output(buzPin,GPIO.HIGH)  #Buzzer ring
    else:
        GPIO.output(buzPin,GPIO.LOW)  #Buzzer stop
    mq2_val = str(read_val)  # int to string
    return mq2_val

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()

```

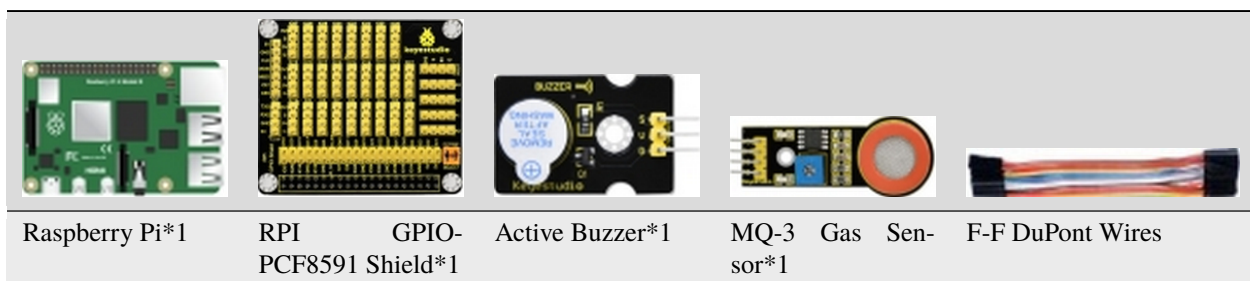
## 6.4.33 Project 33Alcohol Tester

### 1. Description

In this project, you will learn how to use an analog alcohol sensor and Raspberry Pi to detect the alcohol content in the air.

This analog sensor-MQ3 is suitable for detecting the alcohol. It can be used in a breath analyzer. It has a good selectivity because it has higher sensitivity to alcohol and lower sensitivity to Benzine.

### 2. Components

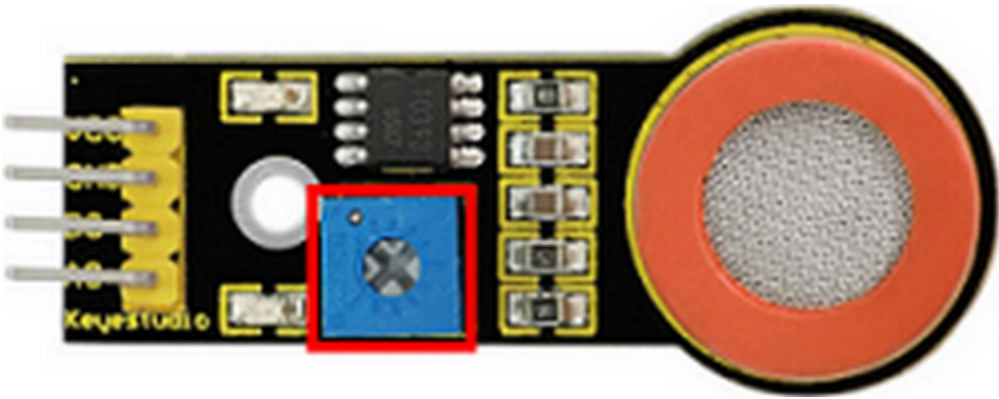


3. Component Knowledge

MQ-3 Alcohol Sensor

This analog gas sensor - MQ3 adapts a gas-sensitive material called tin dioxide (SnO<sub>2</sub>) which is of low conductivity in clean air. Therefore, when there is alcohol vapor detected, its conductivity increases with the increase of the alcohol vapor concentration and it outputs signals (digital and analog signals). The higher the alcohol concentration it senses, the greater the analog value the terminal outputs.

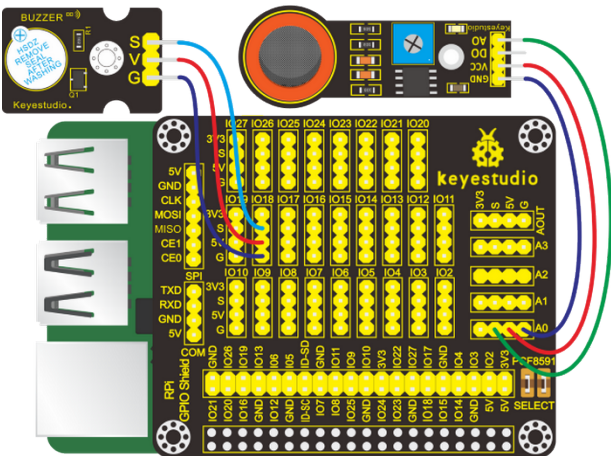
Note: the sensitivity of the alcohol sensor can be adjusted by rotating the potentiometer on it.



Please note that the sensor may not be able to output stable and accurate data immediately, and it needs to be warmed up for about 1 minute to collect stable data.

4. Schematic Diagram

Active Buzzer	RPI GPIO-PCF8591 Shield	MQ-3 Alcohol Sensor	RPI GPIO-PCF8591 Shield
S	SIO18	S	SA0
V	5V	V	5V
G	G	G	G



## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter command `i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 33_alcohol_MQ_3.py
```

