Preface

About SunFounder

SunFounder is a technology company focused on Raspberry Pi and Arduino open source community development. Committed to the promotion of open source culture, we strive to bring the fun of electronics making to people all around the world and enable everyone to be a maker. Our products include learning kits, development boards, robots, sensor modules and development tools. In addition to high quality products, SunFounder also offers video tutorials to help your own project. If you have interest in open source or making something cool, welcome to join us! Visit www.sunfounder.com for more!

About Super Kit

This super kit is suitable for the Raspberry Pi B, model B+, Pi 2 model B, Pi 3 model B , Pi 3 model B+ and 4 Model B. It includes various components and chips that can show different interesting phenomena. You can make it happen by following the experiment instructions, and learn basic knowledge about them. Also you can explore more application after mastering the principle and code. Now get on the road!

In this book, we will show you circuits with both realistic illustrations and schematic diagrams. You can go to our official website www.sunfounder.com to download related code by clicking **LEARN** -> **Get Tutorials** and watch related videos by **VIDEO**. Our example code and tutorials are only applied to the Raspbian system.

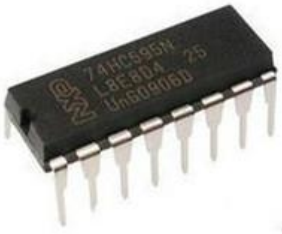
Free Support

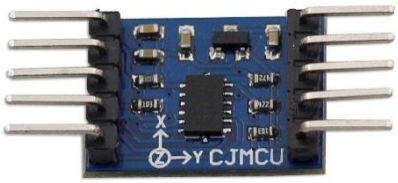
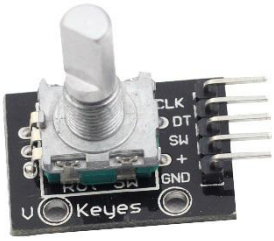
If you have any **TECHNICAL questions**, add a topic under **FORUM** section on our website and we'll reply as soon as possible.

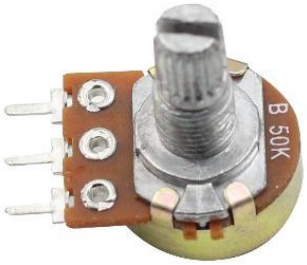
For **NON-TECH questions** like order and shipment issues, please **send an email to service@sunfounder.com**. You're also welcomed to share your projects on FORUM.

Contents

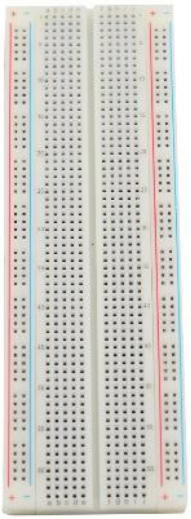
Components List

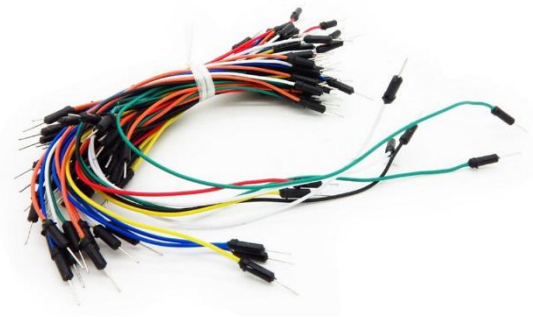
No	Name	Quantity	Component
1	RGB LED	1	
2	Pin Header	40	
3	555 Timer IC	1	
4	Optocoupler (4N35)	2	
5	Shift Register (74HC595)	2	
6	L293D	1	

7	Accelerometer ADXL345	1	
8	Rotary Encoder	1	
9	Button	5	
10	Resistor (220Ω)	8	 (red, red, black, black, brown)
11	Resistor (1kΩ)	8	 (brown, black, black, brown, brown)
12	Resistor (10kΩ)	4	 (brown, black, black, red, brown)
13	Resistor (100kΩ)	4	 (brown, black, black, orange, brown)
14	Resistor (1MΩ)	1	 (brown, black, green, gold)
15	Resistor (5.1MΩ)	1	 (green, brown, green, gold)

16	Switch	1	
17	Potentiometer (50k)	1	
18	Power Supply Module	1	
19	LCD1602	1	
20	Dot Matrix Display (8*8)	1	
21	7-Segment Display	2	

22	DC Motor	1	
23	LED (red)	16	
24	LED (white)	2	
25	LED (green)	2	
26	LED (yellow)	2	
27	NPN Transistor (S8050)	2	

28	PNP Transistor (S8550)	2	
29	Capacitor Ceramic 100nF	4	
30	Capacitor Ceramic 10nF	4	
31	Diode Rectifier	4	
32	Breadboard	1	

33	Male-to-Male Jumper Wire	65	
34	Female-to-Male Dupont Wire	20	
35	Active Buzzer	1	
36	Fan	1	

37	GPIO Extension Board	1	
38	40-pin Ribbon Cable	1	

Notes:

After unpacking, please check that the number of components is correct and that all components are in good condition.

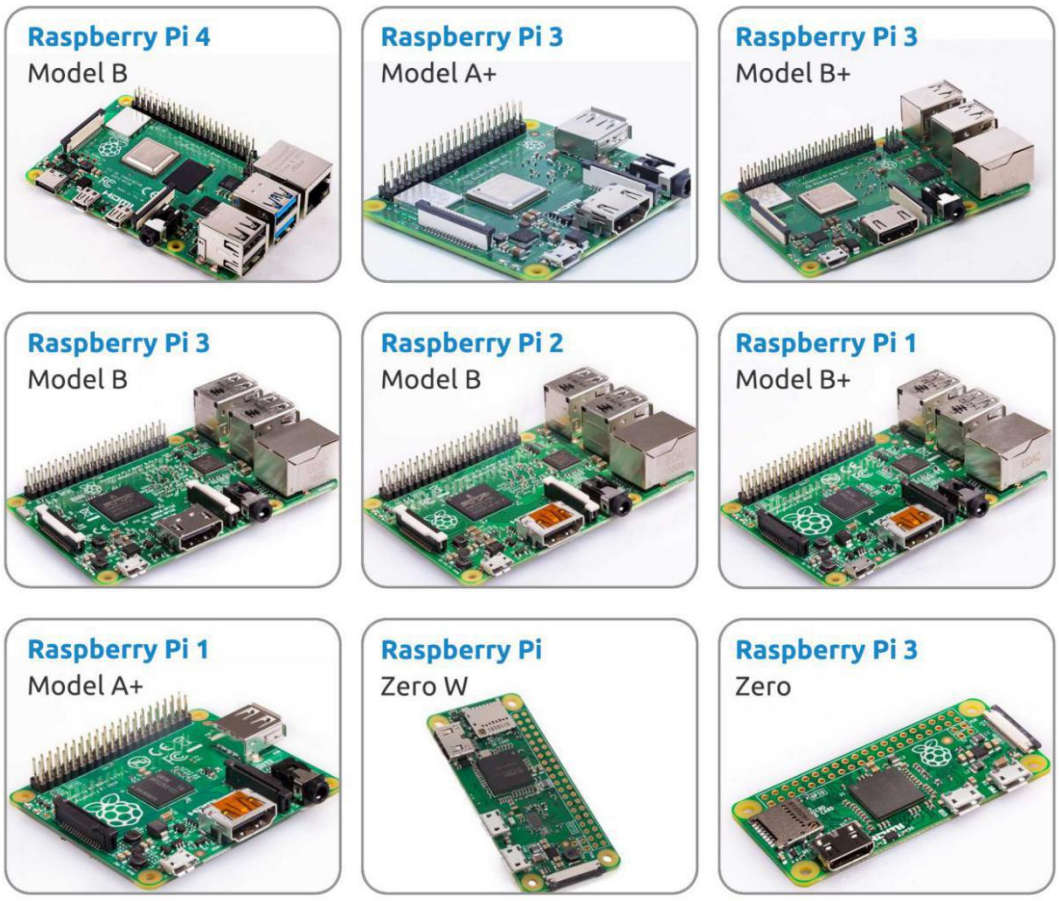
What Do We Need?

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 Raspbian operating system. You will need a micro SD card with a capacity of at least 8 GB.

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.

Preparation

Depending on the different devices you use, you can start up the Raspberry Pi in different methods. If you have a separate screen for Raspberry Pi, follow the instructions in this chapter. Otherwise, please find the corresponding steps in the following chapters.

If You Have A Screen

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

Required Components

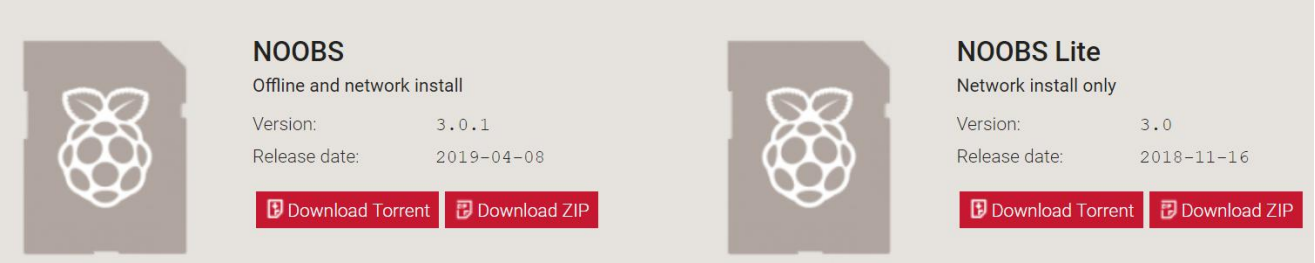
Any Raspberry Pi	1 * 2.5A Power Adapter
1 * Monitor	1 * Monitor Power Adapter
1 * HDMI cable	1 * Micro SD card
1 * Mouse	1 * Keyboard
1 * Personal Computer	

Procedures

Step 1

To download NOOBS from your PC, you can choose **NOOBS** or **NOOBS LITE -** the only difference is that there is a built-in offline Raspbian installer in **NOOBS**, while the **NOOBS LITE** can only be operated online. Here, you are suggested to use the former. Here is the download address of Noobs:

https://www.raspberrypi.org/downloads/noobs/



Step 2

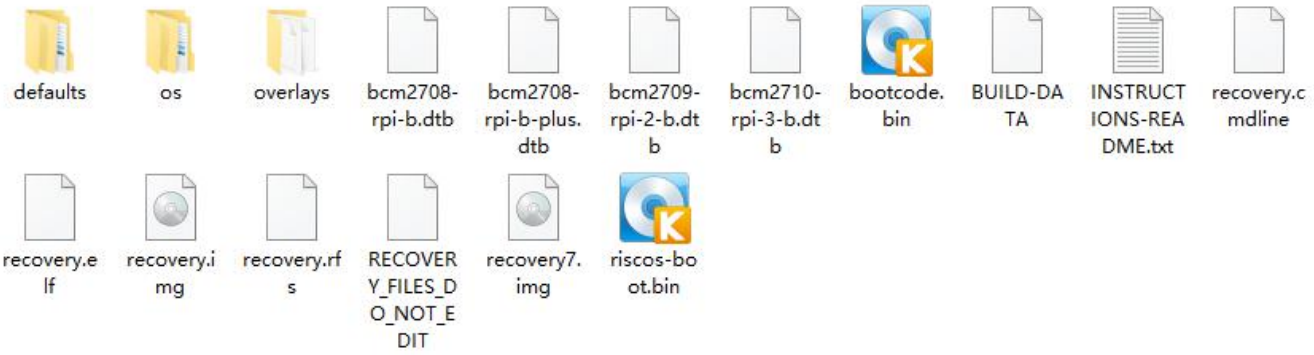
Format the SD card. If there are some important files in the SD card of Raspbian, please backup them first.

Step 3

Next, you will need to extract the files from the NOOBS zip archive you downloaded from the Raspberry Pi website.

- Find the downloaded archive — by default, it should be in your Downloads folder.
- Double-click on it to extract the files, and keep the resulting Explorer/Finder window open.

Finally Select all the files in the NOOBS folder and copy them to the SD card.

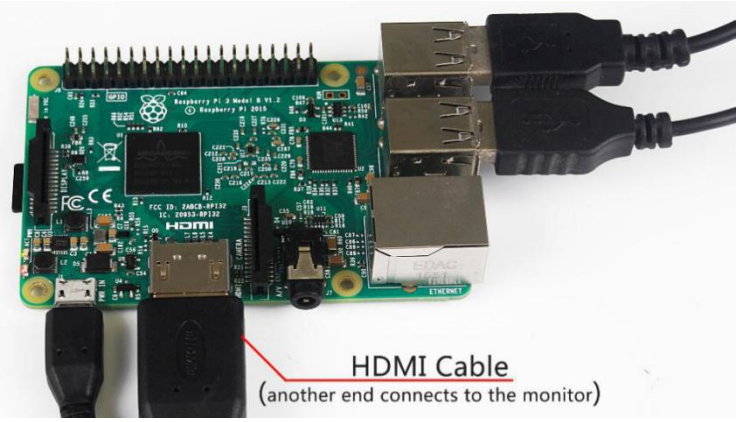


Step 4

All the files transferred, the SD card pops up.

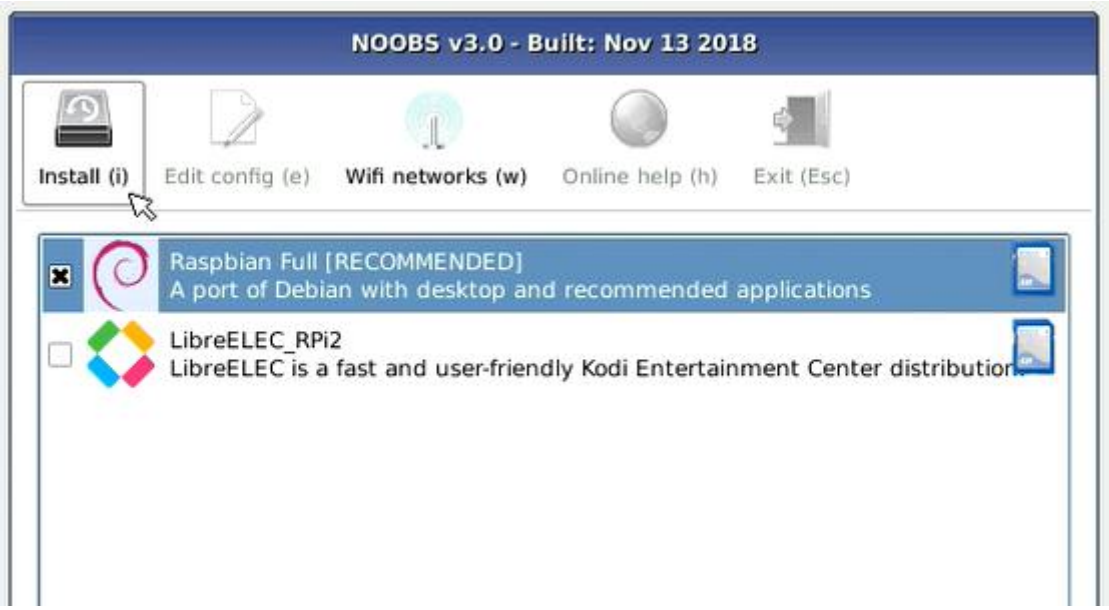
Step 5

Insert the SD card into the Raspberry Pi. In addition, connect the screen, keyboard and mouse to it. Finally power up the Raspberry Pi with a 2.5A power adapter.



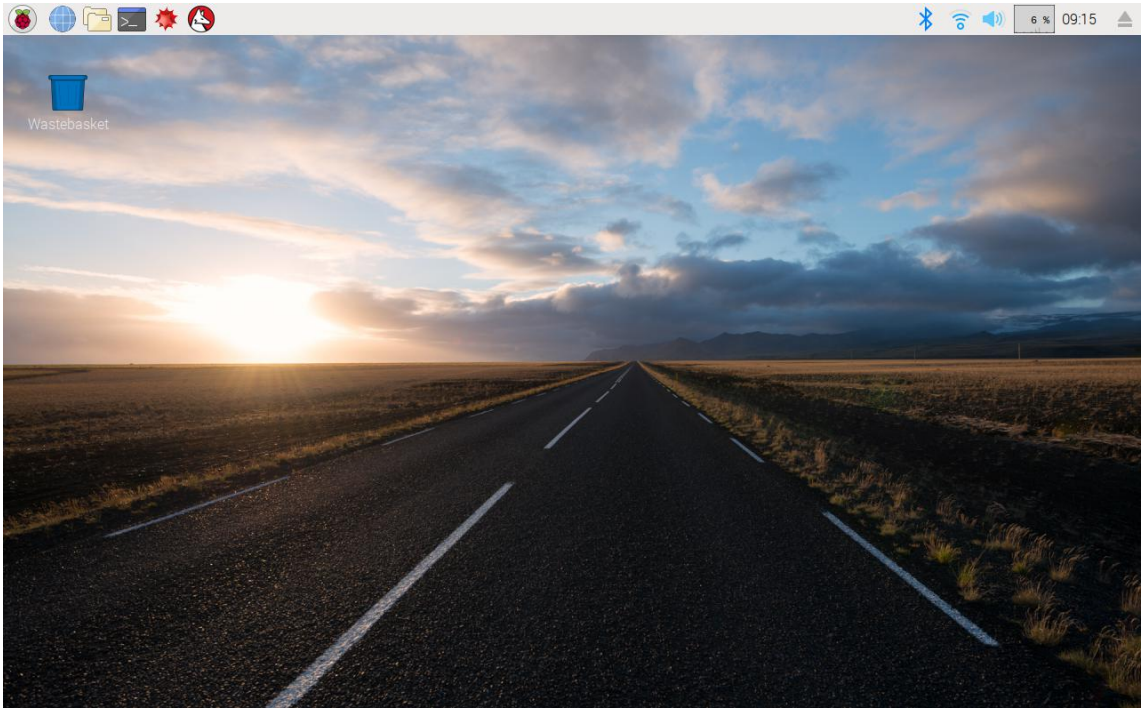
Step 6

It will go to the NOOBS interface after starting up. If you use NOOBS LITE, you need to select Wi-Fi networks (w) first. Tick the checkbox of the Raspbian and click Install in the top left corner. The NOOBS will help to conduct the installation automatically. This process will take a few minutes.



Step 7

When the installation is done, the system will restart automatically and the desktop of the system will appear.



Step 8

If you run Raspberry Pi for the first time, the application of "Welcome to Raspberry Pi" pops up and guides you to perform the initial setup.

Step 9

Set country/region, language and time zone, and then click "next" again.

Step 10

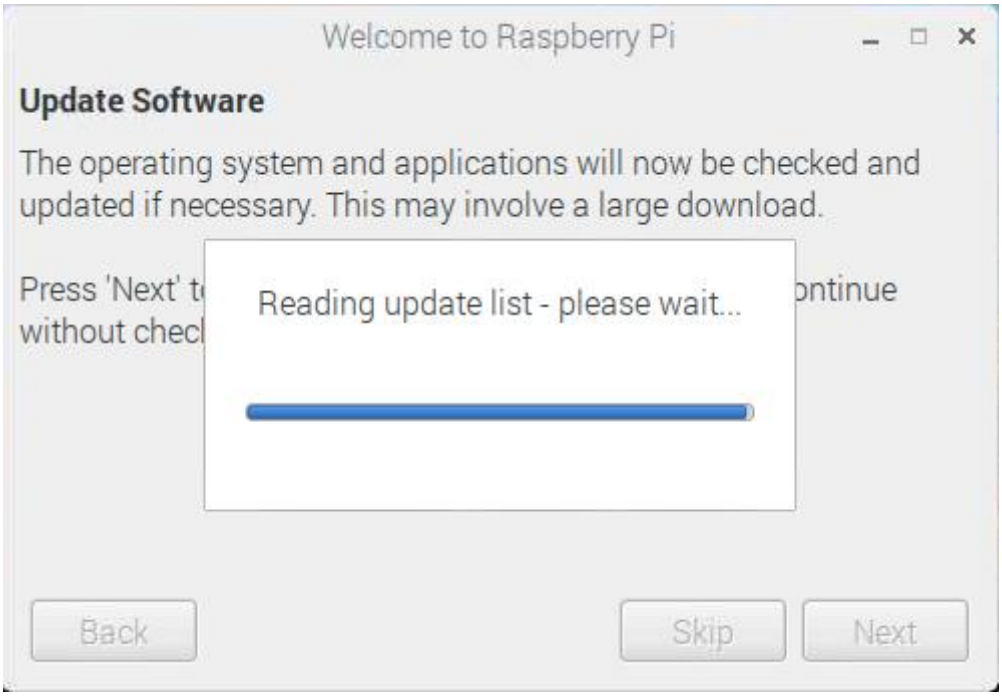
Input the new password of Raspberry Pi and click “Next”.

Step 11

Connect the Raspberry Pi to WIFI and click "Next".

Step 12

Retrieve update.



Step 13

Click "Done" to complete the Settings.

Now we can run the Raspberry Pi.

Note: You can check the complete tutorial of NOOBS on the official website of the Raspberry Pi: *https://www.raspberrypi.org/help/noobs-setup/*.

If You Have No Screen

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

Required Components

Any Raspberry Pi	1 * 2.5A Power Adapter
Micro SD card	1 * Personal computer

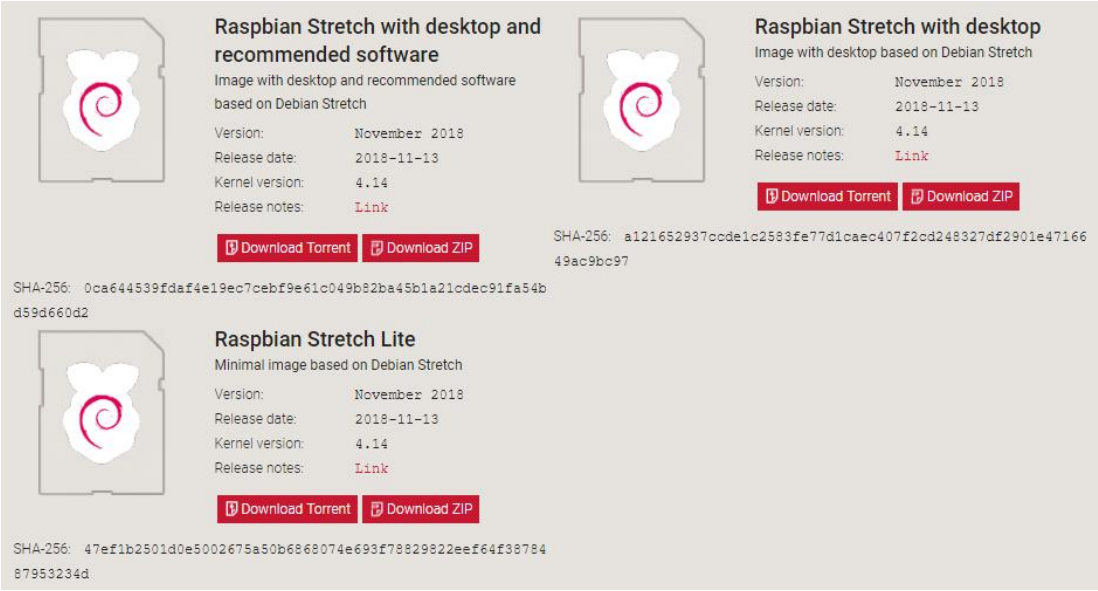
Burn System

Step 1

Prepare the tool of image burning. Here we use the **Etcher**. You can download the software from the link: *https://www.balena.io/etcher/*

Step 2

Download the complete image on the official website by clicking this link: *https://www.raspberrypi.org/downloads/raspbian/*. There are three different kinds of Raspbian Stretches available, among which the Raspbian Stretch with desktop will be the best choice if you have no other special requirements.



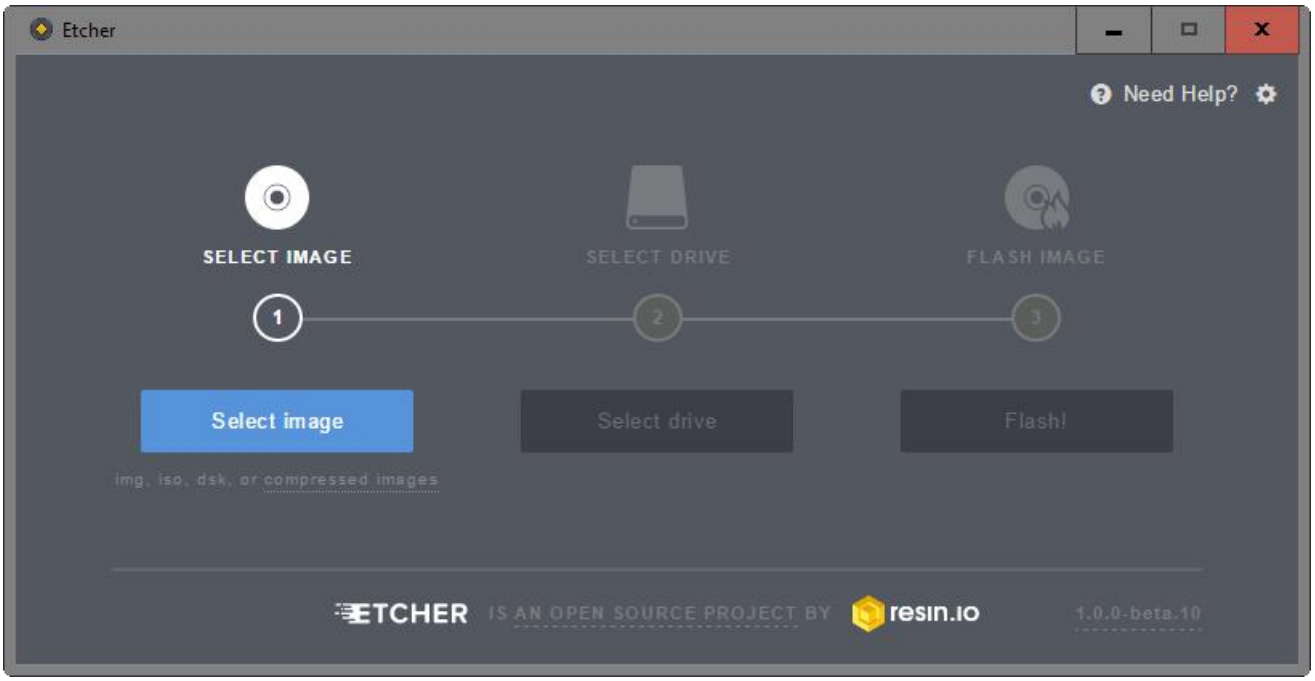
Step 3

Unzip the package downloaded and you will see the *xxxx-xx-xx-raspbian-stretch.img* file inside.

> **Note**: DO NOT extract the file.

Step 4

With the application of Etcher, flash the image file, raspbian into the SD card.



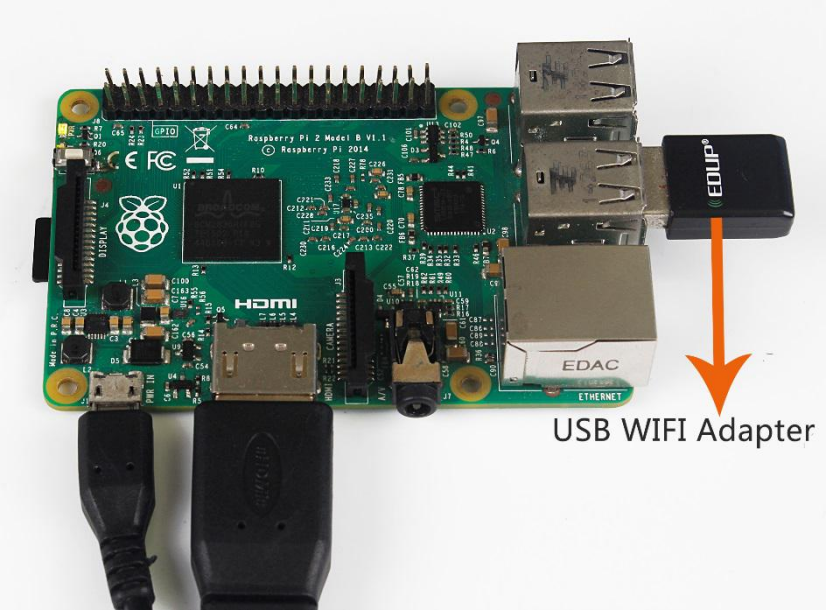
Step 5

At this point, raspbian is installed; however, if you want to apply it ,what you need do next is to complete the settings accordingly.

Connect the Raspberry Pi to the Internet

There are two methods to help get the Raspberry Pi connected to the network: the first one is using a network cable, the other way is using WIFI. We will talk in detail about how to connect via WIFI as below.

Since the 3B and above version of the product, Raspberry Pi has a built-in Wifi function. If what you use is the early version of Raspberry Pi, a USB WIFI Adapter is needed. Log in the website, https://elinux.org/RPi_USB_Wi-Fi_Adapters for more.



If you want to use the WIFI function, you need to modify a WIFI configuration file wpa-supplicant.conf in the SD card by your PC that is located in the directory /etc/wpa-supplicant/.

If your personal computer is working on a linux system, you can access the directory directly to modify the configuration file; however, if your PC use Windows system, then you can't access the directory and what you need next is to go to the directory, */boot/* to create a new file with the same name, wpa-supplicant. conf.

Input the following content in the file.

