SunFounder SunFounder_SensorKit_for_RPi2

www.sunfounder.com

Mar 17, 2023

CONTENTS

1 Components **3**

2 Preparation **13**
- 2.1 What We Need? . . . 13
- 2.2 Installing the OS . . . 15
- 2.3 Set up Your Raspberry Pi . . . 20

3 Libraries **29**
- 3.1 RPi.GPIO . . . 29
- 3.2 WiringPi . . . 30

4 GPIO Extension Board **33**

5 Download the Code **35**

6 Lessons **37**
- 6.1 Lesson 1 Dual-Color LED . . . 37
- 6.2 Lesson 2 RGB LED Module . . . 42
- 6.3 Lesson 3 7-Color Auto-flash LED . . . 49
- 6.4 Lesson 4 Relay Module . . . 51
- 6.5 Lesson 5 Laser Emitter Module . . . 56
- 6.6 Lesson 6 Button Module . . . 61
- 6.7 Lesson 7 Tilt-Switch Module . . . 66
- 6.8 Lesson 8 Vibration Switch . . . 71
- 6.9 Lesson 9 IR Receiver Module . . . 76
- 6.10 Lesson 10 Buzzer Module . . . 81
- 6.11 Lesson 11 Reed Switch . . . 90
- 6.12 Lesson 12 Photo-interrupter . . . 95
- 6.13 Lesson 13 PCF8591 . . . 100
- 6.14 Lesson 14 Rain Detection Module . . . 107
- 6.15 Lesson 15 Joystick PS2 . . . 112
- 6.16 Lesson 16 Potentiometer Module . . . 117
- 6.17 Lesson 17 Hall Sensor . . . 122
- 6.18 Lesson 18 Temperature Sensor . . . 132
- 6.19 Lesson 19 Sound Sensor . . . 142
- 6.20 Lesson 20 Photoresistor Module . . . 147
- 6.21 Lesson 21 Flame Sensor . . . 151
- 6.22 Lesson 22 Gas Sensor . . . 157
- 6.23 Lesson 23 IR Remote Control . . . 163
- 6.24 Lesson 24 Touch Switch . . . 172
- 6.25 Lesson 25 Ultrasonic Ranging Module . . . 177

6.26 Lesson 26 DS18B20 Temperature Sensor . . . 182
6.27 Lesson 27 Rotary Encoder Module . . . 189
6.28 Lesson 28 Humiture Sensor . . . 195
6.29 Lesson 29 IR Obstacle Avoidance Module . . . 202
6.30 Lesson 30 I2C LCD1602 . . . 207
6.31 Lesson 31 Barometer-BMP180 Module . . . 212
6.32 Lesson 32 MPU6050 Gyro Acceleration Sensor . . . 217
6.33 Lesson 33 RTC DS1302 . . . 223
6.34 Lesson 34 Tracking Sensor . . . 231
6.35 Lesson 35 Intelligent Temperature Measurement System . . . 236

7 Appendix **247**
7.1 I2C Configuration . . . 247
7.2 SPI Configuration . . . 249
7.3 Remote Desktop . . . 252

8 FAQ **263**
8.1 C code is not working? . . . 263

9 Thank You **265**

10 Copyright Notice **267**

This sensor kit is suitable for the Raspberry Pi model B+, 2 model B, 3 model B, 3 model B+ and 4 Model B. It includes dozens of different modules for you to learn and we provide corresponding lessons which are simple and useful for better understanding. Hope you can learn their applications quickly and use them in your own projects!

If you want to learn another projects which we don't have, please feel free to send Email and we will update to our online tutorials as soon as possible, any suggestions are welcomed.

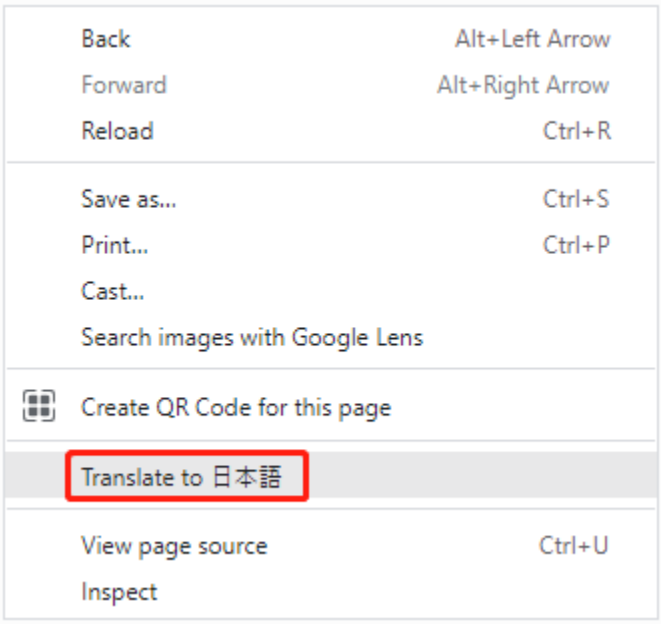
Here is the Email: cs@sunfounder.com.

About the display language

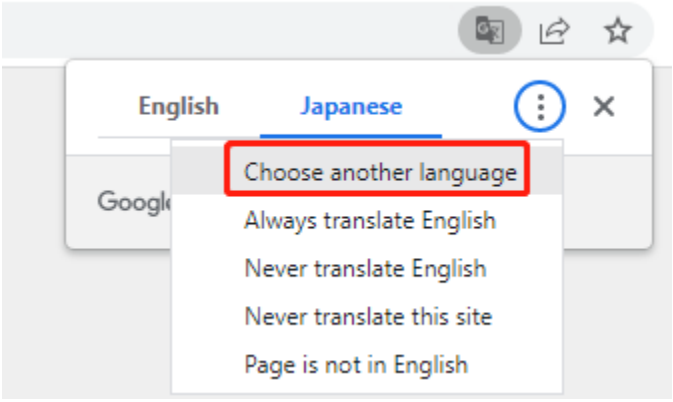
In addition to English, we are working on other languages for this course. Please contact service@sunfounder.com if you are interested in helping, and we will give you a free product in return. In the meantime, we recommend using Google Translate to convert English to the language you want to see.

The steps are as follows.

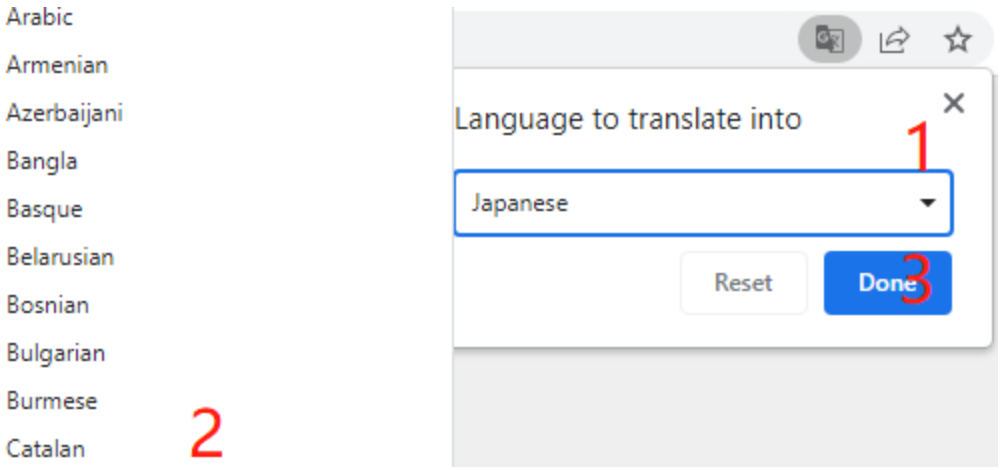
- In this course page, right-click and select **Translate to xx**. If the current language is not what you want, you can change it later.



- There will be a language popup in the upper right corner. Click on the menu button to **choose another language**.



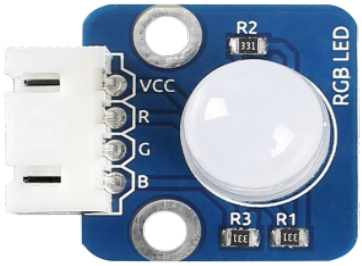
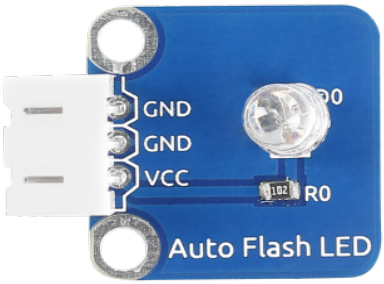
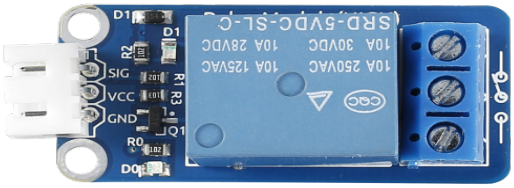
- Select the language from the inverted triangle box, and then click **Done**.


Arabic
Armenian
Azerbaijani
Bangla
Basque
Belarusian
Bosnian
Bulgarian
Burmese
Catalan
2
Language to translate into
1
Japanese
Reset
Done
3

CHAPTER

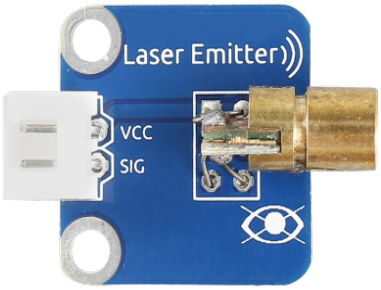
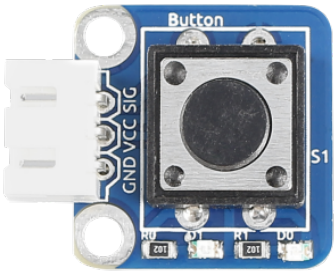
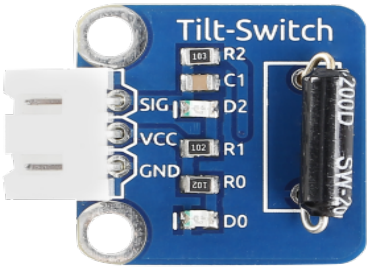
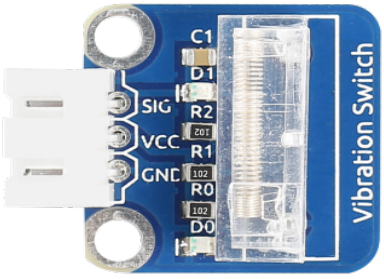
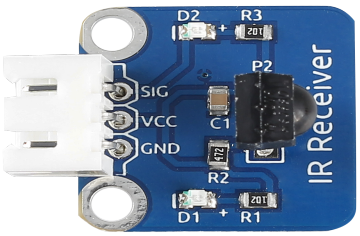
ONE

COMPONENTS

No.	Name	Qty.	Component
1	Dual-Color LED	1	
2	RGB LED	1	
3	Auto Flash LED	1	
4	Relay Module	1	

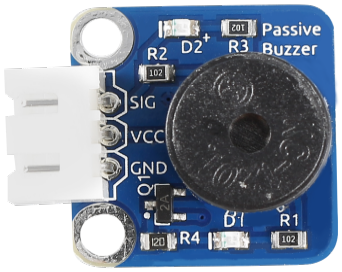
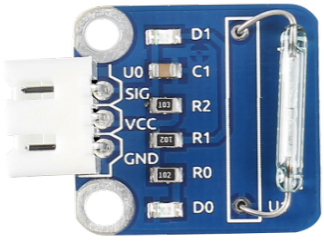
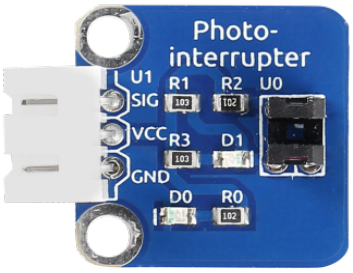
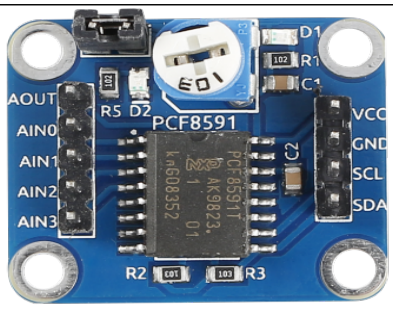
continues on next page

Table 1 – continued from previous page

5	Laser Emitter	1	
6	Button	1	
7	Tilt-Switch	1	
8	Vibration Switch	1	
9	IR Receiver	1	

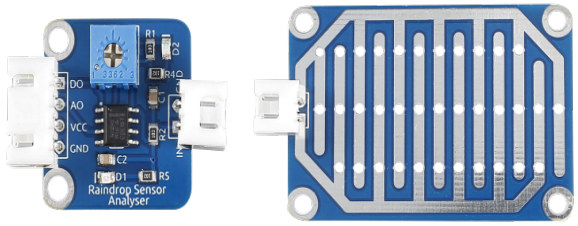
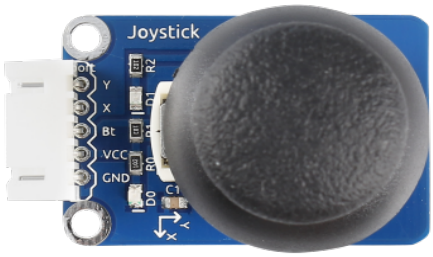
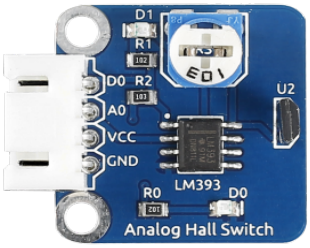
continues on next page

Table 1 – continued from previous page

10	Active Buzzer	1	
11	Passive Buzzer	1	
12	Reed Switch	1	
13	Photo - interrupter	1	
14	AD/DA Converter PCF8591	1	

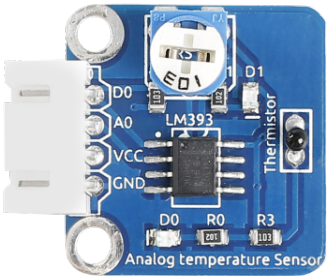
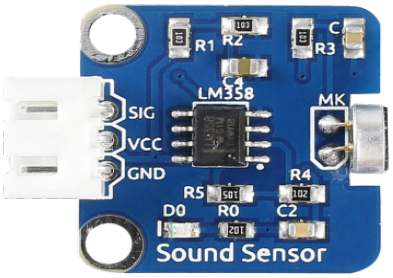
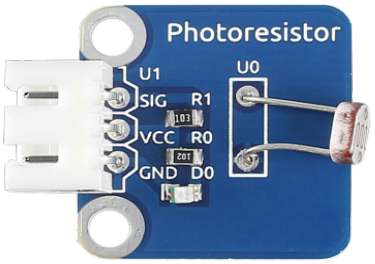
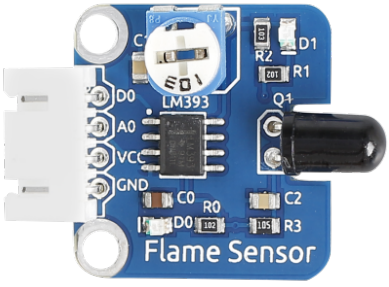
continues on next page

Table 1 – continued from previous page

15	Raindrop Sensor	1	
16	Joystick PS2	1	
17	Potentiometer	1	
18	Analog Hall Sensor	1	
19	Hall Switch Sensor	1	

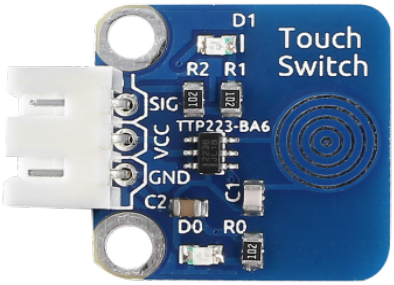
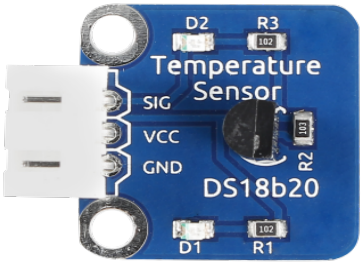
continues on next page

Table 1 – continued from previous page

20	Analog Temperature Sensor	1	
21	Thermistor	1	
22	Sound Sensor	1	
23	Photoresistor	1	
24	Flame Sensor	1	

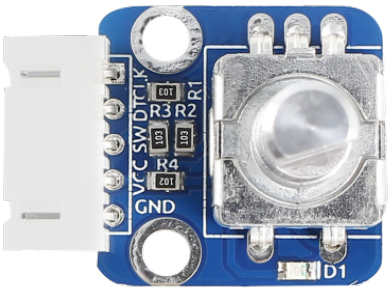
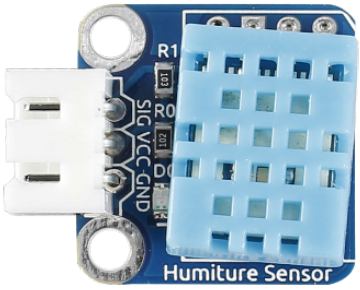
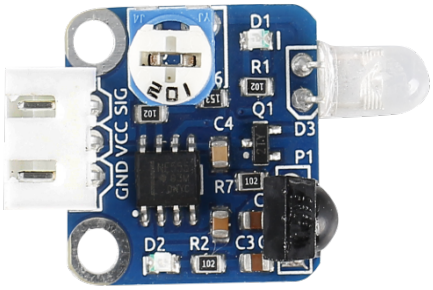
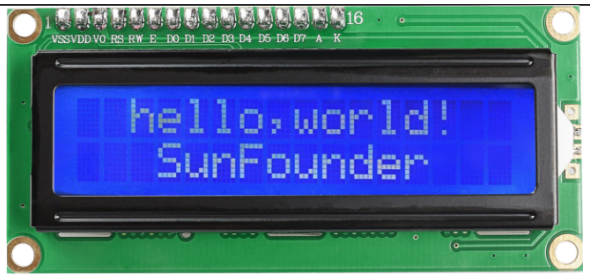
continues on next page

Table 1 – continued from previous page

25	Gas Sensor	1	
26	Remote Control	1	
27	Touch Switch	1	
28	Ultrasonic	1	
29	Temperature Sensor DS18B20	1	

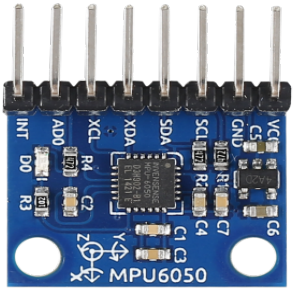
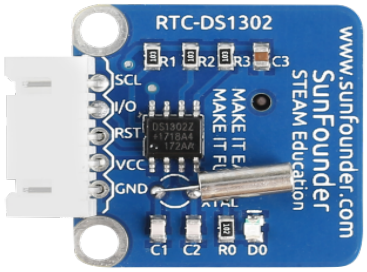
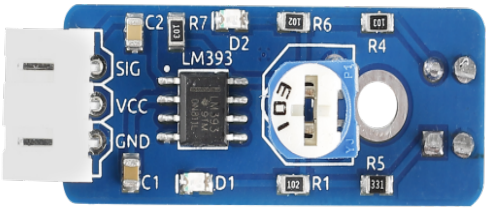
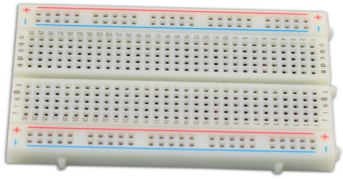
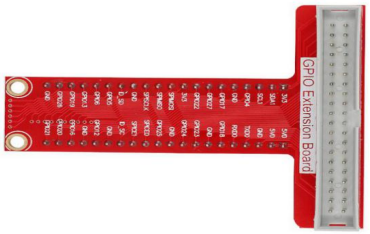
continues on next page

Table 1 – continued from previous page

30	Rotary Encoder	1	
31	Humiture Sensor	1	
32	IR Obstacle Module	1	
33	I2C LCD1602 Module	1	
34	Barometer - BMP180	1	

continues on next page

Table 1 – continued from previous page

35	MPU6050 Module	1	
36	RTC-DS1302 Module	1	
37	Tracking Sensor	1	
38	Breadboard	1	
39	T-Cobbler	1	

continues on next page

Table 1 – continued from previous page

40	40-pin Ribbon Cable for T-Cobbler	1	
41	2-Pin Anti-reverse Cable	2	
42	3-Pin Anti-reverse Cable	5	
43	4-Pin Anti-reverse Cable	5	
44	5-Pin Anti-reverse Cable	5	

continues on next page

Table 1 – continued from previous page

45	Jumper wires (M to F)	20	
46	Jumper wires (M to M)	10	

CHAPTER

TWO

PREPARATION

In this chapter, we firstly learn to start up Raspberry Pi.

Depending on the different devices you use, you can start up the Raspberry Pi in different methods. We have two situations: having a screen or no screen, and you can refer to relevant tutorials respectively. If your Raspberry Pi is set up, you can skip the chapter and go into the next chapter.

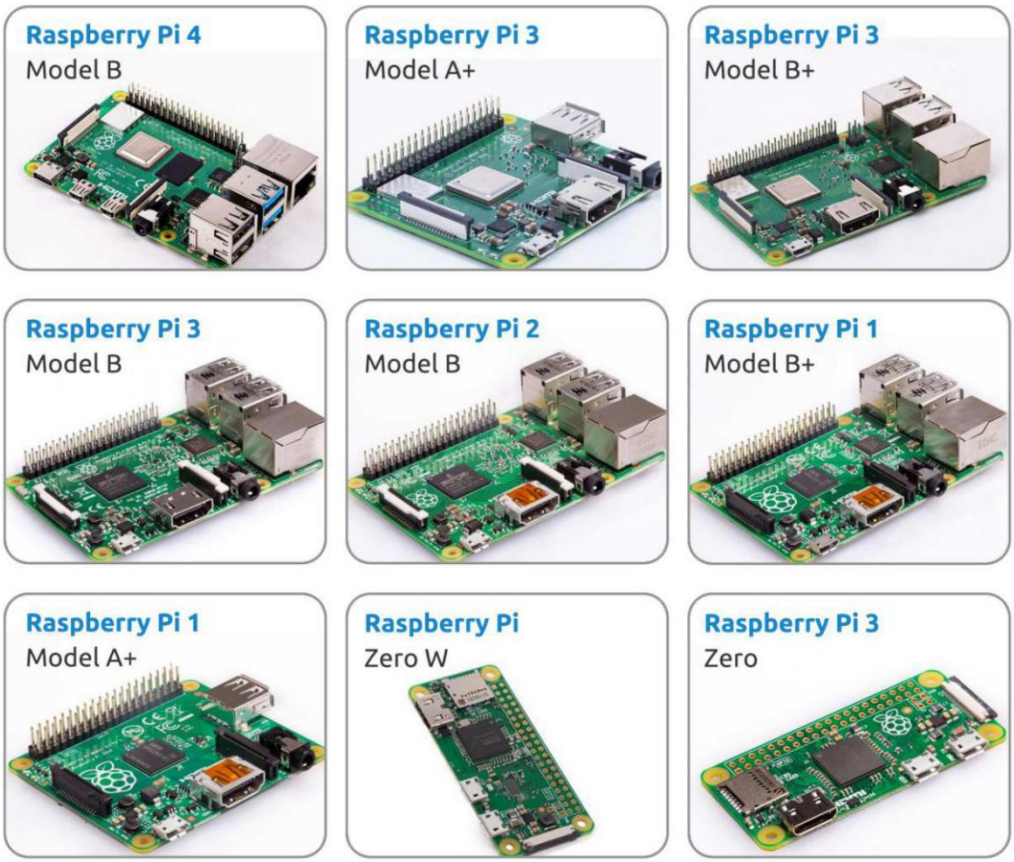
2.1 What We Need?

2.1.1 Required Components

Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

Our kit applies to the following versions of the product of Raspberry Pi :



Power Adapter

To connect to a power socket, the Raspberry Pi has a micro USB port (the same found on many mobile phones). You will need a power supply which provides at least 2.5 amps.

Micro SD Card

Your Raspberry Pi needs an SD card to store all its files and the Raspberry Pi OS. You will need a micro SD card with a capacity of at least 8 GB.

2.1.2 Optional Components

Screen

To view the desktop environment of Raspberry Pi, you need to use the screen that can be a TV screen or a computer monitor. If the screen has built-in speakers, the Pi plays sounds via them.

Mouse & Keyboard

When you use a screen , a USB keyboard and a USB mouse are also needed.

HDMI

The Raspberry Pi has a HDMI output port that is compatible with the HDMI ports of most modern TV and computer monitors. If your screen has only DVI or VGA ports, you will need to use the appropriate conversion line.

Case

You can put the Raspberry Pi in a case; by this means, you can protect your device.

Sound or Earphone

The Raspberry Pi is equipped with an audio port about 3.5 mm that can be used when your screen has no built-in speakers or when there is no screen operation.

2.2 Installing the OS

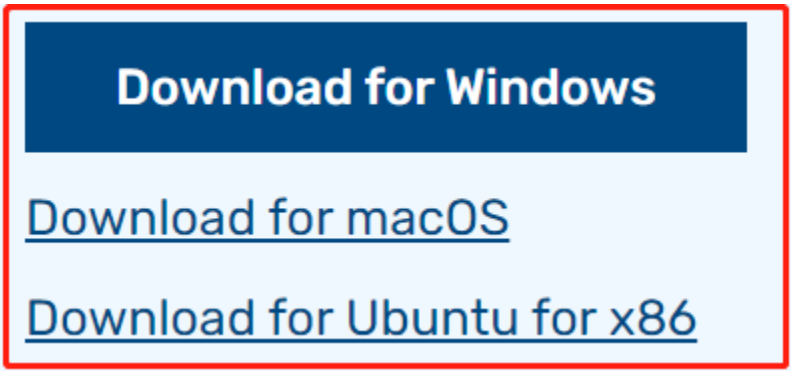
In this chapter, we firstly learn to start up Raspberry Pi. If your Raspberry Pi is set up, you can skip the chapter and go into the next chapter.

Required Components

- Any Raspberry Pi
- 1 * Mirco SD card
- 1 * Personal Computer

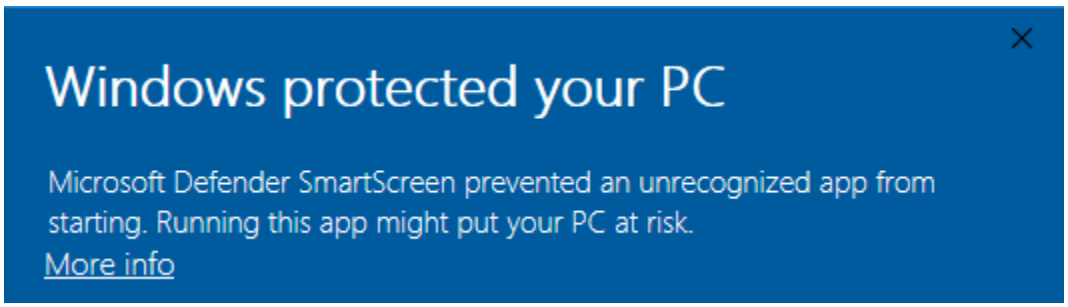
Step 1

Raspberry Pi have developed a graphical SD card writing tool that works on Mac OS, Ubuntu 18.04 and Windows, and is the easiest option for most users as it will download the image and install it automatically to the SD card. Visit the download page: https://www.raspberrypi.org/software/. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



Step 2

When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message: If this pops up, click on More info and then Run anyway, then follow the instructions to install the Raspberry Pi Imager.

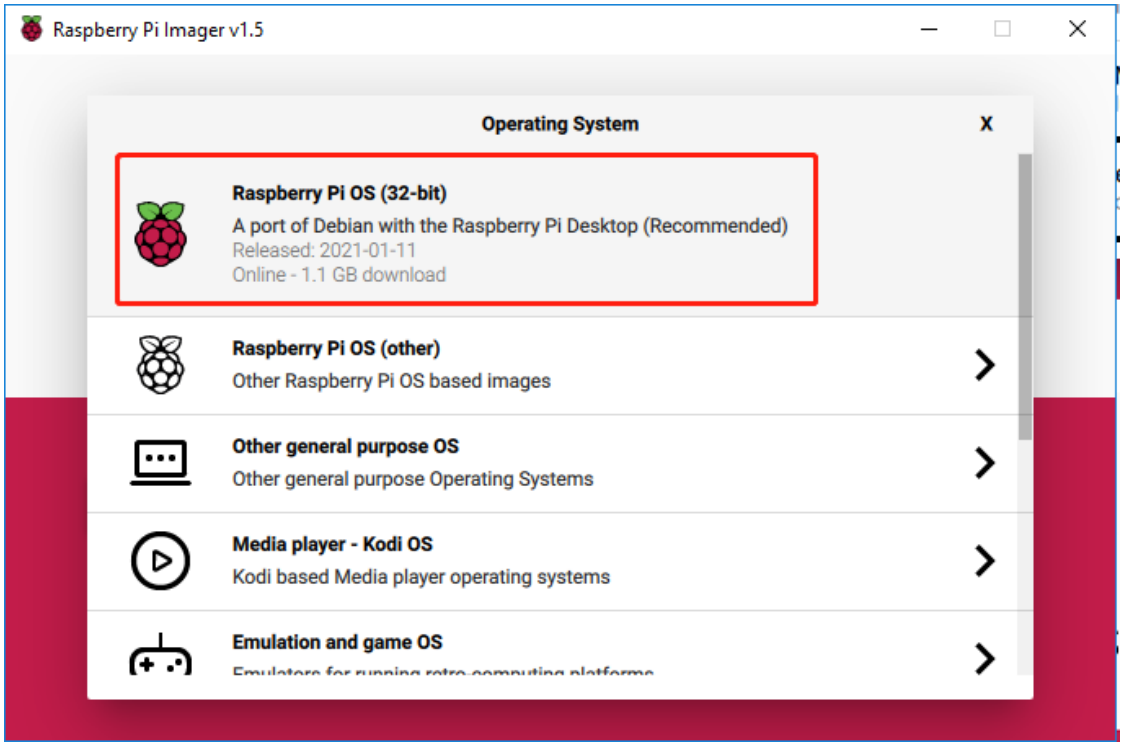


Step 3

Insert your SD card into the computer or laptop SD card slot.

Step 4

In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on.

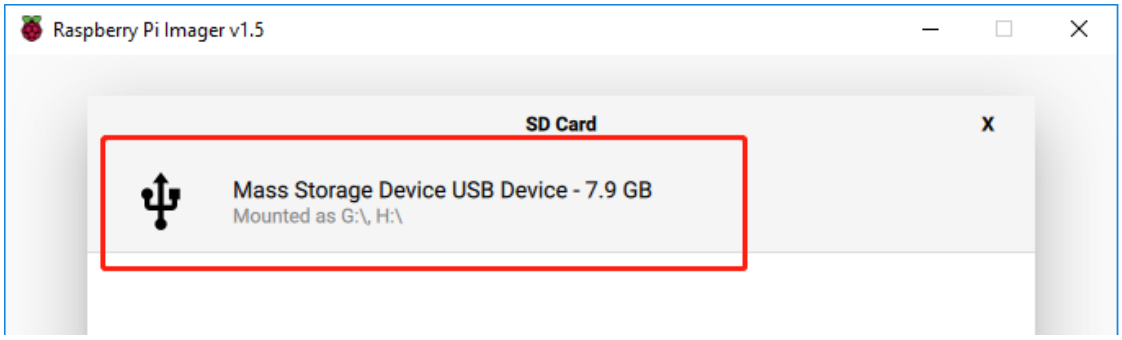


Note:

- You will need to be connected to the internet the first time.
- That OS will then be stored for future offline use(lastdownload.cache, `C:/Users/yourname/AppData/Local/Raspberry Pi/Imager/cache`). So the next time you open the software, it will have the display "Released: date, cached on your computer".

Step 5

Select the SD card you are using.

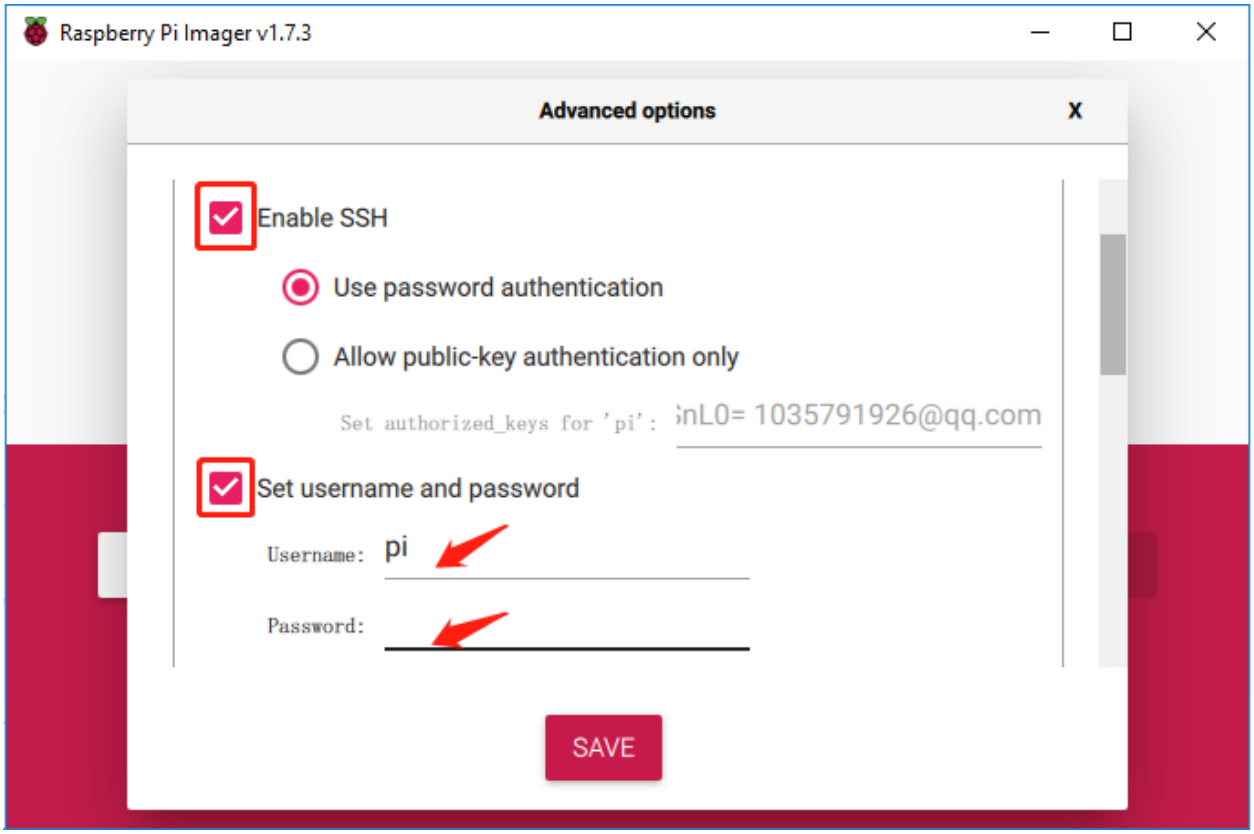


Step 6

Press **Ctrl+Shift+X** or click the **setting** icon to open the Advanced options page to enable SSH and set username and password.

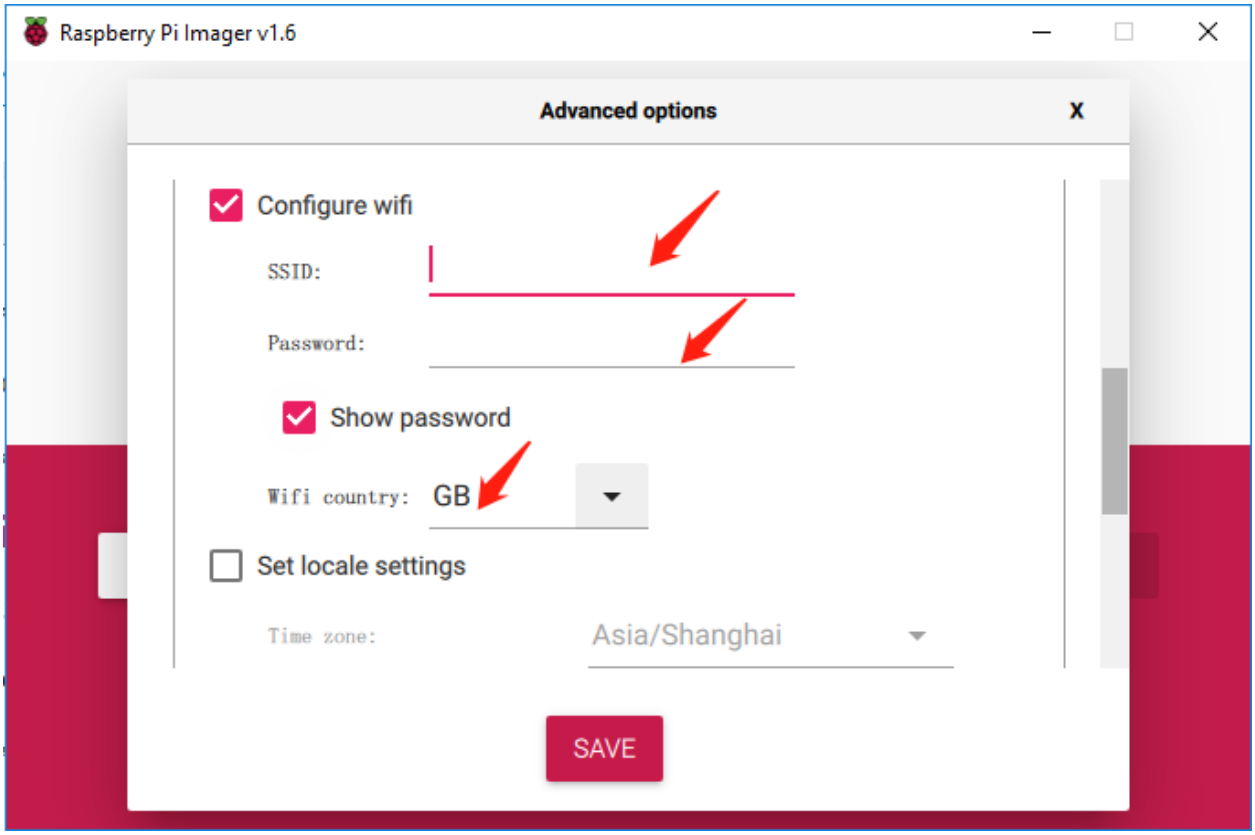
Note:

- Now that the Raspberry Pi doesn't have a default password, you will need to set it yourself. Also, the username can be changed.
- For remote access, you will also need to enable SSH manually.



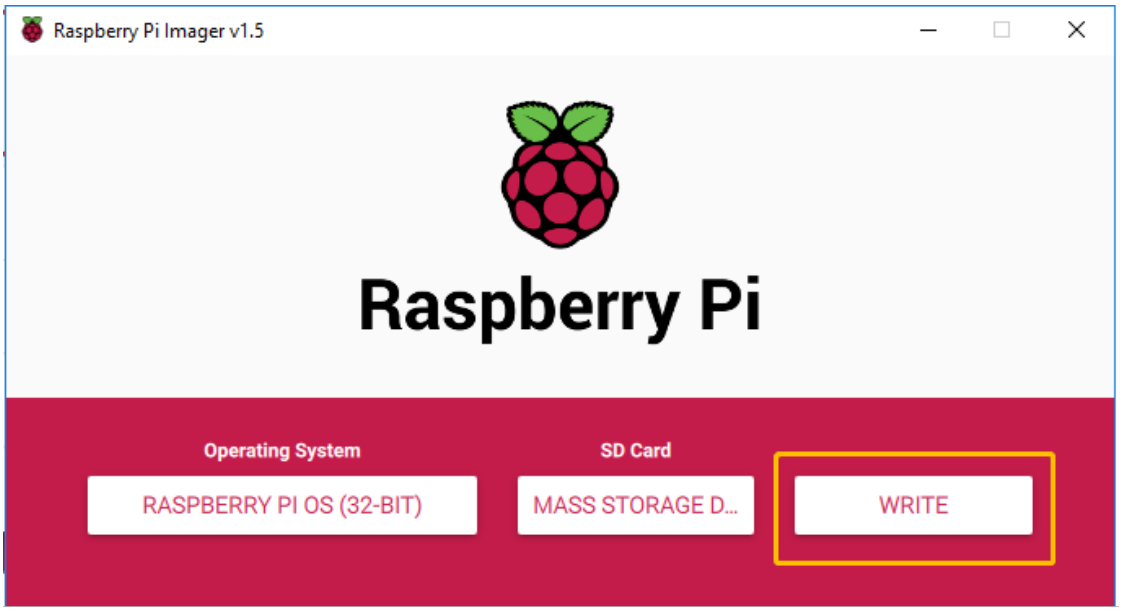
Then scroll down to complete the wifi configuration and click SAVE.

Note: Wi-Fi country should be set the two-letter ISO/IEC alpha2 code for the country in which you are using your Raspberry Pi, please refer to the following link: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements



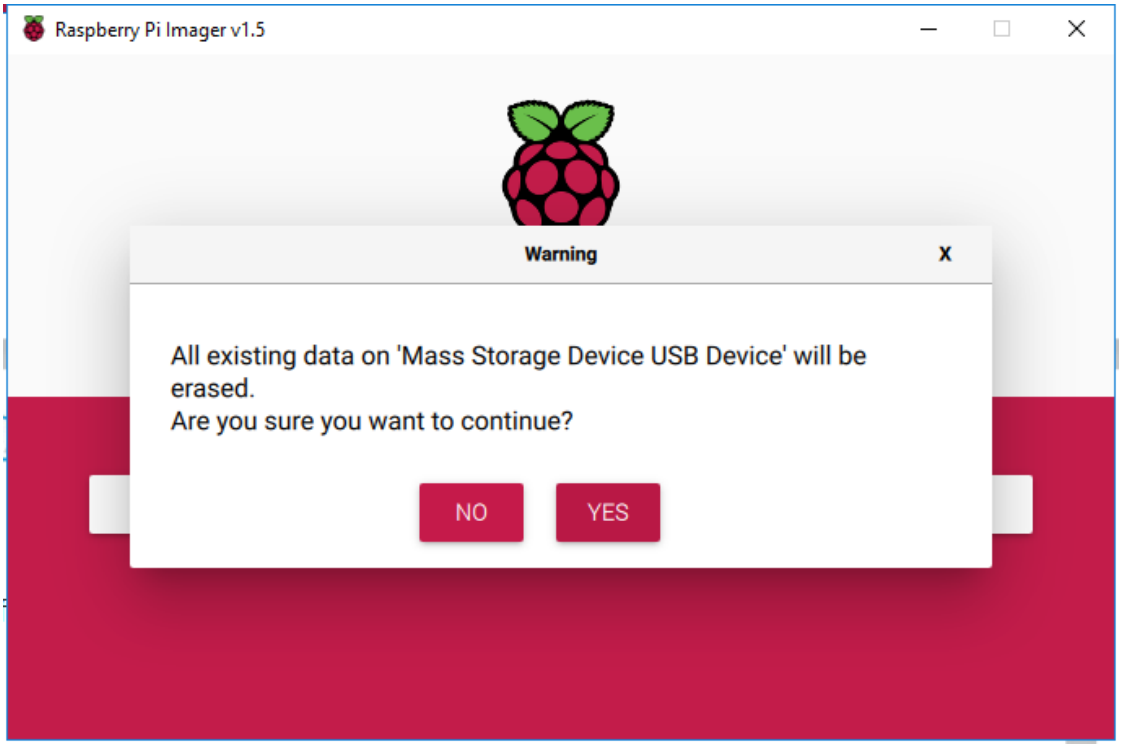
Step 7

Click the WRITE button.



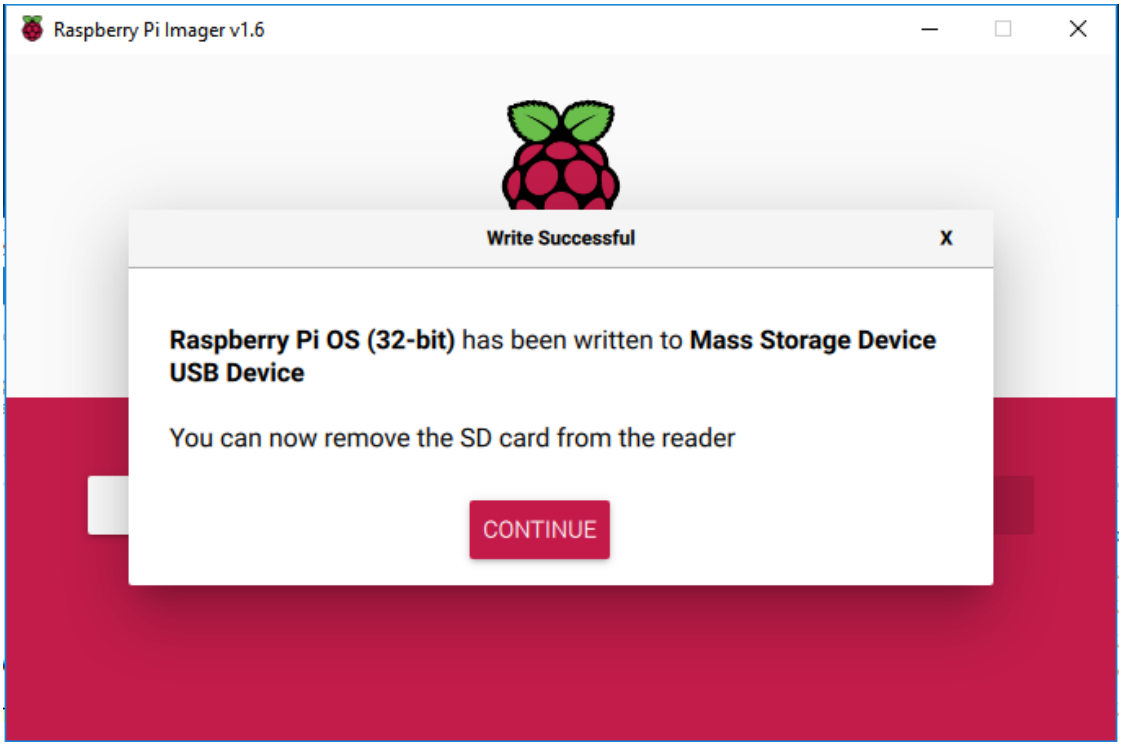
Step 8

If your SD card currently has any files on it, you may wish to back up these files first to prevent you from permanently losing them. If there is no file to be backed up, click Yes.



Step 9

After waiting for a period of time, the following window will appear to represent the completion of writing.



2.3 Set up Your Raspberry Pi

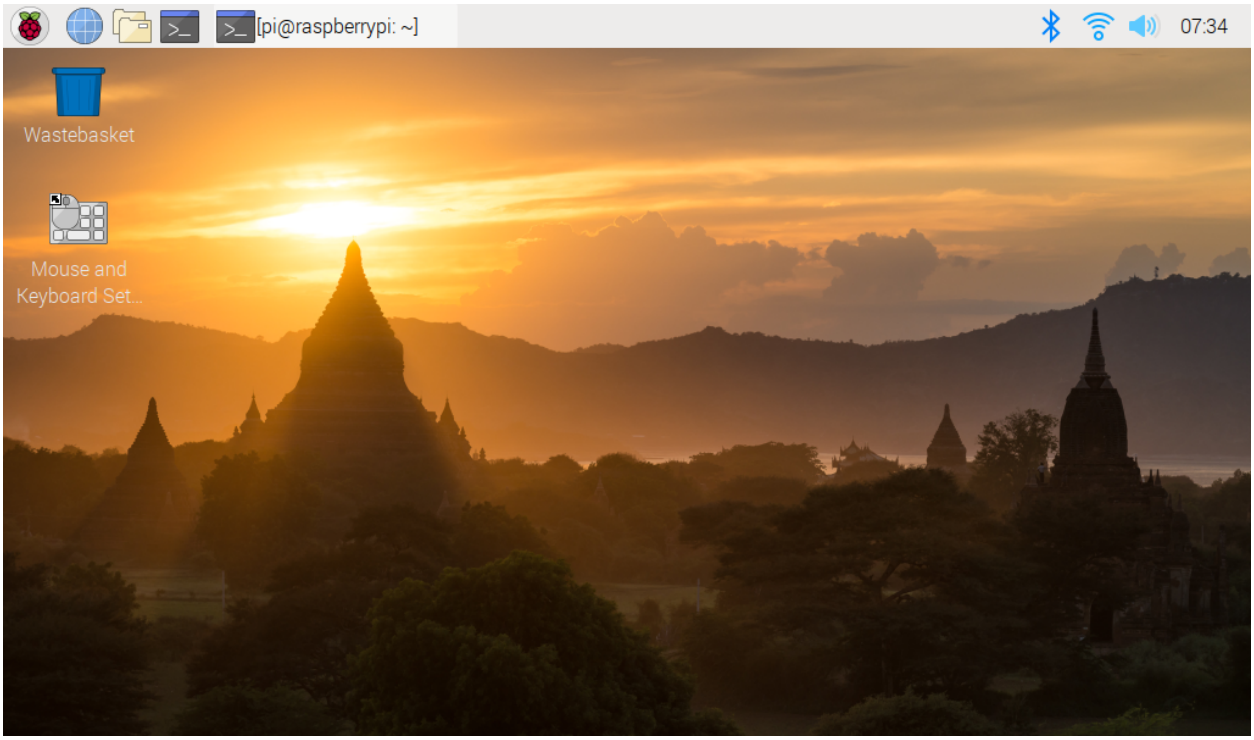
2.3.1 If You Have a Screen

If you have a screen, you can use the NOOBS (New Out Of Box System) to install the Raspberry Pi OS.

Required Components

Any Raspberry Pi	1 * Power Adapter
1 * Monitor	1 * Monitor Power Adapter
1 * HDMI cable	1 * Mirco SD card
1 * Mouse	1 * Keyboard
1 * Personal Computer	

1. Insert the SD card you've set up with Raspberry Pi OS into the micro SD card slot on the underside of your Raspberry Pi.
2. Plug in the Mouse and Keyboard.
3. Connect the screen to Raspberry Pi's HDMI port and make sure your screen is plugged into a wall socket and switched on. (If you use a Raspberry Pi 4, you need to connect the screen to the HDMI0 (nearest the power in port).)
4. Use the power adapter to power the Raspberry Pi. After a few seconds, the Raspberry Pi OS desktop will be displayed.



2.3.2 If You Have No Screen

If we don't have a screen, we can directly write the Raspberry Pi OS system to the Micro SD card and we can control the Raspberry Pi on PC remotely by directly modifying the configuration file of the network settings in the Micro SD card.

Get the IP Address

After the Raspberry Pi is connected to WIFI, we need to get the IP address of it. There are many ways to know the IP address, and two of them are listed as follows.

1. Checking via the router

If you have permission to log in the router(such as a home network), you can check the addresses assigned to Raspberry Pi on the admin interface of router.

The default hostname of the Raspberry Pi OS is **raspberrypi**, and you need to find it. (If you are using ArchLinuxARM system, please find alarmpi.)

2. Network Segment Scanning

You can also use network scanning to look up the IP address of Raspberry Pi. You can apply the software, Advanced IP scanner and so on.

Scan the IP range set, and the name of all connected devices will be displayed. Similarly, the default hostname of the Raspberry Pi OS is **raspberrypi**, now you need to find the hostname.

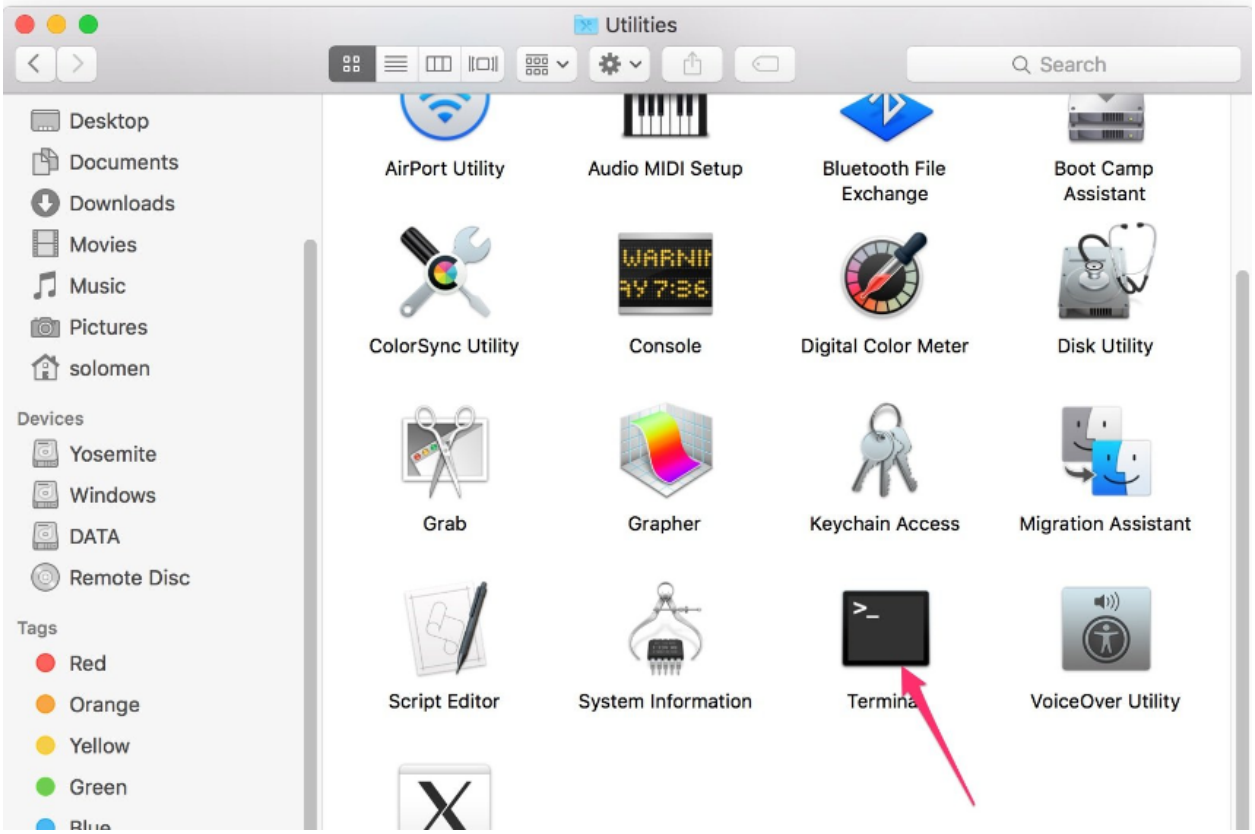
Use the SSH Remote Control

We can open the Bash Shell of Raspberry Pi by applying SSH. Bash is the standard default shell of Linux. The Shell itself is a program written in C that is the bridge linking the customers and Unix/Linux. Moreover, it can help to complete most of the work needed.

For Linux or/Mac OS X Users

Step 1

Go to **Applications->Utilities**, find the **Terminal**, and open it.



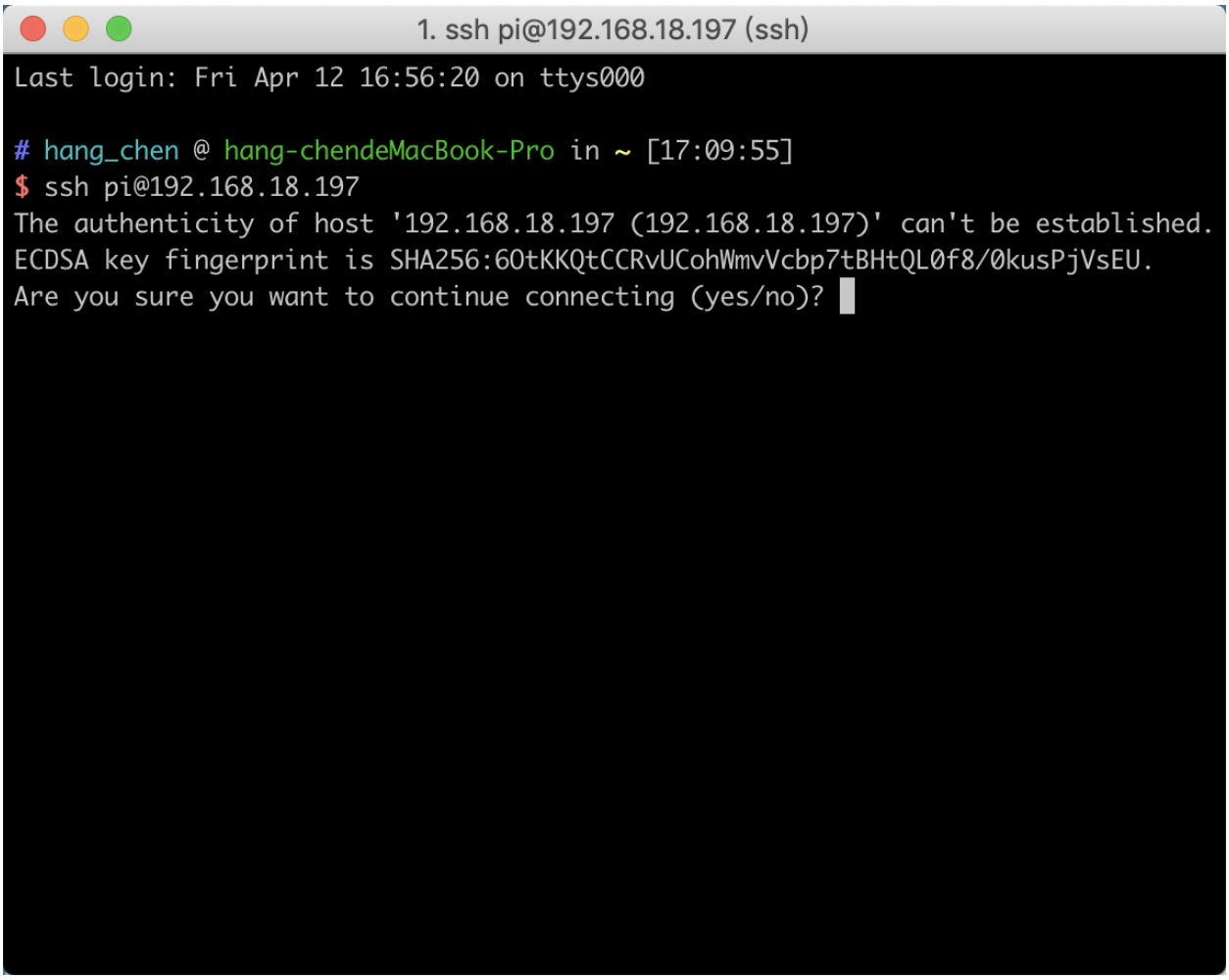
Step 2

Type in **ssh pi@ip_address** . “pi” is your username and “ip_address” is your IP address. For example:

```
ssh pi@192.168.18.197
```

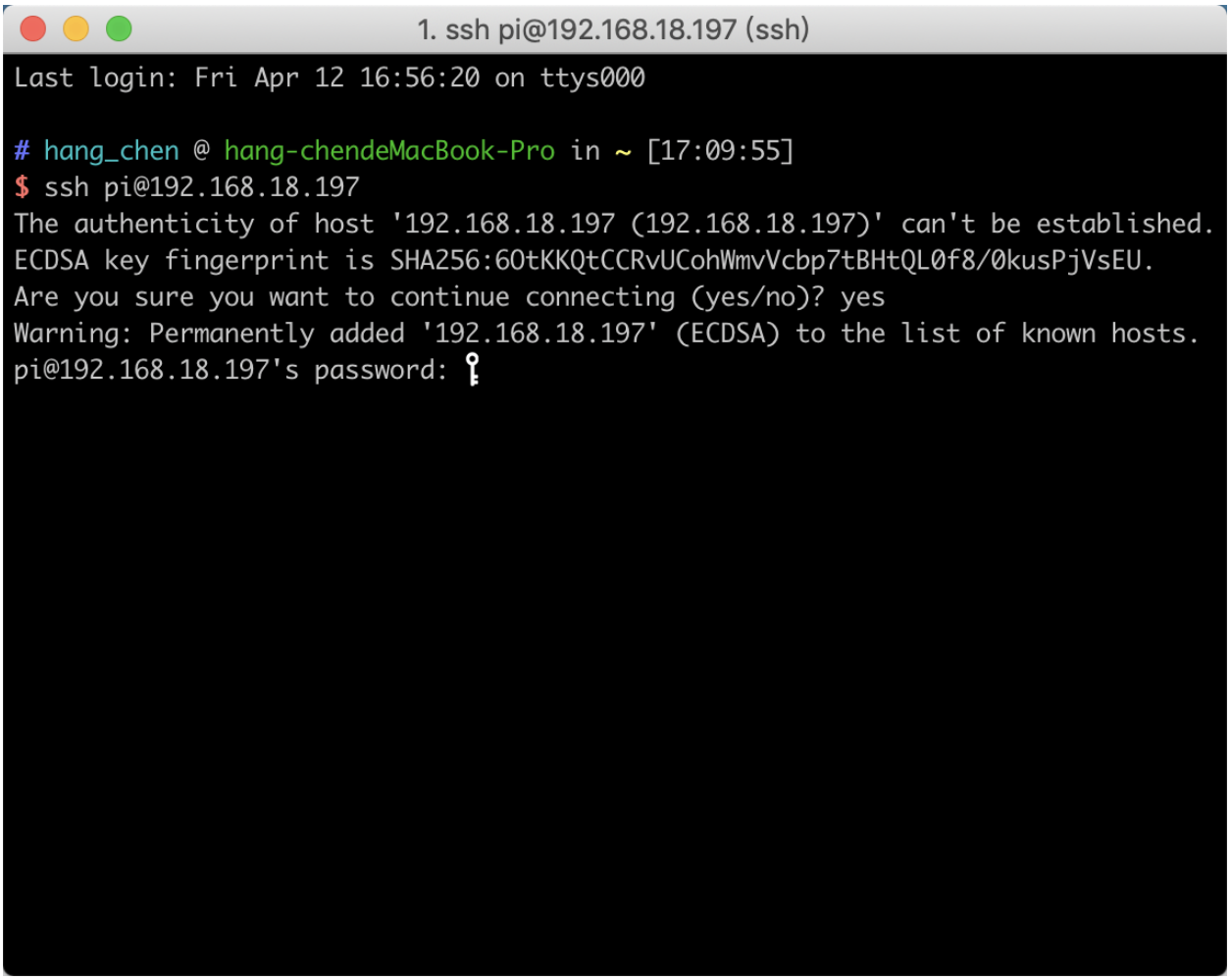
Step 3

Input “yes”.



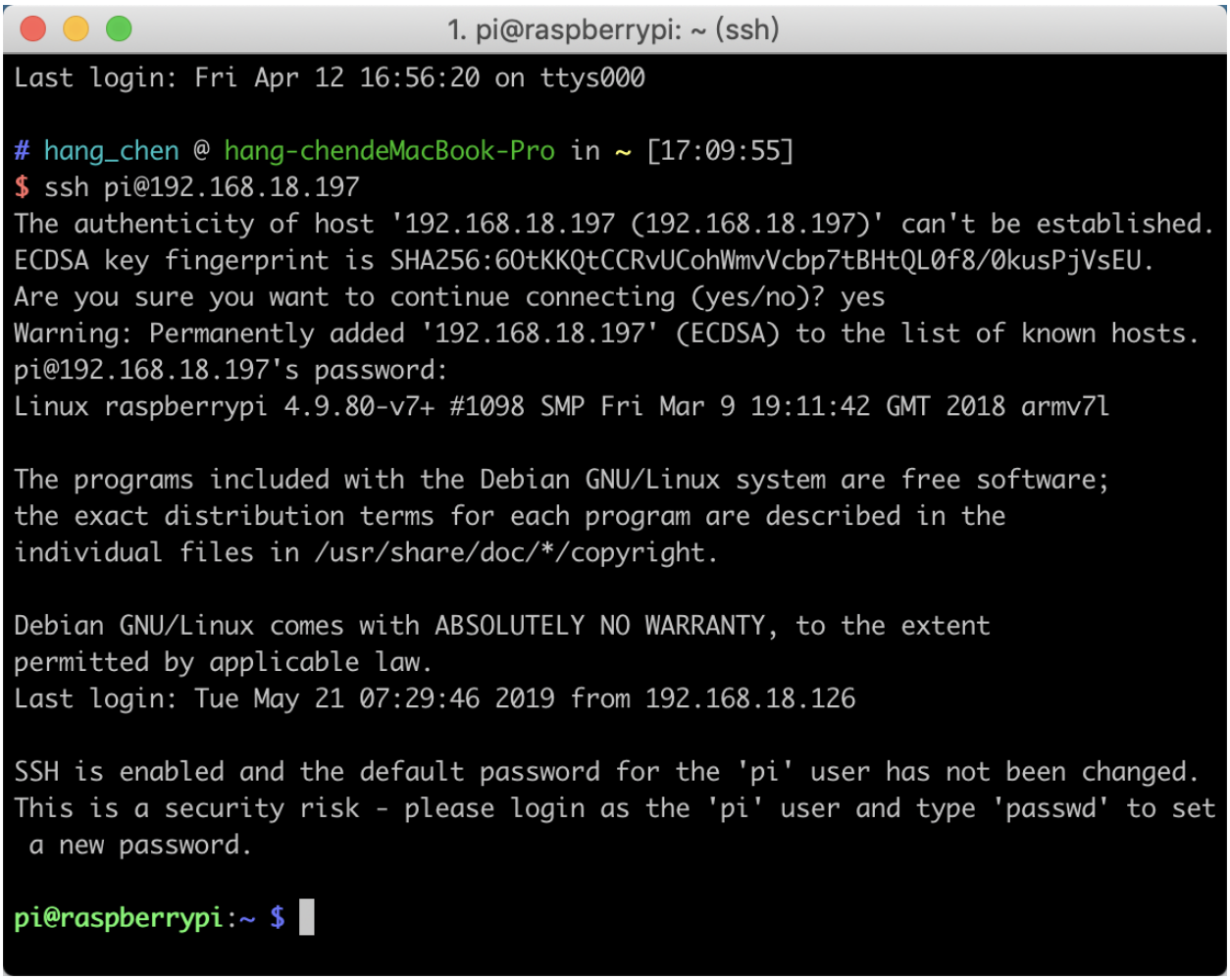
Step 4

Input the passcode and the default password is **raspberry**.



Step 5

We now get the Raspberry Pi connected and are ready to go to the next step.



Note: When you input the password, the characters do not display on window accordingly, which is normal. What you need is to input the correct passcode.

For Windows Users

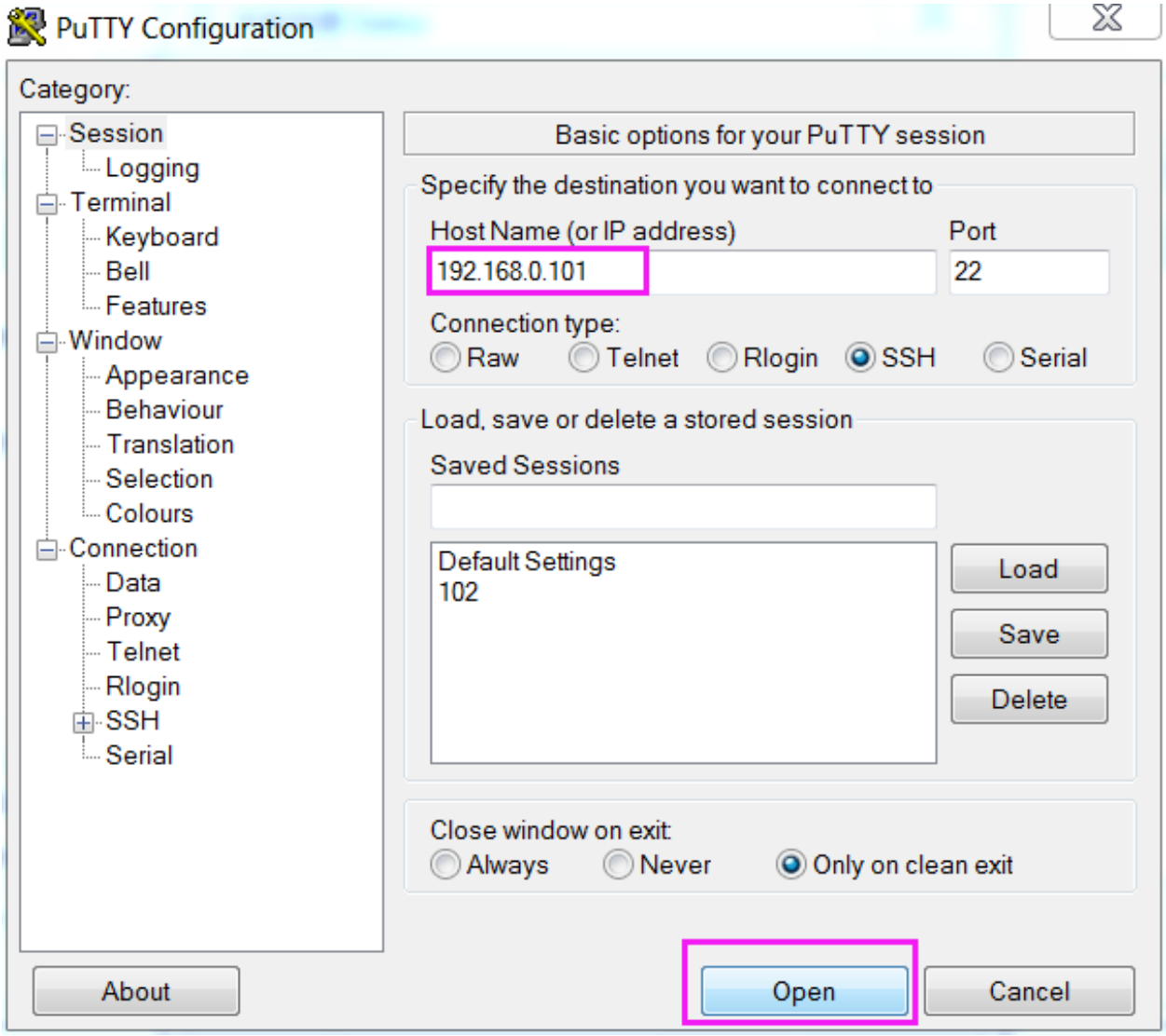
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend PuTTY.

Step 1

Download PuTTY.

Step 2

Open PuTTY and click **Session** on the left tree-alike structure. Enter the IP address of the RPi in the text box under **Host Name (or IP address)** and 22 under **Port** (by default it is 22).

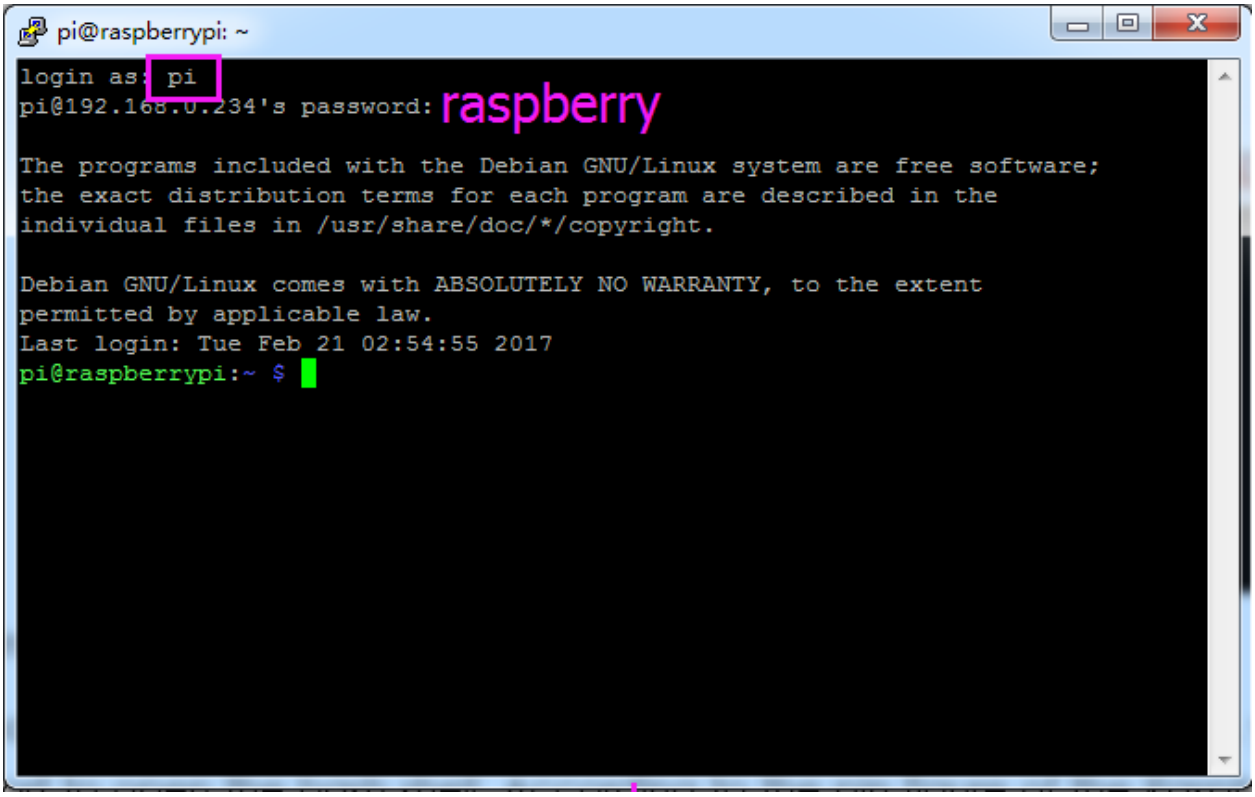


Step 3

Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

When the PuTTY window prompts "**login as:**", type in "**pi**" (the user name of the RPi), and **password:** "raspberry" (the default one, if you haven't changed it).



Step 5

Here, we get the Raspberry Pi connected and it is time to conduct the next steps.

Note: When you input the password, the characters do not display on window accordingly, which is normal. What you need is to input the correct password.

Step 6

Here, we get the Raspberry Pi connected and it is time to conduct the next steps.

Note: If you are not satisfied with using the command window to control the Raspberry Pi, you can also use the remote desktop function, which can help us manage the files in the Raspberry Pi easily. For details on how to do this, please refer to *Remote Desktop*.

CHAPTER

THREE

LIBRARIES

Two important libraries are used in programming with Raspberry Pi, and they are wiringPi and RPi.GPIO. The Raspberry Pi OS image of Raspberry Pi installs them by default, so you can use them directly.

3.1 RPi.GPIO

If you are a Python user, you can program GPIOs with API provided by RPi.GPIO.

RPi.GPIO is a module to control Raspberry Pi GPIO channels. This package provides a class to control the GPIO on a Raspberry Pi. For examples and documents, visit http://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

Test whether RPi.GPIO is installed or not, type in python:

```
python
```

```
pi@raspberrypi:~ $ python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

In Python CLI, input “import RPi.GPIO”, If no error prompts, it means RPi.GPIO is installed.

```
import RPi.GPIO
```

```
pi@raspberrypi:~ $ python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO
>>>
```

If you want to quit python CLI, type in:

```
exit()
```

```
>>> exit()
pi@raspberrypi:~ $
```

3.2 WiringPi

`wiringPi` is a C language GPIO library applied to the Raspberry Pi. It complies with GUN Lv3. The functions in wiringPi are similar to those in the wiring system of Arduino. They enable the users familiar with Arduino to use wiringPi more easily.

`wiringPi` includes lots of GPIO commands which enable you to control all kinds of interfaces on Raspberry Pi.

Please run the following command to install `wiringPi` library.

```
sudo apt-get update
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
./build
```

You can test whether the wiringPi library is installed successfully or not by the following instruction.