## 6. Test Results

After running the program, the terminal displays the analog alcohol value in the air detected by the MQ-3 alcohol sensor. And when the analog value is bigger than 80, the buzzer make a sound; otherwise, it reminds silent.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import time
import smbus

#pcf8591
address=0x48
cmd=0x40
A0=0x40##A0---->port address
A1=0x41
A2=0x42
A3=0x43
bus=smbus.SMBus(1)

#buzzer
buzPin = 18 #set buzPin to 18
GPIO.setmode(GPIO.BCM) # use BCM numbers
GPIO.setwarnings(False)
GPIO.setup(buzPin,GPIO.OUT) #set buzPin OUTPUT mode

def main():
    while True:
        value = analogRead(0)
        print("MQ-3 = %s"%(value))
        time.sleep(0.01)

def analogRead(count):
    read_val=bus.read_byte_data(address,cmd+count)
    if(read_val > 80):
        GPIO.output(buzPin,GPIO.HIGH) #Buzzer ring
    else:
        GPIO.output(buzPin,GPIO.LOW) #Buzzer stop
```

(continues on next page)

(continued from previous page)

```
mq3_val = str(read_val) # int to string
return mq3_val

if __name__ == '__main__':

    try:
        main()
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()
```

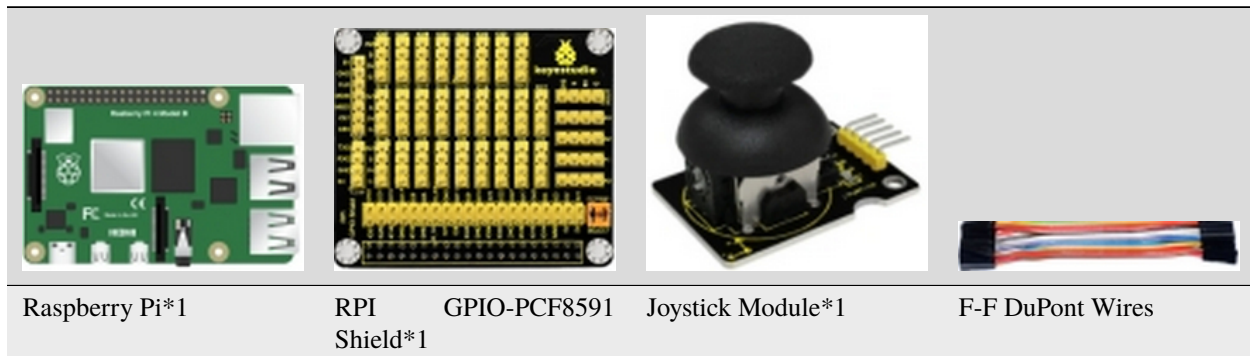
## 6.4.34 Project 34Joystick Module

### 1. Description

Many a people play games with gamepad. But do you know who it work?

Let's learn about it.

### 2. Components



### 3. Component Knowledge

#### Joystick Module

This is a joystick very similar to the 'analog' joysticks on PS2 (PlayStation 2) controllers. It is a self-centering spring loaded joystick, meaning when you release the joystick it will center itself. It also contains a comfortable cup-type knob/cap which gives the feel of a thumb-stick.

It has three signal pins which are connected GND, VCC and signal endB, X, Y). The X pin is **X-axis** (left to right), the Y pin is **Y-axis** (front and back) and signal B end is Z-axis(usually used as digital port and pushbutton).

VCC is connected to V/VCC3.3/5V of MCU, GND to G/GND of MCU and the voltage is around 1.65V/2.5V in initial status.

X axis gives readout of the joystick in the horizontal direction (X-coordinate) i.e. how far left and right the joystick is pushed.

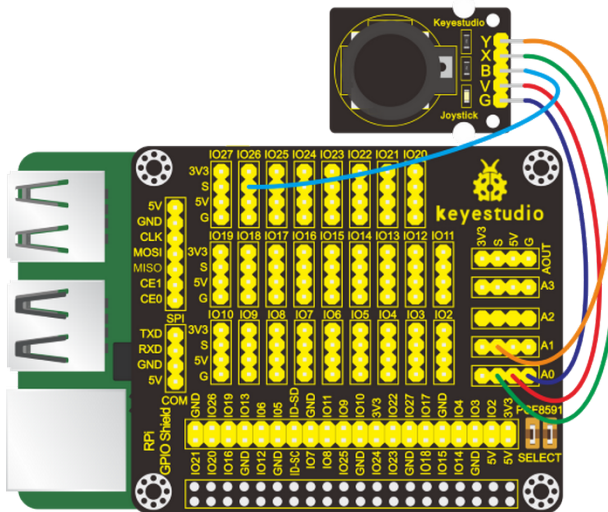


Y axis gives readout of the joystick in the vertical direction (Y-coordinate) i.e. how far up and down the joystick is pushed.

Z axis is the output from the pushbutton. It's normally open, meaning the digital readout from the SW pin will be HIGH. When the button is pushed, it will connect to GND, giving output LOW.

#### 4. Schematic Diagram

Joystick Module	RPI GPIO-PCF8591 Shield
Y	SA1
X	SA0
B	S(IO26)
V	5V
G	G



#### 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter `commandi2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 34_joystick.py
```