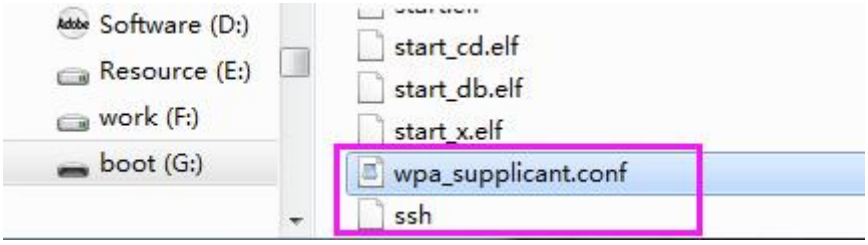
```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB
network={
ssid="WiFi-A"
psk="Sunfounder"
key_mgmt=WPA-PSK
priority=1
}
```

You need to replace "**WiFi-A**" with your custom name of WiFi and "**Sunfounder**" with your password. By doing these, the Raspbian system will move this file to the target

directory automatically to overwrite the original WIFI configuration file when it runs next time.

Start SSH

To use the function of remote control of the Raspberry Pi, you need to start SSH firstly that is a more reliable protocol providing security for remote login sessions and other network services. Generally, SSH of Raspberry Pi is in a disabled state. Additionally, if you want to run it, you need to create a file named SSH under directory /boot/.



Now, the Raspbian system is configured. When the SD card is inserted into the Raspberry Pi, you can use it immediately.

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 system, Raspbian 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 Raspbian system is **raspberrypi**, now you need to find the hostname.

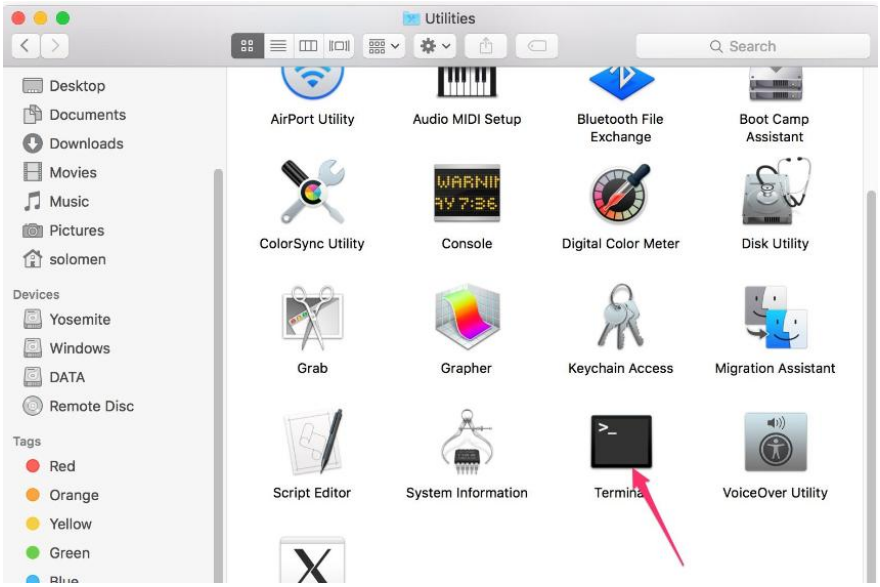
Use the SSH Remote Control

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.



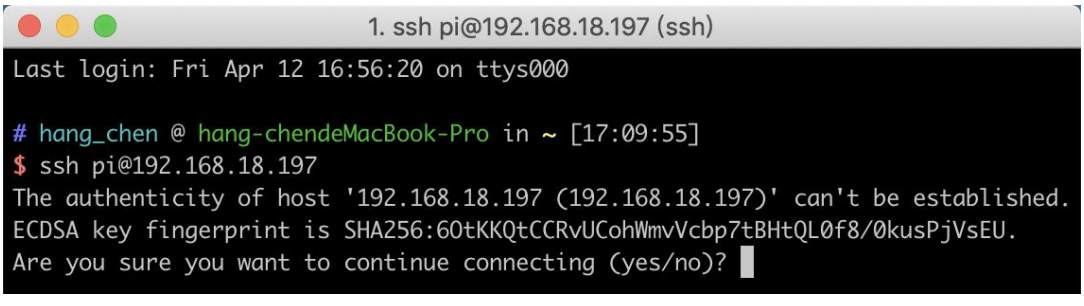
Step 2

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
```

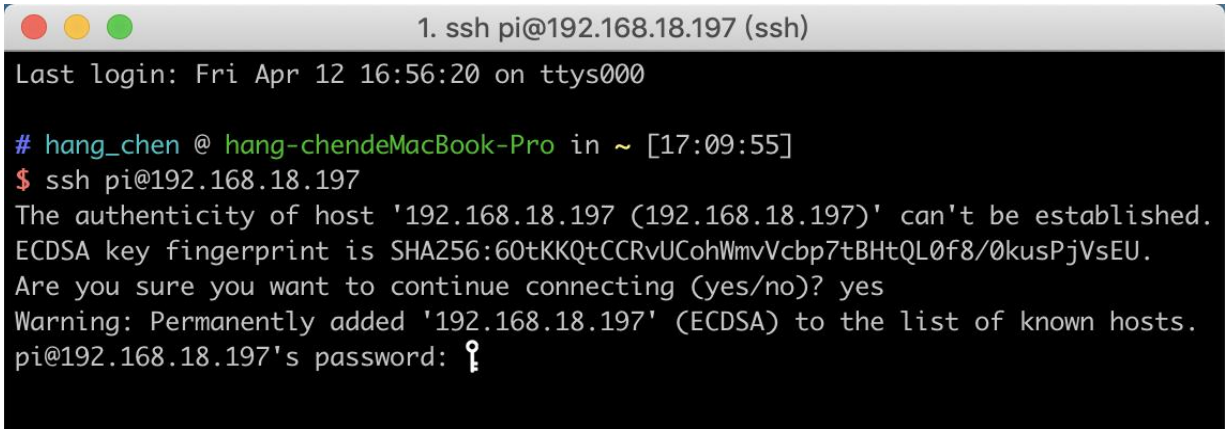
Step 3

Input “yes”.



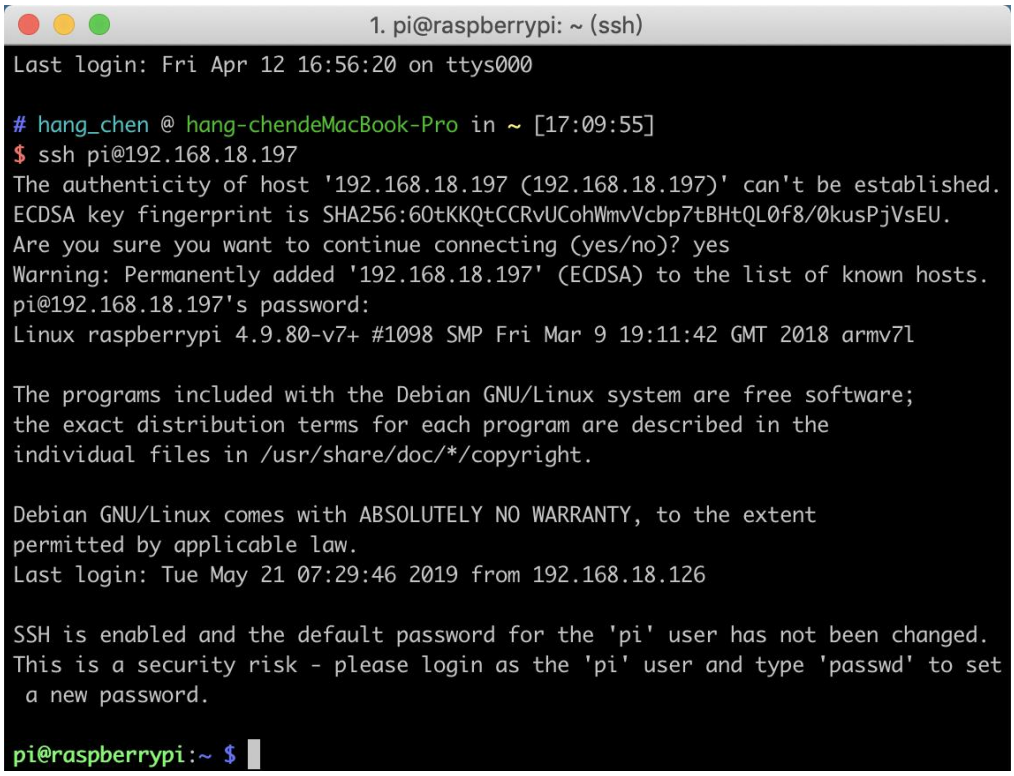
Step 4

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

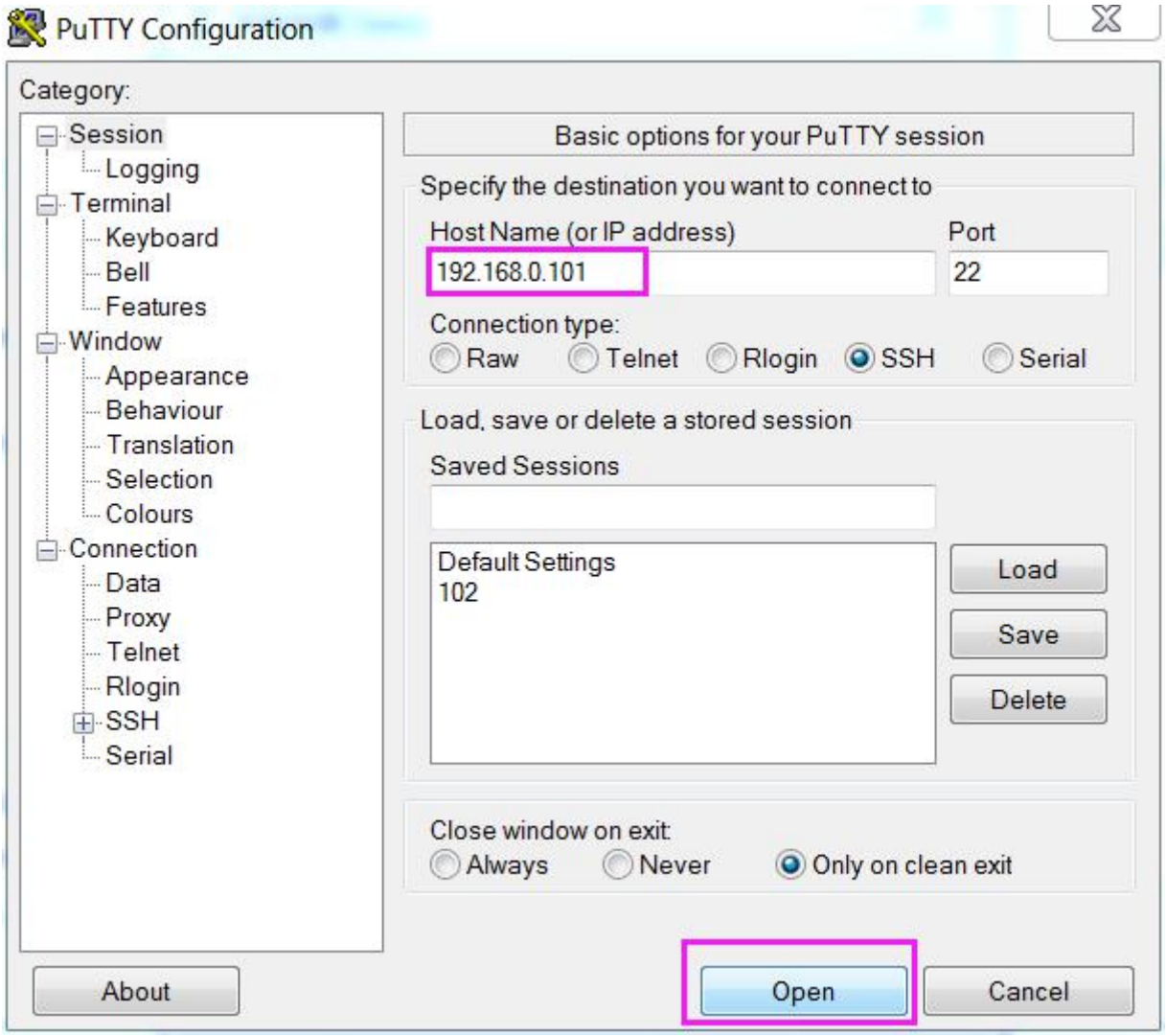
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

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).

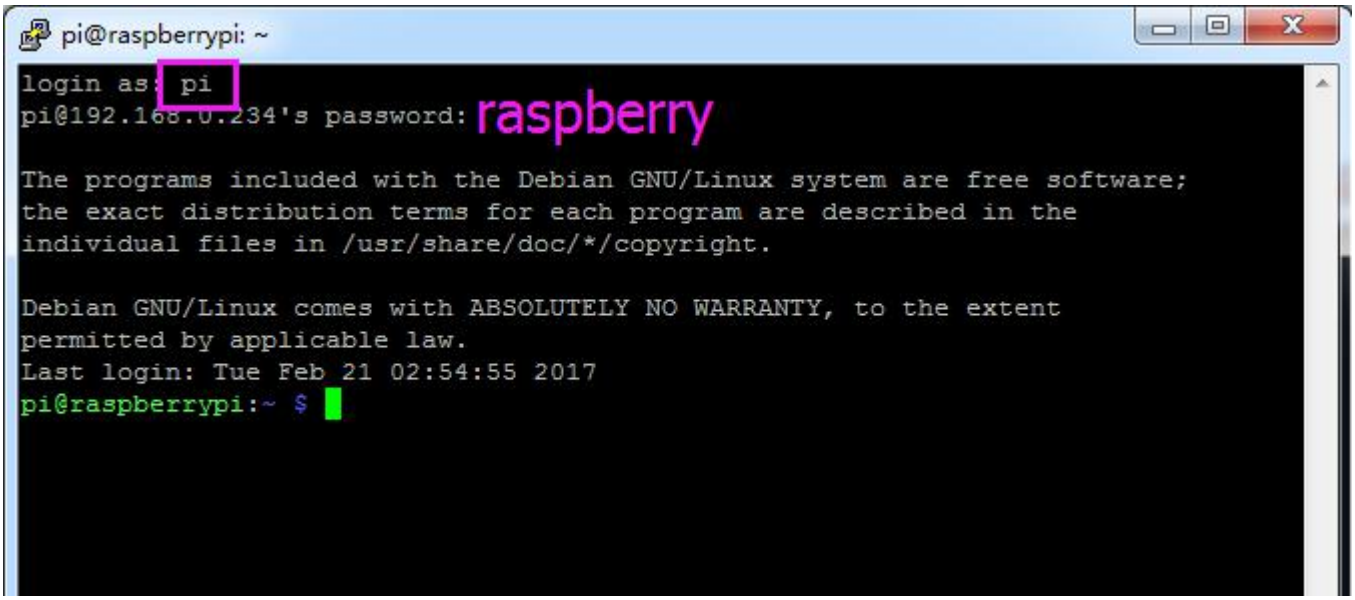


Step 3

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.

Remote Desktop

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. There are two ways to control the desktop of the Raspberry Pi remotely : **VNC** and **XRDP.**

VNC

You can use the function of remote desktop through VNC.

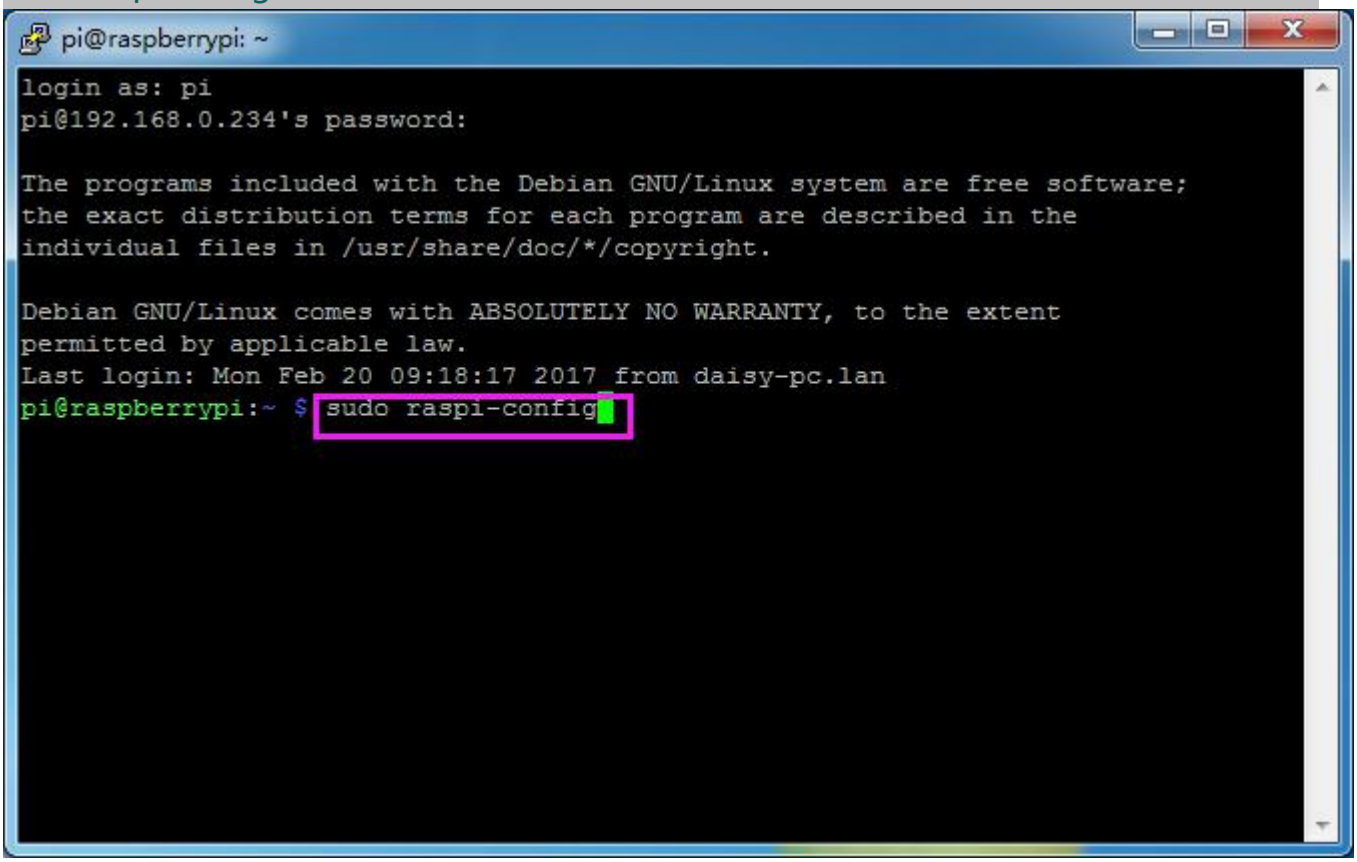
Enable VNC service

The VNC service has been installed in the system. By default, VNC is disabled. You need to enable it in config.

Step 1

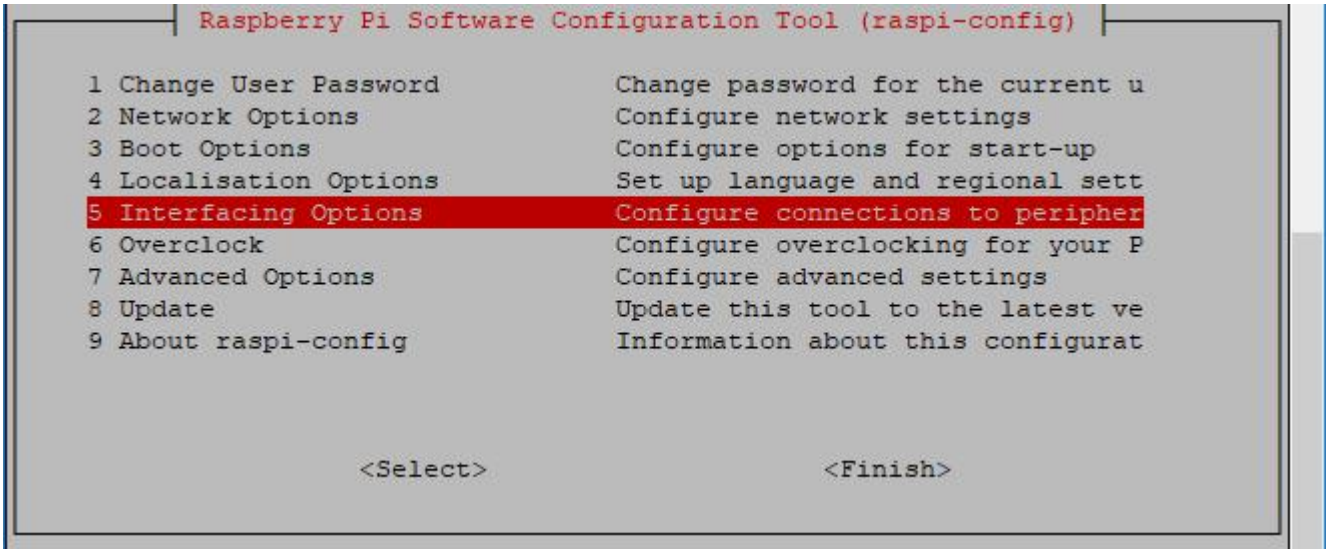
Input the following command:

```
sudo raspi-config
```



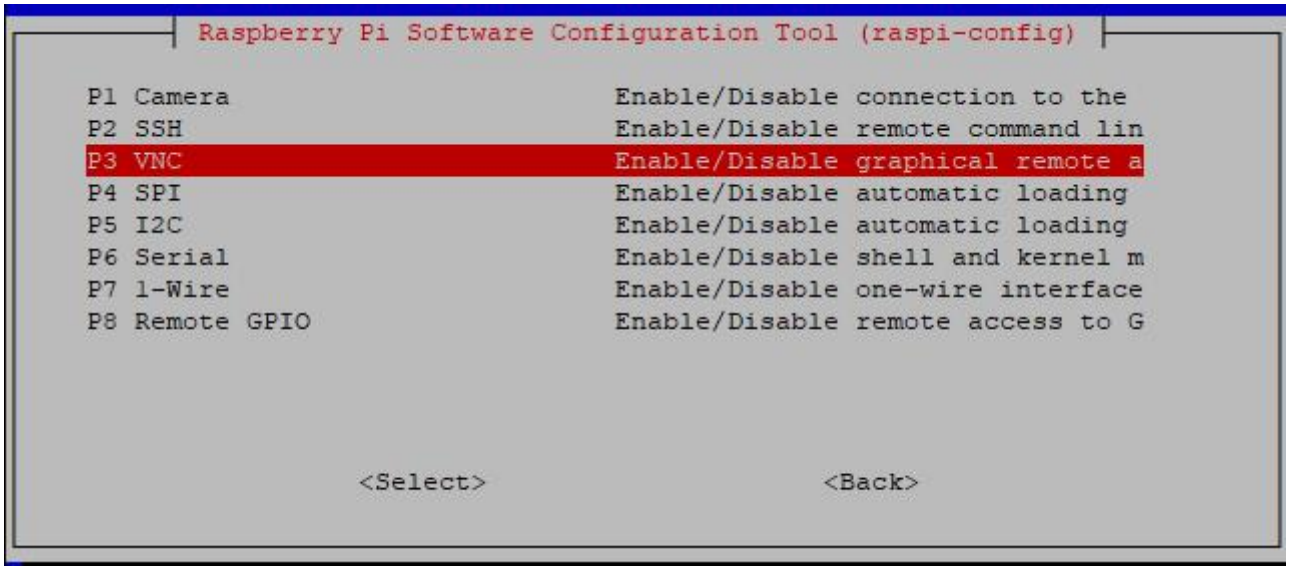
Step 2

On the config interface, select "**Interfacing Options**" by the up, down, left and right keys on the keyboard.



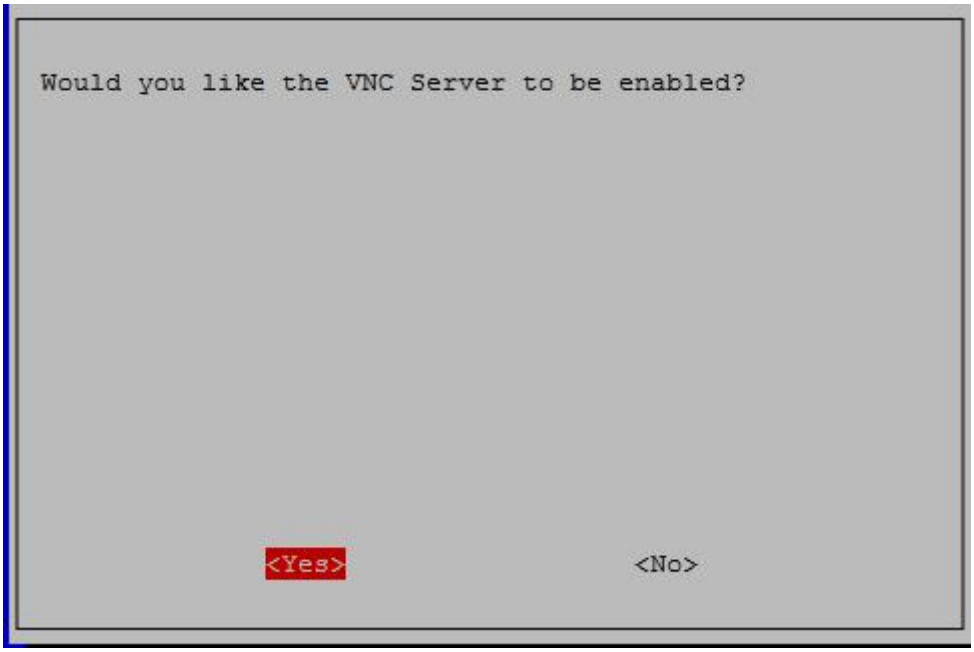
Step 3

Select **VNC**.



Step 4

Select **Yes -> OK -> Finish** to exit the configuration.



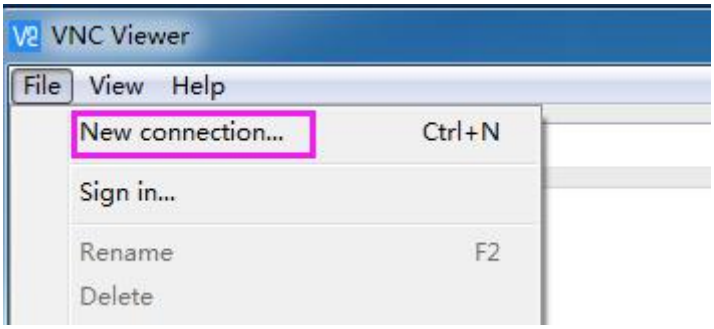
Login to VNC

Step 1

You need to install the **VNC Viewer** on personal computer. After the installation is done, open it.

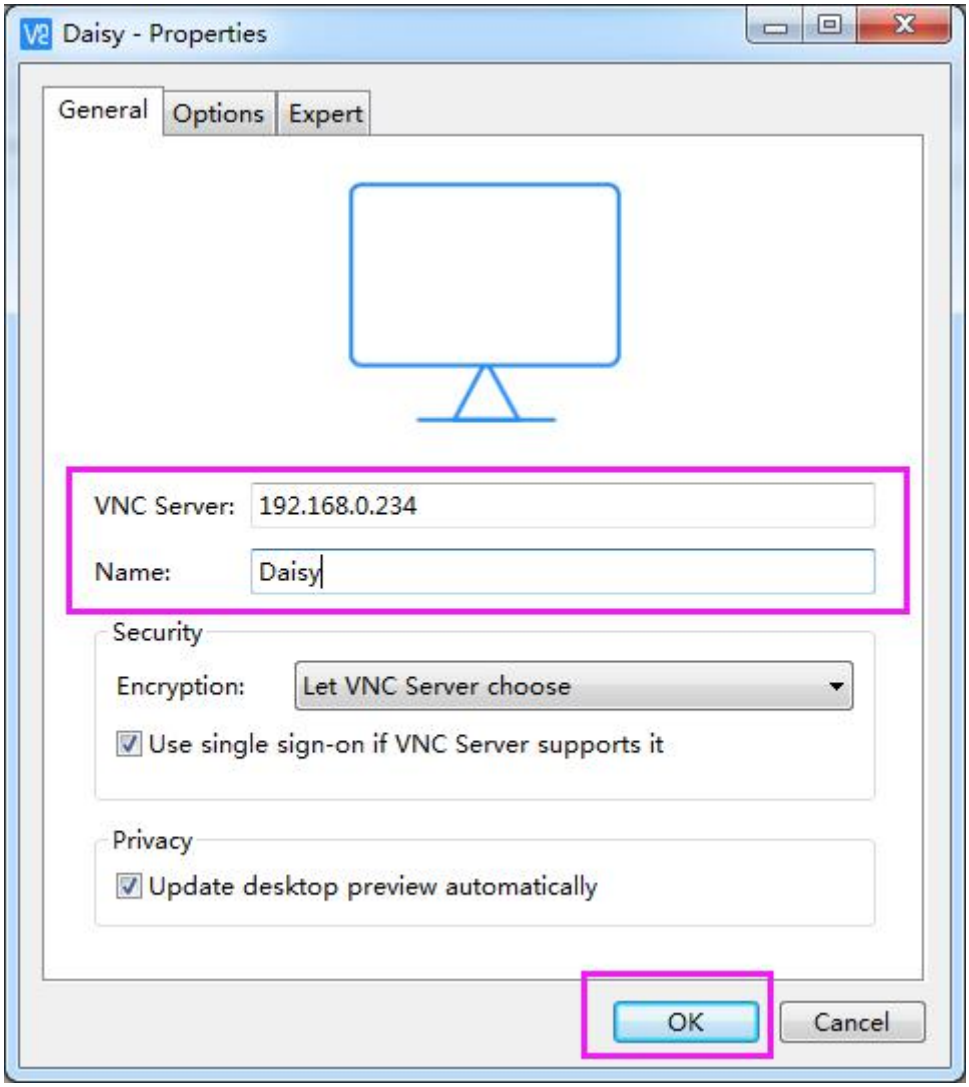
Step 2

Then select "**New connection**".



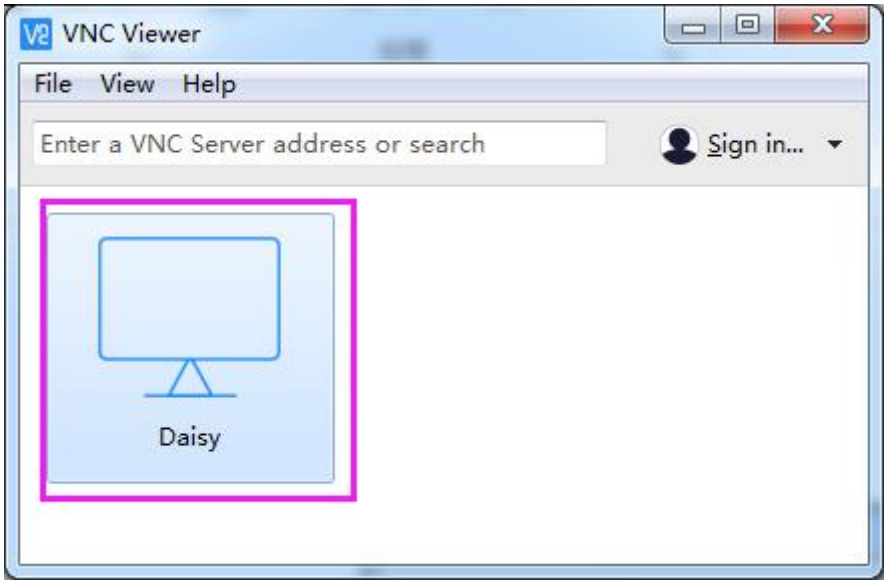
Step 3

Input IP address of Raspberry Pi and any **Name**.



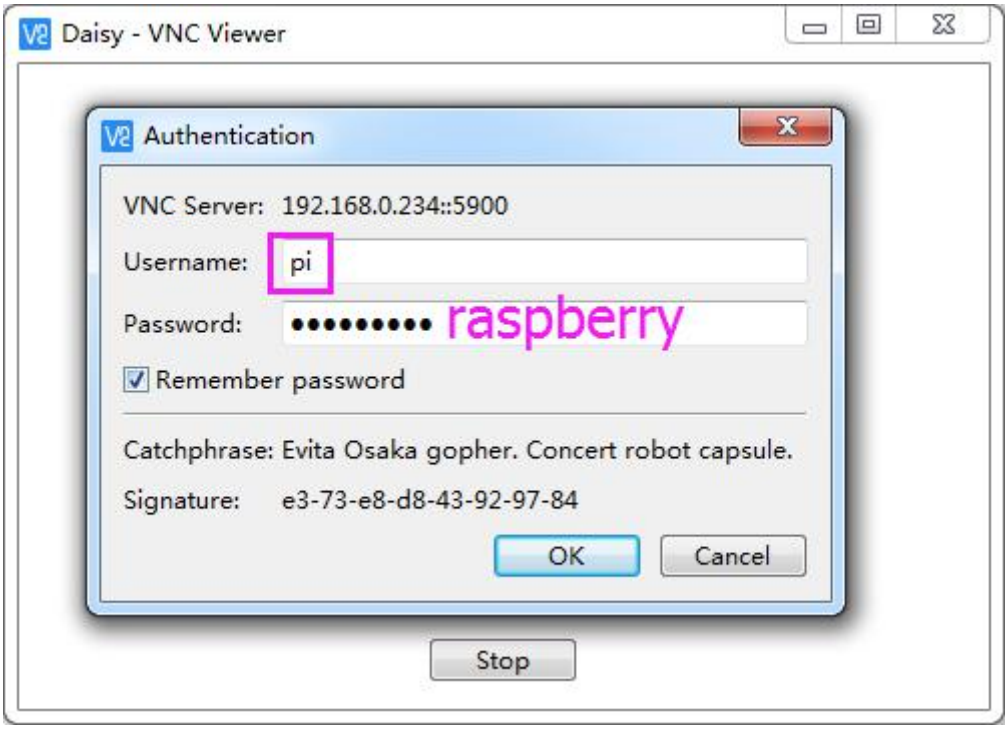
Step 4

Double click the **connection** just created:



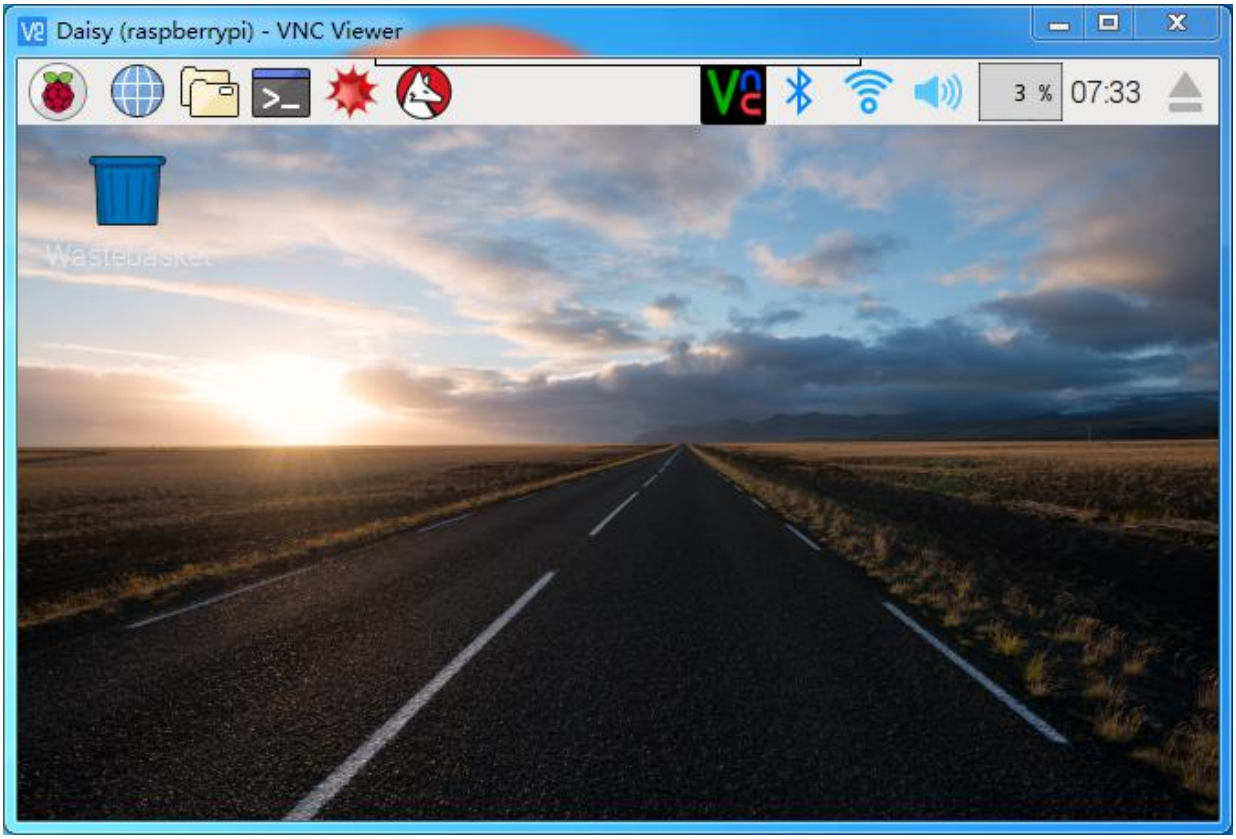
Step 5

Enter Username (**pi**) and Password (**raspberry** by default).



Step 6

Now you can see the desktop of the Raspberry Pi:



XRDP

xrdp 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

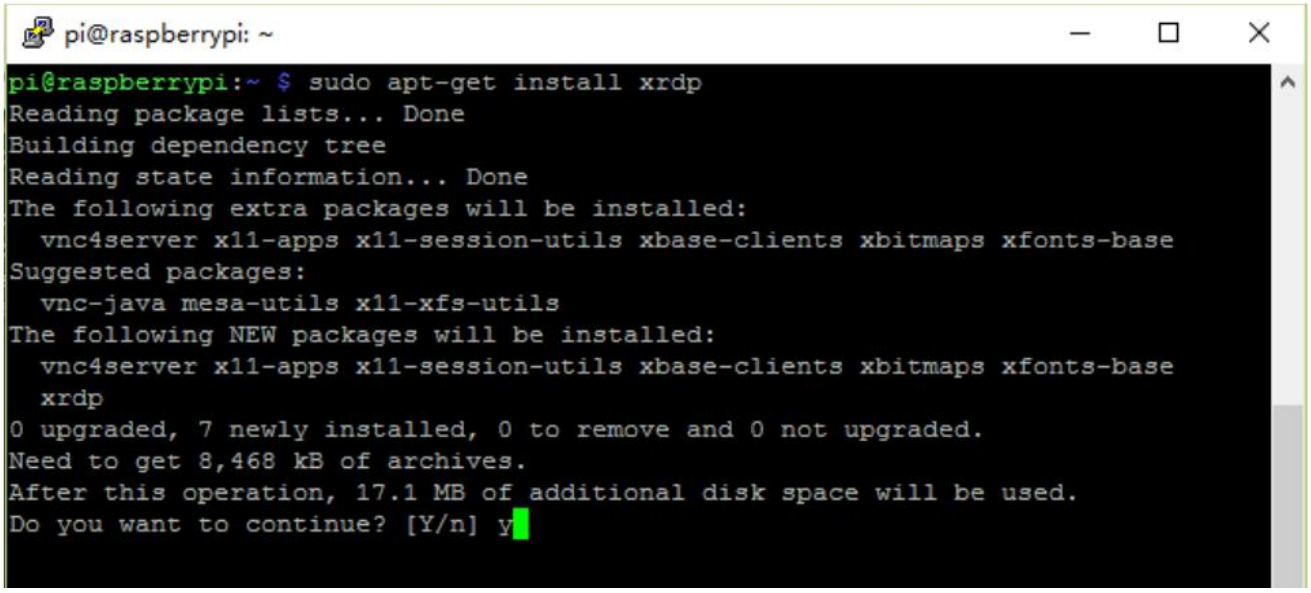
Input the following instructions to install XRDP.

```
sudo apt-get update
sudo apt-get install xrdp
```

Step 3

Later, the installation starts.

Enter "Y", press key “Enter” to confirm.



Step 4

After the installation is completed, you can use Windows remote desktop applications to login to your RPi.

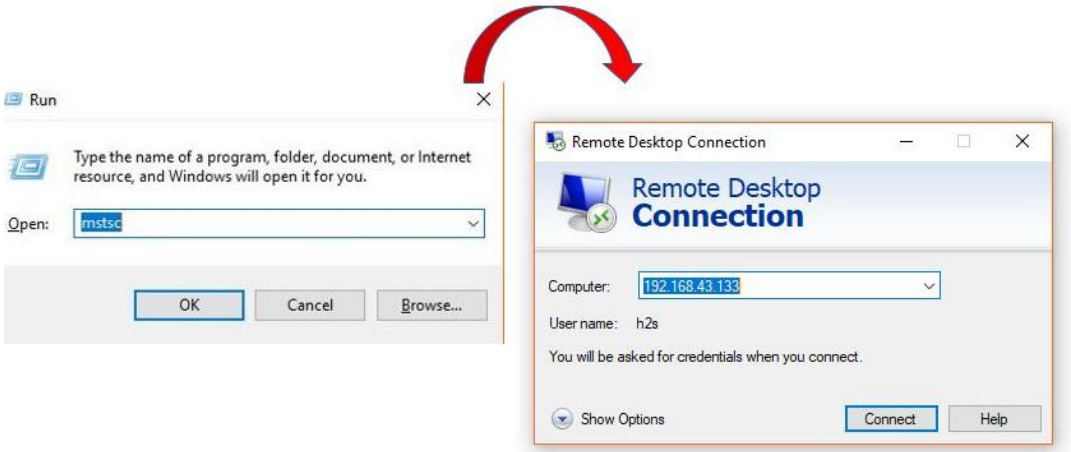
Login to XRDP

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.

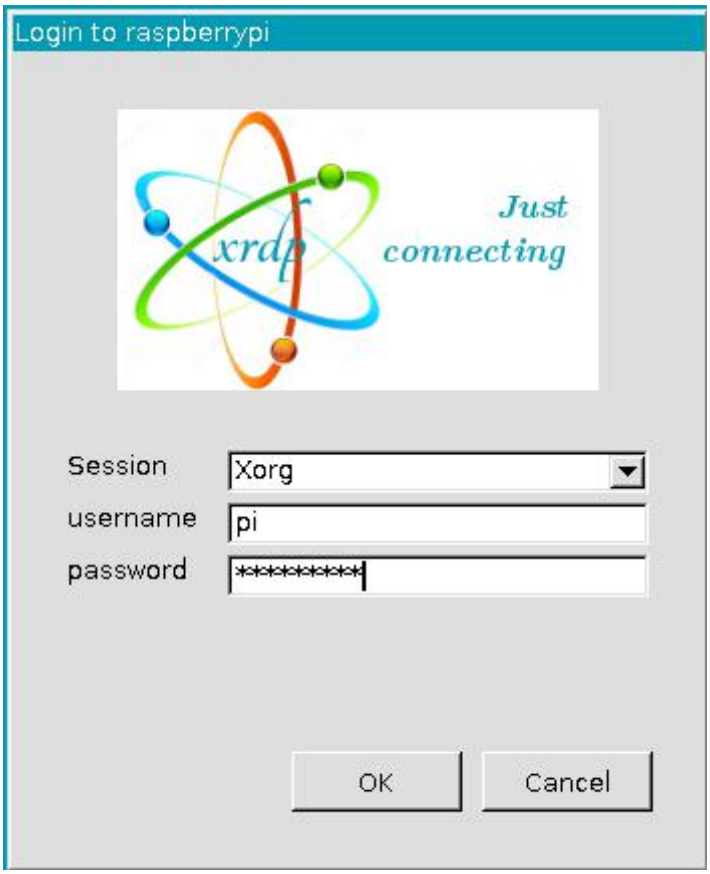
Step 2

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".



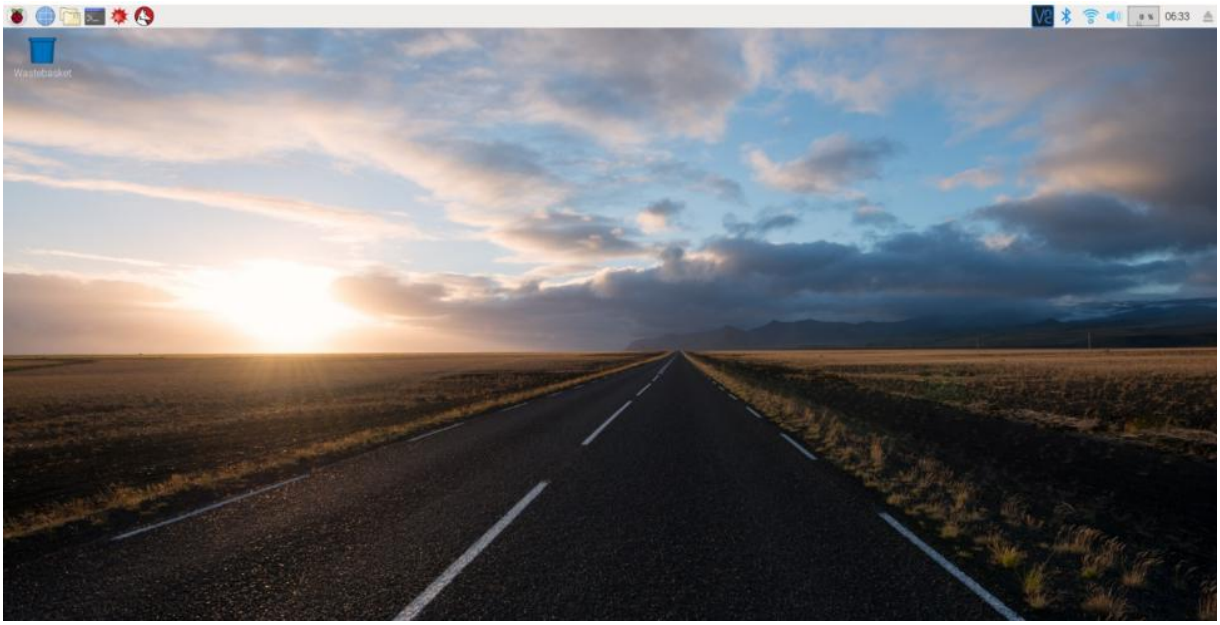
Step 3

There will be xrdp login screen. Enter the user name and password of RPi and click OK. By default, the user name of Raspberry Pi is **"pi"** and the password is **"raspberry"**.



Step 4

Here, you successfully login to RPi by using the remote desktop.



Libraries

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

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:~ $
```

WiringPi

wiringPi is a C language GPIO library applied to the Raspberry Pi platform. 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. You can test whether the wiringPi library is installed successfully or not by the following instructions.

gpio -v

```
pi@raspberrypi:~ $ gpio -v
gpio version: 2.32
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty
```

If the message above appears, the wiringPi is installed successfully.

gpio readall

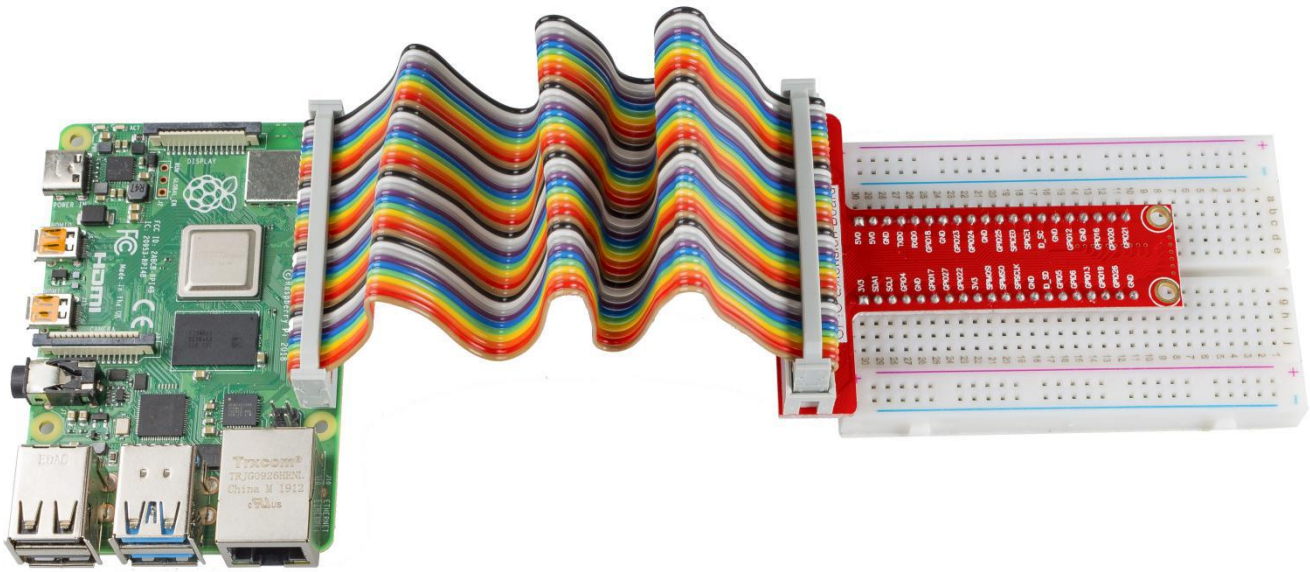
```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  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: http://wiringpi.com/download-and-install/

Raspberry Pi GPIO Extension Board

We apply the GPIO Extension Board to extend the pins of Raspberry Pi to the breadboard and avoid damage caused by frequent plugging and unplugging.

Here, we apply a 40-pin GPIO Extension board and a 40-pin GPIO cable. In case of the potential risk of short circuit, you must build your circuit in strict accordance with the following picture.



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:

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

2) In 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

Download the Code

Change directory to */home/pi*

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

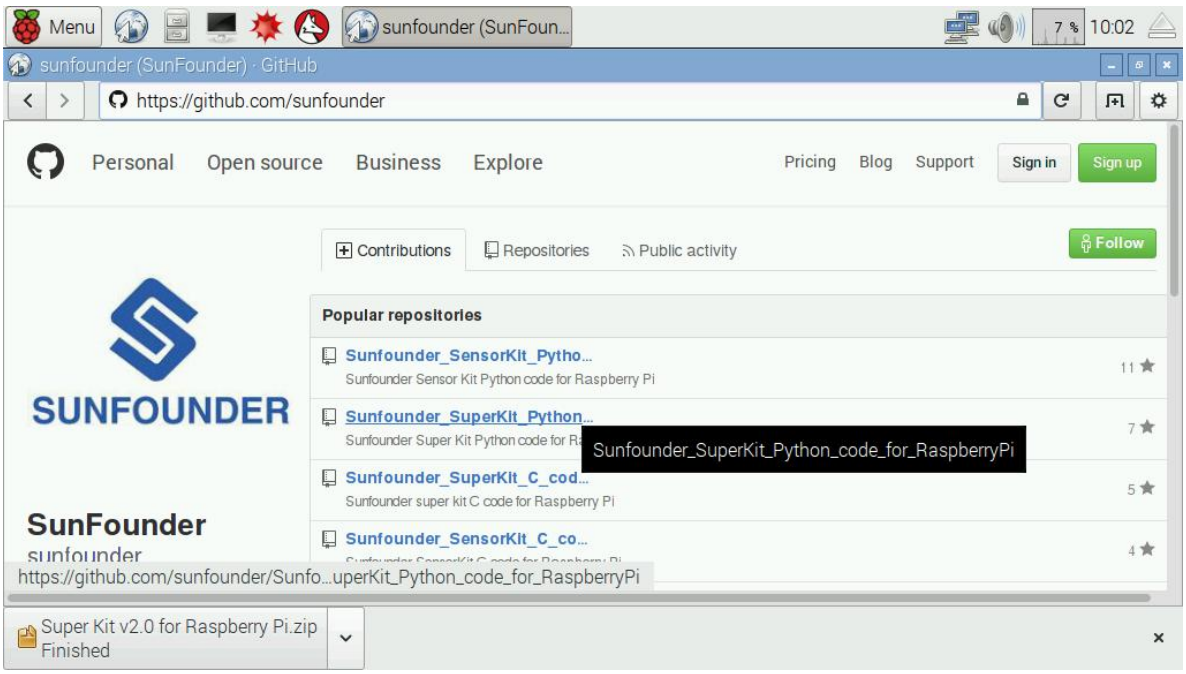
Note: cd, short for change directory is to change to the intended directory from the current path. Informally, here is to go to the path */home/pi/.*

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

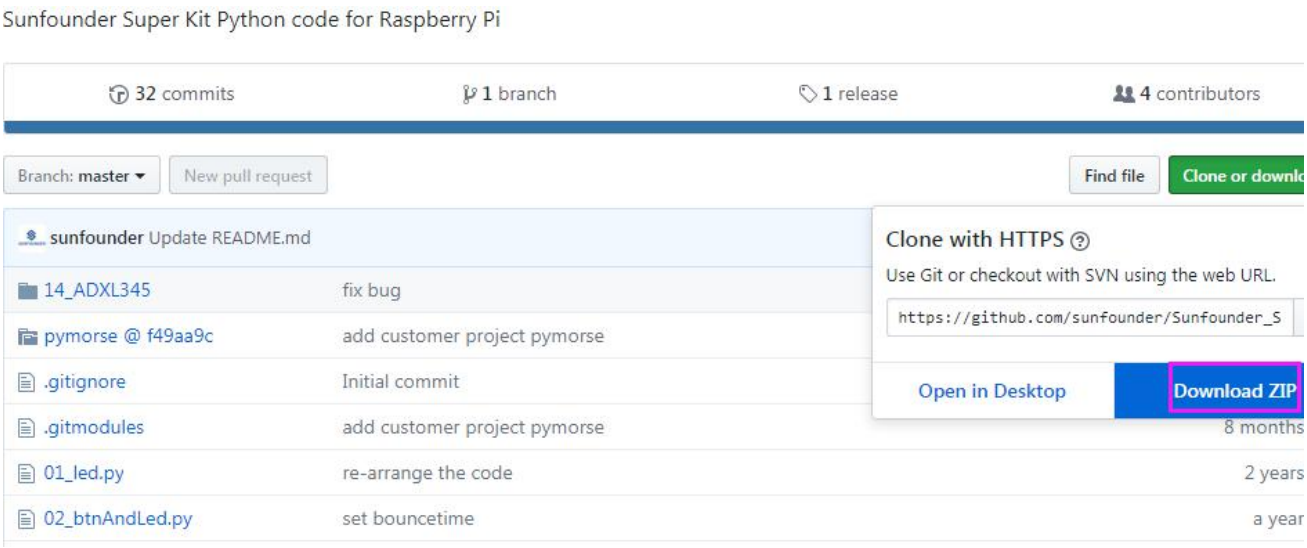
```
git clone https://github.com/sunfounder/Sunfounder_SuperKit_C_code_for_RaspberryPi.git
```

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

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 Super 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!

Lesson 1 Blinking LED

Introduction

In this lesson, we will learn how to program Raspberry Pi to make an LED blink. You can play numerous tricks with an LED as you want. Now get to start and you will enjoy the fun of DIY at once!

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * LED

- 1 * Resistor (220Ω)

- Jumper wires

Principle

Semiconductor light-emitting diode is a type of component which can turn electric energy into light energy via PN junctions. By wavelength, it can be categorized into laser diode, infrared light-emitting diode and visible light-emitting diode which is usually known as light-emitting diode (LED).

When 2V-3V forward voltage is supplied to an LED, it will blink only if forward currents flow through the LED. Usually there are red, yellow, green, blue and color-changing LEDs which change color with different voltages. LEDs are widely used due to their low operating voltage, low current, luminescent stability and small size.

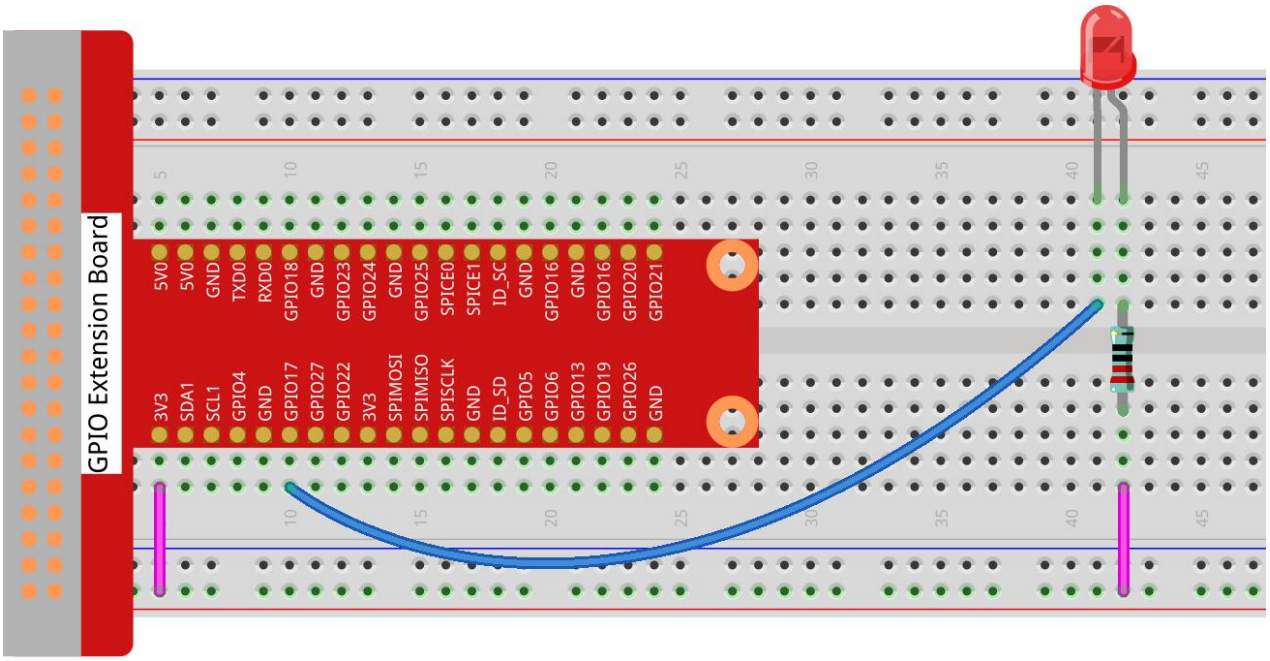
LEDs are diodes too. Hence they have a voltage drop which usually varies from 1V to 3V depending on their types. Generally, they brighten if supplied with a 5mA–30mA current, and we usually use 10mA–20mA. Thus when an LED is used, it is necessary to connect a current-limiting resistor to protect it from being burnt.

Schematic Diagram

In this experiment, connect a 220Ω resistor to the anode of the LED, then the resistor to 3.3 V, and connect the cathode of the LED to GPIO17 (See Raspberry Pi Pin Number Introduction). Write 1 to GPIO17, and the LED will stay off; write 0 to GPIO17, and then the LED will blink, just as indicated by the principle above.

Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/01_LED/
```

Step 3: Compile.

```
gcc led.c -o led -lwiringPi
```

Step 4: Run.

```
sudo ./led
```

For Python Users:

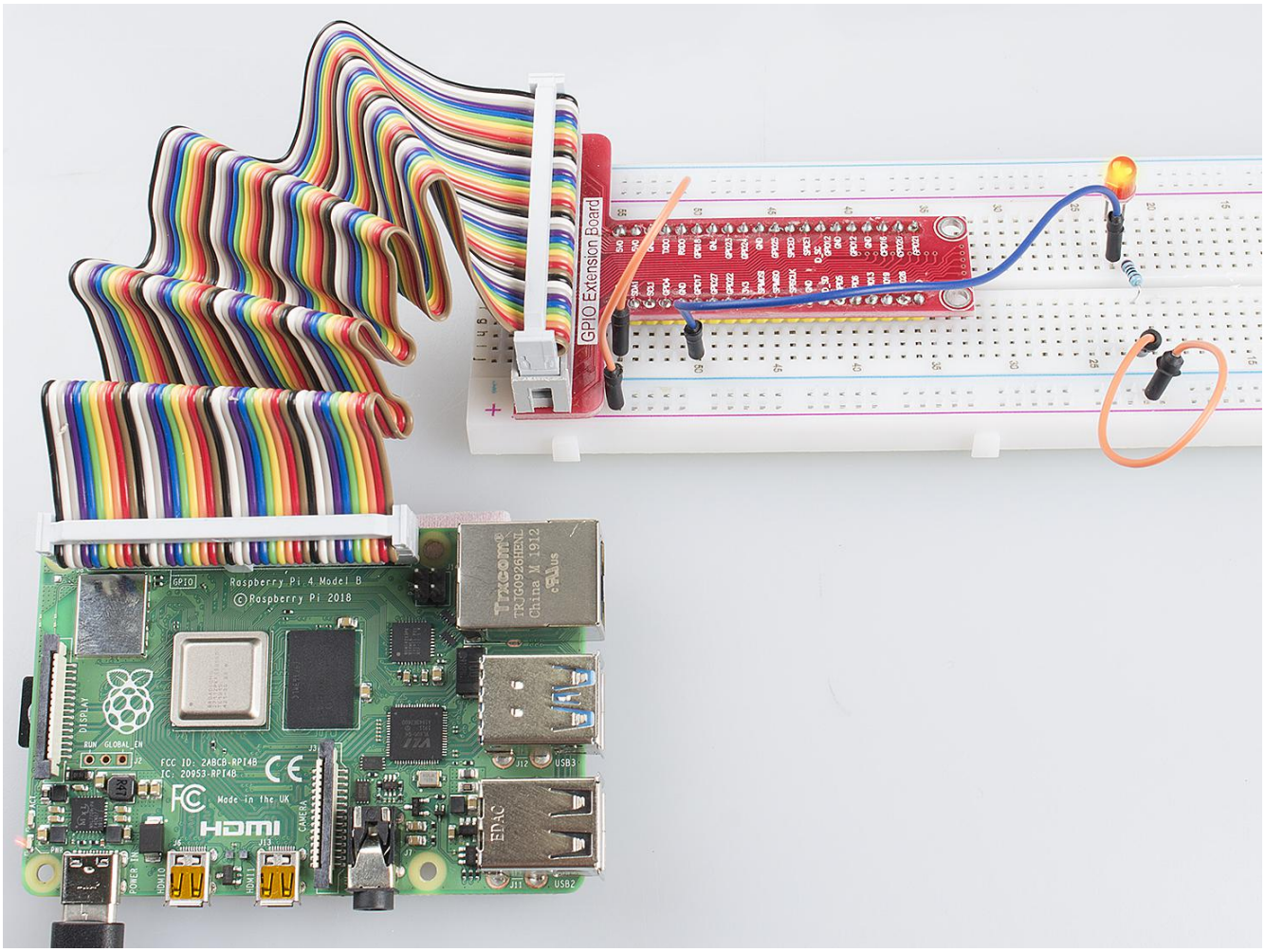
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 01_led.py
```

Now, you should see the LED blink.



Further Exploration

If you want the LED to speed up the blinking, just change the delay time. For example, change the time to *delay (200)* in the program, recompile and run, and then you will see the LED blink faster.

Summary

Raspberry Pi packages many low-level detail designs, which enable you to explore your own apps more conveniently. Maybe that is the charm of Raspberry Pi.

Now you have already learnt how to use the Raspberry Pi GPIO to blink an LED. Keep moving to the next contents.

Tips: For any **TECHNICAL** questions, add a topic under **FORUM** section on our website www.sunfounder.com and we'll reply as soon as possible. For **NON-TECH** questions like order issues, please **email** service@sunfounder.com.

Lesson 2 Controlling an LED by a Button

Introduction

In this lesson, we will learn how to turn an LED on or off by a button.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * LED

- 1 * Button

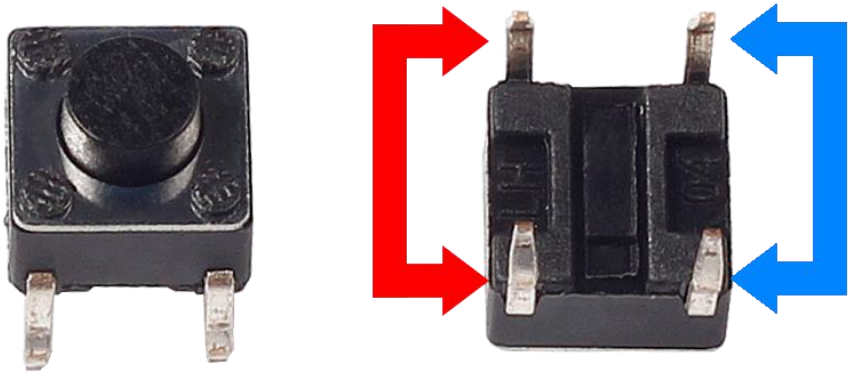
- 1 * Resistor (220Ω)

- Jumper wires

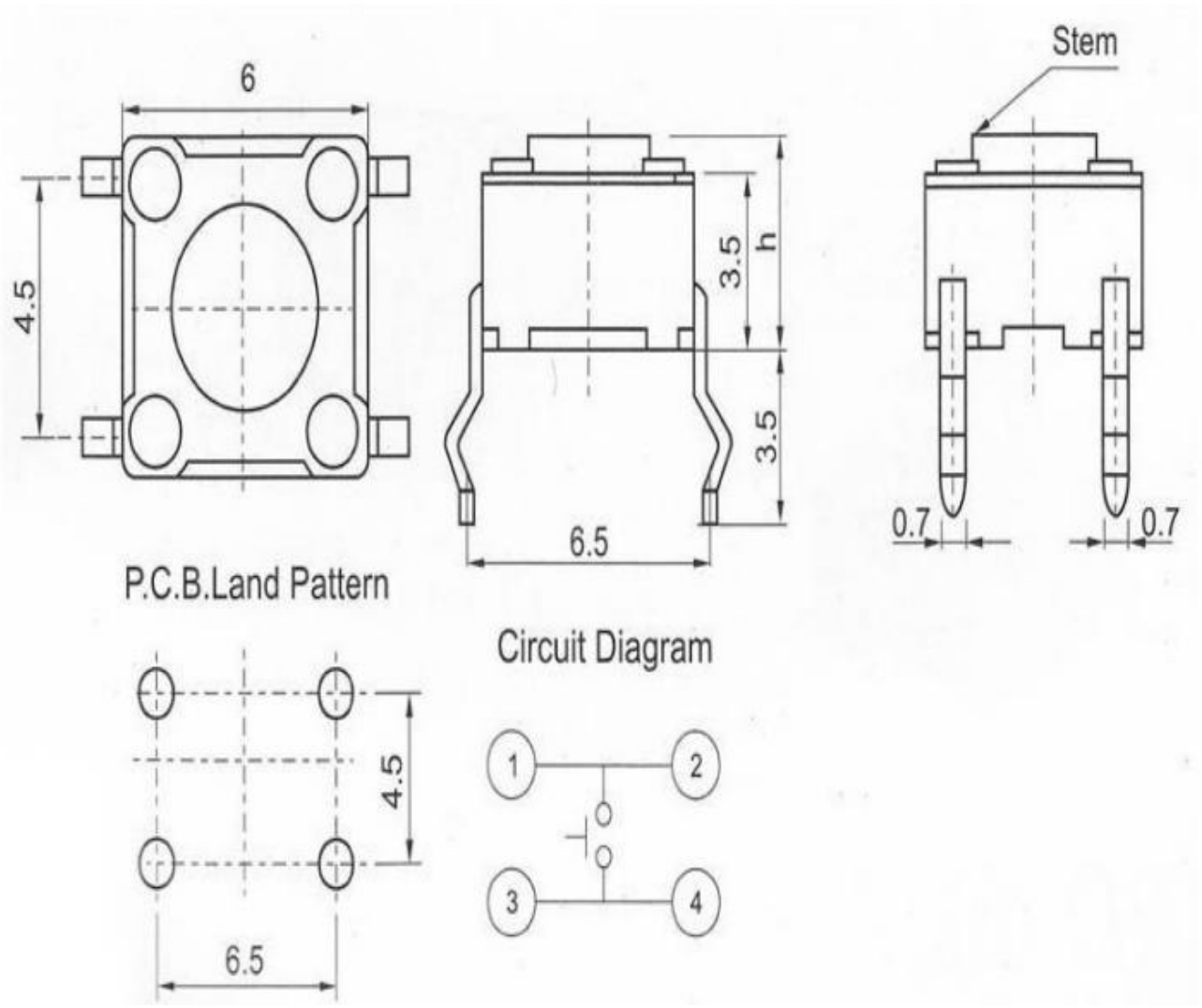
Principle

Button

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used here is a 6mm mini-button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.



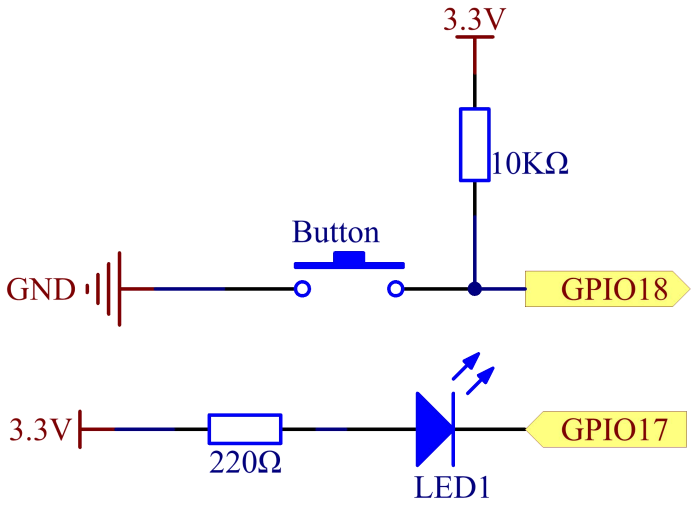
When the button is pressed, the pins pointed by the blue arrow will connect to the pins pointed by the red arrow (see the above figure), thus closing the circuit, as shown in the following diagrams.



Generally, the button can be connected directly to the LED in a circuit to turn on or off the LED, which is comparatively simple. However, sometimes the LED will brighten automatically without any button pressed, which is caused by various kinds of external interference. In order to avoid this interference, a pull-up resistor is used – usually connect a 1K–10KΩ resistor between the button and VCC. It can be connected to VCC to consume the interference when the button is off.

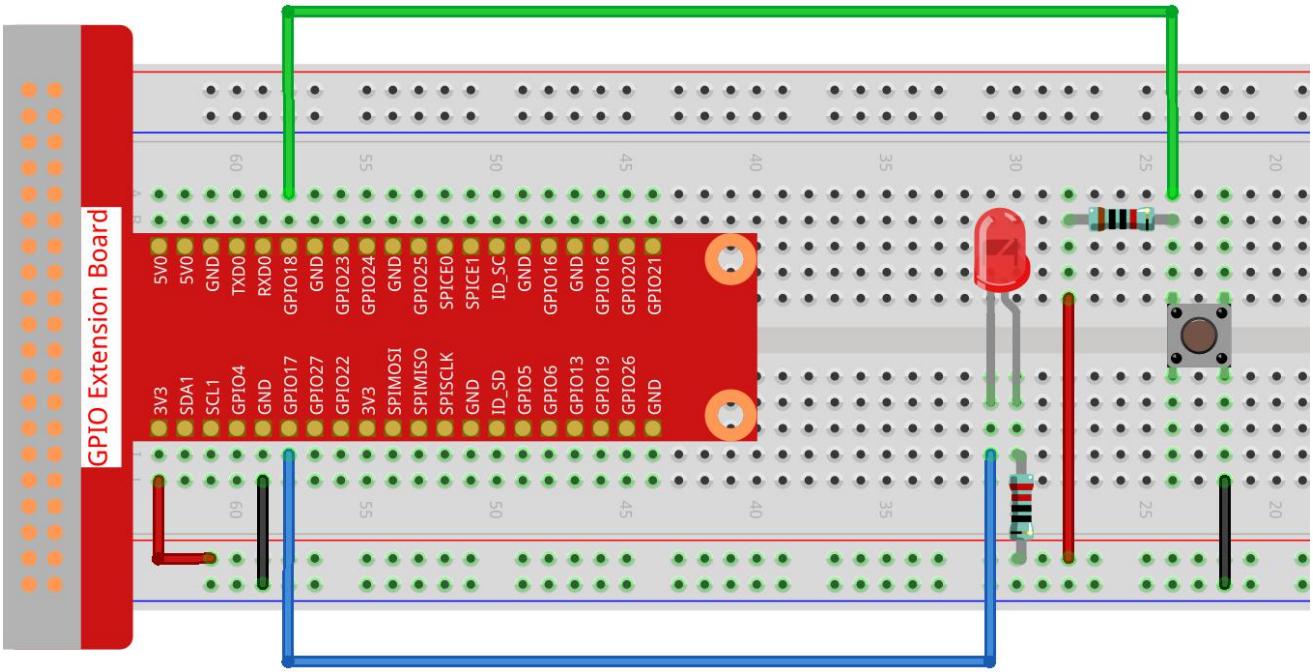
Schematic Diagram

Use a normally open button as the input of Raspberry Pi. When the button is pressed, the GPIO connected to the button will turn into low level (0V). We can detect the state of the GPIO connected to the button through programming. That is, if the GPIO turns into low level, it means the button is pressed. You can run the corresponding code when the button is pressed, and then the LED will light up.



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/02_BtnAndLed/
```

Step 3: Compile.

```
gcc BtnAndLed.c -o BtnAndLed -lwiringPi
```

Step 4: Run.

```
sudo ./BtnAndLed
```

For Python Users:

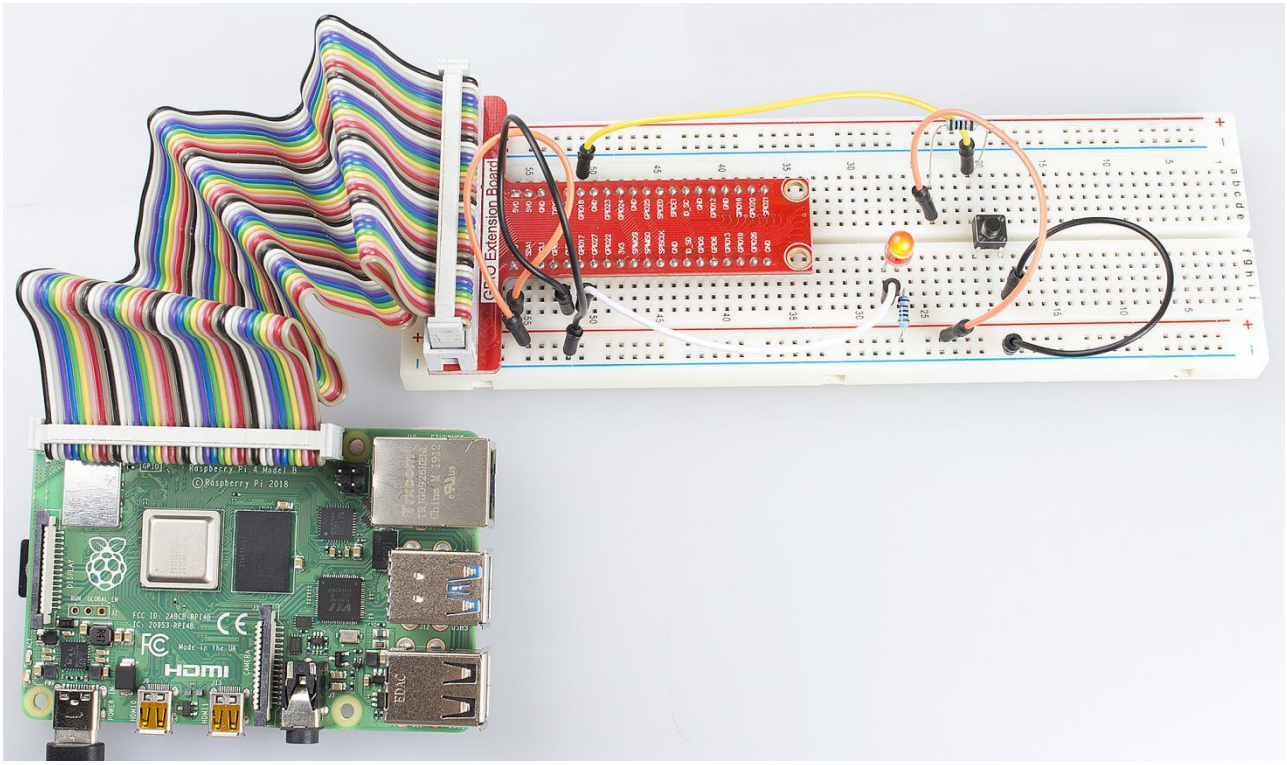
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 02_btnAndLed.py
```

Now, press the button, and the LED will light up; press the button again, and the LED will go out. At the same time, the state of the LE D will be printed on the screen.



Summary

Through this experiment, you have learnt how to control the GPIOs of the Raspberry Pi by programming.

Lesson 3 Flowing LED Lights

Introduction

In this lesson, we will learn how to make eight LEDs blink in various effects as you want based on Raspberry Pi.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

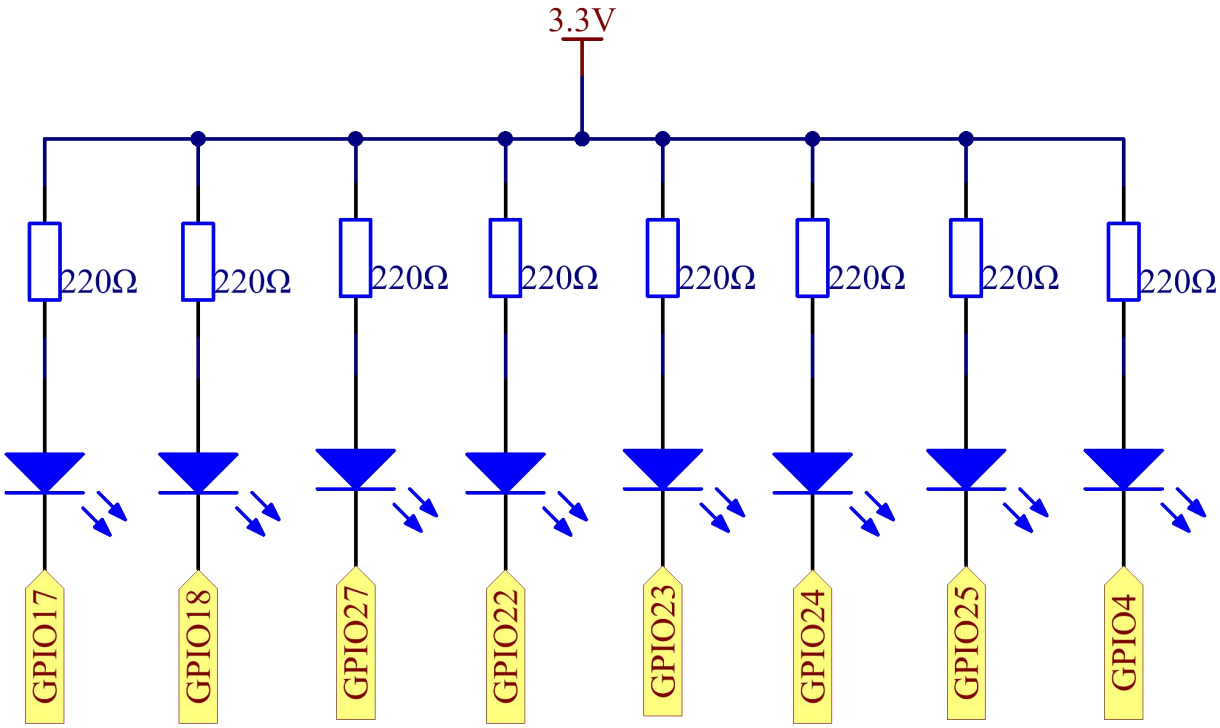
- 8 * LED

- 8 * Resistor (220Ω)

- Jumper wires

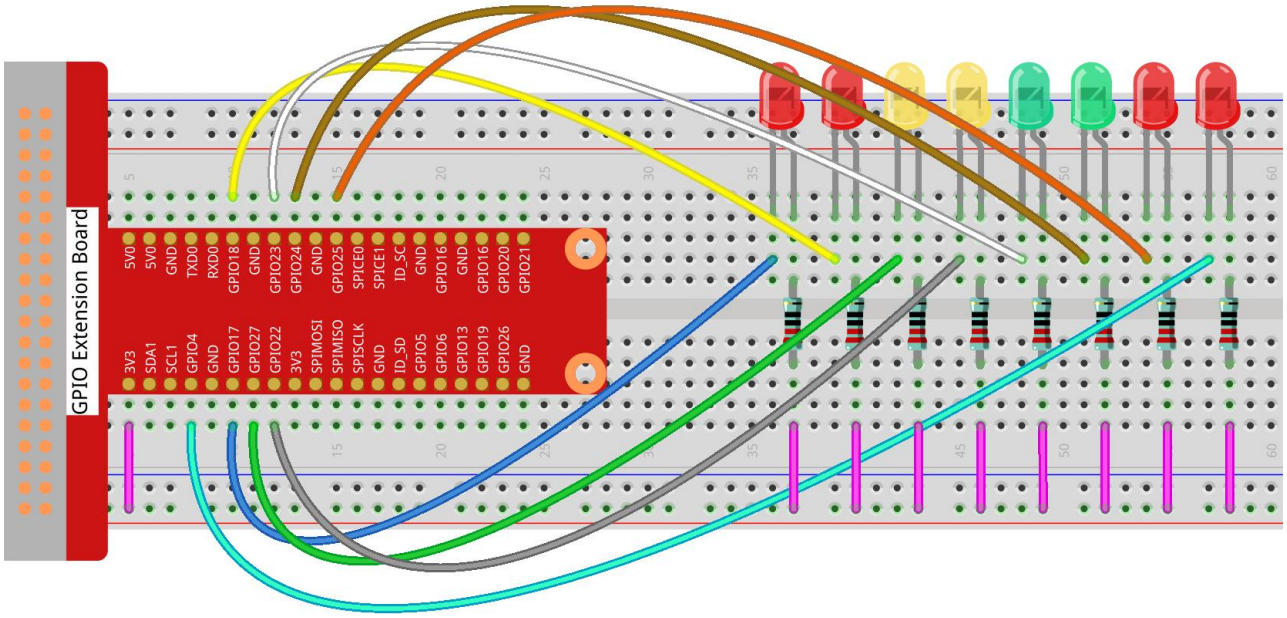
Schematic Diagram

Set GPIO17-GPIO25 to low level in turn by programming, and then LED0-LED7 will light up in turn. You can make eight LEDs blink in different effects by controlling their delay time and the order of lighting up.



Experimental Procedures

Step 1: Build the circuit.



Step 2: GPIO4 is the default pin for onewire driver (w1-gpio). In this lesson, we need to disable the onewire function and use it as an output pin.

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

Commit the following line.

```
#dtoverlay=w1-gpio
```

For C Language Users:

Step 3: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/03_8Led/
```

Step 4: Compile.

```
gcc 8Led.c -o 8Led -lwiringPi
```

Step 5: Run.

```
sudo ./8Led
```

For Python Users:

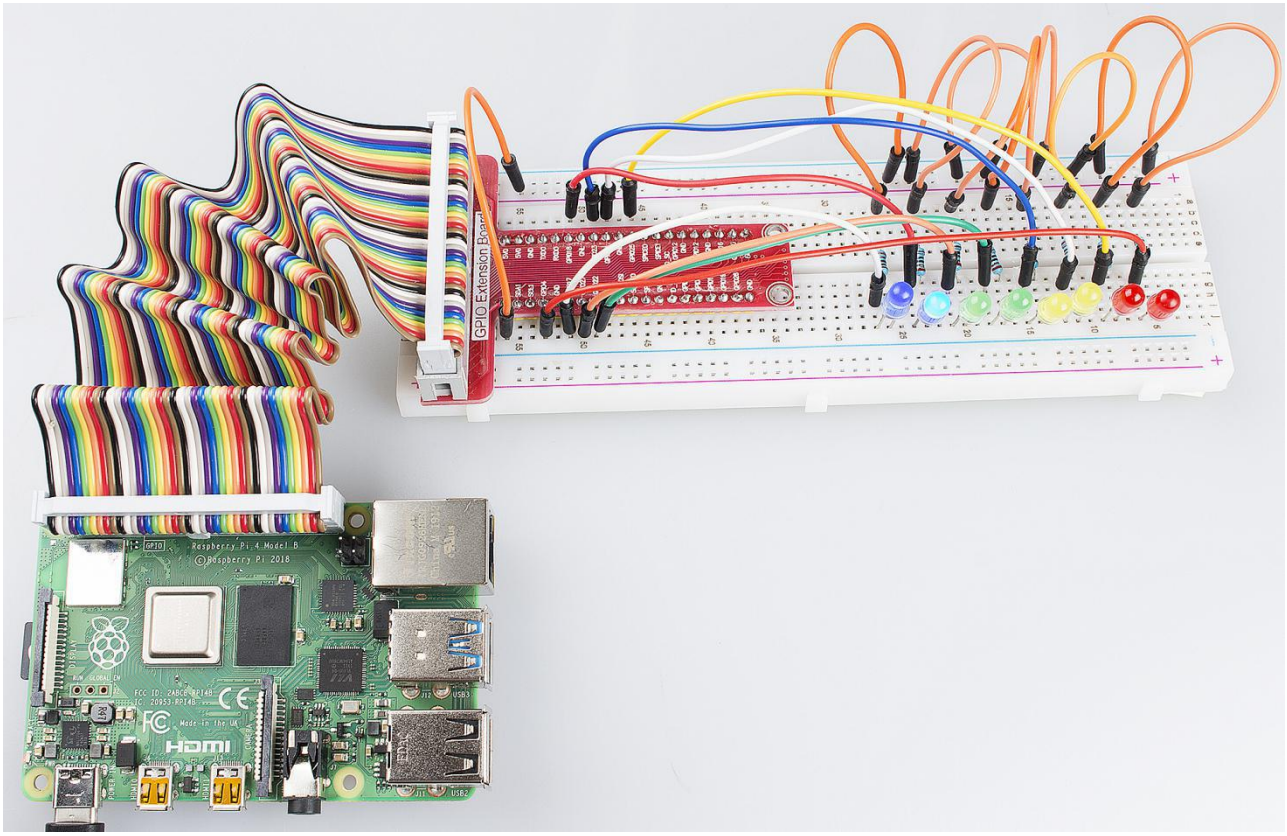
Step 3: Change directory.

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

Step 4: Run.

```
sudo python3 03_8Led.py
```

Then you will see eight LEDs brighten and dim left to right and right to left circularly, just like flowing water.



Further Exploration

You can write the blinking effects of LEDs in an array. If you want to use one of these effects, you can call it in the *main()* function directly.