```
gpio -v
```

```
pi@raspberrypi:~/davinci-kit-for-raspberry-pi/c/1.1.1 $ 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.
```

Check the GPIO with the following command:

```
gpio readall
```

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | 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 | 0 |  7 || 8  | 1 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 1 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 1 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | OUT  | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 0 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 0 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 1 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

For more details about wiringPi, you can refer to WiringPi.

CHAPTER

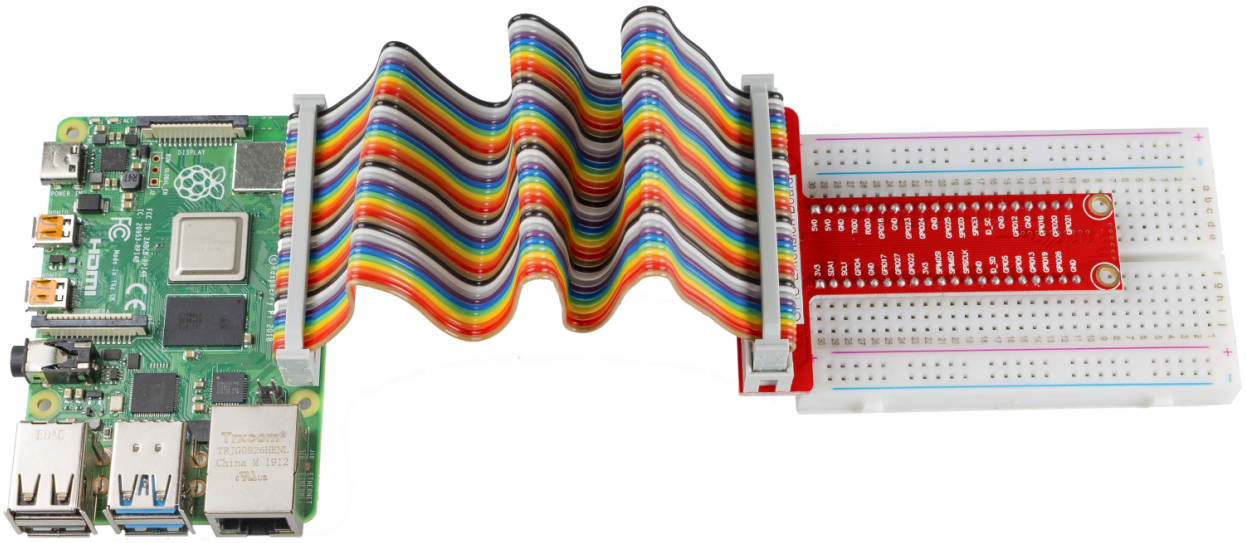
FOUR

GPIO EXTENSION BOARD

Connect to Raspberry Pi

Before starting to learn the commands, you first need to know more about the pins of the Raspberry Pi, which is key to the subsequent study.

We can easily lead out pins of the Raspberry Pi to breadboard by GPIO Extension Board to avoid GPIO damage caused by frequent plugging in or out. This is our 40-pin GPIO Extension Board and GPIO cable for Raspberry Pi model B+, 2 model B and 3, 4 model B.



Pin Number

The pins of Raspberry Pi have three kinds of ways to name and they are wiringPi, BCM and Board. Among these naming methods, 40-pin GPIO Extension board uses the naming method, BCM. But for some special pins, such as I2C port and SPI port, they use the Name that comes with themselves. The following table shows us the naming methods of WiringPi, Board and the intrinsic Name of each pin on GPIO Extension board. For example, for the GPIO17, the Board naming method of it is 11, the wiringPi naming method is 0, and the intrinsic naming method of it is GPIO0.

Note: 1In C Language, what is used is the naming method WiringPi.

2In Python Language, the applied naming methods are Board and BCM, and the function GPIO.setmode() is used to set them.

Name	WiringPi	Board	BCM		Board	WiringPi	Name
		GPIO Extention Board					
3.3V	3V3	1	3V3	5.0V	2	5.0V	5V
SDA	8	3	SDA	5.0V	4	5.0V	5V
SCL	9	5	SCL	GND	6	GND	0V
GPIO7	7	7	GPIO4	TXD	8	15	TXD
0V	GND	9	GND	RXD	10	16	RXD
GPIO0	0	11	GPIO17	GPIO18	12	1	GPIO1
GPIO2	2	13	GPIO27	GND	14	GND	0V
GPIO3	3	15	GPIO22	GPIO23	16	4	GPIO4
3.3V	3.3V	17	3.3V	GPIO24	18	5	GPIO5
MOSI	12	19	MOSI	GND	20	GND	0V
MISO	13	21	MISO	GPIO25	22	6	GPIO6
SCLK	14	23	SCLK	CE0	24	10	CEO
0V	GND	25	GND	CE1	26	11	CE1
IN_SDA	30	27	EED	EEC	28	31	ID_SCL
GPIO21	21	29	GPIO5	GND	30	GND	0V
GPIO22	22	31	GPIO6	GPIO12	32	26	GPIO26
GPIO23	23	33	GPIO13	GND	34	GND	0V
GPIO24	24	35	GPIO19	GPIO16	36	27	GPIO27
GPIO25	25	37	GPIO26	GPIO20	38	28	GPIO28
0V	GND	39	GND	GPIO21	40	29	GPIO29

CHAPTER

FIVE

DOWNLOAD THE CODE

Change directory to */home/pi*

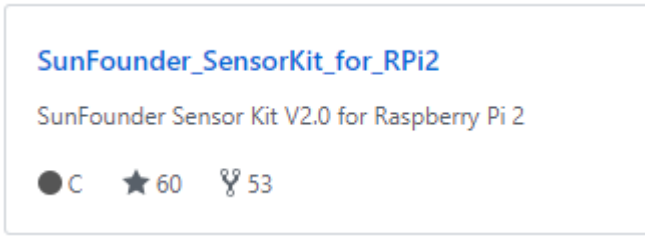
```
cd /home/pi/
```

Note: cd, short for change directory is to change to the intended directory from the current path. Informally, here is to go to the path */home/pi/*.

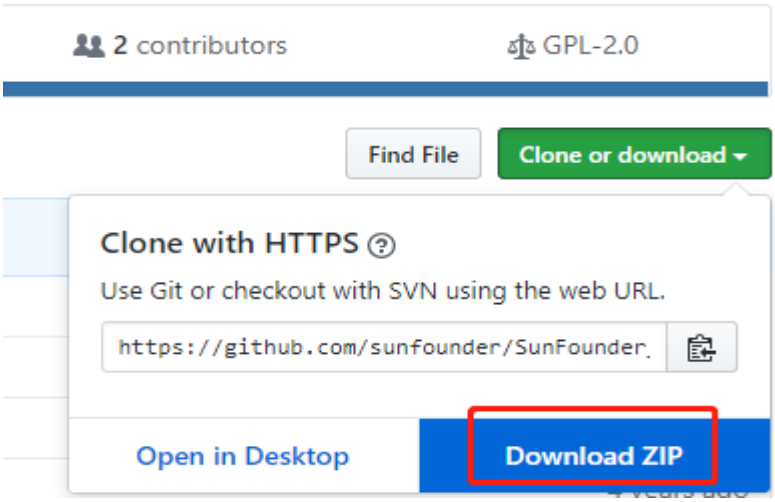
Clone the repository from github (C code and python code)

```
git clone https://github.com/sunfounder/SunFounder_SensorKit_for_RPi2
```

The advantage of this method is that, you can download the latest code any time you want, and then place the code under the path */home/pi/*. But in case of incorrect typing which is possible especially when you're strange to the commands, you can just enter *github.com/sunfounder* at the address bar of a web browser, and on the page directed find the code for Sensor Kit.



Click on the repository. On the page directed, click **Clone or download** on the right side.



After download, transfer the package to */home/pi/*.

Now you can start the experiments. Let's rock!

CHAPTER

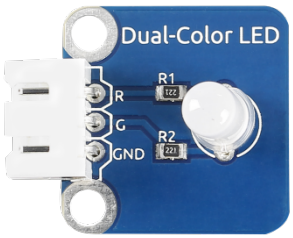
SIX

LESSONS

6.1 Lesson 1 Dual-Color LED

Introduction

A dual-color light emitting diode (LED) is capable of emitting two different colors of light, typically red and green, rather than only one color. It is housed in a 3mm or 5mm epoxy package. It has 3 leads; common cathode or common anode is available. A dual-color LED features two LED terminals, or pins, arranged in the circuit in anti-parallel and connected by a cathode/anode. Positive voltage can be directed towards one of the LED terminals, causing that terminal to emit light of the corresponding color; when the direction of the voltage is reversed, the light of the other color is emitted. In a dual-color LED, only one of the pins can receive voltage at a time. As a result, this type of LED frequently functions as indicator lights for a variety of devices, including televisions, digital cameras, and remote controls.



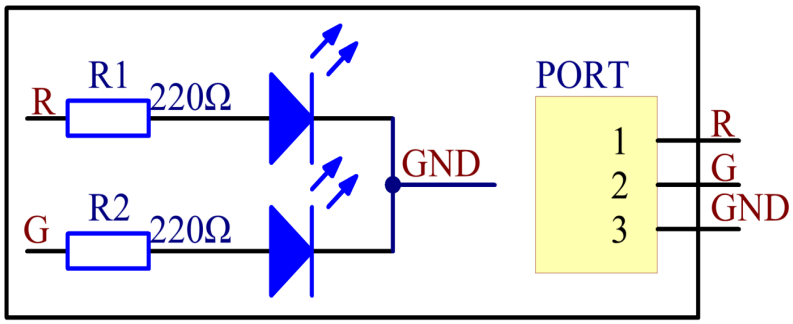
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- Several Jumper wires
- 1 * Dual-color LED module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

Connect pin R and G to GPIOs of Raspberry Pi, program the Raspberry Pi to change the color of the LED from red to green, and then use PWM to mix into other colors.

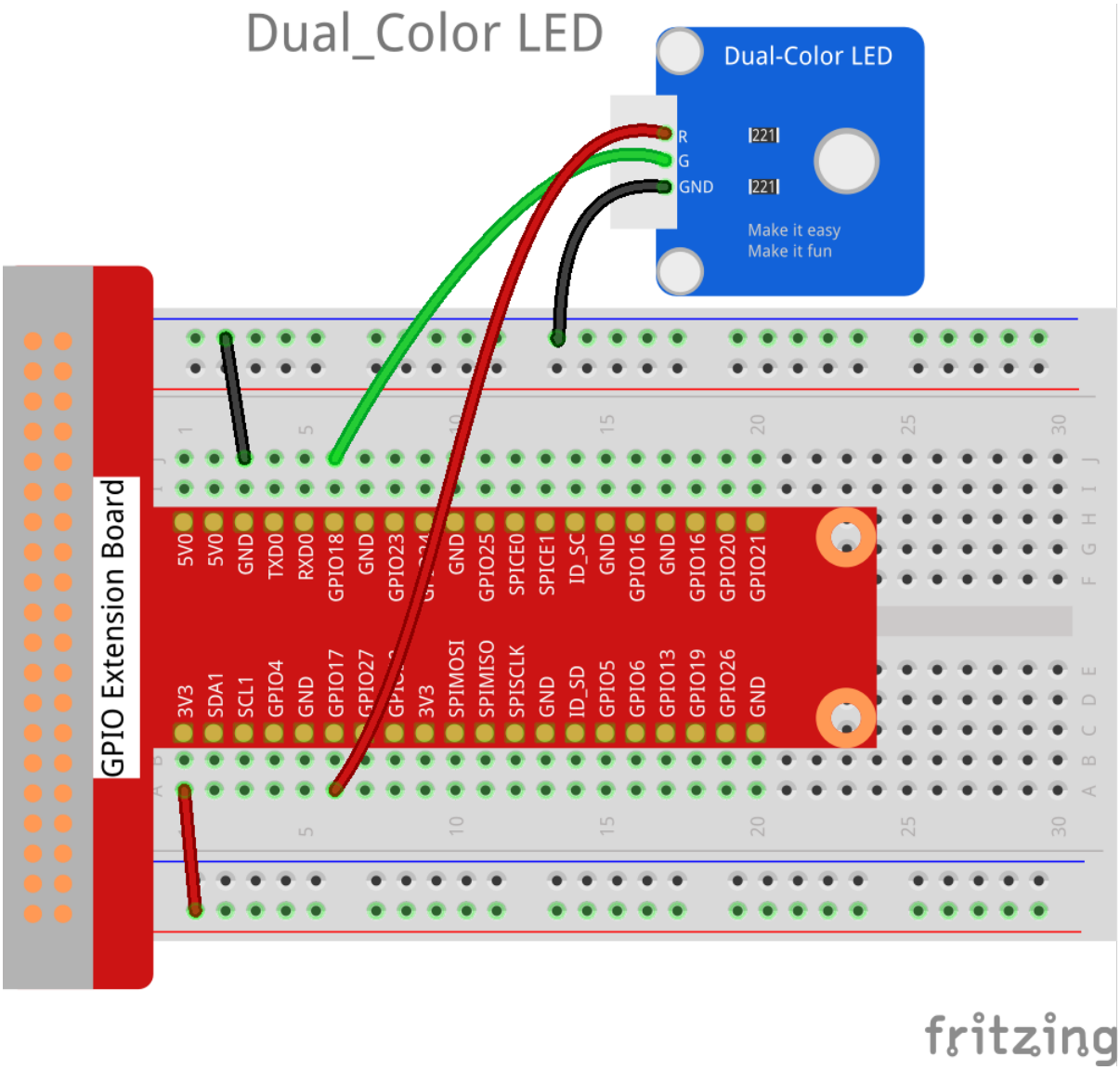
The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Dual-Color LED Module
GPIO0	GPIO17	R
GND	GND	GND
GPIO1	GPIO18	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/01_dule_color_led/
```

Step 3: Compile.

```
gcc dule_color_led.c -lwiringPi -lpthread
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define uchar unsigned char

#define LedPinRed    0
#define LedPinGreen  1

void ledInit(void)
{
    softPwmCreate(LedPinRed,  0, 100);
    softPwmCreate(LedPinGreen,0, 100);
}

void ledColorSet(uchar r_val, uchar g_val)
{
    softPwmWrite(LedPinRed,   r_val);
    softPwmWrite(LedPinGreen, g_val);
}

int main(void)
{
    int i;

    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
        printf("setup wiringPi failed !");
        return 1;
    }
    //printf("linker LedPin : GPIO %d(wiringPi pin)\n",LedPin); //when initialize␣
↪wiring successfully,print message to screen

    ledInit();

    while(1){
        ledColorSet(0xff,0x00);   //red
        delay(500);
        ledColorSet(0x00,0xff);   //green
        delay(500);
        ledColorSet(0xff,0x45);
        delay(500);
        ledColorSet(0xff,0xff);
        delay(500);
        ledColorSet(0x7c,0xfc);
        delay(500);
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

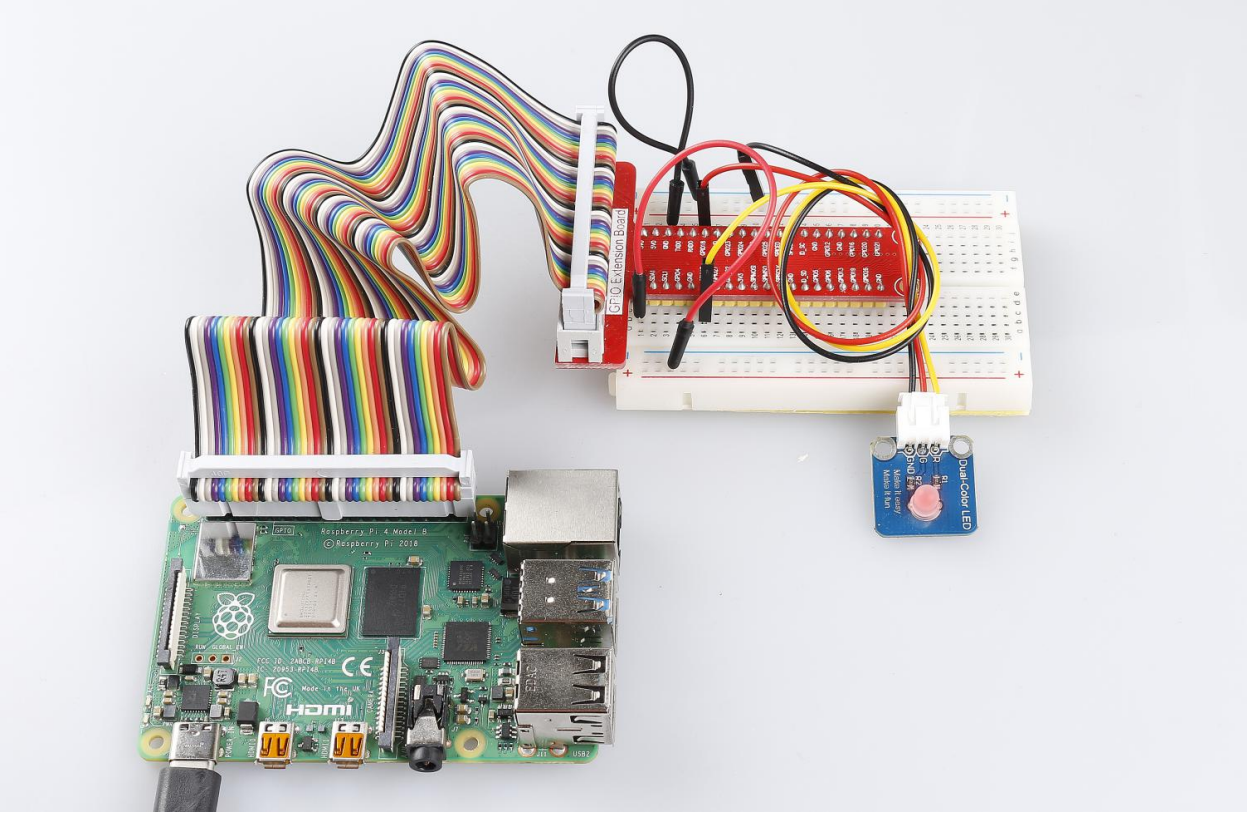
Step 3: Run.

```
sudo python3 01_dule_color_led.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

colors = [0xFF00, 0x00FF, 0x0FF0, 0xF00F]
pins = (11, 12)  # pins is a dict

GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
GPIO.setup(pins, GPIO.OUT)    # Set pins' mode is output
GPIO.output(pins, GPIO.LOW)  # Set pins to LOW(0V) to off led

p_R = GPIO.PWM(pins[0], 2000)  # set Frequece to 2KHz
p_G = GPIO.PWM(pins[1], 2000)

p_R.start(0)       # Initial duty Cycle = 0(leds off)
p_G.start(0)

def map(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def setColor(col):   # For example : col = 0x1122
    R_val = col  >> 8
    G_val = col & 0x00FF

    R_val = map(R_val, 0, 255, 0, 100)
    G_val = map(G_val, 0, 255, 0, 100)

    p_R.ChangeDutyCycle(R_val)      # Change duty cycle
    p_G.ChangeDutyCycle(G_val)

def loop():
    while True:
        for col in colors:
            setColor(col)
            time.sleep(0.5)

def destroy():
    p_R.stop()
    p_G.stop()
    GPIO.output(pins, GPIO.LOW)     # Turn off all leds
    GPIO.cleanup()

if __name__ == "__main__":
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

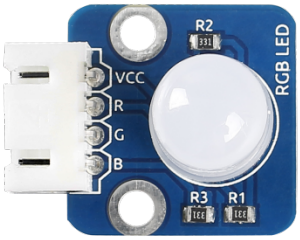
You can see the dual-color LED render green, red, and mixed colors.



6.2 Lesson 2 RGB LED Module

Introduction

RGB LED modules can emit various colors of light. Three LEDs of red, green, and blue are packaged into a transparent or semitransparent plastic shell with four pins led out. The three primary colors of red, green, and blue can be mixed and compose all kinds of colors by brightness, so you can make an RGB LED emit colorful light by controlling the circuit.



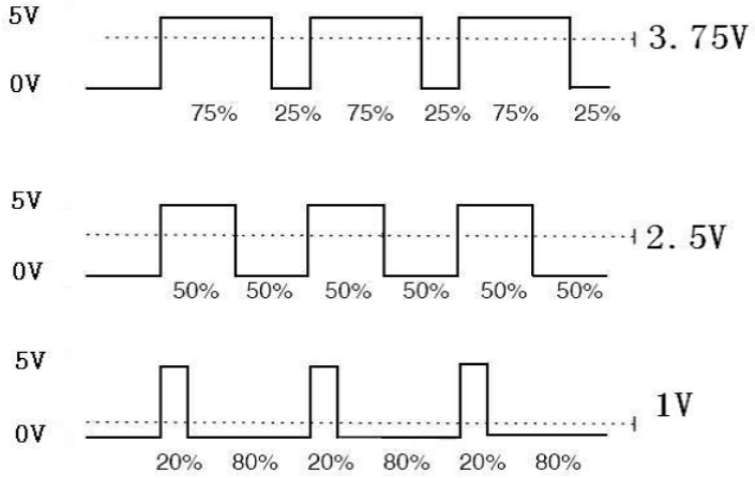
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- Several Jumper wires
- 1 * RGB LED module
- 1 * 4-Pin anti-reverse cable

Experimental Principle

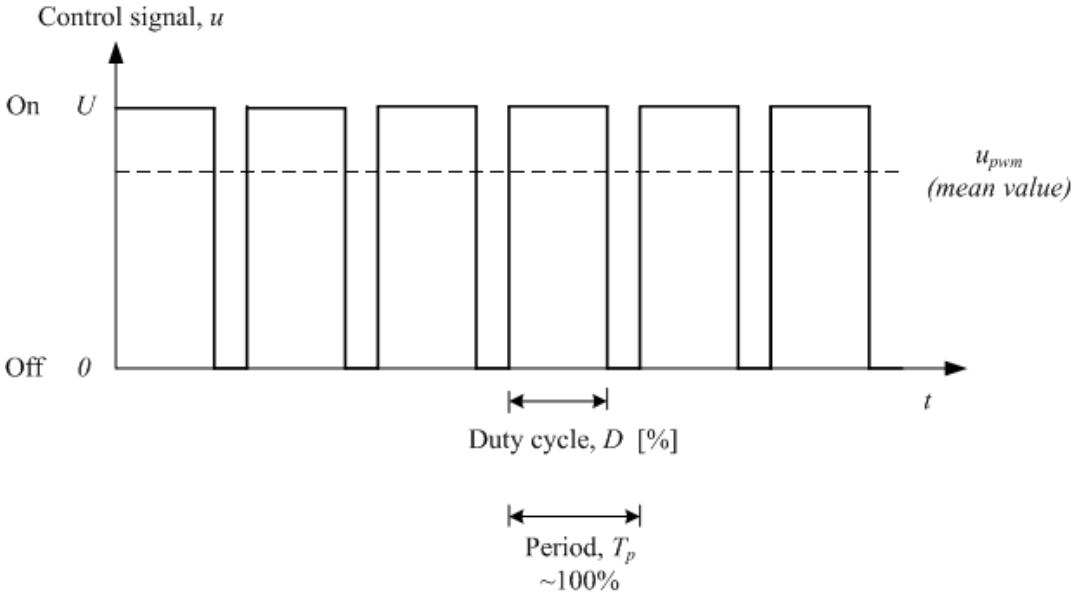
In this experiment, we will use PWM technology to control the brightness of RGB.

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.



We can see from the top oscillogram that the amplitude of DC voltage output is 5V. However, the actual voltage output is only 3.75V through PWM, for the high level only takes up 75% of the total voltage within a period.

Here are the three basic parameters of PWM:

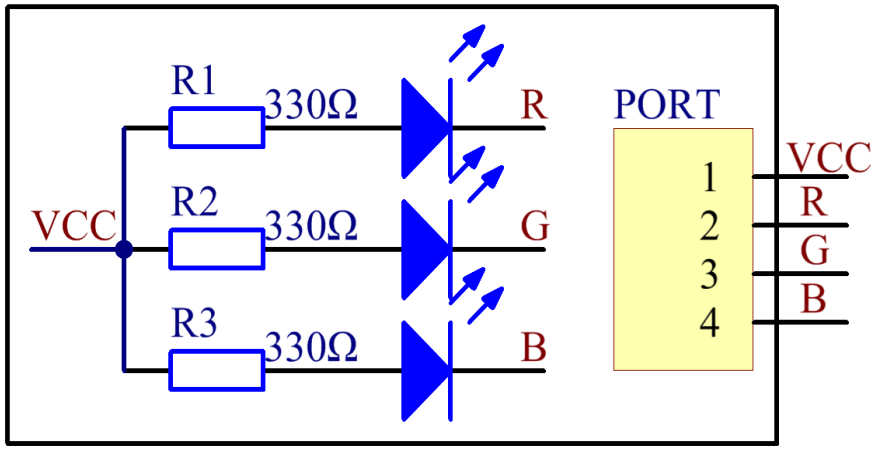


1. The term **duty cycle** describes the proportion of "on" time to the regular interval or "period" of time
2. **Period** describes the reciprocal of pulses in one second.
3. The voltage amplitude here is 0V-5V.

Here we input any value between 0 and 255 to the three pins of the RGB LED to make it display different colors.

RGB LEDs can be categorized into common anode LED and common cathode LED. In this experiment, we use a common cathode RGB LED.

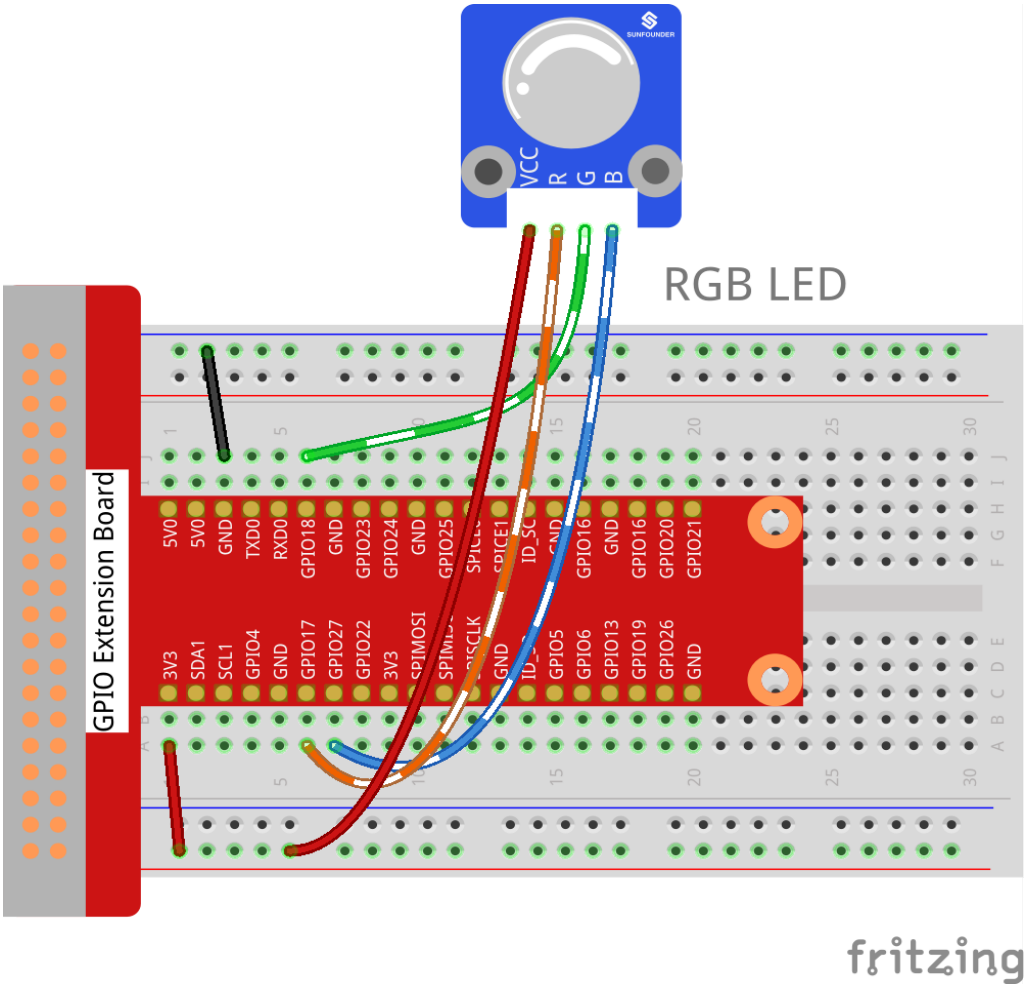
The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit according to the following method.

Raspberry Pi	GPIO Extension Board	RGB LED Module
3.3V	3V3	VCC
GPIO0	GPIO17	R
GPIO1	GPIO18	G
GPIO2	GPIO27	B



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/02_rgb_led/
```

Step 3: Compile.

```
gcc rgb_led.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

(continues on next page)

(continued from previous page)

```
#define uchar unsigned char

#define LedPinRed    0
#define LedPinGreen  1
#define LedPinBlue   2

void ledInit(void)
{
   softPwmCreate(LedPinRed,  0, 100);
   softPwmCreate(LedPinGreen,0, 100);
   softPwmCreate(LedPinBlue, 0, 100);
}

void ledColorSet(uchar r_val, uchar g_val, uchar b_val)
{
   softPwmWrite(LedPinRed,   r_val);
   softPwmWrite(LedPinGreen, g_val);
   softPwmWrite(LedPinBlue,  b_val);
}

int main(void)
{
   int i;

   if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
      printf("setup wiringPi failed !");
      return 1;
   }
   //printf("linker LedPin : GPIO %d(wiringPi pin)\n",LedPin); //when initialize␣
↪wiring successfully,print message to screen

   ledInit();

   while(1){
      ledColorSet(0xff,0x00,0x00);   //red
      delay(500);
      ledColorSet(0x00,0xff,0x00);   //green
      delay(500);
      ledColorSet(0x00,0x00,0xff);   //blue
      delay(500);

      ledColorSet(0xff,0xff,0x00);   //yellow
      delay(500);
      ledColorSet(0xff,0x00,0xff);   //pick
      delay(500);
      ledColorSet(0xc0,0xff,0x3e);
      delay(500);

      ledColorSet(0x94,0x00,0xd3);
      delay(500);
      ledColorSet(0x76,0xee,0x00);
      delay(500);
      ledColorSet(0x00,0xc5,0xcd);
      delay(500);

   }
```

(continues on next page)

(continued from previous page)

```
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 02_rgb_led.py
```

Code

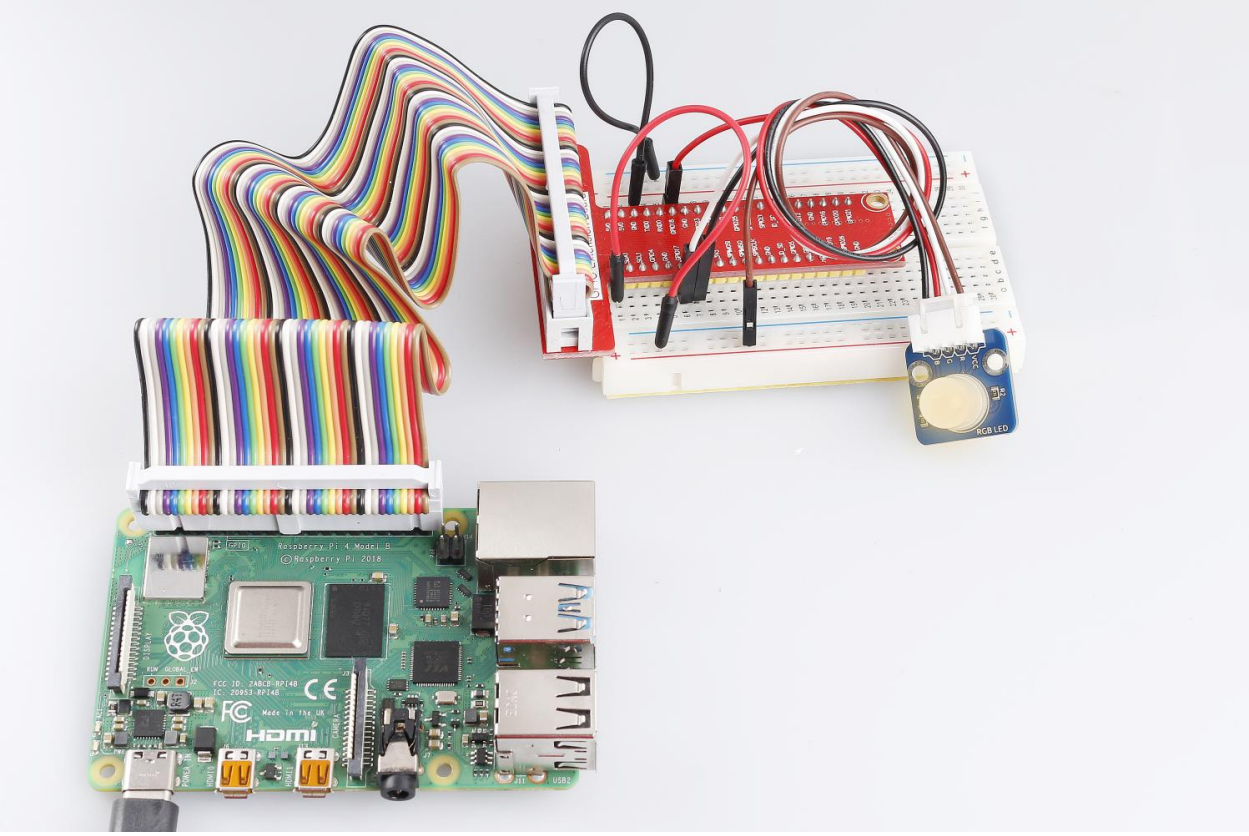
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

colors = [0xFF0000, 0x00FF00, 0x0000FF, 0xFFFF00, 0xFF00FF, 0x00FFFF]
R = 11
G = 12
B = 13

def setup(Rpin, Gpin, Bpin):
    global pins
    global p_R, p_G, p_B
    pins = {'pin_R': Rpin, 'pin_G': Gpin, 'pin_B': Bpin}
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    for i in pins:
        GPIO.setup(pins[i], GPIO.OUT)   # Set pins' mode is output
        GPIO.output(pins[i], GPIO.HIGH) # Set pins to high(+3.3V) to off led

    p_R = GPIO.PWM(pins['pin_R'], 2000)  # set Frequece to 2KHz
    p_G = GPIO.PWM(pins['pin_G'], 1999)
    p_B = GPIO.PWM(pins['pin_B'], 5000)

    p_R.start(100)      # Initial duty Cycle = 0(leds off)
    p_G.start(100)
    p_B.start(100)

def map(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def off():
    GPIO.setmode(GPIO.BOARD)
    for i in pins:
        GPIO.setup(pins[i], GPIO.OUT)   # Set pins' mode is output
        GPIO.output(pins[i], GPIO.HIGH)    # Turn off all leds

def setColor(col):   # For example : col = 0x112233
    R_val = (col & 0xff0000) >> 16
    G_val = (col & 0x00ff00) >> 8
    B_val = (col & 0x0000ff) >> 0

    R_val = map(R_val, 0, 255, 0, 100)
```

(continues on next page)

(continued from previous page)

```
    G_val = map(G_val, 0, 255, 0, 100)
    B_val = map(B_val, 0, 255, 0, 100)

    p_R.ChangeDutyCycle(100-R_val)      # Change duty cycle
    p_G.ChangeDutyCycle(100-G_val)
    p_B.ChangeDutyCycle(100-B_val)

def loop():
    while True:
        for col in colors:
            setColor(col)
            time.sleep(1)

def destroy():
    p_R.stop()
    p_G.stop()
    p_B.stop()
    off()
    GPIO.cleanup()

if __name__ == "__main__":
    try:
        setup(R, G, B)
        loop()
    except KeyboardInterrupt:
        destroy()
```

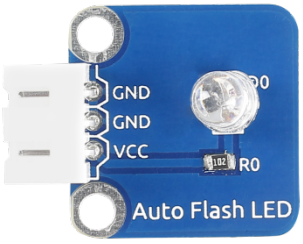
You will see the RGB LED light up, and display different colors in turn.



6.3 Lesson 3 7-Color Auto-flash LED

Introduction

On the 7-Color Auto-flash LED module, the LED can automatically flash built-in colors after power on. It can be used to make quite fascinating light effects.



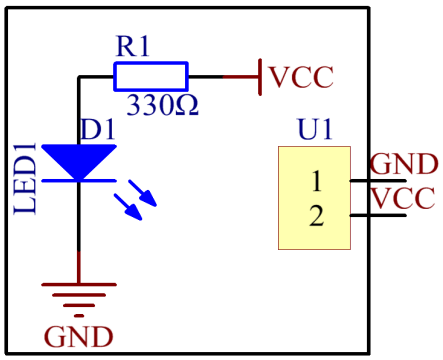
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * 7-color auto-flash LED module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

When it is power on, the 7-color auto-flash LED will flash built-in colors.

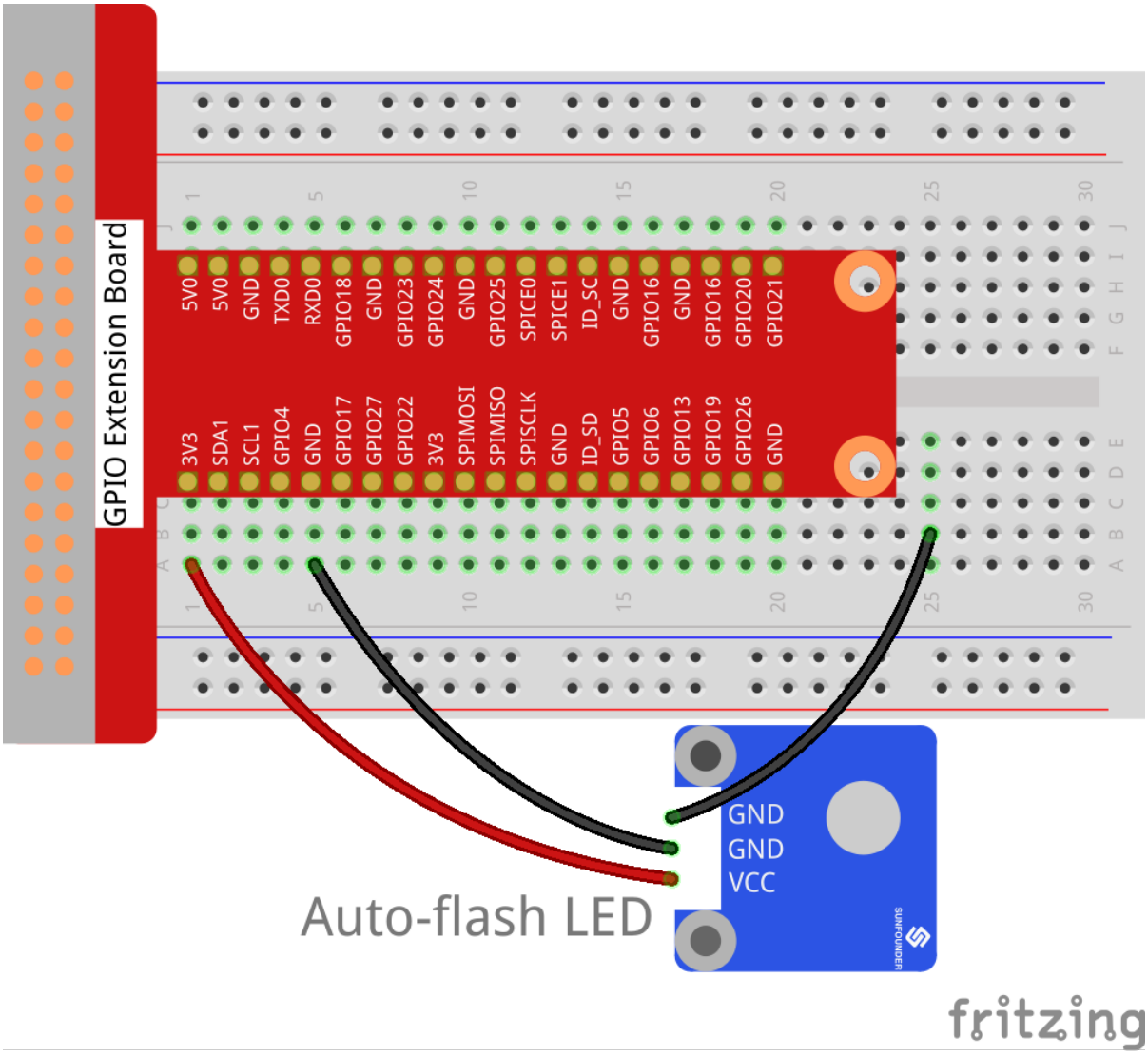
The schematic diagram of the module is as shown below:



Experimental Procedures

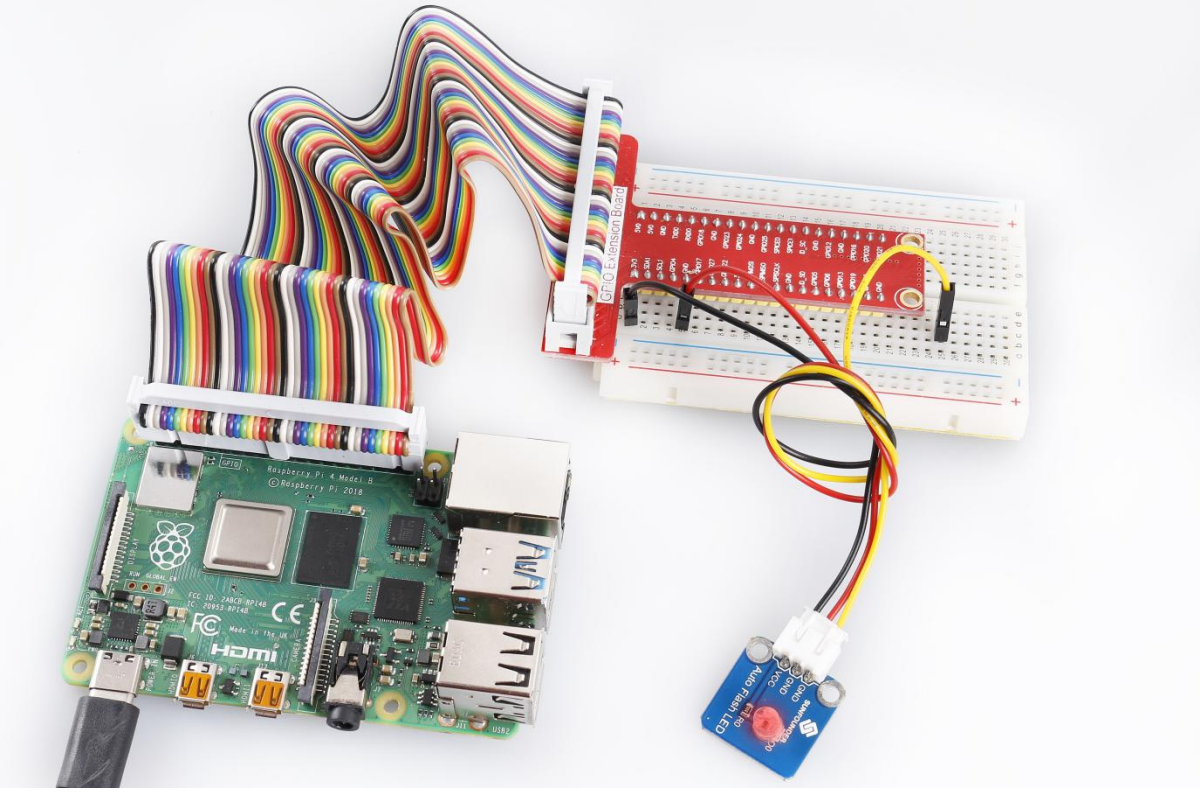
Build the circuit.

Raspberry Pi	GPIO Extension Board	Auto-flash LED Module
GND	GND	GND
3.3V	3V3	VCC



Note: There are two "GND" pins on the module. You only need to connect one of them.

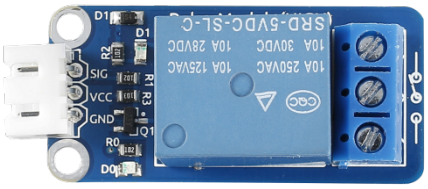
Now, you will see 7-color auto-flash LED flashing seven colors.



6.4 Lesson 4 Relay Module

Introduction

Relay is a device which is used to provide connection between two or more points or devices in response to the input signal applied. It is suitable for driving high power electric equipment, such as light bulbs, electric fans and air conditioning. You can use a relay to control high voltage with low voltage by connecting it to Raspberry Pi.



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- Several Jumper wires
- 1 * Relay module
- 1 * Dual-color LED module
- 2 * 3-Pin anti-reverse cable

Experimental Principle

Relay

– There are 5 parts in every relay:

1. **Electromagnet** – It consists of an iron core wounded by coil of wires. When electricity is passed through, it becomes magnetic. Therefore, it is called electromagnet.

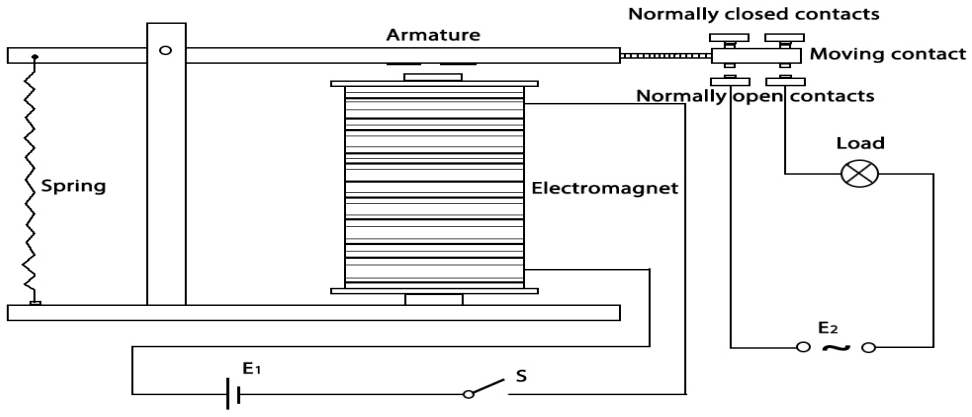
2. **Armature** – The movable magnetic strip is known as armature. When current flows through them, the coil is it energized thus producing a magnetic field which is used to make or break the normally open (N/O) or normally close (N/C) points. And the armature can be moved with direct current (DC) as well as alternating current (AC).

3. **Spring** – When no currents flow through the coil on the electromagnet, the spring pulls the armature away so the circuit cannot be completed.

4. Set of electrical **contacts** – There are two contact points:

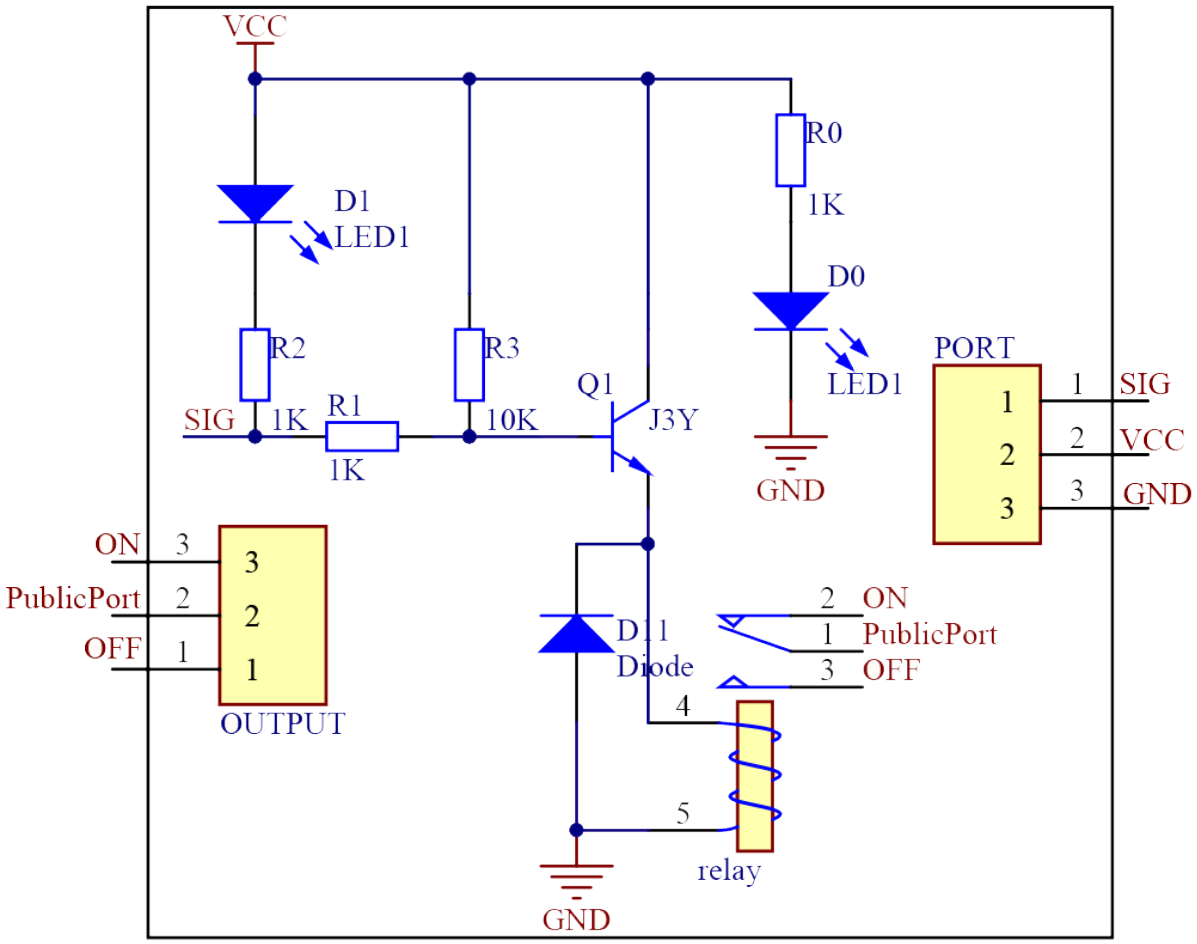
   - Normally open - connected when the relay is activated, and disconnected when it is inactive.
   - Normally close – not connected when the relay is activated, and connected when it is inactive.

5. Molded frame – Relays are covered with plastic for protection.



Connect the SIG pin of this module to GPIO pin. When we make GPIO pin output high level (3.3V) by programming, the transistor will conduct because of current saturation. The normally open contact of the relay will be closed, while the normally closed contact of the relay will be broken; when we make it output low level (0V), the transistor will be cut off, and the relay will recover to initial state.

The schematic diagram of the module is as shown below:

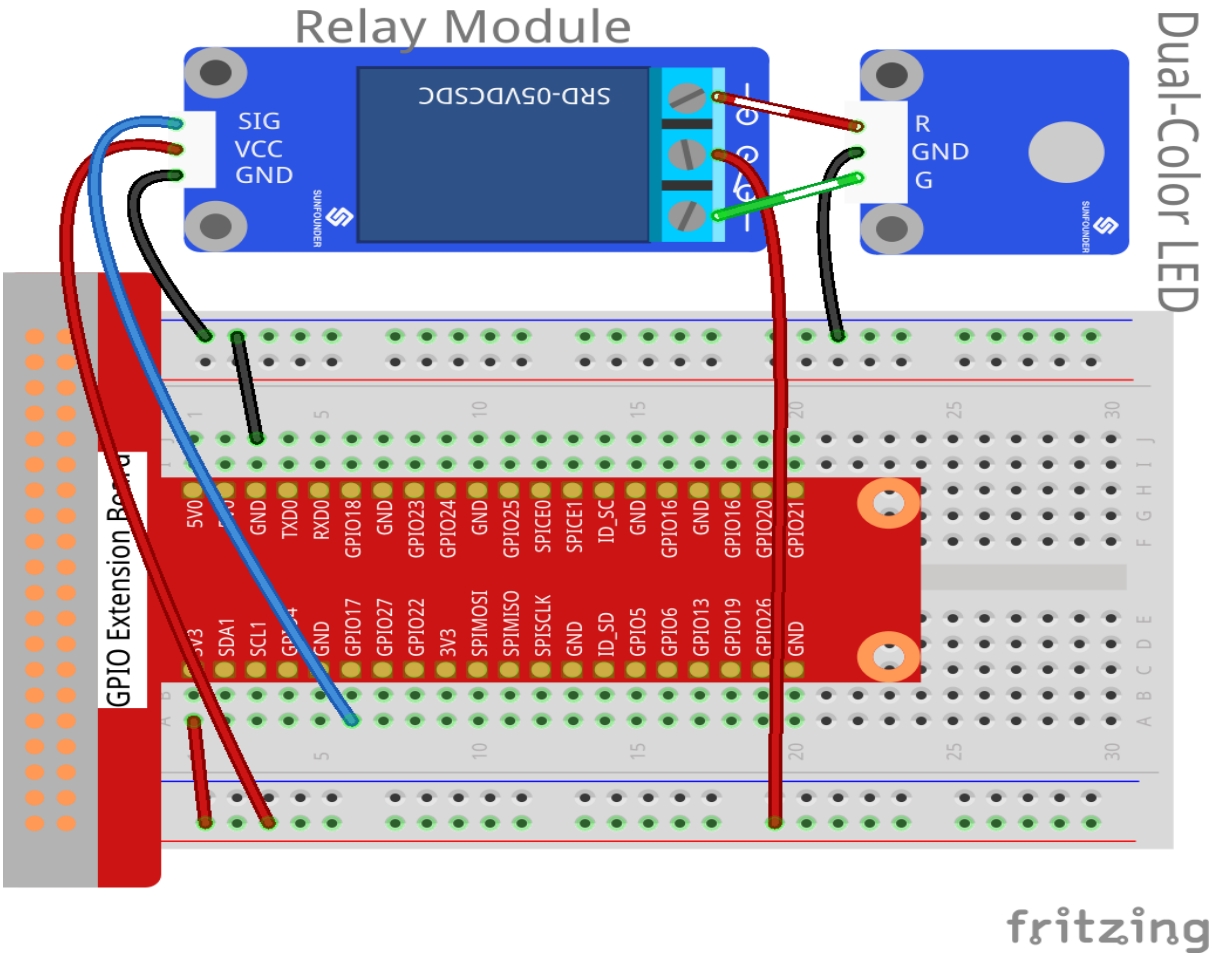


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Relay Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND
3.3V	3V3	COM

Dual-color LED Module	GPIO Extension Board	Relay Module
R	*	Normal Open
GND	GND	*
G	*	Normal Close



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/04_relay/
```

Step 3: Compile.

```
gcc relay.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define RelayPin      0

int main(void)
{
   if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
      printf("setup wiringPi failed !");
      return 1;
   }
//   printf("linker LedPin : GPIO %d(wiringPi pin)\n",VoicePin); //when initialize␣
↪wiring successfully,print message to screen

   pinMode(RelayPin, OUTPUT);

   while(1){
         digitalWrite(RelayPin, LOW);
         delay(1000);
         digitalWrite(RelayPin, HIGH);
         delay(1000);
   }

   return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 04_relay.py
```

Code

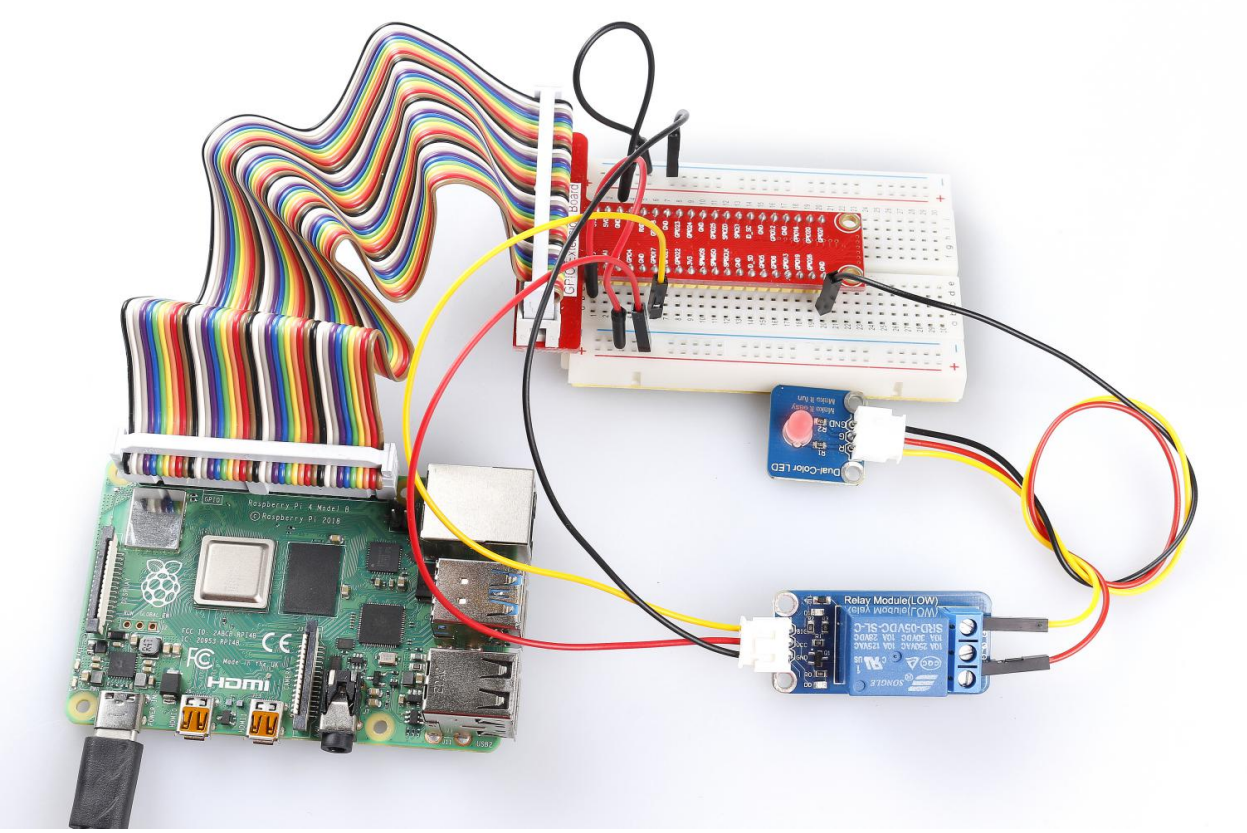
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

RelayPin = 11    # pin11

def setup():
   GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
   GPIO.setup(RelayPin, GPIO.OUT)
   GPIO.output(RelayPin, GPIO.HIGH)

def loop():
   while True:
      #'...relayd on'
      GPIO.output(RelayPin, GPIO.LOW)
      time.sleep(0.5)
      #'relay off...'
      GPIO.output(RelayPin, GPIO.HIGH)
      time.sleep(0.5)
```

(continues on next page)

(continued from previous page)

```
def destroy():
    GPIO.output(RelayPin, GPIO.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()
```

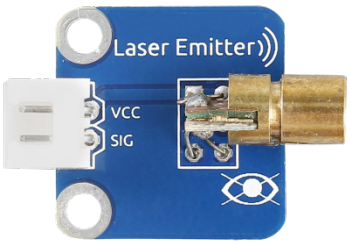
Now, you may hear the ticktock. That's the normally closed contact opened and the normally open contact closed. You can attach a high voltage device you want to control, like a 220V bulb, to the output port of the relay. Then the relay will act as an automatic switch.



6.5 Lesson 5 Laser Emitter Module

Introduction

Laser is widely used in medical treatment, military, and other fields due to its good directivity and energy concentration.



Required Components

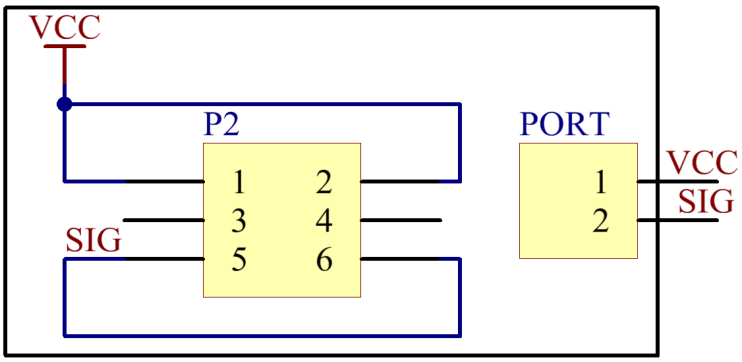
- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Laser Emitter module
- 1 * 2-Pin anti-reverse cable

Experimental Principle

A laser is a device that emits light through a process of optical amplification based on the stimulated emission of electromagnetic radiation. Lasers differ from other sources of light because they emit light coherently.

Spatial coherence allows a laser to be focused to a tight spot, enabling applications like laser cutting and lithography, and a laser beam to stay narrow over long distances (collimation), enabling applications like laser pointers. Lasers can also have high temporal coherence which allows them to have a very narrow spectrum, i.e., they only emit light of a single color. And its temporal coherence can be used to produce pulses of light—as short as a femtosecond.

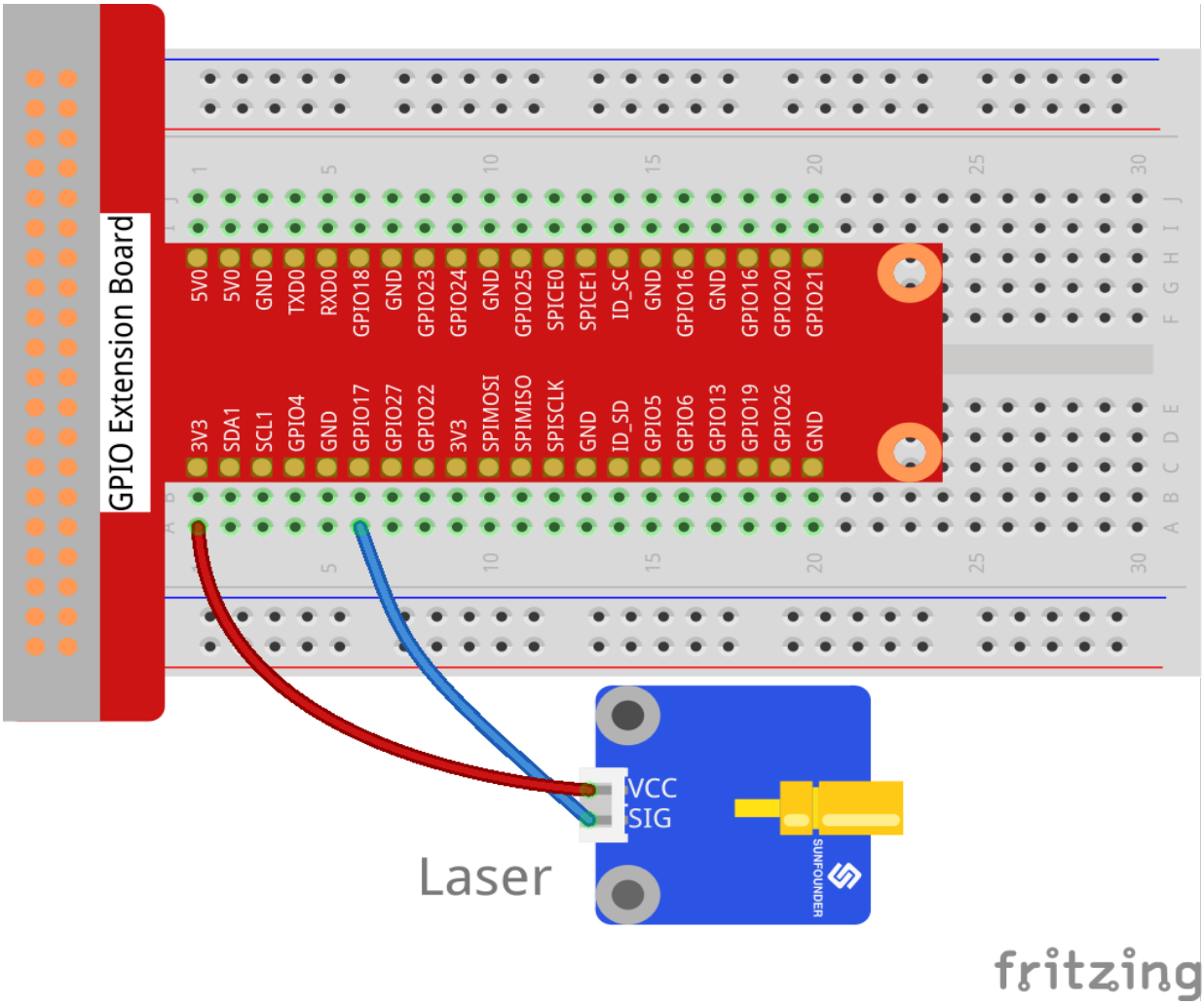
The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Laser Emitter Module
3.3V	3V3	VCC
GPIO0	GPIO17	SIG



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/05_laser/
```

Step 3: Compile.

```
gcc laser.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

(continues on next page)

(continued from previous page)

```
#define LaserPin    0

int main(void)
{
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
        printf("setup wiringPi failed !");
        return 1;
    }
    //printf("linker LedPin : GPIO %d(wiringPi pin)\n",LedPin); //when initialize␣
↪wiring successfully,print message to screen

    pinMode(LaserPin, OUTPUT);

    while(1){
        digitalWrite(LaserPin, HIGH);
        delay(500);
        digitalWrite(LaserPin, LOW);
        delay(500);
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 05_laser.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

LedPin = 11    # pin11

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(LedPin, GPIO.OUT)    # Set LedPin's mode is output
    GPIO.output(LedPin, GPIO.HIGH) # Set LedPin high(+3.3V) to off led

def loop():
    while True:
        #'...Laser on'
        GPIO.output(LedPin, GPIO.LOW)  # led on
        time.sleep(0.5)
        #'Laser off...'
        GPIO.output(LedPin, GPIO.HIGH) # led off
        time.sleep(0.5)

def destroy():
```

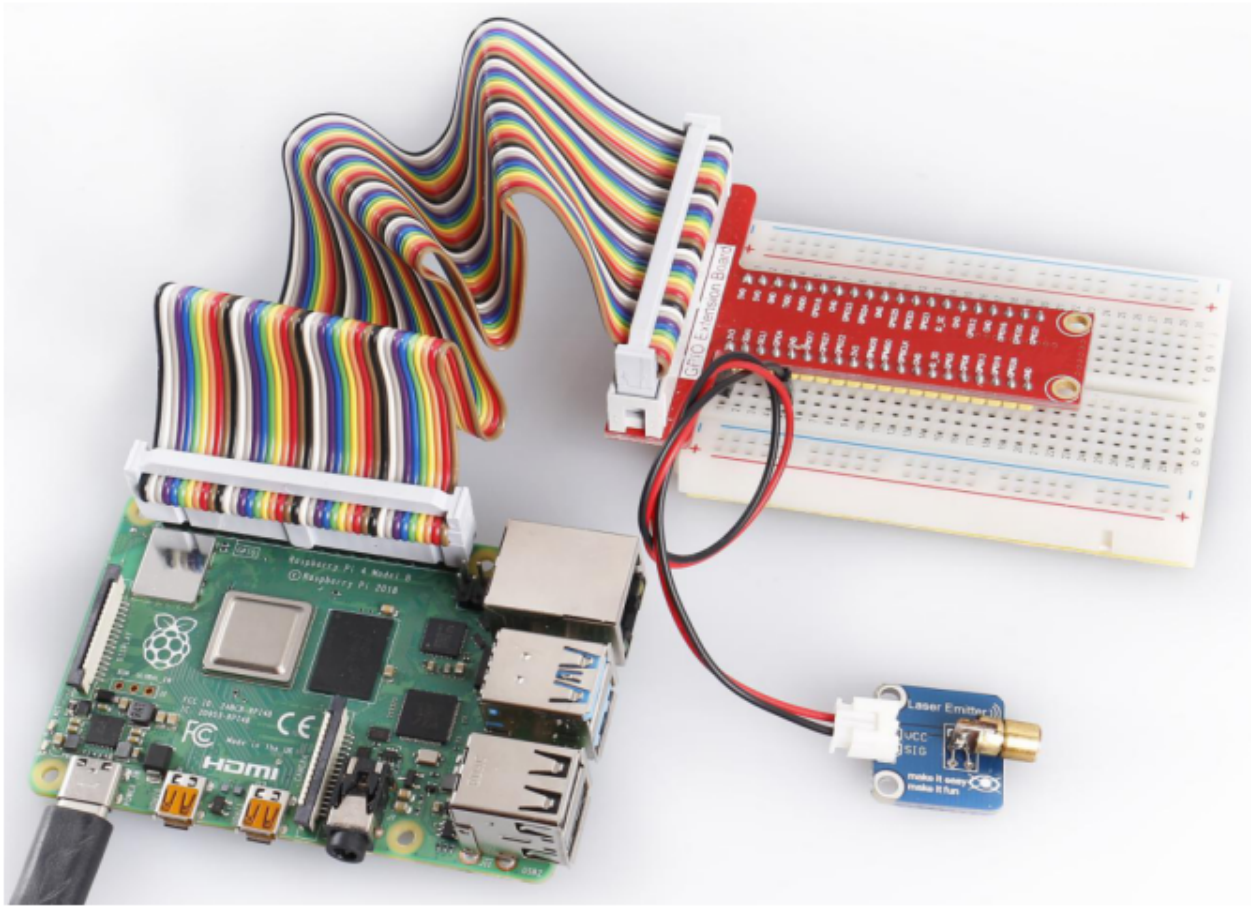
(continues on next page)

(continued from previous page)

```
    GPIO.output(LedPin, GPIO.HIGH)      # led off
    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()
```

Now you can see the module send out Morse signals.

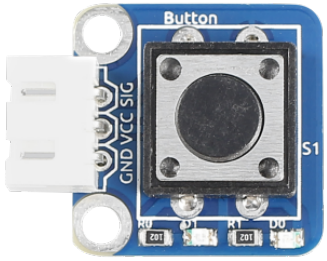
Note: DO NOT look directly at the laser head. It can cause great harm to your eyes. You can point the laser beam to the table and see the light spot flashing on the table.



6.6 Lesson 6 Button Module

Introduction

In this lesson, we will use button module to control a dual-color LED module.



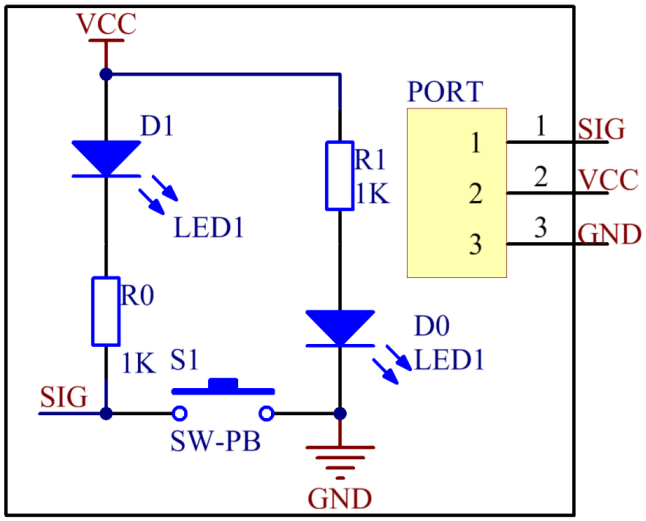
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- Several Jumper wires
- 1 * Button module
- 1 * Dual-color LED module
- 2 * 3-Pin anti-reverse cable

Experimental Principle

Use a normally open button as an input device of Raspberry Pi. When the button is pressed, the General Purpose Input/Output (GPIO) connected to the button will change to low level (0V). You can detect the state of the GPIO through programming. That is, if the GPIO turns into low level, it means the button is pressed, so you can run the corresponding code. In this experiment, we will print a string on the screen and control an LED.

The schematic diagram of the module is as shown below:

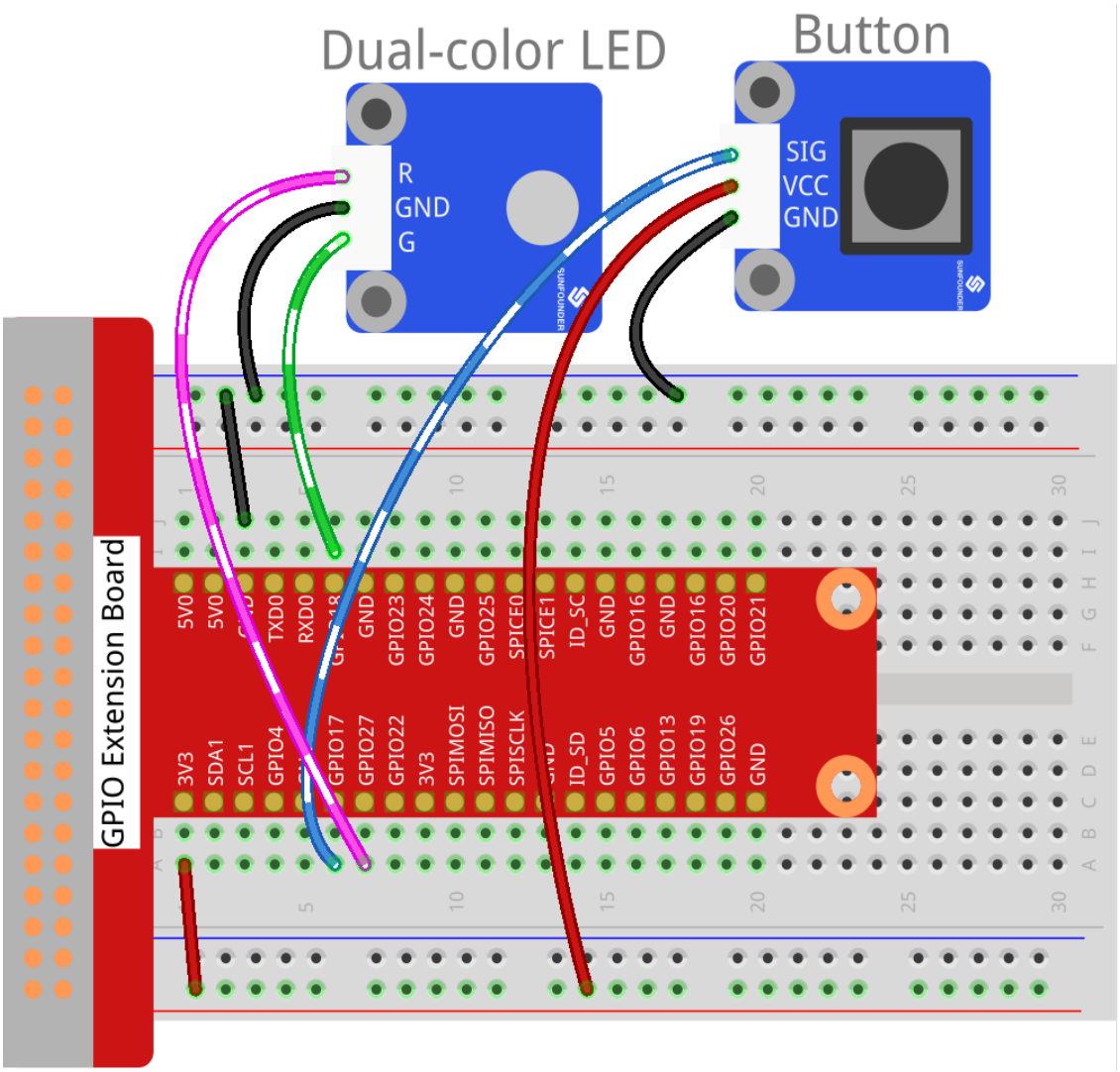


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Button Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-Color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/06_button/
```

Step 3: Compile.

```
gcc button.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define BtnPin              0
#define Gpin                1
#define Rpin                2

void LED(char* color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == "RED")
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == "GREEN")
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

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

    pinMode(BtnPin, INPUT);
    LED("GREEN");

    while(1){
        if(0 == digitalRead(BtnPin)){
            delay(10);
            if(0 == digitalRead(BtnPin)){
                LED("RED");
                printf("Button is pressed\n");
            }
        }
        else if(1 == digitalRead(BtnPin)){
```

(continues on next page)

(continued from previous page)

```
            delay(10);
            if(1 == digitalRead(BtnPin)){
                while(!digitalRead(BtnPin));
                LED("GREEN");
            }
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 06_button.py
```

Code

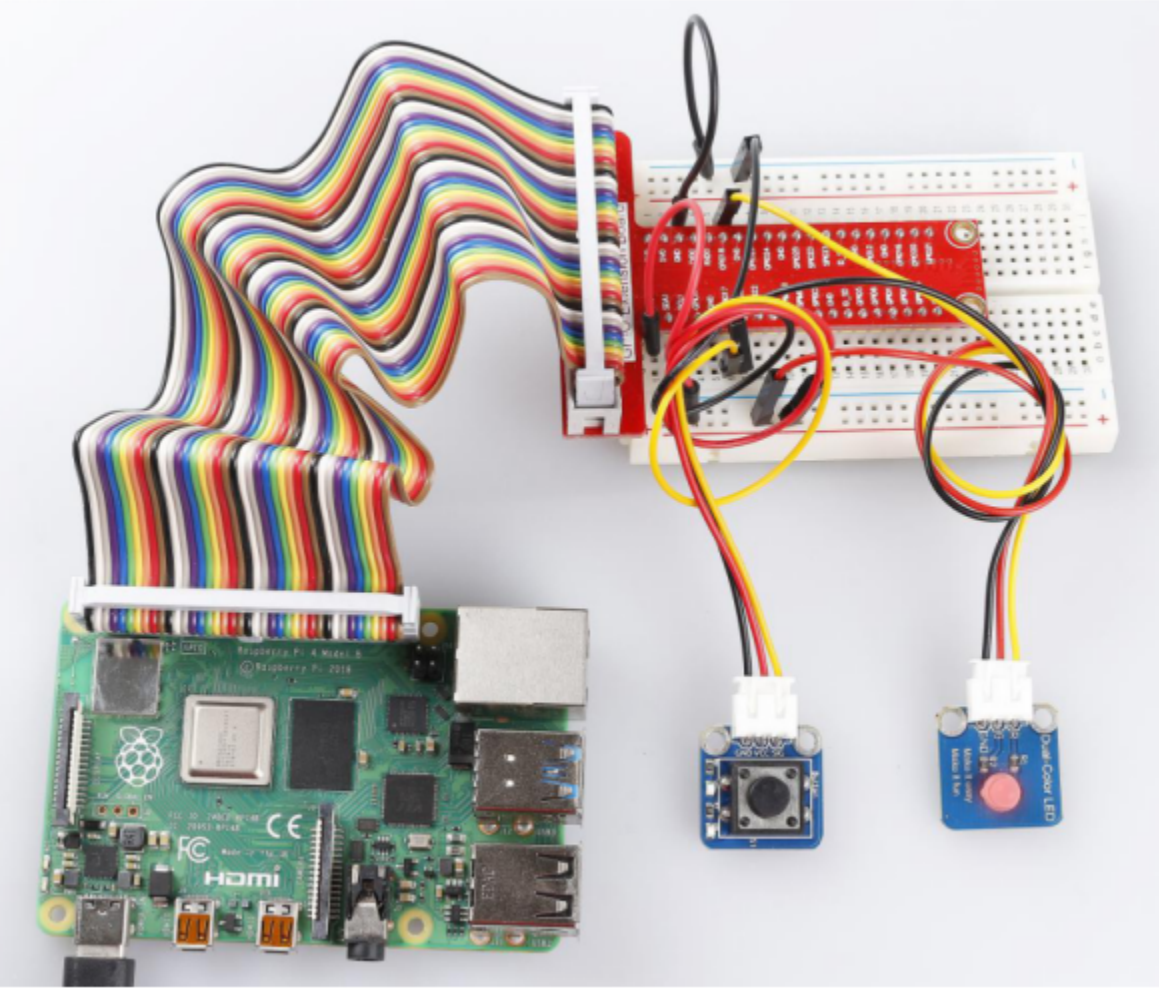
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