## 6. Test Results

Move joystick , the terminal will show the responding data change. If you press it, “The key is pressed” is displayed in the terminal.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

key = 26 # joystick button pin
GPIO.setup(key, GPIO.IN)

address = 0x48 ##address ---> device address
cmd = 0x40      ##DA converter command
A0 = 0x40       ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1)          ##start the bus

def analogRead(count): #function, read analog data
    read_val = bus.read_byte_data(address, cmd+count)
    return read_val

while True:                  ##loop
    #Vout = 10                ##10*0.0196=0.196V
    #bus.write_byte_data(address, cmd, Vout) ##DA converter
    x_val = analogRead(0) ##read A0 data
    y_val = analogRead(1) ##read A1 data
    print("x:%1.0f y:%1.0f" %(x_val, y_val))          ##print data
    if GPIO.input(key):
        print("The key is pressed")

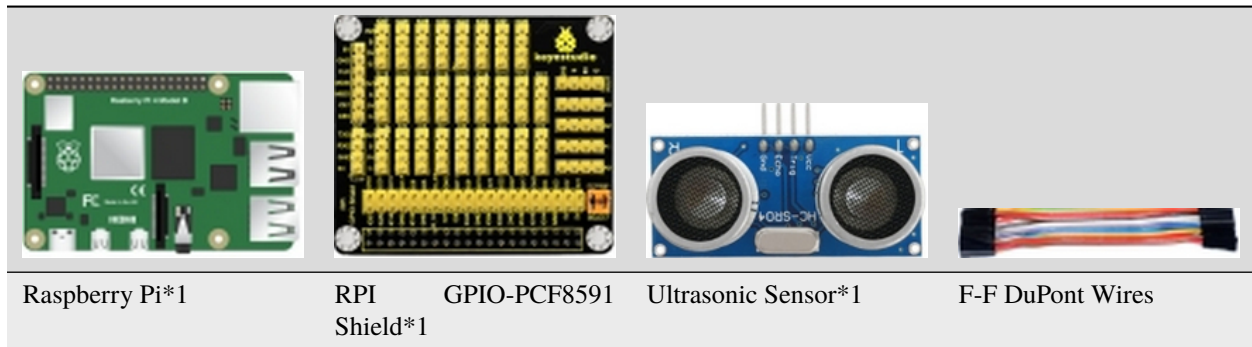
GPIO.cleanup()
```

## 6.4.35 Project 35Ultrasonic Sensor

### 1. Description

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal.

## 2. Components



## 3. Component Knowledge

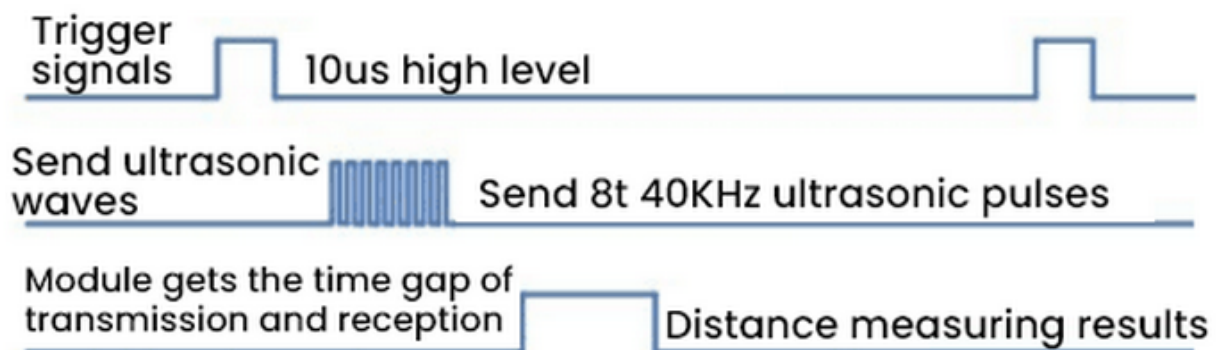
### Ultrasonic Sensor

The ultrasonic module will emit the ultrasonic waves after trigger signal. When the ultrasonic waves encounter the object and are reflected back, the module outputs an echo signal, so it can determine the distance of object from the time difference between trigger signal and echo signal.

The  $t$  is the time that emitting signal meets obstacle and returns.

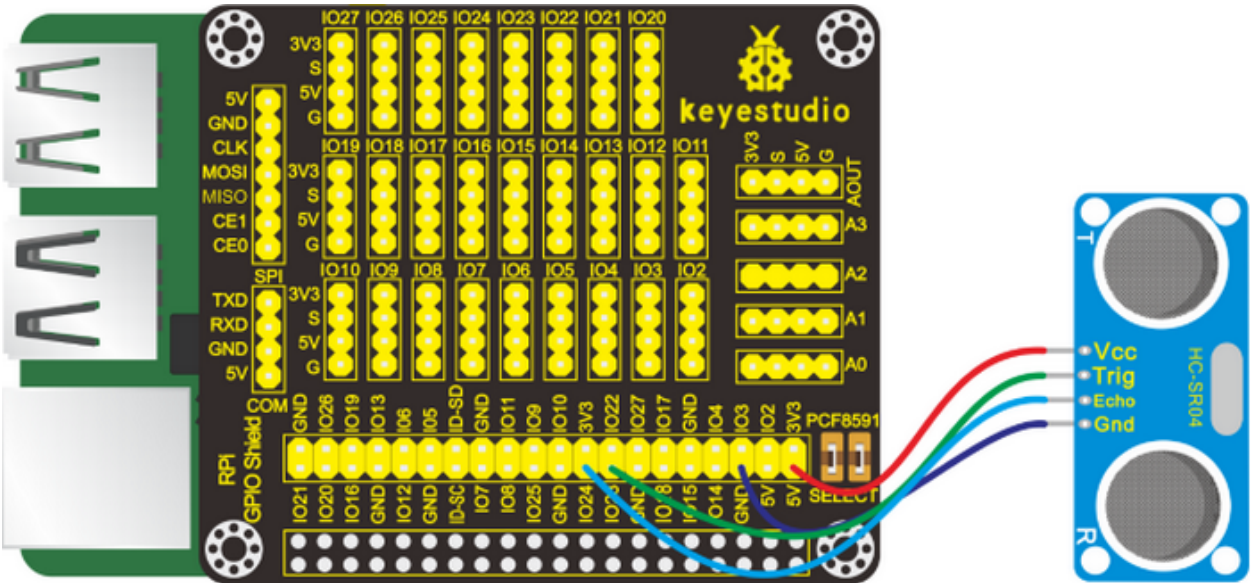
and the propagation speed of sound in the air is about 343m/s, therefore, distance = speed \* time, because the ultrasonic wave emits and comes back, which is 2 times of distance, so it needs to be divided by 2, the distance measured by ultrasonic wave = (speed \* time)/2 .

1. Use method and timing chart of ultrasonic module:
2. Setting the delay time of Trig pin of SR04 to 10s at least, which can trigger it to detect distance.
3. After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step will be completed automatically by the module.
4. If the signal returns, the Echo pin will output a high level, and the duration of the high level is the time from the transmission of the ultrasonic wave to the return.



4. Schematic Diagram

Ultrasonic Sensor	RPI GPIO-PCF8591 Shield
Vcc	5V
Trig	S(IO23)
Echo	S(IO24)
Gnd	GND



5. Run Example Code

Input the following commands in the terminal and press“Enter”:

```
cd /home/pi/pythonCode_A
python 35_ultrasonic.py
```

6. Test Results

The terminal will print the detected distance value, and its unit is cm.

Note: Press Ctrl + C on keyboard to exit code running.

7. Example Code

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

#define GPIO pin
GPIO_TRIGGER = 23
```

(continues on next page)

(continued from previous page)

```

GPIO_ECHO = 24

#set GPIO mode (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    # 10us is the trigger signal
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001) #10us
    GPIO.output(GPIO_TRIGGER, False)

    start_time = time.time()
    stop_time = time.time()

    while GPIO.input(GPIO_ECHO) == 0: #Indicates that the ultrasonic wave has been
    ↳emitted
        start_time = time.time() #Record launch time

    while GPIO.input(GPIO_ECHO) == 1: #Indicates that the returned ultrasound has been
    ↳received
        stop_time = time.time() #Record receiving time

    time_elapsed = stop_time - start_time #Time difference from transmit to receive
    distance = (time_elapsed * 34300) / 2 #Calculate the distance
    return distance #Return to calculated distance

if __name__ == '__main__': #Program entry
    try:
        while True:
            dist = distance() #
            print("Measured Distance = {:.2f} cm".format(dist)) # {:.2f}, Keep two decimal
            ↳places
            time.sleep(0.1)

            # Reset by pressing CTRL + C
    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()

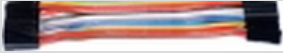

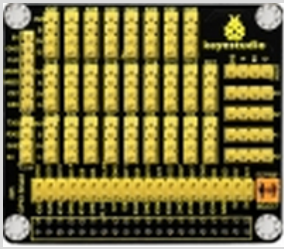

```

## 6.4.36 Project 36Light Intensity Detection

### 1. Description

In this chapter, we will use the TEMENT6000 ambient light sensor whose sensitivity is better than photoresistor. We will learn how to test ambient light intensity using TEMENT6000 environmental sensor and Raspberry Pi.