Lesson 4 Breathing LED

Introduction

In this lesson, we will try something interesting – gradually increase and decrease the luminance of an LED with PWM, just like breathing. So we give it a magical name - Breathing LED.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * LED

- 1 * Resistor (220Ω)

- Jumper wires

Principle

PWM

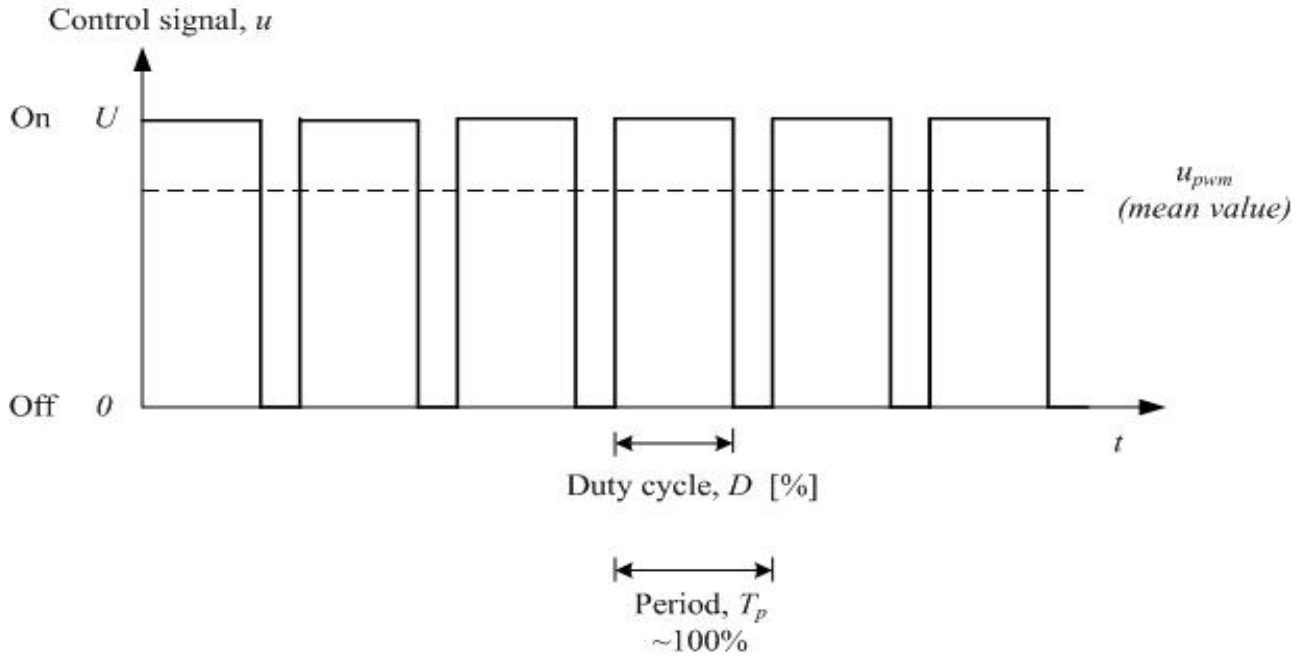
Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (3.3 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called pulse width. To get varying analog values, you change, or modulate, that width. If you repeat this on-off pattern fast enough with some device, an LED for example, the result would be like this: the signal is a steady voltage between 0 and 3.3v controlling the brightness of the LED. (See the PWM description on the official website of Arduino)

Duty Cycle

A duty cycle is the percentage of one period in which a signal is active. A period is the time it takes for a signal to complete an on-and-off cycle. As a formula, a duty cycle may be expressed as:

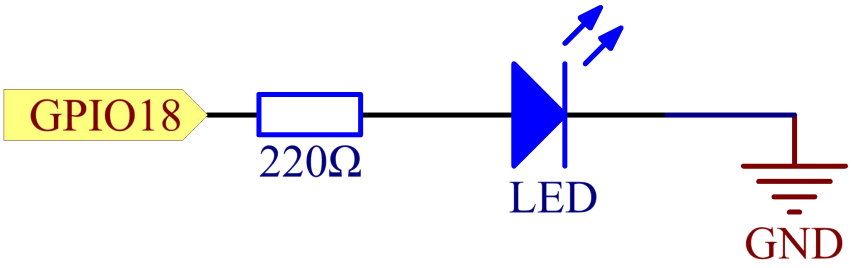
$$D = \frac{\mathrm{T}}{\mathrm{P}} x100\%$$

where D is the duty cycle, T is the time the signal is active, and P is the total period of the signal. Thus, a 60% duty cycle means the signal is on 60% of the time but off 40% of the time. The "on time" for a 60% duty cycle could be a fraction of a second, a day, or even a week, depending on the length of the period.



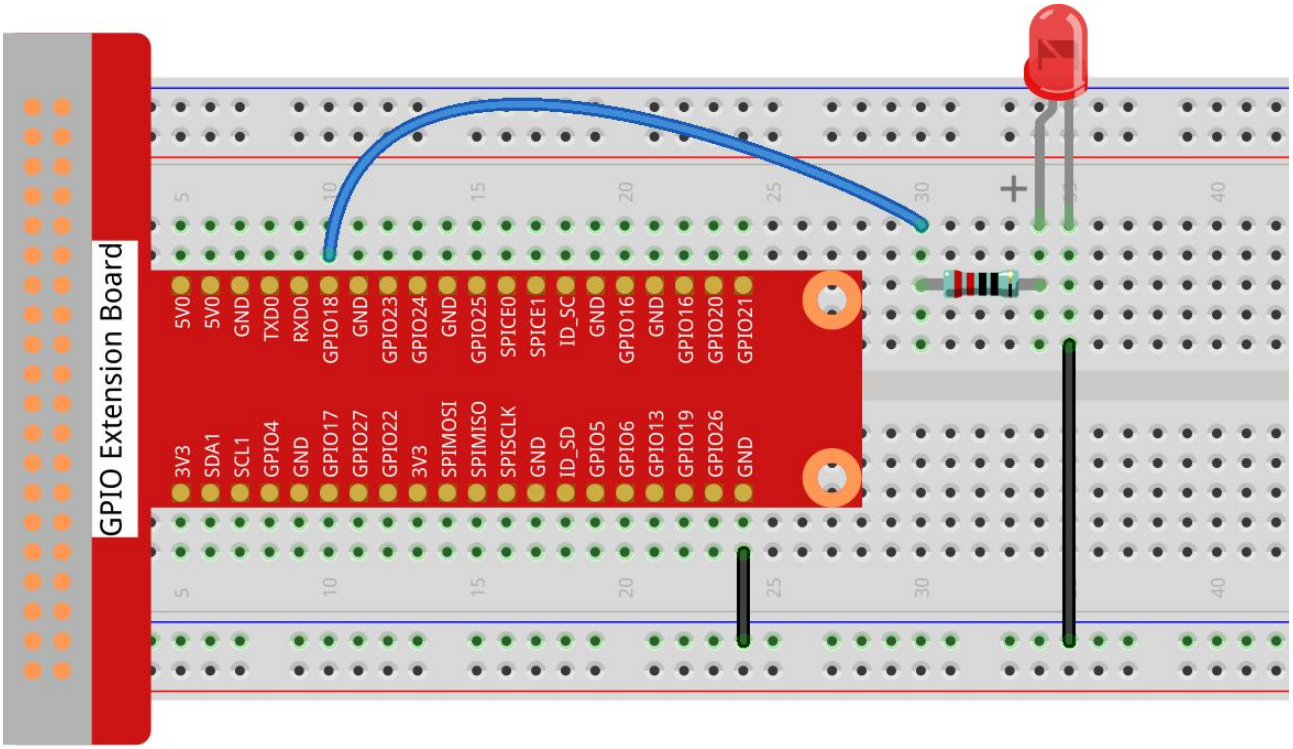
In this experiment, we use this technology to make the LED brighten and dim slowly so it looks like our breath.

Schematic Diagram



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/04_PwmLed
```

Step 3: Compile.

```
gcc PwmLed.c -o PwmLed -lwiringPi -lpthread
```

Step 4: Run.

```
sudo ./PwmLed
```

For Python Users:

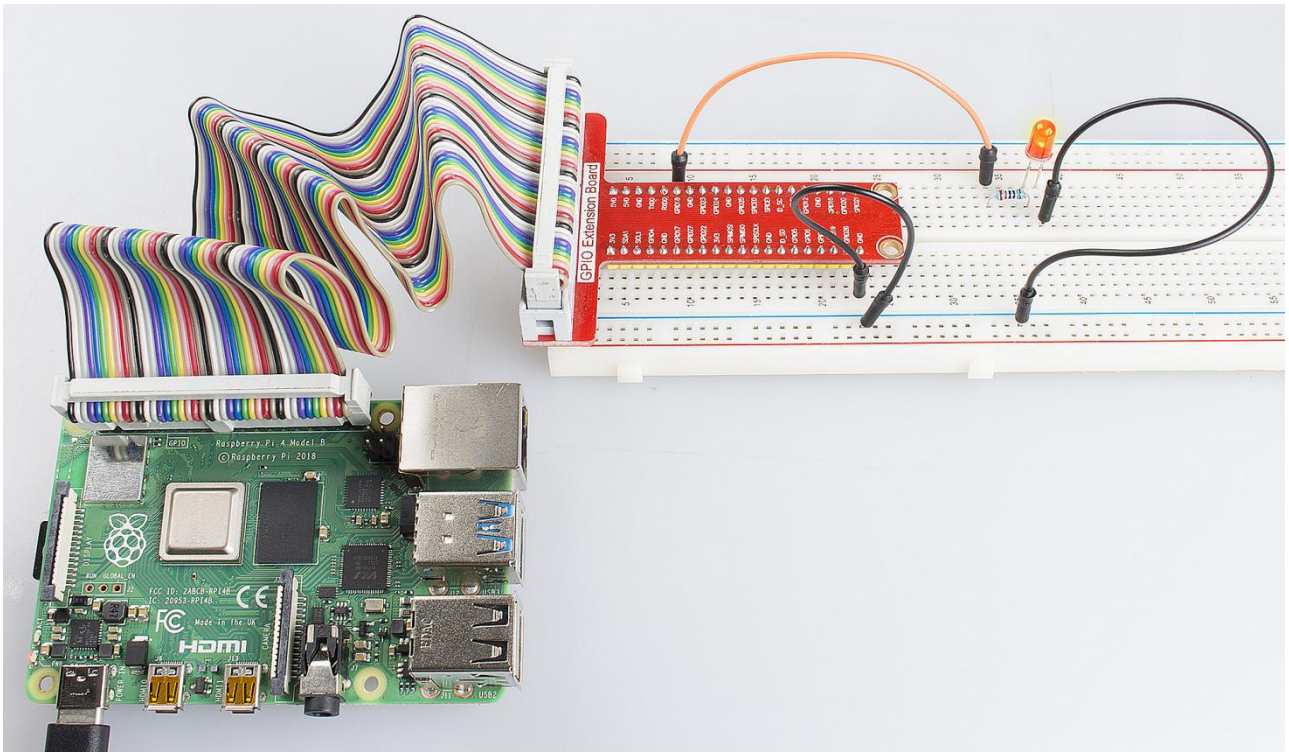
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 04_pwmLed.py
```

Now you will see the gradual change of the LED luminance, between bright and dim.



Summary

Through this experiment, you should have mastered the principle of PWM and how to program Raspberry Pi with PWM. You can try to apply this technology to DC motor speed regulation later.

Lesson 5 RGB LED

Introduction

Previously we've used the PWM technology to control an LED brighten and dim. In this lesson, we will use it to control an RGB LED to flash various kinds of colors.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * RGB LED

- 3 * Resistor (220Ω)

- Several jumper wires

Principle

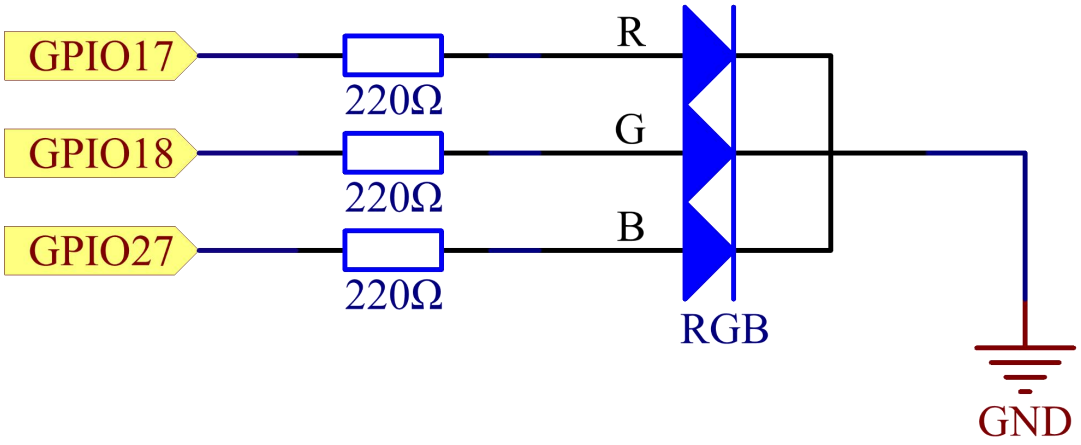
RGB

RGB LEDs emit light in various colors. RGB stands for the red, green, and blue color channels and is an industry color standard. They package three LEDs of red, green, and blue into a transparent or semitransparent plastic shell and have four pins. An RGB LED can display various new colors by changing the three channels and superimposing them, which, according to statistics, can create 16,777,216 different colors.

The three primary colors can be mixed into various colors by brightness. The brightness of LED can be adjusted with PWM. Raspberry Pi has only one channel for hardware PWM output, but it needs three channels to control the RGB LED, which means it is difficult to control the RGB LED with the hardware PWM of Raspberry Pi. Fortunately, the *softPwm* library simulates PWM (softPwm) by programming. You only need to include the header file *softPwm.h* (for C language users), and then call the API it provides to easily control the RGB LED by multi-channel PWM output, so as to display all kinds of color.

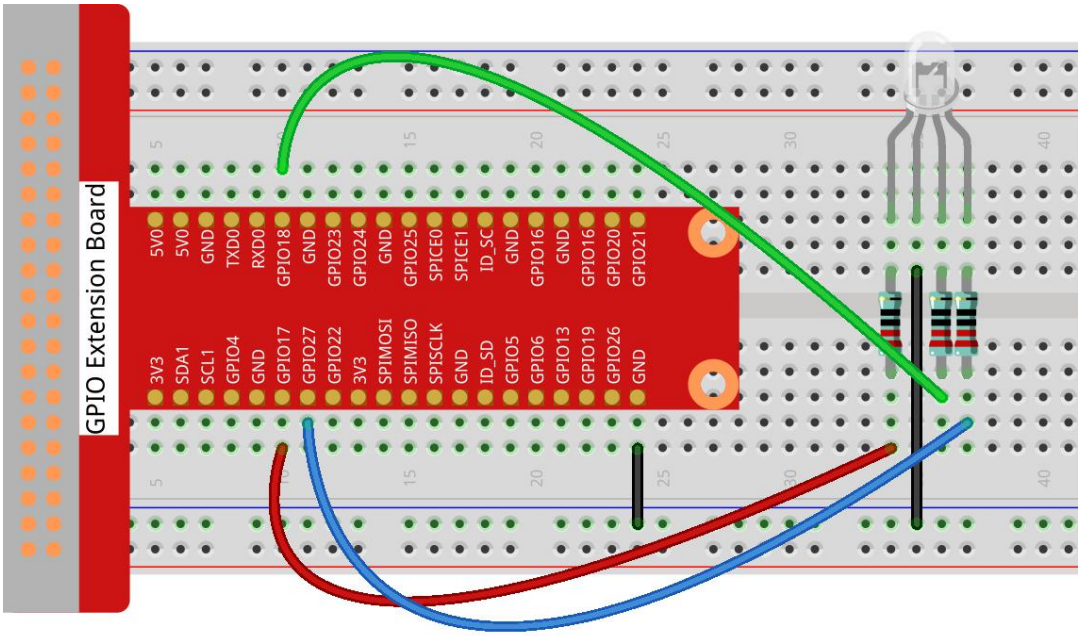
RGB LEDs can be categorized into common anode and common cathode ones. In this experiment, the latter is used.

Schematic Diagram



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/05_RGB/
```

Step 3: Compile.

```
gcc rgb.c -o rgb -lwiringPi -lpthread
```

Step 4: Run.

```
sudo ./rgb
```

For Python Users:

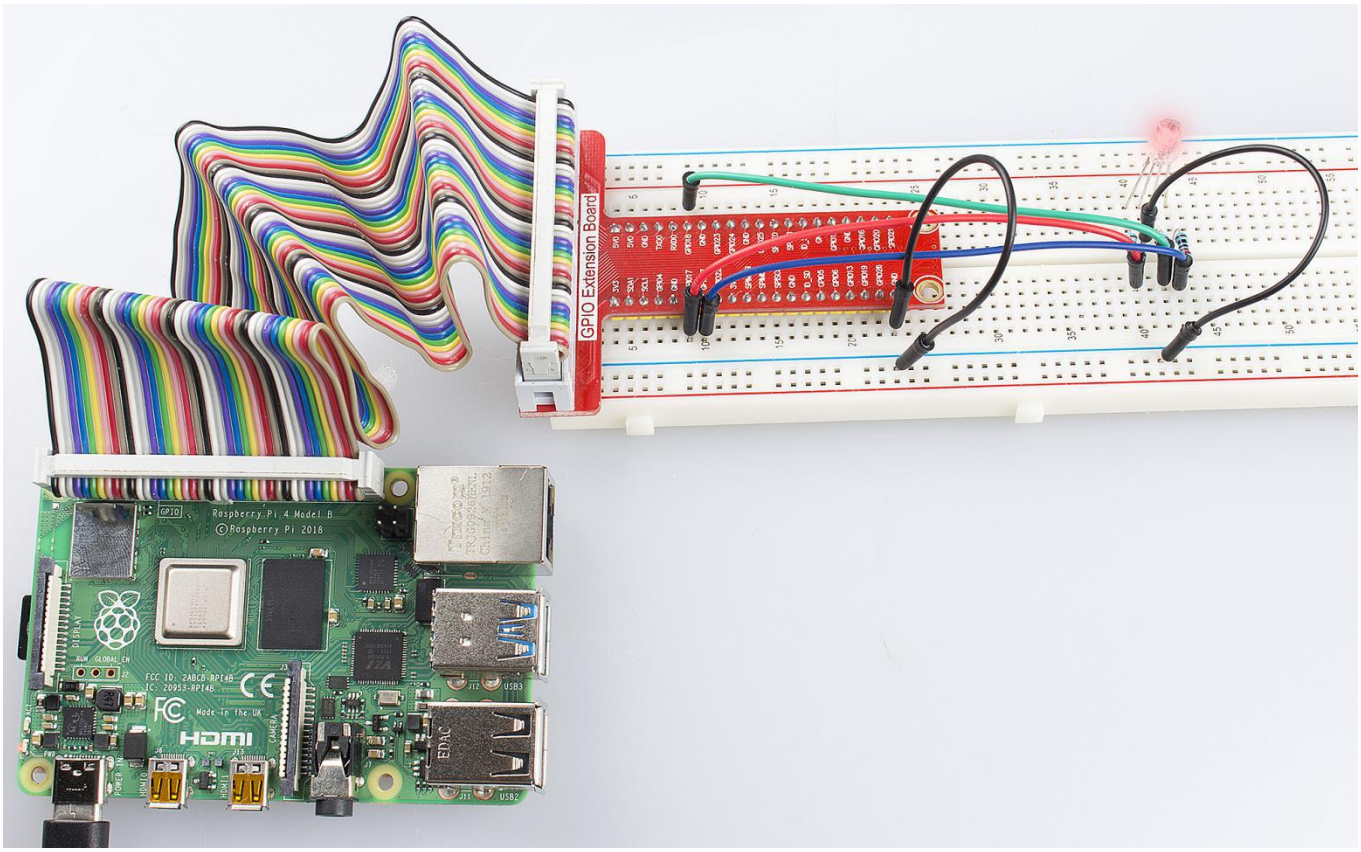
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 05_rgb.py
```

Here you should see the RGB LED flash different colors in turn.



Further Exploration

You can modify the parameters of the function *ledColorSet( )* by yourself, and then compile and run the code to see the color changes of the RGB LED.

Experimental Summary

In this experiment, you have learnt how to control RGB LEDs with the softPwm of Raspberry Pi in this experiment. Try to apply the softPwm to DC motor speed regulation.

Lesson 6 Buzzer

Introduction

In this lesson, we will learn how to drive an active buzzer to beep with a PNP transistor.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * Buzzer (Active)

- 1 * PNP transistor (8550)

- 1 * Resistor (1KΩ)

- Jumper wires

Principle

As a type of electronic buzzer with integrated structure, buzzers, which are supplied by DC power, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive ones (see the following picture). Turn the pins of two buzzers face up, and the one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.

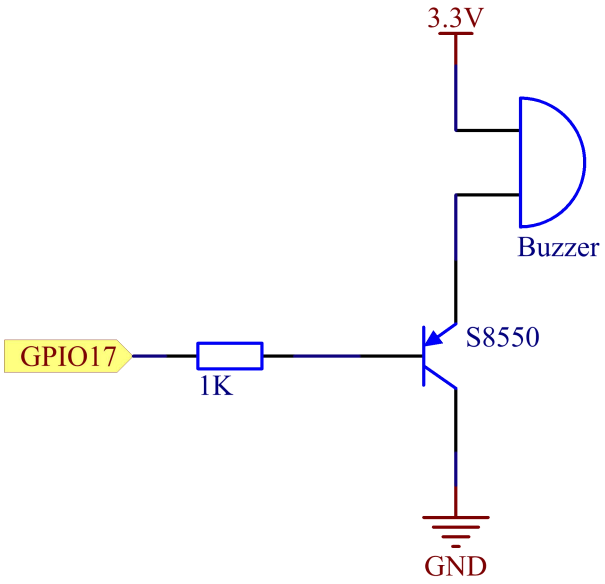
The difference between an active buzzer and a passive buzzer is:

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.

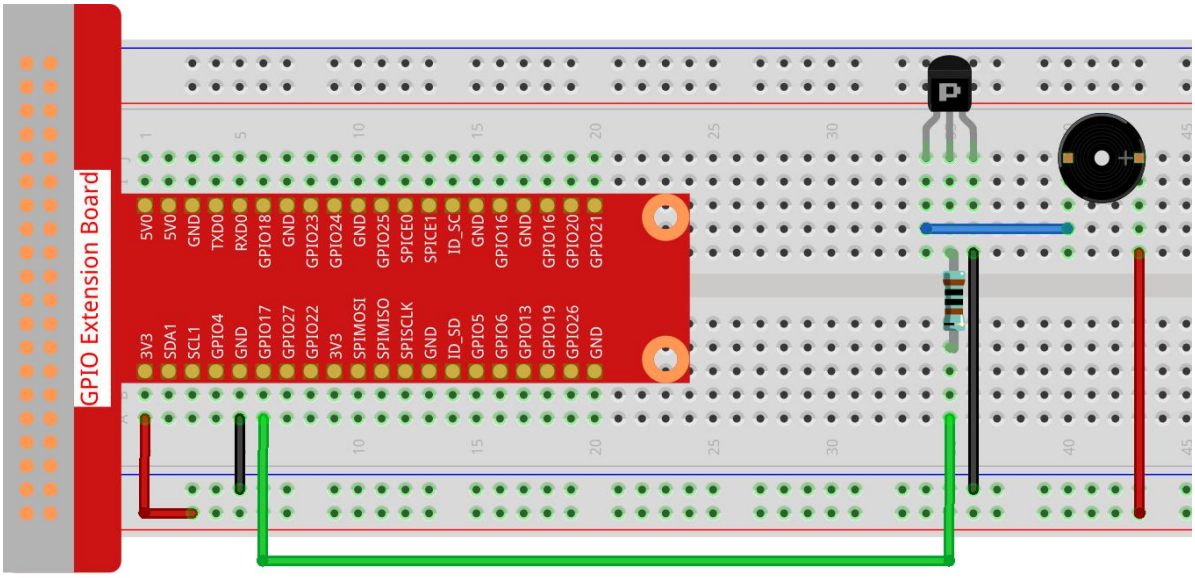
Schematic Diagram

In this experiment, an active buzzer is used. When the GPIO of Raspberry Pi output is supplied with low level (0V) by programming, the transistor will conduct because of current saturation and the buzzer will make sounds. But when high level is supplied to the IO of Raspberry Pi, the transistor will be cut off and the buzzer will not make sounds.



Experimental Procedures

Step 1: Build the circuit (Pay attention to the positive and negative poles of the buzzer)



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/06_Beep/
```

Step 3: Compile.

```
gcc beep.c -o beep -lwiringPi
```

Step 4: Run.

```
sudo ./beep
```

For Python Users:

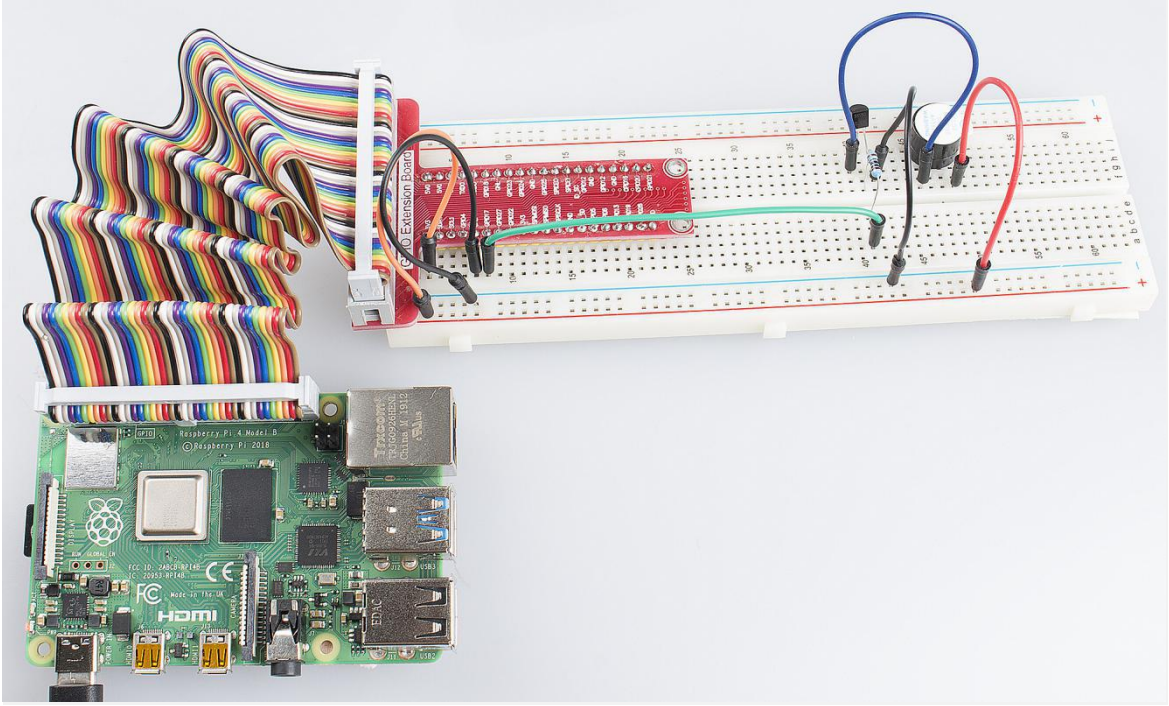
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 06_beep.py
```

Now, you should hear the buzzer make sounds.



Further Exploration

If you have a passive buzzer in hand, you can replace the active buzzer with it. Now you can make a buzzer sound like "do re mi fa so la si do" with just some basic knowledge of programming. Give a try!

Lesson 7 How to Drive a DC Motor

Introduction

In this lesson, we will learn to how to use L293D to drive a DC motor and make it rotate clockwise and counterclockwise.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * L293D

- 1 * DC motor

- 1 * Power Module

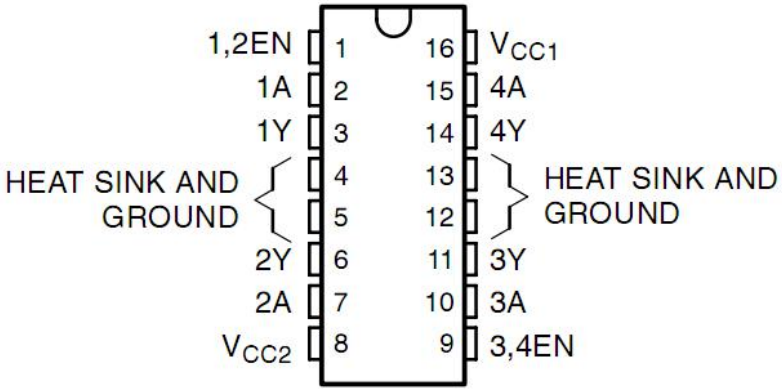
- Jumper wires

Principle

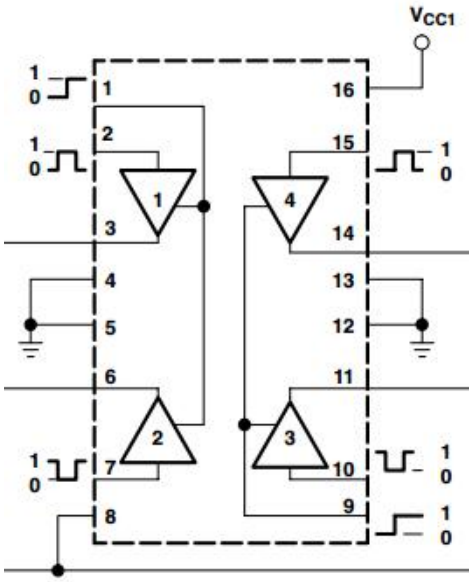
L293D

L293D is a 4-channel motor driver integrated by chip with high voltage and high current. It's designed to connect to standard DTL, TTL logic level, and drive inductive loads (such as relay coils, DC, Stepper Motors) and power switching transistors etc. DC Motors are devices that turn DC electrical energy into mechanical energy. They are widely used in electrical drive for their superior speed regulation performance.

See the figure of pins below. L293D has two pins (Vcc1 and Vcc2) for power supply. Vcc2 is used to supply power for the motor, while Vcc1 to supply for the chip.



The following is the internal structure of L293D. Pin EN is an enable pin and only works with high level; A stands for input and Y for output. You can see the relationship among them at the right bottom. When pin EN is High level, if A is High, Y outputs high level; if A is Low, Y outputs Low level. When pin EN is Low level, the L293D does not work.



INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

DC Motor

This is a 5V DC motor. It will rotate when you give the two terminals of the copper sheet one high and one low level. For convenience, you can weld the pins to it.