BtnPin = 11
Gpin   = 12
Rpin   = 13

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)      # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)      # Set Red Led Pin mode to output
    GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode is␣
↪input, and pull up to high level(3.3V)
    GPIO.add_event_detect(BtnPin, GPIO.BOTH, callback=detect, bouncetime=200)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)

def detect(chn):
    Led(GPIO.input(BtnPin))

def loop():
    while True:
        pass

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    GPIO.cleanup()                     # Release resource
```

(continues on next page)

(continued from previous page)

```
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()
```

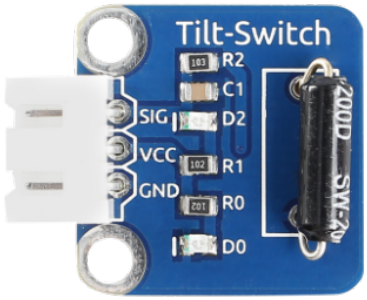
The LED on the module will emit green light. If you press the button, “Button pressed” will be printed on the screen and the LED will emit red light. If you release the button, “Button released” will be printed on the screen and the LED will flash green again.



6.7 Lesson 7 Tilt-Switch Module

Introduction

The tilt-switch module (as shown below) in this kit is a ball tilt-switch with a metal ball inside. It is used to detect inclinations of a small angle.



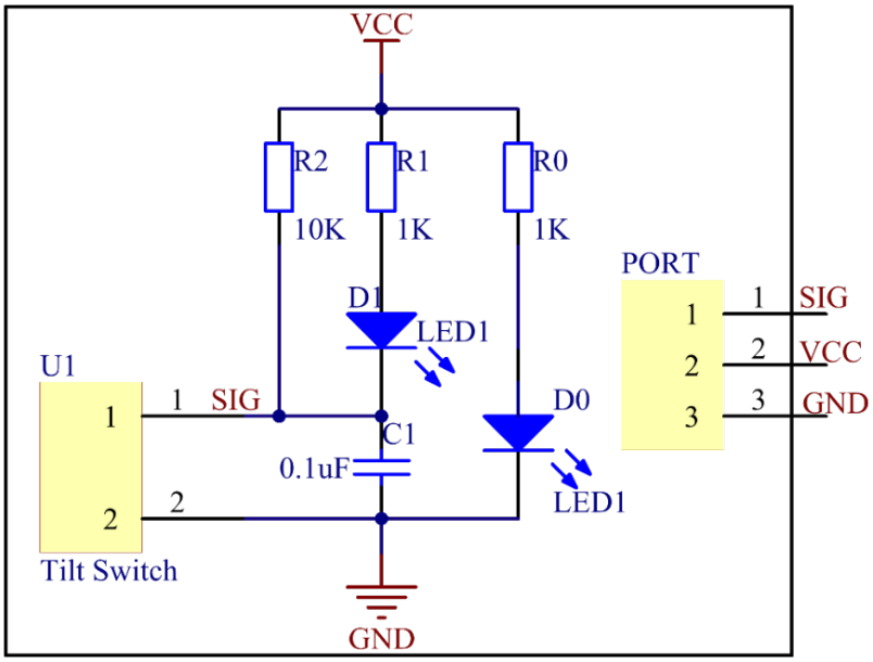
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Dual-color LED module
- 1 * Tilt-switch module
- 2 * 3-Pin anti-reverse cable

Experimental Principle

The principle is very simple. The ball in the tilt-switch changes with different angle of inclination to trigger the circuit. When the ball in tilt switch runs from one end to the other end due to shaking caused by external force, the tilt switch will conduct and the LED will emit red light, otherwise it will break and the LED will emit green light.

The schematic diagram of the module is as shown below:

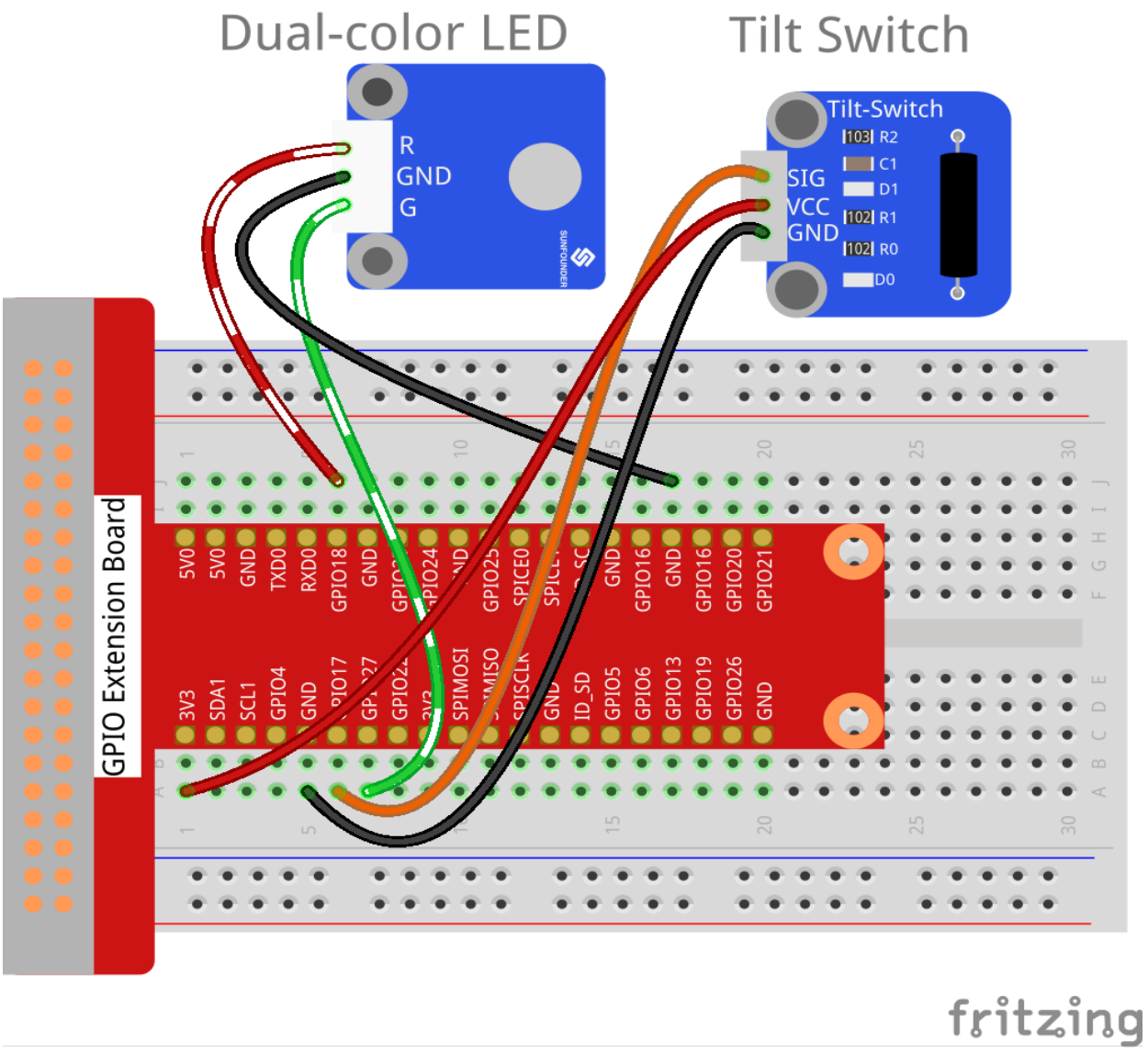


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Tilt Switch Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-Color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/07_tilt_switch/
```

Step 3: Compile.

```
gcc tilt_switch.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define TiltPin             0
#define Gpin                2
#define Rpin                1

void LED(char* color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == "RED")
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == "GREEN")
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

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

    pinMode(TiltPin, INPUT);
    LED("GREEN");

    while(1){
        if(0 == digitalRead(TiltPin)){
            delay(10);
            if(0 == digitalRead(TiltPin)){
                LED("RED");
                printf("Tilt!\n");
            }
        }
        else if(1 == digitalRead(TiltPin)){
            delay(10);
            if(1 == digitalRead(TiltPin)){
                while(!digitalRead(TiltPin));
                LED("GREEN");
            }
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

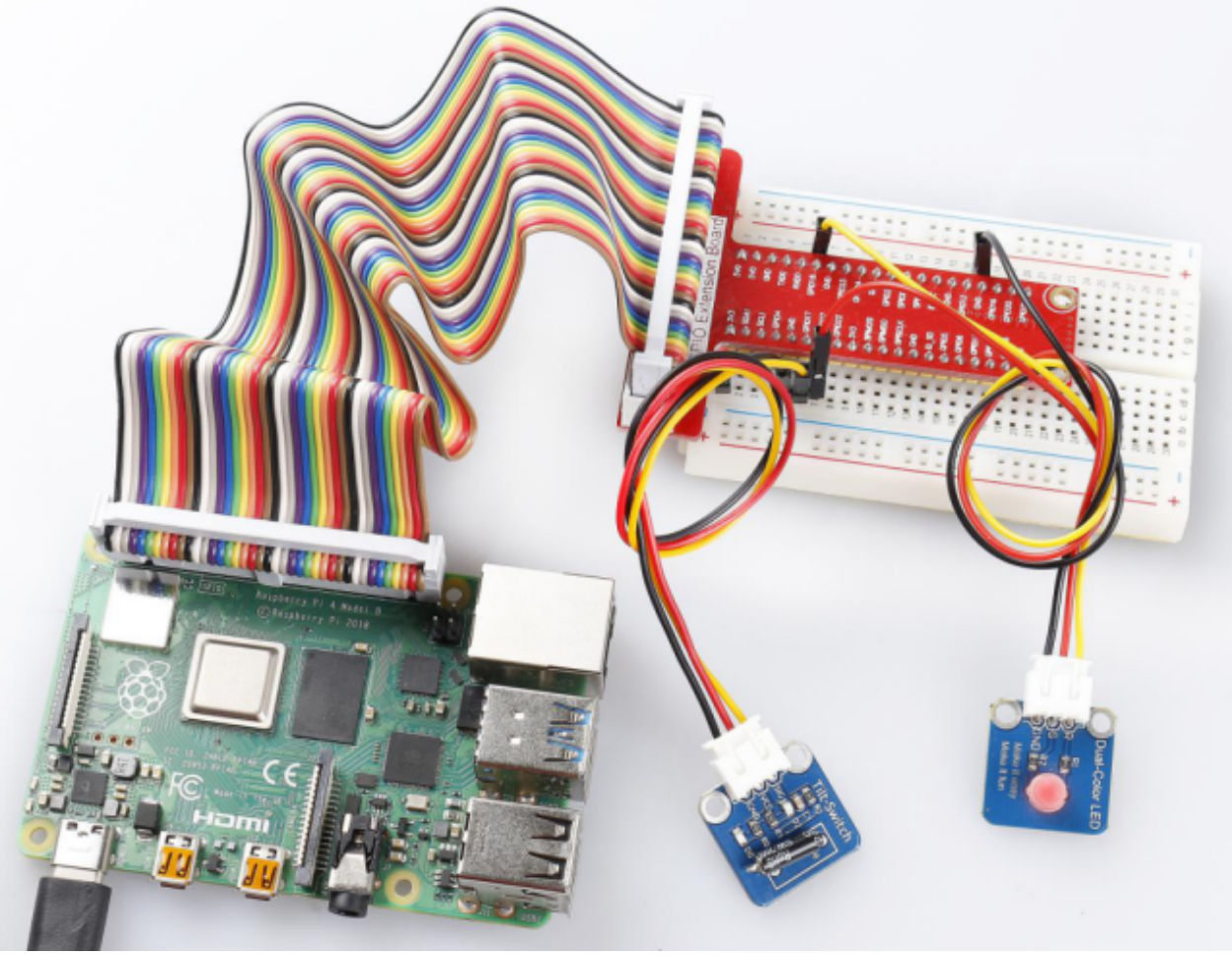
Step 3: Run.

```
sudo python3 07_tilt_switch.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

TiltPin = 11
Gpin   = 13
Rpin   = 12

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)      # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)      # Set Red Led Pin mode to output
    GPIO.setup(TiltPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode is␣
↪input, and pull up to high level(3.3V)
    GPIO.add_event_detect(TiltPin, GPIO.BOTH, callback=detect, bouncetime=200)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)

def detect(chn):
    Led(GPIO.input(TiltPin))

def loop():
    while True:
        pass

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    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()
```

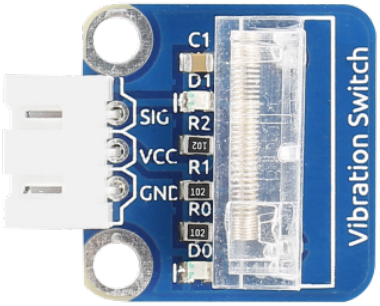
Place the tilt switch module horizontally, and the LED will flash green. If you tilt it, “Tilt!” will be printed on the screen and the LED will change to red. Place it horizontally again, and the LED will flash green again.



6.8 Lesson 8 Vibration Switch

Introduction

A vibration switch, also called spring switch or shock sensor, is an electronic switch which induces shock force and transfers the result to a circuit device thus triggering it to work. It contains the following parts: conductive vibration spring, switch body, trigger pin, and packaging agent.



Required Components

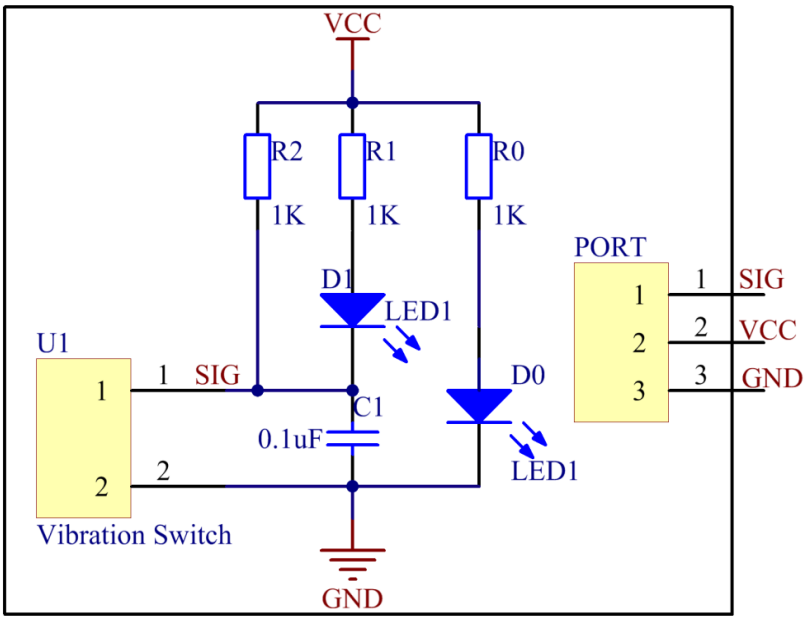
- 1 * Raspberry Pi

- 1 * Breadboard
- 1 * Dual-color LED module
- 1 * Vibration switch module
- 2 * 3-Pin anti-reverse cable

Experimental Principle

In a vibration switch module, the conductive vibration spring and trigger pin are precisely placed in the switch and fixed by adhesive. Normally, the spring and the trigger pin are separated. Once the sensor detects shock, the spring will vibrate and contact with the trigger pin, thus conducting and generating trigger signals.

In this experiment, connect a dual-color LED module to the Raspberry Pi to indicate the changes. When you knock or tap the vibration sensor, it will get turned on and the dual-color LED will flash red. Tap it again and the LED will change to green – just between the two colors for each tap or knock. The schematic diagram:

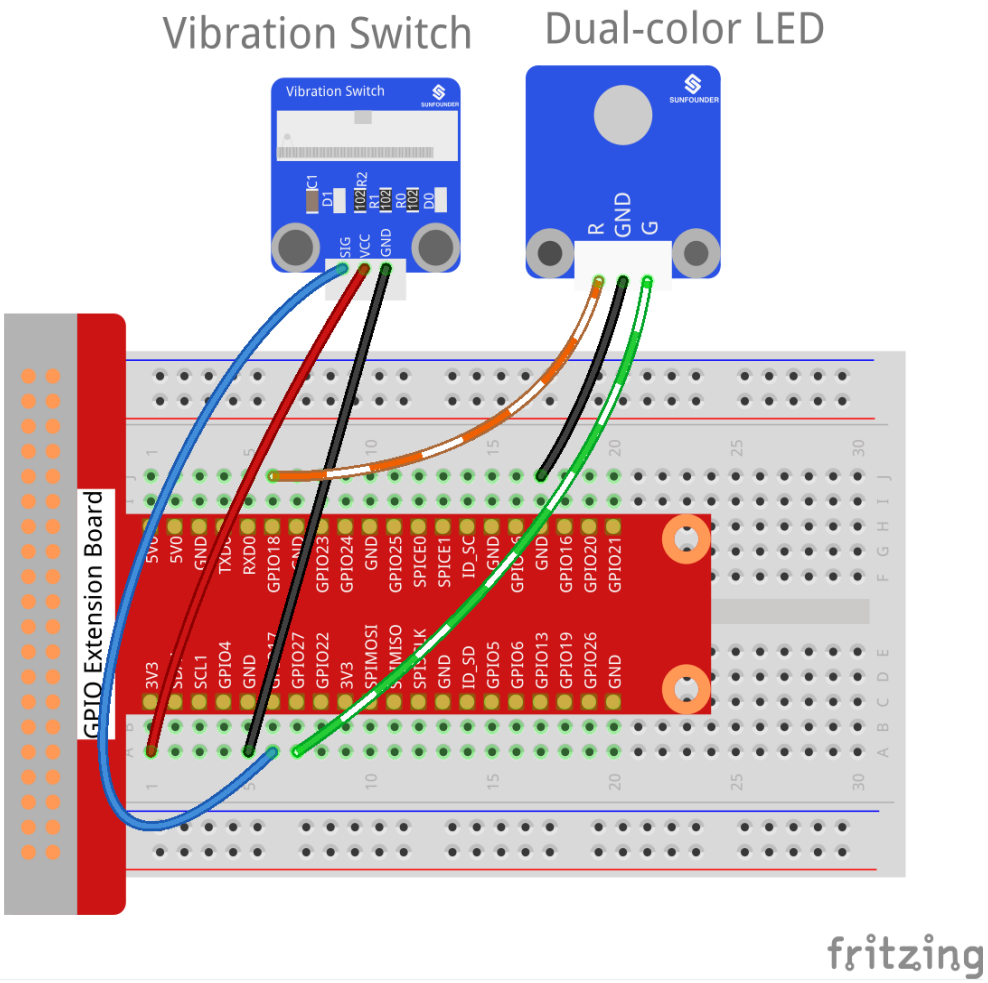


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Vibration Switch Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-Color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/08_vibration_switch/
```

Step 3: Compile.

```
gcc vibration_switch.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

(continues on next page)

(continued from previous page)

```
#define VibratePin  0
#define Gpin                2
#define Rpin                1

void LED(int color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == 0)
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == 1)
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

int main(void)
{
    int status = 0;
    int tmp = 1;
    int value = 1;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(VibratePin, INPUT);

    while(1){
        value = digitalRead(VibratePin);
        if (tmp != value){
            status ++;
            if (status > 1){
                status = 0;
            }
            LED(status);
            delay(1000);
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

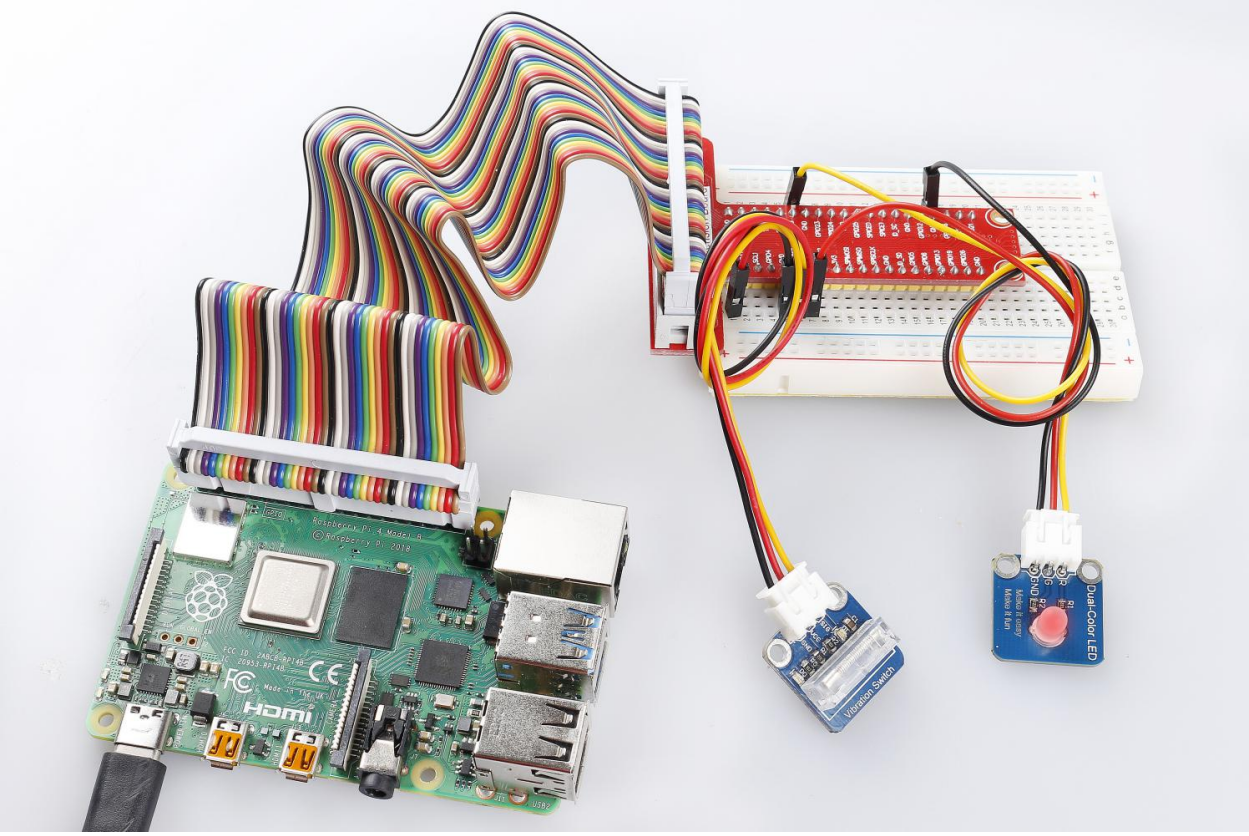
Step 3: Run.

```
sudo python3 08_vibration_switch.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

VibratePin = 11
Gpin   = 13
Rpin   = 12

tmp = 0

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)      # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)      # Set Red Led Pin mode to output
    GPIO.setup(VibratePin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode␣
↪is input, and pull up to high level(3.3V)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)

def loop():
    state = 0
    while True:
        if GPIO.input(VibratePin)==0:
            state = state + 1
            if state > 1:
                state = 0
            Led(state)
            time.sleep(1)

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    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()
```

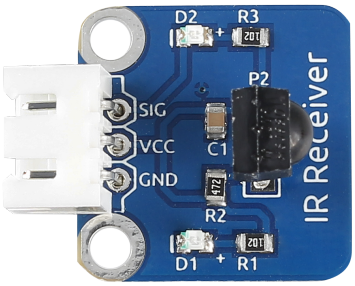
Now tap or knock the module and you can see the dual-color LED flash red. Tap the sensor again, and the LED will change to green. Each tap or knock would make it change between red and green.



6.9 Lesson 9 IR Receiver Module

Introduction

An infrared-receiver (as shown below) is a component which receives infrared signals and can independently receive infrared rays and output signals compatible with TTL level. It is similar with a normal plastic-packaged transistor in size and is suitable for all kinds of infrared remote control and infrared transmission.



Required Components

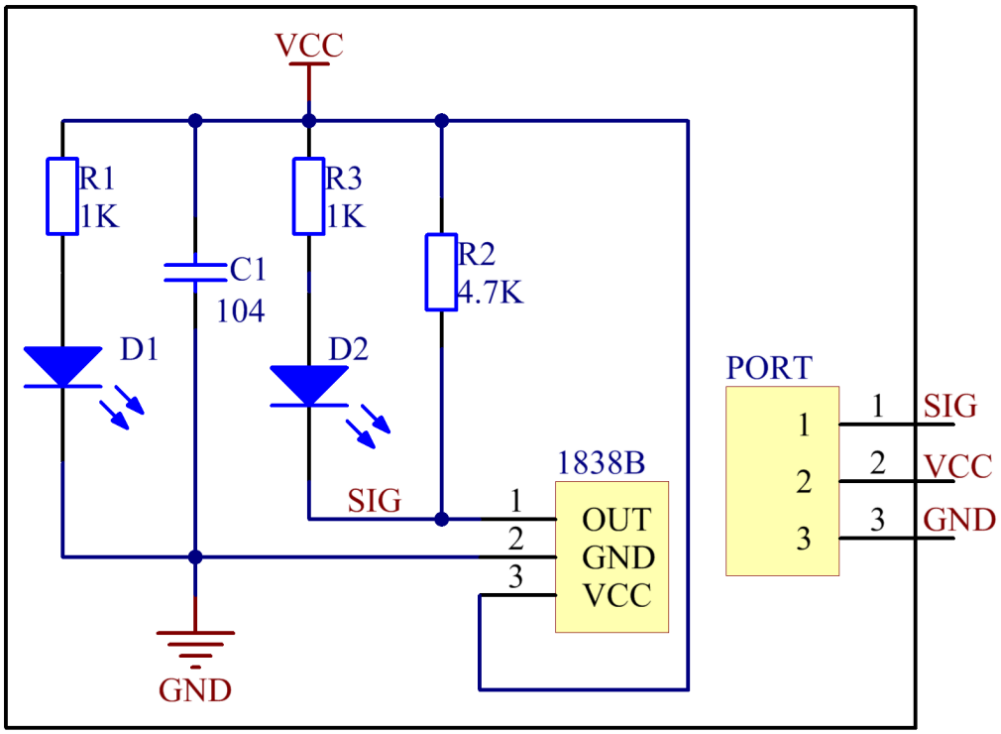
- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * IR receiver module

- 1 * IR Remote Controller
- 1 * 3-Pin anti-reverse cable

Experimental Principle

In this experiment, send signals to IR receiver by pressing buttons on the IR remote controller. The counter will add 1 every time it receives signals; in other words, the increased number indicates IR signals are received.

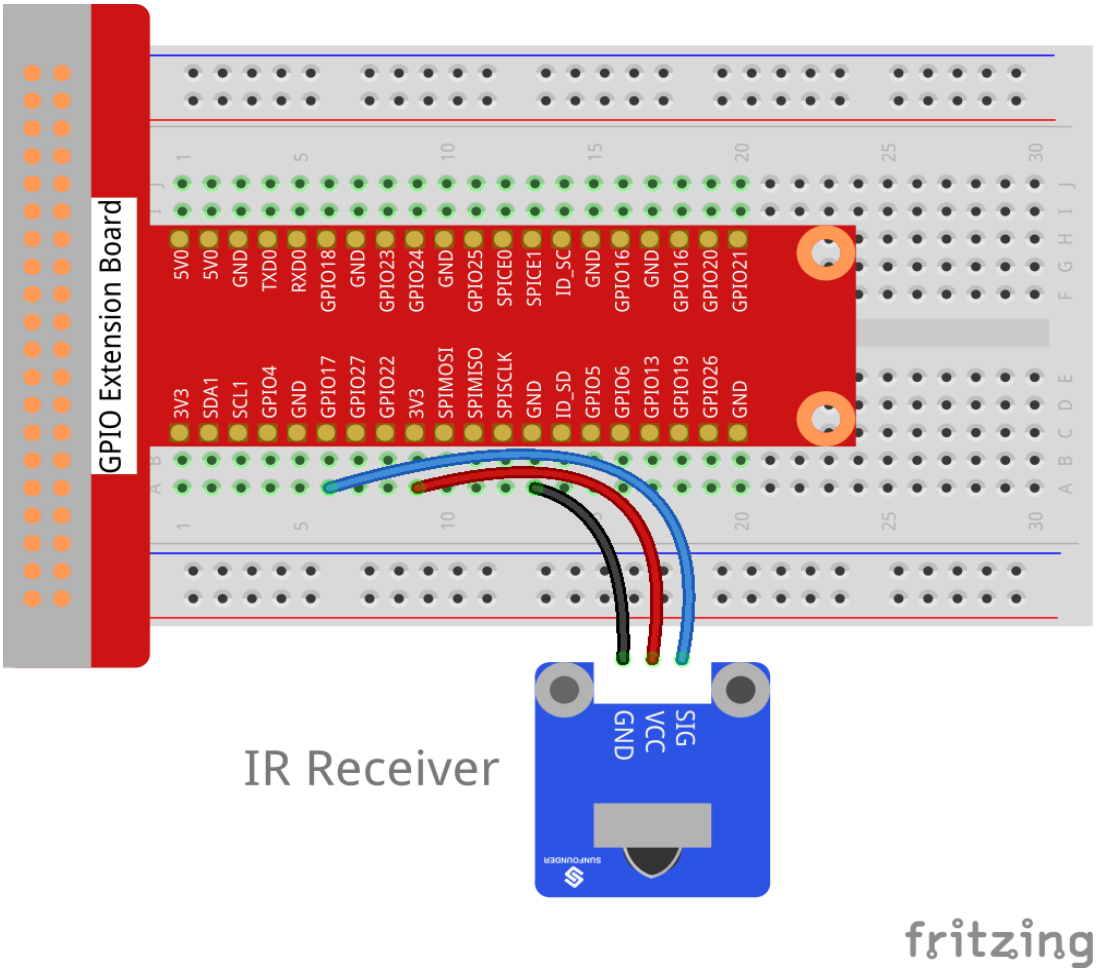
The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	IR Receiver Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/09_ir_receiver/
```

Step 3: Compile.

```
gcc ir_receiver.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define     IR      0
```

(continues on next page)

(continued from previous page)

```
int cnt = 0;

void myISR(void)
{
    printf("Received infrared. cnt = %d\n", ++cnt);
}

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

    if(wiringPiISR(IR, INT_EDGE_FALLING, &myISR) == -1){
        printf("setup ISR failed !");
        return 1;
    }

    //pinMode(IR, INPUT);

    while(1);

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 09_ir_receiver.py
```

Code

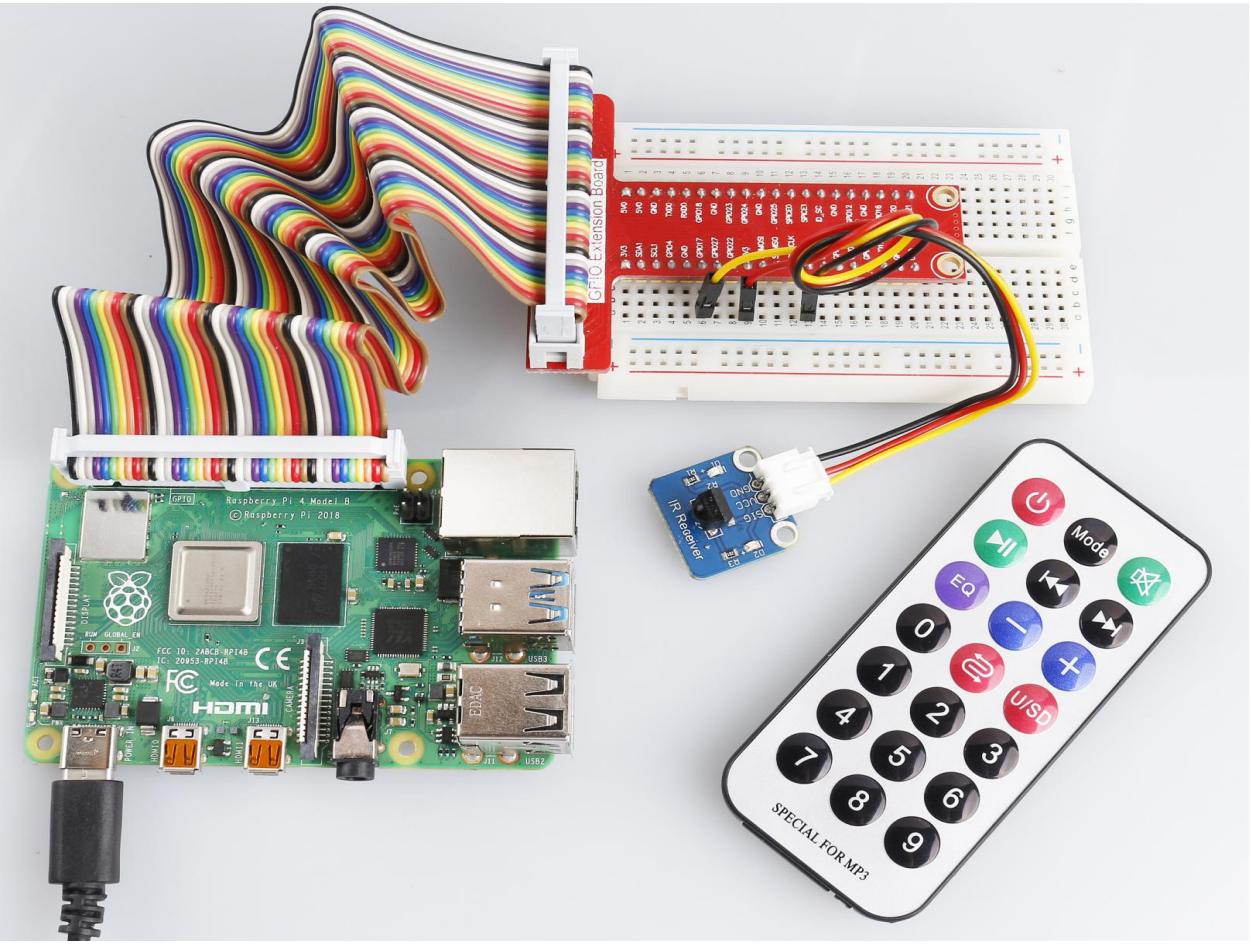
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

IrPin  = 11
count = 0

def setup():
    GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
    GPIO.setup(IrPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def cnt(ev=None):
    global count
    count += 1
    print ('Received infrared. cnt = ', count)

def loop():
    GPIO.add_event_detect(IrPin, GPIO.FALLING, callback=cnt) # wait for falling
    while True:
```

(continues on next page)

(continued from previous page)

```
        pass    # Don't do anything

def destroy():
    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()
```

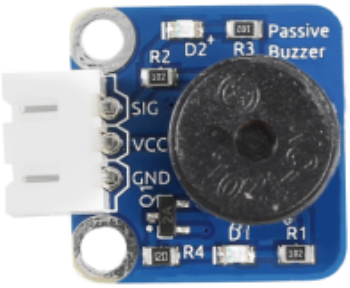
Press any key of the remote. Then you can see the LED on the module blinking, and "Received infrared. cnt = xxx" printed on the screen. "xxx" means the time you pressed the key(s).



6.10 Lesson 10 Buzzer Module

Introduction

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



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Passive buzzer module
- 1 * Active buzzer module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

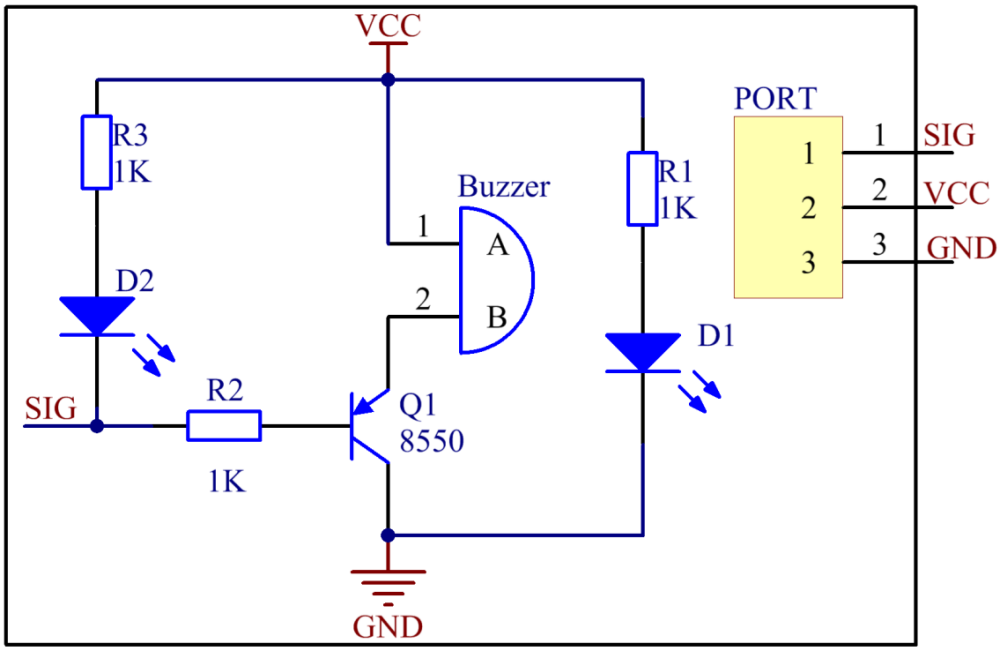
Place the pins of two buzzers face up and you can see the one with a green circuit board is a passive buzzer, while the other with a black tape, instead of a board, is an active buzzer.

Active buzzer Passive buzzer

The difference between an active buzzer and a passive buzzer is:

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 beep 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.

The schematic diagram of the module is as shown below:

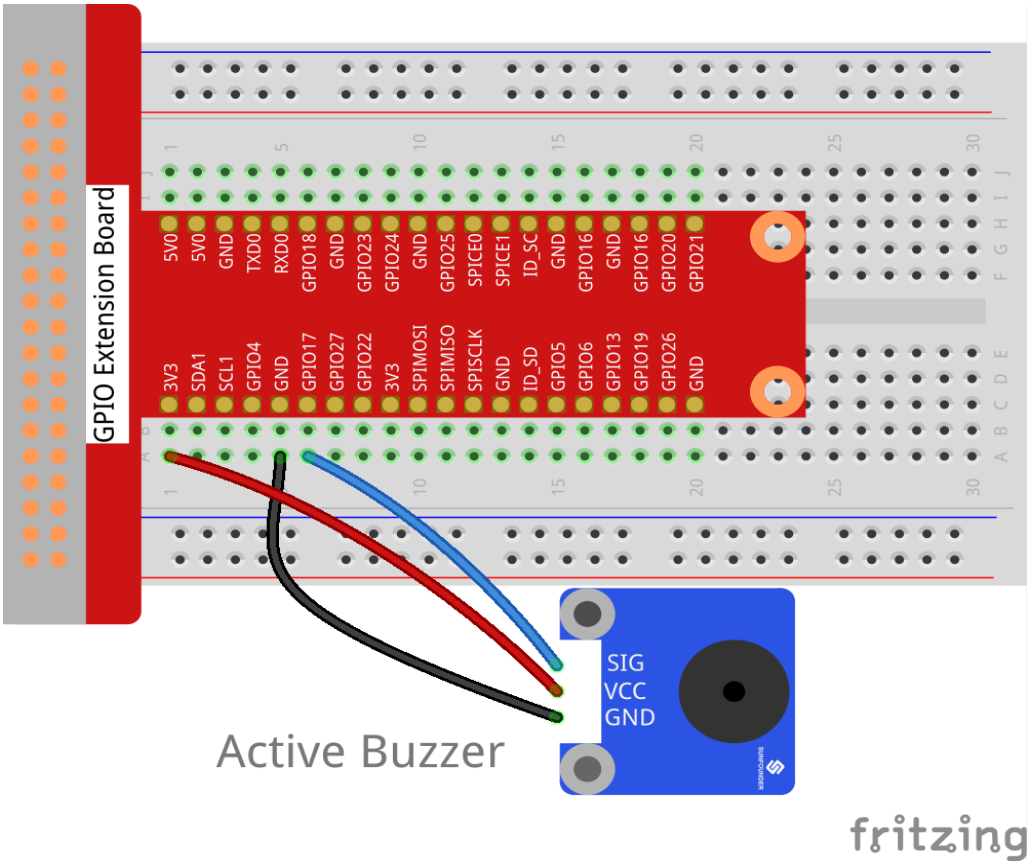


Experimental Procedures

Active Buzzer

Note: The active buzzer has built-in oscillating source, so it will beep as long as it is wired up, but it can only beep with fixed frequency.

Step 1: Build the circuit.



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/10_active_buzzer/
```

Step 3: Compile.

```
gcc active_buzzer.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define BuzzerPin       0

int main(void)
{
```

(continues on next page)

(continued from previous page)

```
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
        printf("setup wiringPi failed !");
        return 1;
    }
//  printf("linker LedPin : GPIO %d(wiringPi pin)\n",VoicePin); //when initialize␣
↪wiring successfully,print message to screen

    pinMode(BuzzerPin,  OUTPUT);

    while(1){
            digitalWrite(BuzzerPin, HIGH);
            delay(100);
            digitalWrite(BuzzerPin, LOW);
            delay(100);
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 10_active_buzzer.py
```

Code

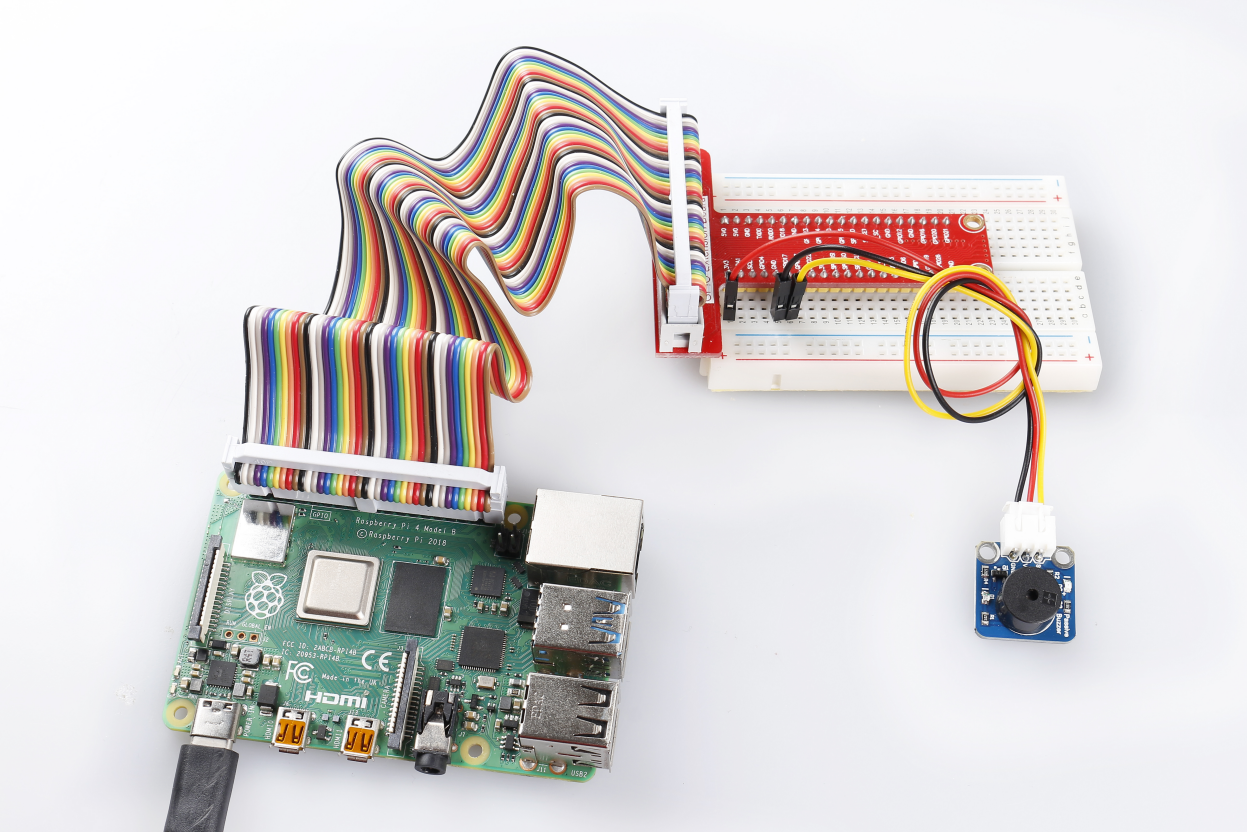
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

Buzzer = 11    # pin11

def setup(pin):
    global BuzzerPin
    BuzzerPin = pin
    GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
    GPIO.setup(BuzzerPin, GPIO.OUT)
    GPIO.output(BuzzerPin, GPIO.HIGH)

def on():
    GPIO.output(BuzzerPin, GPIO.LOW)

def off():
    GPIO.output(BuzzerPin, GPIO.HIGH)

def beep(x):
    on()
    time.sleep(x)
    off()
    time.sleep(x)

def loop():
```

(continues on next page)

(continued from previous page)

```
    while True:
        beep(0.5)

def destroy():
    GPIO.output(BuzzerPin, GPIO.HIGH)
    GPIO.cleanup()                     # Release resource

if __name__ == '__main__':     # Program start from here
    setup(Buzzer)
    try:
        loop()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program␣
↪destroy() will be  executed.
        destroy()
```

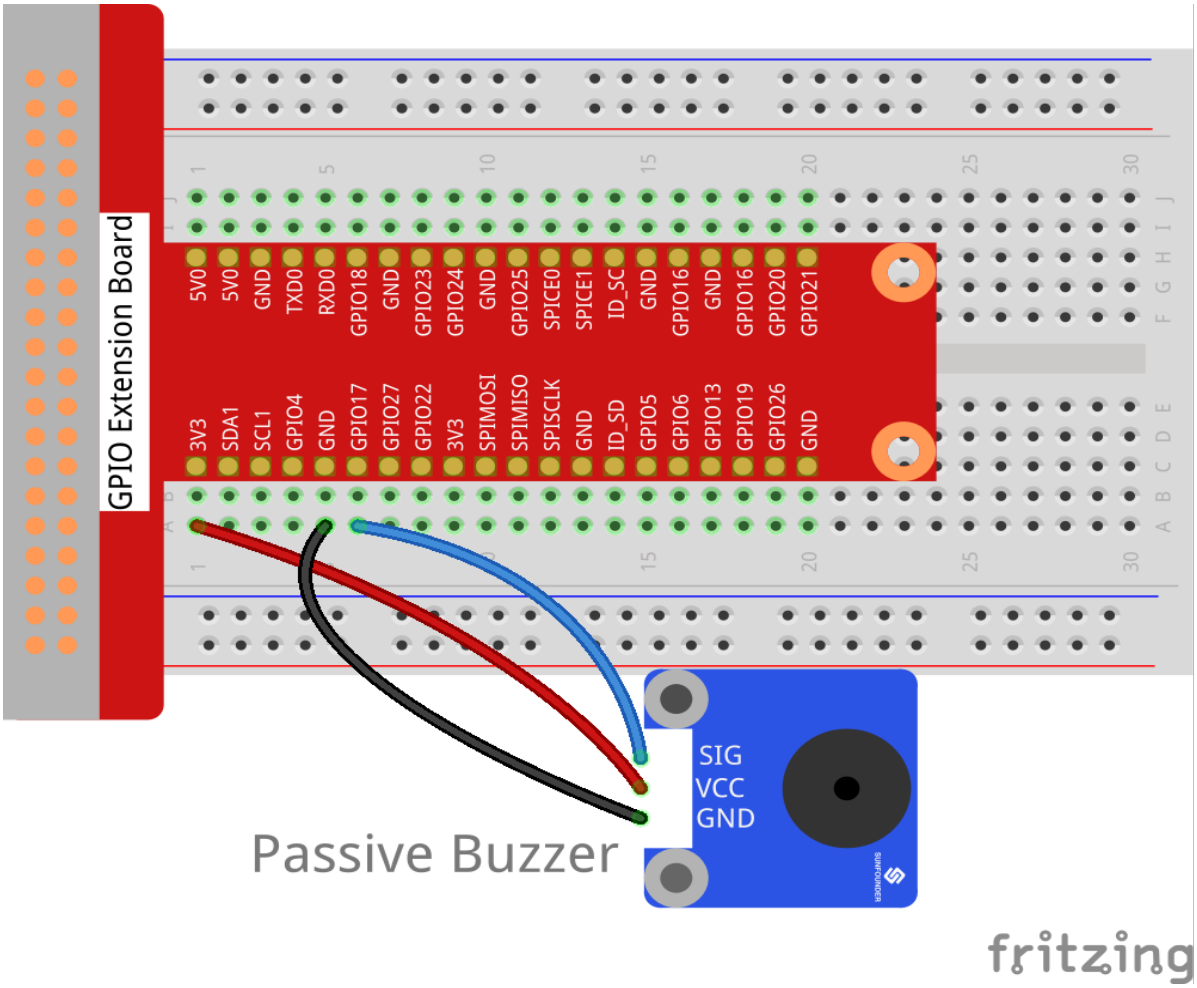
Now you can hear the active buzzer beeping.



Passive Buzzer

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Passive Buzzer Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/10_passive_buzzer/
```

Step 3: Compile.

```
gcc passive_buzzer.c -lwiringPi
```

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define BuzPin    0

#define  CL1  131
#define  CL2  147
```

(continues on next page)

(continued from previous page)

```
#define  CL3  165
#define  CL4  175
#define  CL5  196
#define  CL6  221
#define  CL7  248

#define  CM1  262
#define  CM2  294
#define  CM3  330
#define  CM4  350
#define  CM5  393
#define  CM6  441
#define  CM7  495

#define  CH1  525
#define  CH2  589
#define  CH3  661
#define  CH4  700
#define  CH5  786
#define  CH6  882
#define  CH7  990

int song_1[] = {CM3,CM5,CM6,CM3,CM2,CM3,CM5,CM6,CH1,CM6,CM5,CM1,CM3,CM2,
                CM2,CM3,CM5,CM2,CM3,CM3,CL6,CL6,CL6,CM1,CM2,CM3,CM2,CL7,
                CL6,CM1,CL5};

int beat_1[] = {1,1,3,1,1,3,1,1,1,1,1,1,1,3,1,1,3,1,1,1,1,1,1,2,1,1,
                1,1,1,1,1,1,3};

int song_2[] = {CM1,CM1,CM1,CL5,CM3,CM3,CM3,CM1,CM1,CM3,CM5,CM5,CM4,CM3,CM2,
                CM2,CM3,CM4,CM4,CM3,CM2,CM3,CM1,CM1,CM3,CM2,CL5,CL7,CM2,CM1
                };

int beat_2[] = {1,1,1,3,1,1,1,3,1,1,1,1,1,1,3,1,1,1,2,1,1,1,3,1,1,1,3,3,2,3};

int main(void)
{
    int i, j;

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

    if(softToneCreate(BuzPin) == -1){
        printf("setup softTone failed !");
        return 1;
    }

    while(1){
        printf("music is being played...\n");

        for(i=0;i<sizeof(song_1)/4;i++){
            softToneWrite(BuzPin, song_1[i]);
            delay(beat_1[i] * 500);
        }
```

(continues on next page)

(continued from previous page)

```
        for(i=0;i<sizeof(song_2)/4;i++){
            softToneWrite(BuzPin, song_2[i]);
            delay(beat_2[i] * 500);
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 10_passive_buzzer.py
```

Code

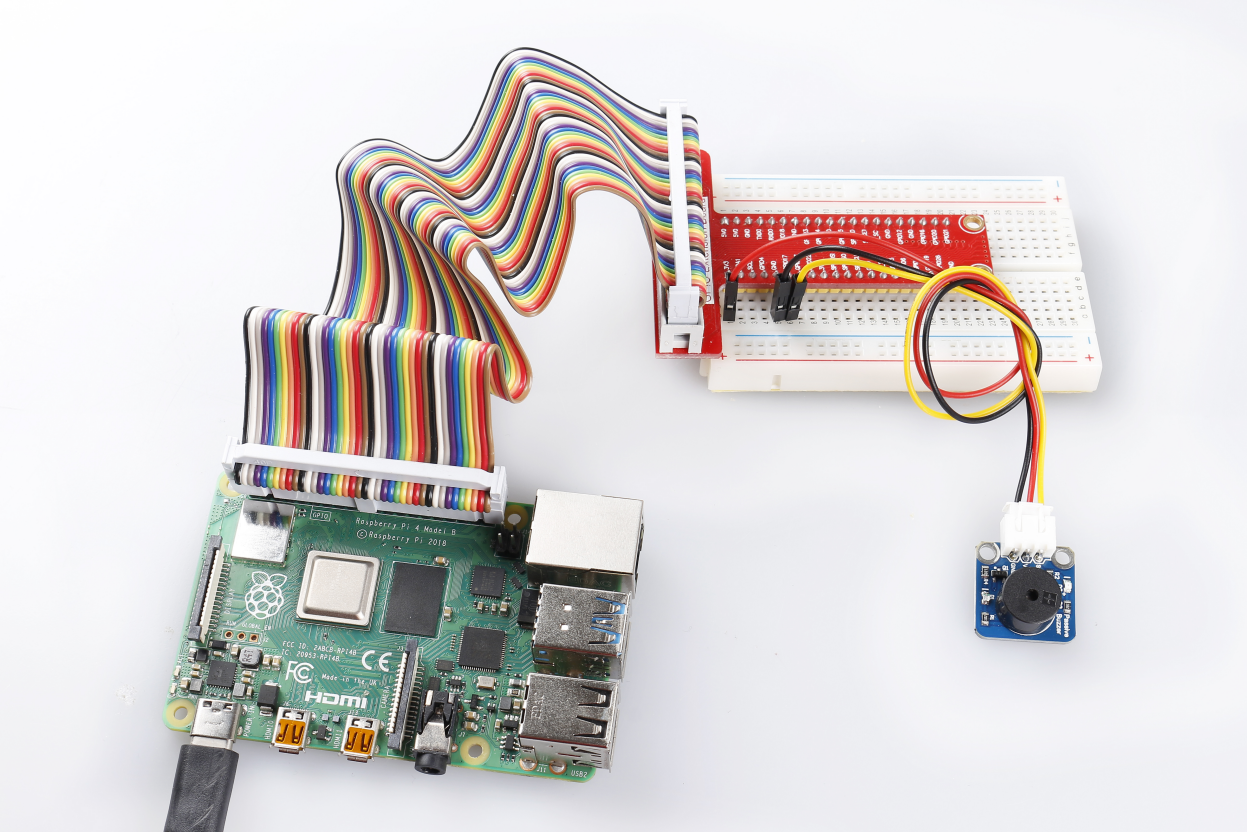
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

Buzzer = 11

CL = [0, 131, 147, 165, 175, 196, 211, 248]          # Frequency of Low C notes

CM = [0, 262, 294, 330, 350, 393, 441, 495]          # Frequency of Middle C notes

CH = [0, 525, 589, 661, 700, 786, 882, 990]          # Frequency of High C notes

song_1 = [  CM[3], CM[5], CM[6], CM[3], CM[2], CM[3], CM[5], CM[6], # Notes of song1
            CH[1], CM[6], CM[5], CM[1], CM[3], CM[2], CM[2], CM[3],
            CM[5], CM[2], CM[3], CM[3], CL[6], CL[6], CL[6], CM[1],
            CM[2], CM[3], CM[2], CL[7], CL[6], CM[1], CL[5] ]

beat_1 = [  1, 1, 3, 1, 1, 3, 1, 1,                          # Beats of song 1, 1␣
↪means 1/8 beats
            1, 1, 1, 1, 1, 1, 3, 1,
            1, 3, 1, 1, 1, 1, 1, 1,
            1, 2, 1, 1, 1, 1, 1, 1,
            1, 1, 3 ]

song_2 = [  CM[1], CM[1], CM[1], CL[5], CM[3], CM[3], CM[3], CM[1], # Notes of song2
            CM[1], CM[3], CM[5], CM[5], CM[4], CM[3], CM[2], CM[2],
            CM[3], CM[4], CM[4], CM[3], CM[2], CM[3], CM[1], CM[1],
            CM[3], CM[2], CL[5], CL[7], CM[2], CM[1]        ]

beat_2 = [  1, 1, 2, 2, 1, 1, 2, 2,                          # Beats of song 2, 1␣
↪means 1/8 beats
            1, 1, 2, 2, 1, 1, 3, 1,
            1, 2, 2, 1, 1, 2, 2, 1,
            1, 2, 2, 1, 1, 3 ]

def setup():
    GPIO.setmode(GPIO.BOARD)                # Numbers GPIOs by physical location
```

(continues on next page)

(continued from previous page)

```
    GPIO.setup(Buzzer, GPIO.OUT)     # Set pins' mode is output
    global Buzz                                    # Assign a global␣
↪variable to replace GPIO.PWM
    Buzz = GPIO.PWM(Buzzer, 440)     # 440 is initial frequency.
    Buzz.start(50)                                 # Start Buzzer pin with 50% duty␣
↪ration

def loop():
    while True:
        #    Playing song 1...
        for i in range(1, 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
        time.sleep(1)                                           # Wait a second␣
↪for next song.

        #    Playing song 2...
        for i in range(1, len(song_2)):     # Play song 1
            Buzz.ChangeFrequency(song_2[i]) # Change the frequency along the song note
            time.sleep(beat_2[i] * 0.5)     # delay a note for beat * 0.5s

def destory():
    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.
        destory()
```

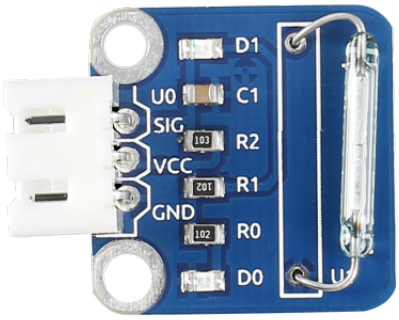
Now you can hear the passive buzzer playing music.



6.11 Lesson 11 Reed Switch

Introduction

A reed switch (as shown below) is used to detect the magnetic field. Hall sensors are generally used to measure the speed of intelligent vehicles and count in assembly lines, while reed switches are often used to detect the existence of a magnetic field.



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard

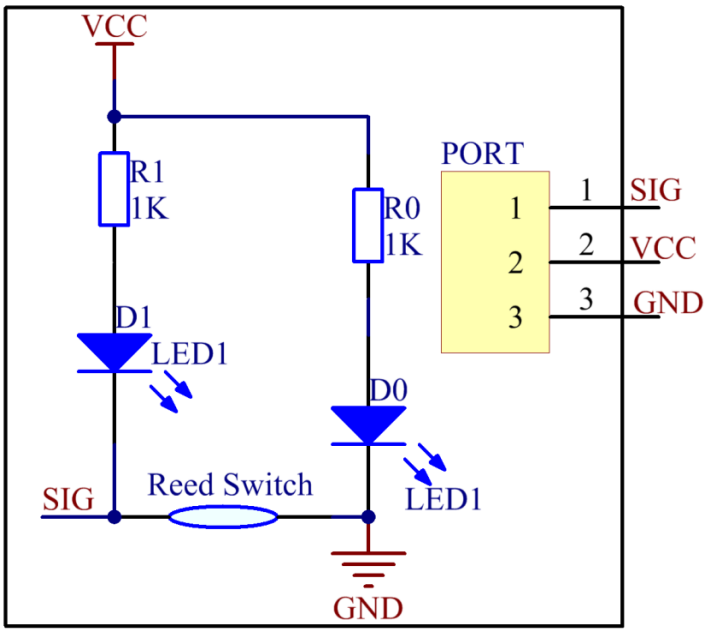
- 1 * Reed switch module
- 1 * Dual-color LED module
- 2 * 3-Pin anti-reverse cable
- 1 * Magnet (Self provided)

Experimental Principle

A reed switch is a type of line switch component that realizes control by magnetic signals. It induces by a magnet. The "switch" here means dry reed pipe, which is a kind of contact passive electronic switch component with the advantage of simple structure, small size, and convenient control. The shell of a reed switch is commonly a sealed glass pipe in which two iron elastic reed electroplates are equipped and inert gases are filled. Normally, the two reeds made of special materials in the glass tube are separated. However, when a magnetic substance approaches the glass tube, the two reeds in the glass tube are magnetized to attract each other and contact under the function of magnetic field lines. As a result, the two reeds will pull together to connect the circuit connected with the nodes.

After external magnetic force disappears, the two reeds will be separated with each other because they have the same magnetism, so the circuit is also disconnected. Therefore, as a line switch component controlling by magnetic signals, the dry reed pipe can be used as a sensor to count, limit positions and so on. At the same time, it is widely used in a variety of communication devices.

The schematic diagram of the module is as shown below:

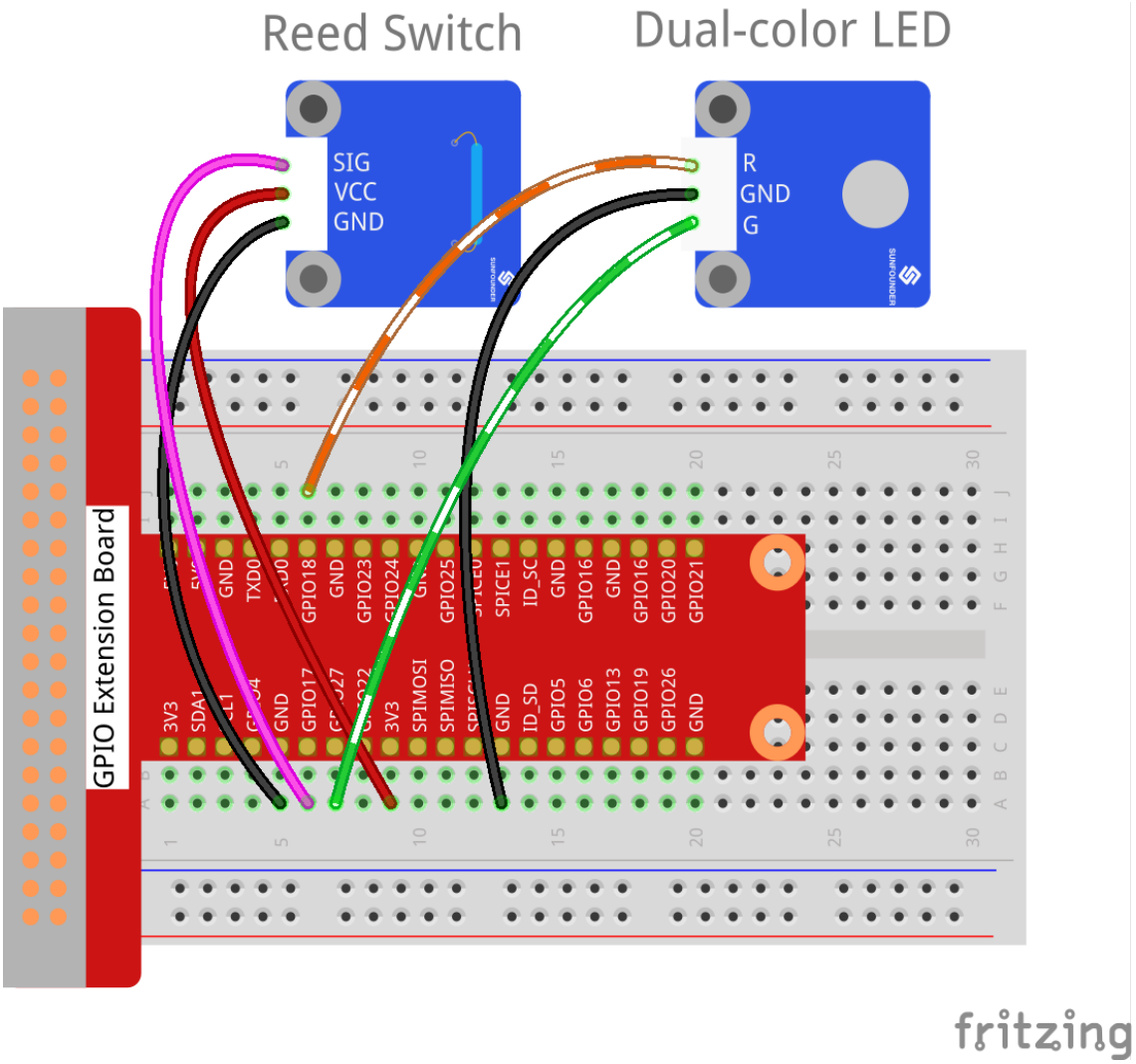


Experimental Procedures

Step 1: Build the circuit

Raspberry Pi	GPIO Extension Board	Reed Switch Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/11_reed_switch/
```

Step 3: Compile.

```
gcc reed_switch.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define ReedPin             0
#define Gpin                2
#define Rpin                1

void LED(char* color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == "RED")
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == "GREEN")
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

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

    pinMode(ReedPin, INPUT);
    LED("GREEN");

    while(1){
        if(0 == digitalRead(ReedPin)){
            delay(10);
            if(0 == digitalRead(ReedPin)){
                LED("RED");
                printf("Detected Magnetic Material!\n");
            }
        }
        else if(1 == digitalRead(ReedPin)){
            delay(10);
            if(1 == digitalRead(ReedPin)){
                while(!digitalRead(ReedPin));
                LED("GREEN");
            }
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

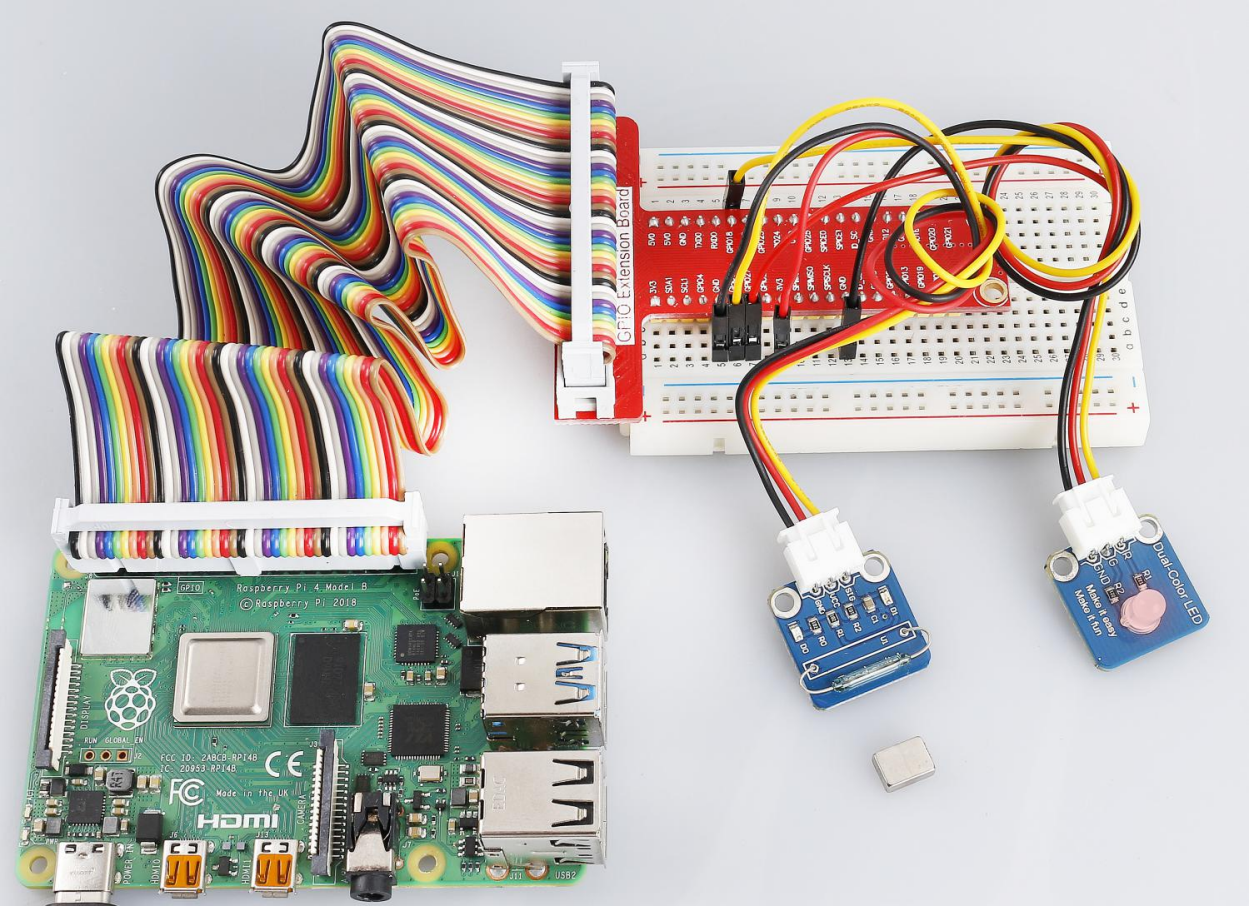
Step 3: Run.

```
sudo python3 11_reed_switch.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

ReedPin = 11
Gpin    = 13
Rpin    = 12

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)      # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)      # Set Red Led Pin mode to output
    GPIO.setup(ReedPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode is␣
↪input, and pull up to high level(3.3V)
    GPIO.add_event_detect(ReedPin, GPIO.BOTH, callback=detect, bouncetime=200)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)

def detect(chn):
    Led(GPIO.input(ReedPin))

def loop():
    while True:
        pass

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    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()
```

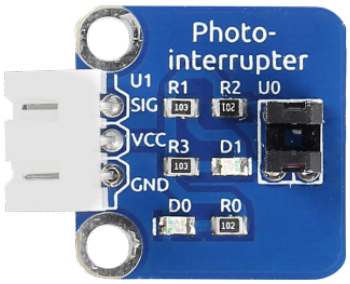
Then the LED will flash green. Place a magnet near the reed switch, “Detected Magnetic Material!” will be printed on the screen and the LED will change to red. Move away the magnet, the LED will turn green again.



6.12 Lesson 12 Photo-interrupter

Introduction

A photo-interrupter (as shown below) is a sensor with a light-emitting component and light-receiving component packaged and placed on face-to-face. It applies the principle that light is interrupted when an object passes through the sensor. Therefore, photo-interrupters are widely used in speed measurement.



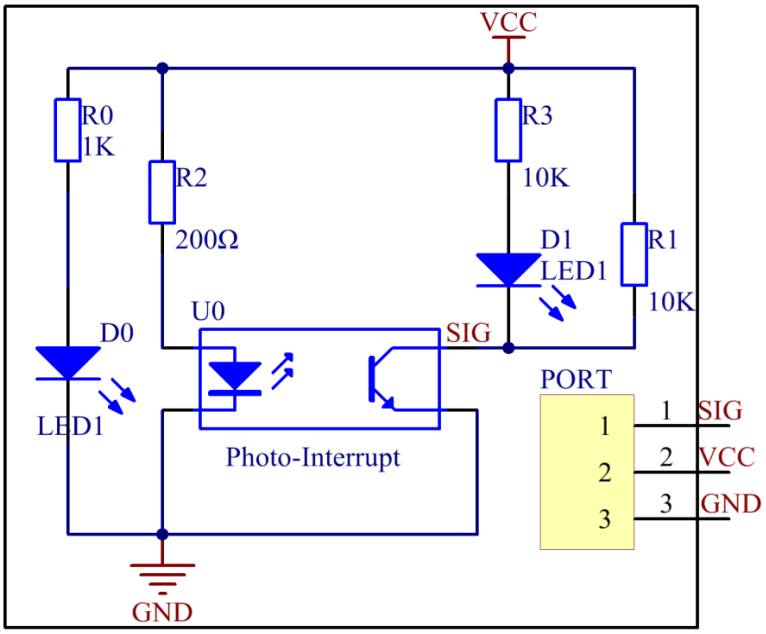
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard

- 1 * Dual-color LED module
- 1 * Photo-interrupter module
- 2 * 3-Pin anti-reverse cable

Experimental Principle

Basically a photo-interrupter consists of two parts: transmitter and receiver. The transmitter (e.g., an LED or a laser) emits light and then the light goes to the receiver. If that light beam between the transmitter and receiver is interrupted by an obstacle, the receiver will detect no incident light even for a moment and the output level will change. In this experiment, we will turn an LED on or off by using this change. The schematic diagram:

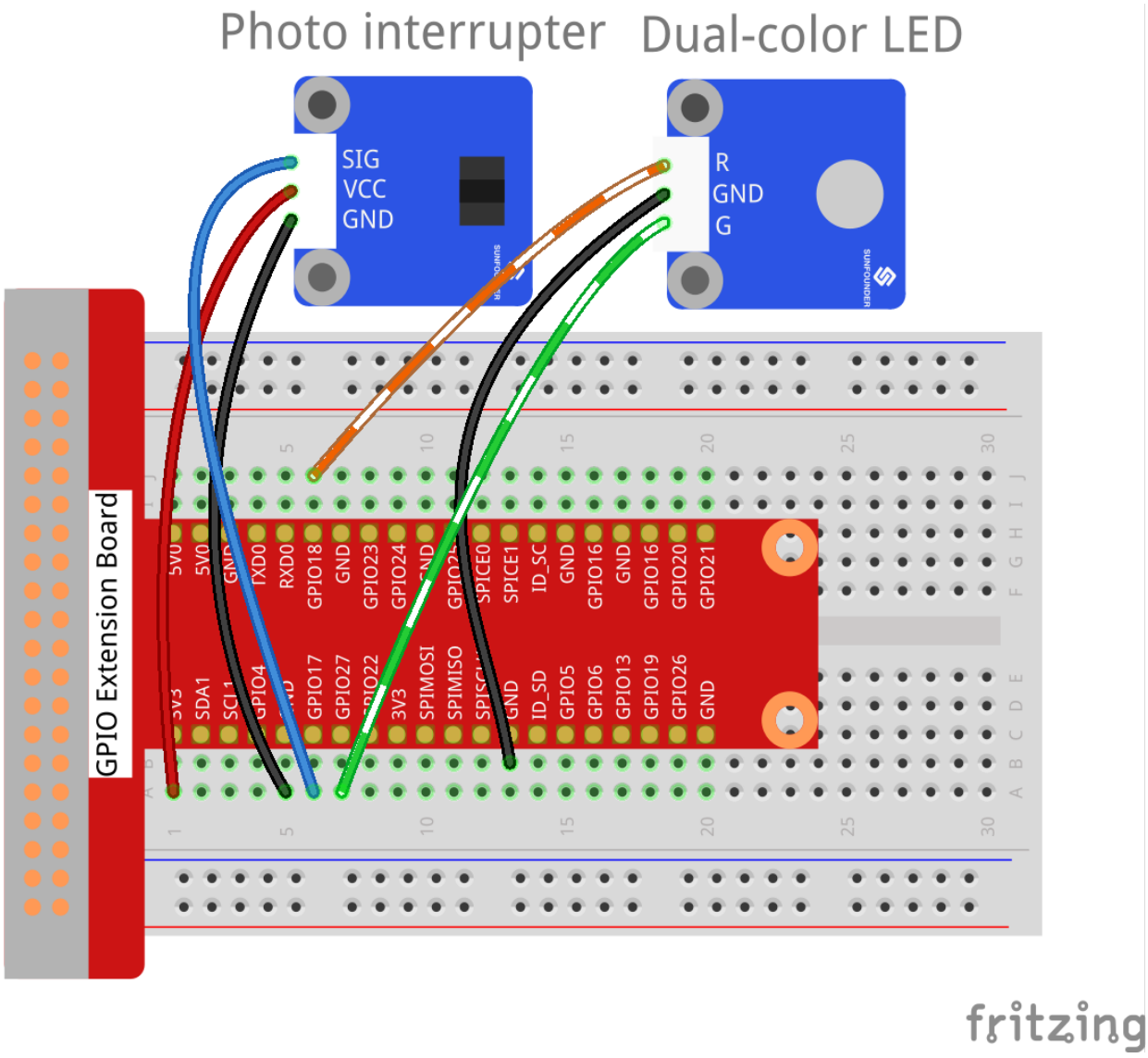


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Photo-interrupter Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/12_photo_interrupter/
```

Step 3: Compile.

```
gcc photo_interrupter.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define LBPin               0  // light break pin set to GPIO0
#define Gpin                2
#define Rpin                1

void LED(int color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == 0){
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == 1){
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
}

void Print(int x){
    if ( x == 0 ){
        printf("Light was blocked\n");
    }
}

int main(void){

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

    pinMode(LBPin, INPUT);
    int temp;
    while(1){
        //Reverse the input of LBPin
        if ( digitalRead(LBPin) == 0 ){
            temp = 1;
        }
        if ( digitalRead(LBPin) == 1 ){
            temp = 0;
        }

        LED(temp);
        Print(temp);
        delay(100);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

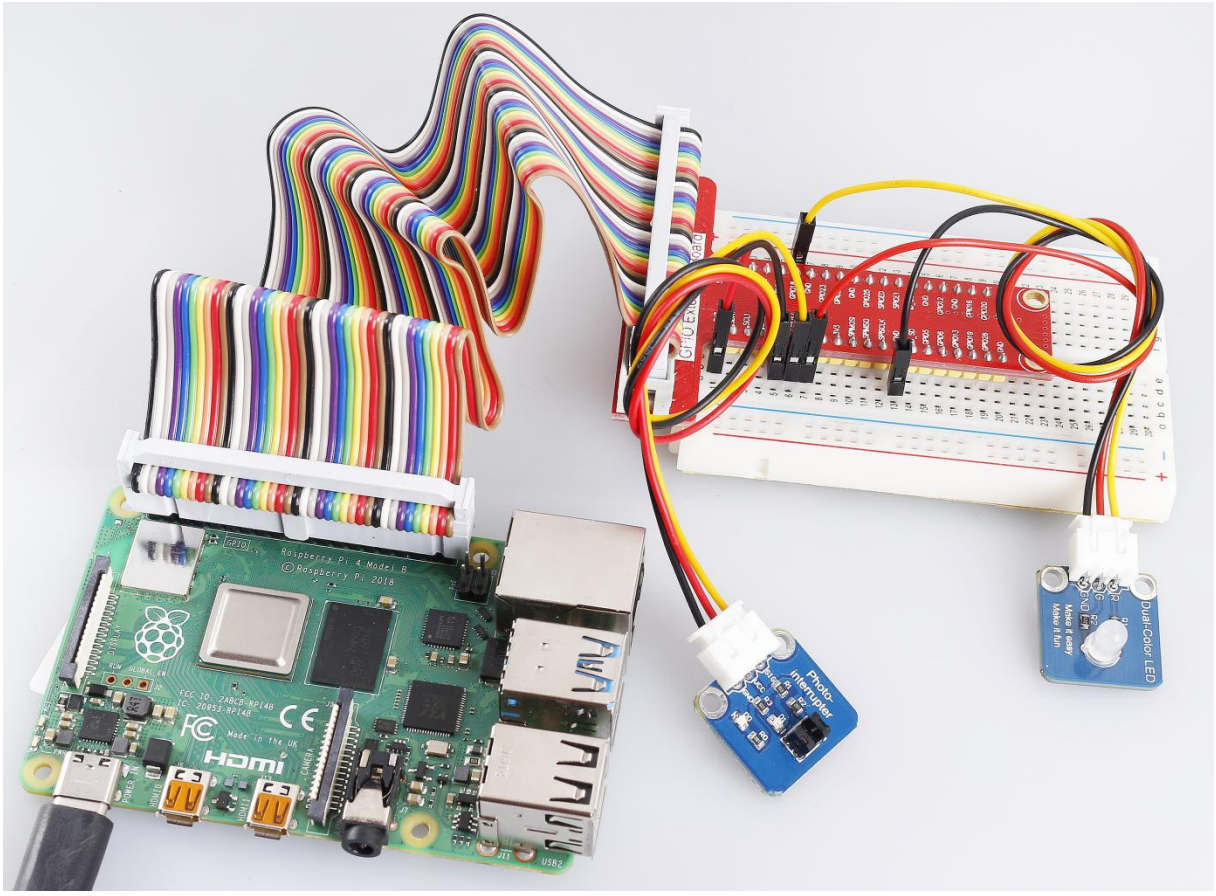
Step 3: Run.