2. Components



Raspberry Pi\*1RPI GPIO-PCF8591 Shield\*1TEMT6000 Ambient Light Sensor\*1AmbientF-F DuPont Wires

3. Component Knowledge

TEMT6000 Ambient Light Sensor

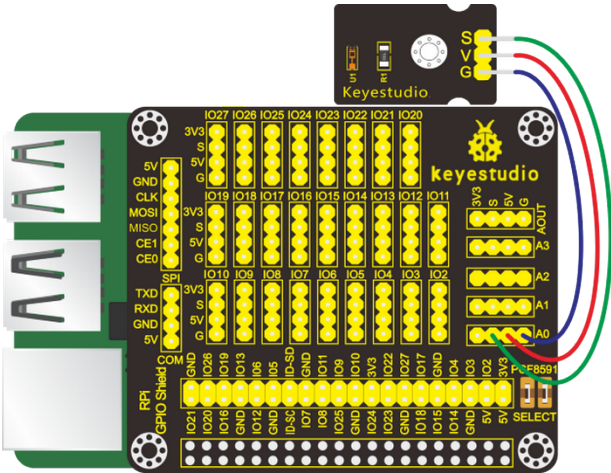
This module is mainly composed of a highly sensitive visible photocell (NPN type) triode, which can magnify the captured tiny light illumination changes by about 100 times, and is easily recognized by the microcontroller for AD conversion.

And the light intensity is directly proportional to current flowing through. Therefore, it is easy to figure out the light intensity as long as its voltage is known.

Its response to visible light illumination is similar to that of the human eye, so that can detect the intensity of ambient light.

4. Schematic Diagram

TEMT6000 Ambient Light Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter command `i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 36_TEMT6000_Ambient_Light.py
```

## 6. Test Results

The terminal will show the ambient light value. The stronger the ambient light, the larger the analog value.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

address = 0x48 ##address ---> device address
cmd = 0x40     ##DA converter command
A0 = 0x40      ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1)          ##start the bus

def analogRead(count):    #function, read analog data
    read_val = bus.read_byte_data(address, cmd+count)
    return read_val

while True:                ##loop
    #Vout = 10              ##10*0.0196=0.196V
    #bus.write_byte_data(address, cmd, Vout) ##DA converter
    value = analogRead(0) ##read A0 data
    print("Ambient Light:%1.0f" %(value)) ##print data

time.sleep(0.5)            ##delay 0.5 second
GPIO.cleanup()
```

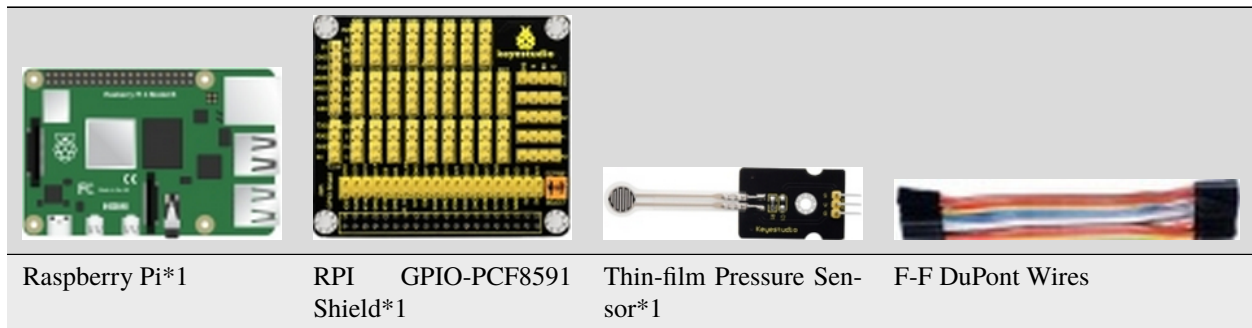
## 6.4.37 Project 37 Pressure Detection

### 1. Description

In the previous project, we learned to obtain a variety of information through different sensors, such as temperatures, light, sound, gases, and so on.

Now let's use the thin-film pressure sensor and the Raspberry Pi to detect external pressure sizes.