Size: 25*20*15MM

Operation Voltage: 1-6V

Free-run current (3V): 70m

A Free-run speed (3V): 13000RPM

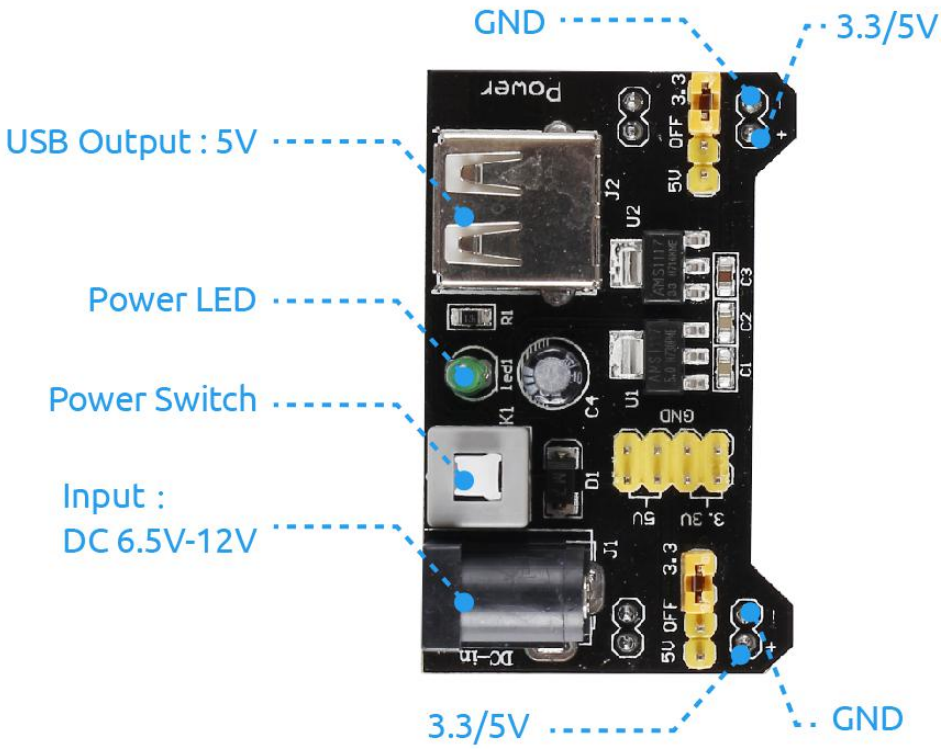
Stall current (3V): 800mA

Shaft diameter: 2mm

Power Supply Module

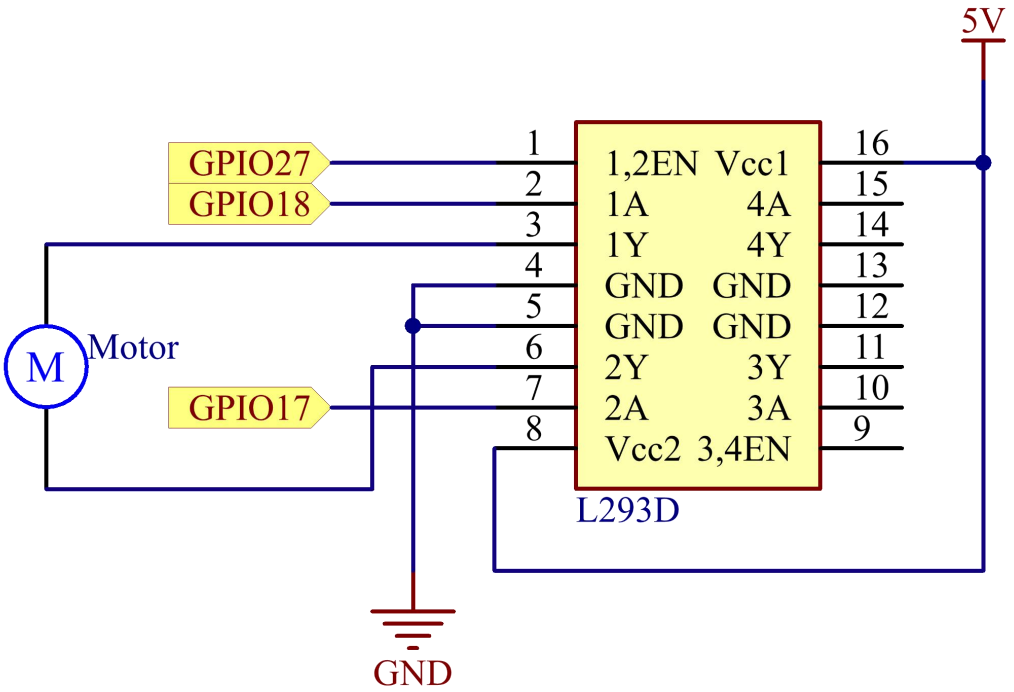
In this experiment, it needs large currents to drive the motor especially when it starts and stops, which will severely interfere with the normal work of Raspberry Pi. Therefore, we separately supply power for the motor by this module to make it run safely and steadily.

You can just plug it in the breadboard to supply power. It provides a voltage of 3.3V and 5V, and you can connect either via a jumper cap included.



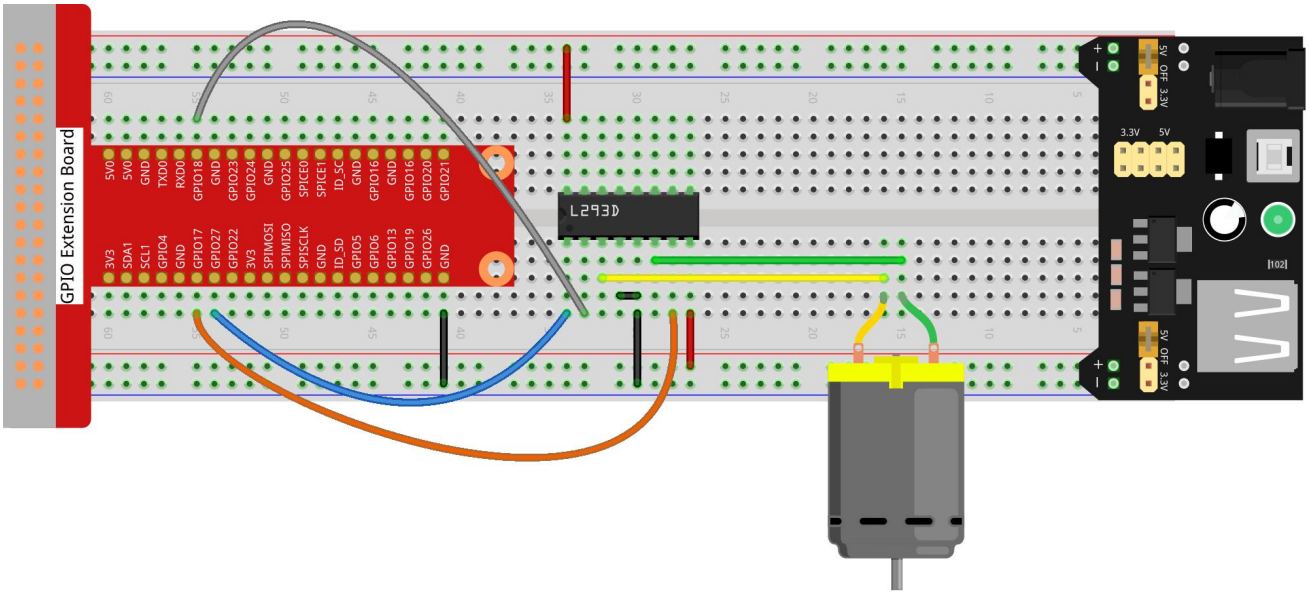
Schematic Diagram

Plug the power supply module in breadboard, and insert the jumper cap to pin of 5V, then it will output voltage of 5V. Connect pin 1 of L293D to GPIO22, and set it as high level. Connect pin2 to GPIO27, and pin7 to GPIO17, then set one pin high, while the other low. Thus you can change the motor's rotation direction.



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/07_Motor/
```

Step 3: Compile.

```
gcc motor.c -o motor -lwiringPi
```

Step 4: Run.

```
sudo ./motor
```

For Python Users:

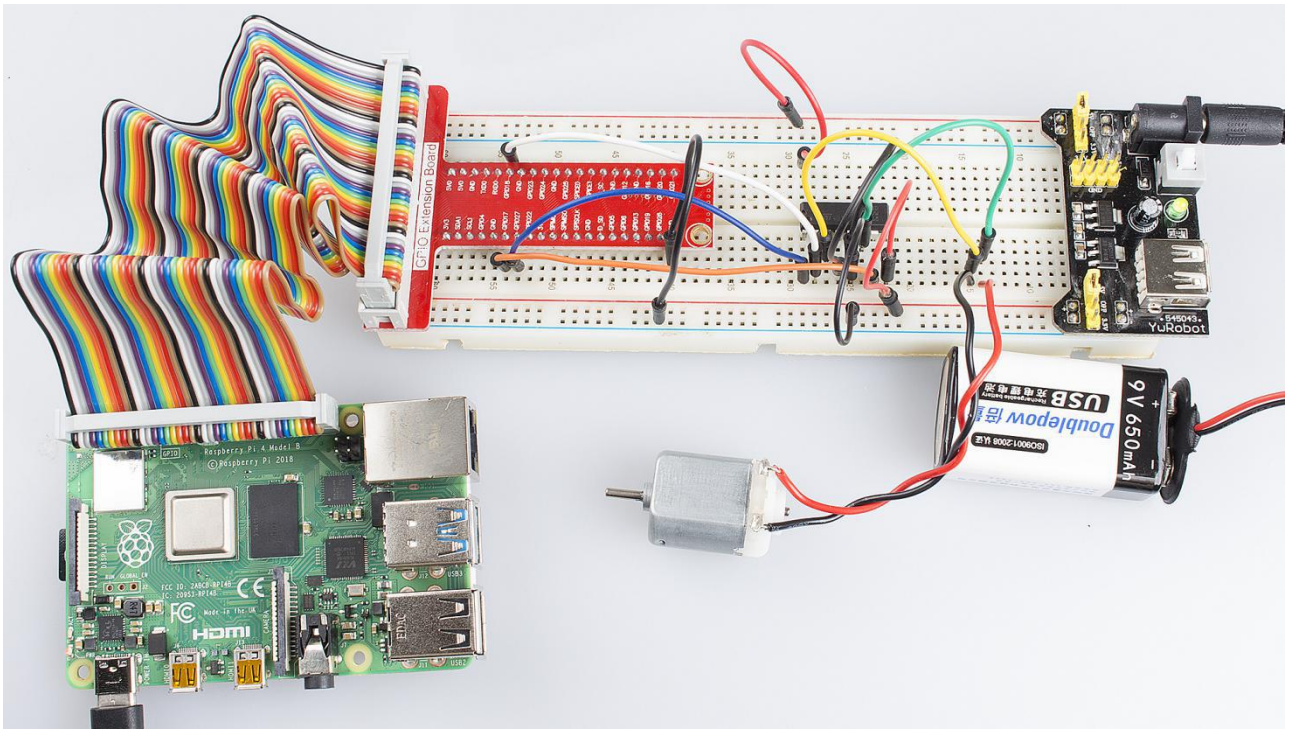
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 07_motor.py
```

Now, you should see the motor blade rotating.



Further Exploration

You can use buttons to control the clockwise and counterclockwise rotation of the motor blade based on the previous lessons. Also you can apply the PWM technology to control the rotation.

Summary

Through this lesson, you have learnt the relative principle and driving mode of DC motors, as well as how to drive a motor by Raspberry Pi. You should also pay special attention to the fact that a DC motor will greatly interfere with the whole circuit when it works, so you need to adopt photoelectric isolation and provide separate power supply. A freewheeling diode is also necessary for the whole system to work reliably and steadily.

Lesson 8 Rotary Encoder

Introduction

A rotary encoder is a type of electro-mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital code. In this lesson, we will learn how to use this device.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * Rotary Encoder Module

- Jumper wires

Principle

A rotary encoder is an electronic switch with a set of regular pulses with strictly timing sequence. When used with IC, it can achieve increment, decrement, page turning and other operations such as mouse scrolling, menu selection, acoustic sound regulation, frequency regulation, toaster temperature regulation, and so on.

There are mainly two types of rotary encoders: absolute and incremental (relative) encoders. The output of absolute encoders indicates the current position of the shaft, making them angle transducers. The output of incremental encoders provides information about the motion of the shaft, which is typically further processed elsewhere into information such as speed, distance, and position.

Most rotary encoders have 5 pins with three functions of turning left, turning right and pressing down:

Pin 4 & 5: switching wiring terminals for pressing down (no different from the buttons mentioned previously, so no more details will be provided here.)

Pin 2: generally connected to ground.

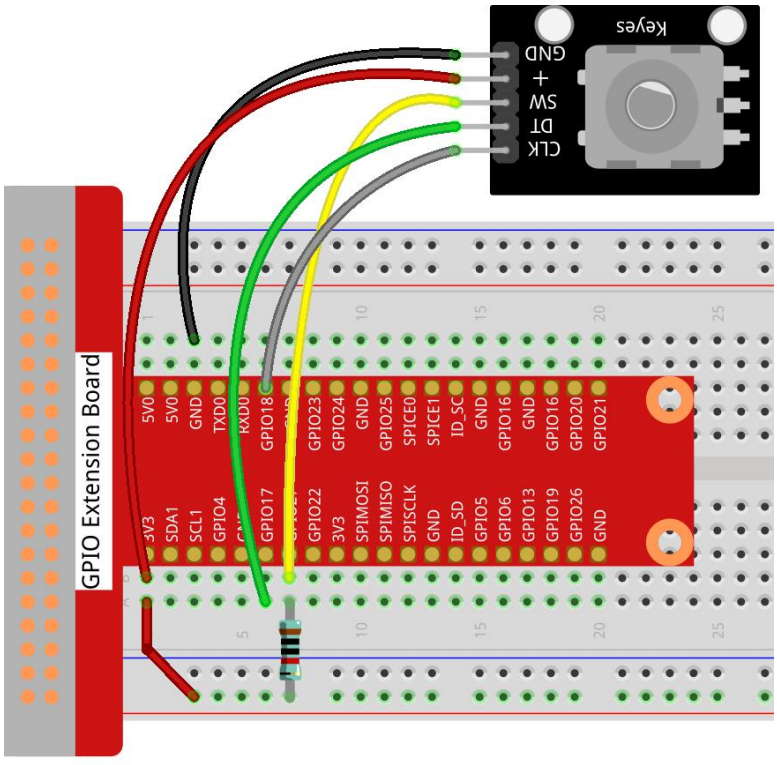
Pin 1 & 3: first connected to a pull-up resistor and then to a microprocessor (in this experiment, to GPIO0 and GPIO1 of Raspberry Pi); when you spin the knob of the encoder clockwise and counterclockwise, there will be pulse outputs in pin 1 and pin 3.

If both GPIO0 and GPIO1 are at high level, the switch rotates clockwise; if GPIO0 is at high level but GPIO1 is low, the switch rotates counterclockwise. Therefore, when programming, you only need to check the state of pin 3 when pin 1 is at high level, and then you can tell whether the switch rotates clockwise or counterclockwise.

Experimental Procedures

Step 1: Build the circuit.

Raspberry Pi	Rotary Encoder
3.3V	+
GND	GND
GPIO17	DT
GPIO18	CLK
GPIO27	SW



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/08_RotaryEncoder/
```

Step 3: Compile.

```
gcc rotaryEncoder.c -o rotaryEncoder -lwiringPi
```

Step 4: Run.

```
sudo ./rotaryEncoder
```

For Python Users:

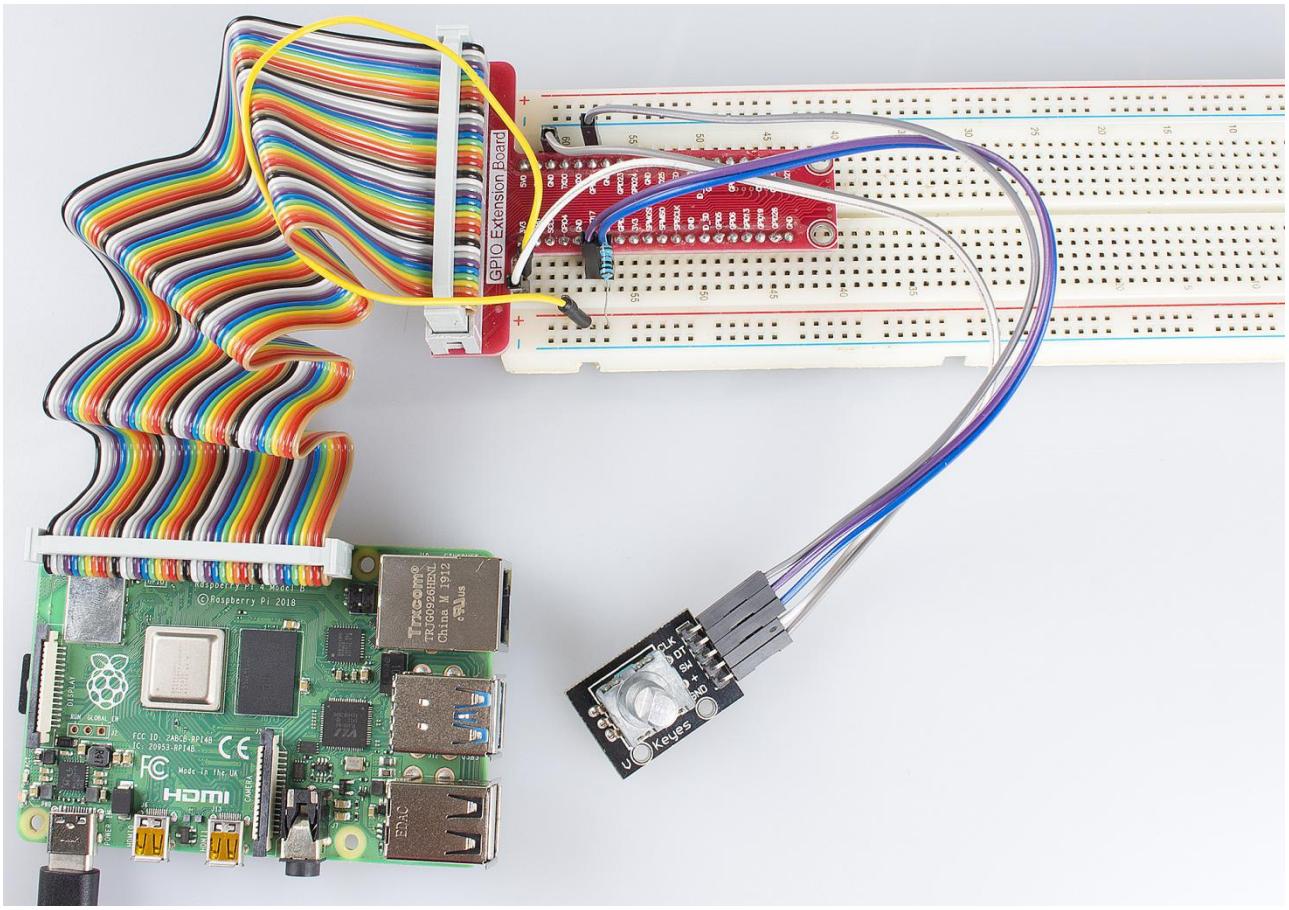
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 08_rotaryEncoder.py
```

Now, gently rotate the encoder to change the value of the variable in the above program, and you will see the value printed on the screen. Rotate the encoder clockwise, the value will increase; or rotate it counterclockwise, the value will decrease.



Further Exploration

In this experiment, the pressing down function of rotary encoder is not involved. Try to explore this function by yourself!

Lesson 9 555 Timer

Introduction

The NE555 Timer, a mixed circuit composed of analog and digital circuits, integrates analog and logical functions into an independent IC, thus tremendously expanding the applications of analog integrated circuits. It is widely used in various timers, pulse generators, and oscillators. In this lesson, we will learn how to use the 555 timer.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * NE555

- 3 * Resistor (1 * 1KΩ, 2 * 10KΩ)

- 2 * Capacitor (100nF)

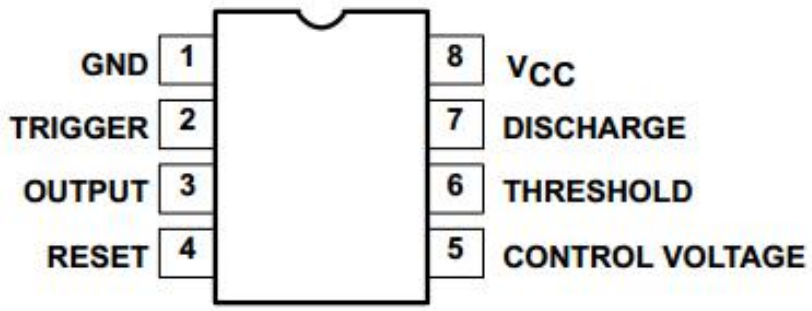
- Jumper wires

Principle

555 IC

A 555 timer is a medium-sized IC device which combines analog and digital functions. With low cost and reliable performance, it just attaches external resistors and capacitors so as to achieve the functions of multivibrator, monostable trigger, Schmitt trigger and other circuits which can generate and transform pulses. It is also used frequently as a timer and widely applied to instruments, household appliances, electronic measurement, automatic control and other fields.

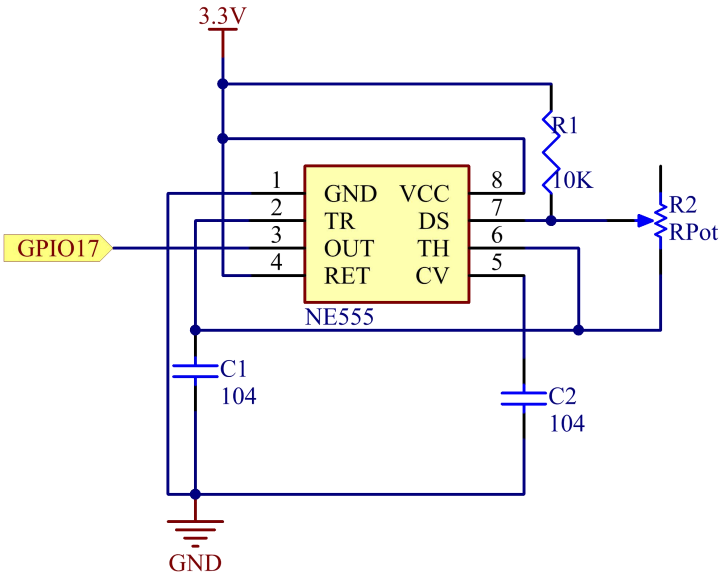
Its pins and their functions:



As shown in the picture, the pins are set dual in-line with the 8-pin package.

- Pin 1 (**GND**): the ground
- Pin 2 (**TRIGGER**): the input of lower comparator
- Pin 3 (**OUTPUT)**: having two states of 0 and 1 decided by the input electrical level
- Pin 4 (**RESET**): outputting low level when supplied a low one
- Pin 5 (**CONTROL VOLTAGE**): changing the upper and lower level trigger values
- Pin 6 (**THRESHOLD)**: the input of the upper comparator
- Pin 7 (**DISCHARGE)**: having two states of suspension and ground connection also decided by the input, and the output of the internal discharge tube
- Pin 8 (**VCC**): the power supply

The 555 timer can work under three modes. In this experiment, the astable mode is used to generate square waves.



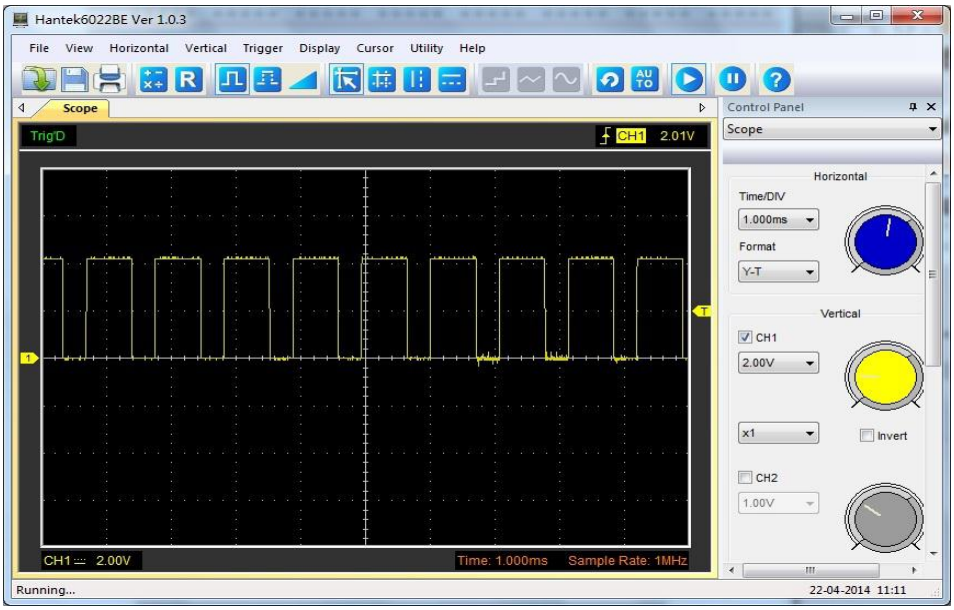
Under the astable mode, the frequency of output waveform of the 555 timer is determined by R_1, R_2 and C_2:

$$f = \frac{1}{\ln 2 * C_2 * (R_1 + 2R_2)}$$

In the above circuit, R_1= R_2=10K Ω = 10^4 Ω; C_2 =100nF= 10^{-7} F, so we can get the frequency:

$$f = \frac{1}{\ln 2 * \ 10^{-7} * \ (10^4 + 2 * \ 10^4)} \approx 481\text{Hz}$$

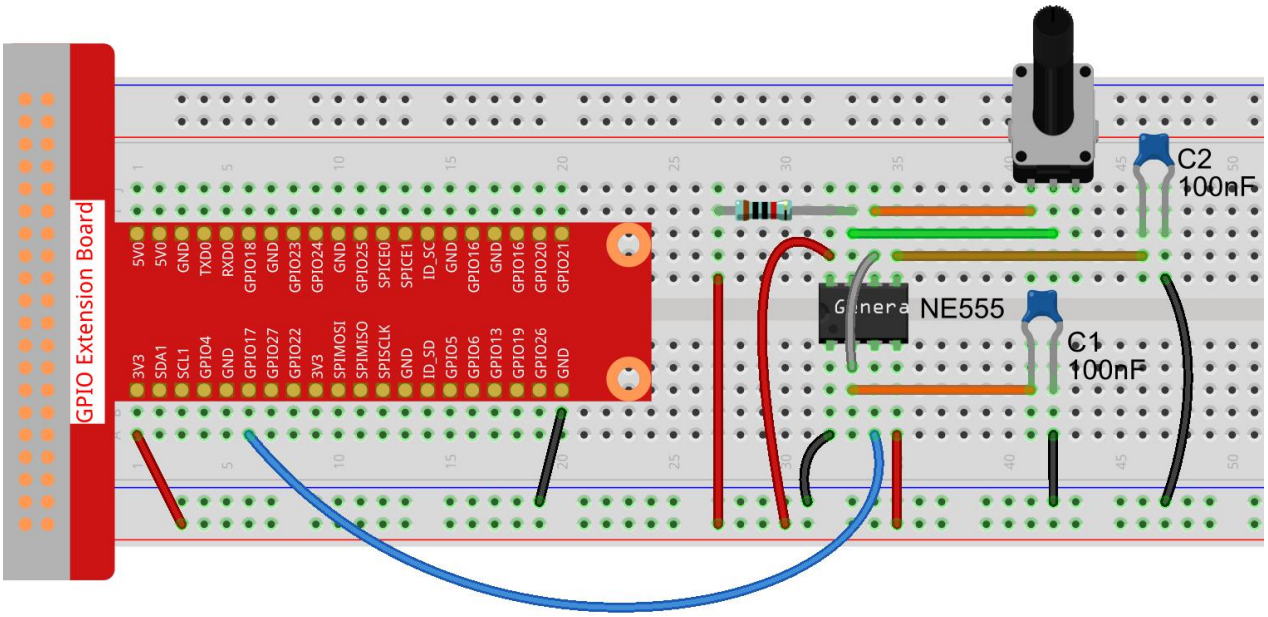
After connecting the circuit according to the schematic diagram, use an oscilloscope to observe the frequency of the output waveform. We can see it is consistent with the above calculated result.



Attach the output pin (e.g. pin 3) of the 555 timer to GPIO17 of the Raspberry Pi, configure GPIO17 as the mode of the rising edge interrupt by programming, and then detect the square wave pulses generated by the 555 timer with interrupt. The work of Interrupt Service Routine (ISR) is to add 1 to a variable.

Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/09_Timer555/
```

Step 3: Compile.

```
gcc timer555.c -o timer555 -lwiringPi
```

Step 4: Run.

```
sudo ./timer555
```

For Python Users:

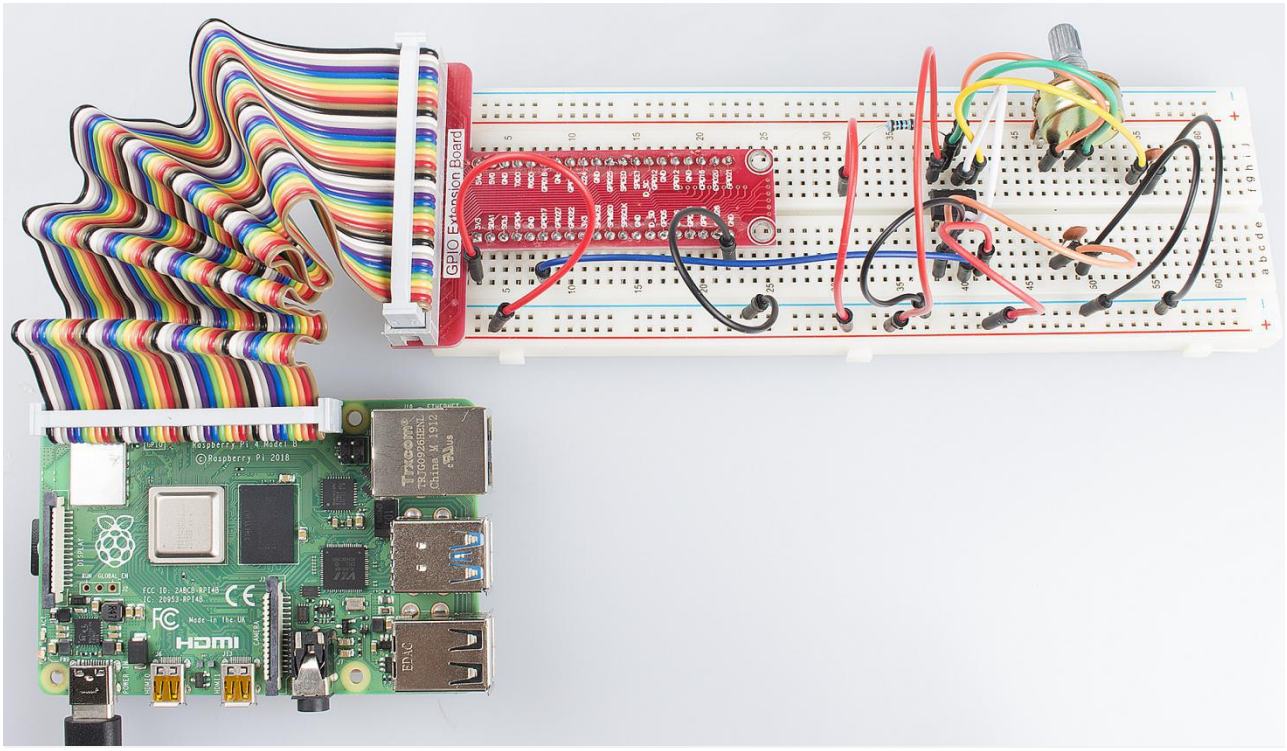
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 09_timer555.py
```

Now, you should see data printed on the display, which are square waves generated by the 555 timer. The program counts pulses by interrupt as we have learned previously.