```
sudo python3 12_photo_interrupter.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

PIPin  = 11
Gpin   = 13
Rpin   = 12

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)      # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)      # Set Red Led Pin mode to output
    GPIO.setup(PIPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode is␣
↪input, and pull up to high level(3.3V)
    GPIO.add_event_detect(PIPin, GPIO.BOTH, callback=detect, bouncetime=200)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)
        print ('Light was blocked')

def detect(chn):
    Led(GPIO.input(PIPin))

def loop():
    while True:
        pass

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    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()
```

Now the LED will light up green. Stick a piece of paper in the gap of photo interrupter. Then “Light was blocked” will be printed on the screen and the LED will flash red. Remove the paper, and the LED will turn green again.

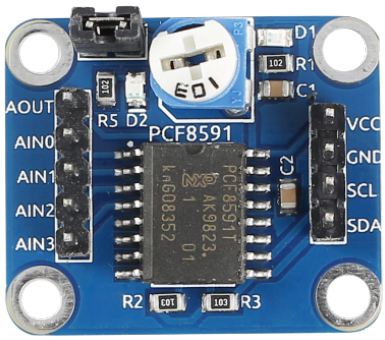


6.13 Lesson 13 PCF8591

Introduction

The PCF8591 is a single-chip, single-supply low-power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I2C-bus interface. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I2C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I2C-bus.

The functions of the device include analog input multiplexing, on-chip track and hold function, 8-bit analog-to-digital conversion and an 8-bit digital-to-analog conversion. The maximum conversion rate is given by the maximum speed of the I2C-bus.



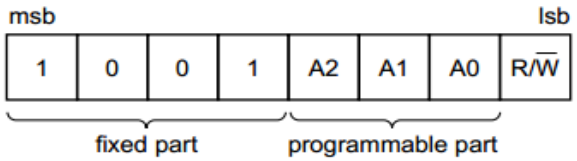
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- Several Jumper wires
- 1 * PCF8591 module
- 1 * Dual-Color LED module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

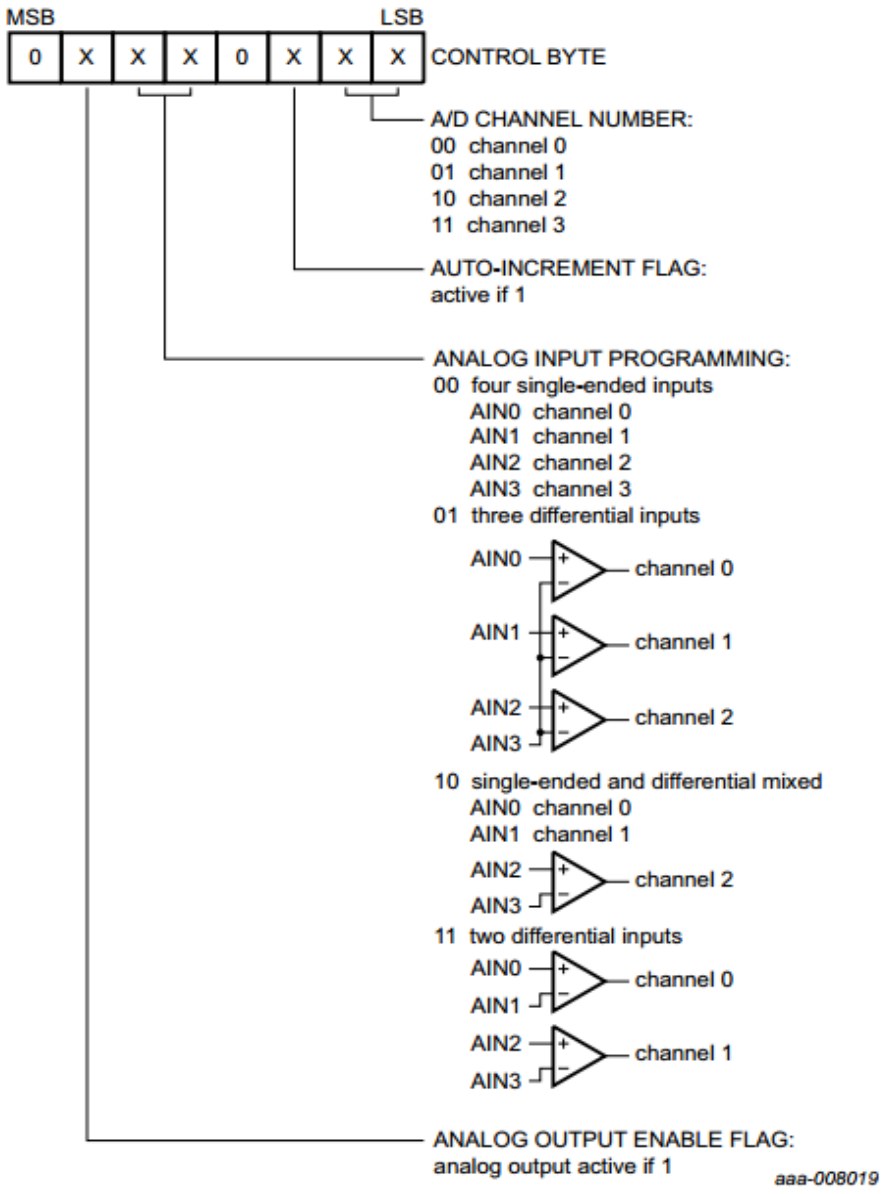
Addressing:

Each PCF8591 device in an I2C-bus system is activated by sending a valid address to the device. The address consists of a fixed part and a programmable part. The programmable part must be set according to the address pins A0, A1 and A2. The address always has to be sent as the first byte after the start condition in the I2C-bus protocol. The last bit of the address byte is the read/write-bit which sets the direction of the following data transfer (see as below).



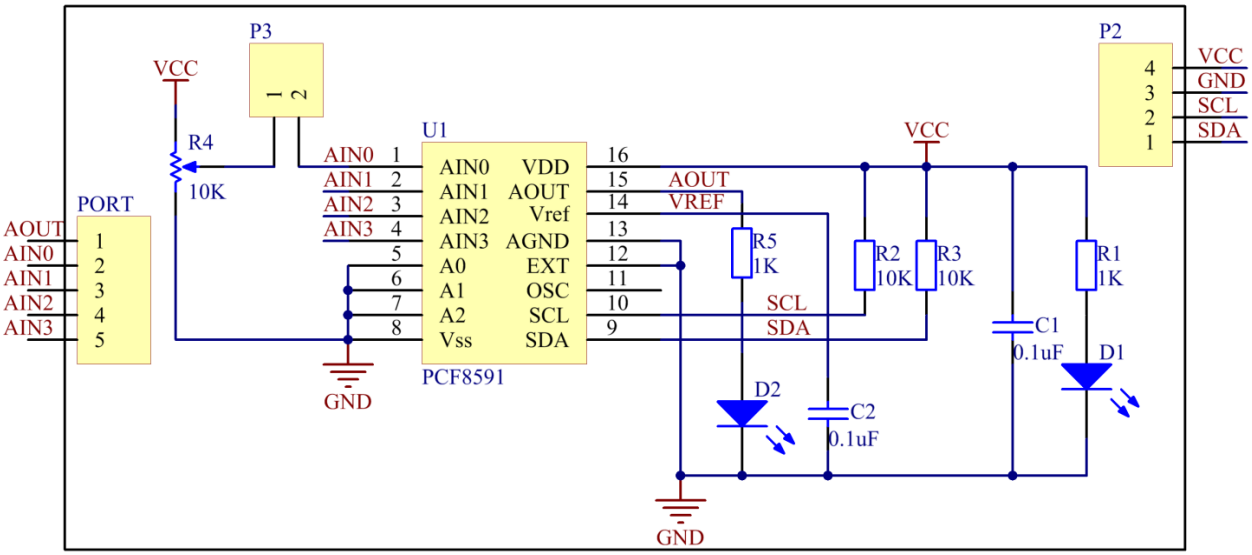
Control byte:

The second byte sent to a PCF8591 device will be stored in its control register and is required to control the device function. The upper nibble of the control register is used for enabling the analog output, and for programming the analog inputs as single-ended or differential inputs. The lower nibble selects one of the analog input channels defined by the upper nibble (see Fig.5). If the auto-increment flag is set, the channel number is incremented automatically after each A/D conversion. See the figure below.



In this experiment, the AIN0 (Analog Input 0) port is used to receive analog signals from the potentiometer module, and AOUT (Analog Output) is used to output analog signals to the dual-color LED module so as to change the luminance of the LED.

The schematic diagram:



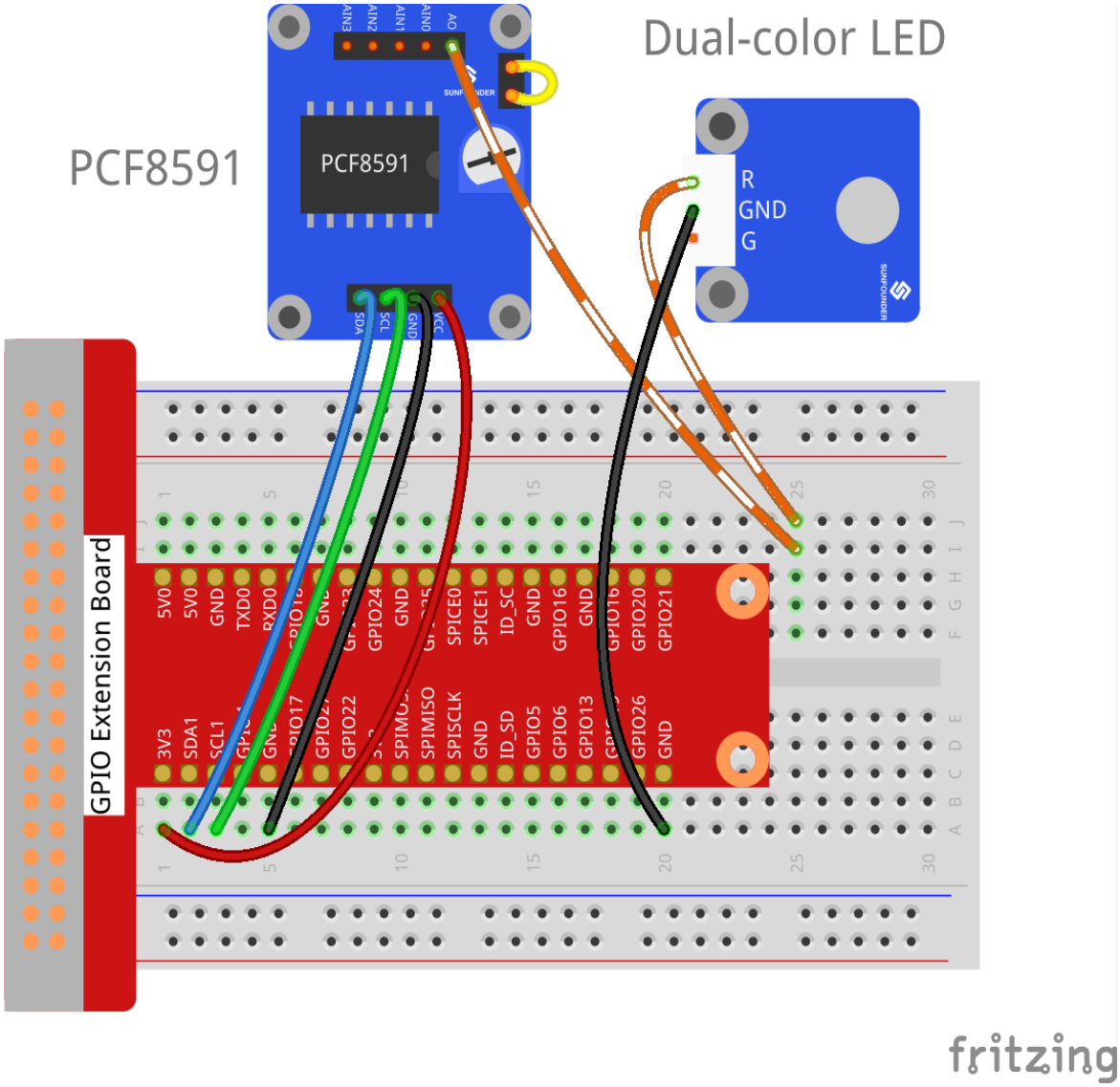
Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Dual-Color Module	GPIO Extension Board	PCF8591 Module
R	*	AOUT
GND	GND	GND
G	*	*

Note: Connect the two pins next to the potentiometer of the PCF8591 module with the jumper cap attached.



Step 2: Setup I2C (see **Appendix**. If you have set I2C, skip this step.)

For C Users:

Step 3: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/13_pcf8591/
```

Step 4: Compile.

```
gcc pcf8591.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 5: Run.

```
sudo ./a.out
```

Code

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

#define PCF       120

int main (void)
{
    int value ;
    wiringPiSetup () ;
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup (PCF, 0x48) ;
    while(1) // loop forever
    {
        value = analogRead  (PCF + 0) ;
        printf("%d\n", value);
        analogWrite (PCF + 0, value) ;
        delay (10) ;
    }
    return 0 ;
}
```

For Python Users:

Step 3: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 4: Run.

```
sudo python3 13_pcf8591.py
```

Note:

- If you get the error `FileNotFoundError: [Errno 2] No such file or directory: '/dev/i2c-1'`, you need to setup I2C (see *I2C Configuration*).
- If you get `ModuleNotFoundError: No module named 'smbus2'` error, please run the command: `sudo pip3 install smbus2`.
- If the error `OSError: [Errno 121] Remote I/O` appears, it means the module is miswired or the module is broken.

Code

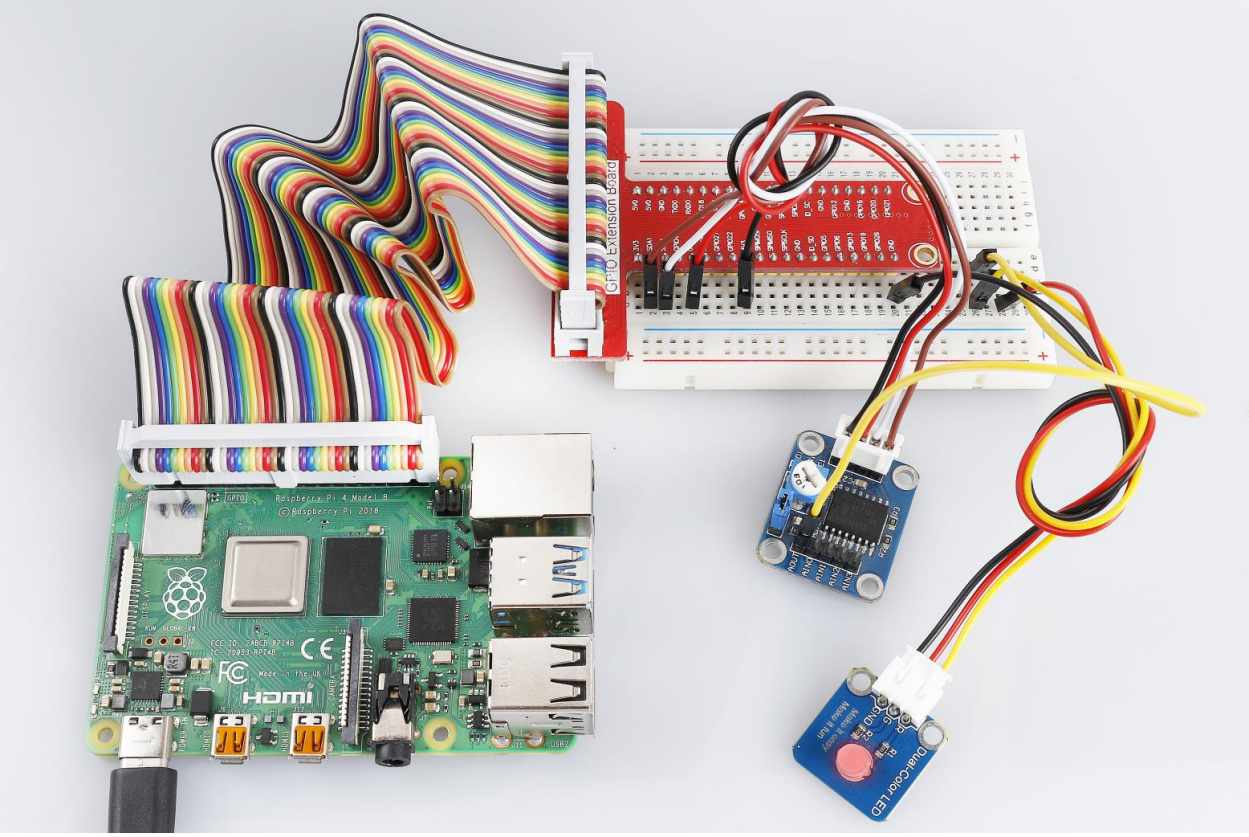
```
#!/usr/bin/env python3
import PCF8591 as ADC

def setup():
    ADC.setup(0x48)

def loop():
```

(continues on next page)

(continued from previous page)

```
    while True:
        print (ADC.read(0))
        ADC.write(ADC.read(0))

def destroy():
    ADC.write(0)

if __name__ == "__main__":
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()
```

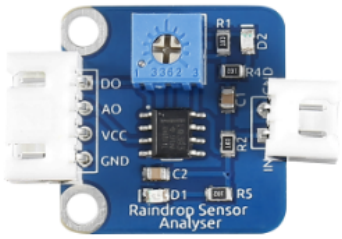
Now, turn the knob of the potentiometer on PCF8591, and you can see the luminance of the LED change and a value between 0 and 255 printed on the screen.

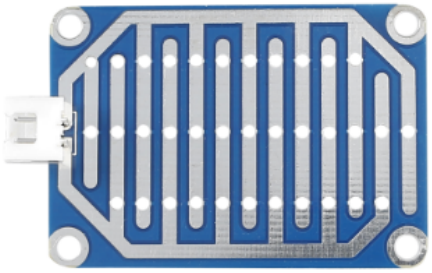


6.14 Lesson 14 Rain Detection Module

Introduction

The rain detection module detects rain on the board. Place the rain detection board in the open air. When it is raining, the rain detection module will sense the raindrops and send signals to the Raspberry Pi.



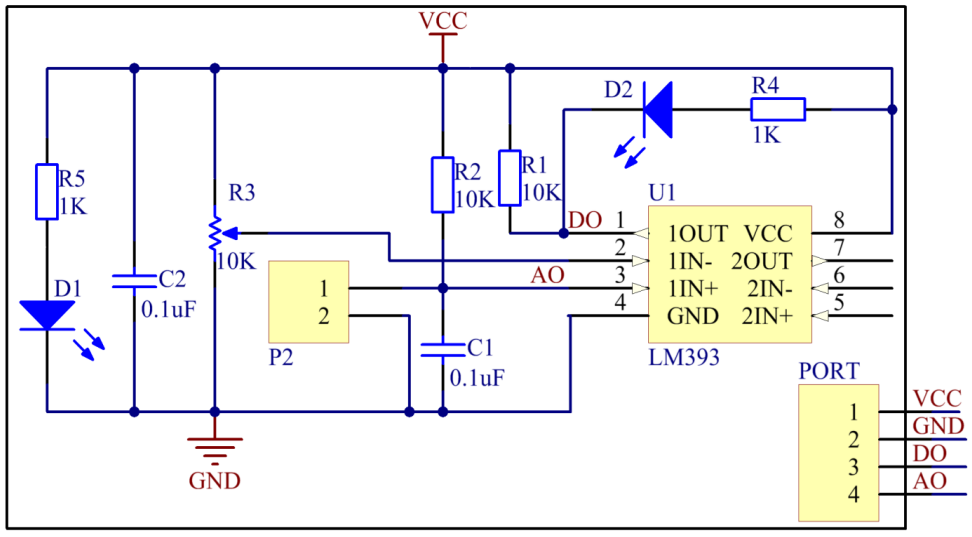


Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Rain Detection module
- 1 * PCF8591
- 1 * LM393
- 1 * 2-Pin ribbon cable
- 1 * 4-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

There are two metal wires that are close to each other but do not cross on the rain detection board. When rain drops on the board, the two metal wires will conduct, thus there is a voltage between the two metal wires. The schematic diagram:



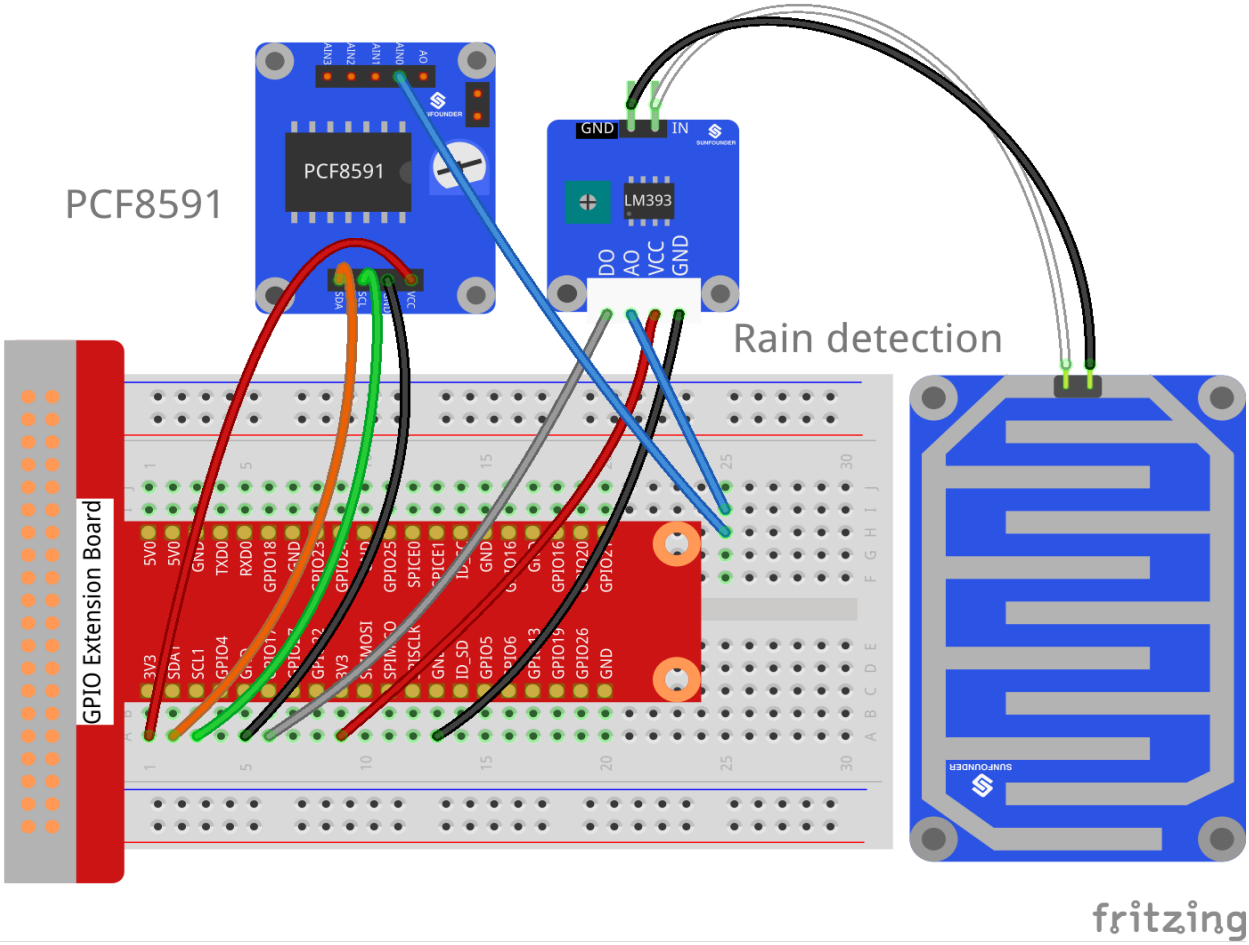
Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

LM393	GPIO Extension Board	PCF8591 Module
DO	GPIO17	*
AO	*	AIN0
VCC	3V3	VCC
GND	GND	GND

Note: The two pins on the rain detection board are exactly the same. You can connect them to pin IN and GND on LM393.



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/14_rain_detector/
```

Step 3: Compile.

```
gcc rain_detector.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define             PCF     120
#define             DOpin   0

void Print(int x)
{
    switch(x)
    {
        case 1:
            printf("\n***************\n"  );
            printf(  "* Not Raining *\n"  );
            printf(  "***************\n\n");
        break;
        case 0:
            printf("\n*************\n"  );
            printf(  "* Raining!! *\n"  );
            printf(  "*************\n\n");
        break;
        default:
            printf("\n**********************\n"  );
            printf(  "* Print value error. *\n"  );
            printf(  "**********************\n\n");
        break;
    }
}

int main()
{
    int analogVal;
    int tmp, status;

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    // Setup pcf8591 on base pin 120, and address 0x48
```

(continues on next page)

(continued from previous page)

```
    pcf8591Setup(PCF, 0x48);

    pinMode(DOpin, INPUT);

    status = 0;
    while(1) // loop forever
    {
        analogVal = analogRead(PCF + 0);
        printf("%d\n", analogVal);

        tmp = digitalRead(DOpin);

        if (tmp != status)
        {
            Print(tmp);
            status = tmp;
        }

        delay (200);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 14_rain_detector.py
```

Code

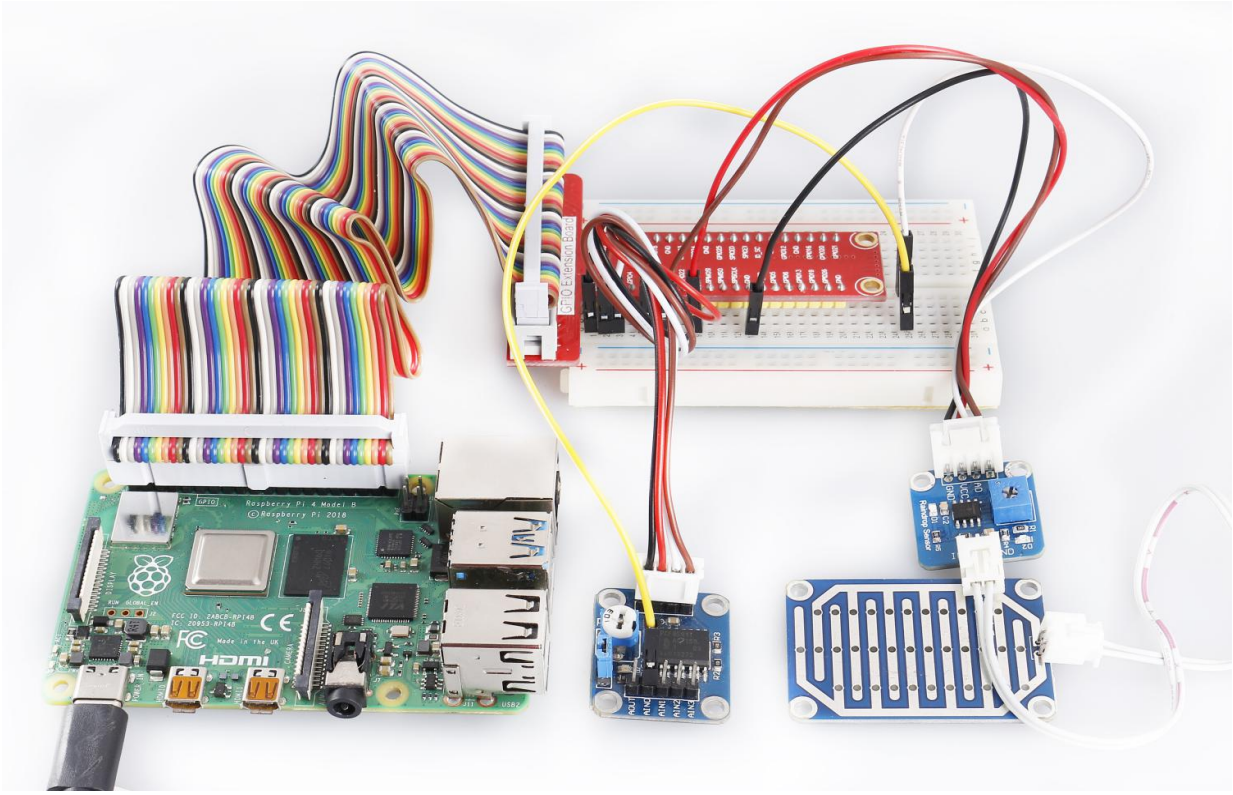
```
#!/usr/bin/env python3
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time
import math

DO = 17
GPIO.setmode(GPIO.BCM)

def setup():
    ADC.setup(0x48)
    GPIO.setup(DO, GPIO.IN)

def Print(x):
    if x == 1:
        print ('')
        print ('    ***************')
        print ('    * Not raining *')
        print ('    ***************')
        print ('')
    if x == 0:
        print ('')
```

(continues on next page)

(continued from previous page)

```
        print ('    **************')
        print ('    * Raining!! *')
        print ('    **************')
        print ('')

def loop():
    status = 1
    while True:
        print (ADC.read(0))

        tmp = GPIO.input(DO);
        if tmp != status:
            Print(tmp)
            status = tmp

        time.sleep(0.2)

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        pass
```

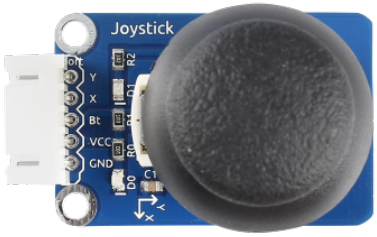
Now drop some water onto the rain detection board until "**raining**" displayed on the screen. You can adjust the potentiometer on LM393 to detect the threshold of rainfall.



6.15 Lesson 15 Joystick PS2

Introduction

There are five operation directions for joystick PS2: up, down, left, right and press-down.



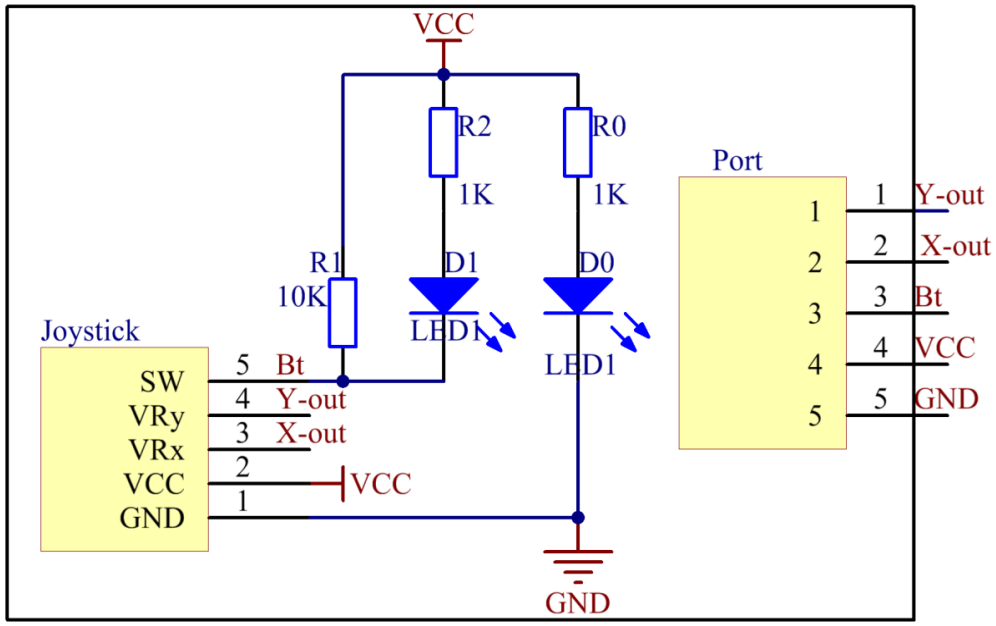
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * PCF8591
- 1 * Joystick PS2 module
- 1 * 5-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

This module has two analog outputs (corresponding to X and Y coordinates) and one digital output representing whether it is pressed on Z axis.

In this experiment, we connect pin X and Y to the analog input ports of the A/D convertor so as to convert analog quantities into digital ones. Then program on Raspberry Pi to detect the moving direction of the Joystick. The schematic diagram:

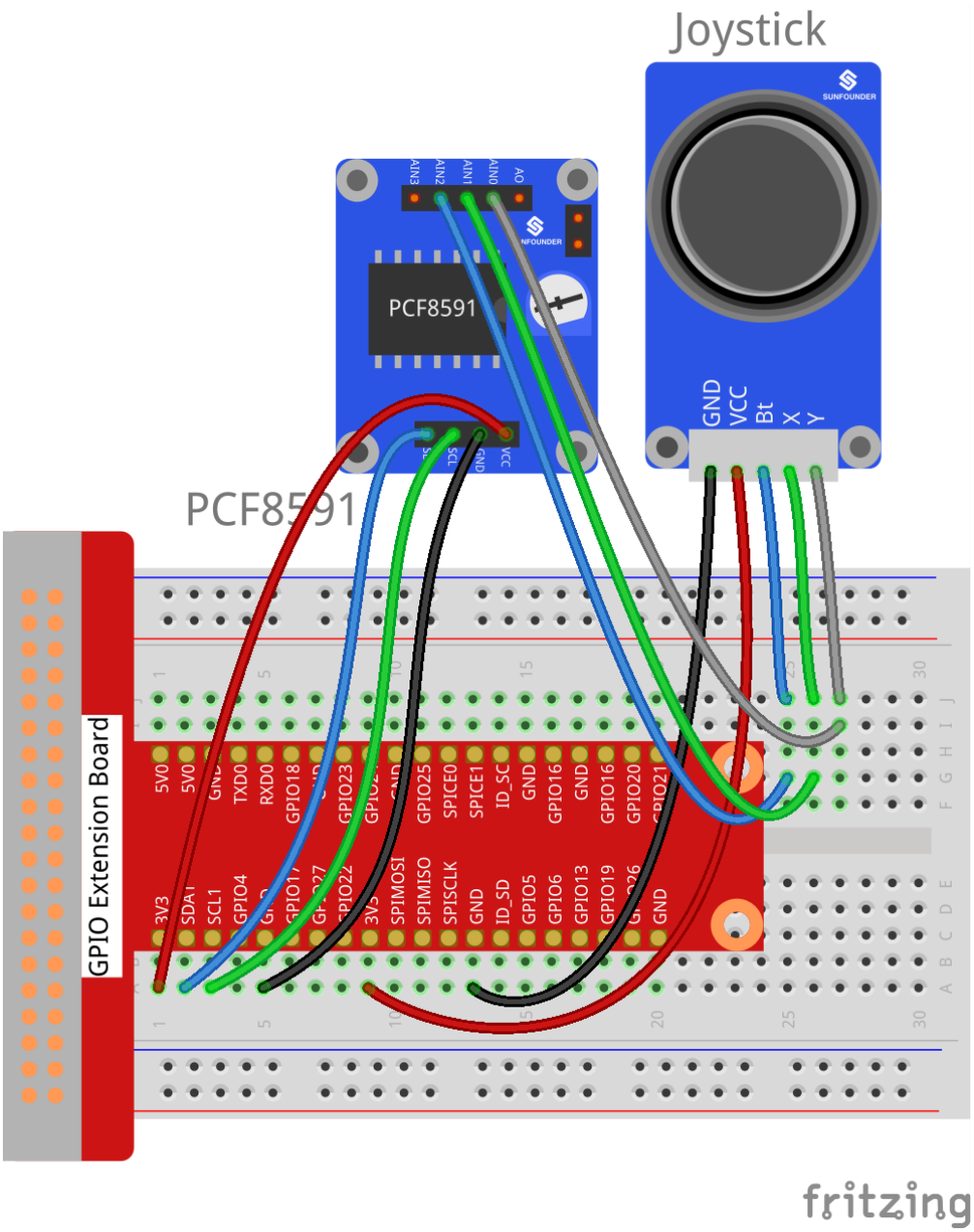


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Joystick PS2	GPIO Extension Board	PCF8591 Module
Y	*	AIN0
X	*	AIN1
Bt	*	AIN2
VCC	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/15_joystick_PS2/
```

Step 3: Compile.

```
gcc joystick_PS2.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define PCF       120
#define uchar       unsigned char

int AIN0 = PCF + 0;
int AIN1 = PCF + 1;
int AIN2 = PCF + 2;

char *state[7] = {"home", "up", "down", "left", "right", "pressed"};

int direction(){
    int x, y, b;
    int tmp=0;
    x = analogRead(AIN1);
    y = analogRead(AIN0);
    b = analogRead(AIN2);
    if (y <= 30)
        tmp = 1;            // up
    if (y >= 225)
        tmp = 2;            // down

    if (x >= 225)
        tmp = 3;            // left
    if (x <= 30)
        tmp = 4;            // right

    if (b <= 30)
        tmp = 5;            // button preesd
    if (x-125<15 && x-125>-15 && y-125<15 && y-125>-15 && b >= 60)
        tmp = 0;            // home position

    return tmp;
}

int main (void)
{
    int tmp=0;
    int status = 0;
    wiringPiSetup ();
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup (PCF, 0x48);
    while(1) // loop forever
    {
        tmp = direction();
        if (tmp != status)
        {
            printf("%s\n", state[tmp]);
            status = tmp;
        }
```

(continues on next page)

(continued from previous page)

```
    }
    return 0 ;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 15_joystick_PS2.py
```

Code

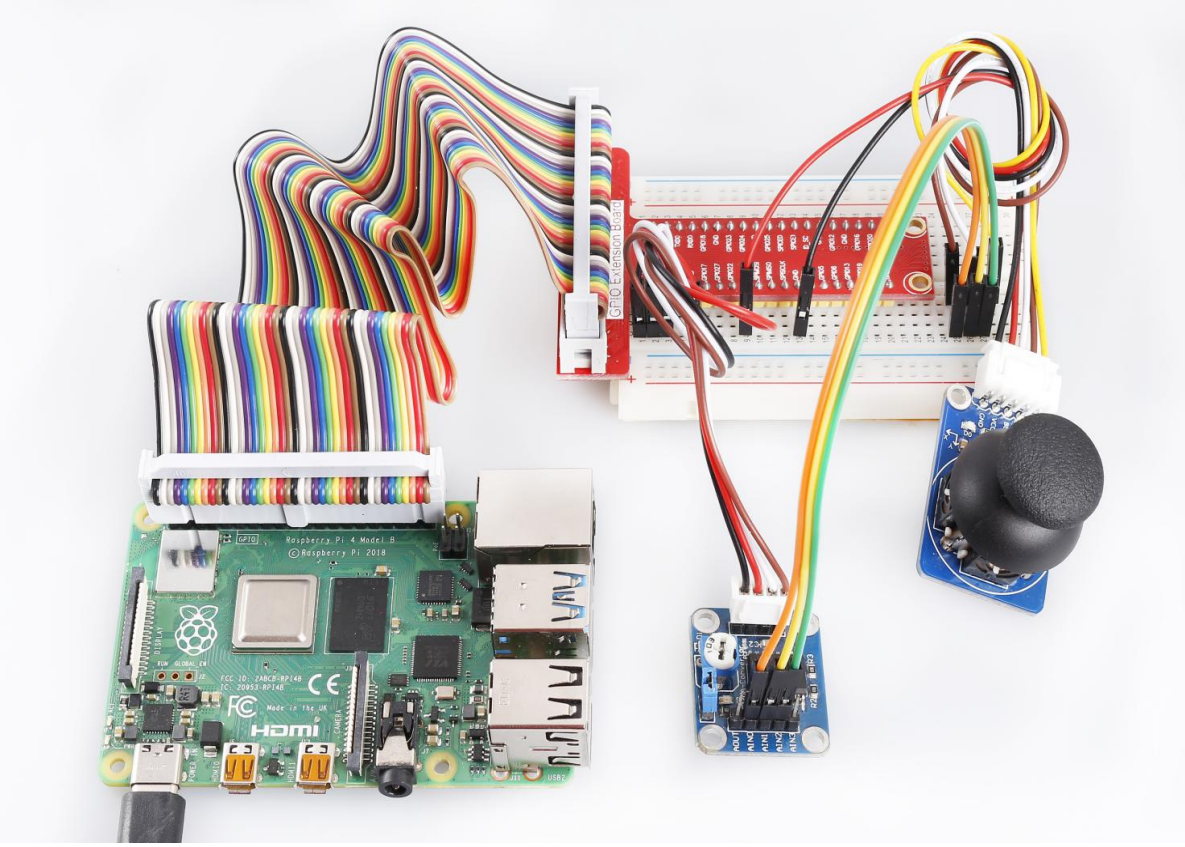
```
#!/usr/bin/env python3
import PCF8591 as ADC
import time

def setup():
    ADC.setup(0x48)                                    # Setup PCF8591
    global state

def direction():    #get joystick result
    state = ['home', 'up', 'down', 'left', 'right', 'pressed']
    i = 0
    if ADC.read(0) <= 30:
        i = 1               #up
    if ADC.read(0) >= 225:
        i = 2               #down

    if ADC.read(1) >= 225:
        i = 3               #left
    if ADC.read(1) <= 30:
        i = 4               #right

    if ADC.read(2) <= 30:
        i = 5               # Button pressed

    if ADC.read(0) - 125 < 15 and ADC.read(0) - 125 > -15   and ADC.read(1) - 125 <␣
↪15 and ADC.read(1) - 125 > -15 and ADC.read(2) == 255:
        i = 0

    return state[i]

def loop():
    status = ''
    while True:
        tmp = direction()
        if tmp != None and tmp != status:
            print (tmp)
            status = tmp

def destroy():
    pass
```

(continues on next page)

(continued from previous page)

```
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()
```

Now push the rocker upwards, and a string "**up**" will be printed on the screen; push it downwards, and "**down**" will be printed; if you push it left, "**Left**" will be printed on; If you push it right, and "**Right**" will be printed; If you press down the cap, "**Button Pressed**" will be printed on the screen.



6.16 Lesson 16 Potentiometer Module

Introduction

A potentiometer is a device which is used to vary the resistance in an electrical circuit without interrupting the circuit.

Required Components

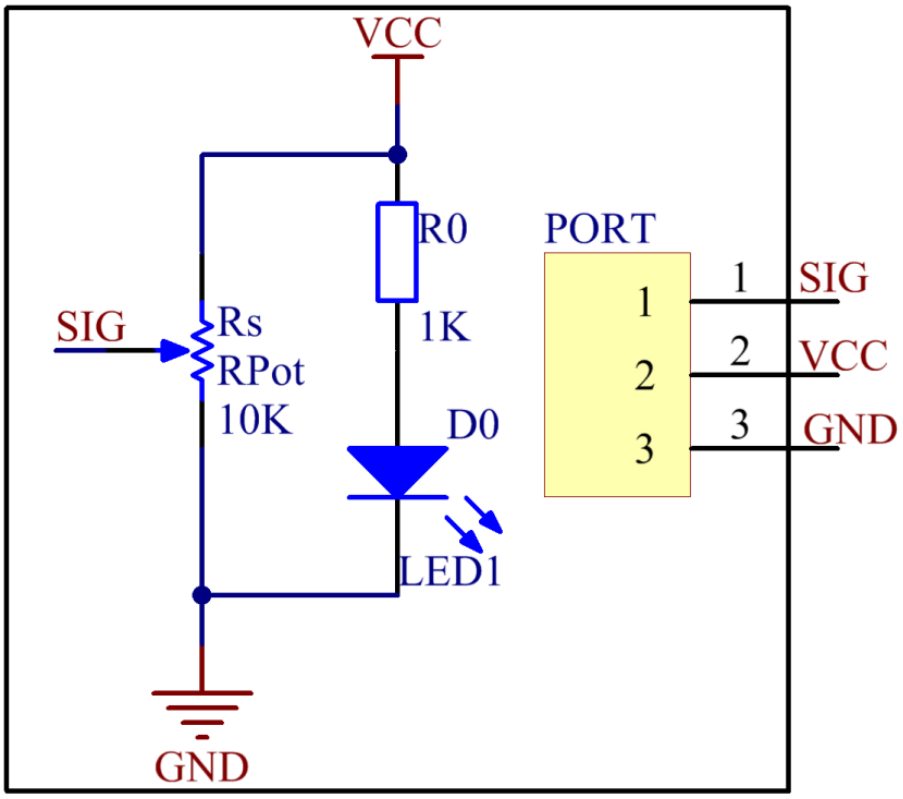
- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Potentiometer module
- 1 * Dual-Color LED module
- 2 * 3-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

An analog potentiometer is an analog electronic component. What's the difference between an analog one and a digital one? Simply put, a digital potentiometer refers to just two states like on/off, high/low levels, i.e. either 0 or 1, while a digital one supports analog signals like a number from 1 to 1000. The signal value changes over time instead of keeping an exact number. Analog signals include light intensity, humidity, temperature, and so on.

In this experiment, PCF8591 is used to read the analog value of the potentiometer and output the value to LED. Connect pin SIG of the potentiometer to pin AIN0 of PCF8591. Connect pin R or Pin G of the Dual-Color LED to pin AOUT of PCF8591 to observe the change of LED.

The schematic diagram of the module is as shown below:



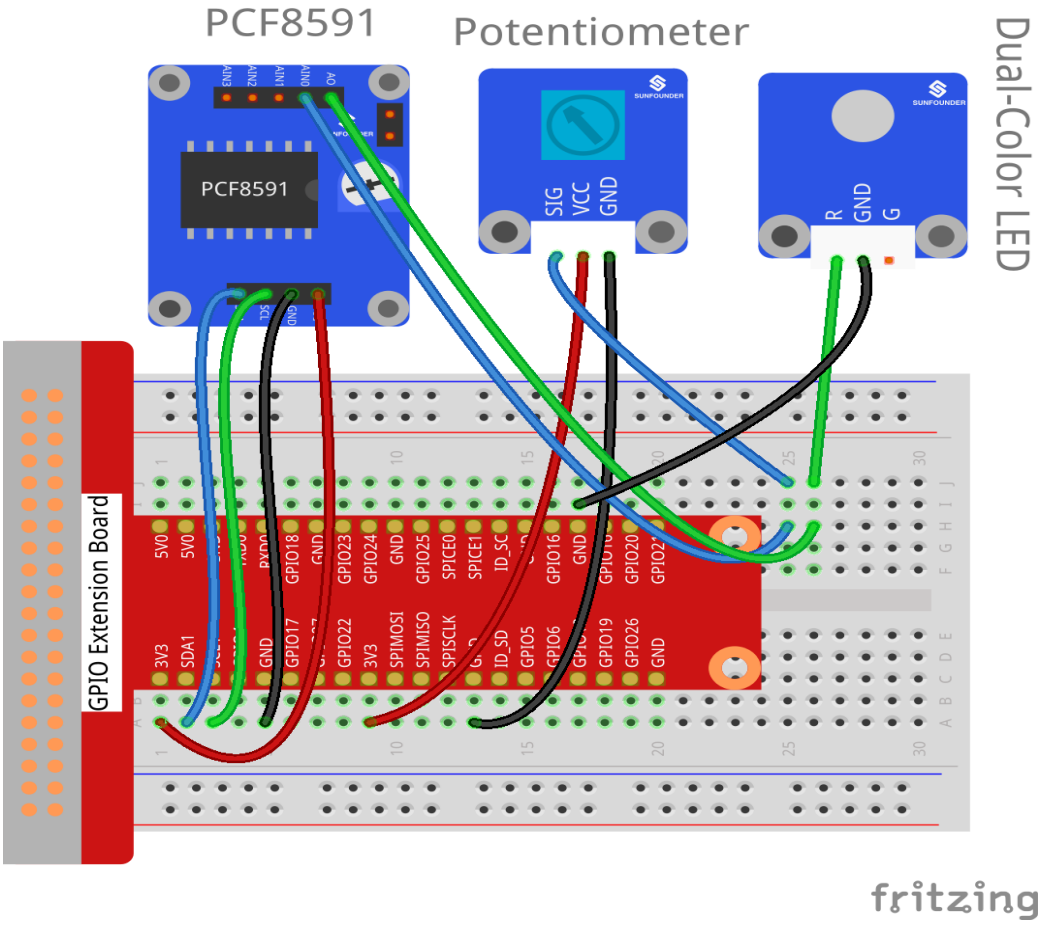
Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Potentiometer	GPIO Extension Board	PCF8591 Module
SIG	*	AIN0
VCC	3V3	VCC
GND	GND	GND

Dual-Color Module	GPIO Extension Board	PCF8591 Module
R	*	AOUT
GND	GND	GND
G	*	*



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/16_potentiometer/
```

Step 3: Compile.

```
gcc potentiometer.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define PCF       120

int main (void)
{
    int value ;
    wiringPiSetup () ;
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup (PCF, 0x48) ;
    while(1) // loop forever
    {
        value = analogRead  (PCF + 0) ;
        printf("Value: %d\n", value);
        analogWrite (PCF + 0, value) ;
        delay (200) ;
    }
    return 0 ;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 16_potentiometer.py
```

Code

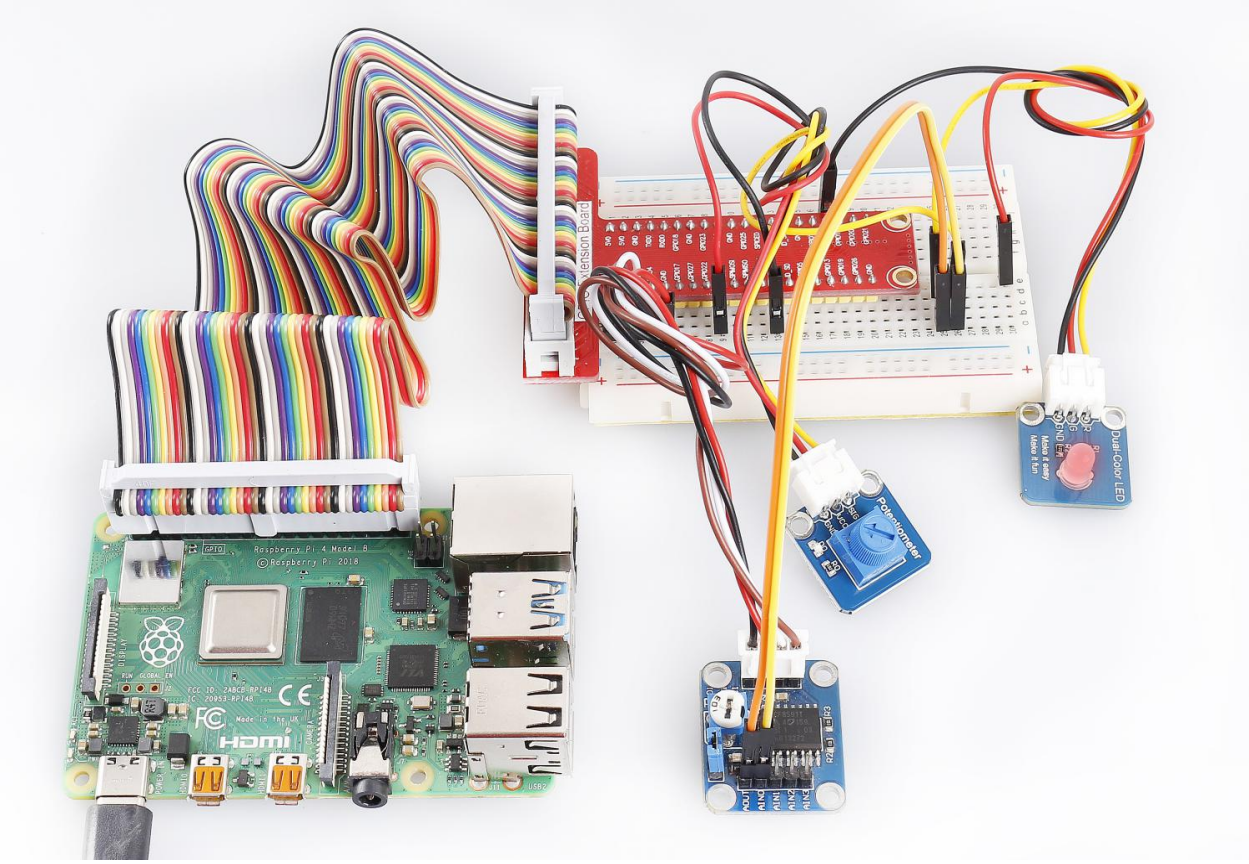
```
#!/usr/bin/env python3
import PCF8591 as ADC
import time

def setup():
    ADC.setup(0x48)

def loop():
    status = 1
    while True:
        print ('Value:', ADC.read(0))
        Value = ADC.read(0)
        outvalue = map(Value,0,255,120,255)
        ADC.write(outvalue)
        time.sleep(0.2)
def destroy():
    ADC.write(0)

def map(x, in_min, in_max, out_min, out_max):
        '''To map the value from arange to another'''
        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

if __name__ == '__main__':
    try:
```

(continues on next page)

(continued from previous page)

```
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()
```

Turn the knob of the potentiometer, and you can see the value printed on the screen change from 0 (minimum) to 255 (maximum).



6.17 Lesson 17 Hall Sensor

Introduction

Based on Hall Effect, a Hall sensor is a one that varies its output voltage in response to a magnetic field. Hall sensors are used for proximity switching, positioning, speed detection, and current sensing applications.

Hall sensors can be categorized into linear (analog) Hall sensors and switch Hall sensors. A switch Hall sensor consists of voltage regulator, Hall element, differential amplifier, Schmitt trigger, and output terminal and it outputs digital values. A linear Hall sensor consists of Hall element, linear amplifier, and emitter follower and it outputs analog values. If you add a comparator to a linear (analog) Hall sensor it will be able to output both analog and digital signals.

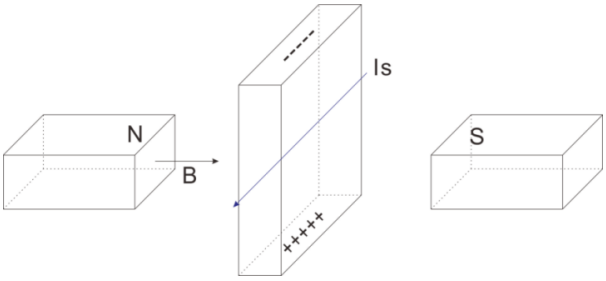
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Analog Hall Switch module
- 1 * Dual-color LED module
- 1 * Switch hall module
- 1 * PCF8591
- 2 * 3-Pin anti-reverse cable
- 1 * 4-Pin anti-reverse cable
- Several Jumper wires

Experimental Principles

Hall Effect

Hall Effect is a kind of electromagnetic effect. It was discovered by Edwin Hall in 1879 when he was researching conductive mechanism about metals. The effect is seen when a conductor is passed through a uniform magnetic field. The natural electron drift of the charge carriers causes the magnetic field to apply a Lorentz force (the force exerted on a charged particle in an electromagnetic field) to these charge carriers. The result is what is seen as a charge separation, with a buildup of either positive or negative charges on the bottom or on the top of the plate.

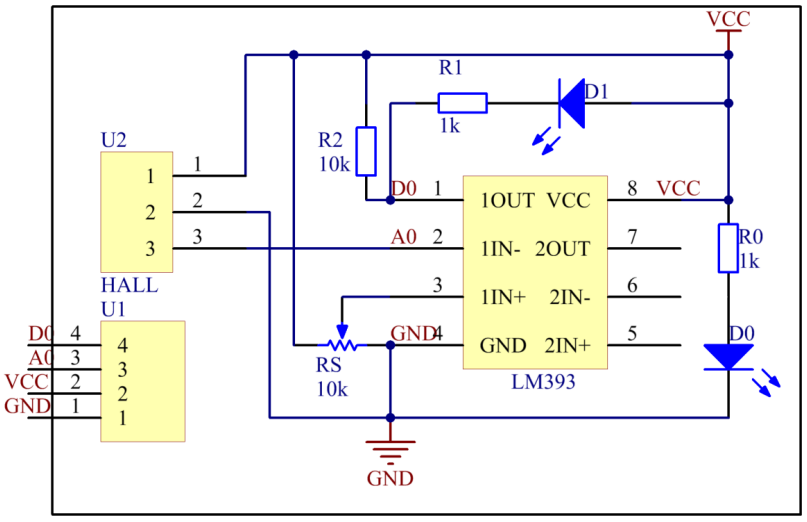


Hall sensor

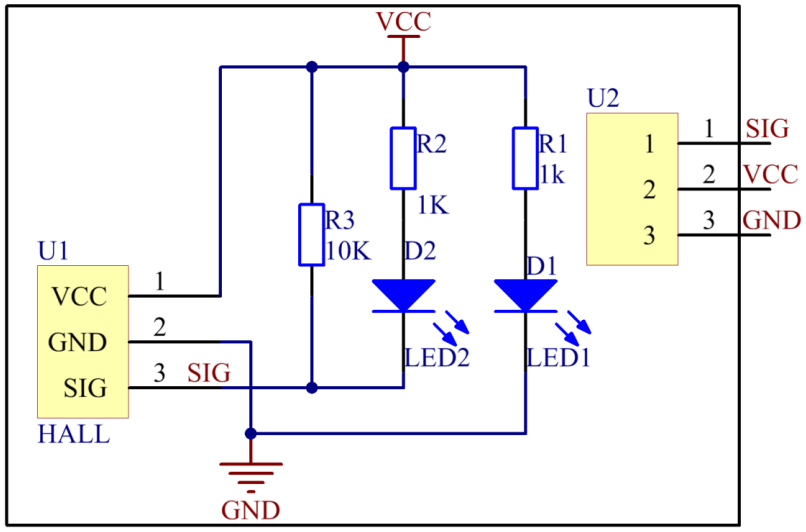
A Hall sensor is a kind of magnetic field sensor based on it.

Electricity carried through a conductor will produce a magnetic field that varies with current, and a Hall sensor can be used to measure the current without interrupting the circuit. Typically, the sensor is integrated with a wound core or permanent magnet that surrounds the conductor to be measured.

The schematic diagram of the analog Hall sensor module:



The schematic diagram of the Switch hall module:



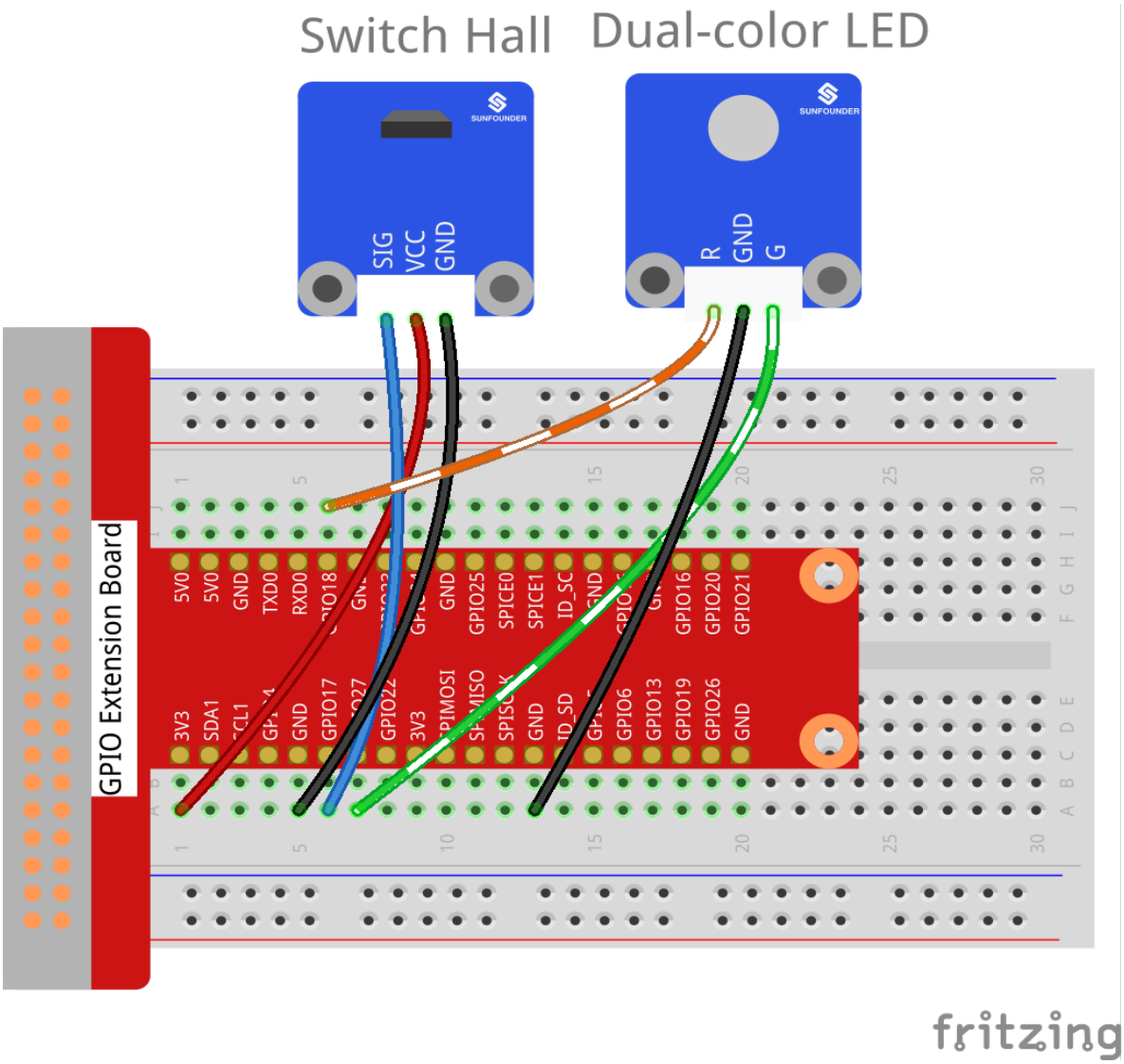
Experimental Procedures

For switch Hall sensor, take the following steps.

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Switch Hall Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/17_switch_hall/
```

Step 3: Compile.

```
gcc switch_hall.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define HallPin             0
#define Gpin                2
#define Rpin                1

void LED(char* color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == "RED")
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == "GREEN")
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

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

    pinMode(HallPin, INPUT);
    LED("GREEN");

    while(1){
        if(0 == digitalRead(HallPin)){
            delay(10);
            if(0 == digitalRead(HallPin)){
                LED("RED");
                printf("Detected magnetic materials \n");
            }
        }
        else if(1 == digitalRead(HallPin)){
            delay(10);
            if(1 == digitalRead(HallPin)){
                while(!digitalRead(HallPin));
                LED("GREEN");
            }
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

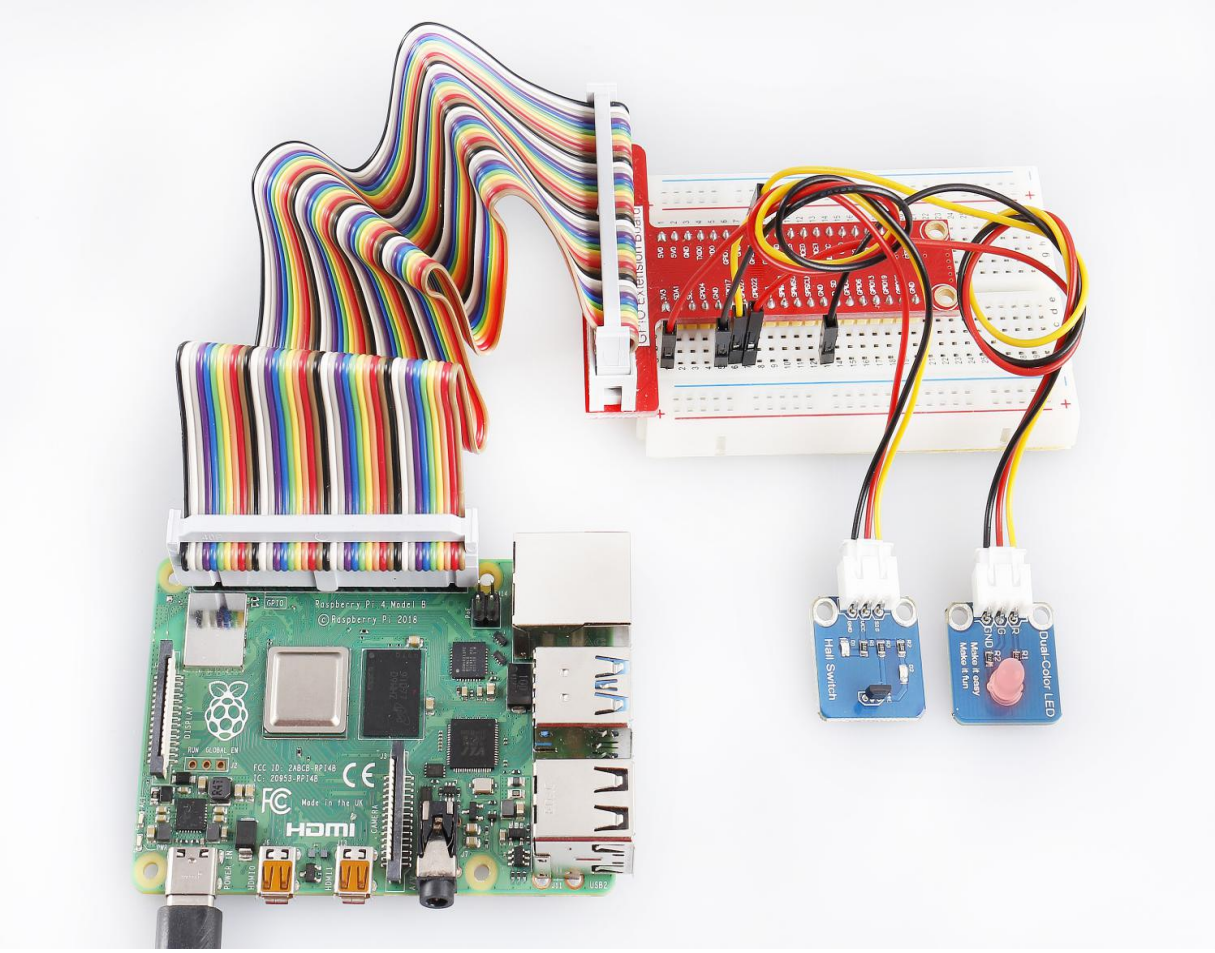
Step 3: Run.

```
sudo python3 17_switch_hall.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

HallPin = 11
Gpin   = 13
Rpin   = 12

def setup():
    GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)     # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)     # Set Red Led Pin mode to output
    GPIO.setup(HallPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode is␣
↪input, and pull up to high level(3.3V)
    GPIO.add_event_detect(HallPin, GPIO.BOTH, callback=detect, bouncetime=200)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)

def Print(x):
    if x == 0:
        print ('    **********************************')
        print ('    *   Detected magnetic materials   *')
        print ('    **********************************')

def detect(chn):
    Led(GPIO.input(HallPin))
    Print(GPIO.input(HallPin))

def loop():
    while True:
        pass

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    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()
```

Put a magnet close to the Switch Hall sensor. Then a string "**Detected magnetic materials**" will be printed on the screen and the LED will light up.

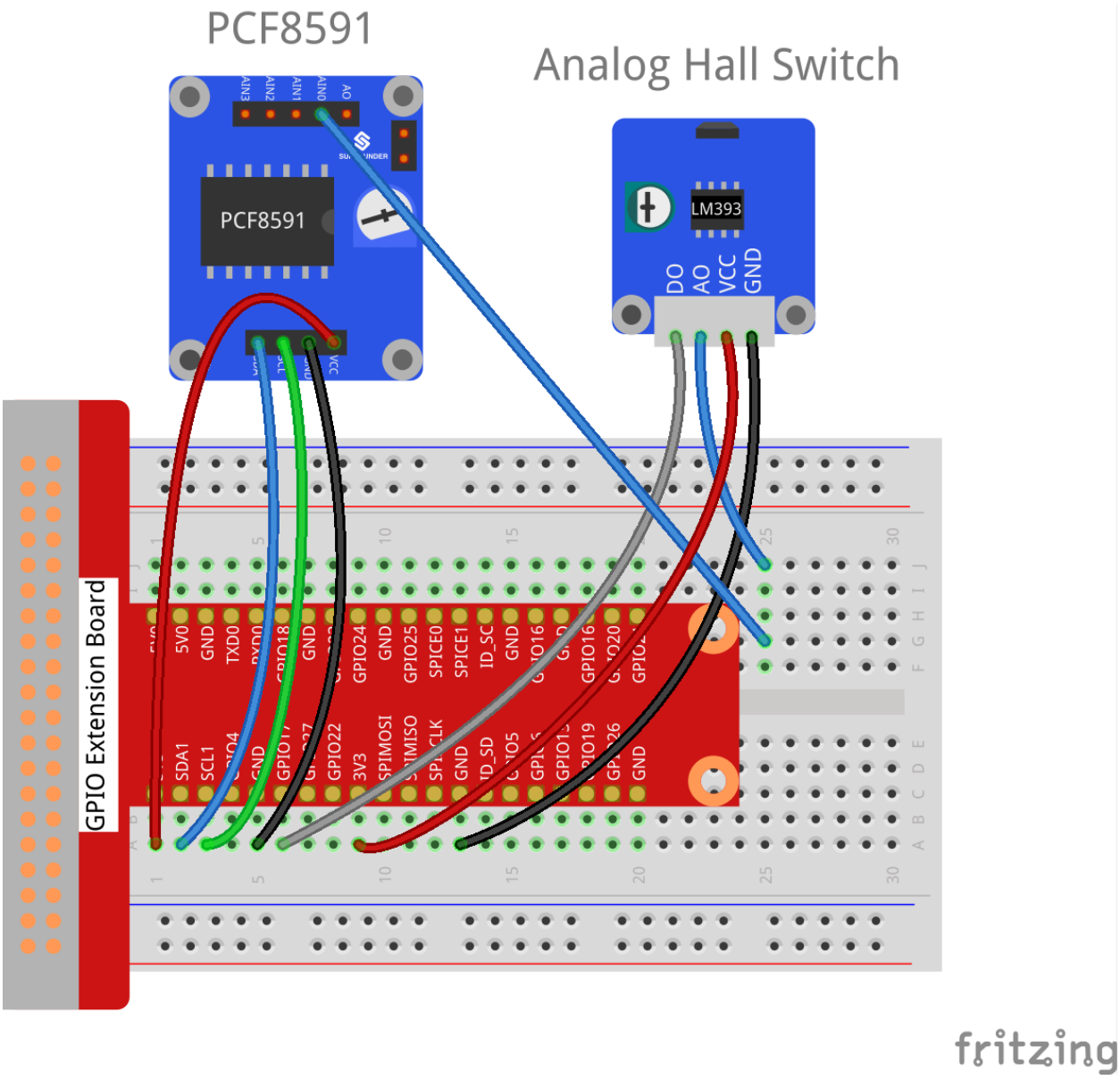


For **Analog Hall Switch**, take the following steps.

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Analog Hall Switch	GPIO Extension Board	PCF8591 module
DO	GPIO17	*
AO	*	AIN0
VCC	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/17_analog_hall_switch/
```

Step 3: Compile.

```
gcc analog_hall_switch.c -lwiringPi
```

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define PCF       120

int main (void)
{
    int res, tmp, status;
    wiringPiSetup ();
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup (PCF, 0x48);
    status = 0;
    while(1) // loop forever
    {
        res = analogRead(PCF + 0);
        printf("Current intensity of magnetic field : %d\n", res);
        if (res - 133 < 5 || res - 133 > -5)
            tmp = 0;
        if (res < 128) tmp = -1;
        if (res > 138) tmp =  1;
        if (tmp != status)
        {
            switch(tmp)
            {
                case 0:
                    printf("\n*****************\n"  );
                    printf(  "* Magnet: None. *\n"  );
                    printf(  "*****************\n\n");
                    break;
                case -1:
                    printf("\n******************\n"  );
                    printf(  "* Magnet: North. *\n"  );
                    printf(  "******************\n\n");
                    break;
                case 1:
                    printf("\n******************\n"  );
                    printf(  "* Magnet: South. *\n"  );
                    printf(  "******************\n\n");
                    break;
            }
            status = tmp;
        }
        delay (200);
    }
    return 0 ;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

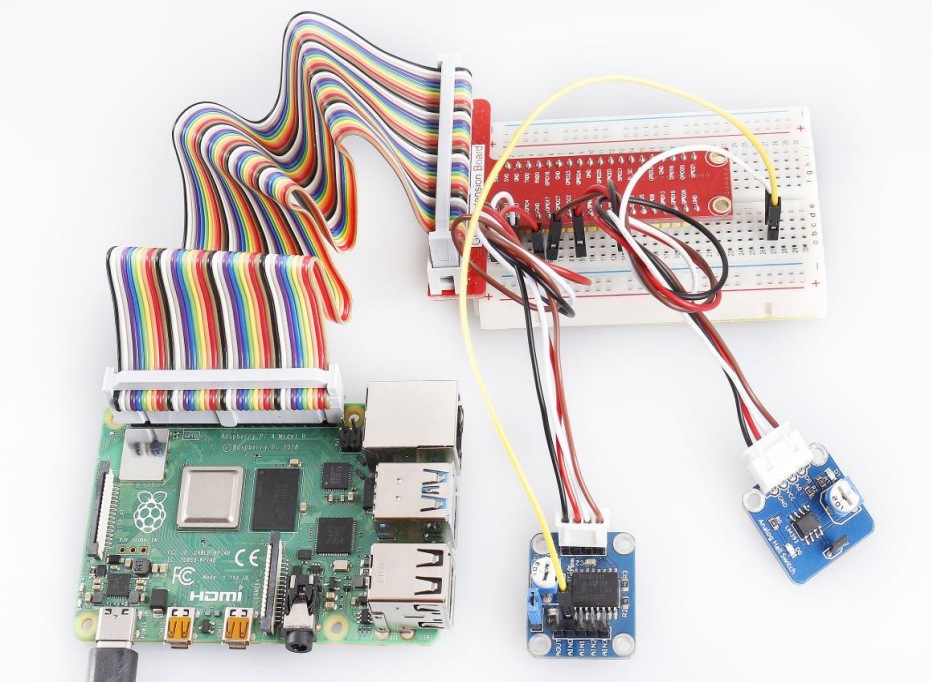
```
sudo python3 17_analog_hall_switch.py
```

Code

```
#/usr/bin/env python3
import RPi.GPIO as GPIO
import PCF8591 as ADC
import time

def setup():
    ADC.setup(0x48)

def Print(x):
    if x == 0:
        print ('')
        print ('*************')
        print ('* No Magnet *')
        print ('*************')
        print ('')
    if x == 1:
        print ('')
        print ('****************')
        print ('* Magnet North *')
        print ('****************')
        print ('')
    if x == -1:
        print ('')
        print ('****************')
        print ('* Magnet South *')
        print ('****************')
        print ('')

def loop():
    status = 0
    while True:
        res = ADC.read(0)
        print ('Current intensity of magnetic field : ', res)
        if res - 133 < 5 and res - 133 > -5:
            tmp = 0
        if res < 128:
            tmp = -1
        if res > 138:
            tmp = 1
        if tmp != status:
            Print(tmp)
            status = tmp
        time.sleep(0.2)

if __name__ == '__main__':
    setup()
    loop()
```

Now “Current intensity of magnetic field : xxx ” will be displayed on the screen. Put the magnet close to the analog Hall sensor, with the north magnetic pole towards the sensor, and then ” Magnet: North.” will be displayed. Move the magnet away, and ” Magnet: None.” will be printed. If the magnet approaches the sensor with the south magnetic pole towards it, ” Magnet: South.” will be printed on the screen.

Note: Pin D0 of the Analog Hall Sensor will output “0” only when the south pole of the magnet approaches it, otherwise it will output “1”.