### 2. Components:



### 3. Component Knowledge

#### Thin-film Pressure Sensor

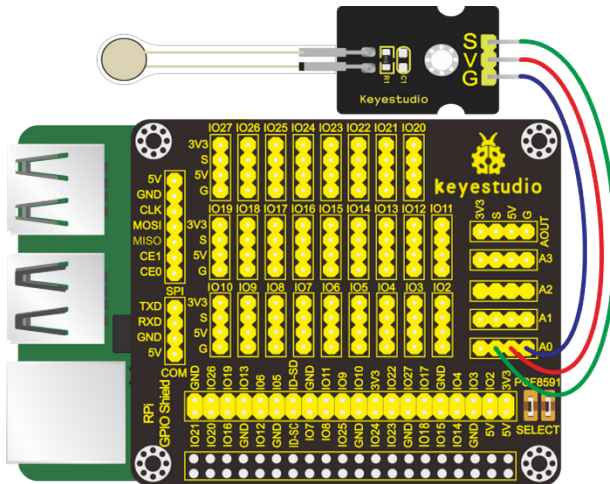
This sensor adopts the flexible Nano pressure-sensitive material with an ultra-thin film pad. It has the functions of water-proof and pressure detection.

When the sensor detects the outside pressure, the resistance of sensor will make a change. So we can use the circuit to convert the pressure signal that senses pressure change into the corresponding electric signal output. In this way, we can know the conditions of pressure changes by detecting the signal changes.

### 4. Schematic Diagram

Thin-film Pressure Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G





## 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter command `i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter":

```
cd /home/pi/pythonCode_A
python 37_pressure_transducer.py
```

## 6. Test Results

The terminal will print the external pressure analog value. The greater the external pressure, the larger the analog value; on the contrary, the smaller the analog value.

Note: Press Ctrl + C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

address = 0x48 ##address ---> device address
cmd = 0x40     ##DA converter command
A0 = 0x40      ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1)          ##start the bus

def analogRead(count):        #function, read analog data
```

(continues on next page)

(continued from previous page)

```

    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True:                                ##loop
    #Vout = 10                               ##10*0.0196=0.196V
    #bus.write_byte_data(address,cmd,Vout) ##DA converter
    value = analogRead(0) ##read A0 data
    print("pressure value:%1.0f" %(value))    ##print data

time.sleep(0.5)                             ##delay 0.5 second
GPIO.cleanup()