Lesson 10 Driving LEDs by 74HC595

Introduction

In this lesson, we will learn how to use 74HC595 to make eight LEDs blink regularly.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * 74HC595

- 8 * LED

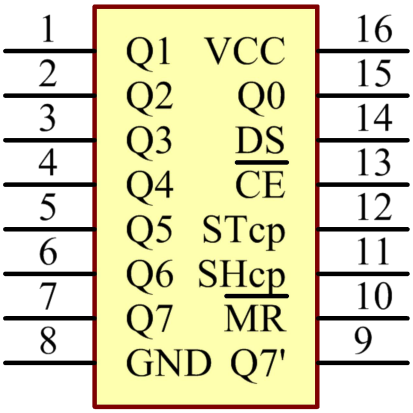
- 8 * Resistor (220Ω)

- Jumper wires

Principle

74HC595

The 74HC595 consists of an 8－bit shift register and a storage register with three－state parallel outputs. It converts serial input into parallel output so that you can save IO ports of an MCU. The 74HC595 is widely used to indicate multipath LEDs and drive multi-bit segment displays. "Three-state" mentioned above refers to the fact that you can set the output pins as either high, low or high impedance. With data latching, the instant output will not be affected during the shifting; with data output, you can cascade 74HC595s more easily. Compatible with low voltage TTL circuit, 74HC595 can transform serial input of 8-bit data into parallel output of 8-bit data. So it is often used to extend GPIO for embedded system and drive low power devices.



Pins of 74HC595 and their functions:

Q0-Q7: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display directly.

Q7': Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series

MR: Reset pin, active at low level; here it is directly connected to 5V.

SH_CP: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.

ST_CP: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.

OE: Output enable pin, active at low level, connected to GND.

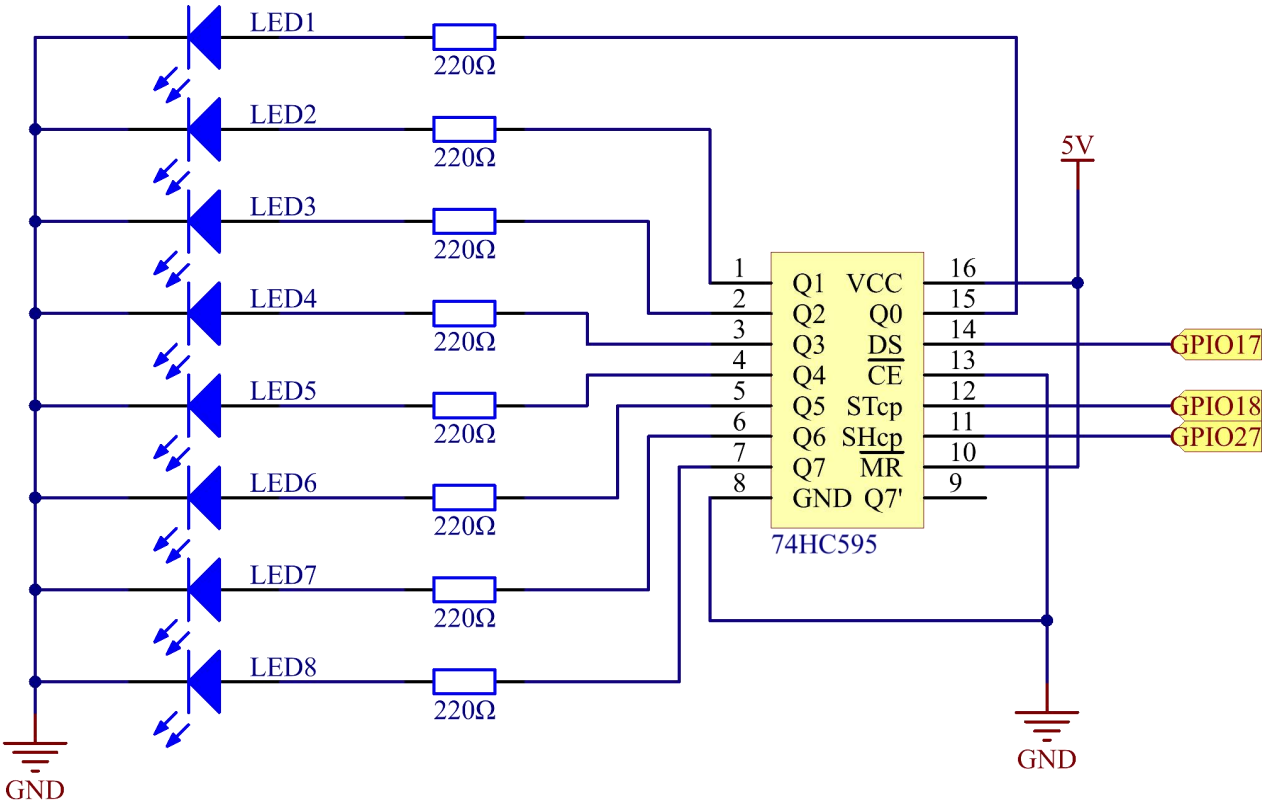
DS: Serial data input pin

VCC: Positive supply voltage

GND: Ground

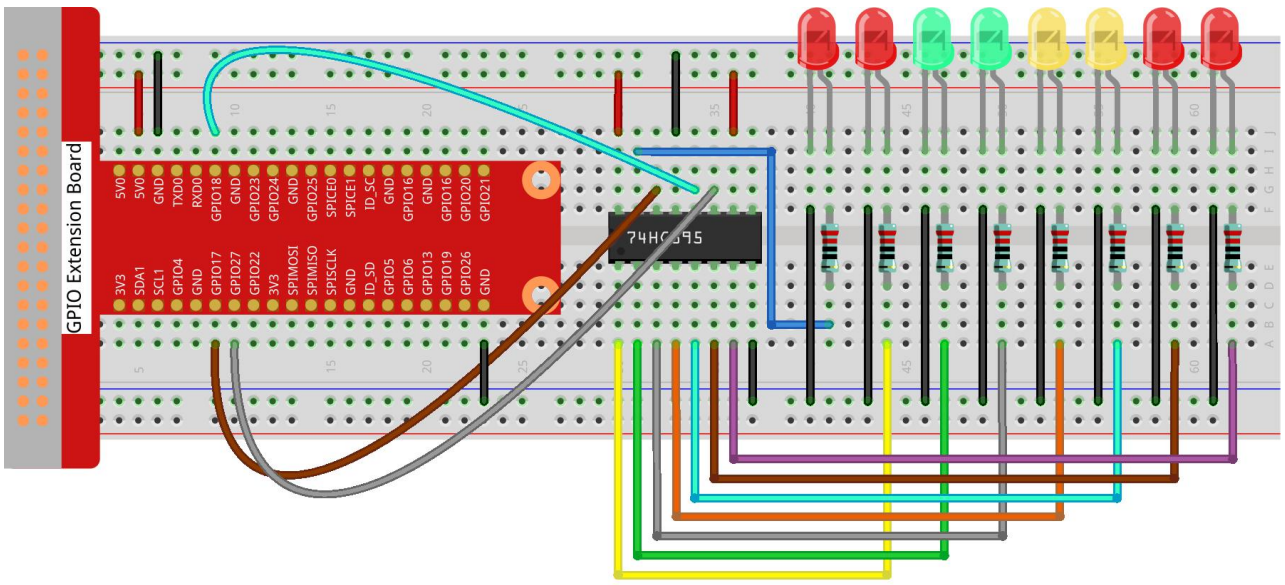
Schematic Diagram

In this experiment, connect ST_CP to Raspberry Pi GPIO18, SH_CP to GPIO27, and DS to GPIO17. Input data in DS pin to the shift register when SH_CP (the clock input of the shift register) is at the rising edge, and to the memory register when ST_CP (the clock input of the memory) is at the rising edge. Then you can control the states of SH_CP and ST_CP via Raspberry Pi GPIO to transform serial input data into parallel output data so as to save Raspberry Pi GPIOs.



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/10_74HC595_LED/
```

Step 3: Compile.

```
gcc 74HC595_LED.c -o 74HC595_LED -lwiringPi
```

Step 4: Run.

```
sudo ./74HC595_LED
```

For Python Users:

Step 2: Change directory.

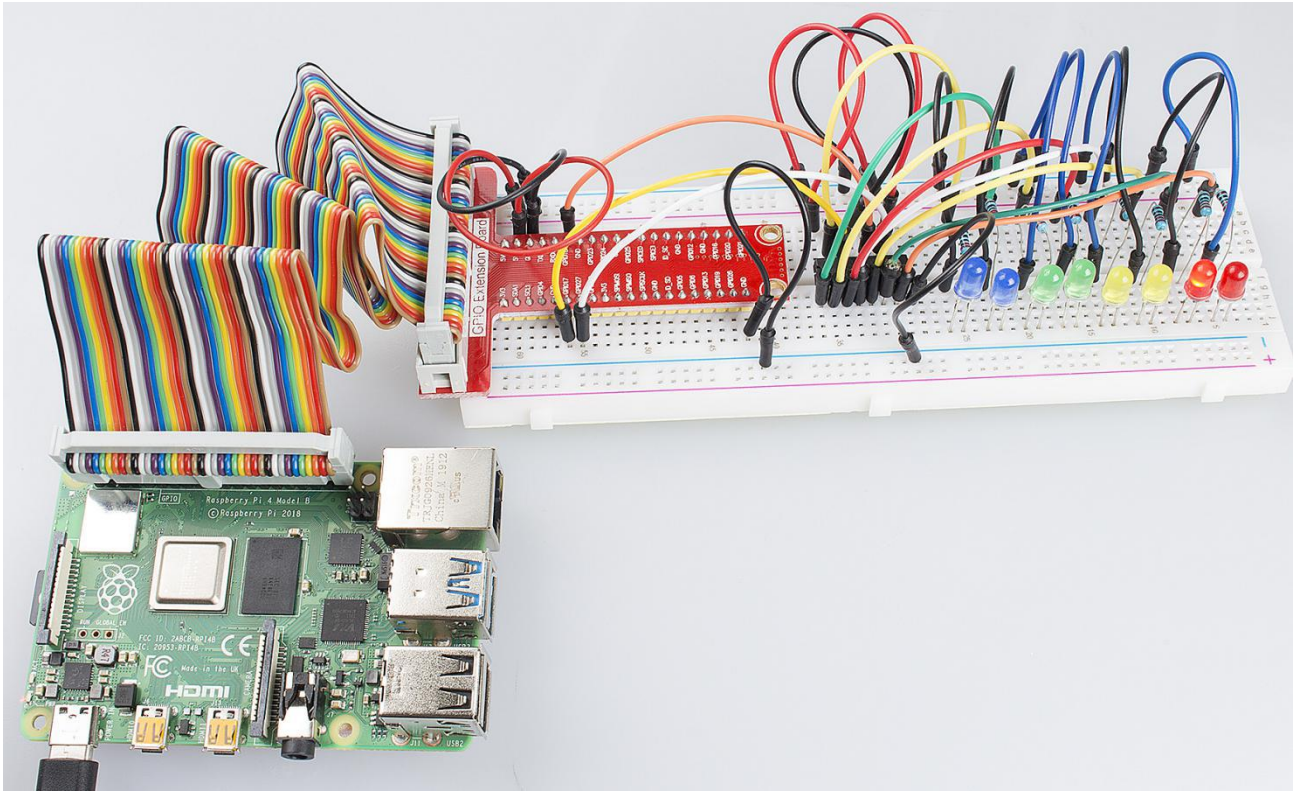
```
cd /home/pi/Sunfounder_SuperKit_Python_code_for_RaspberryPi/
```

Step 3: Run.

```
sudo python3 10_74HC595_LED.py
```

Here you should see eight LEDs blink regularly.

Further Exploration



In this experiment, three Raspberry Pi GPIOs are used to separately control 8 LEDs based on 74HC595. In fact, 74HC595 has another powerful function – cascade. With cascade, you can use a microprocessor to control more peripherals. We'll check more details later.

Lesson 11 Driving 7-Segment Display by 74HC595

Introduction

Since we've got some knowledge of the 74HC595 in the previous lesson, now let's try to use it and drive a 7-segment display to show a figure from 0 to 9.

Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * 74HC595
- 1 * 7-segment display
- 2 * Resistor (220KΩ,10K)
- 1 * Button
- Jumper wires

Principle

7-Segment Display

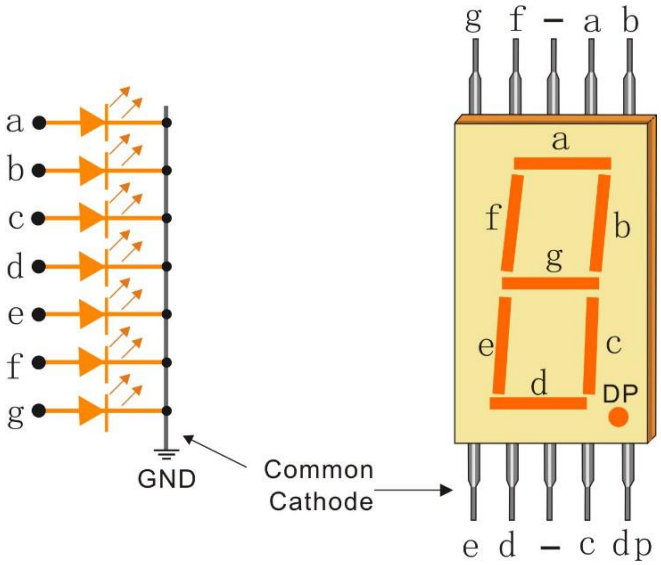
A 7-segment display is an 8-shaped component which packages 7 LEDs. Each LED is called a segment – when energized, one segment forms part of a numeral (both decimal and hexadecimal) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point (DP) when two or more 7-segment displays are connected together to display numbers greater than ten.

Each of the LEDs in the display is given a positional segment with one of its connection pins led out from the rectangular plastic package. These LED pins are labeled from "a" through to "g" representing each individual LED. The other LED pins are connected together forming a common pin. So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will brighten and others stay dim, thus showing the corresponding character on the display.

The common pin of the display generally tells its type. There are two types of pin connection: a pin of connected cathodes and one of connected anodes, indicating Common Cathode (CC) and Common Anode (CA). As the name suggests, a CC display has all the cathodes of the 7 LEDs connected when a CA display has all the anodes of the 7 segments connected.

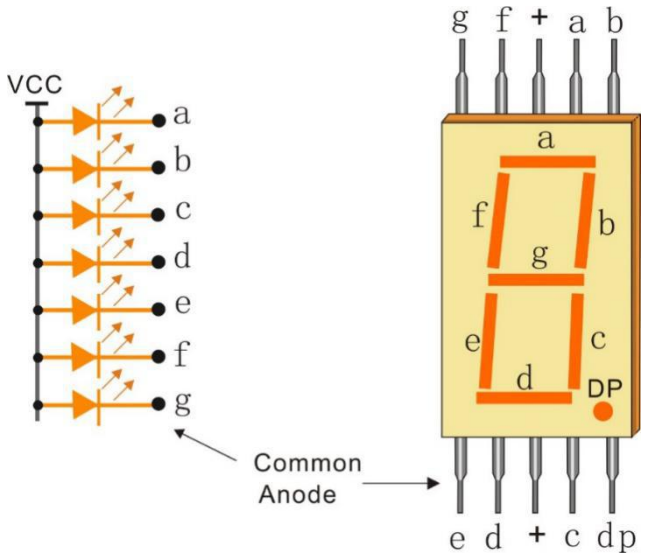
Common Cathode 7-Segment Display

In a common cathode display, the cathodes of all the LED segments are connected to the logic "0" or ground. Then an individual segment (a-g) is energized by a "HIGH", or logic "1" signal via a current limiting resistor to forward bias the anode of the segment.



Common Anode 7-Segment Display

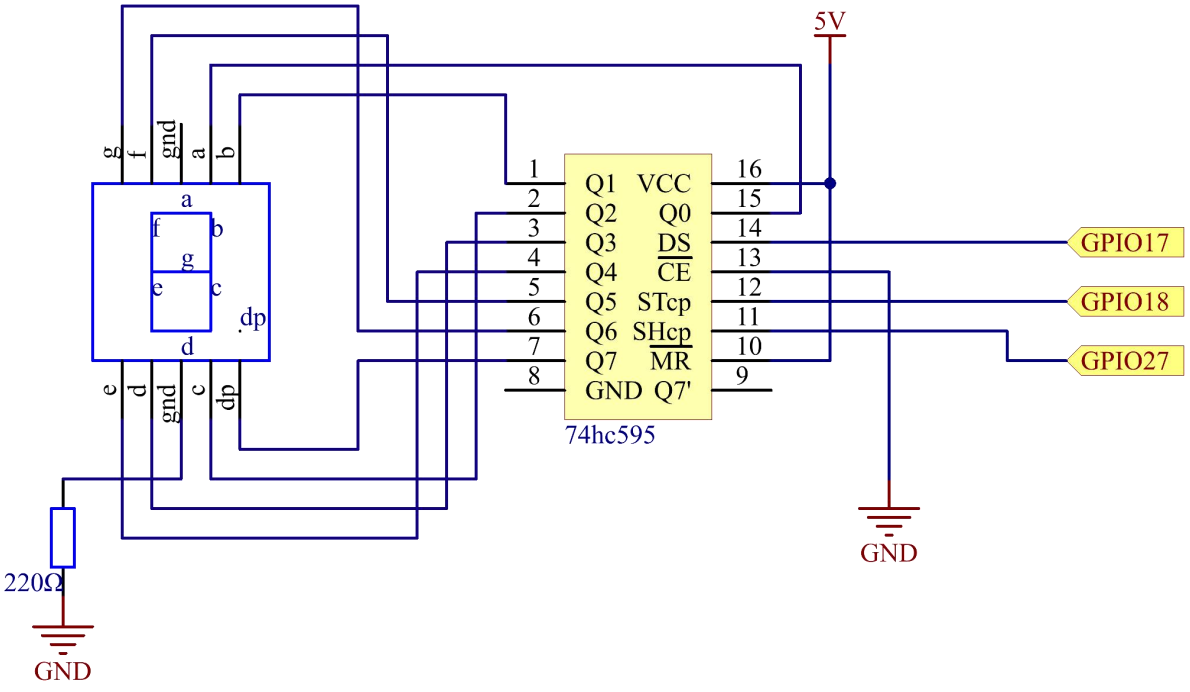
In a common anode display, the anodes of all the LED segments are connected to the logic "1". Then an individual segment (a-g) is energized by a ground, logic "0" or "LOW" signal via a current limiting resistor to the cathode of the segment.



In this experiment, a common cathode 7-segment display is use. It should be connected to ground. When the anode of an LED in a certain segment is at high level, the corresponding segment will light up; when it is at low, the segment will stay dim.

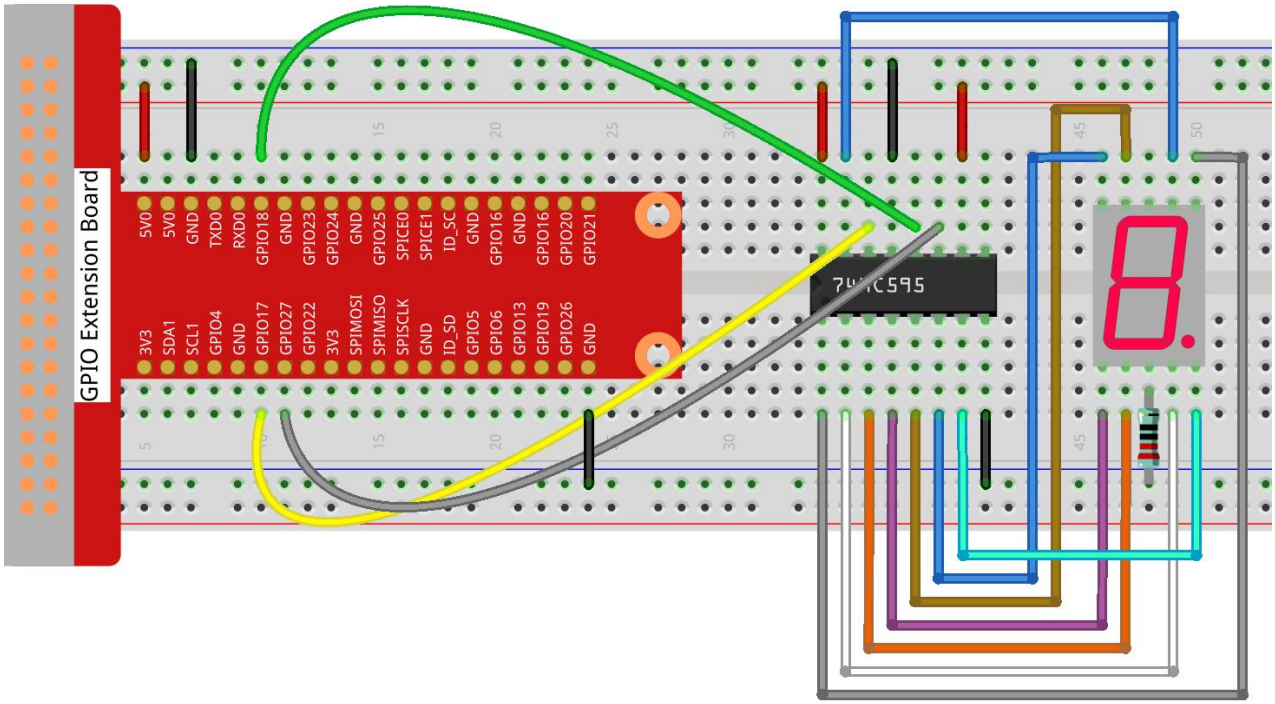
Schematic Diagram

Connect pin ST_CP of 74HC595 to Raspberry Pi GPIO18, SH_CP to GPIO27, DS to GPIO17, parallel output ports to 8 segments of the LED segment display. Input data in DS pin to shift register when SH_CP (the clock input of the shift register) is at the rising edge, and to the memory register when ST_CP (the clock input of the memory) is at the rising edge. Then you can control the states of SH_CP and ST_CP via the Raspberry Pi GPIOs to transform serial data input into parallel data output so as to save Raspberry Pi GPIOs and drive the display.



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/11_Segment/
```

Step 3: Compile.

```
gcc segment1.c -o segment1 -lwiringPi
```

Step 4: Run.

```
sudo ./segment1
```

For Python Users:

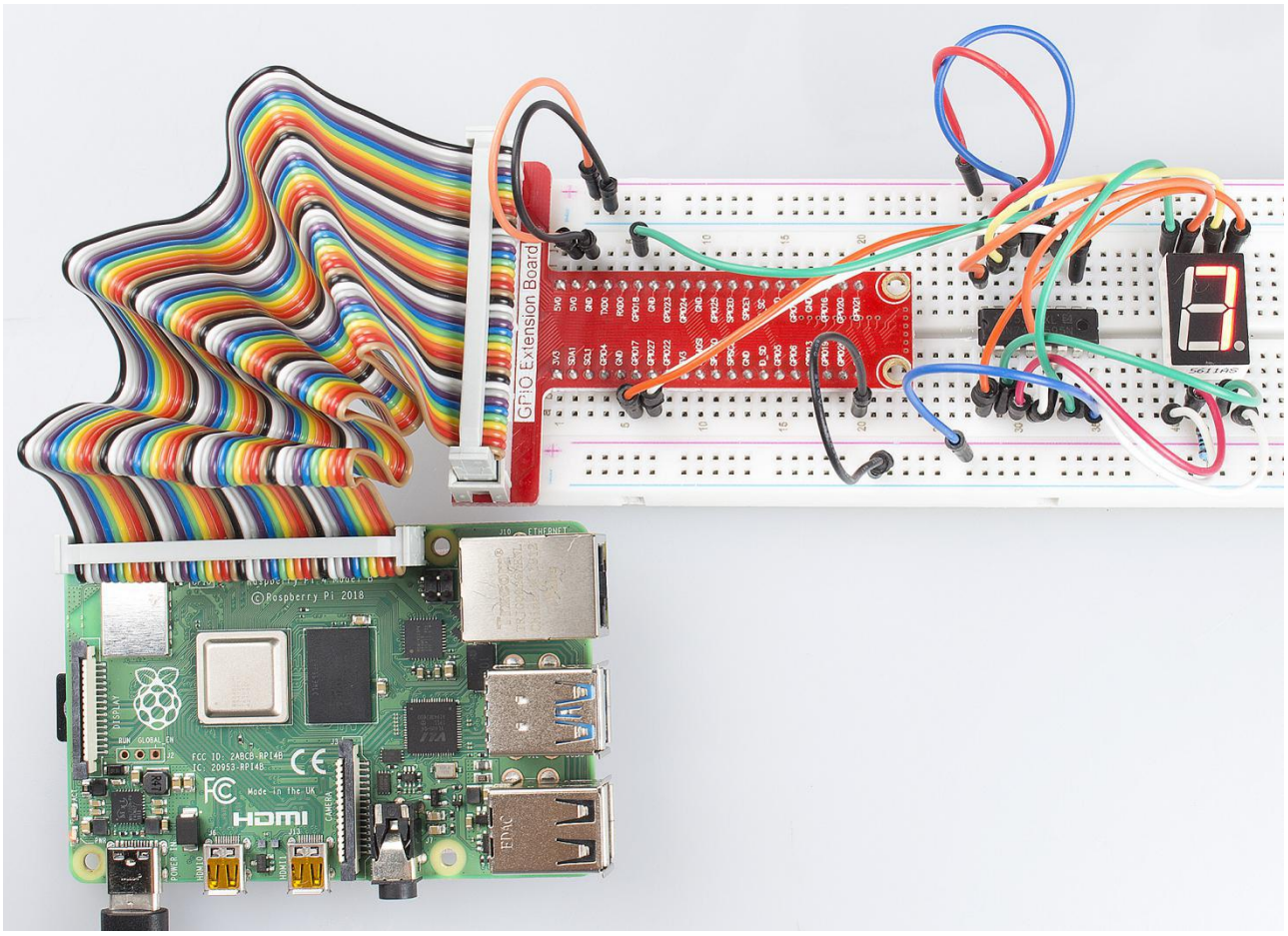
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 11_segment.py
```

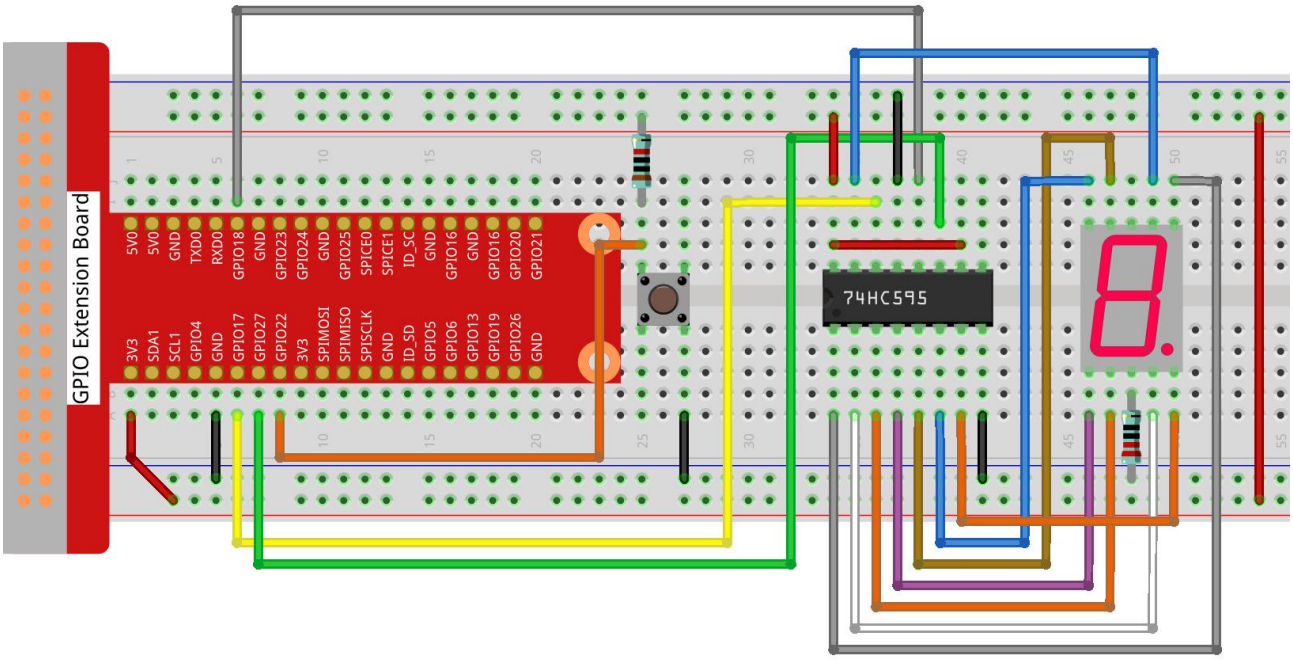
You should see the 7-segment display from 0 to 9, and A to F.



Further Exploration

You can slightly modify the hardware and software based on this experiment to make a dice. For hardware, add a button to the original board.

Build the circuit:



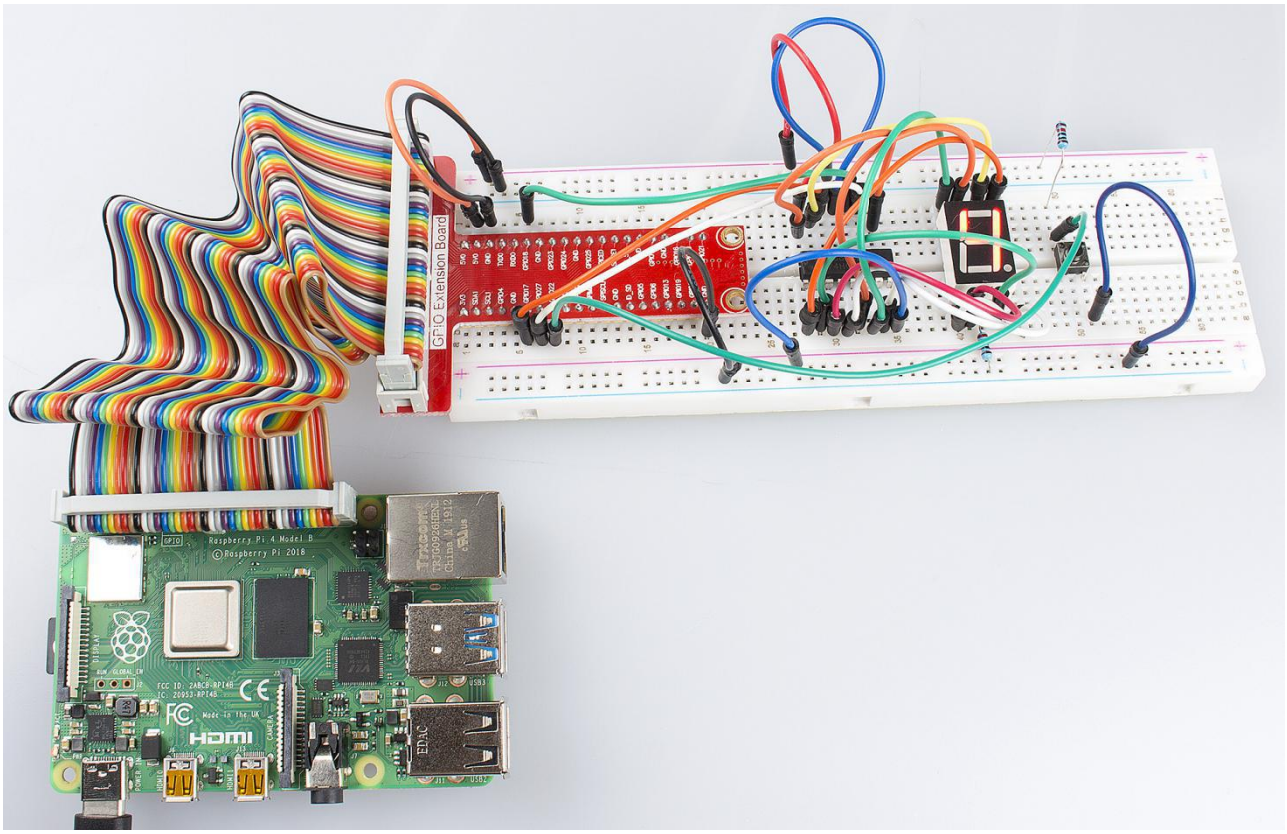
Next, go to *11_Segment*, and compile *dice.c*

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/11_Segment/
gcc dice.c -lwiringPi
```

Run.