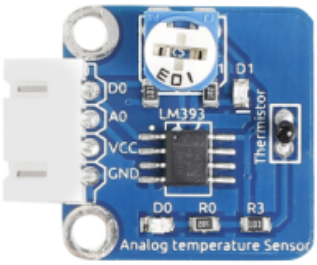


6.18 Lesson 18 Temperature Sensor

Introduction

A temperature sensor is a component that senses temperature and converts it into output signals. By material and component features, temperature sensors can be divided into two types: thermal resistor and thermocouple. Thermistor is one kind of the former type. It is made of semiconductor materials; most thermistors are negative temperature coefficient (NTC) ones, the resistance of which decreases with rising temperature. Since their resistance changes acutely with temperature changes, thermistors are the most sensitive temperature sensors.

There are two kinds of thermistor module in this kit (as shown below).



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Analog-temperature Sensor module
- 1 * Thermistor module
- 1 * PCF8591
- 1 * 3-Pin anti-reverse cable

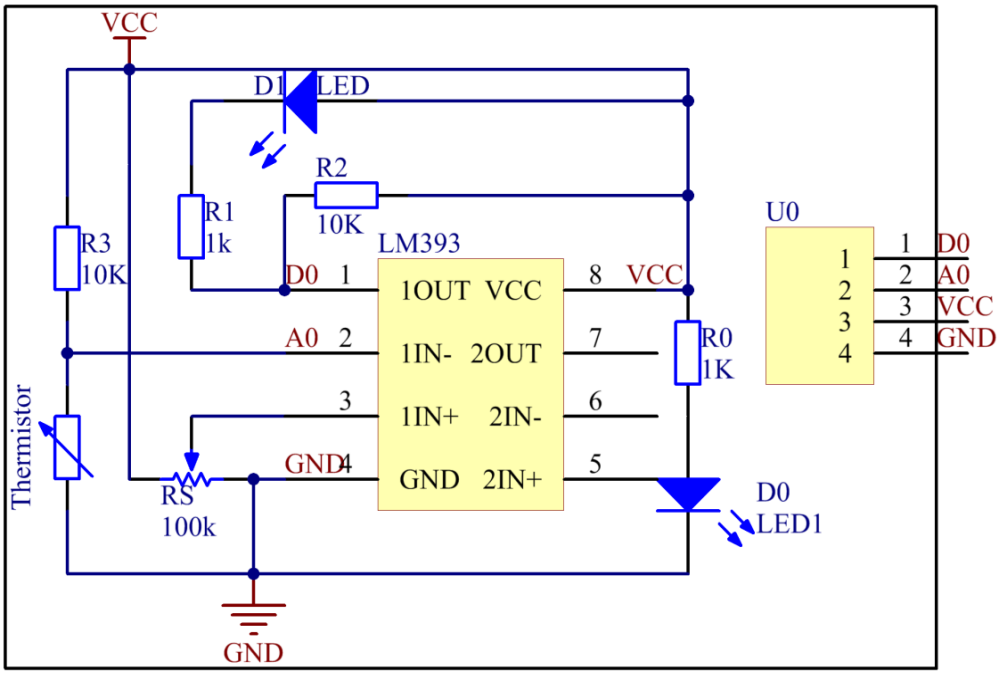
- 1 * 4-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

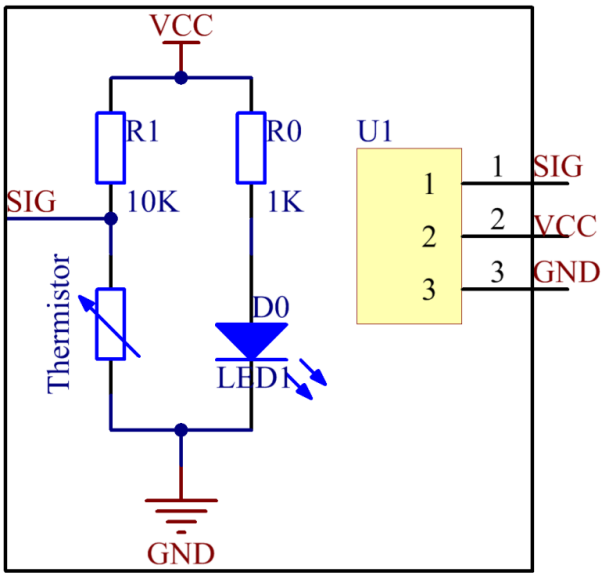
This module is based on the principle of the thermistor, whose resistance varies significantly with ambient temperature. When the ambient temperature increases, the resistance of the thermistor decreases; when decreases, it increases. It can detect surrounding temperature changes in a real-time manner.

In this experiment, we use an analog-digital converter PCF8591 to convert analog signals into digital ones.

The schematic diagram for analog temperature sensor:



The schematic diagram for the thermistor module:



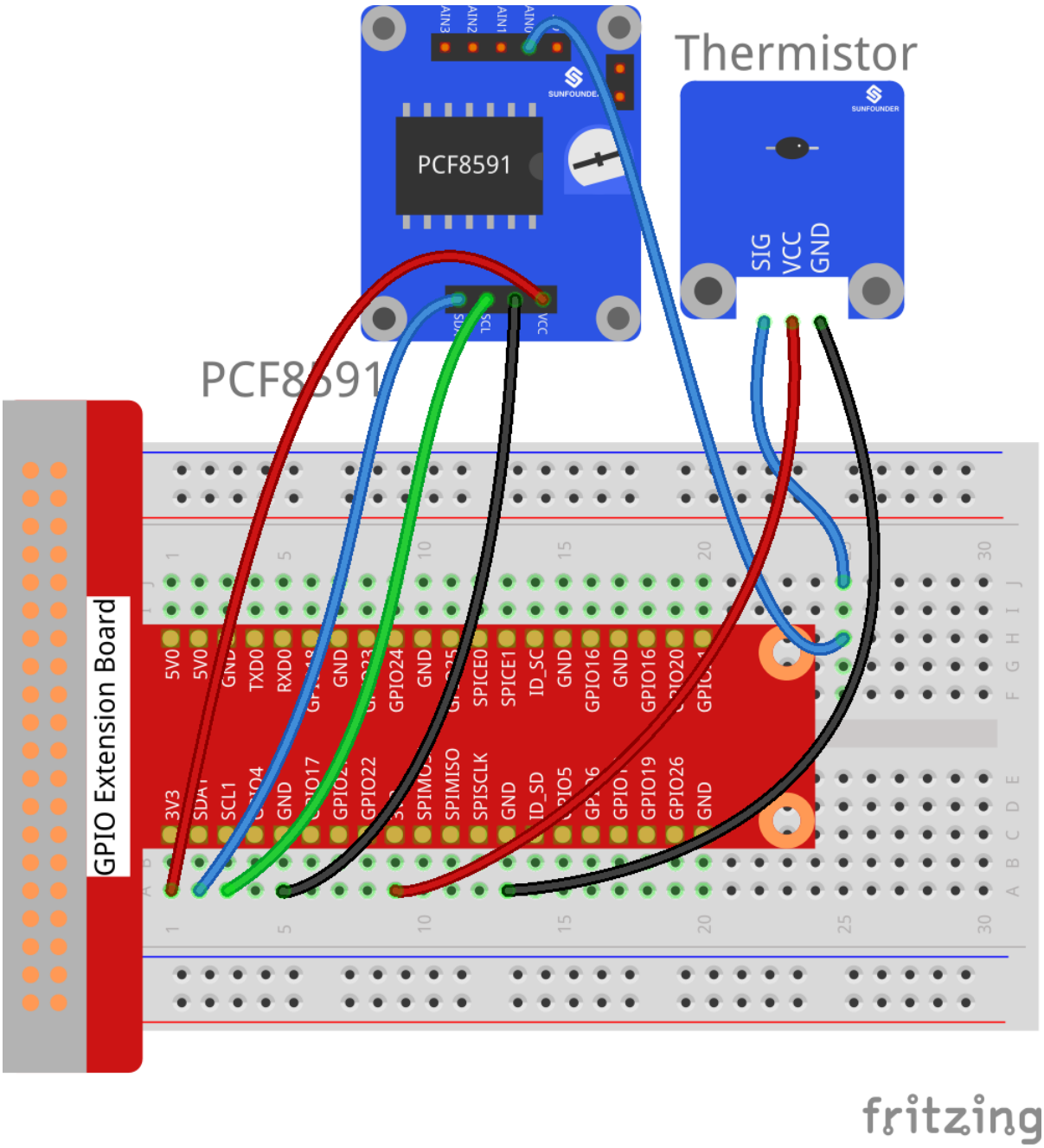
Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

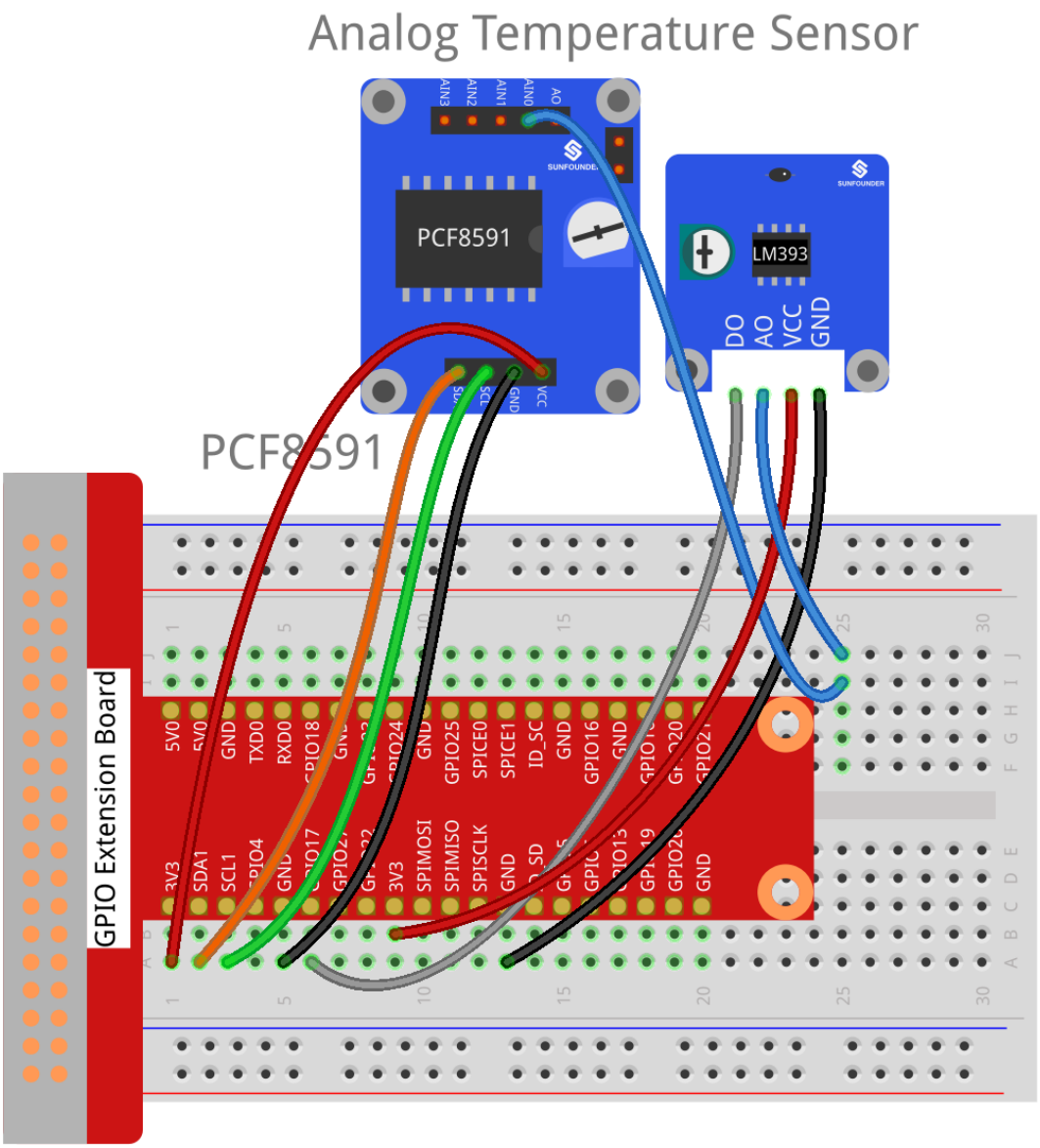
For thermistor module:

Thermistor Module	GPIO Extension Board	PCF8591 Module
SIG	*	AIN0
VCC	3V3	VCC
GND	GND	GND



For analog temperature sensor module

Analog Temperature Module	GPIO Extension Board	PCF8591 Module
DO	GPIO17	*
AO	*	AIN0
VCC	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/18_thermistor/
```

Step 3: Compile.

```
gcc thermistor.c -lwiringPi -lm
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define             PCF     120
#define             DOpin   0

void Print(int x)
{
    switch(x)
    {
        case 0:
            printf("\n************\n"  );
            printf(  "* Too Hot! *\n"  );
            printf(  "************\n\n");
        break;
        case 1:
            printf("\n**********\n"  );
            printf(  "* Better~ *\n"  );
            printf(  "**********\n\n");
        break;
        default:
            printf("\n********************\n"  );
            printf(  "* Print value error. *\n"  );
            printf(  "********************\n\n");
        break;
    }
}

int main()
{
    unsigned char analogVal;
    double Vr, Rt, temp;
    int tmp, status;

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup(PCF, 0x48);

    pinMode(DOpin, INPUT);

    status = 0;
    while(1) // loop forever
    {
        printf("loop");
        analogVal = analogRead(PCF + 0);
        Vr = 5 * (double)(analogVal) / 255;
        Rt = 10000 * (double)(Vr) / (5 - (double)(Vr));
        temp = 1 / (((log(Rt/10000)) / 3950)+(1 / (273.15 + 25)));
```

(continues on next page)

(continued from previous page)

```
        temp = temp - 273.15;
        printf("Current temperature : %lf\n", temp);

        // For a threshold, uncomment one of the code for
        // which module you use. DONOT UNCOMMENT BOTH!
        //-----------------------------------------
        // 1. For Analog Temperature module(with DO)
        tmp = digitalRead(DOpin);

        // 2. For Thermister module(with sig pin)
        // if (temp > 33) tmp = 0;
        // else if (temp < 31) tmp = 1;
        //-----------------------------------------

        if (tmp != status)
        {
            Print(tmp);
            status = tmp;
        }

        delay (200);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 18_thermistor.py
```

Code

```
#!/usr/bin/env python3
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time
import math

DO = 17
GPIO.setmode(GPIO.BCM)

def setup():
    ADC.setup(0x48)
    GPIO.setup(DO, GPIO.IN)

def Print(x):
    if x == 1:
        print ('')
        print ('***********')
        print ('* Better~ *')
        print ('***********')
        print ('')
```

(continues on next page)

(continued from previous page)

```
    if x == 0:
        print ('')
        print ('************')
        print ('* Too Hot! *')
        print ('************')
        print ('')

def loop():
    status = 1
    tmp = 1
    while True:
        analogVal = ADC.read(0)
        Vr = 5 * float(analogVal) / 255
        Rt = 10000 * Vr / (5 - Vr)
        temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
        temp = temp - 273.15
        print ('temperature = ', temp, 'C')

        # For a threshold, uncomment one of the code for
        # which module you use. DONOT UNCOMMENT BOTH!
        ##################################################
        # 1. For Analog Temperature module(with DO)
        tmp = GPIO.input(DO)
        #
        # 2. For Thermister module(with sig pin)
        #if temp > 33:
        #   tmp = 0
        #elif temp < 31:
        #   tmp = 1
        ##################################################

        if tmp != status:
            Print(tmp)
            status = tmp

        time.sleep(0.2)

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        pass
```

Now touch the thermistor and you can see the value of current temperature printed on the screen change accordingly.

Temperature alarm setting:

If you use the **Analog Temperature Sensor** module, uncomment the line under **1**:

For C language:

```
// For a threshold, uncomment one of the code for
// which module you use. DONOT UNCOMMENT BOTH!
//----------------------------------------
// 1. For Analog Temperature module(with DO)
tmp = digitalRead(DO);
```

(continues on next page)

(continued from previous page)

```
// 2. For Thermister module(with sig pin)
// if (temp > 33) tmp = 0;
// else if (temp < 31) tmp = 1;
```

For Python

```
################################################
# 1. For Analog Temperature module(with DO)
tmp = GPIO.input(DO);

# 2. For Thermister module(with sig pin)
#if temp > 33:
# tmp = 0;
#elif temp < 31:
# tmp = 1;
################################################
```

If you use the **Thermistor module**, uncomment the lIne under **2**:

For C language:

```
// For a threshold, uncomment one of the code for
// which module you use. DONOT UNCOMMENT BOTH!
//-----------------------------------------
// 1. For Analog Temperature module(with DO)
// tmp = digitalRead(DO);

// 2. For Thermister module(with sig pin)
if (temp > 33) tmp = 0;
else if (temp < 31) tmp = 1;
//-----------------------------------------
```

For Python

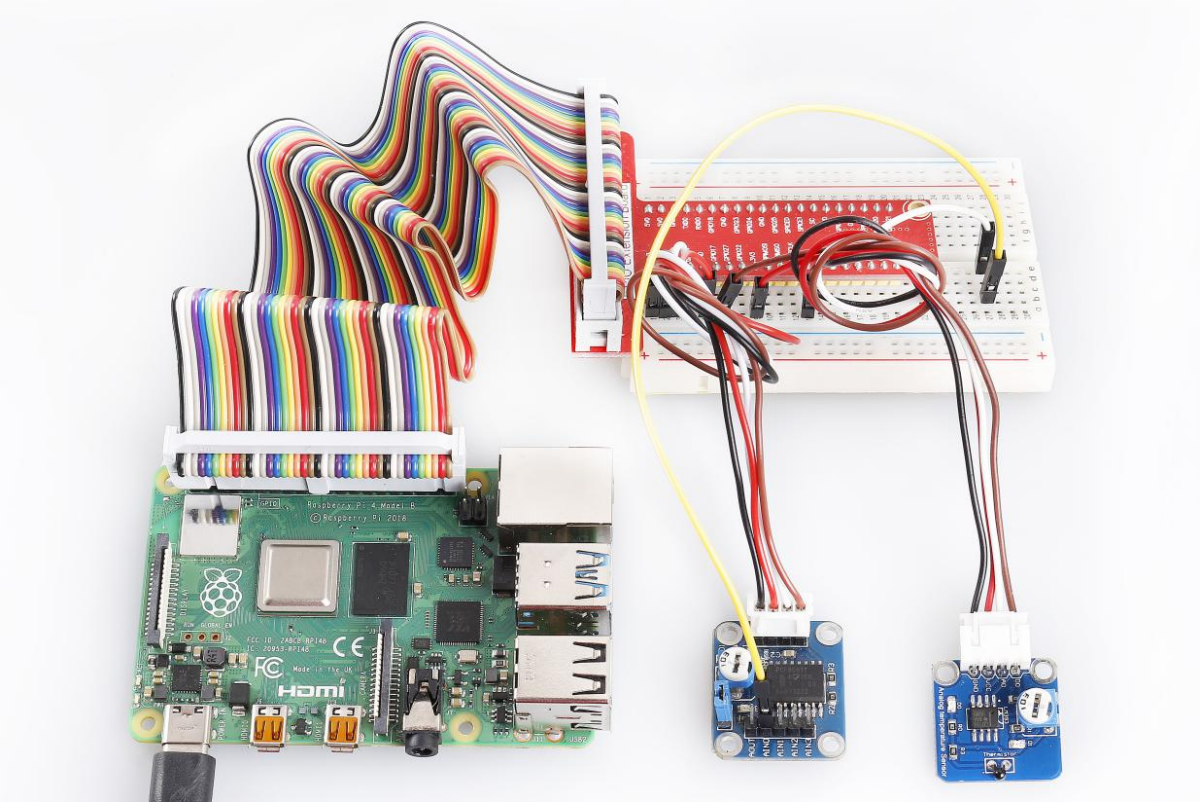
```
################################################
# 1. For Analog Temperature module(with DO)
#tmp = GPIO.input(DO);
#
# 2. For Thermister module(with sig pin)
if temp > 33:
    tmp = 0;
elif temp < 31:
    tmp = 1;
################################################
```

After editing the code, repeat step 2, 3, and 4 (or step 2, 3 for Python users).

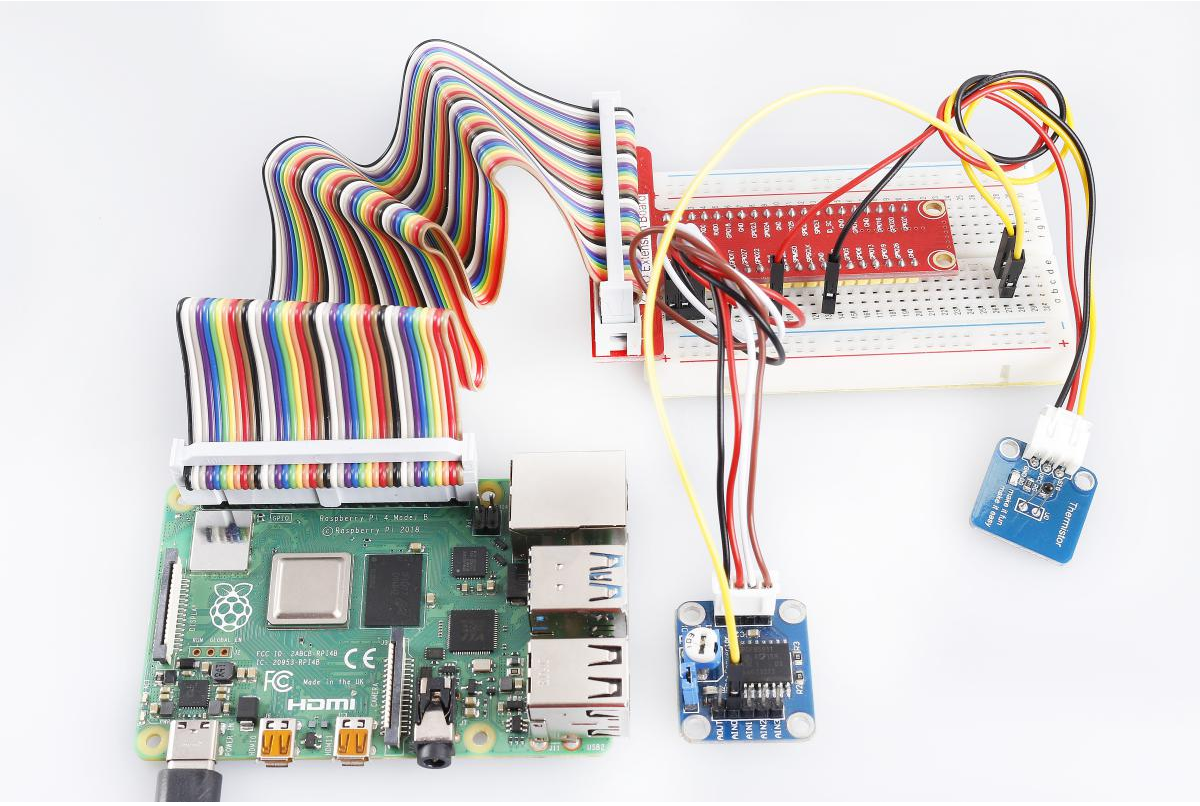
You can still see temperature value printed on the screen constantly. If you pinch the thermistor for a while, its temperature will rise slowly. “Too Hot!” will be printed on the screen. Release your fingers, and let it stay in the open air for a while, or blow on the module. When the temperature drops down slowly, “Better” will be printed.

Note: The analog temperature sensor adjusts alarm temperature by the potentiometer on the module. The thermistor changes the alarm temperature by program.

The physical picture for analog temperature sensor:



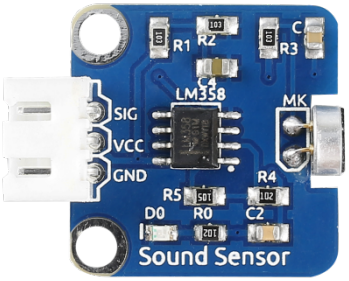
The physical picture for thermistor module:



6.19 Lesson 19 Sound Sensor

Introduction

Sound sensor is a component that receives sound waves and converts them into electrical signal. It detects the sound intensity in ambient environment like a microphone.



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * PCF8591
- 1 * Sound sensor module
- 1 * 3-Pin anti-reverse cable
- Several Jumper wires

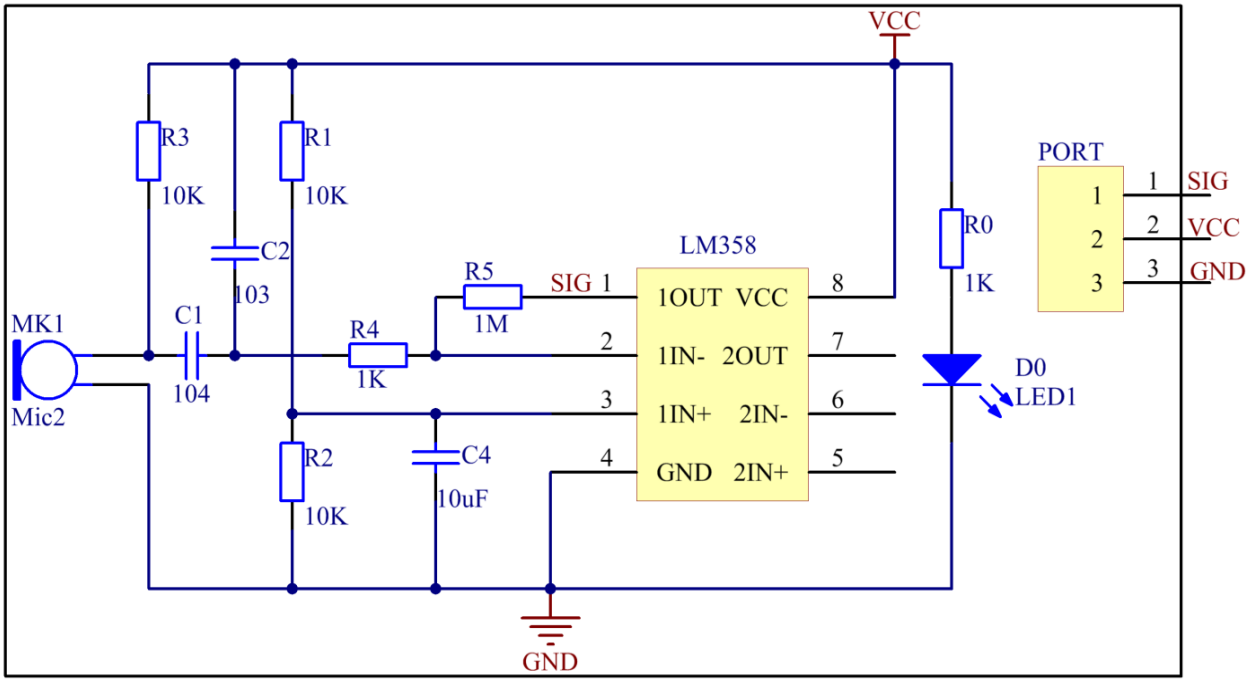
Experimental Principle

The microphone on the sensor module can convert audio signals into electrical signals (analog quantity), then convert analog quantity into digital quantity by PCF8591 and transfer them to MCU.

LM358 is a dual-channel operational amplifier. It contains two independent, high gain, and internally compensated amplifiers, but we will only use one of them in this experiment. The microphone transforms sound signals into electrical signals and then sends out the signals to pin 2 of LM358 and outputs them to pin 1 (that's, pin SIG of the module) via the external circuit. Then use PCF8591 to read analog values.

PCF8591 is an 8-bit resolution, 4-channel A/D1-channel D/A conversion chip. We connect the output terminal (SIG) to AIN0 of PCF8591 so as to detect the strength of voice signal in a real-time manner.

The schematic diagram of the module is as shown below:

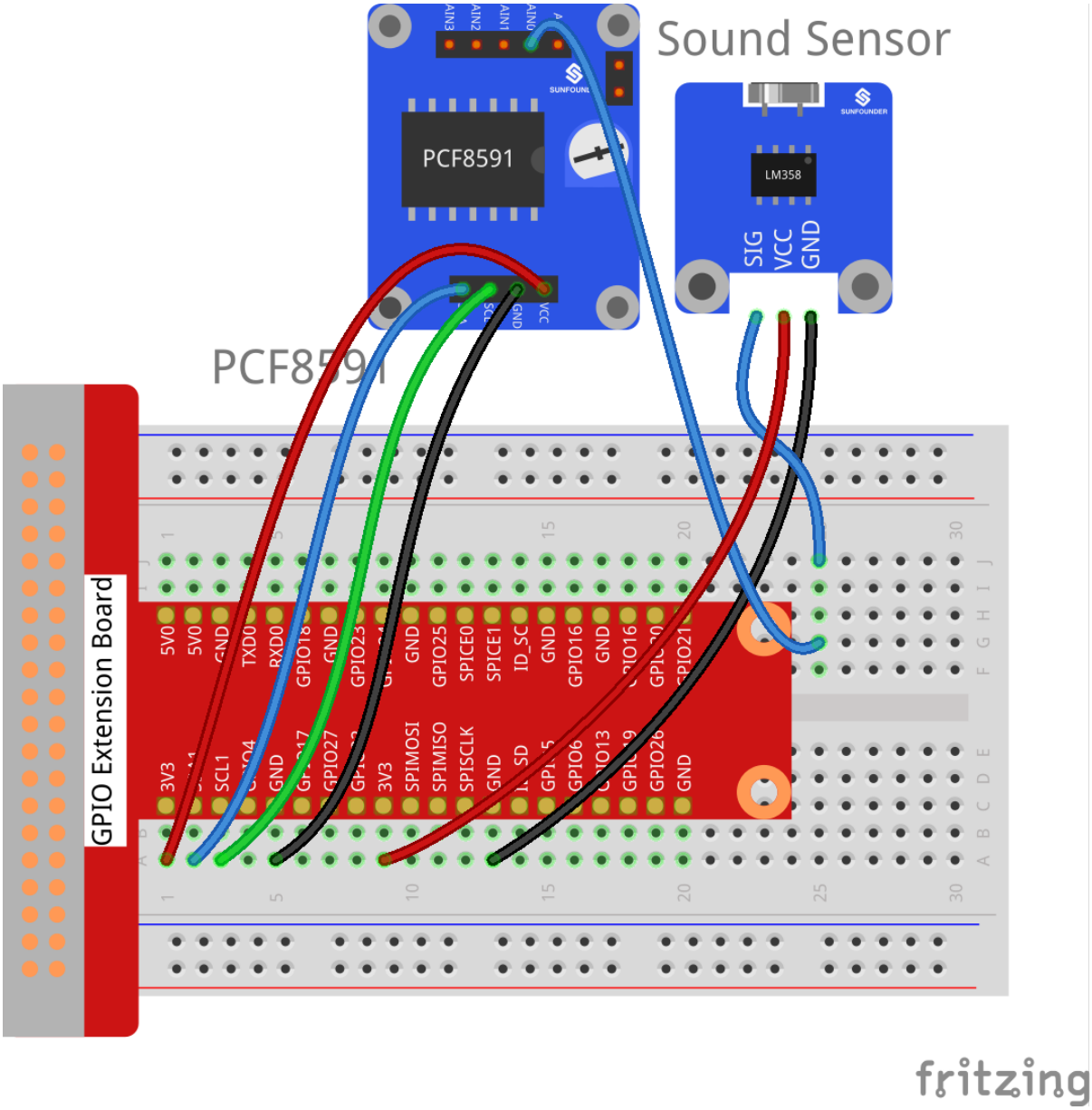


Experimental Procedures

Step 1: Build the circuit according to the following method.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Sound Sensor Module	GPIO Extension Board	PCF8591 Module
SIG	*	AIN0
VCC	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/19_sound_sensor/
```

Step 3: Compile.

```
gcc sound_sensor.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define PCF       120

int main (void)
{
    int value;
    int count = 0;
    wiringPiSetup ();
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup (PCF, 0x48);
    while(1) // loop forever
    {
        value = analogRead  (PCF + 0);
        printf("value: %d\n", value);
        if (value < 80){
            printf("Voice In!! \n");
        }
        delay(100);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 19_sound_sensor.py
```

Code

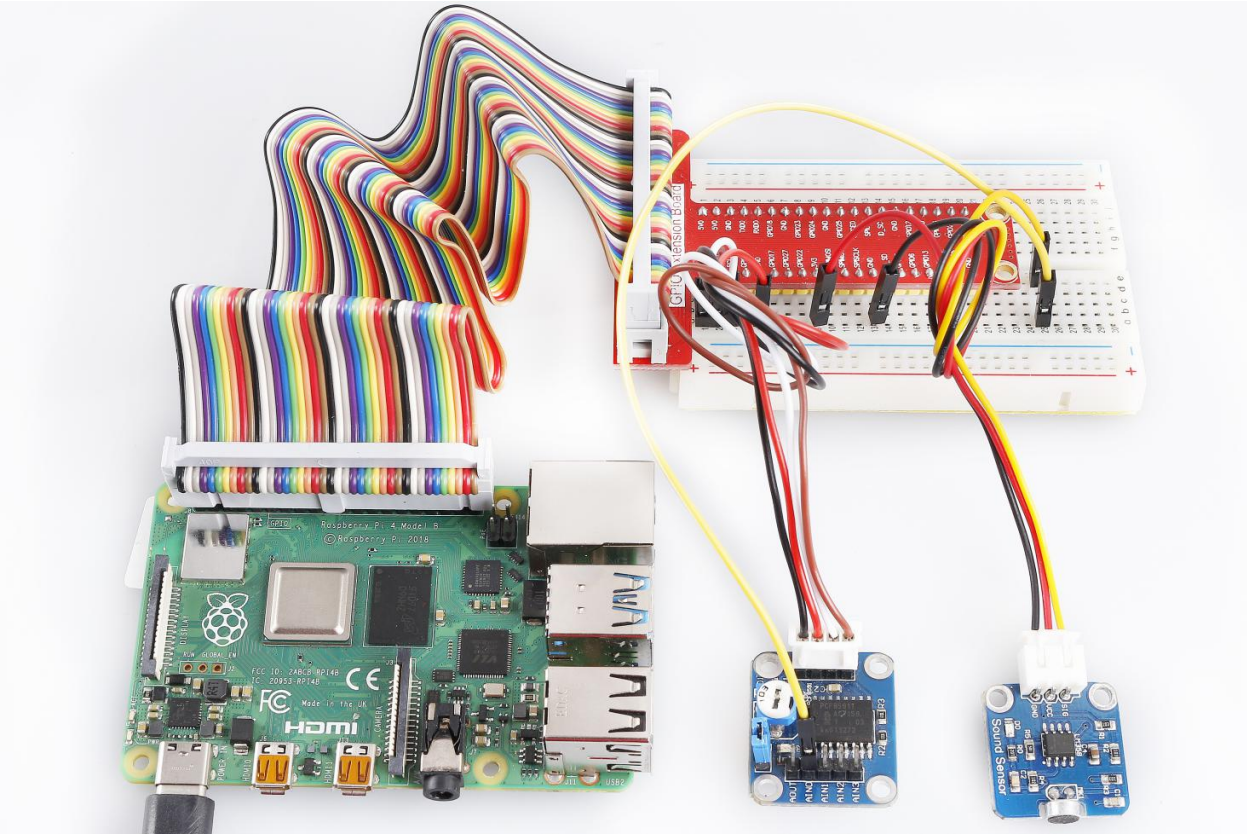
```
#!/usr/bin/env python3
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

def setup():
    ADC.setup(0x48)

def loop():
    count = 0
    while True:
        voiceValue = ADC.read(0)
        if voiceValue:
            print ("Value:", voiceValue)
            if voiceValue < 50:
```

(continues on next page)

(continued from previous page)

```
                print ("Voice In!! ", count)
                count += 1
            time.sleep(0.2)

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        pass
```

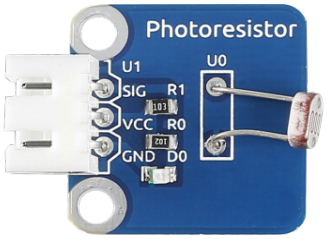
Now, speak close to or blow to the microphone, and you can see “Voice In!!” printed on the screen.



6.20 Lesson 20 Photoresistor Module

Introduction

A photoresistor is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity.

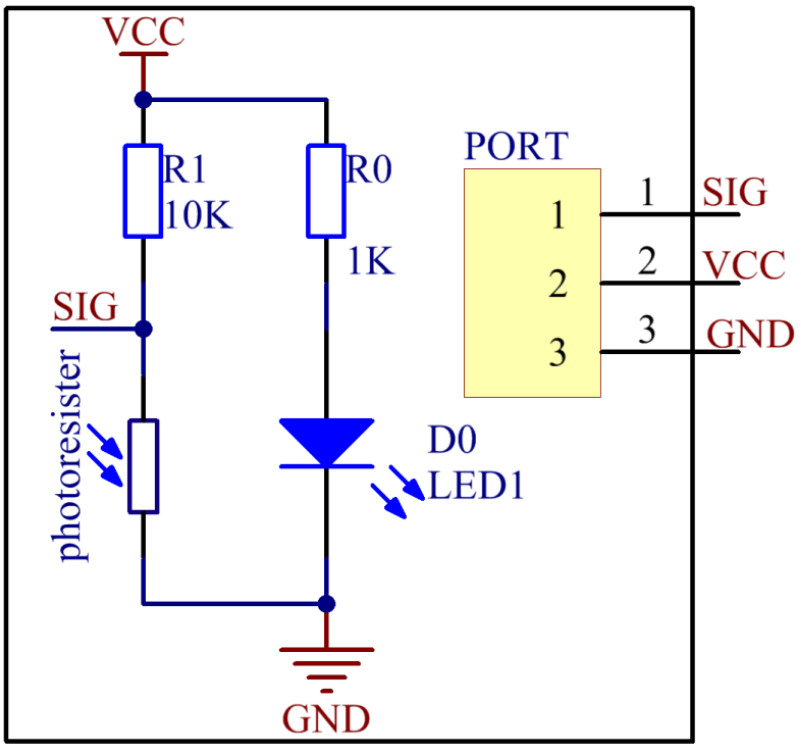


Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * PCF8591
- 1 * Photoresistor module
- 1 * 3-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

With light intensity increasing, the resistance of a photoresistor will decrease. Thus the output voltage changes. Analog signals collected by the photoresistor are converted to digital signals through PCF8591. Then these digital signals are transmitted to Raspberry Pi and printed on the screen. The schematic diagram:

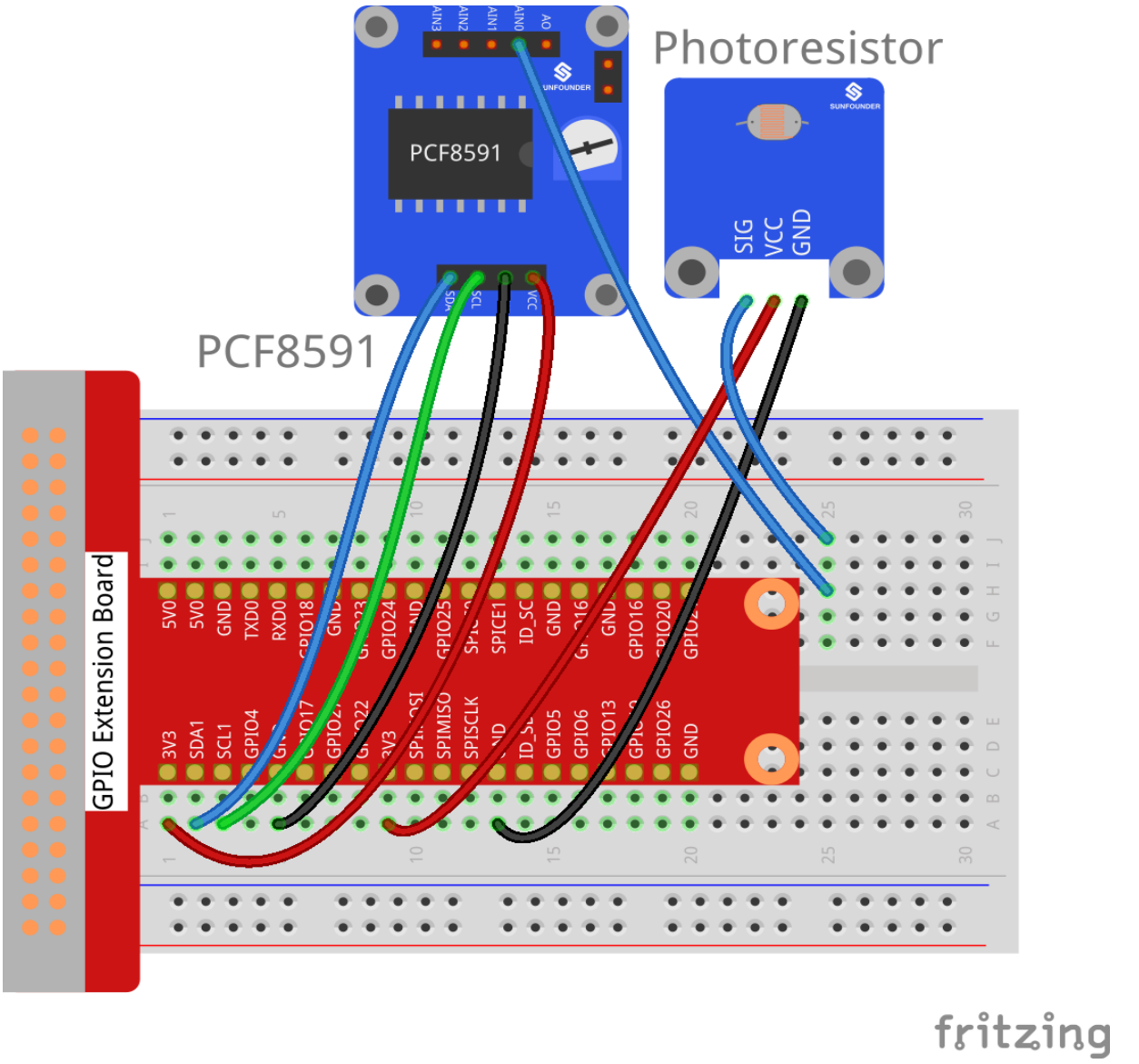


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Photoresistor	GPIO Extension Board	PCF8591 Module
SIG	*	AIN0
VCC	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/20_photoresistor/
```

Step 3: Compile.

```
gcc photoresistor.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define             PCF     120
#define             DOpin   0

int main()
{
    int analogVal;

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup(PCF, 0x48);

    while(1) // loop forever
    {
        analogVal = analogRead(PCF + 0);
        printf("Value: %d\n", analogVal);

        delay (200);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 20_photoresistor.py
```

Code

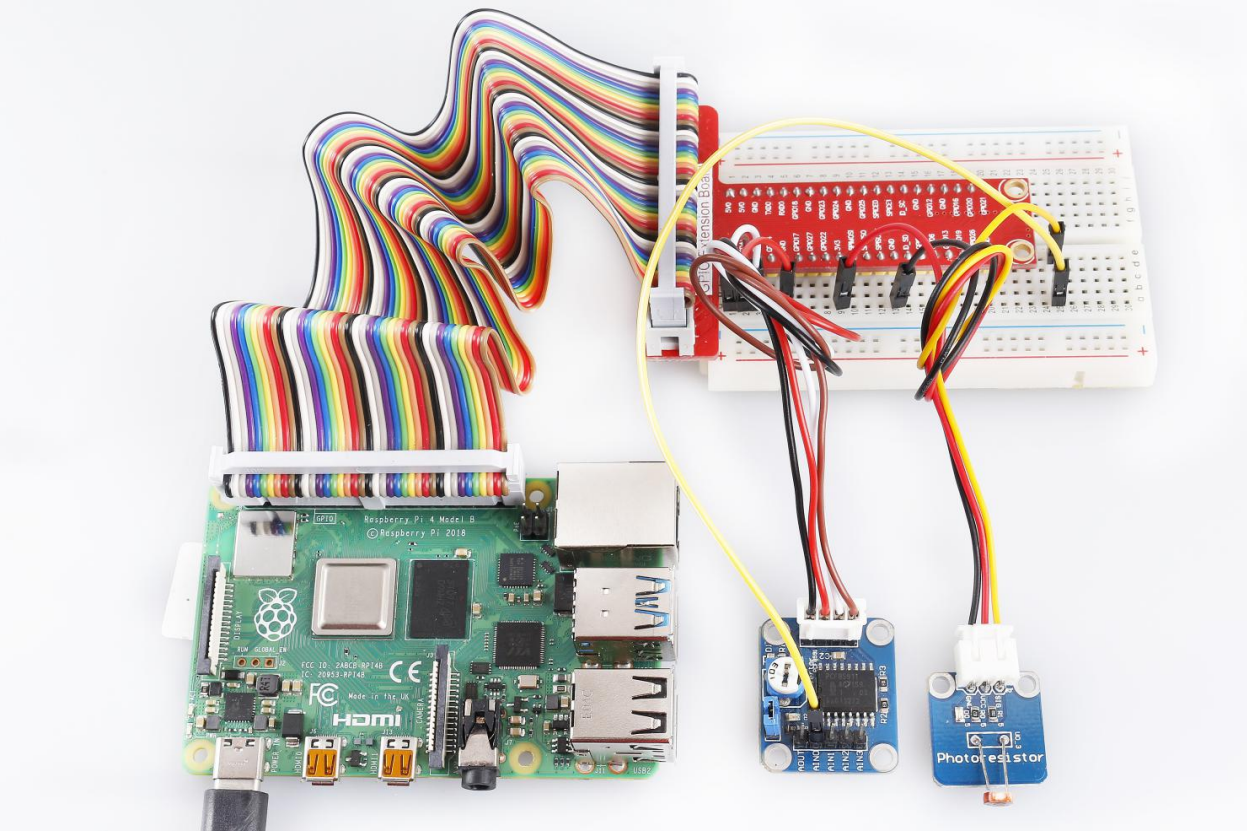
```
#!/usr/bin/env python3
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time

DO = 17
GPIO.setmode(GPIO.BCM)

def setup():
    ADC.setup(0x48)
    GPIO.setup(DO, GPIO.IN)


def loop():
    status = 1
```

(continues on next page)

(continued from previous page)

```
    while True:
        print ('Value: ', ADC.read(0))

        time.sleep(0.2)

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        pass
```

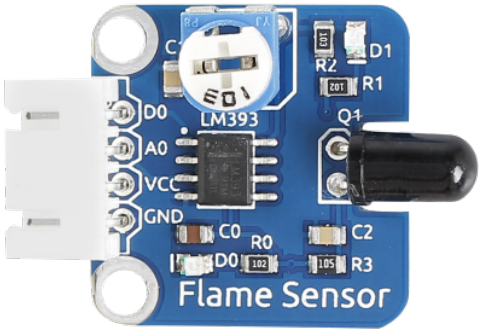
Now, change light intensity (e.g. cover the module with a pad), and the value printed on the screen will change accordingly.



6.21 Lesson 21 Flame Sensor

Introduction

A flame sensor (as shown below) performs detection by capturing infrared rays with specific wavelengths from flame. It can be used to detect and warn of flames.

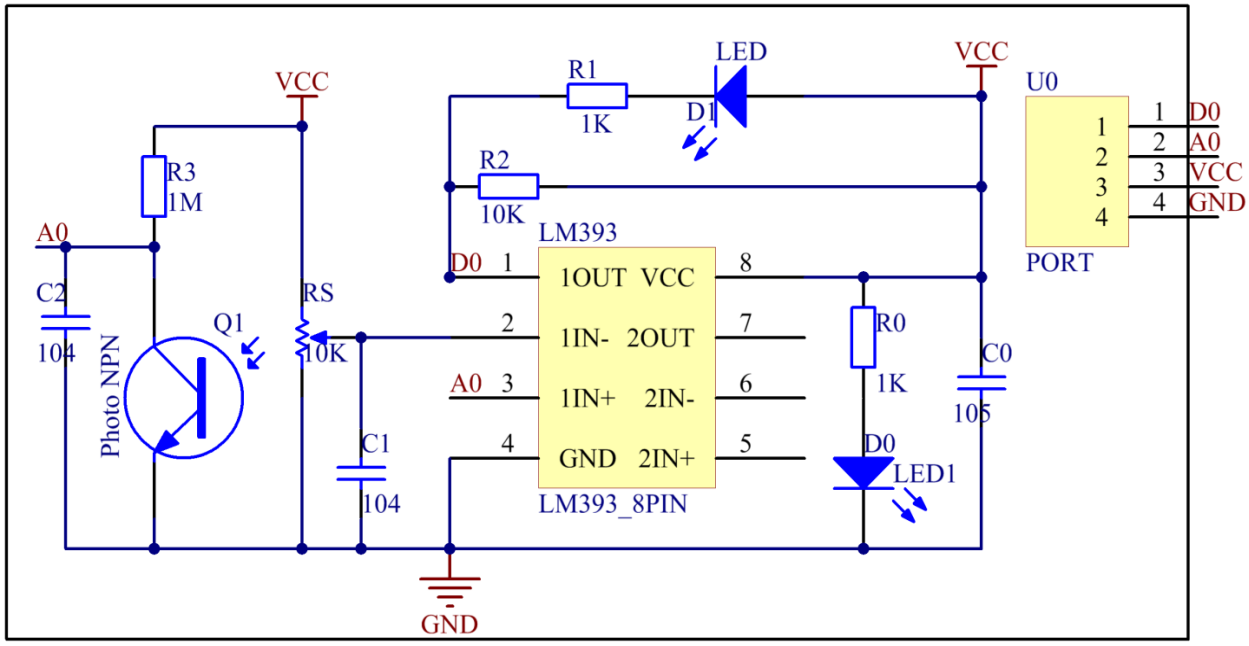


Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Flame sensor module
- 1 * PCF8591
- 1 * 4-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

There are several types of flame sensors. In this experiment, we will use a far-infrared flame sensor. It can detect infrared rays with wavelength ranging from 700nm to 1000nm. A far-infrared flame probe converts the strength changes of external infrared light into current changes. And then it convert analog quantities into digital ones. In this experiment, connect pin D0 of the Flame Sensor module to a GPIO of Raspberry Pi to detect by programming whether any flame exists. The schematic diagram:

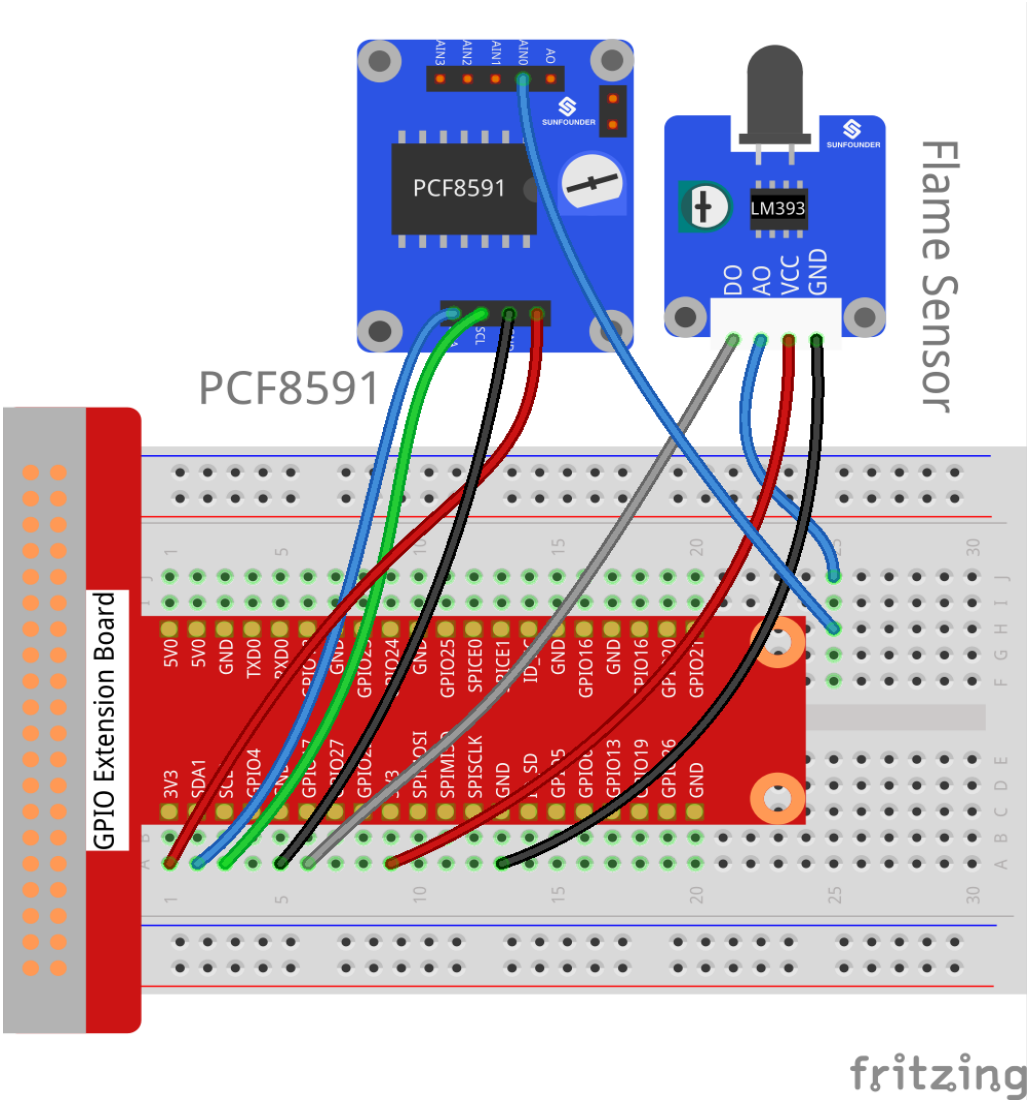


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Flame Sensor	GPIO Extension Board	PCF8591 Module
DO	GPIO17	*
AO	*	AIN0
VCC	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/21_flame_sensor/
```

Step 3: Compile.

```
gcc flame_sensor.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define             PCF     120
#define             DOpin   0

void Print(int x)
{
    switch(x)
    {
        case 1:
            printf("\n*********\n"  );
            printf(  "* Saft~ *\n"  );
            printf(  "*********\n\n");
        break;
        case 0:
            printf("\n*********\n"  );
            printf(  "* Fire! *\n"  );
            printf(  "*********\n\n");
        break;
        default:
            printf("\n**********************\n"  );
            printf(  "* Print value error. *\n"  );
            printf(  "**********************\n\n");
        break;
    }
}

int main()
{
    int analogVal;
    int tmp, status;

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    // Setup pcf8591 on base pin 120, and address 0x48
```

(continues on next page)

(continued from previous page)

```
    pcf8591Setup(PCF, 0x48);

    pinMode(DOpin, INPUT);

    status = 0;
    while(1) // loop forever
    {
        analogVal = analogRead(PCF + 0);
        printf("%d\n", analogVal);

        tmp = digitalRead(DOpin);

        if (tmp != status)
        {
            Print(tmp);
            status = tmp;
        }

        delay (200);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 21_flame_sensor.py
```

Code

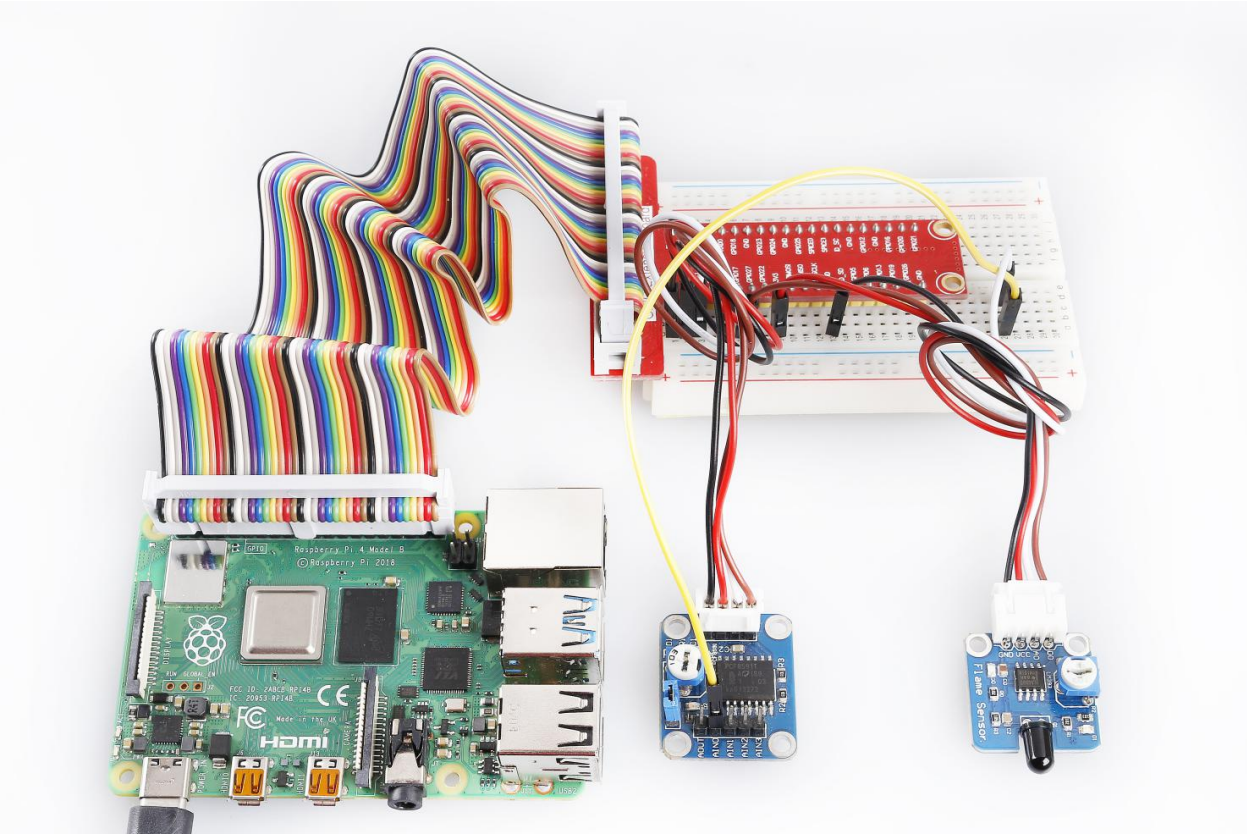
```
#!/usr/bin/env python3
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time
import math

DO = 17
GPIO.setmode(GPIO.BCM)

def setup():
    ADC.setup(0x48)
    GPIO.setup(DO, GPIO.IN)

def Print(x):
    if x == 1:
        print ('')
        print ('    *********')
        print ('    * Safe~ *')
        print ('    *********')
        print ('')
    if x == 0:
        print ('')
```

(continues on next page)

(continued from previous page)

```
        print ('    *********')
        print ('    * Fire! *')
        print ('    *********')
        print ('')

def loop():
    status = 1
    while True:
        print (ADC.read(0))

        tmp = GPIO.input(DO);
        if tmp != status:
            Print(tmp)
            status = tmp

        time.sleep(0.2)

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        pass
```

Now, ignite a lighter near the sensor, within the range of 80cm, and “Fire!” will be displayed on the screen. If you put out the lighter or just move the flames away from the flame sensor, “Safe~” will be displayed then.



6.22 Lesson 22 Gas Sensor

Introduction

Gas Sensor MQ-2 is a sensor for flammable gas and smoke by detecting the concentration of combustible gas in the air. They are used in gas detecting equipment for smoke and flammable gasses in household, industry or automobile.

Required Components

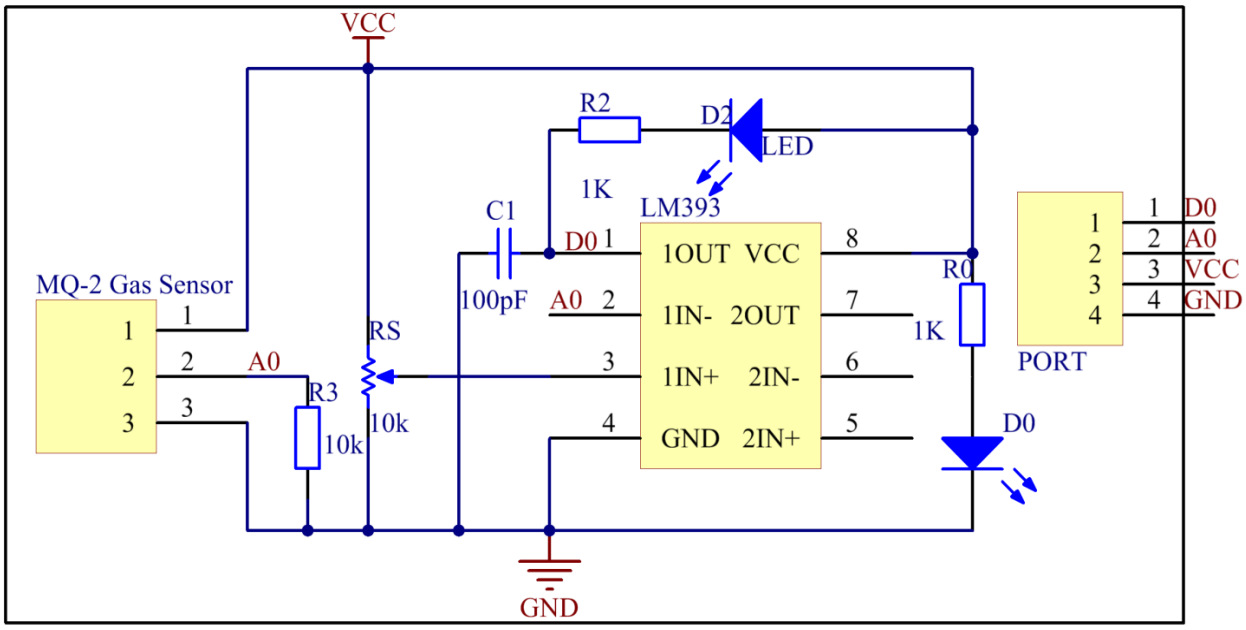
- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Active Buzzer module
- 1 * PCF8591
- 1 * Gas sensor module
- 1 * 3-Pin anti-reverse cable
- 1 * 4-Pin anti-reverse cable
- Several Jumper wires

Experimental Principle

MQ-2 gas sensor is a kind of surface ion type and N-type semiconductors, which uses tin oxide semiconductor gas sensitive material. When ambient temperature is in 200 ~ 300°C, tin oxide will adsorb oxygen in the air and form oxygen anion adsorption to decrease electron density in semiconductor so as to increase its resistance. When in contact with the smoke, if grain boundary barrier is modulated by the smoke and changed, it could cause surface conductivity change. So you can gain the information of the smoke existence, The higher the smoke concentration is, the more conductive the material becomes, thus the lower the output resistance is.

In this experiment, if harmful gases reach a certain concentration, the buzzer will beep to warn.

The schematic diagram of the module is as shown below:



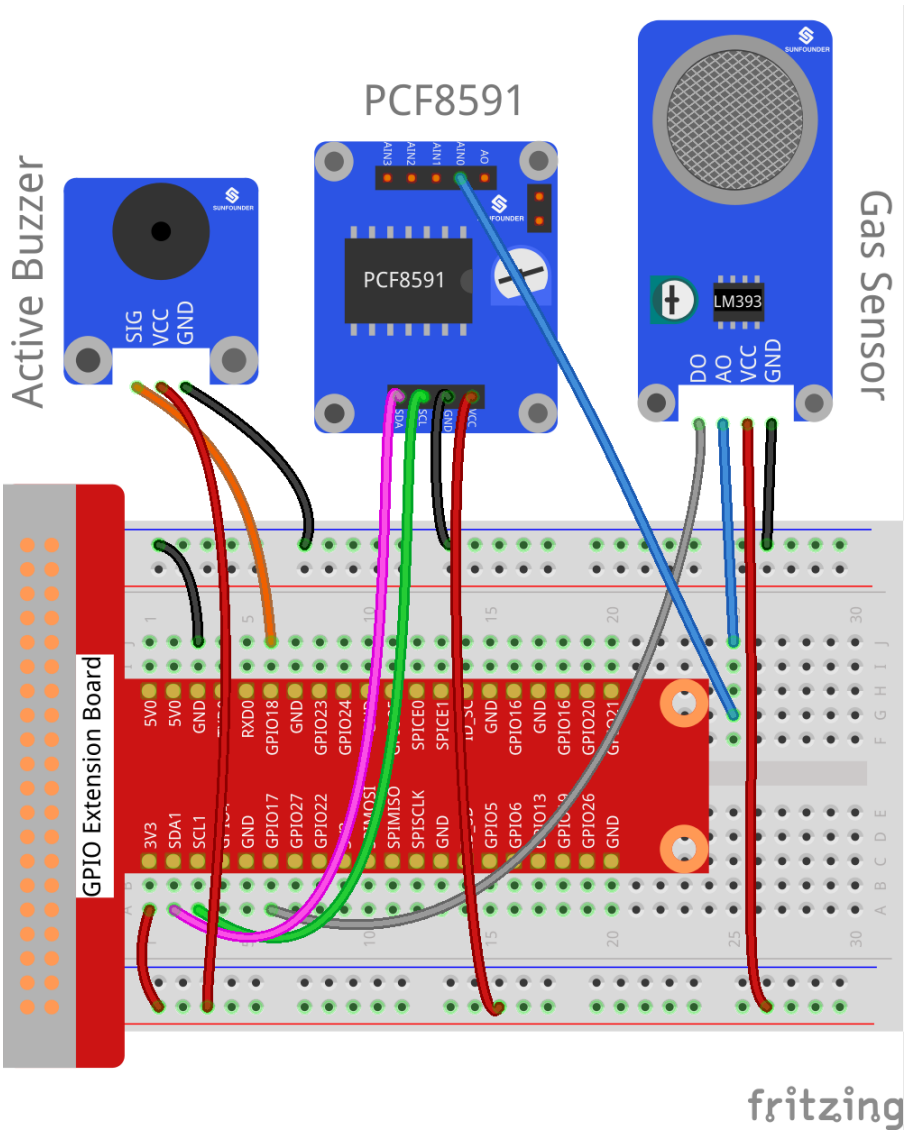
Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Gas Sensor Module	GPIO Extension Board	PCF8591 Module
DO	GPIO17	*
AO	*	AIN0
VCC	3V3	*
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Active Buzzer Module
GPIO1	GPIO18	SIG
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/22_gas_sensor/
```

Step 3: Compile.

```
gcc gas_sensor.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define             PCF     120
#define             DOpin   0
#define             Buzz    1

void Print(int x)
{
    switch(x)
    {
        case 1:
            printf("\n*********\n"  );
            printf(  "* Saft~ *\n"  );
            printf(  "*********\n\n");
        break;
        case 0:
            printf("\n***************\n"  );
            printf(  "* Danger Gas! *\n"  );
            printf(  "***************\n\n");
        break;
        default:
            printf("\n**********************\n"  );
            printf(  "* Print value error. *\n"  );
            printf(  "**********************\n\n");
        break;
    }
}

int main()
{
    int analogVal;
    int tmp, status, count;

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    // Setup pcf8591 on base pin 120, and address 0x48
    pcf8591Setup(PCF, 0x48);

    pinMode (DOpin, INPUT);
    pinMode (Buzz,  OUTPUT);
    digitalWrite(Buzz, HIGH);

    status = 0;
    count = 0;
    while(1) // loop forever
    {
        analogVal = analogRead(PCF + 0);
        printf("%d\n", analogVal);

        tmp = digitalRead(DOpin);

        if (tmp != status)
```

(continues on next page)

(continued from previous page)

```
        {
            Print(tmp);
            status = tmp;
        }

        if (status == 0)
        {
            count ++;
            if (count % 2 == 0)
                {digitalWrite(Buzz, HIGH);}
            else
                {digitalWrite(Buzz, LOW);}
        }
        else
        {
            count = 0;
            digitalWrite(Buzz, HIGH);
        }
        delay (200);
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 22_gas_sensor.py
```

Code

```
#!/usr/bin/env python3
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time
import math

DO = 17
Buzz = 18
GPIO.setmode(GPIO.BCM)

def setup():
    ADC.setup(0x48)
    GPIO.setup      (DO,    GPIO.IN)
    GPIO.setup      (Buzz,  GPIO.OUT)
    GPIO.output     (Buzz,  1)

def Print(x):
    if x == 1:
        print ('')
        print ('    *********')
        print ('    * Safe~ *')
        print ('    *********')
```

(continues on next page)

(continued from previous page)

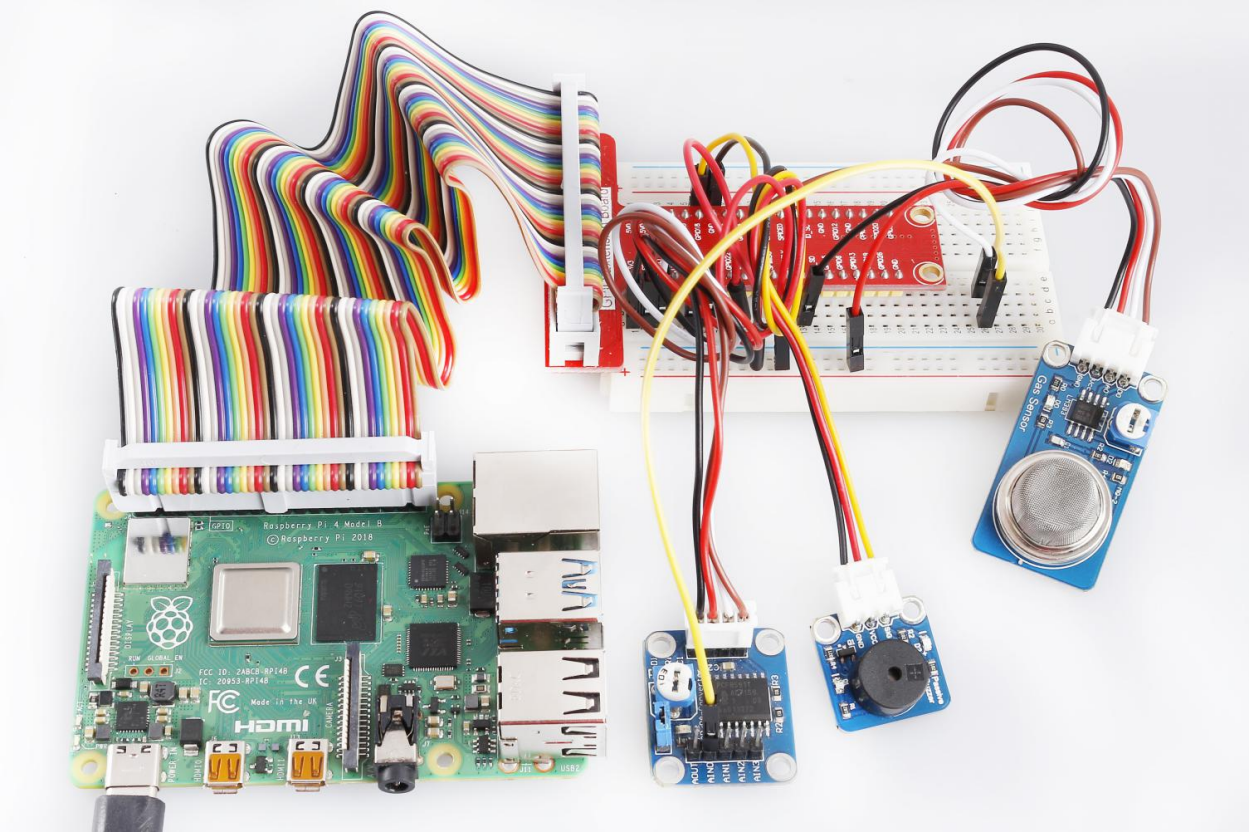
```
        print ('')
    if x == 0:
        print ('')
        print ('    ***************')
        print ('    * Danger Gas! *')
        print ('    ***************')
        print ('')

def loop():
    status = 1
    count = 0
    while True:
        print (ADC.read(0))

        tmp = GPIO.input(DO)
        if tmp != status:
            Print(tmp)
            status = tmp
        if status == 0:
            count += 1
            if count % 2 == 0:
                GPIO.output(Buzz, 1)
            else:
                GPIO.output(Buzz, 0)
        else:
            GPIO.output(Buzz, 1)
            count = 0

        time.sleep(0.2)

def destroy():
    GPIO.output(Buzz, 1)
    GPIO.cleanup()

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()
```

Place a lighter close to the MQ-2 gas sensor, and press the switch to release gasses. A value between 0 and 255 will be displayed on the screen. If harmful gases reach a certain concentration, the buzzer will beep, and "Danger Gas!" will be printed on the screen.

You can also turn the shaft of the potentiometer on the module to raise or reduce the concentration threshold.

The MQ-2 gas sensor needs to be heated up for a while. Wait until the value printed on screen stays steady and the sensor gets warm, which means it can work normally and sensitively at that time.

Note: It is normal that the gas sensor generates heat. Actually, the higher the temperature is, the sensor is more sensitive.



6.23 Lesson 23 IR Remote Control

Introduction

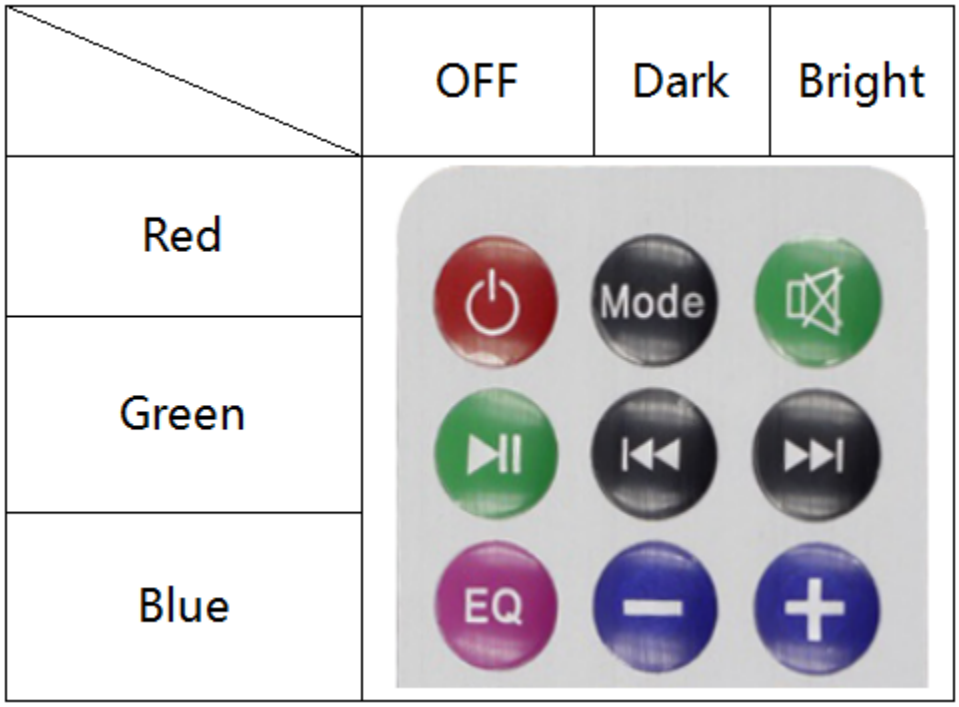
Each button of an IR remote control (as shown below) has a string of specific encoding. When a button is pressed, the IR transmitter in the remote control will send out the corresponding IR encoding signals. On the other side, when the IR receiver receives certain encoding signals, it will decode them to identify which button is pressed.

Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * IR Receiver
- 1 * RGB LED module
- 1 * IR Remote Control
- 1 * 3-Pin anti-reverse cable
- 1 * 4-Pin anti-reverse cable

Experimental Principle

In this experiment, we use the lirc library to read infrared signals returned by buttons of the remote control and translate them to button values. Then use liblircclient-dev (C) and pylirc (Python) to simplify the process for reading values from the remote control. In this experiment use 9 buttons on the top of the remote to control the color of the RGB LED module. Each row represents one color, and each column represents the brightness.

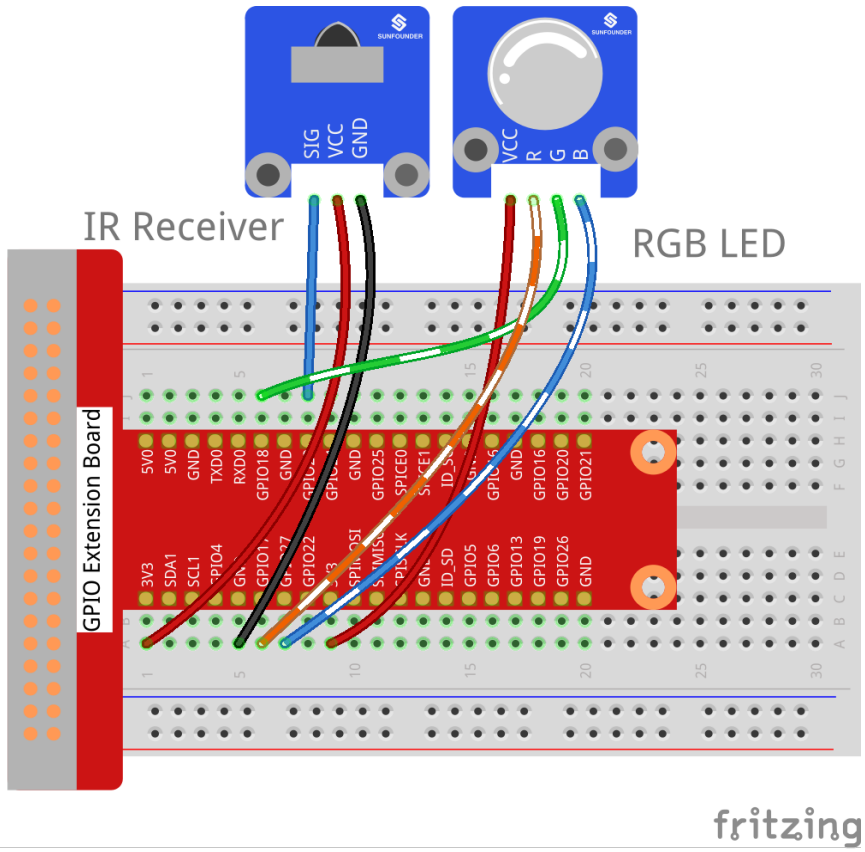


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	IR Receiver Module
GPIO4	GPIO23	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	RGB LED Module
3.3V	3V3	VCC
GPIO0	GPIO17	R
GPIO1	GPIO18	G
GPIO2	GPIO27	B



Step 2: Install the lirc:

```
sudo apt-get update
sudo apt install lirc
```

Step 3: Set up lirc.

Open your `/boot/config.txt` file:

```
sudo nano /boot/config.txt
```

Add this to the file:

```
# Uncomment this to enable the infrared communication
dtoverlay=gpio-ir,gpio_pin=23
dtoverlay=gpio-ir-tx,gpio_pin=22
```

Press Ctrl +O and Ctrl +X, save and exit .

Step 4: edit `/etc/lirc/lirc_options.conf`.

Open the `/etc/lirc/lirc_options.conf`

```
sudo nano /etc/lirc/lirc_options.conf
```

Modify the file as below:

```
driver = default
device = /dev/lirc1
```

Step 5: Copy the configuration file to `/home/pi` and `/etc/lirc`:

```
cd /home/pi/SunFounder_SensorKit_for_RPi2
cp lircd.conf /home/pi
sudo cp lircd.conf /etc/lirc/
```

Step 6: Reboot the Raspberry Pi after the change.