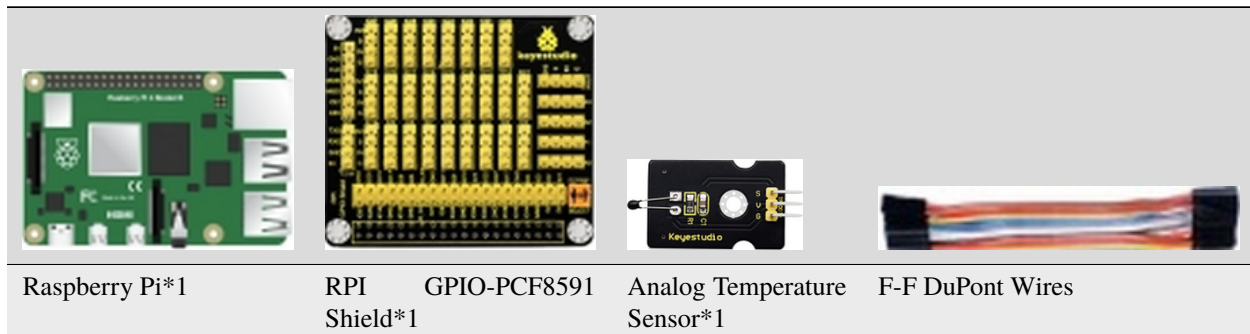
```

## 6.4.38 Project 38 Temperature Detection

### 1. Description

Thermistor is a resistor, and its resistance depends on temperature and temperature changes. Therefore, we can use this feature to make a thermometer.

### 2. Components

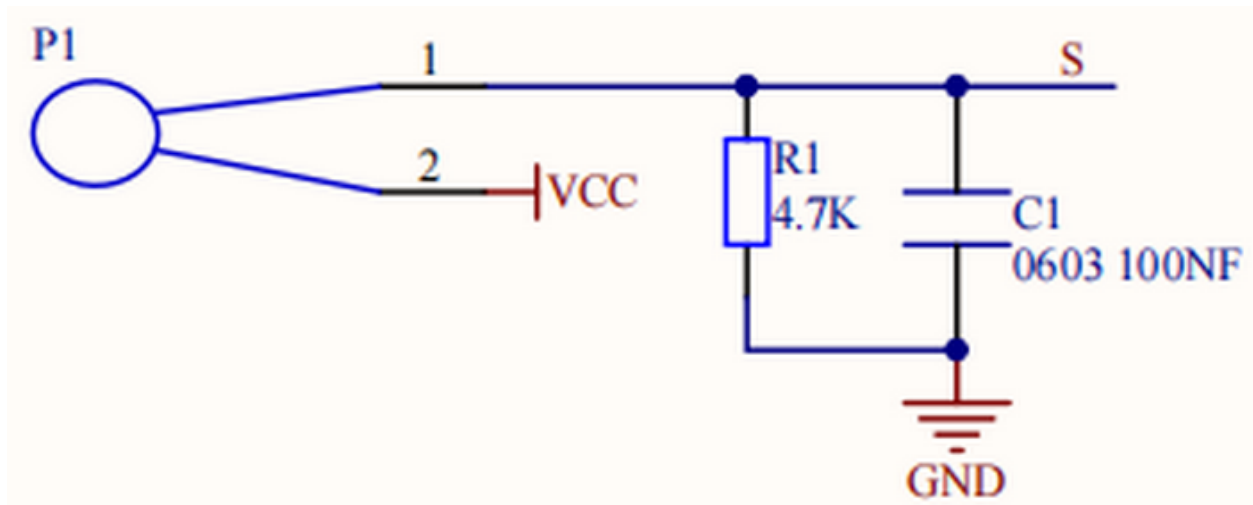


### 3. Component Knowledge

#### Analog Temperature Sensor

The main part of this sensor is a thermistor which is quite sensitive to temperature. When it senses the changes of temperature, it makes changes in its resistance. This function of it can be used to detect temperature. Therefore, it has found applications in gardening, home alarm systems and other devices.

The NTC-MF52AT thermistor of 10K (P1) S and resistor R1 of 4.7K are connected in series. The resistance value of the thermistor alters with temperature changes.



Calculation of NTC thermistor:

**The calculation formula of the for NTC thermistor is:**  $R_t = R * EXP[B * (1/T_1 - 1/T_2)]$  Among them,  $T_1$  and  $T_2$  refer to degrees, which is the temperature in Kelvin;

**$R_t$**  is the resistance of the thermistor at temperature  $T_1$ ;

**$R$**  is the nominal resistance of the thermistor at normal temperature  $T_2$ , and the resistance of the 10K thermistor at 25°C is 10K (that is,  $R=10K$ );  $T_2 = (273.15 + 25)$ ;

**EXP[n]** represents  $e^n$  (  $e$  to the  $n$ th power );

The value of  $B$  is an important parameter of thermistor and  **$B=3950$** .

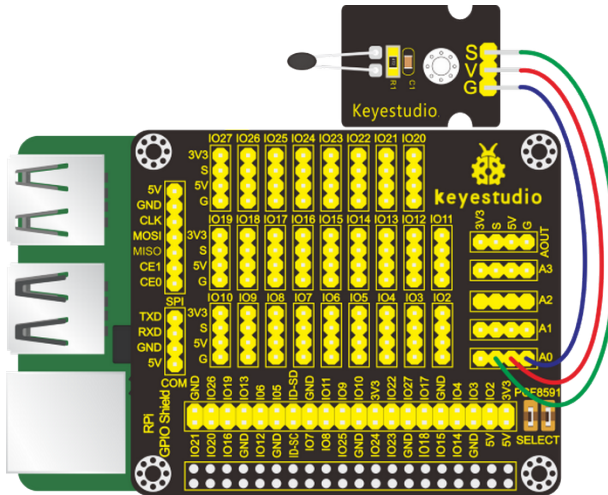
We can use the value measured by the ADC converter to get the resistance value of the thermistor, and then use the formula to get the temperature value. Therefore, the temperature formula can be derived as  **$T_1=1/(\ln(R_t/R)/B+1/T_2)$** , where **ln** can be converted to **log**, that is.

$$T_1 = 1/(\log(R_t/R)/B + 1/T_2)$$

The corresponding Celsius temperature is  **$t=T_1-273.15$** , and the deviation is  $\pm 0.5$ .

#### 4. Schematic Diagram

Analog Temperature Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



#### 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first(enter command `i2cdetect -y 1` and press “Enter”). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press “Enter”:

```
cd /home/pi/pythonCode_A
python 38_analog_temperature.py
```

#### 6. Test Results

The terminal will print the ADC value, the voltage value, and the temperature value of the analog temperature sensor.

Note: Press Ctrl + C on keyboard to exit code running.

#### 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import math
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

(continues on next page)

(continued from previous page)

```

address = 0x48 ##address ---> device address
cmd = 0x40      ##DA converter command
A0 = 0x40       ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1)          ##start the bus

def analogRead(count):      #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True:                  ##loop
    value = analogRead(0)     # read ADC value A0 pin
    voltage = value / 255.0 * 5.0      # calculate voltage
    Rt = 4.7 * (5.0 / voltage) - 4.7 ; #calculate resistance value of thermistor, 5.
    ↪ 0*(R/(Rt+R))=voltage,>>>Rt=R*(5.0/voltage)-R
    tempK = 1/(1/(273.15 + 25) + math.log(Rt/4.7)/3950.0)
    ↪ # calculate temperature (Kelvin)
    tempC = tempK - 273.15      # calculate temperature (Celsius)
    print ('ADC Value : %d, Voltage : %.2f, Temperature : %.2f'%(value,voltage,tempC))