```
sudo reboot
```

Step 7: Test the IR receiver.

Check if lirc module is loaded:

```
ls /dev/li*
```

You should see this:

```
/dev/lirc0 /dev/lirc1
```

Step 8: Run the command to start outputting raw data from the IR receiver:

```
irw
```

When you press a button on the remote, you can see the button name printed on the screen.

```
pi@raspberrypi:~ $ irw
0000000000000001 00 KEY_CHANNELDOWN ./lircd.conf
0000000000000003 00 KEY_CHANNELUP ./lircd.conf
0000000000000002 00 KEY_CHANNEL ./lircd.conf
0000000000000004 00 KEY_PREVIOUS ./lircd.conf
0000000000000005 00 KEY_NEXT ./lircd.conf
0000000000000006 00 KEY_PLAYPAUSE ./lircd.conf
0000000000000008 00 KEY_VOLUMEDOWN ./lircd.conf
0000000000000007 00 KEY_VOLUMEUP ./lircd.conf
0000000000000009 00 KEY_EQUAL ./lircd.conf
0000000000000015 00 BTN_1 ./lircd.conf
0000000000000014 00 BTN_0 ./lircd.conf
000000000000000a 00 KEY_NUMERIC_0 ./lircd.conf
000000000000000b 00 KEY_NUMERIC_1 ./lircd.conf
```

If it does not appear, somewhere may be incorrectly configured. Check again that you've connected everything and haven't crossed any wires.

For C Users:

Step 9: Download LIRC client library:

```
sudo apt-get install liblircclient-dev
```

Step 10: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/23_ircontrol/
```

Step 11: Copy the `lircrc` file to `/etc/lirc/lirc/`:

```
sudo cp lircrc /etc/lirc/
```

Step 12: Compile.

```
gcc ircontrol.c -lwiringPi -llirc_client
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 13: Run.

```
sudo ./a.out
```

Code

```
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <lirc/lirc_client.h>
#include <time.h>

#define uchar unsigned char

#define LedPinRed    0
#define LedPinGreen  1
#define LedPinBlue   2

uchar color[3] = {0xff, 0xff, 0xff};
uchar Lv[3]    = {0xff, 0x44, 0x00};

char *keymap[21] ={
    " KEY_CHANNELDOWN ",
    " KEY_CHANNEL ",
    " KEY_CHANNELUP ",
    " KEY_PREVIOUS ",
    " KEY_NEXT ",
    " KEY_PLAYPAUSE ",
    " KEY_VOLUMEDOWN ",
    " KEY_VOLUMEUP ",
    " KEY_EQUAL ",
    " KEY_NUMERIC_0 ",
    " BTN_0 ",
    " BTN_1 ",
    " KEY_NUMERIC_1 ",
    " KEY_NUMERIC_2 ",
    " KEY_NUMERIC_3 ",
    " KEY_NUMERIC_4 ",
    " KEY_NUMERIC_5 ",
    " KEY_NUMERIC_6 ",
    " KEY_NUMERIC_7 ",
    " KEY_NUMERIC_8 ",
    " KEY_NUMERIC_9 "};

void ledInit(void)
{
    softPwmCreate(LedPinRed,  0, 100);
```

(continues on next page)

(continued from previous page)

```
    softPwmCreate(LedPinGreen,0, 100);
    softPwmCreate(LedPinBlue, 0, 100);
}

void ledColorSet()
{
    softPwmWrite(LedPinRed,   color[0]);
    softPwmWrite(LedPinGreen, color[1]);
    softPwmWrite(LedPinBlue,  color[2]);
}

int key(char *code){
    int i;
    int num;
    for (i=0; i<21; i++){
        if (strstr(code, keymap[i])){
            num = i;
        }
    }
    return num + 1;
}

int RGB(int i){
    switch(i){
        case 1: color[0] = Lv[0]; printf("Red OFF\n"); break;
        case 2: color[0] = Lv[1]; printf("Light Red\n"); break;
        case 3: color[0] = Lv[2]; printf("Dark Red\n"); break;
        case 4: color[1] = Lv[0]; printf("Green OFF\n"); break;
        case 5: color[1] = Lv[1]; printf("Light Green\n"); break;
        case 6: color[1] = Lv[2]; printf("Dark Green\n"); break;
        case 7: color[2] = Lv[0]; printf("Blue OFF\n"); break;
        case 8: color[2] = Lv[1]; printf("Light Blue\n"); break;
        case 9: color[2] = Lv[2]; printf("Dark Green\n"); break;
    }
}

int main(void)
{
    struct lirc_config *config;
    int buttonTimer = millis();
    char *code;
    char *c;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }

    if(lirc_init("lirc",1)==-1)
        exit(EXIT_FAILURE);

    ledInit();
    ledColorSet();

    if(lirc_readconfig(NULL,&config,NULL)==0)
    {
        while(lirc_nextcode(&code)==0)
        {
```

(continues on next page)

(continued from previous page)

```
            if(code==NULL) continue;{
                if (millis() - buttonTimer  > 400){
                    RGB(key(code));
                    ledColorSet(color);
                }
            }
            free(code);
        }
        lirc_freeconfig(config);
    }
    lirc_deinit();
    exit(EXIT_SUCCESS);
    return 0;
}
```

For Python Users:

Step 9: Install `lirc` Python packages:

```
sudo pip3 install lirc
```

Step 10: Change directory:

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 11: Run.

```
sudo python3 23_ircontrol.py
```

Code

```
import lirc
import time
import RPi.GPIO as GPIO

# client = lirc.Client()
# print(client.version())

''' RGB config'''
Rpin = 17
Gpin = 18
Bpin = 27

Lv = [0, 20, 90] # Light Level
color = [0, 0, 0]

p_R = None
p_G = None
p_B = None


def setColor(color):
    # global p_R, p_G, p_B
    p_R.ChangeDutyCycle(100 - color[0])     # Change duty cycle
    p_G.ChangeDutyCycle(100 - color[1])
    p_B.ChangeDutyCycle(100 - color[2])
```

(continues on next page)

(continued from previous page)

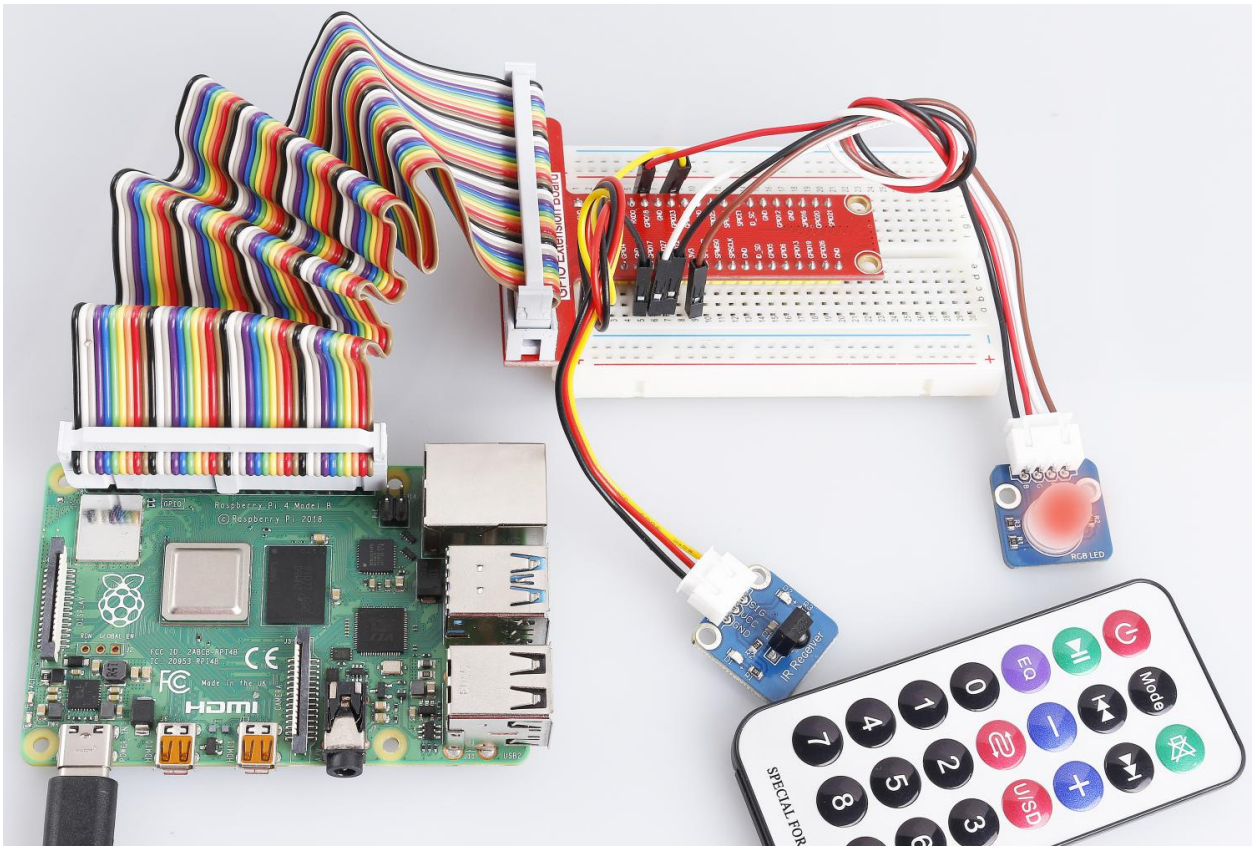
```
def x():
    setColor(color)

def setup():
    global p_R, p_G, p_B
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(Rpin, GPIO.OUT)
    GPIO.setup(Gpin, GPIO.OUT)
    GPIO.setup(Bpin, GPIO.OUT)

    p_R = GPIO.PWM(Rpin, 2000) # Set Frequece to 2KHz
    p_G = GPIO.PWM(Gpin, 2000)
    p_B = GPIO.PWM(Bpin, 2000)

    p_R.start(100)
    p_G.start(100)
    p_B.start(100)

def map(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def key_handler(key:str):
    global color

    if key == 'KEY_CHANNELDOWN':
        color[0] = Lv[0]
        print ('Red OFF')

    elif key == 'KEY_CHANNEL':
        color[0] = Lv[1]
        print ('Dark Red')

    elif key == 'KEY_CHANNELUP':
        color[0] = Lv[2]
        print ('Bright Red')

    elif key == 'KEY_PREVIOUS':
        color[1] = Lv[0]
        print ('Green OFF')

    elif key == 'KEY_NEXT':
        color[1] = Lv[1]
        print ('Dark Green')

    elif key == 'KEY_PLAYPAUSE':
        color[1] = Lv[2]
        print ('Bright Green')

    elif key == 'KEY_VOLUMEDOWN':
        color[2] = Lv[0]
        print ('Blue OFF')

    elif key == 'KEY_VOLUMEUP':
        color[2] = Lv[1]
        print ('Dark Blue')

    elif key == 'KEY_EQUAL':
```

(continues on next page)

(continued from previous page)

```
        color[2] = Lv[2]
        print ('Bright BLUE')

    setColor(color)


def loop():
    with lirc.LircdConnection(timeout=5.0) as conn:
        conn.connect()
        while True:
            try:
                # print(conn.readline()) # 0000000000000001 00 KEY_CHANNELDOWN ./
↪lircd.conf
                key = conn.readline().split(' ')[2] #KEY_CHANNELDOWN
                # print(key)
                key_handler(key)
            except TimeoutError:
                # print('Timeout')
                pass

def destroy():
    p_R.stop()
    p_G.stop()
    p_B.stop()
    GPIO.output(Rpin, GPIO.HIGH)     # Turn off all leds
    GPIO.output(Gpin, GPIO.HIGH)
    GPIO.output(Bpin, GPIO.HIGH)
    GPIO.cleanup()


if __name__ == "__main__":
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()
```

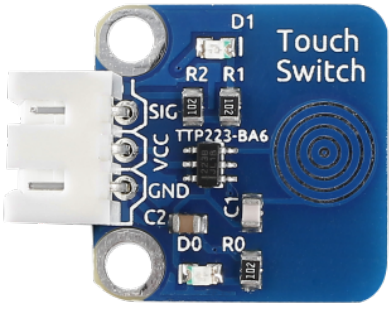
Each of the top three rows of buttons on the remote control represents a kind of color, i.e. red, green, and blue, top to bottom. Each column represents off, light, and dark. For example, press the second button (light) on the first row (red), and the LED will flash light red. You can use the remote to generate 27 colors in total (including all the LEDs off). Try to change the color of the RGB LED with the 9 buttons!



6.24 Lesson 24 Touch Switch

Introduction

A touch sensor operate with the conductivity of human body. When you touch the metal on the base electrode of the transistor, the level of pin SIG will turn over.



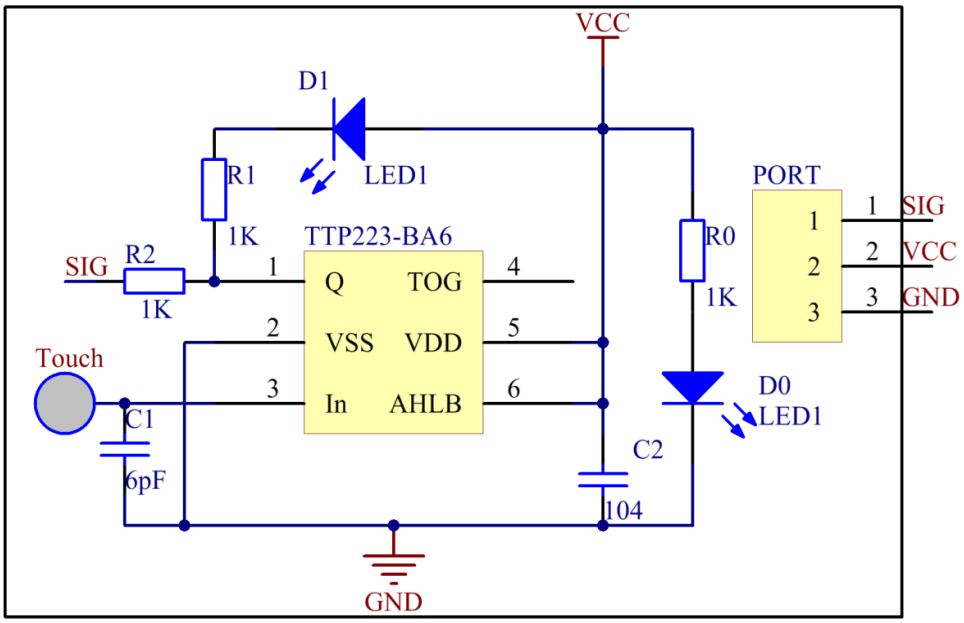
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Touch sensor module
- 1 * Dual-Color LED module

- 2 * 3-Pin anti-reverse cable

Experimental Principle

In this experiment, touch the base electrode of the transistor by fingers to make it conduct as human body itself is a kind of conductor and an antenna that can receive electromagnetic waves in the air. These electromagnetic wave signals collected from the human body are amplified by the transistor and processed by the comparator on the module to output steady signals. The schematic diagram:

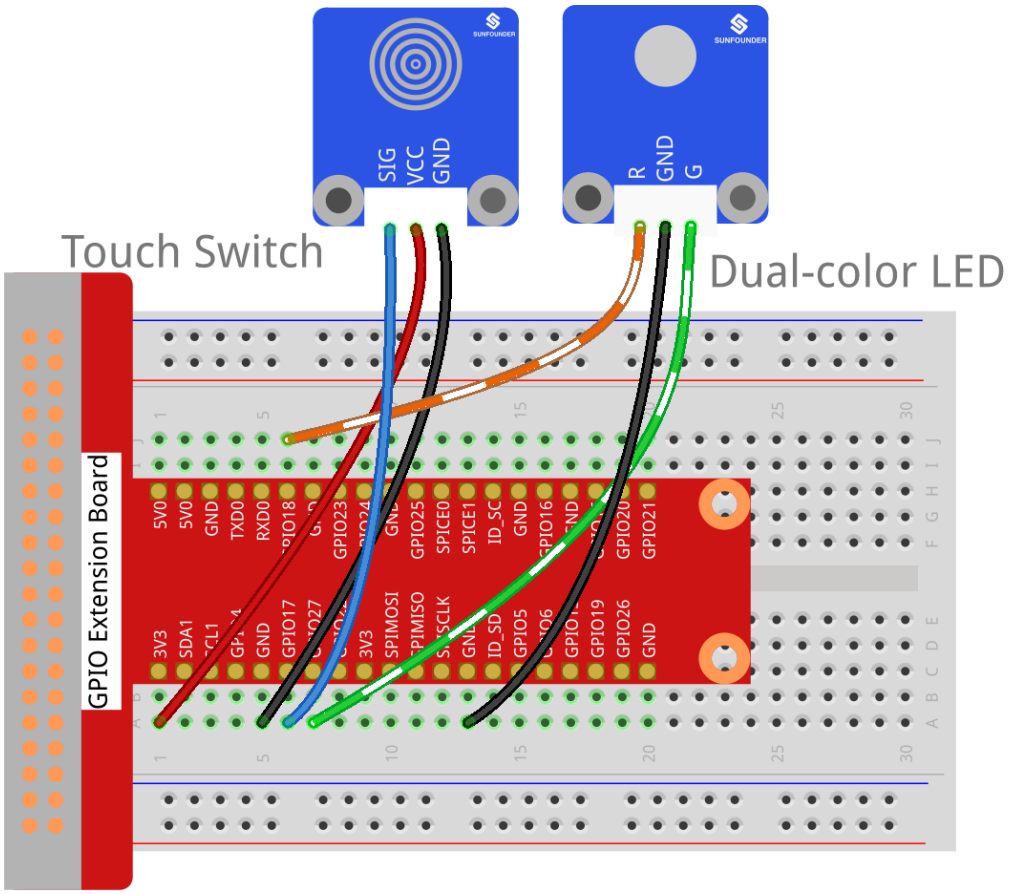


Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Touch Sensor Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	Dual-Color LED Module
GPIO1	GPIO18	R
GND	GND	GND
GPIO2	GPIO27	G



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/24_touch_switch/
```

Step 3: Compile.

```
gcc touch_switch.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define TouchPin    0
```

(continues on next page)

(continued from previous page)

```
#define Gpin                2
#define Rpin                1

int tmp = 0;

void LED(int color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == 0)
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == 1)
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

void Print(int x){
    if (x != tmp){
        if (x == 0)
            printf("...ON\n");
        if (x == 1)
            printf("OFF..\n");
        tmp = x;
    }
}

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

    pinMode(TouchPin, INPUT);

    while(1){
        LED(digitalRead(TouchPin));
        Print(digitalRead(TouchPin));
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 24_touch_switch.py
```

Code

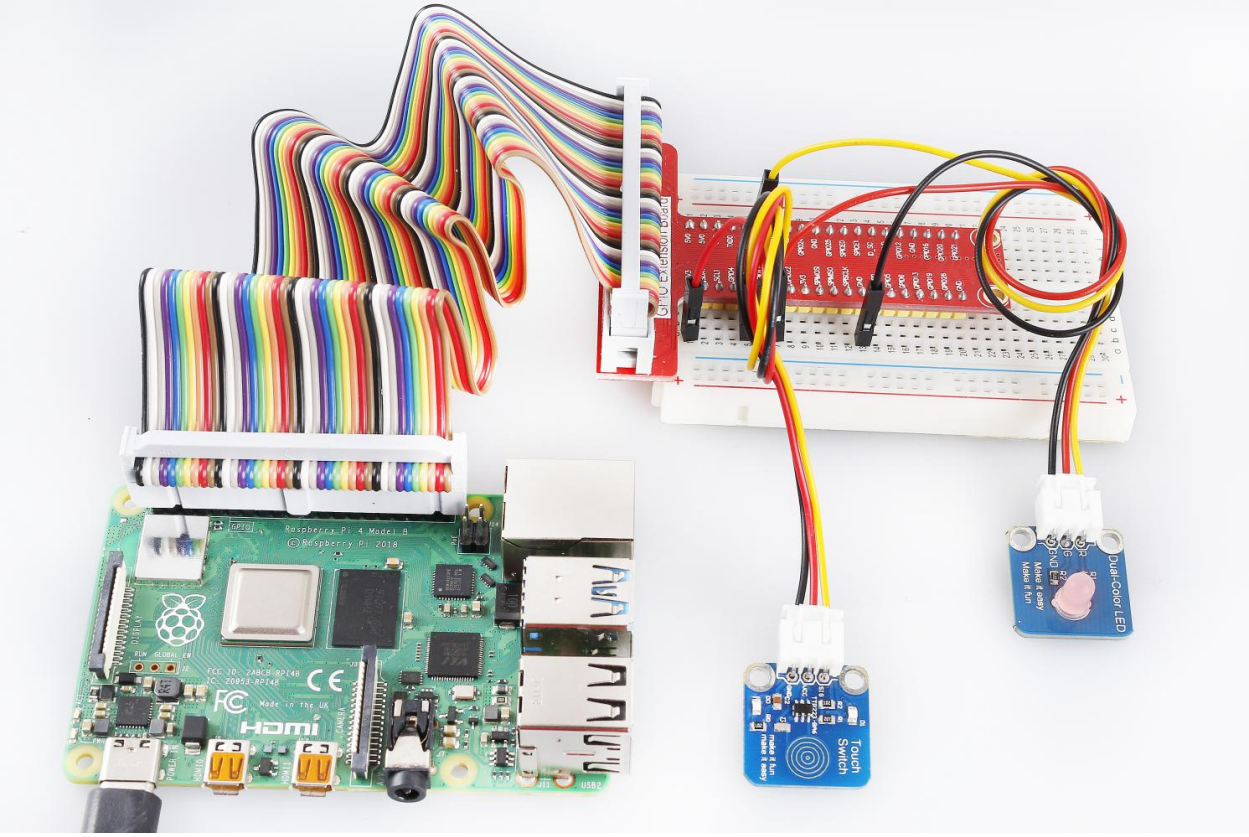
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

TouchPin = 11
Gpin   = 13
Rpin   = 12

tmp = 0

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)      # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)      # Set Red Led Pin mode to output
    GPIO.setup(TouchPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's mode is␣
↪input, and pull up to high level(3.3V)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
        GPIO.output(Gpin, 1)

def Print(x):
    global tmp
    if x != tmp:
        if x == 0:
            print ('    **********')
            print ('    *     ON *')
            print ('    **********')

        if x == 1:
            print ('    **********')
            print ('    * OFF    *')
            print ('    **********')
        tmp = x

def loop():
    while True:
        Led(GPIO.input(TouchPin))
        Print(GPIO.input(TouchPin))

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)       # Green led off
    GPIO.output(Rpin, GPIO.HIGH)       # Red led off
    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.
```

(continues on next page)

(continued from previous page)

```
        destroy()
```

Now, touch the metal disk, you can see the LED change its colors and "ON" and "OFF" printed on the screen.



6.25 Lesson 25 Ultrasonic Ranging Module

Introduction

The ultrasonic sensor uses sound to accurately detect objects and measure distances. It sends out ultrasonic waves and converts them into electronic signals.

Required Components

- 1 * Raspberry Pi
- 1 * Breadboard

- 1 * Ultrasonic ranging module
- 1 * 4-Pin anti-reverse cable

Experimental Principle

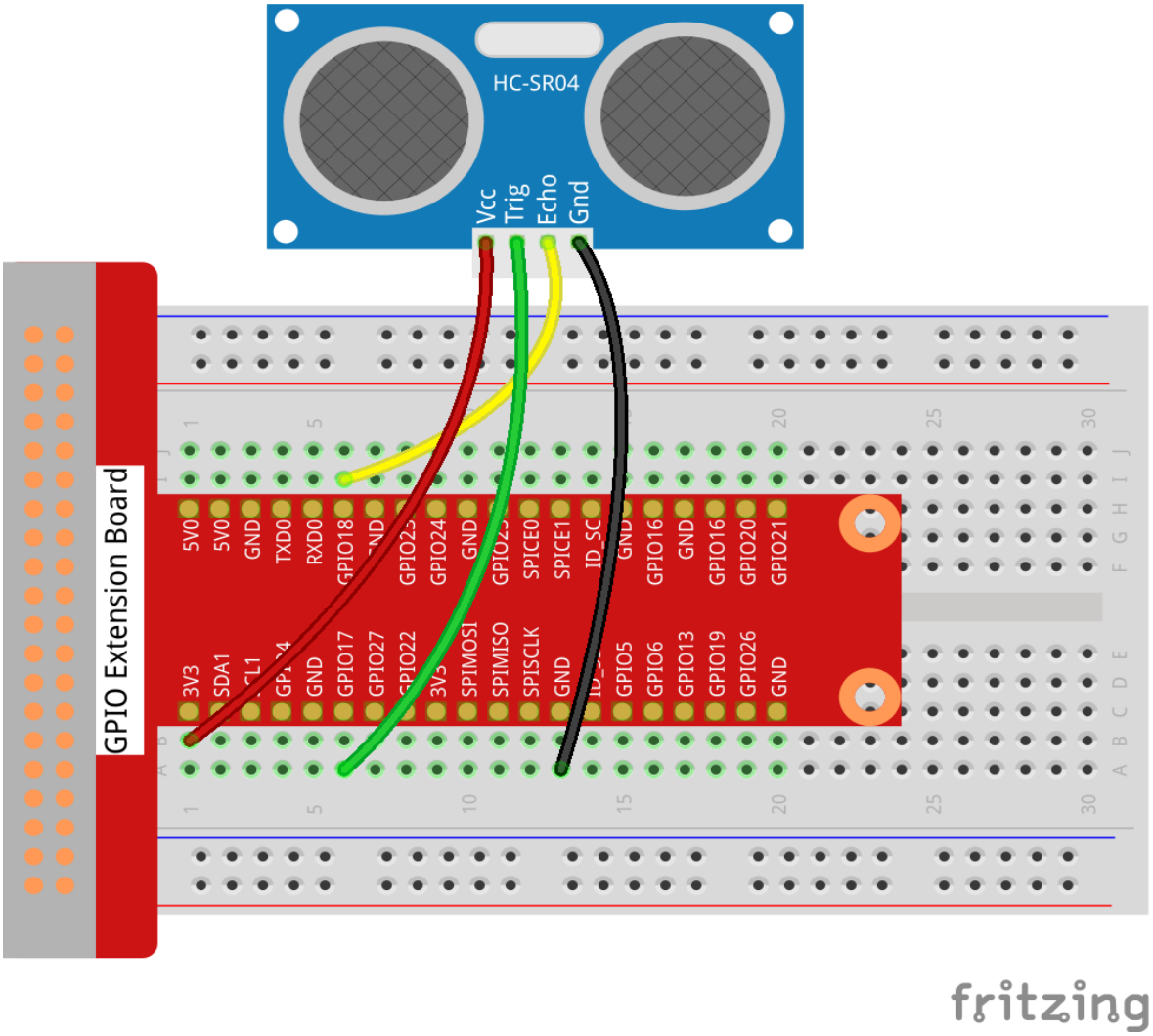
This sensor works by sending a sound wave out and calculating the time it takes for the sound wave to get back to the ultrasonic sensor. By doing this, it can tell us how far away objects are relative to the ultrasonic sensor.

Test distance = (high level time * velocity of sound (340M/S)) / 2 (in meters)

Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Ultrasonic Ranging Module
3.3V	3V3	VCC
GPIO0	GPIO17	Trig
GPIO1	GPIO18	Echo
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/25_ultrasonic_ranging/
```

Step 3: Compile.

```
gcc ultrasonic_ranging.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

```
#include <wiringPi.h>
#include <stdio.h>
#include <sys/time.h>

#define Trig    0
#define Echo    1

void ultraInit(void)
{
    pinMode(Echo, INPUT);
    pinMode(Trig, OUTPUT);
}

float disMeasure(void)
{
    struct timeval tv1;
    struct timeval tv2;
    long time1, time2;
    float dis;

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

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

    while(!(digitalRead(Echo) == 1));
    gettimeofday(&tv1, NULL);           //

    while(!(digitalRead(Echo) == 0));
    gettimeofday(&tv2, NULL);           //

    time1 = tv1.tv_sec * 1000000 + tv1.tv_usec;   //
    time2  = tv2.tv_sec * 1000000 + tv2.tv_usec;

    dis = (float)(time2 - time1) / 1000000 * 34000 / 2;  //

    return dis;
```

(continues on next page)

(continued from previous page)

```
}

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("%0.2f cm\n\n",dis);
        delay(300);
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 25_ultrasonic_ranging.py
```

Code

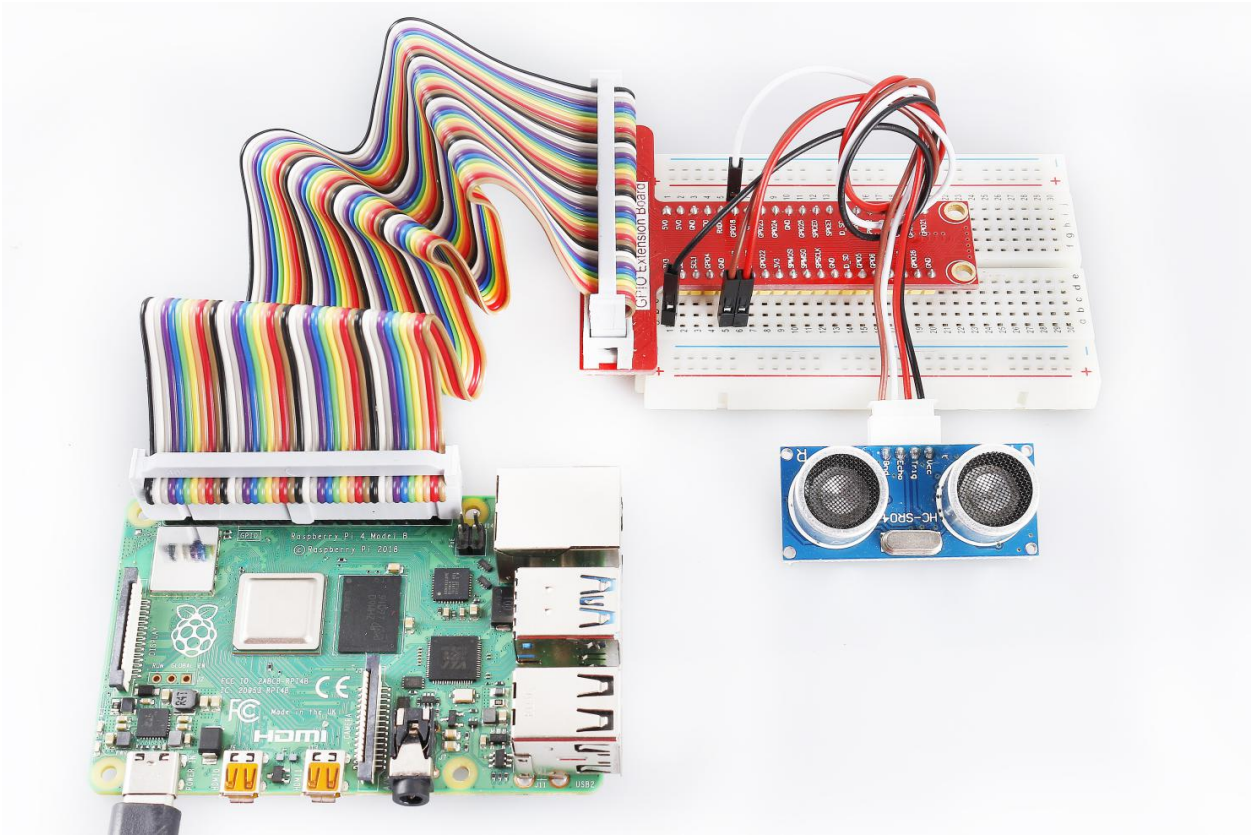
```
#!/usr/bin/env python3

import RPi.GPIO as GPIO
import time

TRIG = 11
ECHO = 12

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)

def distance():
    GPIO.output(TRIG, 0)
    time.sleep(0.000002)

    GPIO.output(TRIG, 1)
    time.sleep(0.00001)
    GPIO.output(TRIG, 0)

    while GPIO.input(ECHO) == 0:
        a = 0
```

(continues on next page)

(continued from previous page)

```
    time1 = time.time()
    while GPIO.input(ECHO) == 1:
        a = 1
    time2 = time.time()

    during = time2 - time1
    return during * 340 / 2 * 100

def loop():
    while True:
        dis = distance()
        print (dis, 'cm')
        print ('')
        time.sleep(0.3)

def destroy():
    GPIO.cleanup()

if __name__ == "__main__":
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

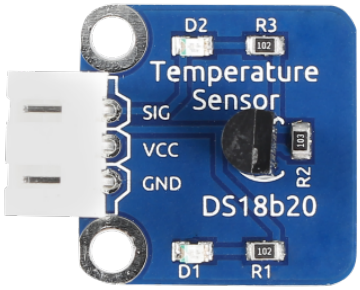
Now you can see the distance between the ultrasonic ranging module and the obstacle (like your palm) in front on the screen. Sway your hand over the ultrasonic ranging module slowly and observe the distance printed on the screen.



6.26 Lesson 26 DS18B20 Temperature Sensor

Introduction

Temperature Sensor DS18B20 is a commonly used digital temperature sensor featured with small size, low-cost hardware, strong anti-interference capability and high precision. The digital temperature sensor is easy to wire and can be applied a various occasions after packaging. Different from conventional AD collection temperature sensors, it uses a 1-wire bus and can directly output temperature data.



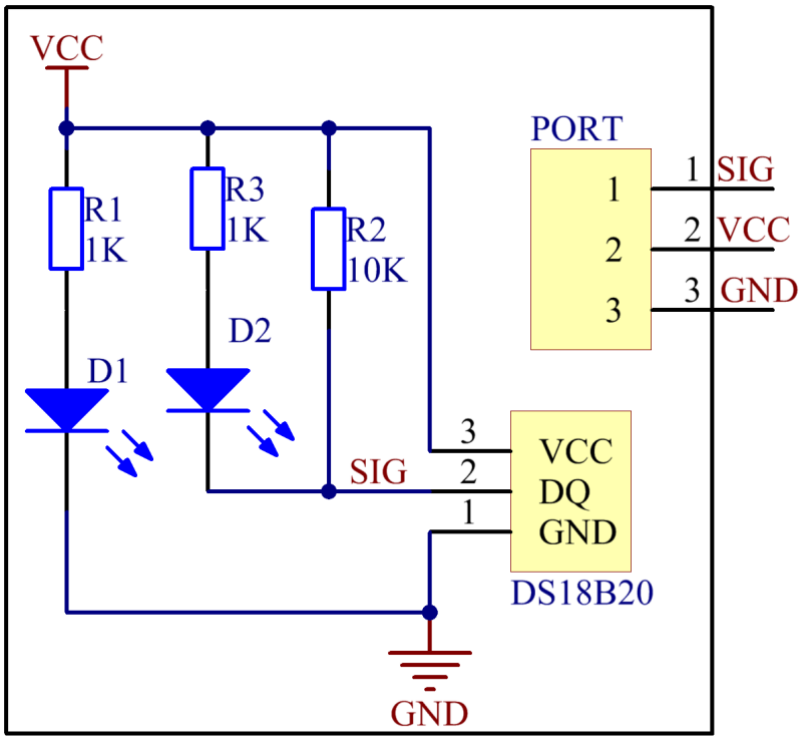
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * DS18B20 Temperature Sensor module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

With a unique single-wire interface, DS18B20 requires only one pin for a two-way communication with a microprocessor. It supports multi-point networking to measure multi-point temperatures. Eight sensors can be connected at most, because it will consume too much power supply and cause low voltage thus harming the stability of transmission.

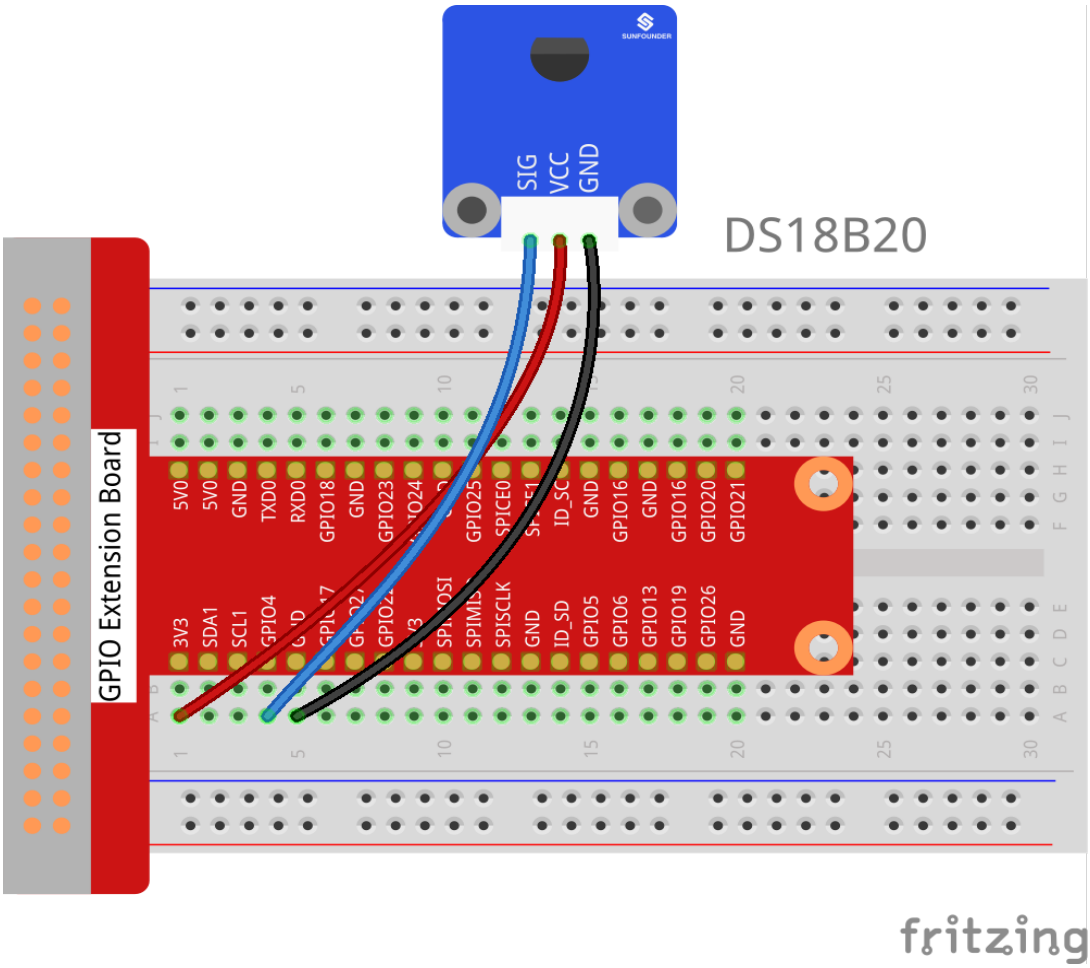
When using the DS18B20, you need to connect a 10K resistor to the middle pin DQ to pull up the level. The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit according to the following method.

Raspberry Pi	GPIO Extension Board	DS18B20 Temperature Sensor
GPIO7	GPIO4	SIG
3.3V	3V3	VCC
GND	GND	GND



Step 2: Upgrade your kernel.

```
sudo apt-get update
sudo apt-get upgrade
```

Step 3: You can edit that file with nano.

```
sudo nano /boot/config.txt
```

Then scroll to the bottom and type.

```
dtoverlay = w1-gpio
```

Then reboot with

```
sudo reboot
```

Mount the device drivers and confirm whether the device is effective or not.

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
cd /sys/bus/w1/devices/
ls
```

The result is as follows:

```
root@rasberrypi:/sys/bus/w1/devices# ls
28-00000495db35 w1_bus_master1
```

28-00000495db35 is an external temperature sensor device, but it may vary with every client. This is the serial number of your ds18b20.

Step 4: Check the current temperature.

```
cd 28-00000495db35
ls
```

The result is as follows:

```
root@rasberrypi:/sys/bus/w1/devices/28-00000495db35# ls
driver  id  name  power  subsystem  uevent  w1_slave
cat w1_slave
```

The result is as follows:

```
root@raspberrypi:/sys/bus/w1_slave/28-00000495db35# cat w1_slave
a3 01 4b 46 7f ff 0d 10 ce : crc=ce YES
a3 01 4b 46 7f ff 0d 10 ce t=26187
```

The second line t=26187 is current temperature value. If you want to convert it to degree Celsius, you can divide by 1000, that is, the current temperature is 26187/1000=26.187 °C.

For C Users:

Step 2: Change directory and edit.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/26_ds18b20/
```

Open the `ds18b20.c` to change the sensor address.

```
nano ds18b20.c
```

Find the following line, replace "28-00000495db35" with your sensor address. Save and exit.

```
fd = open("/sys/bus/w1/devices/28-031590bf4aff/w1_slave", O_RDONLY);
```

Step 6: Compile.

```
gcc ds18b20.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 7: Run.

```
sudo ./a.out
```

Code

```
#include <wiringPi.h>
#include <sys/types.h>
#include <sys/stat.h>
```

(continues on next page)

(continued from previous page)

```
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>

#define             BUFSIZE         128

typedef unsigned char uchar;
typedef unsigned int  uint;

float tempRead(void)
{
    float temp;
    int i, j;
    int fd;
    int ret;

    char buf[BUFSIZE];
    char tempBuf[5];

    fd = open("/sys/bus/w1/devices/28-031590bf4aff/w1_slave", O_RDONLY);

    if(-1 == fd){
        perror("open device file error");
        return 1;
    }

    while(1){
        ret = read(fd, buf, BUFSIZE);
        if(0 == ret){
            break;
        }
        if(-1 == ret){
            if(errno == EINTR){
                continue;
            }
            perror("read()");
            close(fd);
            return 1;
        }
    }

    for(i=0;i<sizeof(buf);i++){
        if(buf[i] == 't'){
            for(j=0;j<sizeof(tempBuf);j++){
                tempBuf[j] = buf[i+2+j];
            }
        }
    }

    temp = (float)atoi(tempBuf) / 1000;

    close(fd);

    return temp;
}
```

(continues on next page)

(continued from previous page)

```
int main(void)
{
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    float temp;
    while(1){
        temp = tempRead();
        printf("Current temperature : %0.3f\n", temp);
    }
    return 0;
}
```

For Python Users:

Step 5: Change directory and edit.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
nano 26_ds18b20.py
```

Find the following code, replace `28-031590bf4aff` with your sensor address.

```
def setup():
    global ds18b20
    for i in os.listdir('/sys/bus/w1/devices'):
        if i != 'w1_bus_master1':
            ds18b20 = '28-031590bf4aff'
```

Press Ctrl + X -> Y -> Enter to save and exit.

Step 6: Run.

```
sudo python3 26_ds18b20.py
```

Code

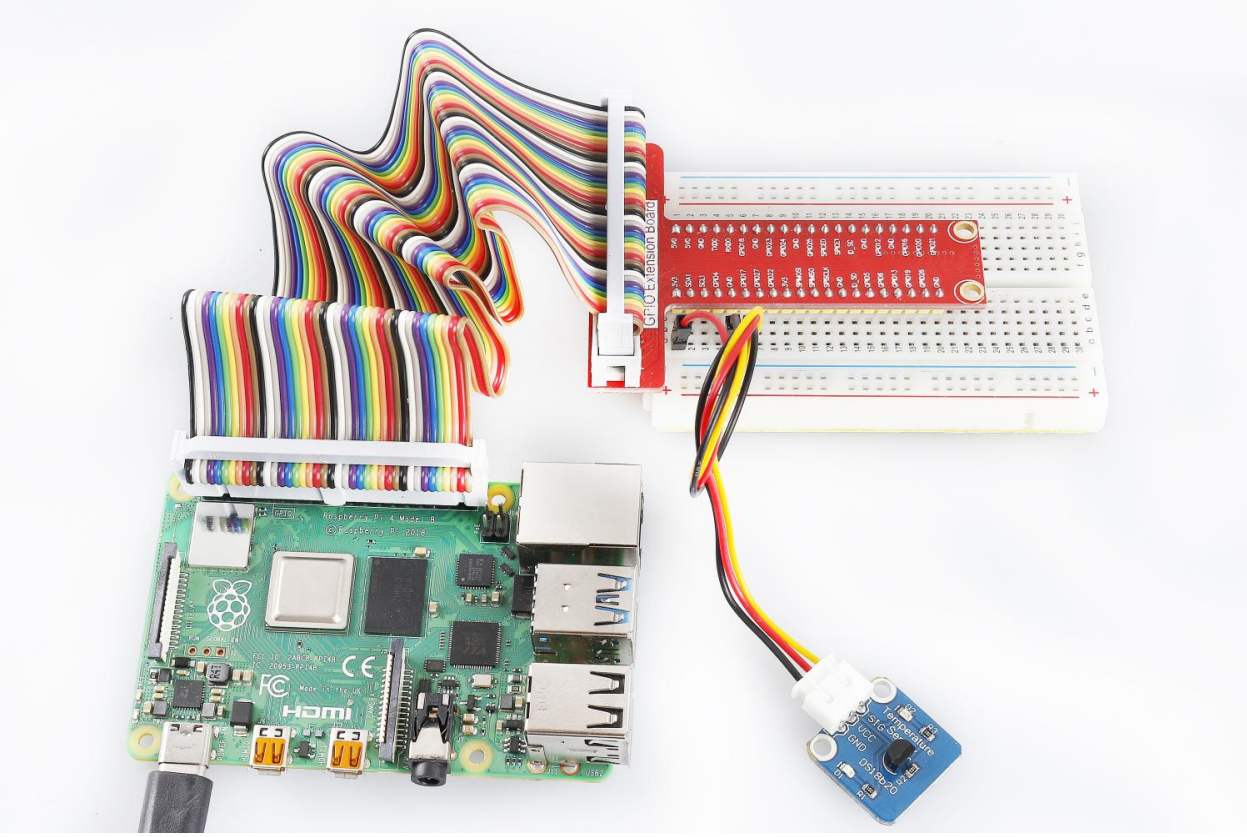
```
#!/usr/bin/env python3
import os

ds18b20 = ''

def setup():
    global ds18b20
    for i in os.listdir('/sys/bus/w1/devices'):
        if i != 'w1_bus_master1':
            ds18b20 = '28-031590bf4aff'

def read():
#    global ds18b20
    location = '/sys/bus/w1/devices/' + ds18b20 + '/w1_slave'
    tfile = open(location)
    text = tfile.read()
    tfile.close()
    secondline = text.split("\n")[1]
    temperaturedata = secondline.split(" ")[9]
```

(continues on next page)

(continued from previous page)

```
    temperature = float(temperaturedata[2:])
    temperature = temperature / 1000
    return temperature

def loop():
    while True:
        if read() != None:
            print ("Current temperature : %0.3f C" % read())

def destroy():
    pass

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()
```

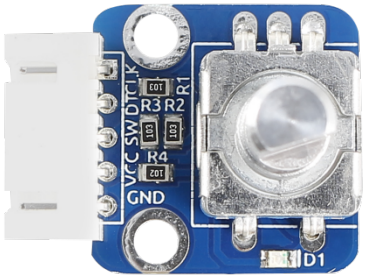
Now, you can see the current temperature value displayed on the screen.



6.27 Lesson 27 Rotary Encoder Module

Introduction

A rotary encoder is an electro-mechanical device that converts the angular position or motion of a shaft or axle to analog or digital code. Rotary encoders are usually placed at the side which is perpendicular to the shaft. They act as sensors for detecting angle, speed, length, position, and acceleration in automation field.

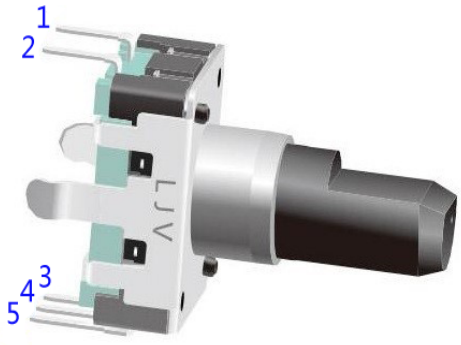


Required Components

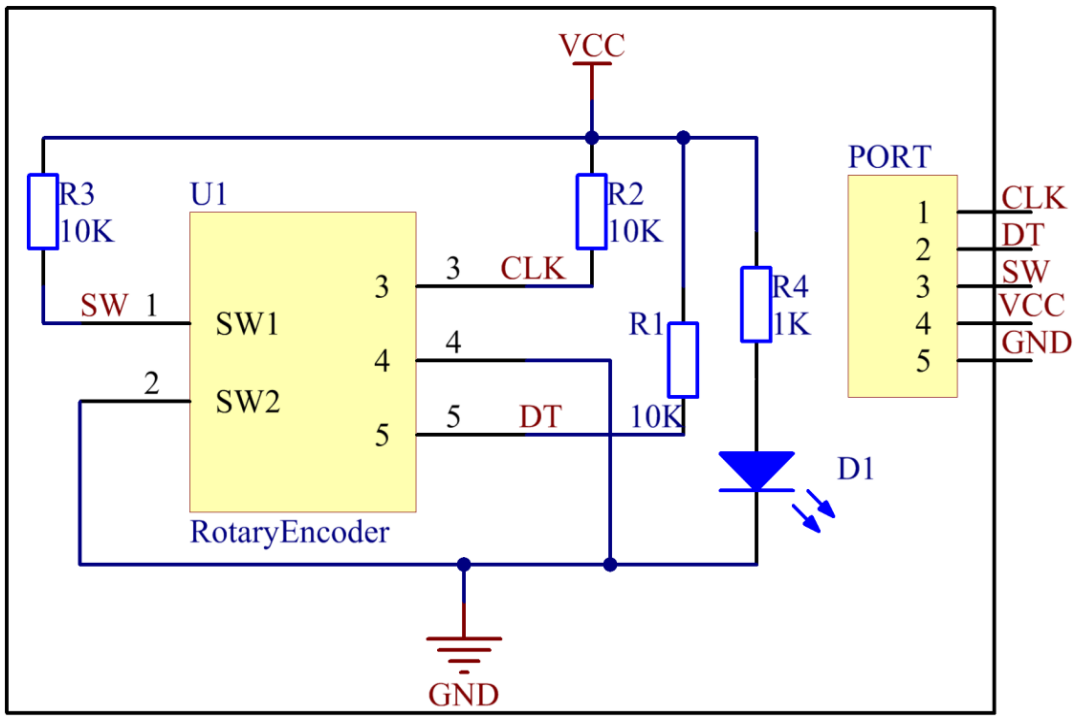
- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Rotary Encoder module
- 1 * 5-Pin anti-reverse cable

Experimental Principle

Most rotary encoders have 5 pins with three functions of turning left & right and pressing down. Pin 1 and pin 2 are switch wiring terminals used to press. They are similar to buttons previously mentioned, so we will no longer discuss them in this experiment. Pin 4 is generally connected to ground. Pin 3 and pin 5 are first connected to pull-up resistor and then to the microprocessor. In this experiment, they are connected to GPIO0 and GPIO1 of Raspberry Pi. When it is rotated left and right, there will be pulse inputs in pin 1 and pin 3.



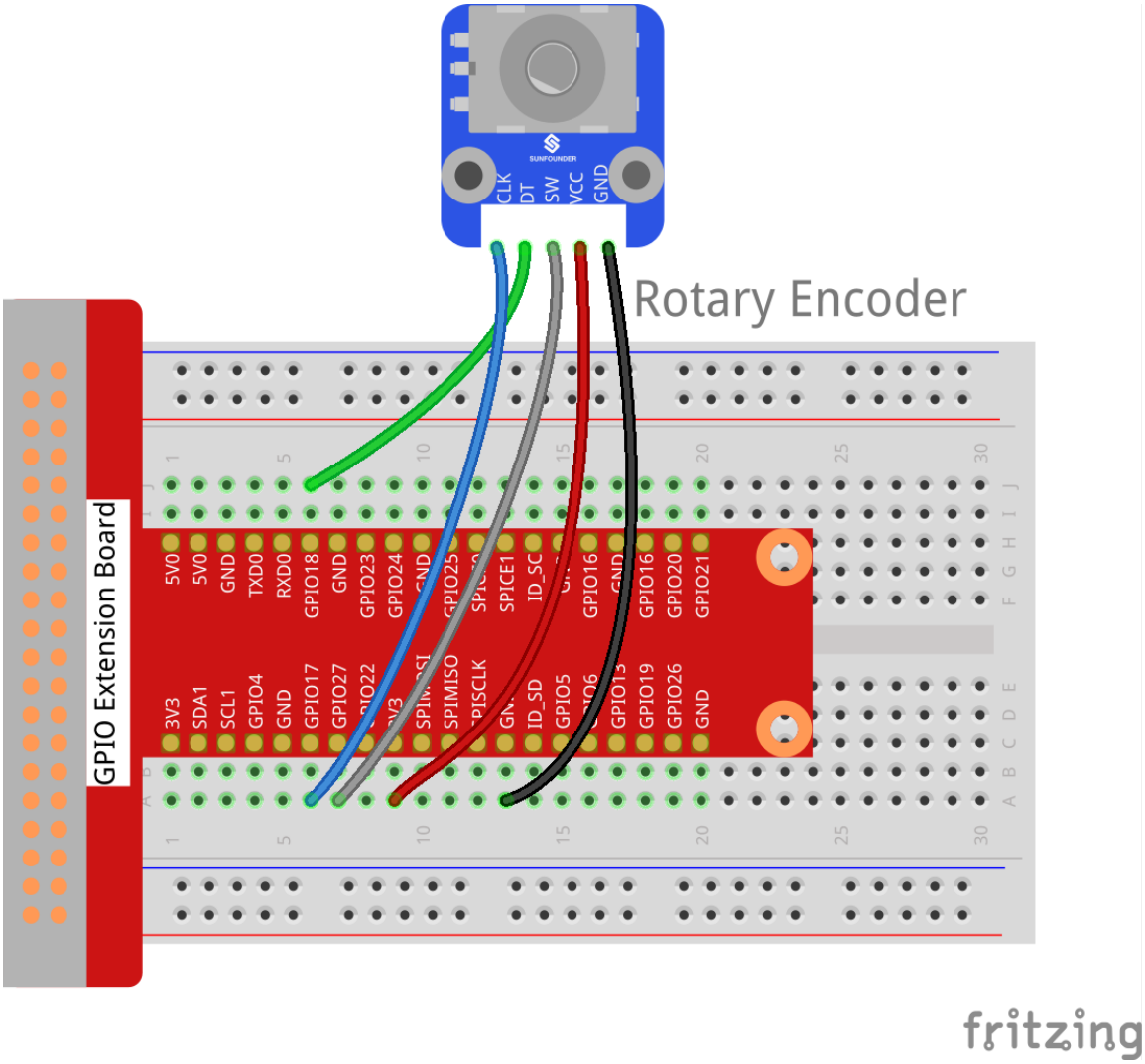
It shows that if output 1 is high and output 2 is high, then the switch rotates clockwise; if output 1 is high and output 2 is low, then the switch rotates counterclockwise. As a result, during SCM programming, if output 1 is high, then you can tell whether the rotary encoder rotates left or right as long as you know the state of output 2.



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Rotary Encoder Module
GPIO0	GPIO17	CLK
GPIO1	GPIO18	DT
GPIO2	GPIO27	SW
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/27_rotary_encoder/
```

Step 3: Compile.

```
gcc rotary_encoder.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <wiringPi.h>

#define  RoAPin    0
#define  RoBPin    1
#define  SWPin     2

static volatile int globalCounter = 0 ;

unsigned char flag;
unsigned char Last_RoB_Status;
unsigned char Current_RoB_Status;

void btnISR(void)
{
    globalCounter = 0;
}

void rotaryDeal(void)
{
    Last_RoB_Status = digitalRead(RoBPin);

    while(!digitalRead(RoAPin)){
        Current_RoB_Status = digitalRead(RoBPin);
        flag = 1;
    }

    if(flag == 1){
        flag = 0;
        if((Last_RoB_Status == 0)&&(Current_RoB_Status == 1)){
            globalCounter ++;
        }
        if((Last_RoB_Status == 1)&&(Current_RoB_Status == 0)){
            globalCounter --;
        }
    }
}

int main(void)
{
    if(wiringPiSetup() < 0){
        fprintf(stderr, "Unable to setup wiringPi:%s\n",strerror(errno));
        return 1;
    }

    pinMode(SWPin, INPUT);
    pinMode(RoAPin, INPUT);
    pinMode(RoBPin, INPUT);

    pullUpDnControl(SWPin, PUD_UP);

    if(wiringPiISR(SWPin, INT_EDGE_FALLING, &btnISR) < 0){
        fprintf(stderr, "Unable to init ISR\n",strerror(errno));
        return 1;
```

(continues on next page)

(continued from previous page)

```
    }

    int tmp = 0;

    while(1){
        rotaryDeal();
        if (tmp != globalCounter){
            printf("%d\n", globalCounter);
            tmp = globalCounter;
        }
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 27_rotary_encoder.py
```

Code

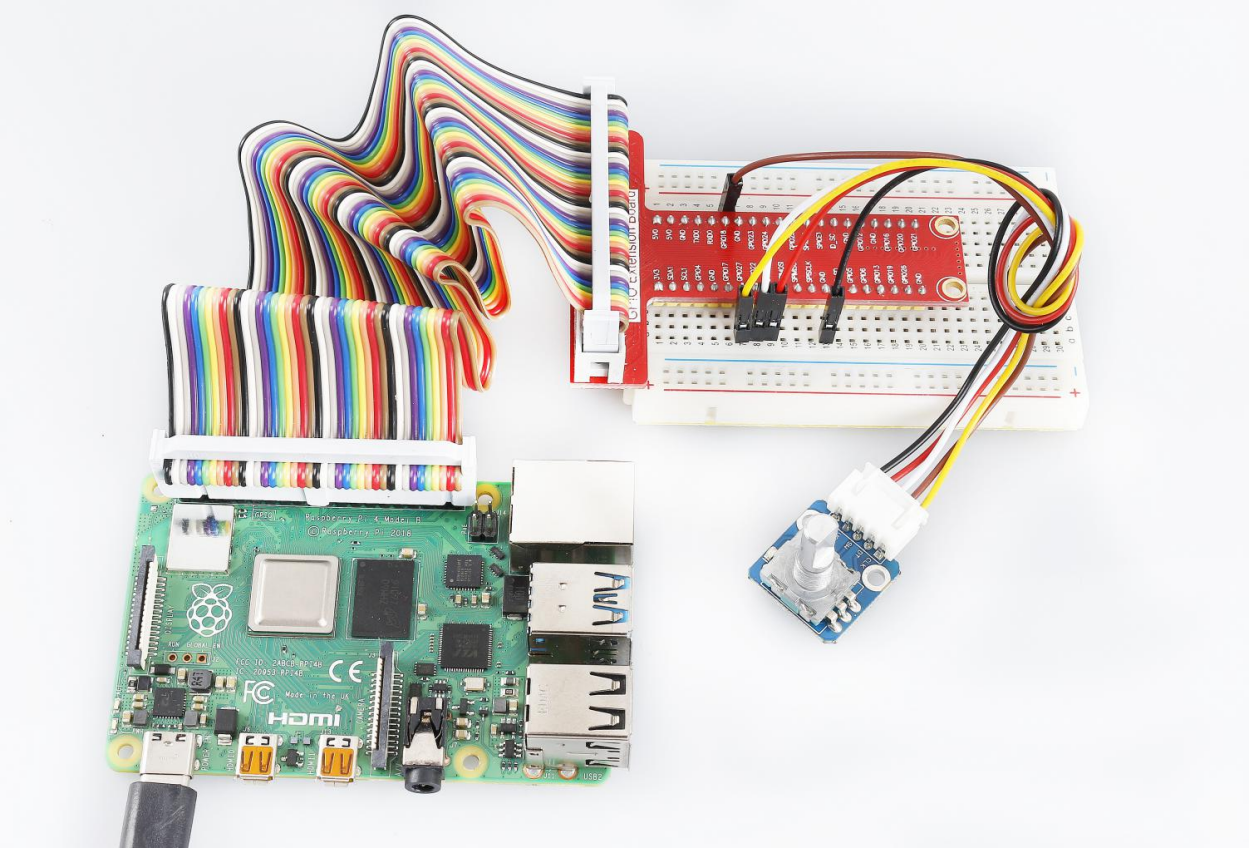
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

RoAPin = 11    # CLK Pin
RoBPin = 12    # DT Pin
BtnPin = 13    # Button Pin

globalCounter = 0

flag = 0
Last_RoB_Status = 0
Current_RoB_Status = 0

def setup():
    GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
    GPIO.setup(RoAPin, GPIO.IN)    # input mode
    GPIO.setup(RoBPin, GPIO.IN)
    GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def rotaryDeal():
    global flag
    global Last_RoB_Status
    global Current_RoB_Status
    global globalCounter
    Last_RoB_Status = GPIO.input(RoBPin)
    while(not GPIO.input(RoAPin)):
        Current_RoB_Status = GPIO.input(RoBPin)
        flag = 1
    if flag == 1:
```

(continues on next page)

(continued from previous page)

```
        flag = 0
        if (Last_RoB_Status == 0) and (Current_RoB_Status == 1):
            globalCounter = globalCounter + 1
        if (Last_RoB_Status == 1) and (Current_RoB_Status == 0):
            globalCounter = globalCounter - 1

def btnISR(channel):
    global globalCounter
    globalCounter = 0

def loop():
    global globalCounter
    tmp = 0 # Rotary Temperary

    GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=btnISR)
    while True:
        rotaryDeal()
        if tmp != globalCounter:
            print ('globalCounter = %d' % globalCounter)
            tmp = globalCounter

def destroy():
    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()
```

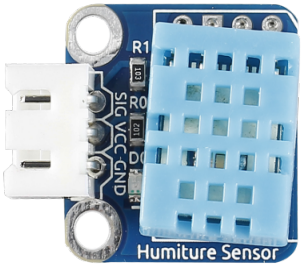
Now rotate the shaft of the rotary encoder, and the value printed on the screen will change. Rotate the rotary encoder clockwise, the value will increase; Rotate it counterclockwise, the value will decrease; Press the rotary encoder, the value will be reset to 0.



6.28 Lesson 28 Humiture Sensor

Introduction

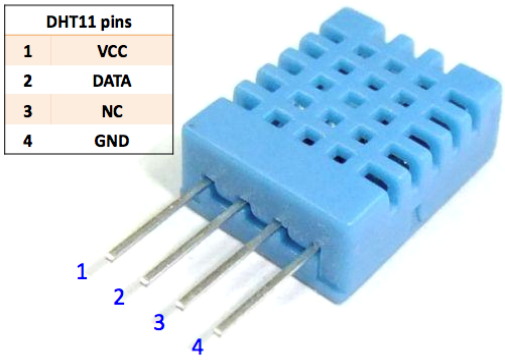
The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability.



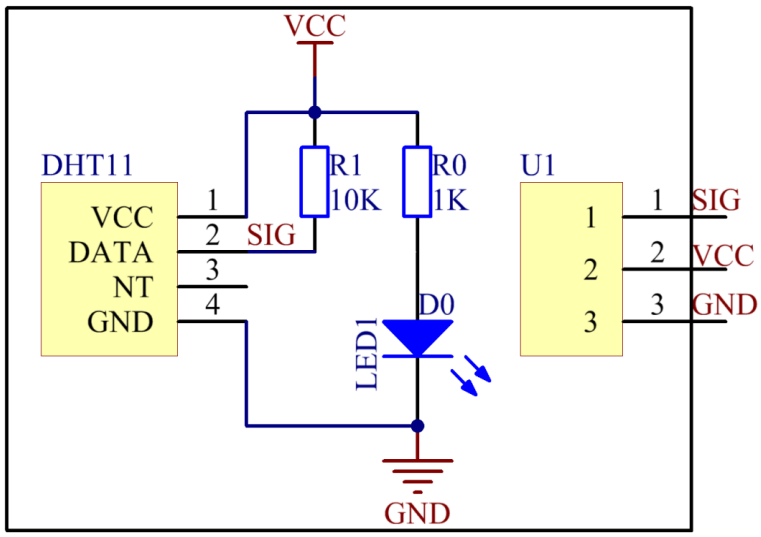
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Humiture module
- 1 * 3-Pin anti-reverse cable

Experimental Principle



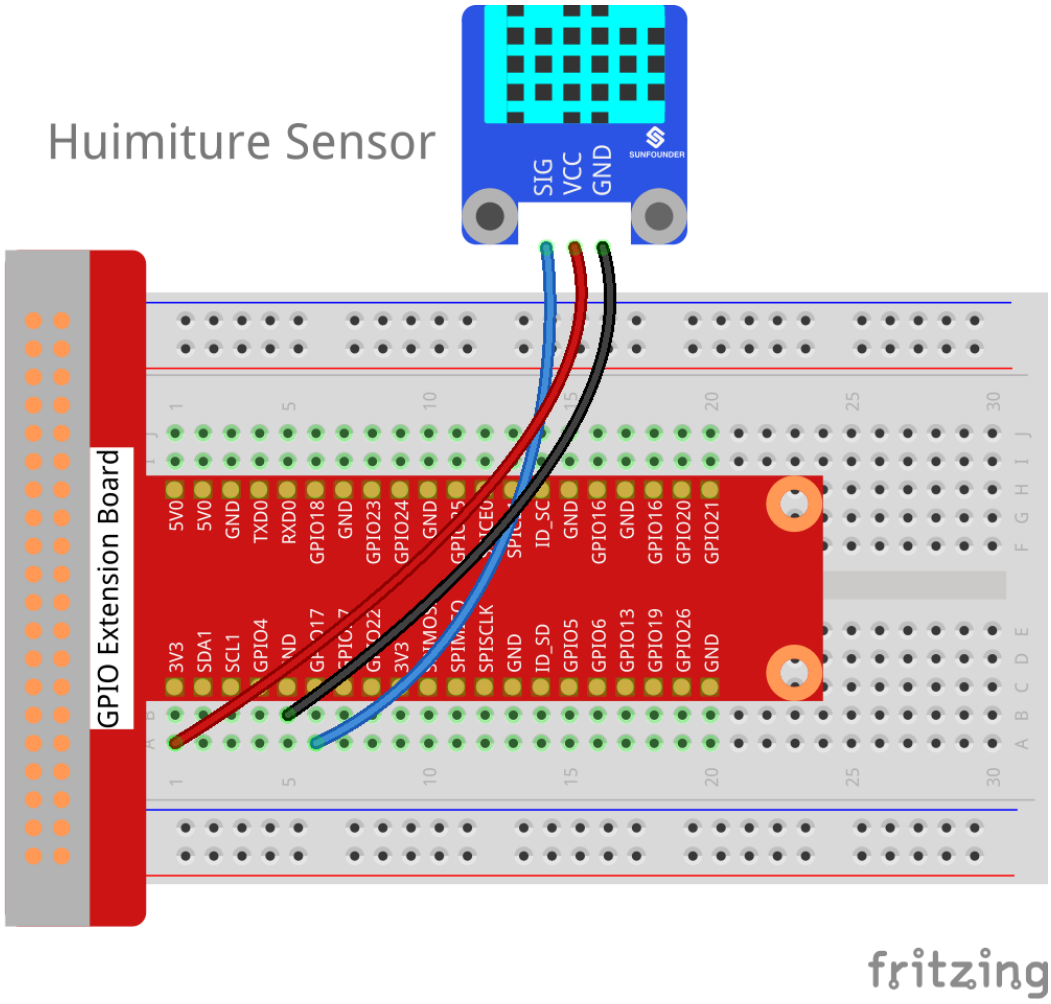
Only three pins are available for use: VCC, GND, and DATA. The communication process begins with the DATA line sending start signal to DHT11, and DHT11 receives the signal and returns an answer signal, then the host receives the answer signal and begins to receive 40-bit humiture data (8-bit humidity integer + 8-bit humidity decimal + 8-bit temperature integer + 8-bit temperature decimal + 8-bit checksum). For more information, please refer to the datasheet of DHT11.



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Humiture Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/28_humiture/
```

Step 3: Compile.

```
gcc humiture.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

(continues on next page)

(continued from previous page)

```
#include <stdint.h>

#define MAXTIMINGS 85

#define DHTPIN 0

int dht11_dat[5] = {0,0,0,0,0};

void read_dht11_dat()
{
    uint8_t laststate = HIGH;
    uint8_t counter = 0;
    uint8_t j = 0, i;
    float f; // fahrenheit

    dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;

    // pull pin down for 18 milliseconds
    pinMode(DHTPIN, OUTPUT);
    digitalWrite(DHTPIN, LOW);
    delay(18);
    // then pull it up for 40 microseconds
    digitalWrite(DHTPIN, HIGH);
    delayMicroseconds(40);
    // prepare to read the pin
    pinMode(DHTPIN, INPUT);

    // detect change and read data
    for ( i=0; i< MAXTIMINGS; i++) {
        counter = 0;
        while (digitalRead(DHTPIN) == laststate) {
            counter++;
            delayMicroseconds(1);
            if (counter == 255) {
                break;
            }
        }
        laststate = digitalRead(DHTPIN);

        if (counter == 255) break;

        // ignore first 3 transitions
        if ((i >= 4) && (i%2 == 0)) {
            // shove each bit into the storage bytes
            dht11_dat[j/8] <<= 1;
            if (counter > 16)
                dht11_dat[j/8] |= 1;
            j++;
        }
    }

    if ((j >= 40) &&
            (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_
↪dat[3]) & 0xFF)) ) {
        f = dht11_dat[2] * 9. / 5. + 32;
        printf("Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)\n",
                dht11_dat[0], dht11_dat[1], dht11_dat[2], dht11_dat[3], f);
```

(continues on next page)

(continued from previous page)

```
    }
}

int main (void)
{

    printf ("Raspberry Pi wiringPi DHT11 Temperature test program\n") ;

    if (wiringPiSetup () == -1)
        exit (1) ;

    while (1)
    {
        read_dht11_dat();
        delay(1000); // wait 1sec to refresh
    }

    return 0 ;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 28_humiture.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

DHTPIN = 17

GPIO.setmode(GPIO.BCM)

MAX_UNCHANGE_COUNT = 100

STATE_INIT_PULL_DOWN = 1
STATE_INIT_PULL_UP = 2
STATE_DATA_FIRST_PULL_DOWN = 3
STATE_DATA_PULL_UP = 4
STATE_DATA_PULL_DOWN = 5

def read_dht11_dat():
    GPIO.setup(DHTPIN, GPIO.OUT)
    GPIO.output(DHTPIN, GPIO.HIGH)
    time.sleep(0.05)
    GPIO.output(DHTPIN, GPIO.LOW)
    time.sleep(0.02)
    GPIO.setup(DHTPIN, GPIO.IN, GPIO.PUD_UP)

    unchanged_count = 0
```

(continues on next page)

(continued from previous page)

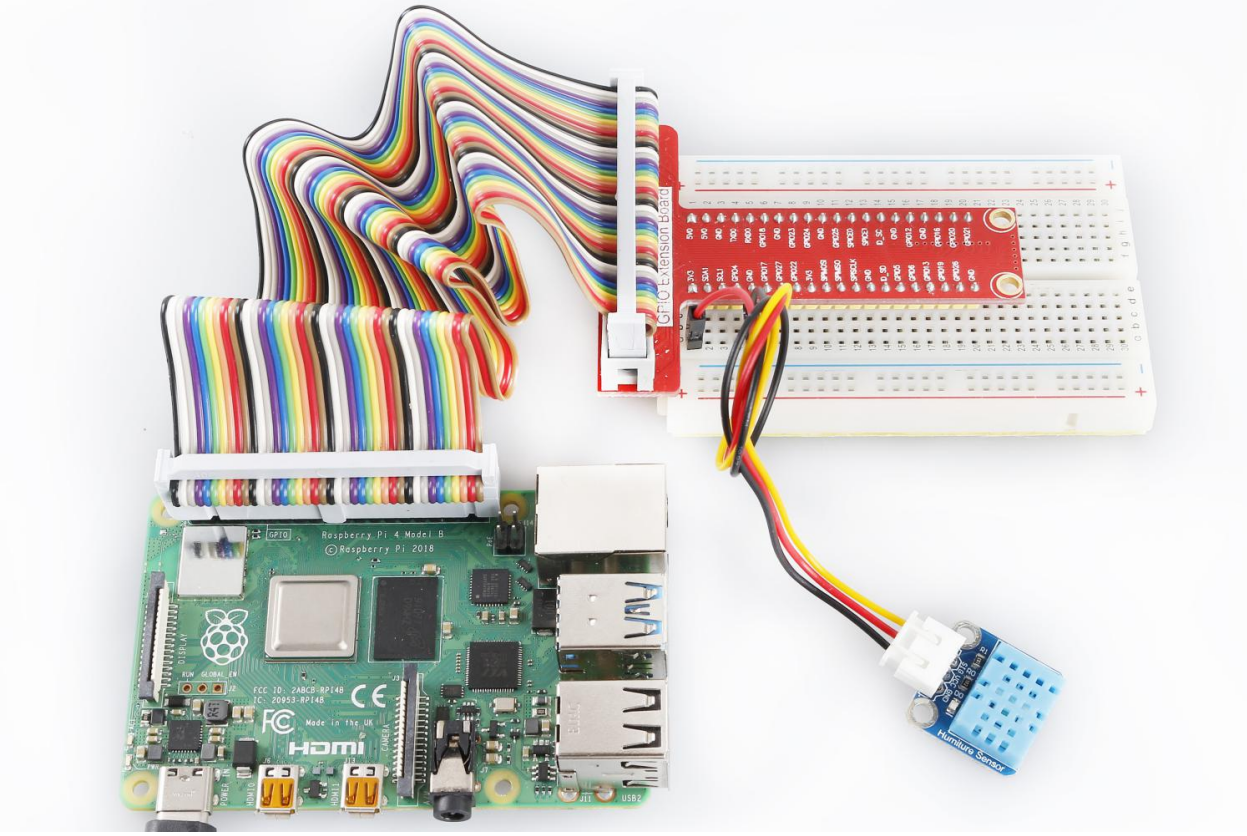
```
    last = -1
    data = []
    while True:
        current = GPIO.input(DHTPIN)
        data.append(current)
        if last != current:
            unchanged_count = 0
            last = current
        else:
            unchanged_count += 1
            if unchanged_count > MAX_UNCHANGE_COUNT:
                break

    state = STATE_INIT_PULL_DOWN

    lengths = []
    current_length = 0

    for current in data:
        current_length += 1

        if state == STATE_INIT_PULL_DOWN:
            if current == GPIO.LOW:
                state = STATE_INIT_PULL_UP
            else:
                continue
        if state == STATE_INIT_PULL_UP:
            if current == GPIO.HIGH:
                state = STATE_DATA_FIRST_PULL_DOWN
            else:
                continue
        if state == STATE_DATA_FIRST_PULL_DOWN:
            if current == GPIO.LOW:
                state = STATE_DATA_PULL_UP
            else:
                continue
        if state == STATE_DATA_PULL_UP:
            if current == GPIO.HIGH:
                current_length = 0
                state = STATE_DATA_PULL_DOWN
            else:
                continue
        if state == STATE_DATA_PULL_DOWN:
            if current == GPIO.LOW:
                lengths.append(current_length)
                state = STATE_DATA_PULL_UP
            else:
                continue
    if len(lengths) != 40:
        #print ("Data not good, skip")
        return False

    shortest_pull_up = min(lengths)
    longest_pull_up = max(lengths)
    halfway = (longest_pull_up + shortest_pull_up) / 2
    bits = []
    the_bytes = []
```

(continues on next page)

(continued from previous page)

```
    byte = 0

    for length in lengths:
        bit = 0
        if length > halfway:
            bit = 1
        bits.append(bit)
    #print ("bits: %s, length: %d" % (bits, len(bits)))
    for i in range(0, len(bits)):
        byte = byte << 1
        if (bits[i]):
            byte = byte | 1
        else:
            byte = byte | 0
        if ((i + 1) % 8 == 0):
            the_bytes.append(byte)
            byte = 0
    #print (the_bytes)
    checksum = (the_bytes[0] + the_bytes[1] + the_bytes[2] + the_bytes[3]) & 0xFF
    if the_bytes[4] != checksum:
        #print ("Data not good, skip")
        return False

    return the_bytes[0], the_bytes[2]

def main():
    print ("Raspberry Pi wiringPi DHT11 Temperature test program\n")
    while True:
        result = read_dht11_dat()
        if result:
            humidity, temperature = result
            print ("humidity: %s %%,  Temperature: %s C" % (humidity, temperature))
        time.sleep(1)

def destroy():
    GPIO.cleanup()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        destroy()
```

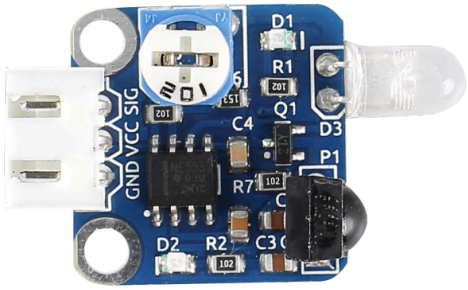
Now, you can see humidity and temperature value printed on the screen.



6.29 Lesson 29 IR Obstacle Avoidance Module

Introduction

An IR obstacle avoidance module (as shown below) is used in this Lesson.



Required Components

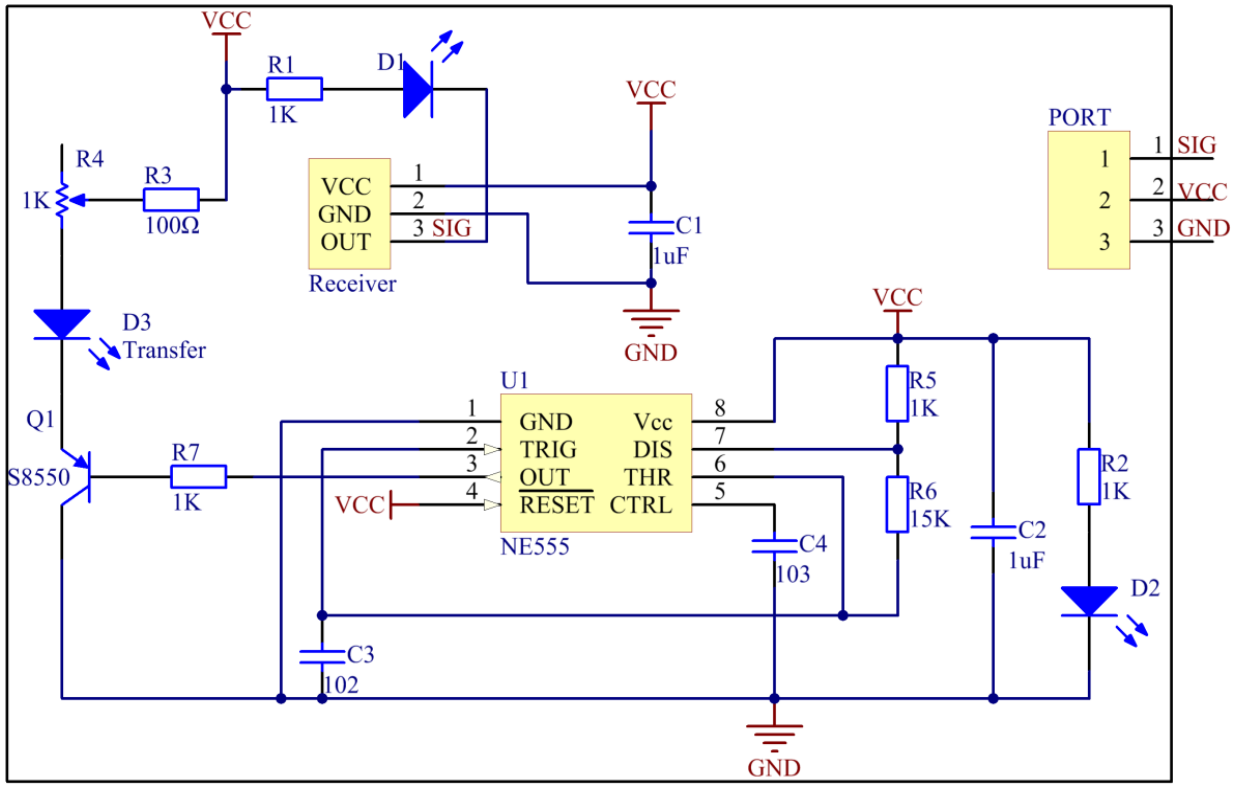
- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * IR Obstacle module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

An obstacle avoidance sensor mainly consists of an infrared-transmitter, an infrared-receiver and a potentiometer. According to the reflecting feature of an object, if there is no obstacle, emitted infrared ray will weaken with the propagation distance and finally disappear. If there is an obstacle, when infrared ray encounters an obstacle, it will be reflected back to the infrared-receiver. Then the infrared-receiver detects this signal and confirms an obstacle exists ahead.

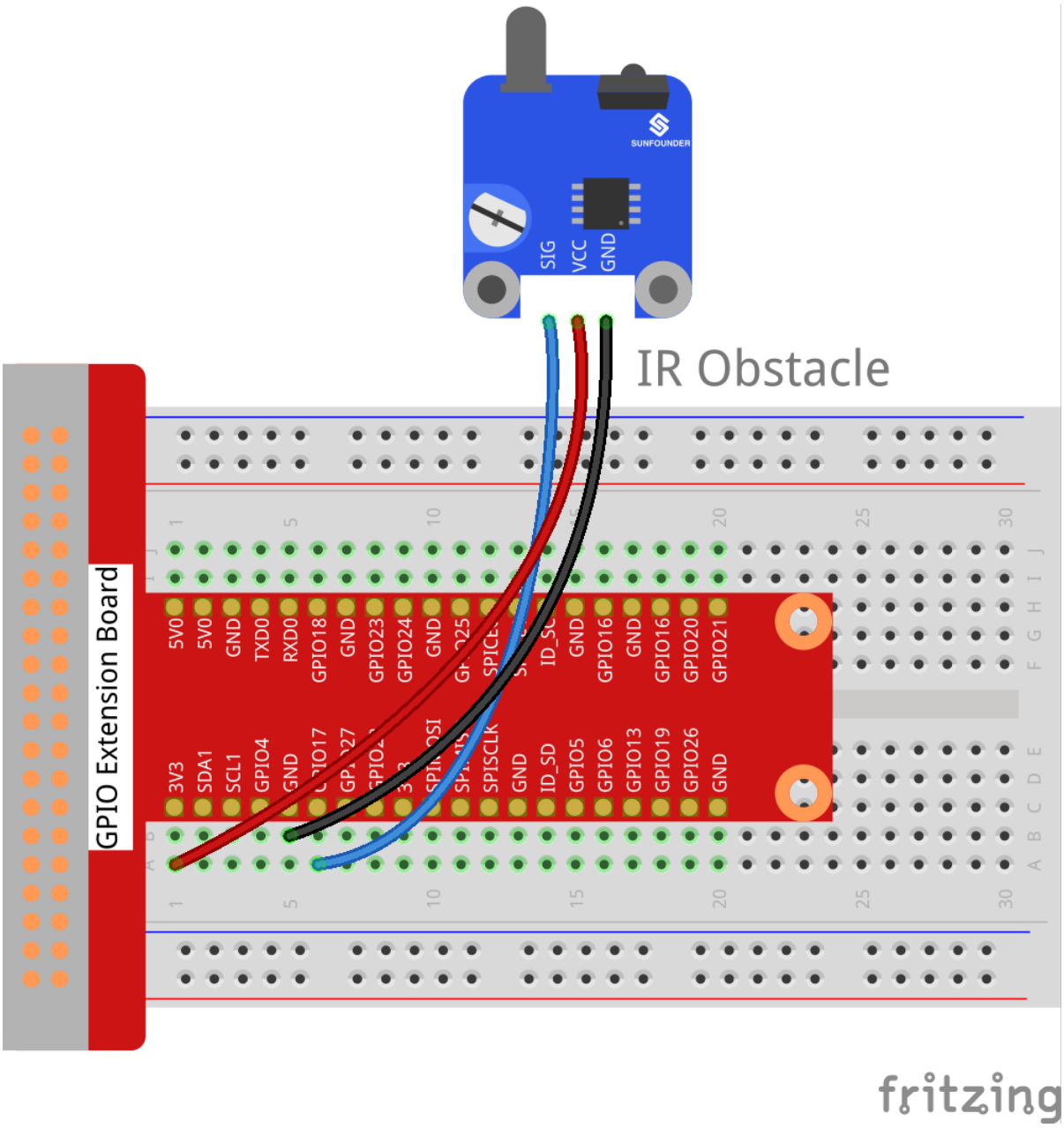
Note: The detection distance of the infrared sensor is adjustable - you may adjust it by the potentiometer.

The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit.



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/29_ir_obstacle/
```

Step 3: Compile.

```
gcc ir_obstacle.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define ObstaclePin      0

void myISR(void)
{
    printf("Detected Barrier !\n");
}

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

    if(wiringPiISR(ObstaclePin, INT_EDGE_FALLING, &myISR) < 0){
        printf("Unable to setup ISR !!!\n");
        return 1;
    }

    while(1){
        ;
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 29_ir_obstacle.py
```

Code

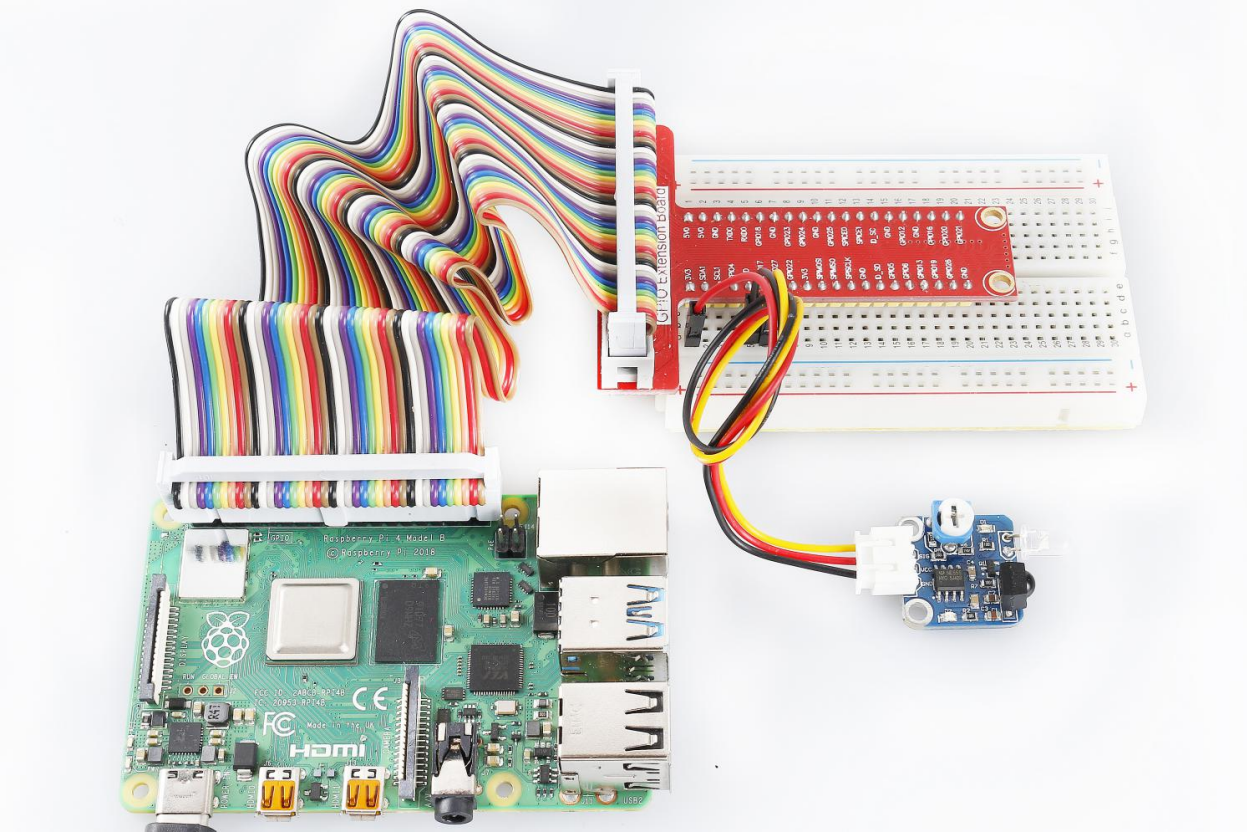
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

ObstaclePin = 11

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(ObstaclePin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def loop():
    while True:
```

(continues on next page)

(continued from previous page)

```
        if (0 == GPIO.input(ObstaclePin)):
            print ("Detected Barrier!")


def destroy():
    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()
```

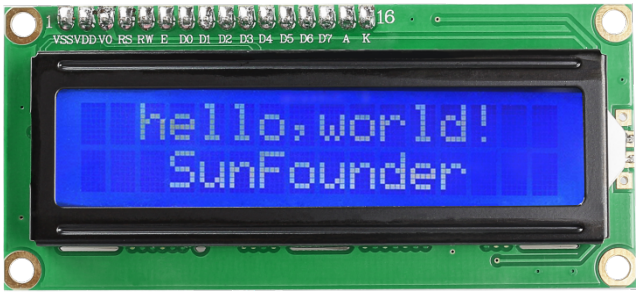
Now, if there is an obstacle ahead, a string “Detected Barrier!” will be printed on the screen.



6.30 Lesson 30 I2C LCD1602

Introduction

LCD1602 is a character type liquid crystal display, which can display 32 (16*2) characters at the same time. It has 16 pins, of which at least 7 would be used each time. You can use a PCF8574 I2C chip to expand I/O ports so only two GPIO ports would be occupied.



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * I2C LCD1602
- Several jumper wires

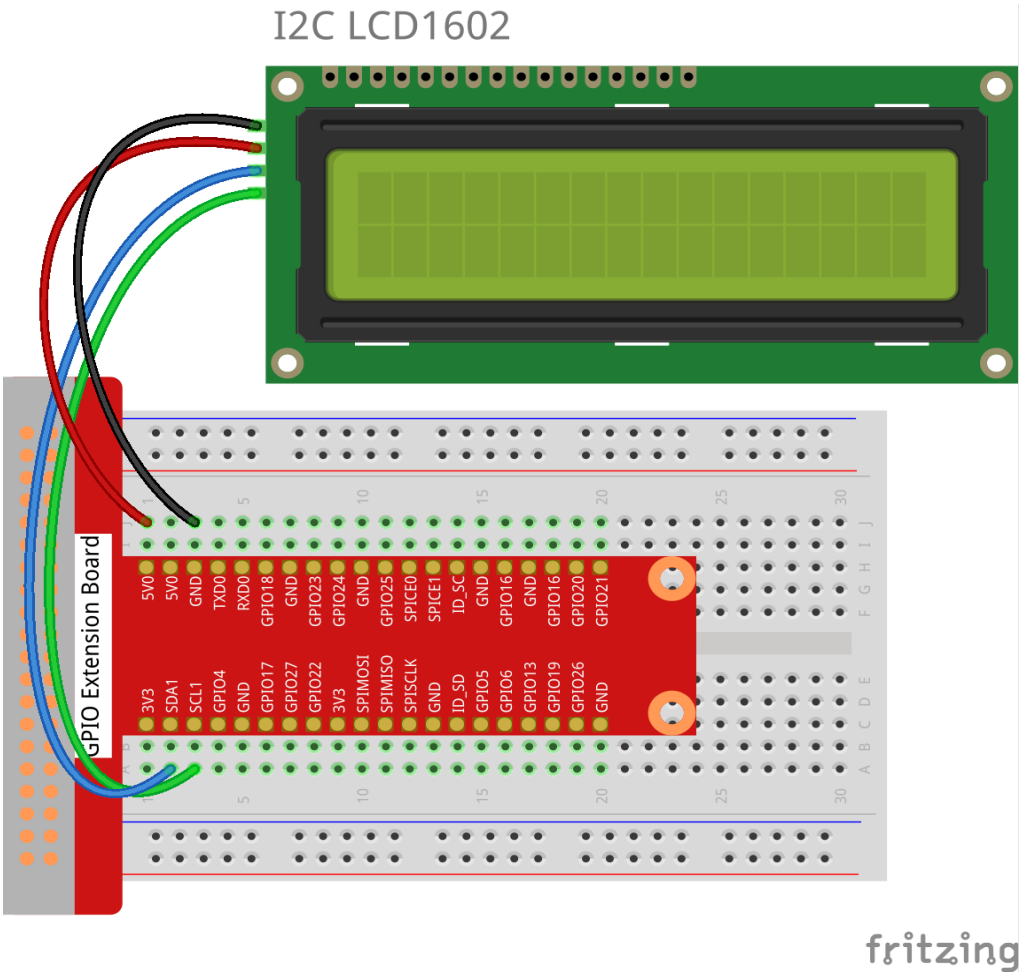
Experimental Principle

In this experiment, I2C is used to configure LCD so that you can control the LCD1602 to display characters. The I2C slave address of I2C LCD1602 here is 0x27.

Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	I2C LCD1602 Module
SCL	SCL1	SCL
SDA	SDA1	SDA
5V	5V0	VCC
GND	GND	GND



Step 2: Setup I2C(see Appendix. If you have set I2C, skip this step.)

For C Users:

Step 3: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/30_i2c_lcd1602/
```

Step 4: Compile.

```
gcc i2c_lcd1602.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 5: Run.

```
sudo ./a.out
```

Note:

- You can try screwing the potentiometer on the back if the code and wiring are fine, but the LCD still does not show the content.

Code

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <string.h>

int LCDAddr = 0x27;
int BLEN = 1;
int fd;

void write_word(int data){
    int temp = data;
    if ( BLEN == 1 )
        temp |= 0x08;
    else
        temp &= 0xF7;
    wiringPiI2CWrite(fd, temp);
}

void send_command(int comm){
    int buf;
    // Send bit7-4 firstly
    buf = comm & 0xF0;
    buf |= 0x04;                    // RS = 0, RW = 0, EN = 1
    write_word(buf);
    delay(2);
    buf &= 0xFB;                    // Make EN = 0
    write_word(buf);

    // Send bit3-0 secondly
    buf = (comm & 0x0F) << 4;
    buf |= 0x04;                    // RS = 0, RW = 0, EN = 1
    write_word(buf);
    delay(2);
    buf &= 0xFB;                    // Make EN = 0
    write_word(buf);
}

void send_data(int data){
    int buf;
    // Send bit7-4 firstly
    buf = data & 0xF0;
    buf |= 0x05;                    // RS = 1, RW = 0, EN = 1
    write_word(buf);
    delay(2);
    buf &= 0xFB;                    // Make EN = 0
    write_word(buf);

    // Send bit3-0 secondly
    buf = (data & 0x0F) << 4;
    buf |= 0x05;                    // RS = 1, RW = 0, EN = 1
    write_word(buf);
    delay(2);
    buf &= 0xFB;                    // Make EN = 0
    write_word(buf);
```

(continues on next page)

(continued from previous page)

```
}

void init(){
    send_command(0x33);      // Must initialize to 8-line mode at first
    delay(5);
    send_command(0x32);      // Then initialize to 4-line mode
    delay(5);
    send_command(0x28);      // 2 Lines & 5*7 dots
    delay(5);
    send_command(0x0C);      // Enable display without cursor
    delay(5);
    send_command(0x01);      // Clear Screen
    wiringPiI2CWrite(fd, 0x08);
}

void clear(){
    send_command(0x01);      //clear Screen
}

void write(int x, int y, char data[]){
    int addr, i;
    int tmp;
    if (x < 0)  x = 0;
    if (x > 15) x = 15;
    if (y < 0)  y = 0;
    if (y > 1)  y = 1;

    // Move cursor
    addr = 0x80 + 0x40 * y + x;
    send_command(addr);

    tmp = strlen(data);
    for (i = 0; i < tmp; i++){
        send_data(data[i]);
    }
}

void main(){
    fd = wiringPiI2CSetup(LCDAddr);
    init();
    write(0, 0, "Greetings!");
    write(1, 1, "From SunFounder");
    delay(2000);
    clear();
}
```

For Python Users:

Step 3: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 4: Run.

```
sudo python3 30_i2c_lcd1602.py
```

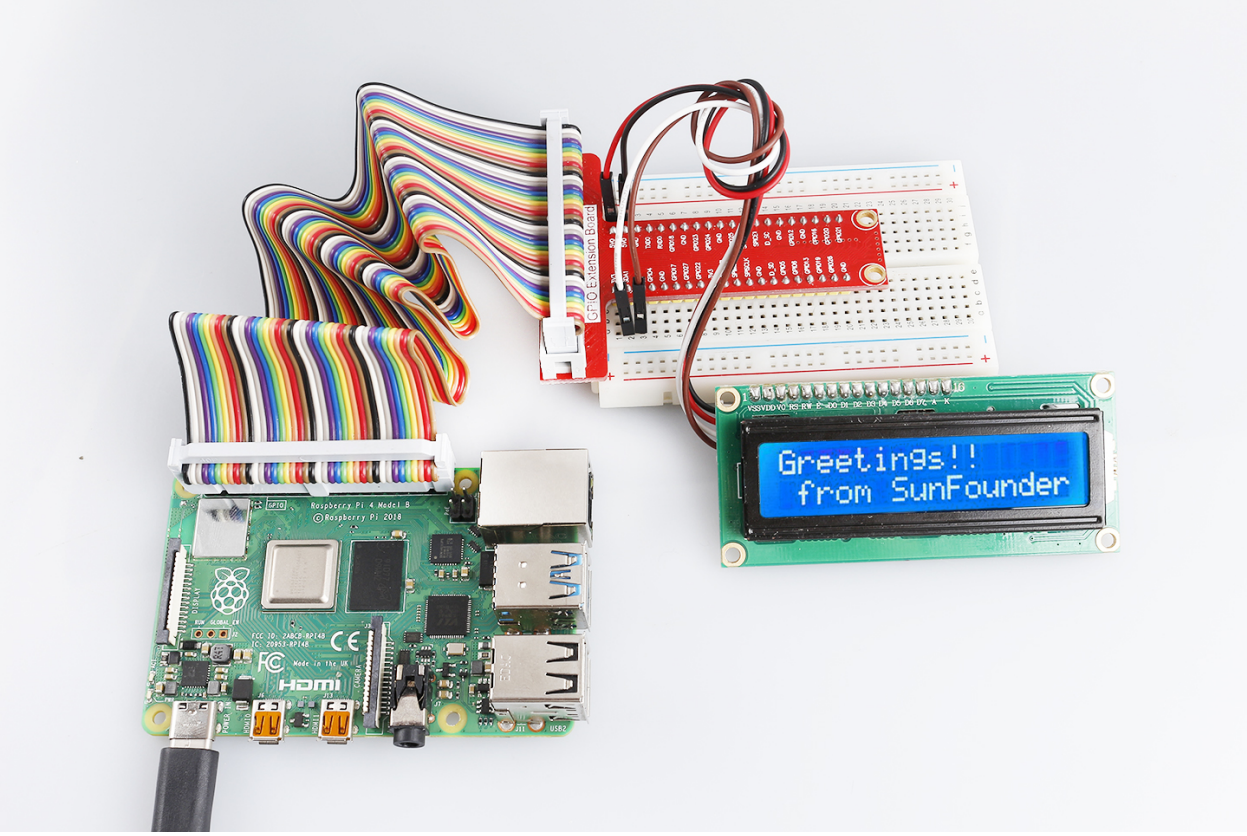
Note:

- If you get the error `FileNotFoundError: [Errno 2] No such file or directory: '/dev/i2c-1'`, you need to setup I2C (see Appendix -> I2C Configuration).
- If you get `ModuleNotFoundError: No module named 'smbus2'` error, please run the command: `sudo pip3 install smbus2`.
- If the error `OSError: [Errno 121] Remote I/O` appears, it means the module is miswired or the module is broken.
- If the module is connected correctly and still has the error `TimeoutError: [Errno 110] Connection timed out`, it means that the module is broken, please contact service@sunfounder.com. It is also possible to test if the I2C address appears with the command `i2cdetect -y 1` if you have the I2C tools installed (`sudo apt-get install i2c-tools`).

Code

```
#!/usr/bin/env python3
import LCD1602
import time

def setup():
    LCD1602.init(0x27, 1)    # init(slave address, background light)
    LCD1602.write(0, 0, 'Greetings!!')
    LCD1602.write(1, 1, 'from SunFounder')
    time.sleep(2)

def loop():
    space = '                '
    greetings = 'Thank you for buying SunFounder Sensor Kit for Raspberry! ^_^'
    greetings = space + greetings
    while True:
        tmp = greetings
        for i in range(0, len(greetings)):
            LCD1602.write(0, 0, tmp)
            tmp = tmp[1:]
            time.sleep(0.8)
            LCD1602.clear()

def destroy():
    pass

if __name__ == "__main__":
    try:
        setup()
        #loop()
        while True:
            pass
    except KeyboardInterrupt:
        destroy()
```

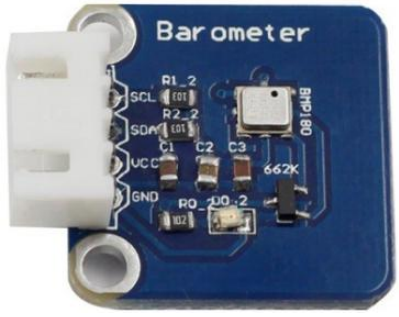
Now you can see “Greetings! From SunFounder” displayed on the LCD.



6.31 Lesson 31 Barometer-BMP180 Module

Introduction

The BMP180 barometer is the new digital barometric pressure sensor, with a very high performance, which enables applications in advanced mobile devices, such as smart phones, tablets and sports devices. It complies with the BMP085 but boasts many improvements, like a smaller size and more digital interfaces.

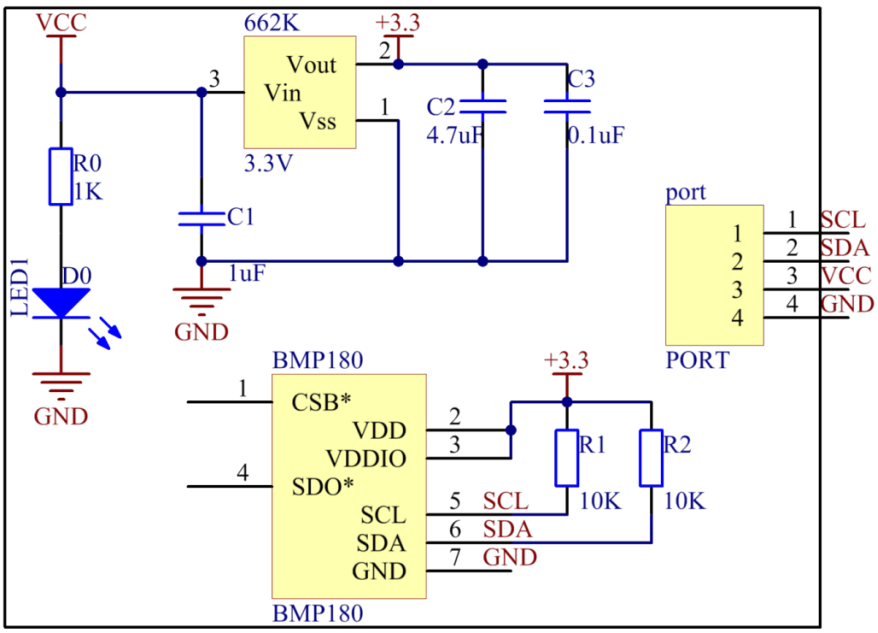


Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Barometer module
- 1 * 4-Pin anti-reverse cable

Experimental Principle

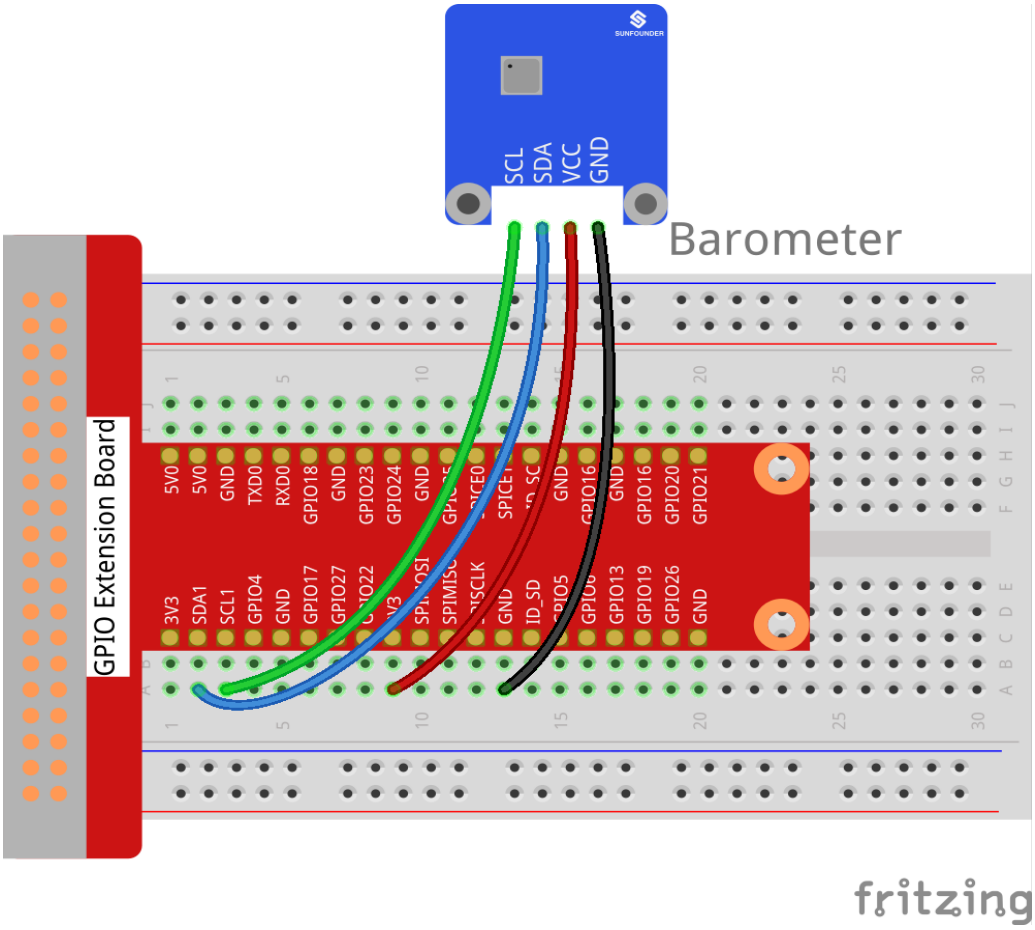
Use a barometer to measure air pressure and temperature. The schematic diagram of the module is as follows:



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Barometer
SCL	SCL1	SCL
SDA	SDA1	SDA
3.3V	3V3	VCC
GND	GND	GND



Step 2: Setup I2C (see Appendix . If you have set I2C, skip this step.)

For C Users:

Step 3: Download libi2c-dev.

```
sudo apt-get install libi2c-dev
```

Step 4: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/31_barometer/
```

Step 5: Compile.

```
gcc barometer.c bmp180.c -lm -lwiringPi -lwiringPiDev
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 6: Run.

```
sudo ./a.out
```

Note:

- If you get the error "`Unable to open I2C device: No such file or directory`", you need to setup I2C (see *I2C Configuration*).

Code

```
#include "bmp180.h"
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv){
    char *i2c_device = "/dev/i2c-1";
    int address = 0x77;

    void *bmp = bmp180_init(address, i2c_device);

    if(bmp != NULL){
        int i;
        for(i = 0; i < 100; i++) {
            float t = bmp180_temperature(bmp);
            long p = bmp180_pressure(bmp);
            float alt = bmp180_altitude(bmp);
            printf("temperature = %.2f, pressure = %lu, altitude = %.2f\n", t, p,␣
↪alt);
            usleep(2 * 1000 * 1000);
        }
    }
    return 0;
}
```

For Python Users:

Step 3: We'll need to install some utilities for the Raspberry Pi to communicate over I2C.

```
git clone https://github.com/adafruit/Adafruit_Python_BMP.git
cd Adafruit_Python_BMP
sudo python3 setup.py install
```

Step 4: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 5: Run.

```
sudo python3 31_barometer.py
```

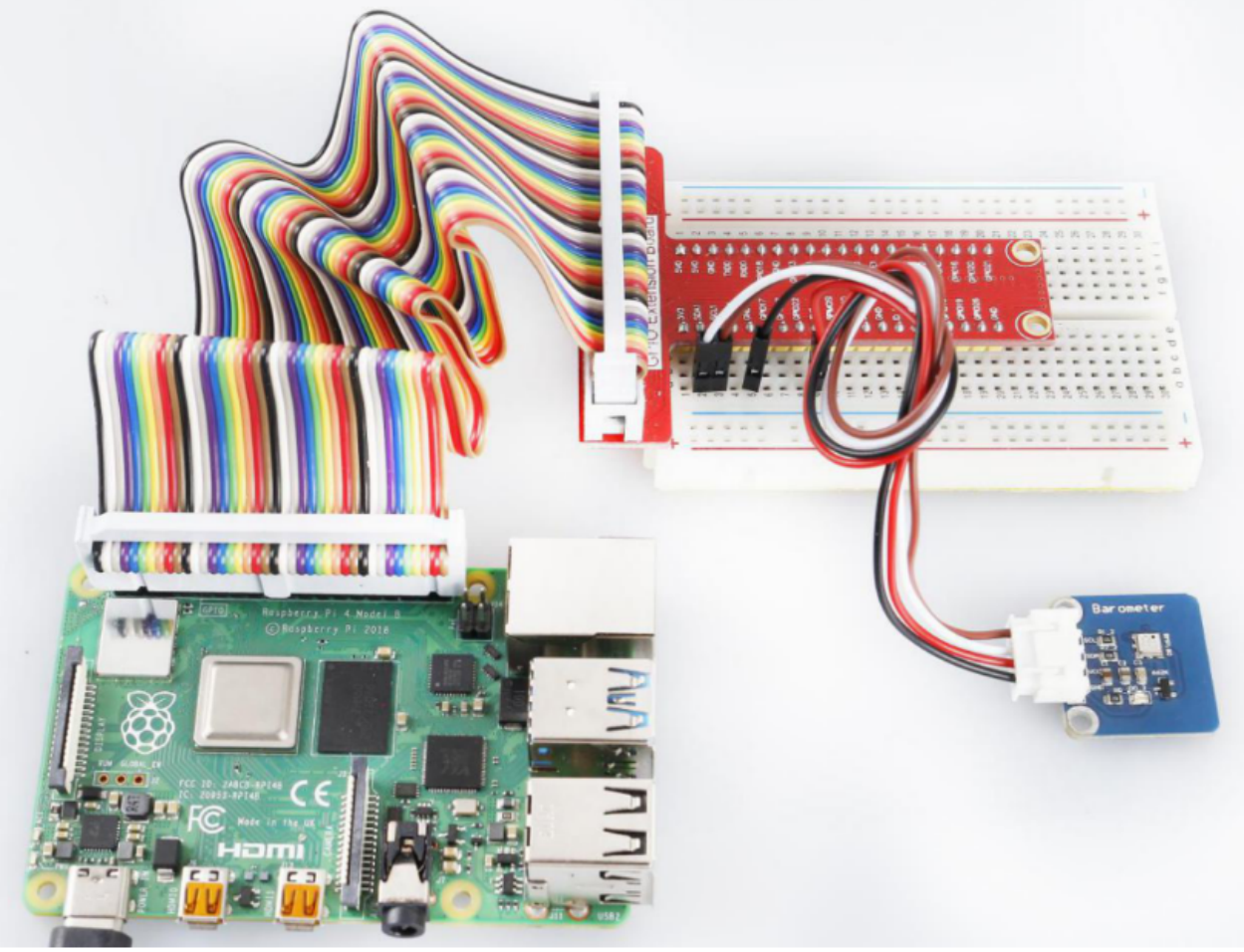
Note:

- If you get the error `FileNotFoundError: [Errno 2] No such file or directory: '/dev/i2c-1'`, you need to setup I2C (see Appendix -> I2C Configuration).
- If the error `OSError: [Errno 121] Remote I/O` appears, it means the module is miswired or the module is broken.
- If the module is connected correctly and still has the error `TimeoutError: [Errno 110] Connection timed out`, it means that the module is broken, please contact service@sunfounder.com. It is also possible to test if the I2C address appears with the command `i2cdetect -y 1` if you have the I2C tools installed (`sudo apt-get install i2c-tools`).

Code

```
import Adafruit_BMP.BMP085 as BMP085
import time

def setup():
    print ('\n Barometer begins...')

def loop():
    while True:
        sensor = BMP085.BMP085()
        temp = sensor.read_temperature()    # Read temperature to veriable temp
        pressure = sensor.read_pressure()   # Read pressure to veriable pressure

        print ('')
        print ('      Temperature = {0:0.2f} C'.format(temp))               # Print␣
↪temperature
        print ('      Pressure = {0:0.2f} Pa'.format(pressure))     # Print pressure
        time.sleep(1)
        print ('')

def destroy():
    pass

if __name__ == '__main__':          # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

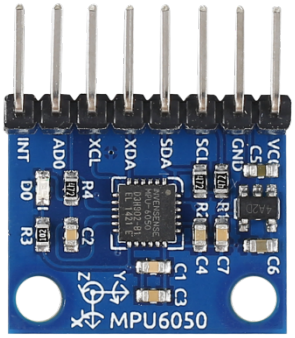
Now you can see the temperature and pressure value displayed on the screen.



6.32 Lesson 32 MPU6050 Gyro Acceleration Sensor

Introduction

The MPU-6050 is the world's first and only 6-axis motion tracking devices designed for the low power, low cost, and high performance requirements of smartphones, tablets and wearable sensors.



Required Components

- 1 * Raspberry Pi

- 1 * Breadboard
- 1 * MPU-6050 module
- Several Jumper wires

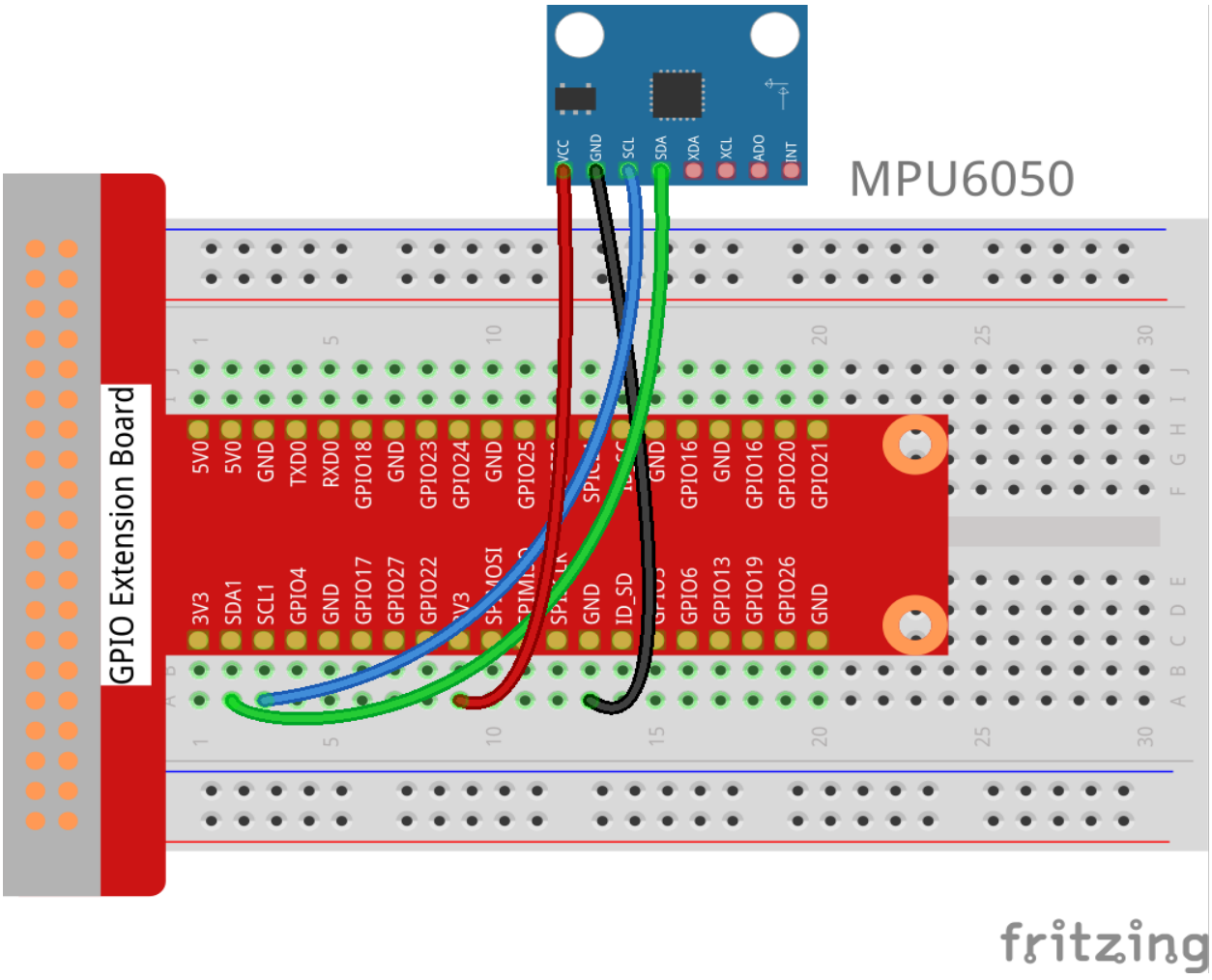
Experimental Principle

In this experiment, use I2C to obtain the values of the three-axis acceleration sensor and three-axis gyroscope for MPU6050 and display them on the screen.

Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	MPU-6050 Module
SCL	SCL1	SCL
SDA	SDA1	SDA
3.3V	3V3	VCC
GND	GND	GND



Step 2: Setup I2C (see Appendix. If you have set I2C, skip this step.)

For C Users:

Step 3: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/32_mpu6050/
```

Step 4: Compile.

```
gcc 32_mpu6050.c -lwiringPi -lm
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 5: Run.

```
sudo ./a.out
```

Code

```
#include  <wiringPiI2C.h>
#include <wiringPi.h>
#include  <stdio.h>
#include  <math.h>

int fd;
int acclX, acclY, acclZ;
int gyroX, gyroY, gyroZ;
double acclX_scaled, acclY_scaled, acclZ_scaled;
double gyroX_scaled, gyroY_scaled, gyroZ_scaled;

int read_word_2c(int addr)
{
int val;
val = wiringPiI2CReadReg8(fd, addr);
val = val << 8;
val += wiringPiI2CReadReg8(fd, addr+1);
if (val >= 0x8000)
    val = -(65536 - val);

return val;
}

double dist(double a, double b)
{
return sqrt((a*a) + (b*b));
}

double get_y_rotation(double x, double y, double z)
{
double radians;
radians = atan2(x, dist(y, z));
return -(radians * (180.0 / M_PI));
}

double get_x_rotation(double x, double y, double z)
{
double radians;
radians = atan2(y, dist(x, z));
```

(continues on next page)

(continued from previous page)

```
return (radians * (180.0 / M_PI));
}

int main()
{
fd = wiringPiI2CSetup (0x68);
wiringPiI2CWriteReg8 (fd,0x6B,0x00);//disable sleep mode
printf("set 0x6B=%X\n",wiringPiI2CReadReg8 (fd,0x6B));

while(1) {

    gyroX = read_word_2c(0x43);
    gyroY = read_word_2c(0x45);
    gyroZ = read_word_2c(0x47);

    gyroX_scaled = gyroX / 131.0;
    gyroY_scaled = gyroY / 131.0;
    gyroZ_scaled = gyroZ / 131.0;

    //Print values for the X, Y, and Z axes of the gyroscope sensor.
    printf("My gyroX_scaled: %f\n", gyroX_scaled);
    printf("My gyroY_scaled: %f\n", gyroY_scaled);
    printf("My gyroZ_scaled: %f\n", gyroZ_scaled);

    acclX = read_word_2c(0x3B);
    acclY = read_word_2c(0x3D);
    acclZ = read_word_2c(0x3F);

    acclX_scaled = acclX / 16384.0;
    acclY_scaled = acclY / 16384.0;
    acclZ_scaled = acclZ / 16384.0;

    //Print the X, Y, and Z values of the acceleration sensor.
    printf("My acclX_scaled: %f\n", acclX_scaled);
    printf("My acclY_scaled: %f\n", acclY_scaled);
    printf("My acclZ_scaled: %f\n", acclZ_scaled);

    printf("My X rotation: %f\n", get_x_rotation(acclX_scaled, acclY_scaled, acclZ_
↪scaled));
    printf("My Y rotation: %f\n", get_y_rotation(acclX_scaled, acclY_scaled, acclZ_
↪scaled));


    delay(100);
}
return 0;
}
```

For Python Users:

Step 3: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 4: Run.

```
sudo python3 32_mpu6050.py
```

Note:

- If you get the error `FileNotFoundError: [Errno 2] No such file or directory: '/dev/i2c-1'`, you need to setup I2C (see Appendix -> I2C Configuration).
- If you get `ModuleNotFoundError: No module named 'smbus2'` error, please run the command: `sudo pip3 install smbus2`.
- If the error `OSError: [Errno 121] Remote I/O` appears, it means the module is miswired or the module is broken.
- If the module is connected correctly and still has the error `TimeoutError: [Errno 110] Connection timed out`, it means that the module is broken, please contact service@sunfounder.com. It is also possible to test if the I2C address appears with the command `i2cdetect -y 1` if you have the I2C tools installed (`sudo apt-get install i2c-tools`).

Code

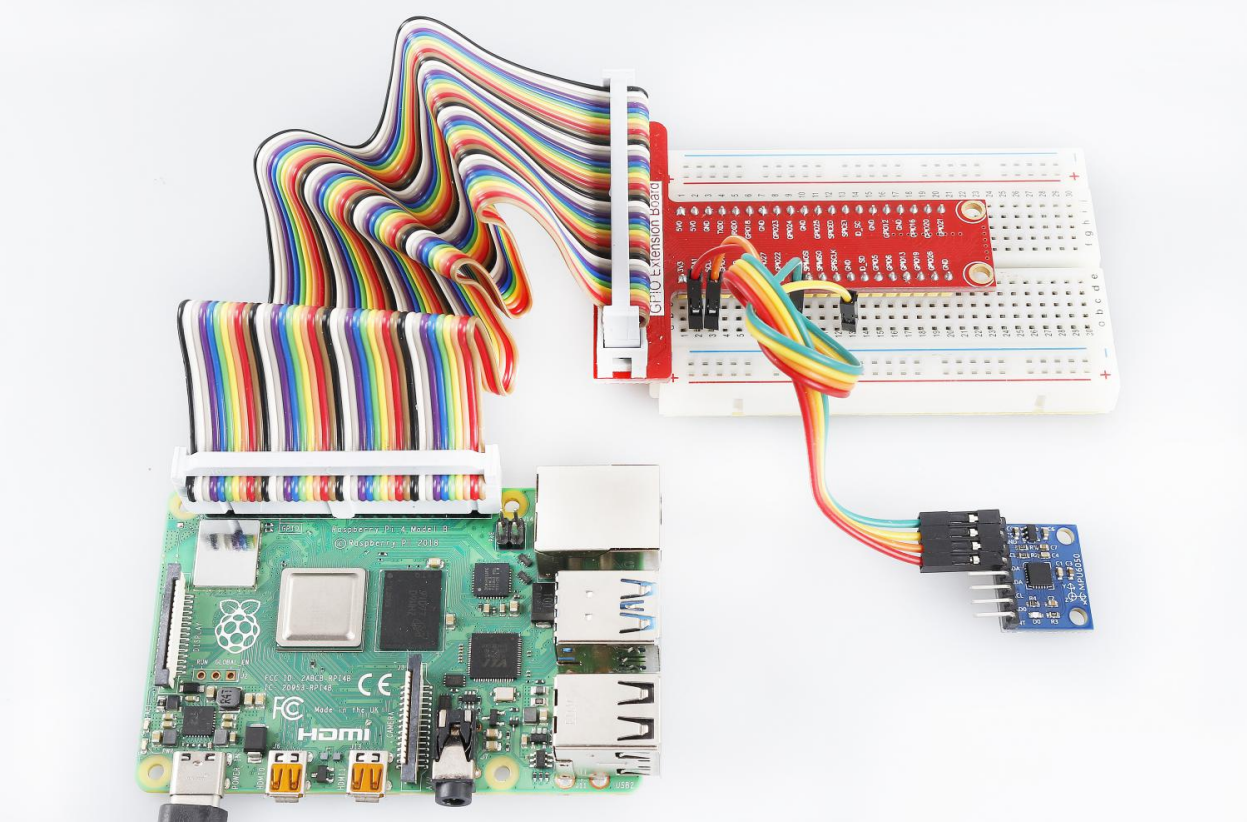
```
#!/usr/bin/python3
import smbus2 as smbus
import math
import time

# Power management registers
power_mgmt_1 = 0x6b
power_mgmt_2 = 0x6c

def read_byte(adr):
    return bus.read_byte_data(address, adr)

def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high << 8) + low
    return val

def read_word_2c(adr):
    val = read_word(adr)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val

def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)
```

(continues on next page)

(continued from previous page)

```
bus = smbus.SMBus(1) # or bus = smbus.SMBus(1) for Revision 2 boards
address = 0x68       # This is the address value read via the i2cdetect command

# Now wake the 6050 up as it starts in sleep mode
bus.write_byte_data(address, power_mgmt_1, 0)

while True:
    time.sleep(0.1)
    gyro_xout = read_word_2c(0x43)
    gyro_yout = read_word_2c(0x45)
    gyro_zout = read_word_2c(0x47)

    print ("gyro_xout : ", gyro_xout, " scaled: ", (gyro_xout / 131))
    print ("gyro_yout : ", gyro_yout, " scaled: ", (gyro_yout / 131))
    print ("gyro_zout : ", gyro_zout, " scaled: ", (gyro_zout / 131))

    accel_xout = read_word_2c(0x3b)
    accel_yout = read_word_2c(0x3d)
    accel_zout = read_word_2c(0x3f)

    accel_xout_scaled = accel_xout / 16384.0
    accel_yout_scaled = accel_yout / 16384.0
    accel_zout_scaled = accel_zout / 16384.0

    print ("accel_xout: ", accel_xout, " scaled: ", accel_xout_scaled)
    print ("accel_yout: ", accel_yout, " scaled: ", accel_yout_scaled)
    print ("accel_zout: ", accel_zout, " scaled: ", accel_zout_scaled)

    print ("x rotation: " , get_x_rotation(accel_xout_scaled, accel_yout_scaled,␣
↪accel_zout_scaled))
    print ("y rotation: " , get_y_rotation(accel_xout_scaled, accel_yout_scaled,␣
↪accel_zout_scaled))

    time.sleep(0.5)
```

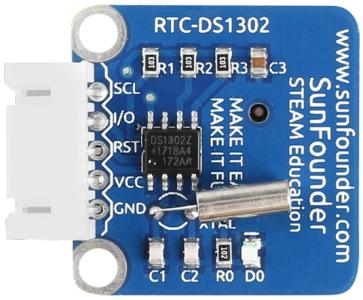
Now you can see the values of the acceleration sensor, gyroscope, and XY-axis rotation read by MPU6050 printed on the screen constantly.



6.33 Lesson 33 RTC DS1302

Introduction

DS1302 is a trickle charging clock chip, launched by DALLAS in America. With a built-in real-time clock/calendar and a 31-byte static RAM, it can communicate with MCU through simple serial interfaces. The real-time clock/calendar circuit provides information about second, minute, hour, day, week, month, and year. DS1302 can automatically adjust the number of days per month and days in leap year. You can determine to use a 24-hour or 12-hour system by AM/PM selection.



Required Components

- 1 * Raspberry Pi
- 1 * Breadboard

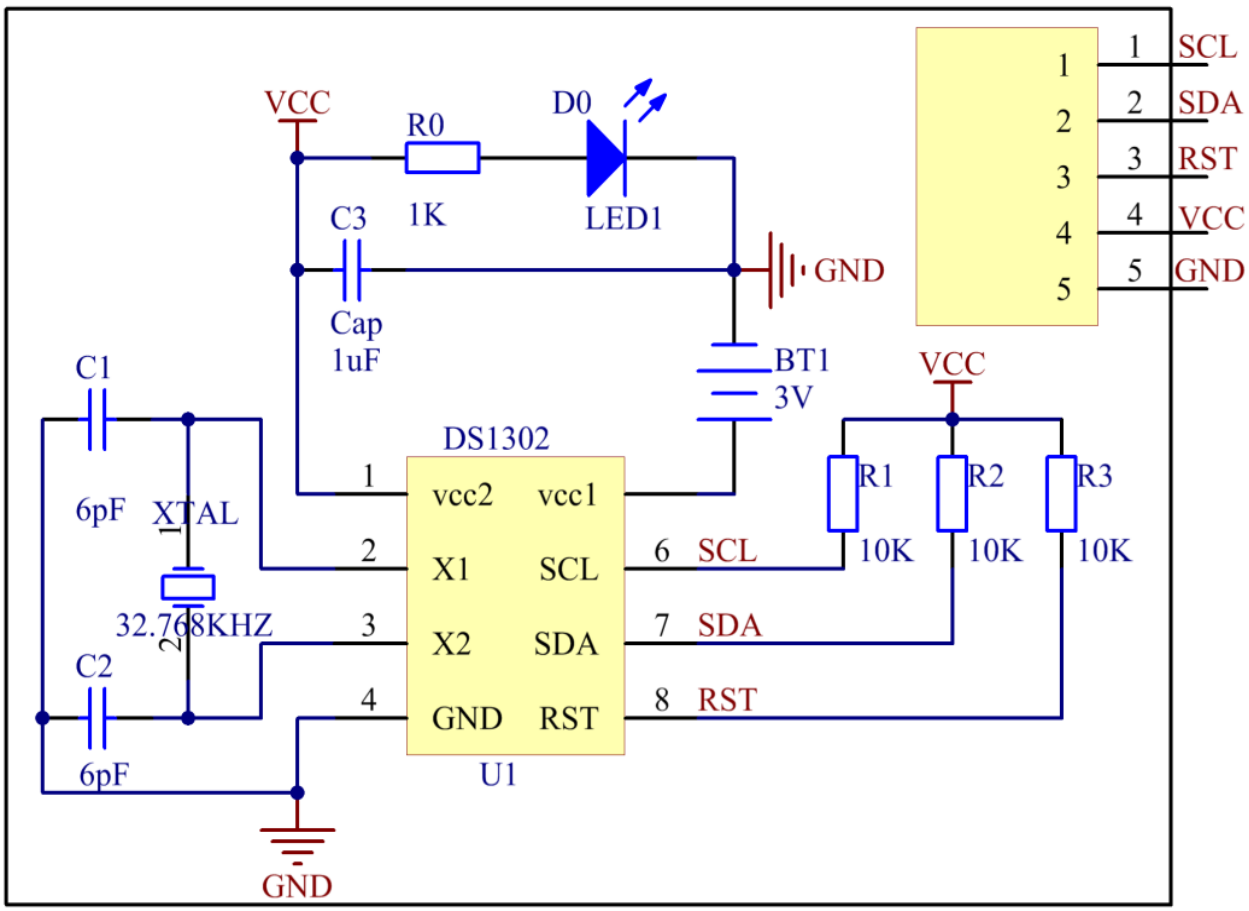
- 1 * DS1302 RTC module
- 1 * 5-Pin anti-reverse cable

Experimental Principle

Interfacing the DS1302 with a microprocessor is simplified by using synchronous serial communication. Only three wires are required to communicate with the clock/RAM: RST, serial data (SDA) and serial clock (SCL). SDA can be transferred to and from the clock/RAM one byte at a time or in a burst of up to 31 bytes.

After the time of the DS1302 is set manually, the MCU starts to read the accurate time and date returned by DS1302.

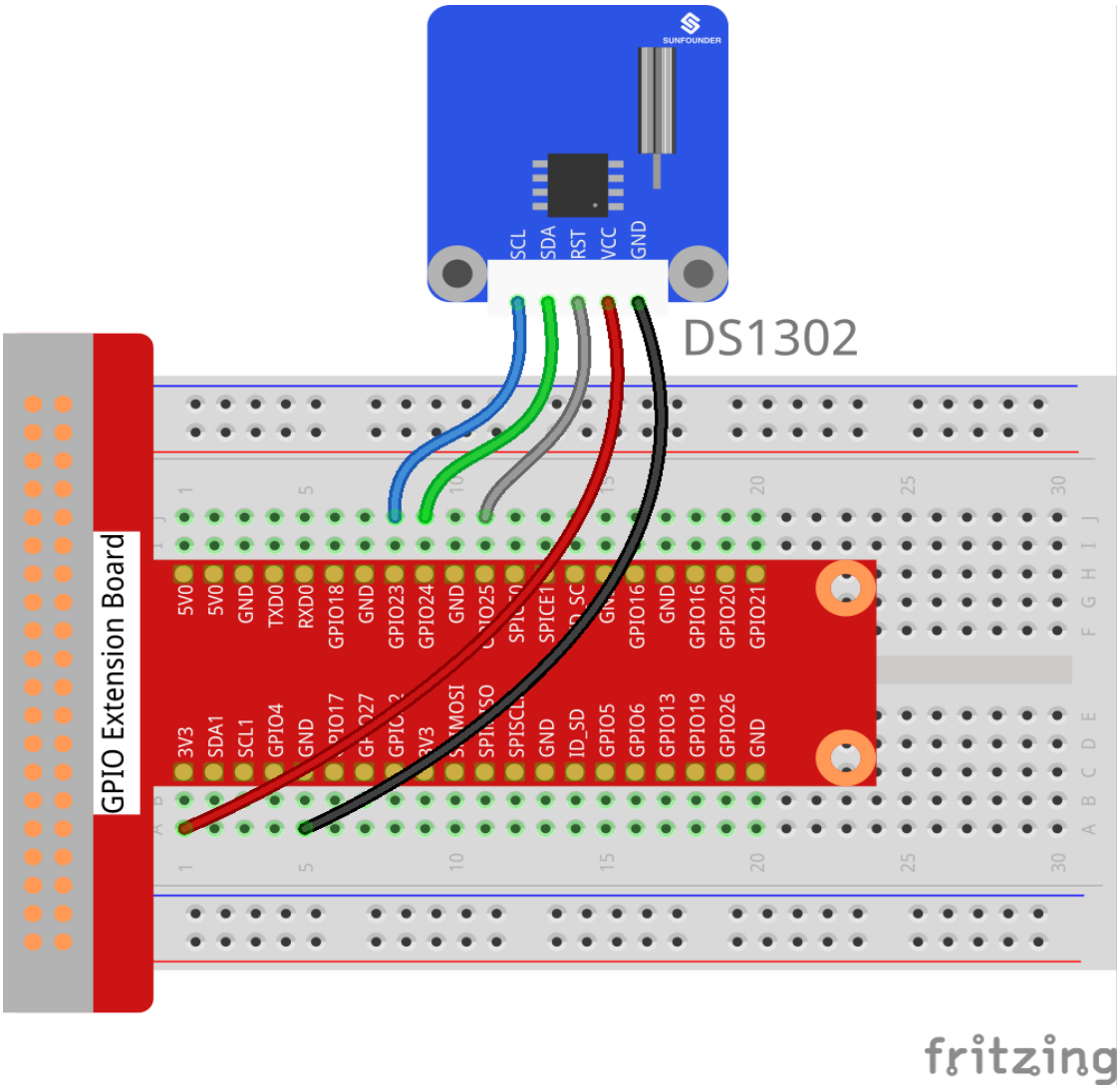
The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	RTC DS1302 Module
GPIO4	GPIO23	SCL
GPIO5	GPIO24	I/O or SDA
GPIO6	GPIO25	RST
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/33_ds1302/
```

Step 3: Compile:

```
gcc rtc_ds1302.c -lwiringPi -lwiringPiDev
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Set up time by:

```
sudo ./a.out -sdsc
```

Set year, month, date as YYYYMMDD

Set hour, minute, second as HHMMSS(24-hour clock)

Set weekday (0 as Sunday)

Step 5: Run:

```
sudo ./a.out
```

Code

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

#include <wiringPi.h>
#include <ds1302.h>

// Register defines

#define     RTC_SECS          0
#define     RTC_MINS          1
#define     RTC_HOURS         2
#define     RTC_DATE          3
#define     RTC_MONTH         4
#define     RTC_DAY           5
#define     RTC_YEAR          6
#define     RTC_WP            7
#define     RTC_TC            8
#define     RTC_BM           31

static unsigned int masks [] = { 0x7F, 0x7F, 0x3F, 0x3F, 0x1F, 0x07, 0xFF } ;

// bcdToD: dToBCD:
static int bcdToD (unsigned int byte, unsigned int mask)
{
unsigned int b1, b2 ;
byte &= mask ;
b1 = byte & 0x0F ;
b2 = ((byte >> 4) & 0x0F) * 10 ;
return b1 + b2 ;
}

static unsigned int dToBcd (unsigned int byte)
{
byte = byte % 100;
return ((byte / 10) << 4) + (byte % 10) ;
}

// ramTest:
static int ramTestValues [] =
{ 0x00, 0xFF, 0xAA, 0x55, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x00, 0xF0,␣
↪0x0F, -1 } ;

static int ramTest (void)
{
int addr ;
int got ;
int i = 0 ;
```

(continues on next page)

(continued from previous page)

```
int errors = 0 ;
int testVal ;

printf ("DS1302 RAM TEST\n") ;

testVal = ramTestValues [i] ;

while (testVal != -1)
{
    for (addr = 0 ; addr < 31 ; ++addr)
    ds1302ramWrite (addr, testVal) ;

    for (addr = 0 ; addr < 31 ; ++addr)
    if ((got = ds1302ramRead (addr)) != testVal)
    {
    printf ("DS1302 RAM Failure: Address: %2d, Expected: 0x%02X, Got: 0x%02X\n",
        addr, testVal, got) ;
    ++errors ;
    }
    testVal = ramTestValues [++i] ;
}

for (addr = 0 ; addr < 31 ; ++addr)
    ds1302ramWrite (addr, addr) ;

for (addr = 0 ; addr < 31 ; ++addr)
    if ((got = ds1302ramRead (addr)) != addr)
    {
    printf ("DS1302 RAM Failure: Address: %2d, Expected: 0x%02X, Got: 0x%02X\n",
        addr, addr, got) ;
    ++errors ;
    }

if (errors == 0)
    printf ("-- DS1302 RAM TEST: OK\n") ;
else
    printf ("-- DS1302 RAM TEST FAILURE. %d errors.\n", errors) ;

return 0 ;
}

// setLinuxClock:
static int setLinuxClock (void)
{
char dateTime [20] ;
char command [64] ;
int  clock [8] ;


printf ("Setting the Linux Clock from the DS1302... ") ; fflush (stdout) ;

ds1302clockRead (clock) ;

// [MMDDhhmm[[CC]YY][.ss]]

sprintf (dateTime, "%02d%02d%02d%02d%02d%02d.%02d",
    bcdToD (clock [RTC_MONTH], masks [RTC_MONTH]),
```

(continues on next page)

(continued from previous page)

```
    bcdToD (clock [RTC_DATE],  masks [RTC_DATE]),
    bcdToD (clock [RTC_HOURS], masks [RTC_HOURS]),
    bcdToD (clock [RTC_MINS],  masks [RTC_MINS]),
    20,
    bcdToD (clock [RTC_YEAR],  masks [RTC_YEAR]),
    bcdToD (clock [RTC_SECS],  masks [RTC_SECS])) ;

sprintf (command, "/bin/date %s", dateTime) ;
system (command) ;

return 0 ;
}

// setDSclock:
static int setDSclock (void)
{
struct tm t ;
time_t now ;
int clock [8] ;
int time = 0 ;
int date = 0 ;
int weekday = 0 ;

printf ("Setting the clock in the DS1302 from type in... ") ;

printf ("\n\nEnter Date(YYYYMMDD): ") ;
scanf ("%d", &date) ;
printf ("Enter time(HHMMSS, 24-hour clock): ") ;
scanf ("%d", &time) ;
printf ("Enter Weekday(0 as sunday): ") ;
scanf ("%d", &weekday) ;
//  printf("\ndate: %d,  time: %d\n\n", date, time) ;

clock [ 0] = dToBcd (time % 100) ;  // seconds
clock [ 1] = dToBcd (time / 100 % 100) ;    // mins
clock [ 2] = dToBcd (time / 100 / 100) ;    // hours
clock [ 3] = dToBcd (date % 100) ;  // date
clock [ 4] = dToBcd (date / 100 % 100) ;    // months 0-11 --> 1-12
clock [ 5] = dToBcd (weekday) ;     // weekdays (sun 0)
clock [ 6] = dToBcd (date / 100 / 100) ;       // years
clock [ 7] = 0 ;                    // W-Protect off

ds1302clockWrite (clock) ;

printf ("OK\n") ;

return 0 ;
}

int main (int argc, char *argv [])
{
int i ;
int clock [8] ;
int year ;
int month ;
int date ;
int hour ;
```

(continues on next page)

(continued from previous page)

```
int minute ;
int second ;
int weekday ;

wiringPiSetup () ;
ds1302setup   (4, 5, 6) ;

if (argc == 2)
{
    /**/ if (strcmp (argv [1], "-slc") == 0)
    return setLinuxClock () ;
    else if (strcmp (argv [1], "-sdsc") == 0)
    return setDSclock () ;
    else if (strcmp (argv [1], "-rtest") == 0)
    return ramTest () ;
    else
    {
    printf ("Usage: ds1302 [-slc | -sdsc | -rtest]\n") ;
    return EXIT_FAILURE ;
    }
}

for (i = 0 ;; ++i)
{
    printf ("%5d:  ", i) ;

    ds1302clockRead (clock) ;

    hour   = bcdToD (clock [2], masks [2]) ;
    minute = bcdToD (clock [1], masks [1]) ;
    second = bcdToD (clock [0], masks [0]) ;

    date  = bcdToD (clock [3], masks [3]) ;
    month = bcdToD (clock [4], masks [4]) ;
    year  = bcdToD (clock [6], masks [6]) + 2000 ;
    weekday = bcdToD (clock [5], masks [5]) ;

    printf ("  %04d-%02d-%02d", year, month, date) ;
    printf ("  %02d:%02d:%02d", hour, minute, second) ;

    switch (weekday){
    case 0: printf ("  SUN") ; break;
    case 1: printf ("  MON") ; break;
    case 2: printf ("  TUE") ; break;
    case 3: printf ("  WED") ; break;
    case 4: printf ("  THU") ; break;
    case 5: printf ("  FRI") ; break;
    case 6: printf ("  SAT") ; break;
    }

    printf ("\n") ;

    delay (200) ;
}
return 0 ;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

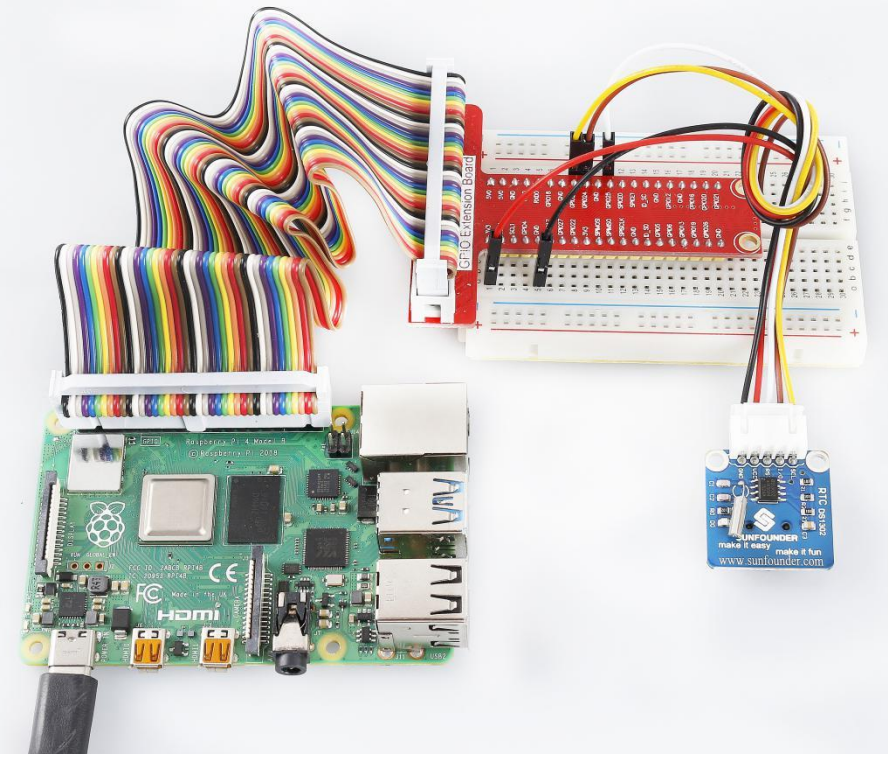
Step 3: Run.

```
sudo python3 33_ds1302.py
```

Code

```
#!/usr/bin/env python3
from datetime import datetime
from ds1302 import DS1302
from sys import version_info
import time

if version_info.major == 2:
    input = raw_input

rtc = DS1302()

def setup():
    print ('')
    print ('')
    print (rtc.get_datetime())
    print ('')
    print ('')
    a = input( "Do you want to setup date and time?(y/n) ")
    if a == 'y' or a == 'Y':
        date = input("Input date:(YYYY MM DD) ")
        time = input("Input time:(HH MM SS) ")
        date = list(map(lambda x: int(x), date.split()))
        time = list(map(lambda x: int(x), time.split()))
        print ('')
        print ('')
        rtc.set_datetime(datetime(date[0], date[1], date[2], time[0], time[1],␣
↪time[2]))
        dt = rtc.get_datetime()
        print ("You set the date and time to:", dt)

def loop():
    while True:
        a = rtc.get_datetime()
        print (a)
        time.sleep(0.5)

def destory():
    pass                            # 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.
        destory()
```

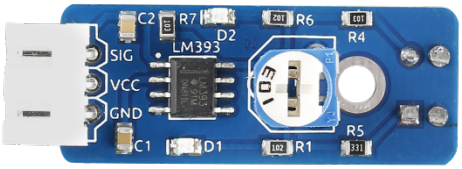
Now you can see the time on the screen.



6.34 Lesson 34 Tracking Sensor

Introduction

The infrared tracking sensor uses a TRT5000 sensor. The blue LED of TRT5000 is the emission tube and after electrified it emits infrared light invisible to human eye. The black part of the sensor is for receiving; the resistance of the resistor inside changes with the infrared light received.



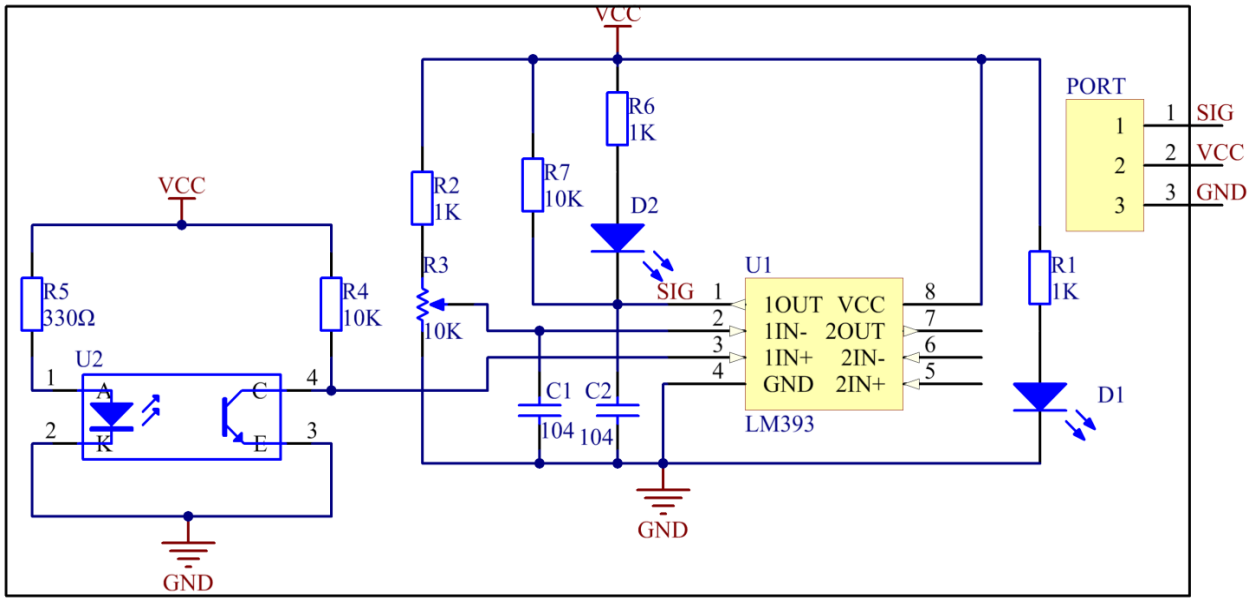
Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Tracking sensor module
- 1 * 3-Pin anti-reverse cable

Experimental Principle

When the infrared transmitter emits rays to a piece of paper, if the rays shine on a white surface, they will be reflected and received by the receiver, and pin SIG will output low level; If the rays encounter black lines, they will be absorbed,

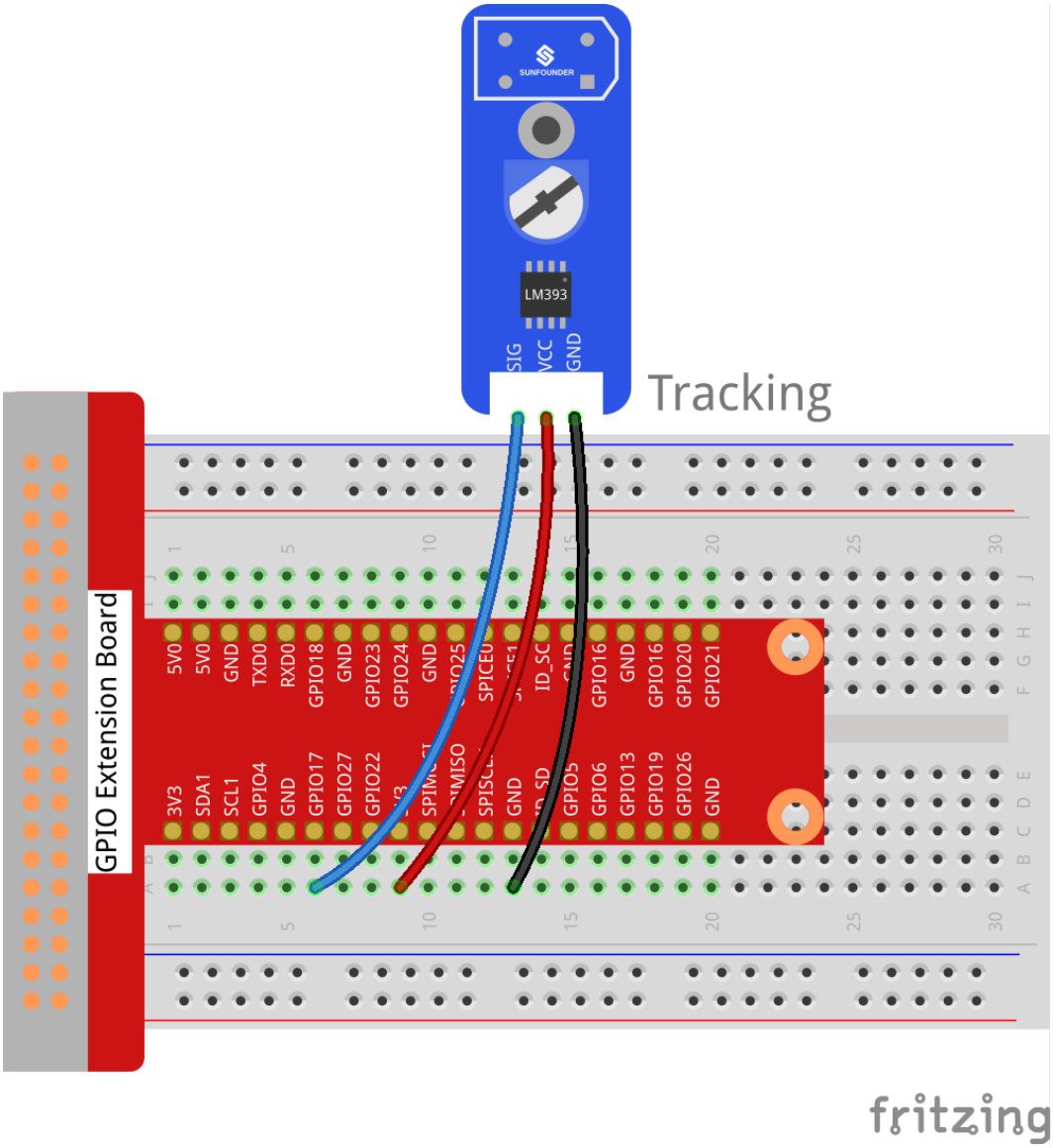
thus the receiver gets nothing, and pin SIG will output high level. The schematic diagram of the module is as shown below:



Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	GPIO Extension Board	Tracking Sensor Module
GPIO0	GPIO17	SIG
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/34_tracking/
```

Step 3: Compile.