time.sleep(0.02)
GPIO.cleanup()

```

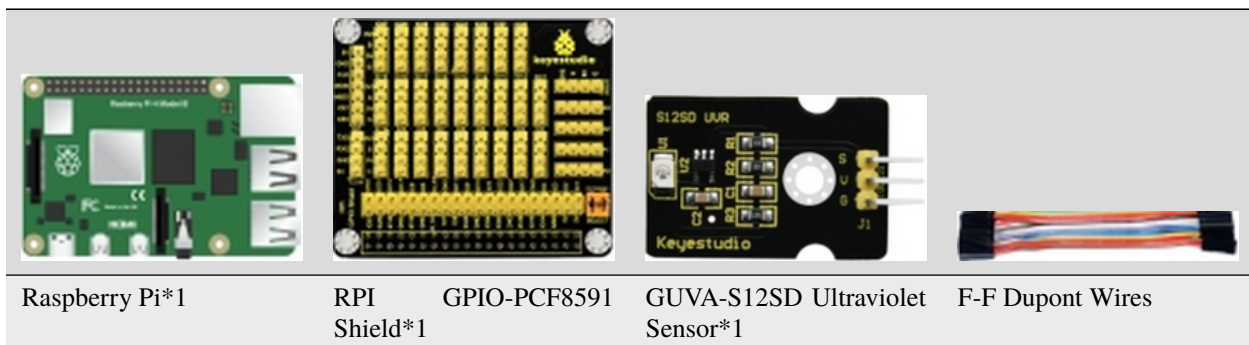
## 6.4.39 Project 39Ultraviolet Light Detection

### 1. Description

UV is a kind of physics optics. The main ultraviolet light source in nature is the sun. Most of the ultraviolet rays emitted by sunlight is absorbed by ozone in the atmosphere, and very few partially irradiates to the earth.

We can detect how much the ultraviolet rays of sunlight using ultraviolet sensors.

### 2. Components



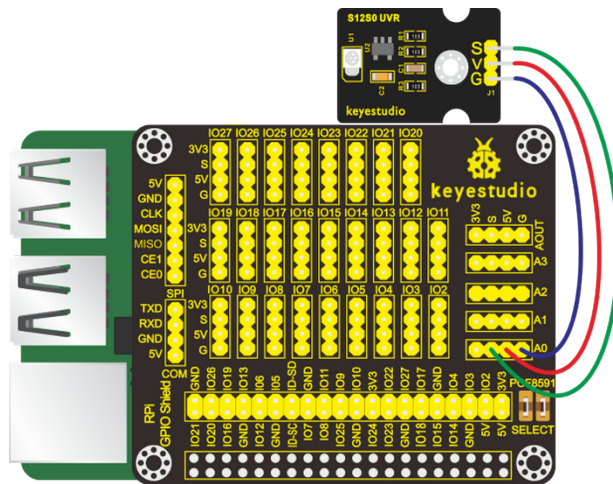
### 3. Component Knowledge

#### GUVA-S12SD Ultraviolet Sensor

It can detect UV and UV index, applied to some wearable devices such as watches and smartphones. It can also be used to monitor the intensity of ultraviolet rays, or ultraviolet flame detectors when used as ultraviolet sterilization items. Its output current is proportional to light intensity. This sensor is mainly for ultraviolet measurements in the sun and the UVA lamp strength measurement and UVI detection.

### 4. Schematic Diagram

GUVA-S12SD Ultraviolet Sensor	RPI GPIO-PCF8591 Shield
S	S(A0)
V	5V
G	G



### 5. Run Example Code

Special Note: The I2C communication method is used in the experiment. We need to check the iic address first (enter `command i2cdetect -y 1` and press "Enter"). If failed, check the wiring is correct or not. If correct, you need to enable I2C communication function of Raspberry Pi, project 24 is for your reference.

After enabling the I2C communication input the following commands and press "Enter" :

```
cd /home/pi/pythonCode_A
python 39_ultraviolet_ray.py
```

## 6. Test Results

Turn on an ultraviolet pen (we don't provide) and point at the ultraviolet sensor , the terminal will print out the ultraviolet intensity value.

Note: Press Ctrl +C on keyboard to exit code running.

## 7. Example Code

```
import RPi.GPIO as GPIO
import smbus
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

address = 0x48 ##address ---> device address
cmd = 0x40 ##DA converter command
A0 = 0x40 ##A0 ----> port address
A1 = 0x41
A2 = 0x42
A3 = 0x43
bus = smbus.SMBus(1) ##start the bus

def analogRead(count): #function,read analog data
    read_val = bus.read_byte_data(address,cmd+count)
    return read_val

while True: ##loop
    value = analogRead(0) ##read A0 data
    print("ultraviolet intensity:%1.0f" %(value)) ##print data
    time.sleep(0.05) ##delay 0.05 second

GPIO.cleanup()
```