```
gcc tracking.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 4: Run.

```
sudo ./a.out
```

Code

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

#define TrackSensorPin    0
#define LedPin            1

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

    pinMode(TrackSensorPin, INPUT);
    pinMode(LedPin,  OUTPUT);

    while(1){
        if(digitalRead(TrackSensorPin) == LOW){
            printf("White line is detected\n");
            digitalWrite(LedPin, LOW);     //led on
            delay(100);
            digitalWrite(LedPin, HIGH);    //led off
        }
        else{
            printf("...Black line is detected\n");
            delay(100);
        }
    }

    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 3: Run.

```
sudo python3 34_tracking.py
```

Code

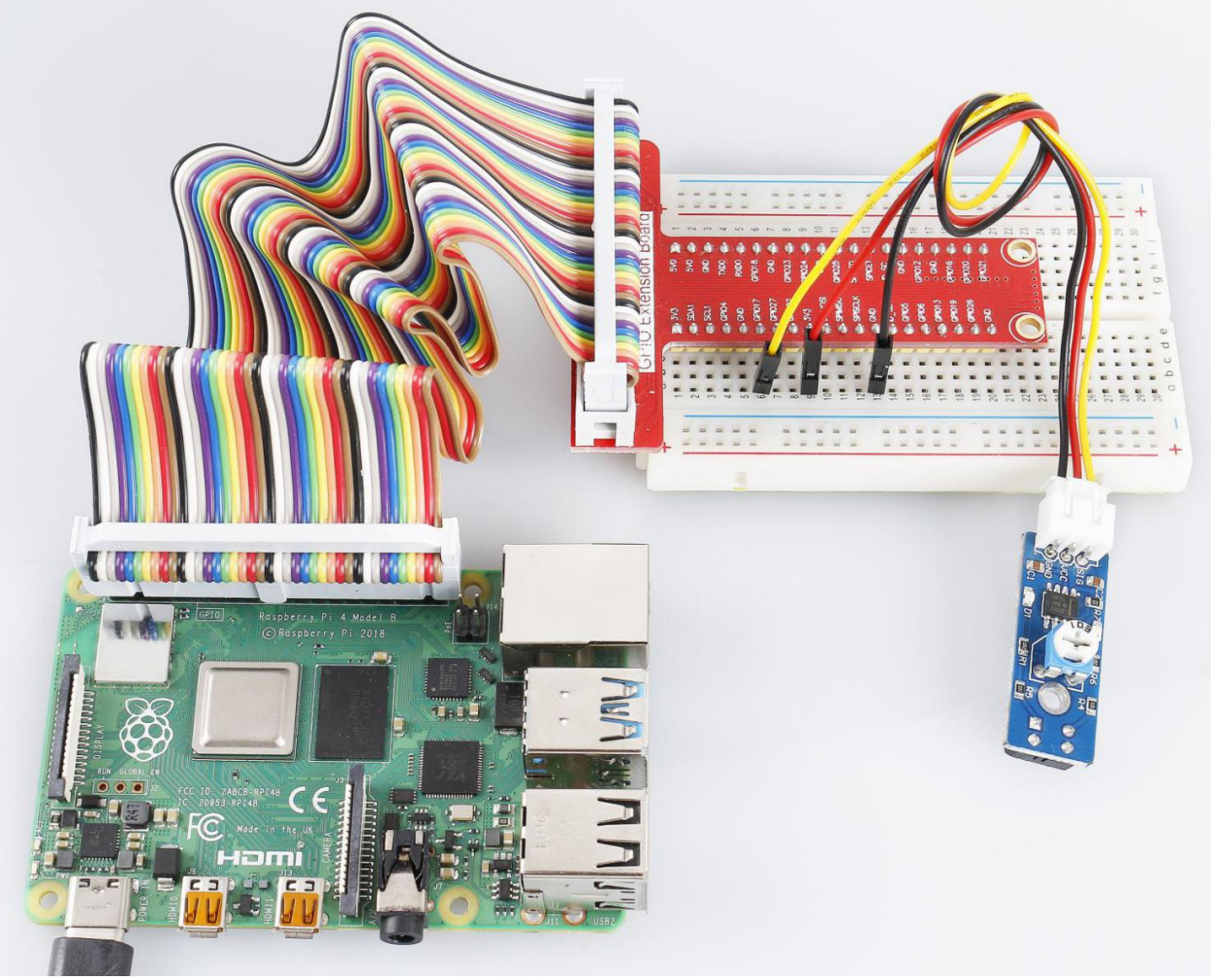
```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

TrackPin = 11
LedPin   = 12

def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(LedPin, GPIO.OUT)    # Set LedPin's mode is output
    GPIO.setup(TrackPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.output(LedPin, GPIO.HIGH) # Set LedPin high(+3.3V) to off led
```

(continues on next page)

(continued from previous page)

```
def loop():
    while True:
        if GPIO.input(TrackPin) == GPIO.LOW:
            print ('White line is detected')
            time.sleep(0.5)
            GPIO.output(LedPin, GPIO.LOW)  # led on
        else:
            print ('...Black line is detected')
            time.sleep(0.5)
            GPIO.output(LedPin, GPIO.HIGH) # led off

def destroy():
    GPIO.output(LedPin, GPIO.HIGH)     # led off
    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()
```

When the tracking sensor encounters black lines, a string “Black Line is detected” will be printed on the screen.



6.35 Lesson 35 Intelligent Temperature Measurement System

Introduction

In this experiment, we will use some modules together to build an intelligent temperature measurement system.

Required Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * Active Buzzer
- 1 * RGB LED Module
- 1 * DS18B20 Temperature Sensor
- 1 * PCF8591
- 1 * Joystick PS2
- Several Jumper wires

Experimental Principle

It is similar with lesson 26. The only difference is that we can adjust the lower limit and upper limit value by joystick PS2 when programming.

As mentioned previously, joystick PS2 has five operation directions: up, down, left, right and press-down. Well, in this experiment, we will use the left and right directions to control the upper limit value and up/down direction to control the lower limit. If you press down the joystick, the system will log out.

Experimental Procedures

Step 1: Build the circuit.

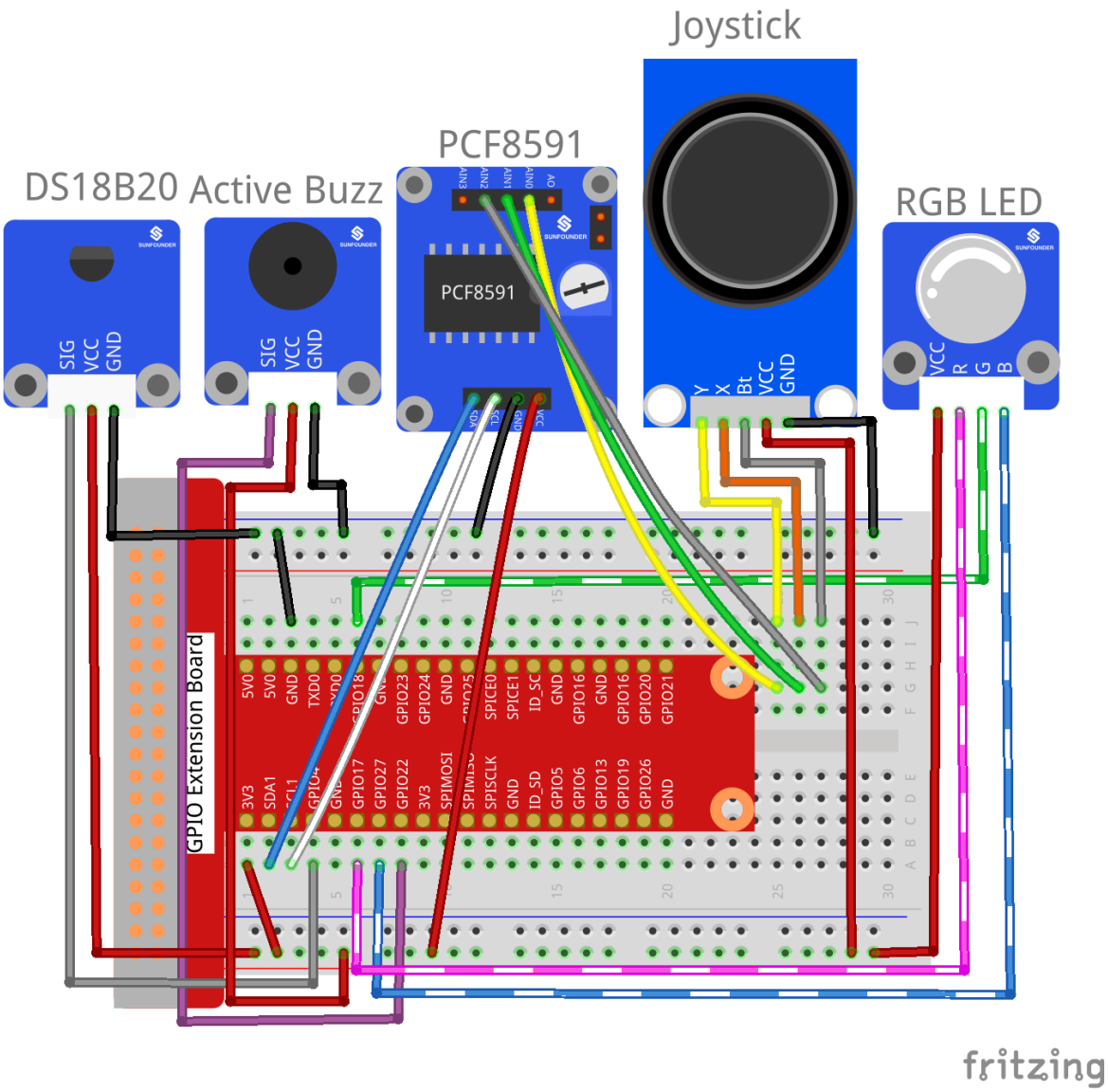
Raspberry Pi	GPIO Extension Board	DS18B20 Module
GPIO7	GPIO4	SIG
3.3V	3V3	VCC
GND	GND	GND

Raspberry Pi	GPIO Extension Board	PCF8591 Module
SDA	SDA1	SDA
SCL	SCL1	SCL
3.3V	3V3	VCC
GND	GND	GND

Joystick PS2	GPIO Extension Board	PCF8591 Module
Y	*	AIN0
X	*	AIN1
Bt	*	AIN2
VCC	3V3	*
GND	GND	*

Raspberry Pi	GPIO Extension Board	RGB LED Module
GPIO0	GPIO17	R
GPIO1	GPIO18	G
GPIO2	GPIO27	B
3.3V	3V3	VCC

Raspberry Pi	GPIO Extension Board	Active Buzzer Module
GPIO3	GPIO22	SIG
3.3V	3V3	VCC
GND	GND	GND



For C Users:

Step 2: Check the address of your sensor.

```
ls /sys/bus/w1/devices/
```

It may be like this:

```
28-031467805fff w1_bus_master1
```

Copy or write down 28-XXXXXXX. It is the address of your sensor.

Step 2: Change directory and edit.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/C/35_expand02/
nano temp_monitor.c
```

Find the function *float tempRead(void)*, and the line "fd = open(XXXXXX)". Replace "28-031467805ff" with your sensor address.

```
float tempRead(void)
{
float temp;
int i,j;
int fd;
int ret;

char buf[BUFSIZE];
char tempBuf[5];

fd = open("/sys/bus/w1/devices/28-031467805fff/w1_slave",O_RDONLY);

if(-1 == fd){
perror("open device file error");
return 1;
}
```

Save and exit.

Step 4: Compile.

```
gcc temp_monitor.c -lwiringPi
```

Note: If it does not work after running, or there is an error prompt `wiringPi.h: No such file or directory`, please refer to *WiringPi* to install it.

Step 5: Run.

```
sudo ./a.out
```

Code

```
#include <wiringPi.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <pcf8591.h>

#define     LedRed          0
#define     LedGreen        1
#define     LedBlue         2

#define     PCF                     120
#define             Beep            3
#define             BUFSIZE         128

typedef unsigned char uchar;
typedef unsigned int  uint;

static int sys_state = 1;

static int AIN0 = PCF + 0;
```

(continues on next page)

(continued from previous page)

```
static int AIN1 = PCF + 1;
static int AIN2 = PCF + 2;

void beepInit(void)
{
    pinMode(Beep, OUTPUT);
}

void beep_on(void)
{
    digitalWrite(Beep, LOW);
}

void beep_off(void)
{
    digitalWrite(Beep, HIGH);
}

void beepCtrl(int t)
{
    beep_on();
    delay(t);
    beep_off();
    delay(t);
}

float tempRead(void)
{
    float temp;
    int i, j;
    int fd;
    int ret;

    char buf[BUFSIZE];
    char tempBuf[5];

    fd = open("/sys/bus/w1/devices/28-031590bf4aff/w1_slave", O_RDONLY);

    if(-1 == fd){
        perror("open device file error");
        return 1;
    }

    while(1){
        ret = read(fd, buf, BUFSIZE);
        if(0 == ret){
            break;
        }
        if(-1 == ret){
            if(errno == EINTR){
                continue;
            }
            perror("read()");
            close(fd);
            return 1;
        }
    }
```

(continues on next page)

(continued from previous page)

```
    for(i=0;i<sizeof(buf);i++){
        if(buf[i] == 't'){
            for(j=0;j<sizeof(tempBuf);j++){
                tempBuf[j] = buf[i+2+j];
            }
        }
    }

    temp = (float)atoi(tempBuf) / 1000;

    close(fd);

    return temp;
}

void ledInit(void)
{
    pinMode(LedRed,   OUTPUT);
    pinMode(LedGreen, OUTPUT);
    pinMode(LedBlue,  OUTPUT);
}

/* */
void ledCtrl(int n, int state)
{
    digitalWrite(n, state);
}

void joystickquit(void)
{
    sys_state = 0;
    printf("interrupt occur !\n");
}

uchar get_joyStick_state(void)
{
    uchar tmp = 0;
    uchar xVal = 0, yVal = 0, zVal = 0;

    xVal = analogRead(AIN1);
    if(xVal >= 250){
        tmp = 1;
    }
    if(xVal <= 5){
        tmp = 2;
    }

    yVal = analogRead(AIN0);
    if(yVal >= 250){
        tmp = 4;
    }
    if(yVal <= 5){
        tmp = 3;
    }

    zVal = analogRead(AIN2);
```

(continues on next page)

(continued from previous page)

```
    if(zVal <= 5){
        tmp = 5;
    }

    if(xVal-127<30 && xVal-127>-30 && yVal-127<30 && yVal-127>-30 && zVal>127){
        tmp = 0;
    }
// Uncomment this line to see the value of joystick.
//  printf("x = %d, y = %d, z = %d",xVal,yVal,zVal);
    return tmp;
}

int main(void)
{
    int i;
    uchar joyStick = 0;
    float temp;
    uchar low = 26, high = 30;

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }

    pcf8591Setup(PCF, 0x48);
    ledInit();
    beepInit();

    printf("System is running...\n");

    while(1){
        flag:
        joyStick = get_joyStick_state();

        switch(joyStick){
            case 1 : --low;  break;
            case 2 : ++low;  break;
            case 3 : ++high; break;
            case 4 : --high; break;
            case 5 : joystickquit(); break;
            default: break;
        }

        if(low >= high){
            printf("Error, lower limit should be less than upper limit\n");
            goto flag;
        }

        printf("The lower limit of temperature : %d\n", low);
        printf("The upper limit of temperature : %d\n", high);

        temp = tempRead();

        printf("Current temperature : %0.3f\n", temp);

        if(temp < low){
            ledCtrl(LedBlue,  LOW);
```

(continues on next page)

(continued from previous page)

```
            ledCtrl(LedRed,   HIGH);
            ledCtrl(LedGreen, LOW);
            for(i = 0;i < 3; i++){
                beepCtrl(400);
            }
        }
        if(temp >= low && temp < high){
            ledCtrl(LedBlue,  HIGH);
            ledCtrl(LedRed,   HIGH);
            ledCtrl(LedGreen, LOW);
        }
        if(temp >= high){
            ledCtrl(LedBlue,  HIGH);
            ledCtrl(LedRed,   LOW);
            ledCtrl(LedGreen, HIGH);
            for(i = 0;i < 3; i++){
                beepCtrl(80);
            }
        }

        if(sys_state == 0){
            ledCtrl(LedRed, LOW);
            ledCtrl(LedGreen, LOW);
            ledCtrl(LedBlue, LOW);
            beep_off();
            printf("SyStem will be off...\n");
            break;
        }
    }
    return 0;
}
```

For Python Users:

Step 2: Change directory.

```
cd /home/pi/SunFounder_SensorKit_for_RPi2/Python/
```

Step 4: Run.

```
sudo python3 35_temp_monitor.py
```

Code

```
#!/usr/bin/env python3
import RPi.GPIO as  GPIO
import importlib
import time
import sys

# BOARD pin numbering
LedR        =       11
LedG        =       12
LedB        =       13
Buzz        =       15

#ds18b20 = '28-031590bf4aff'
```

(continues on next page)

(continued from previous page)

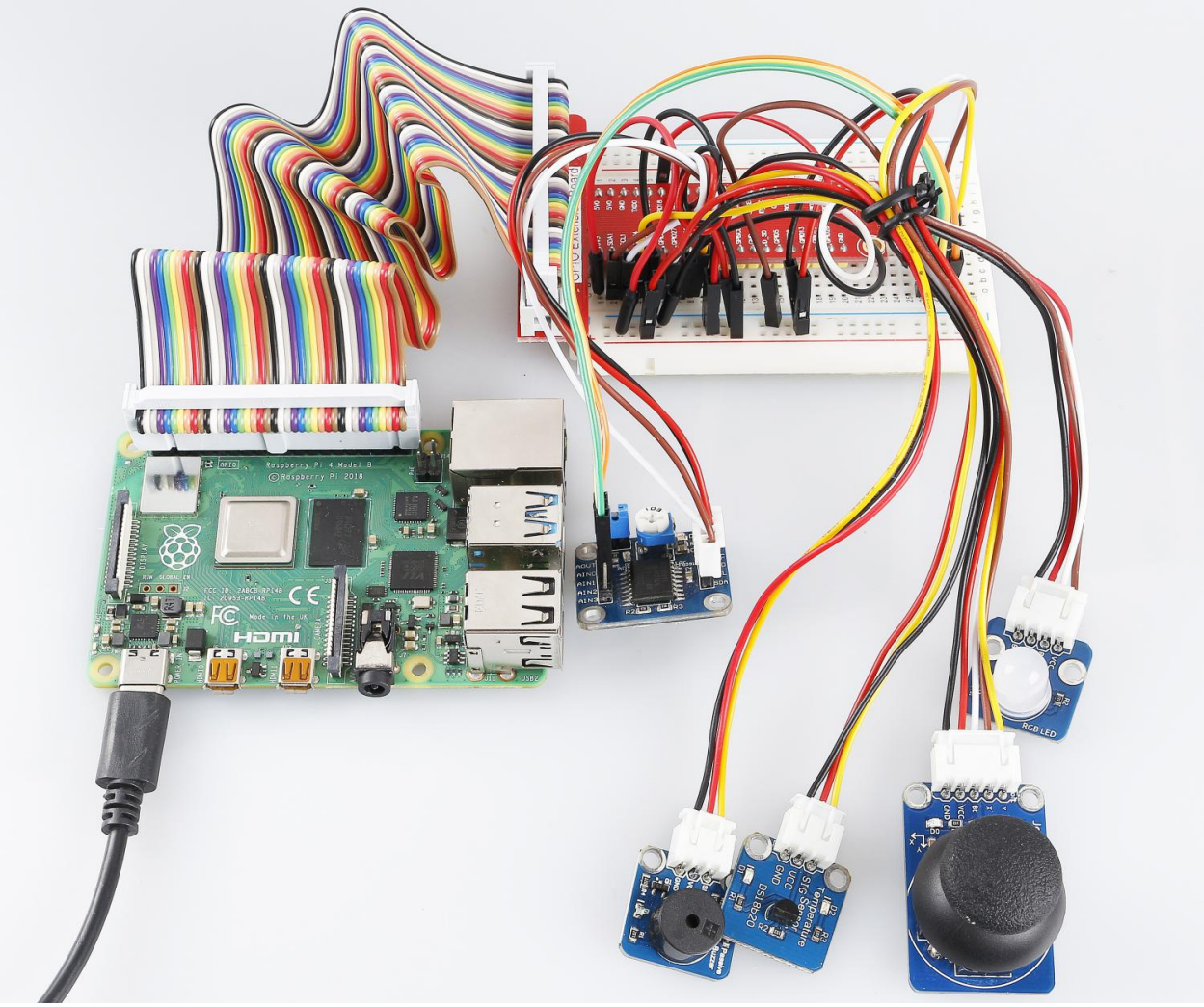
```
#location = '/sys/bus/w1/devices/' + ds18b20 + '/w1_slave'

joystick    =       importlib.import_module('15_joystick_PS2')
ds18b20             =       importlib.import_module('26_ds18b20')
beep                =       importlib.import_module('10_active_buzzer')
rgb                 =       importlib.import_module('02_rgb_led')

joystick.setup()
ds18b20.setup()
beep.setup(Buzz)
rgb.setup(LedR, LedG, LedB)

color = {'Red':0xFF0000, 'Green':0x00FF00, 'Blue':0x0000FF}

def setup():
    global lowl, highl
    lowl = 29
    highl = 31

def edge():
    global lowl, highl
    temp = joystick.direction()
    if temp == 'Pressed':
        destroy()
        quit()
    if temp == 'up' and lowl < highl-1:
        highl += 1
    if temp == 'down' and lowl >= -5:
        highl -= 1
    if temp == 'right' and highl <= 125:
        lowl += 1
    if temp == 'left' and lowl < highl-1:
        lowl -= 1

def loop():
    while True:
        edge()
        temp = ds18b20.read()
        print ('The lower limit of temperature : ', lowl)
        print ('The upper limit of temperature : ', highl)
        print ('Current temperature : ', temp)
        if float(temp) < float(lowl):
            rgb.setColor(color['Blue'])
            for i in range(0, 3):
                beep.beep(0.5)
        if temp >= float(lowl) and temp < float(highl):
            rgb.setColor(color['Green'])
        if temp >= float(highl):
            rgb.setColor(color['Red'])
            for i in range(0, 3):
                beep.beep(0.1)

def destroy():
    beep.destroy()
    joystick.destroy()
    ds18b20.destroy()
    rgb.destroy()
```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()
```

Now, you can pull the shaft of the joystick left and right to set the upper limit value, and up and down to set the lower limit value. Then, if the ambient temperature reaches the upper limit value or lower limit value, the buzzer will beep in a different frequency to warn.



CHAPTER

SEVEN

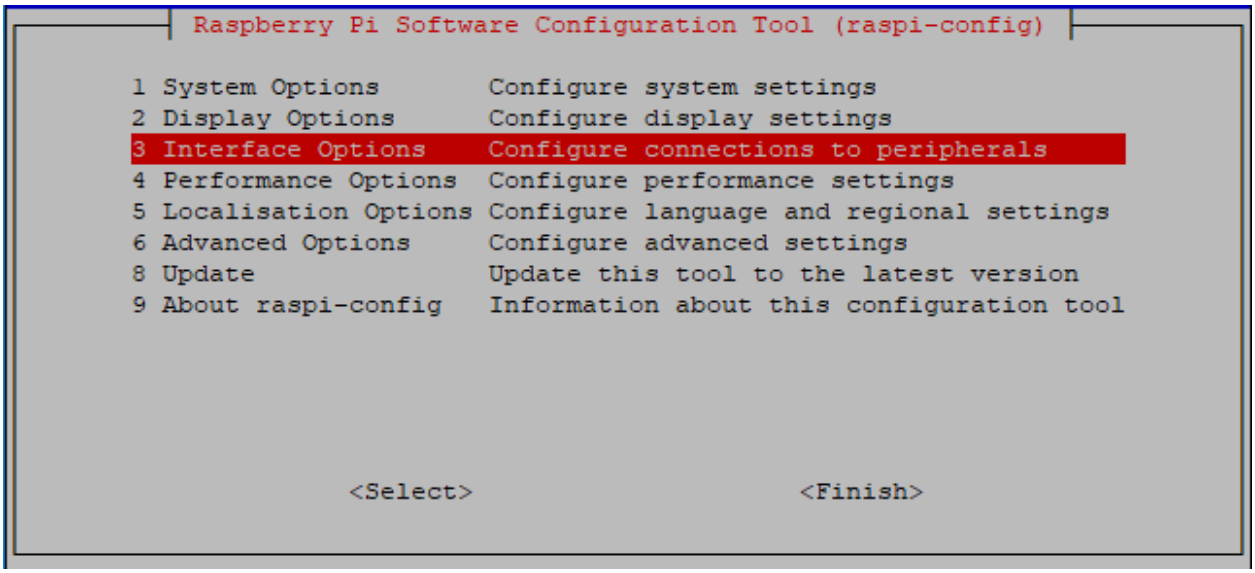
APPENDIX

7.1 I2C Configuration

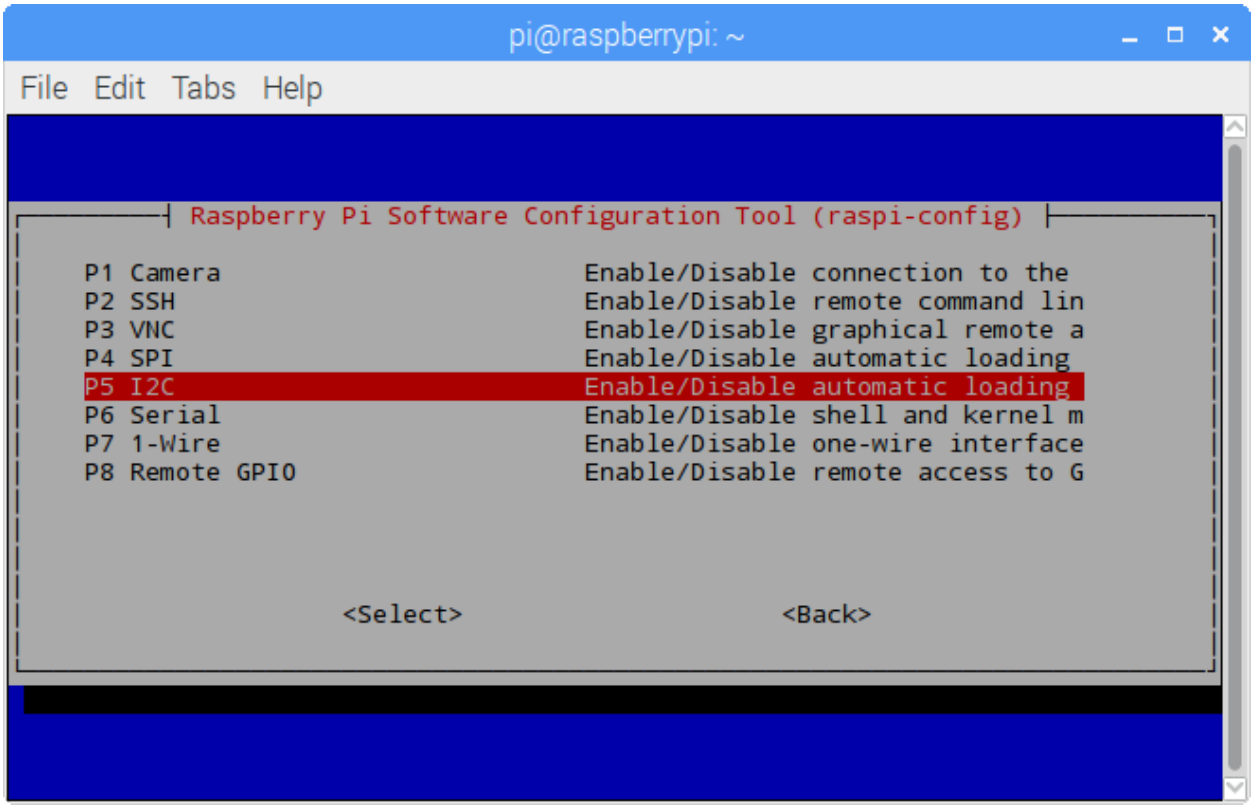
Step 1: Enable the I2C port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

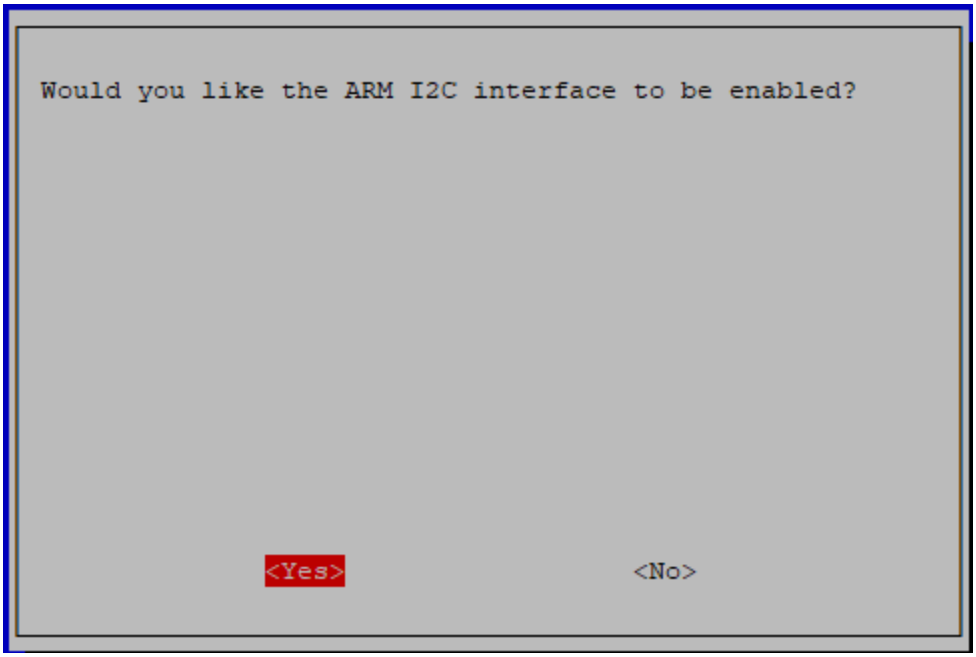
3 Interfacing options



P5 I2C



<Yes>, then <Ok> -> <Finish>



Step 2: Check whether the i2c modules are loaded and active.

```
lsmod | grep i2c
```

Then the following codes will appear (the number may be different).

```
i2c_dev                 6276    0
i2c_bcm2708             4121    0
```

Step 3: Install i2c-tools.

```
sudo apt-get install i2c-tools
```

Step 4: Check the address of the I2C device.

```
i2cdetect -y 1      # For Raspberry Pi 2 and higher version
```

```
i2cdetect -y 0      # For Raspberry Pi 1
```

```
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: -- -- -- -- -- -- -- --
```

If there is an I2C device connected, the address of the device will be displayed.

Step 5:

For C language users: Install libi2c-dev.

```
sudo apt-get install libi2c-dev
```

For Python users: Install smbus for I2C.

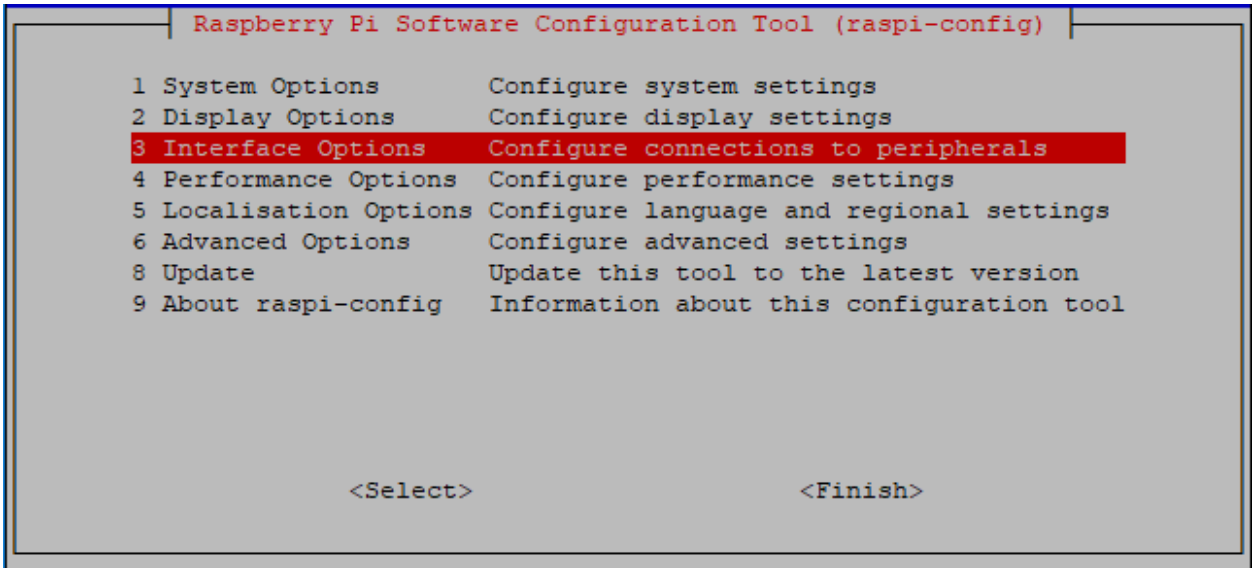
```
sudo pip3 install smbus2
```

7.2 SPI Configuration

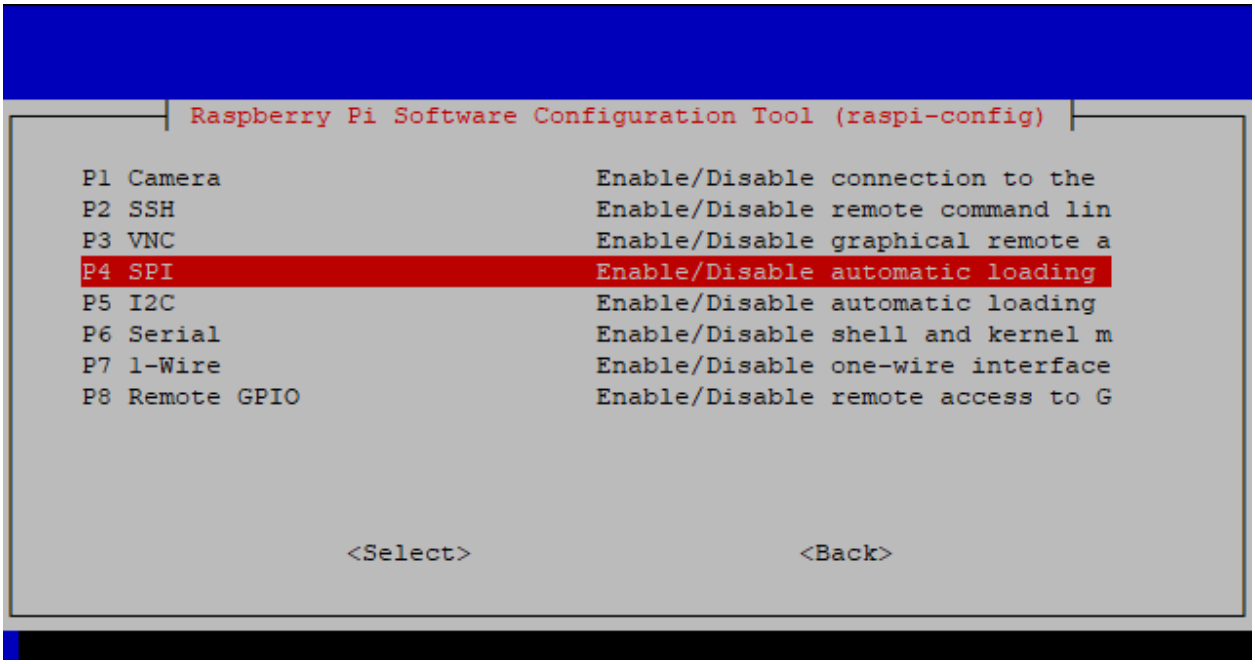
Step 1: Enable the SPI port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

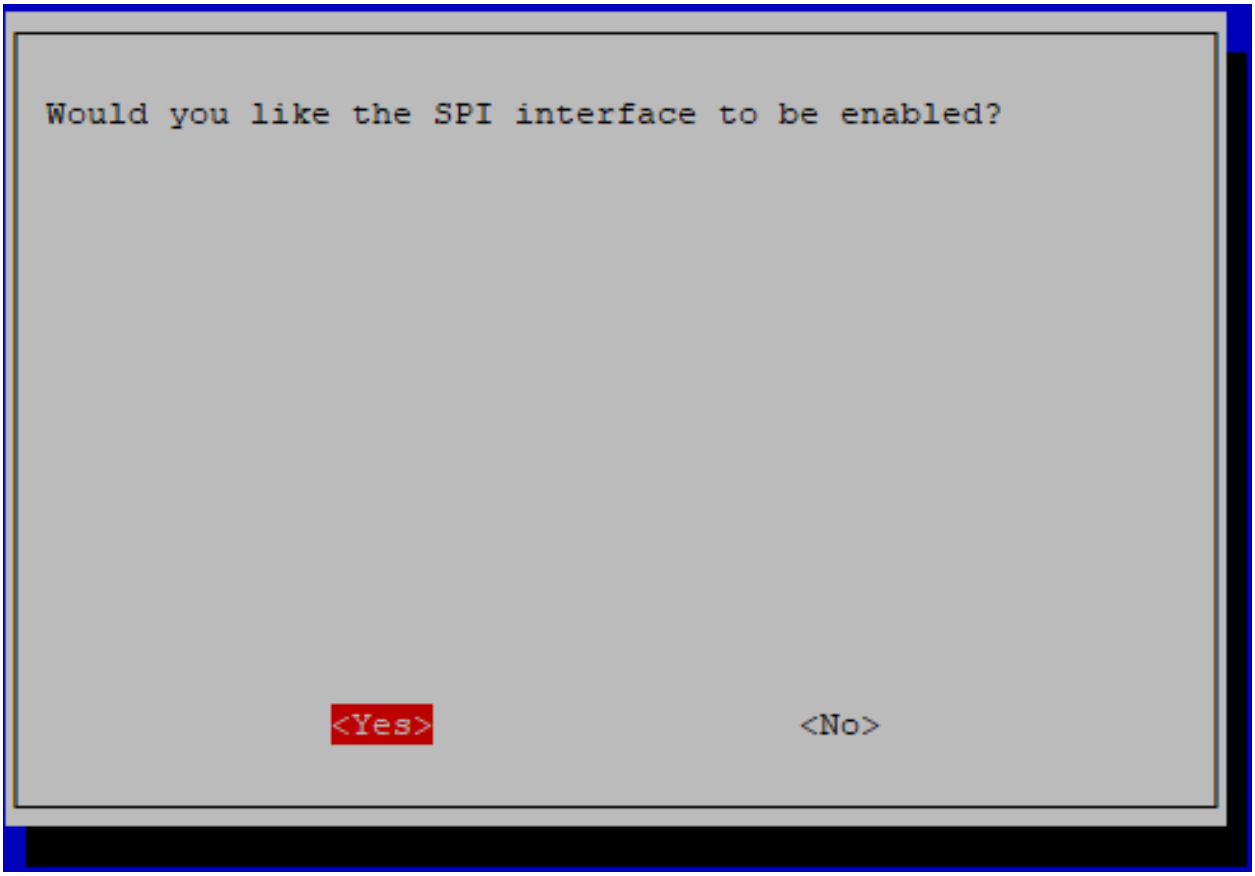
3 Interfacing options



P4 SPI



<YES>, then click **<OK>** and **<Finish>**. Now you can use the `sudo reboot` command to reboot the Raspberry Pi.



Step 2: Check that the spi modules are loaded and active.

```
ls /dev/sp*
```

Then the following codes will appear (the number may be different).

```
/dev/spidev0.0  /dev/spidev0.1
```

Step 3: Install Python module SPI-Py.

```
git clone https://github.com/lthiery/SPI-Py.git
cd SPI-Py
sudo python3 setup.py install
```

Note: This step is for python users, if you use C language, please skip.

7.3 Remote Desktop

There are two ways to control the desktop of the Raspberry Pi remotely:

VNC and **XRDP**, you can use any of them.

7.3.1 VNC

You can use the function of remote desktop through VNC.

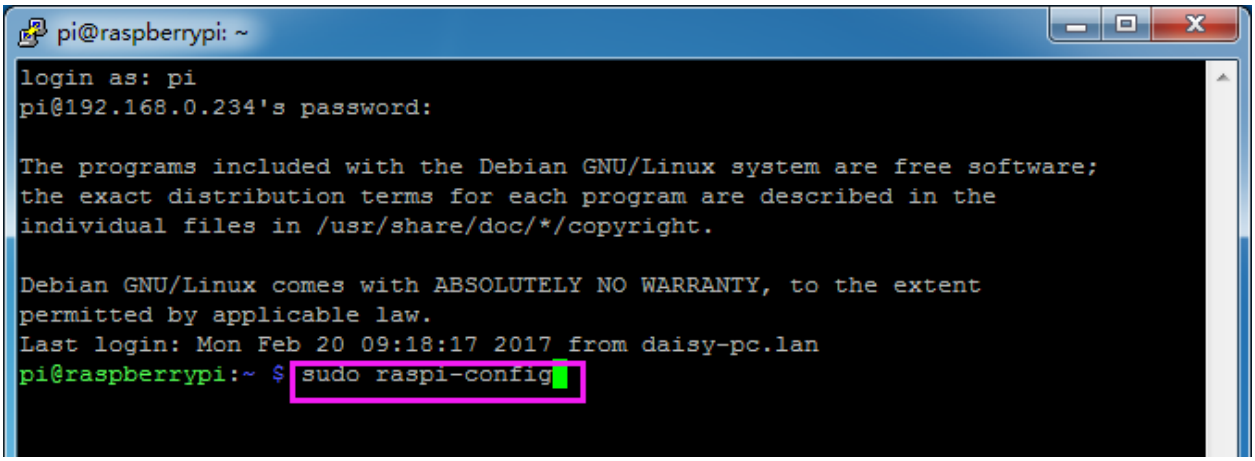
Enable VNC service

The VNC service has been installed in the system. By default, VNC is disabled. You need to enable it in config.

Step 1

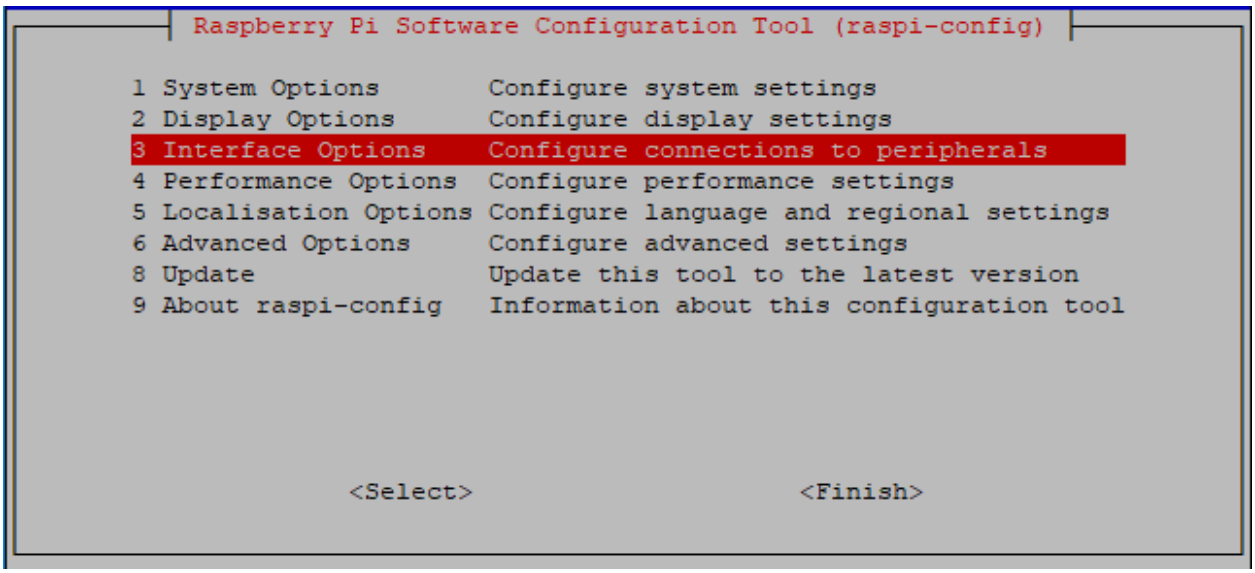
Input the following command:

```
sudo raspi-config
```



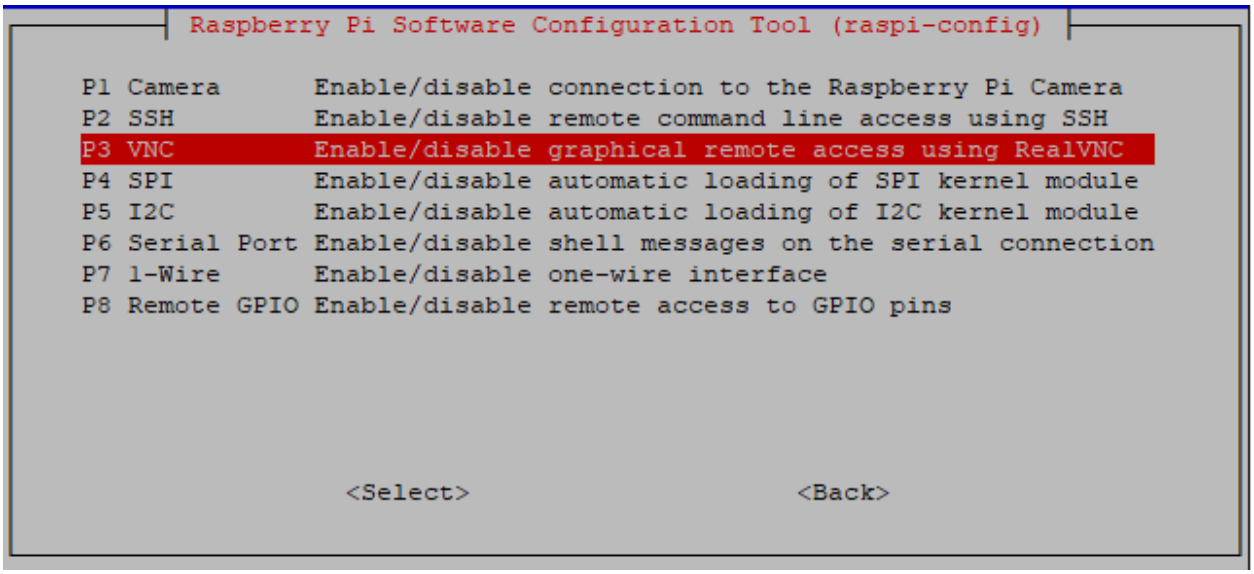
Step 2

Choose **3 Interfacing Options** by press the down arrow key on your keyboard, then press the **Enter** key.



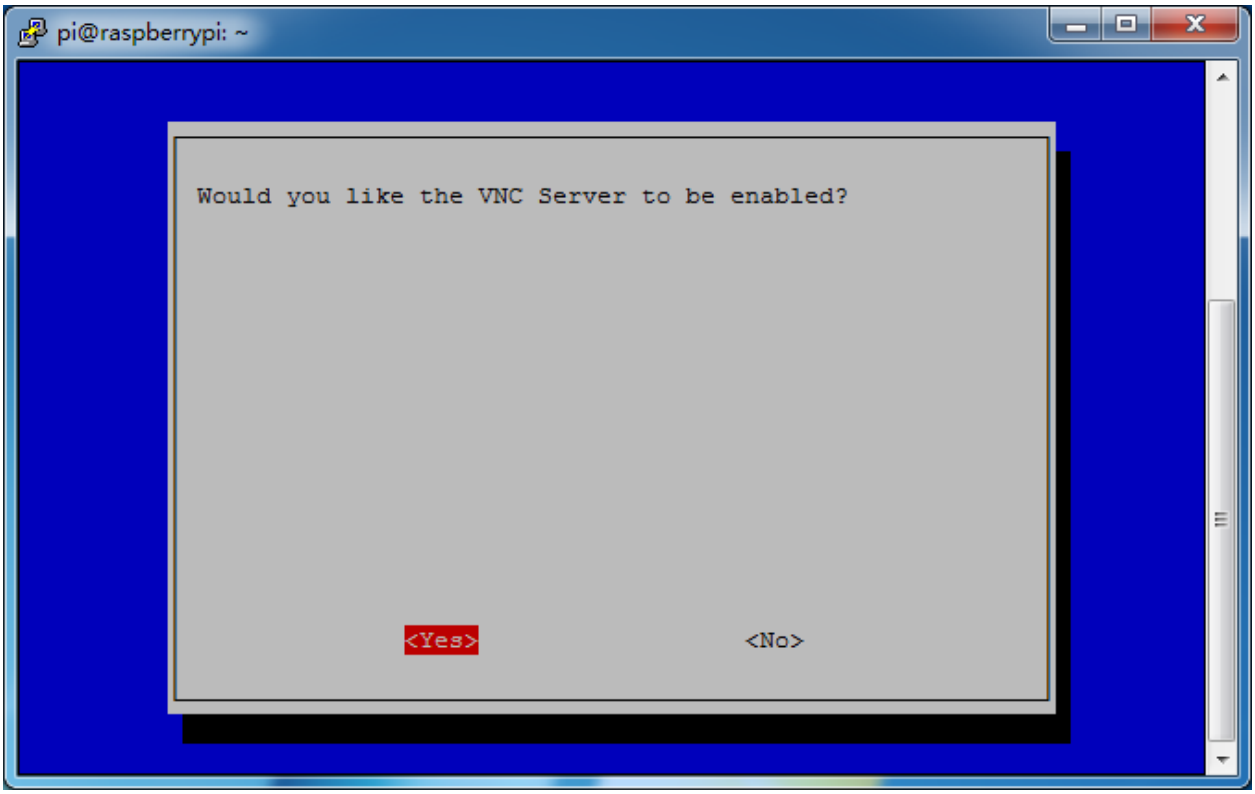
Step 3

P3 VNC



Step 4

Select **Yes -> OK -> Finish** to exit the configuration.



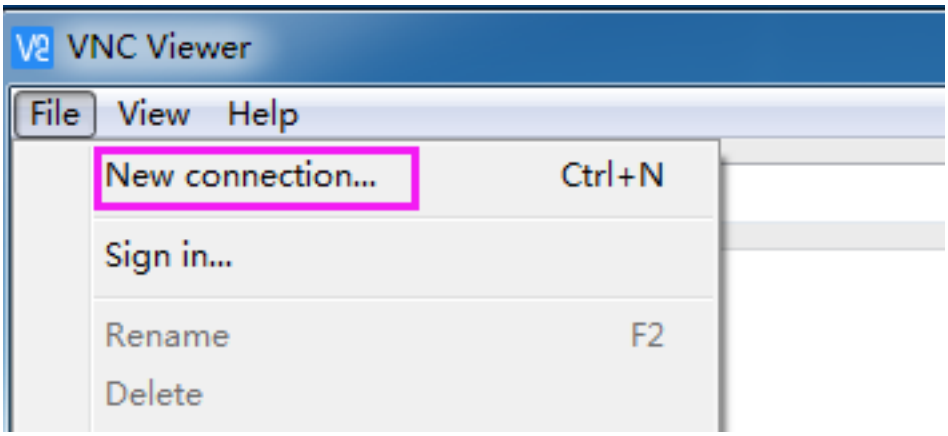
Login to VNC

Step 1

You need to download and install the VNC Viewer on personal computer. After the installation is done, open it.

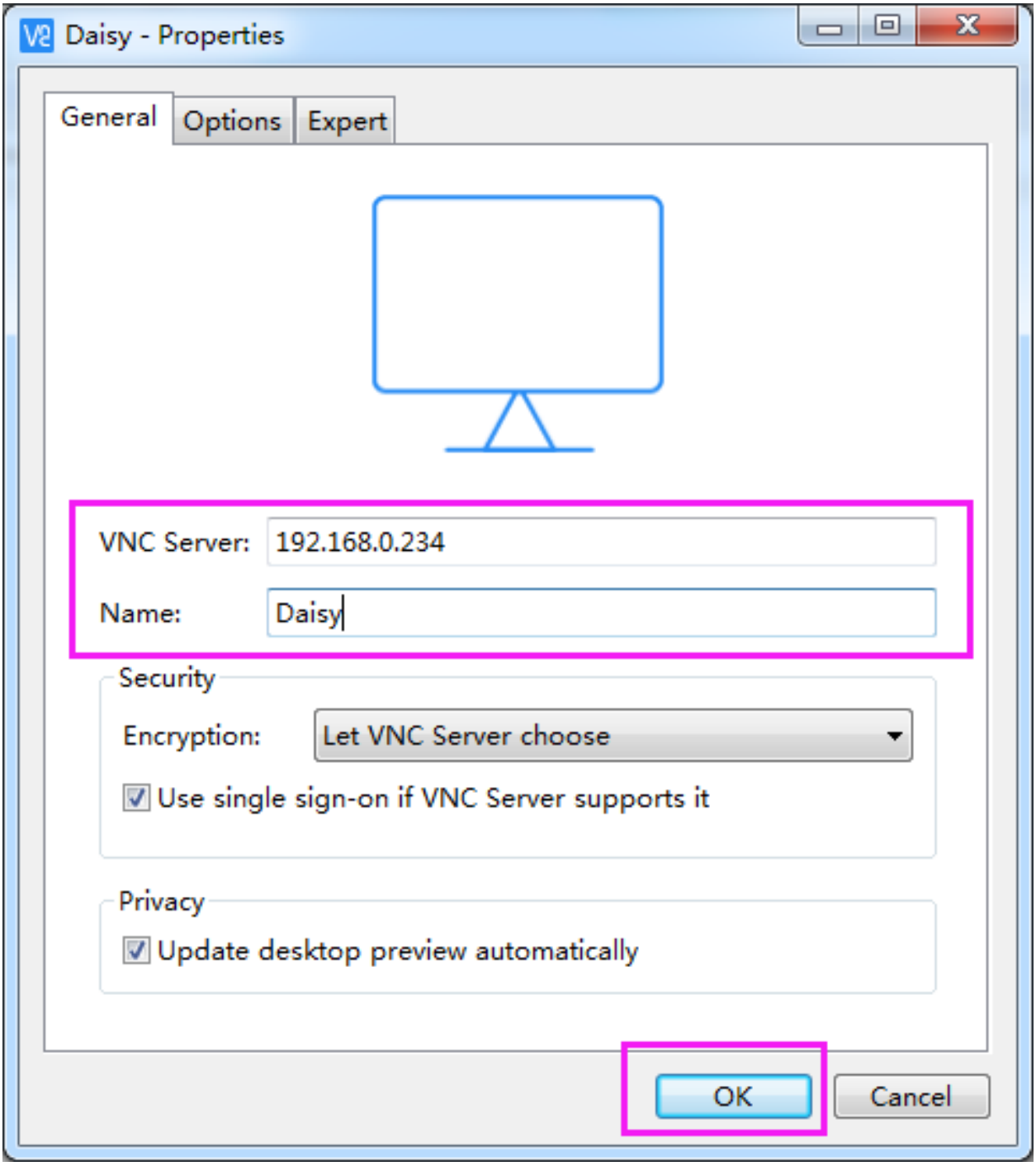
Step 2

Then select "**New connection**".



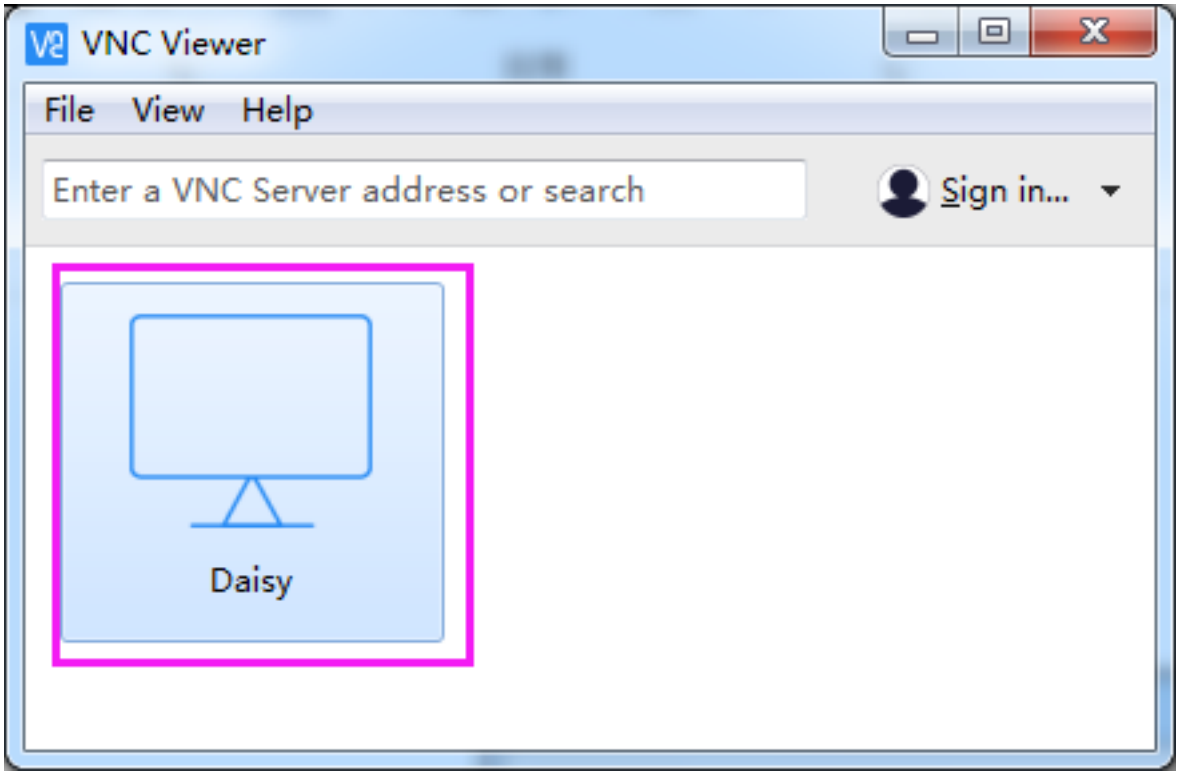
Step 3

Input IP address of Raspberry Pi and any **Name**.



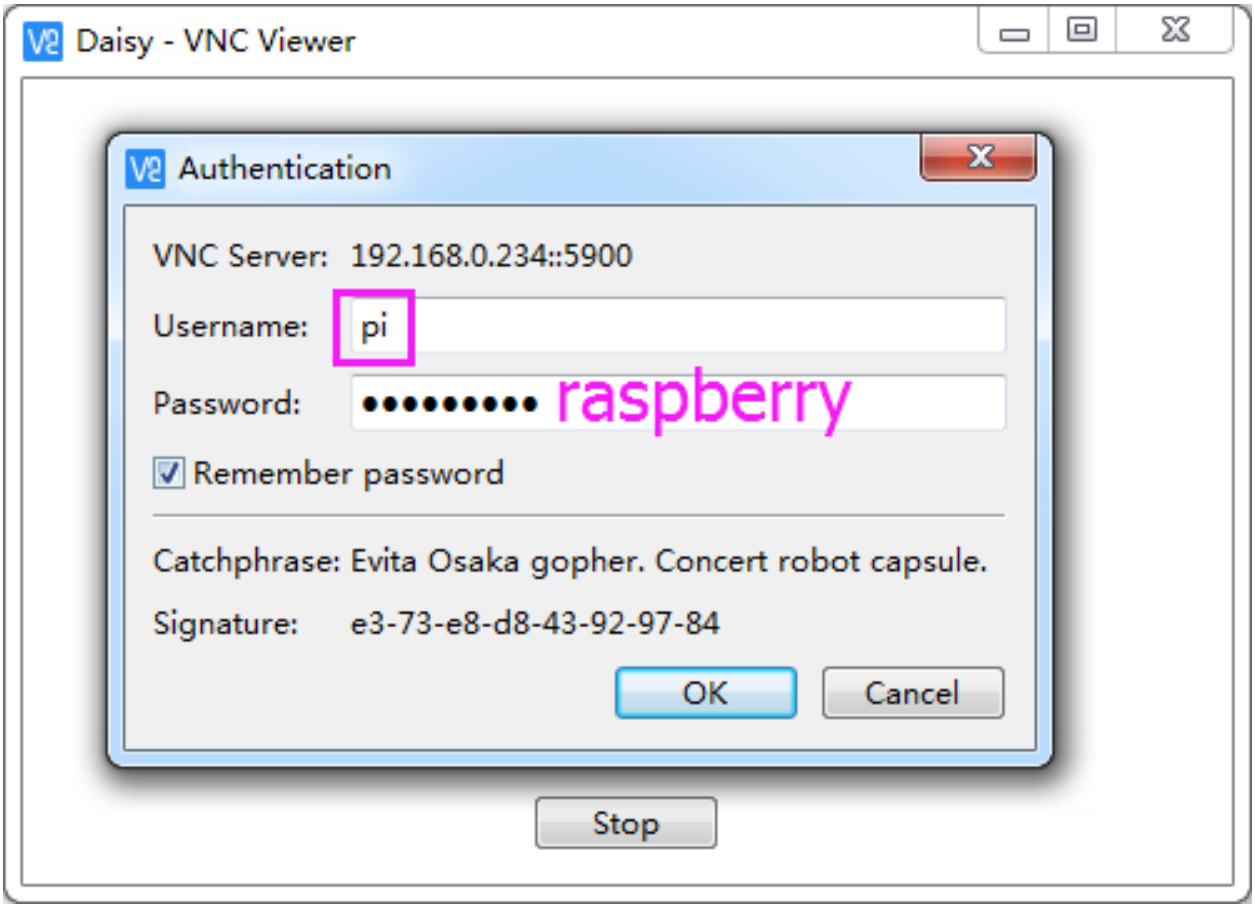
Step 4

Double click the **connection** just created:



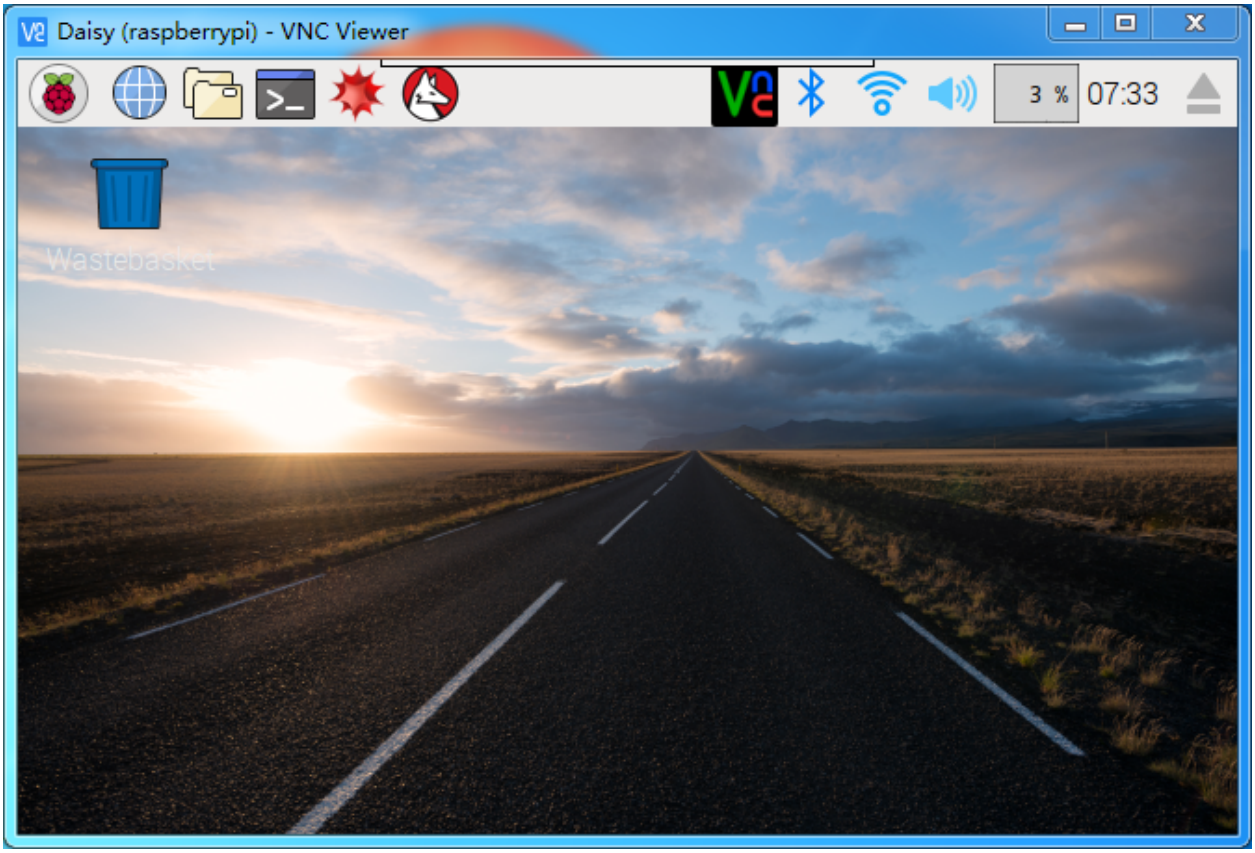
Step 5

Enter Username (**pi**) and Password (**raspberry** by default).



Step 6

Now you can see the desktop of the Raspberry Pi:



That's the end of the VNC part.

7.3.2 XRDP

Another method of remote desktop is XRDP, it provides a graphical login to remote machines using RDP (Microsoft Remote Desktop Protocol).

Install XRDP

Step 1

Login to Raspberry Pi by using SSH.

Step 2

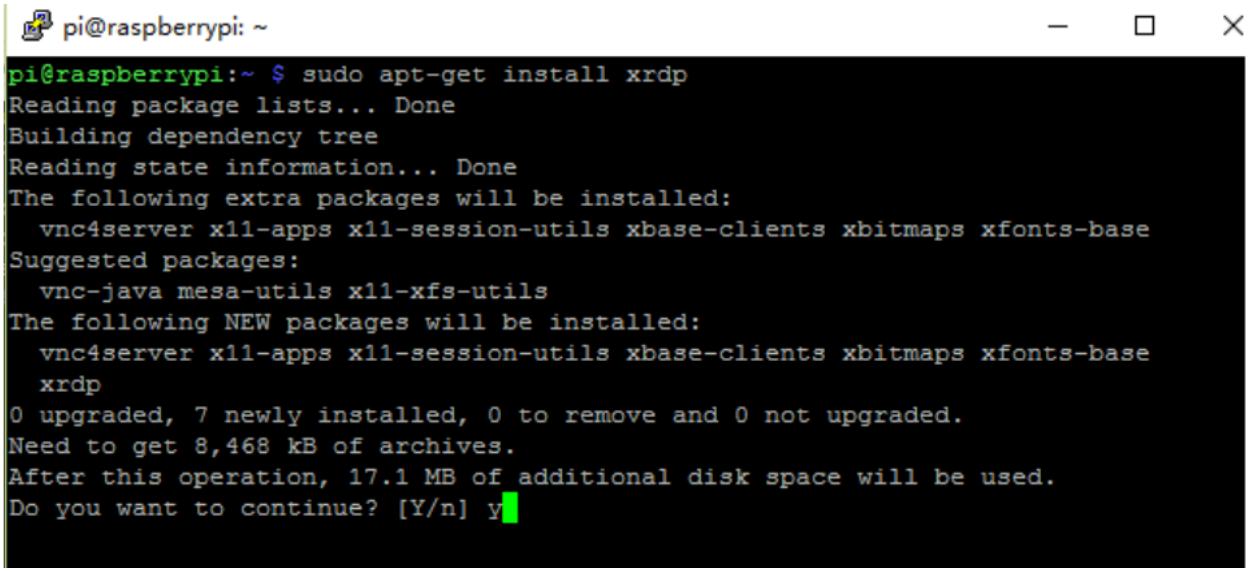
Input the following instructions to install XRDP.

```
sudo apt-get update
sudo apt-get install xrdp
```

Step 3

Later, the installation starts.

Enter "Y", press key "Enter" to confirm.



Step 4

Finished the installation, you should login to your Raspberry Pi by using Windows remote desktop applications.

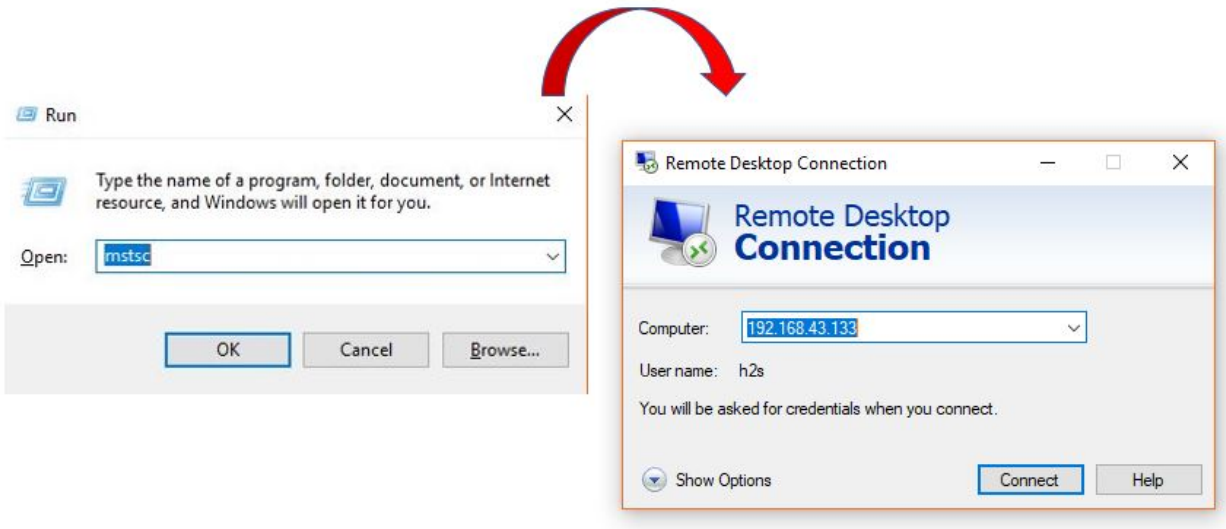
Login to XRDP

Step 1

If you are a Windows user, you can use the Remote Desktop feature that comes with Windows. If you are a Mac user, you can download and use Microsoft Remote Desktop from the APP Store, and there is not much difference between the two. The next example is Windows remote desktop.

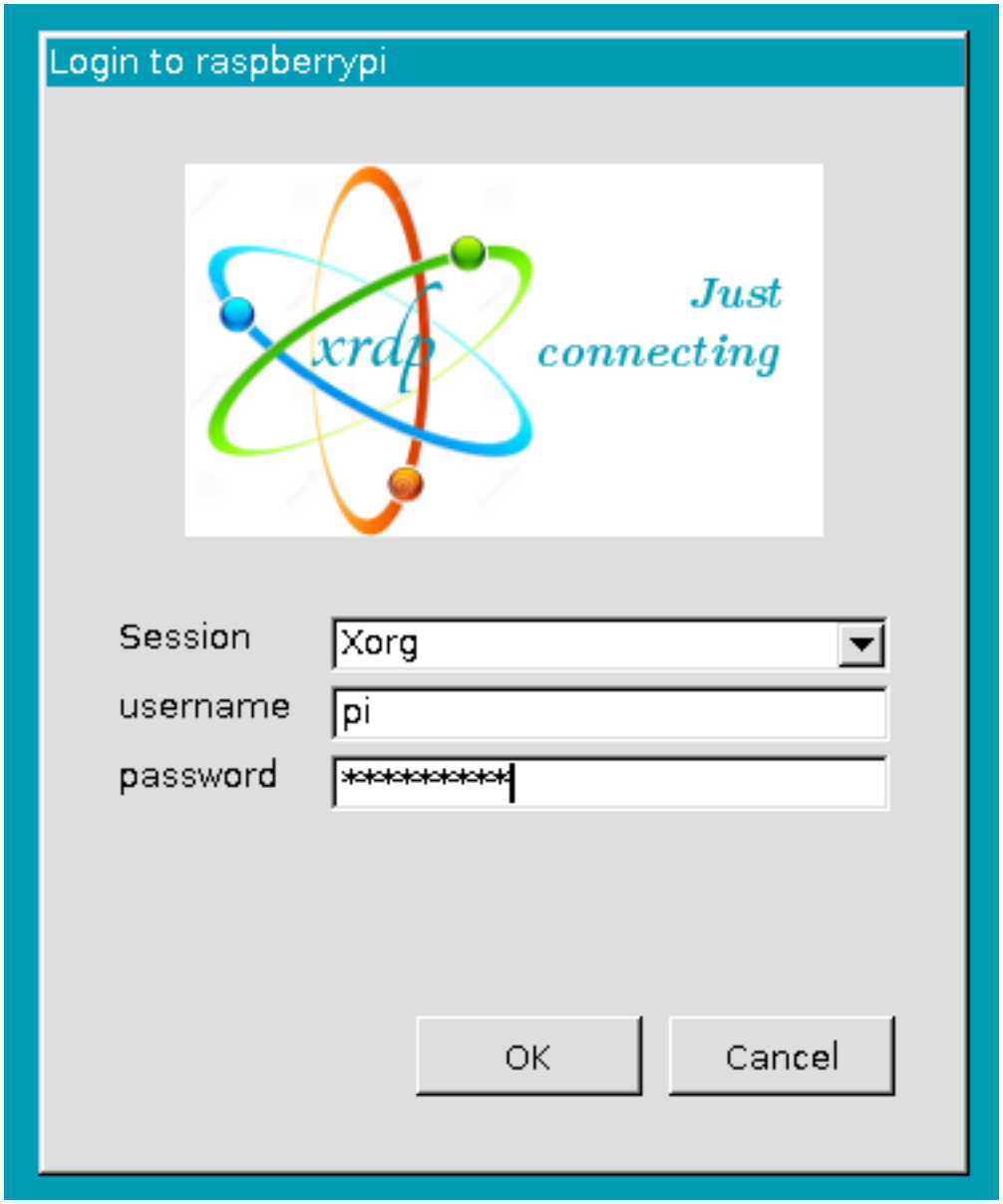
Step 2

Type in "**mstsc**" in Run (WIN+R) to open the Remote Desktop Connection, and input the IP address of Raspberry Pi, then click on "Connect".



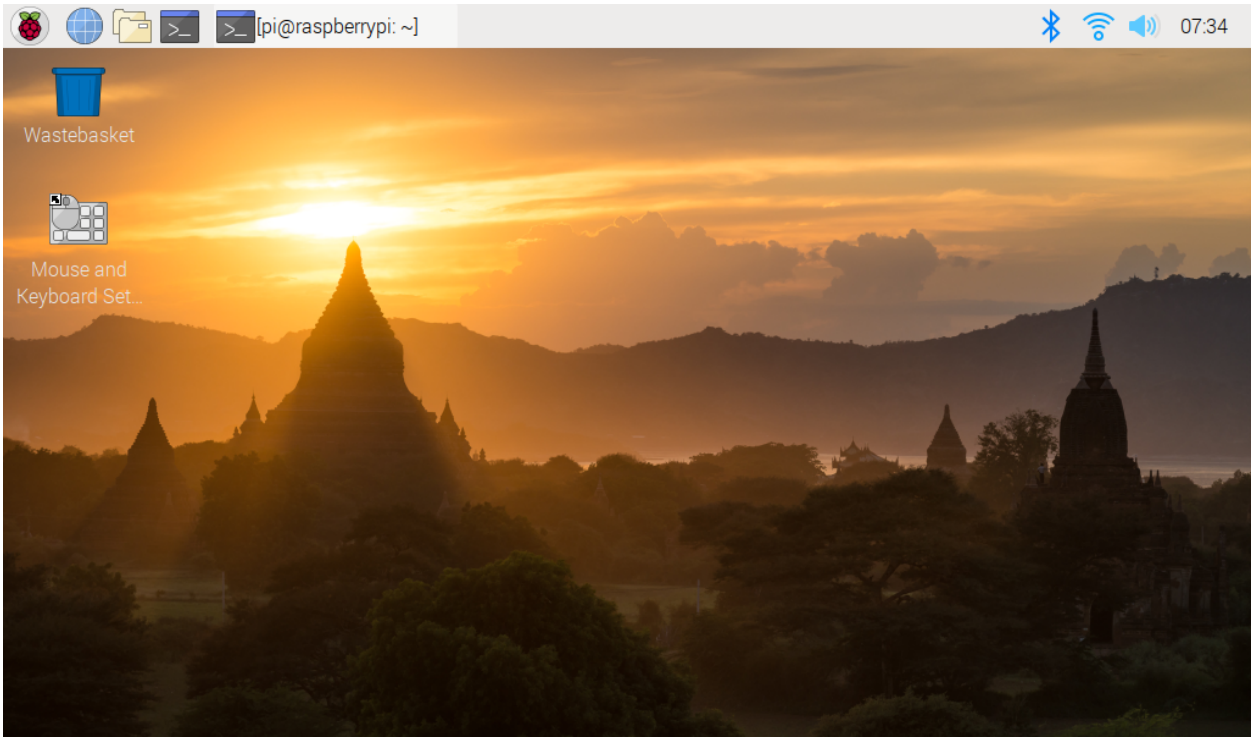
Step 3

Then the xrdp login page pops out. Please type in your username and password. After that, please click “OK”. At the first time you log in, your username is “pi” and the password is “raspberry”.



Step 4

Here, you successfully login to RPi by using the remote desktop.



Copyright Notice

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.

CHAPTER

EIGHT

FAQ

8.1 C code is not working?

- Check your wiring for problems.
- Check if the code is reporting errors, if so, refer to: *WiringPi*.
- Has the code been compiled before running.
- If all the above 3 conditions are OK, it may be that your wiringPi version (2.50) is not compatible with your Raspberry Pi 4B and above, refer to *WiringPi* to manually upgrade it to version 2.52.

CHAPTER

NINE

THANK YOU

Thanks to the evaluators who evaluated our products, the veterans who provided suggestions for the tutorial, and the users who have been following and supporting us. Your valuable suggestions to us are our motivation to provide better products!

Particular Thanks

- Len Davisson
- Kalen Daniel
- Juan Delacosta

Now, could you spare a little time to fill out this questionnaire?

Note: After submitting the questionnaire, please go back to the top to view the results.

CHAPTER

TEN

COPYRIGHT NOTICE

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study,investigation, enjoyment, or other non-commercial or nonprofit purposes, under therelated regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.