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

Now you should see a number flashing between 0 and 6 quickly on the segment display. Press the button on the breadboard, and the display will statically display a random number between 0 and 6 for 2 seconds and then circularly flash randomly between 0 and 6 again.



Summary

Through this lesson, you may have mastered the basic principle and programming for 7-segment display based on Raspberry Pi, as well as more knowledge about using 74HC595. Now you can apply what you've learnt and put it into practice to create your own works!

Lesson 12 Driving Dot-Matrix by 74HC595

Introduction

With low-voltage scanning, dot matrix LED displays have advantages such as power saving, long service life, low cost, high brightness, a wide angle of view, long visual range, waterproofness, and so on. They can meet the needs of different applications and thus have a broad development prospect. In this lesson, we will learn how to use 74HC595 to drive an LED dot-matrix.

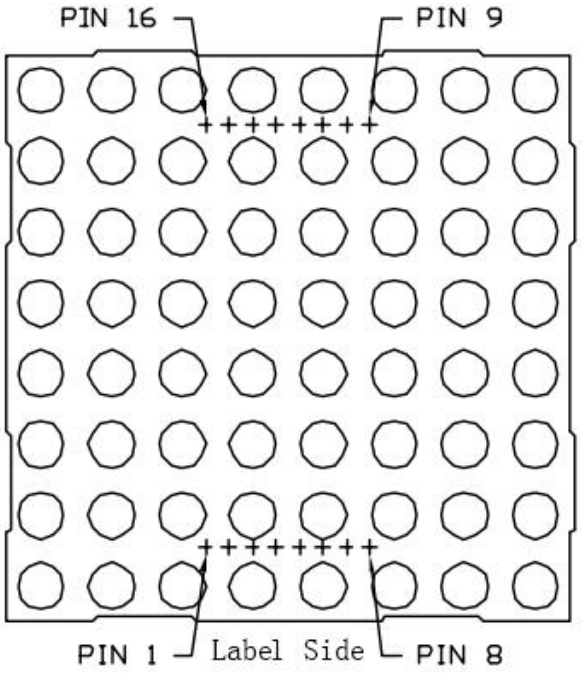
Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 2 * 74HC595
- 1 * Dot-Matrix
- Jumper wires

Principle

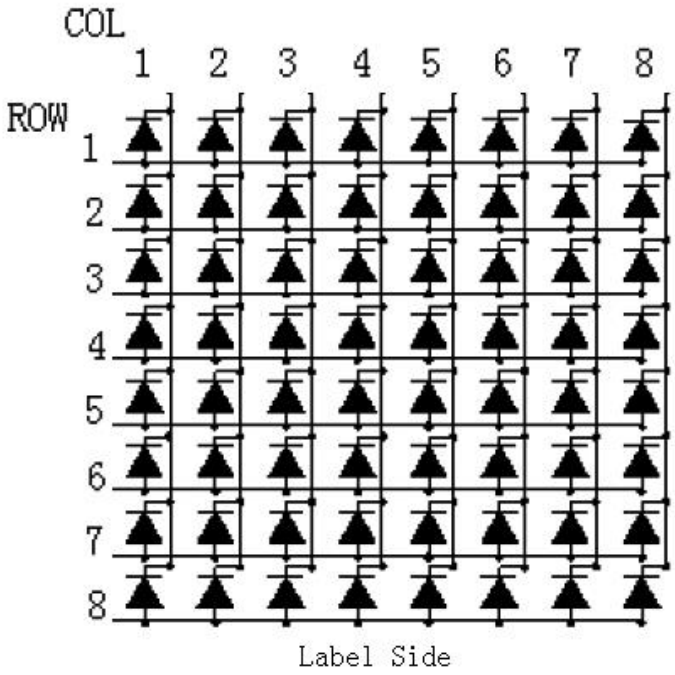
Dot Matrix

The external view:



Pin definition:

Define the row and column numbering at first (only for the dot matrix whose model number ends with BS)



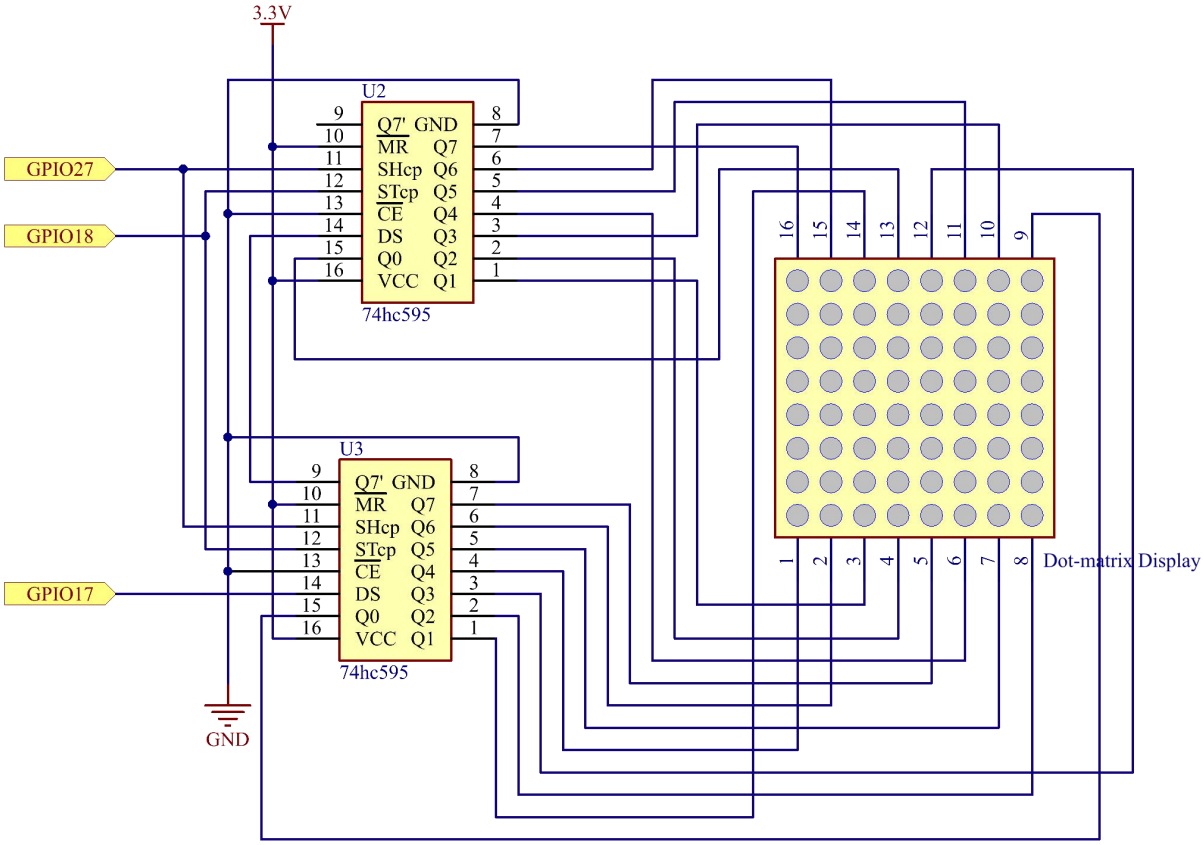
Pin numbering corresponding to the above rows and columns:

COL	1	2	3	4	5	6	7	8
Pin No.	13	3	4	10	6	11	15	16
ROW	1	2	3	4	5	6	7	8
Pin No.	9	14	8	12	1	7	2	5

The 8*8 dot matrix is made up of sixty-four LEDs and each LED is placed at the cross point of a row and a column. When the electrical level of a certain row is High and the electrical level of a certain column is Low, the corresponding LED at their cross point will light up. For example, to turn on the LED at the first dot, you should set ROW 1 to high level and COL 1 to low, so the LED at the first dot brightens; to turn on all the LEDs on the first row, set ROW 1 to high level and COL 1-8 to low, and then all the LEDs on the first row will light up; similarly, set COL 1 to low level and ROW 1-8 to high level, and all the LEDs on the first column will light up.

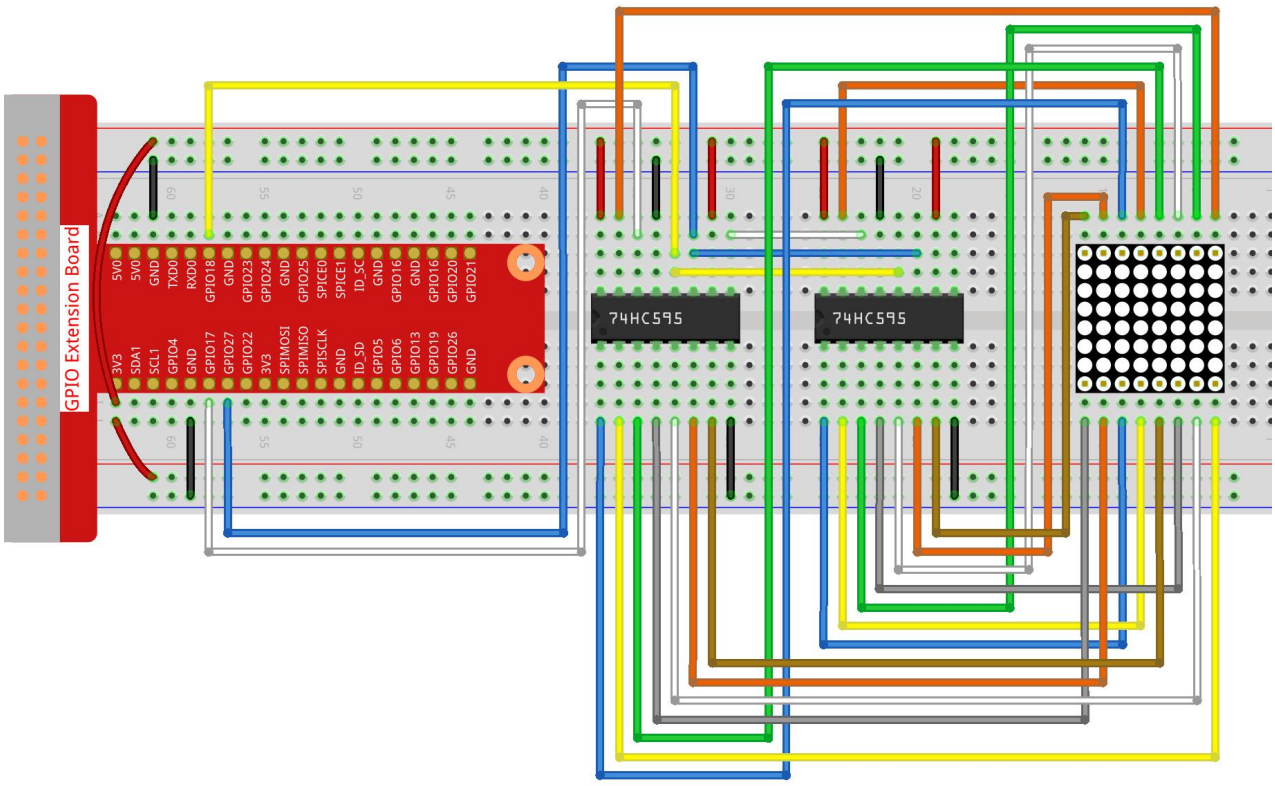
The principle of 74HC595 has been illustrated previously. One chip is used to control the rows of the dot matrix while the other, the columns.

Schematic Diagram



Experimental Procedures

Step 1: Build the circuit.



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/12_DotMatrix/
```

Step 3: Compile.

```
gcc dotMatrix.c -o dotMatrix -lwiringPi
```

Step 4: Run.

```
sudo ./dotMatrix
```

For Python Users:

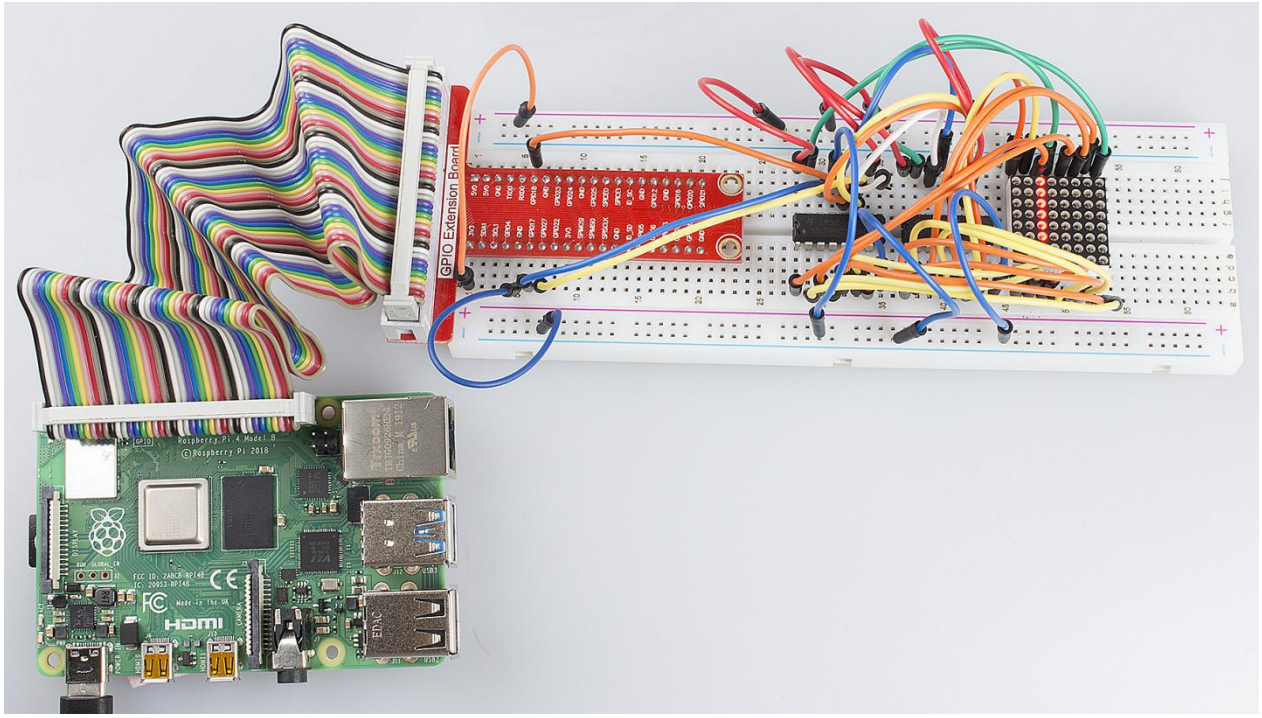
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 12_DotMatrix.py
```

You should see LEDs light up as you control.



Summary

Through this lesson, you have got the basic principle of LED dot matrix and how to program the Raspberry Pi to drive an LED dot matrix based on 74HC595 cascade. With the knowledge learnt, try more fascinating creations!

Lesson 13 LCD1602

Introduction

In this lesson, we will learn how to use LCD1602 to display characters and strings.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * LCD1602

- 1 * Potentiometer

- Jumper wires

Principle

LCD1602, or character type LCD1602, is a dot matrix LCD module specially used to display letters, figures, symbols, and so on. It consists of many 16*2 dot matrixes, and each one is composed of 5*7 or 5*11 character bit. Each character bit can display one character. There is a dot space between each adjacent character bit. Also there is a dot space between each row. The dot space functions as a character space or line space; thus, LCD1602 cannot display graphics very well. It is widely used in pocket instruments and low power application systems due to its micro power consumption, small size, richness in contents, ultra-thinness and lightness.

LCD1602 uses the standard 16-pin port, among which:

Pin 1 (GND): connected to Ground

Pin 2 (Vcc): connected to 5V power supply

Pin 3 (Vo): used to adjust the contrast of LCD1602; the level is lowest when it's connected to a positive power supply, and highest when connected to ground (you can connect a 10K potentiometer to adjust its contrast when using LCD1602)

Pin 4 (RS): register select pin that controls where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

Pin 5 (R/W): to read/write signals; it reads signals when supplied with high level (1), and writes when low level (0) (in this experiment, you only need to write data to LCD1602, so just connect this pin to ground)

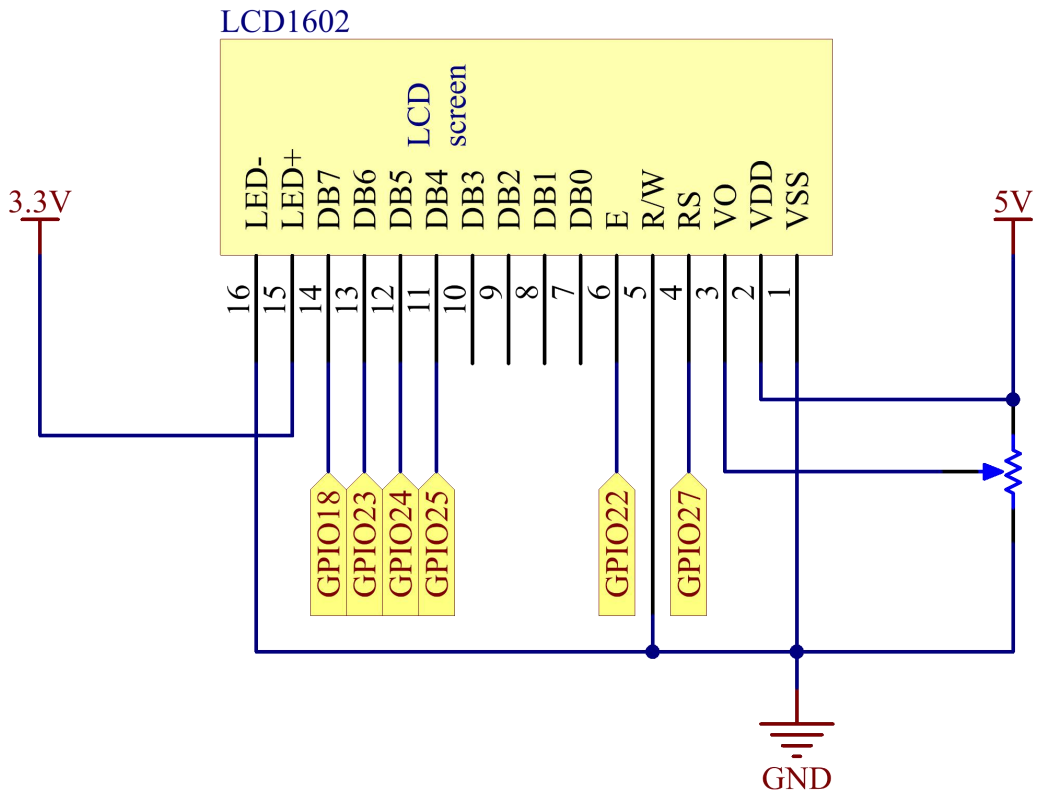
Pin 6 (E): An enable pin that, when low-level energy is supplied, causes the LCD module to execute relevant instructions

Pin 7 (D0-D7): pins that read and write data

A and K: controlling LCD backlight

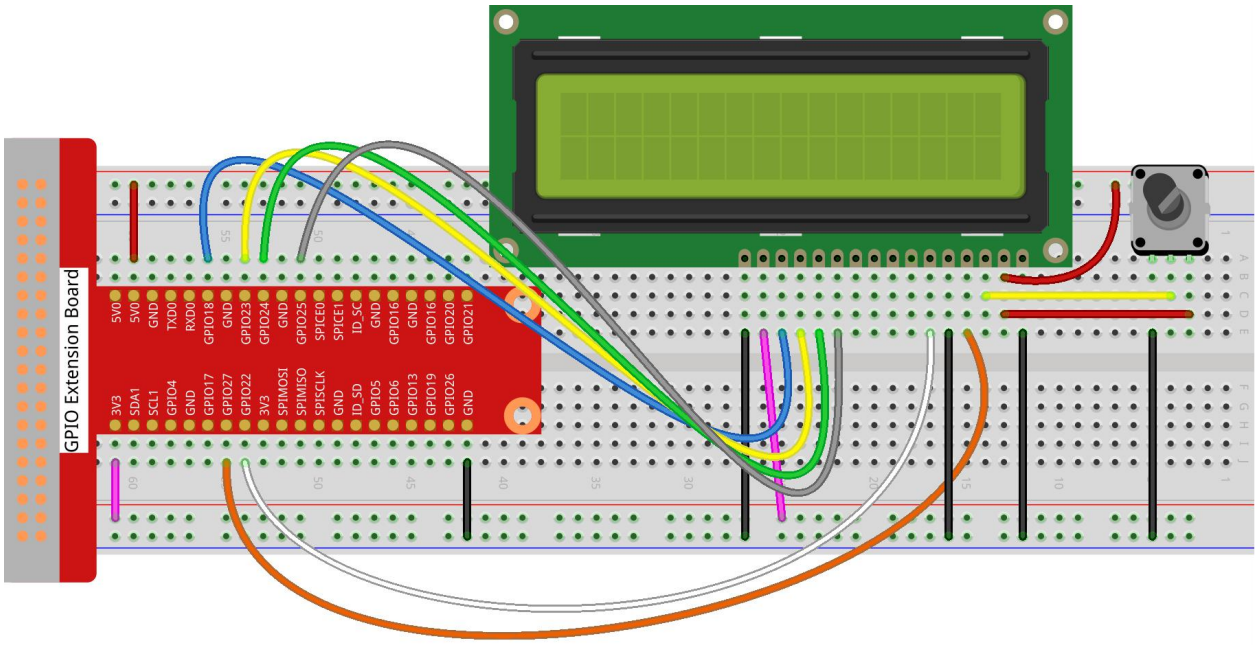
LCD1602 has two operation modes: 4-bit and 8-bit. When the IOs of microprocessor (MCU) are insufficient, you can choose 4-bit mode, under which only pins D4~D7 are used. After connecting the circuit, you can operate LCD1602 by Raspberry Pi.

Schematic Diagram



Experimental Procedures

Step1: Build the circuit (please be sure the pins are connected correctly. Otherwise, characters will not be displayed properly):



For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/13_LCD1602/
```

Step 3: Compile.

```
gcc lcd1602_2.c -o lcd1602_2 -lwiringPiDev -lwiringPi
```

Step 4: Run.

```
sudo ./lcd1602_2
```

For Python Users:

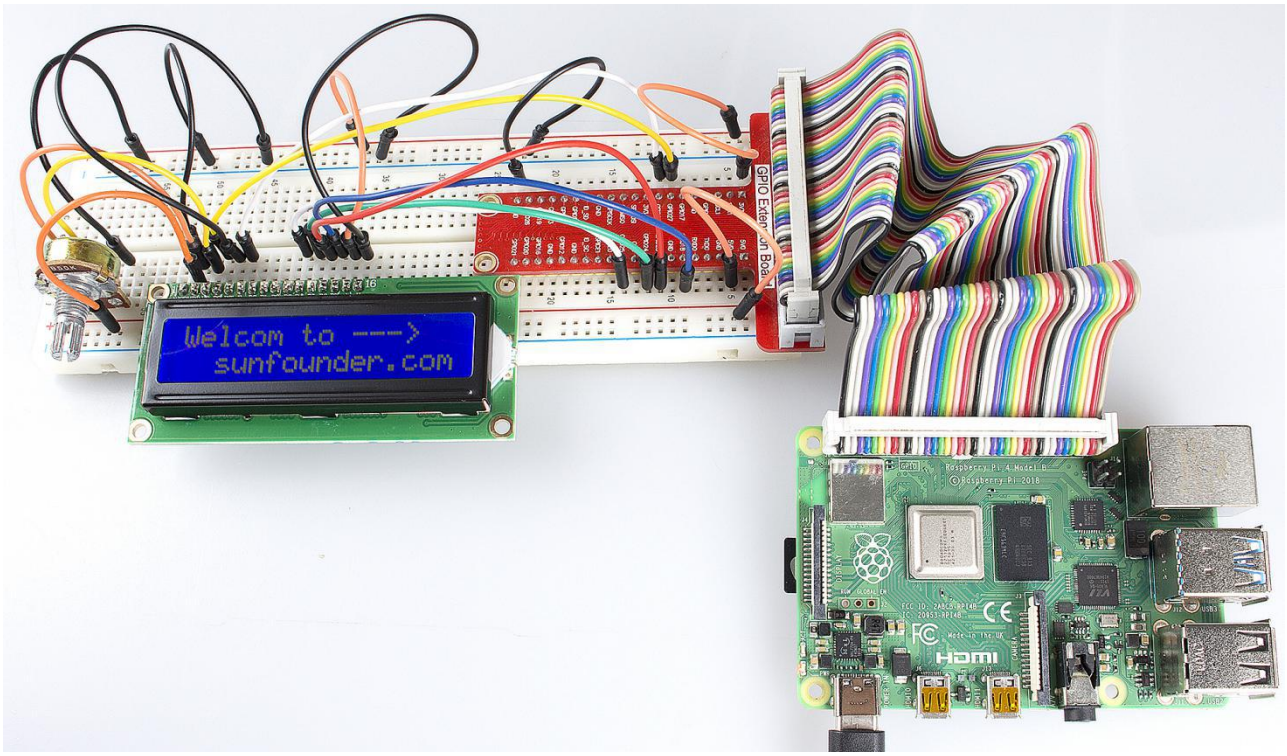
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 13_lcd1602.py
```

You should see two lines of characters displayed on the LCD1602: "**SUNFOUNDER**" and "**Hello World ! :)**".



Further Exploration

In this experiment, the LCD1602 is driven in the 4-bit mode. You can try programming by yourself to drive it in the 8-bit mode.

Lesson 14 ADXL345

Introduction

In this lesson, we will learn how to use the acceleration sensor ADXL345.

Components

- 1 * Raspberry Pi

- 1 * Breadboard

- 1 * ADXL345 module

- Jumper wires

Principle

ADXL345

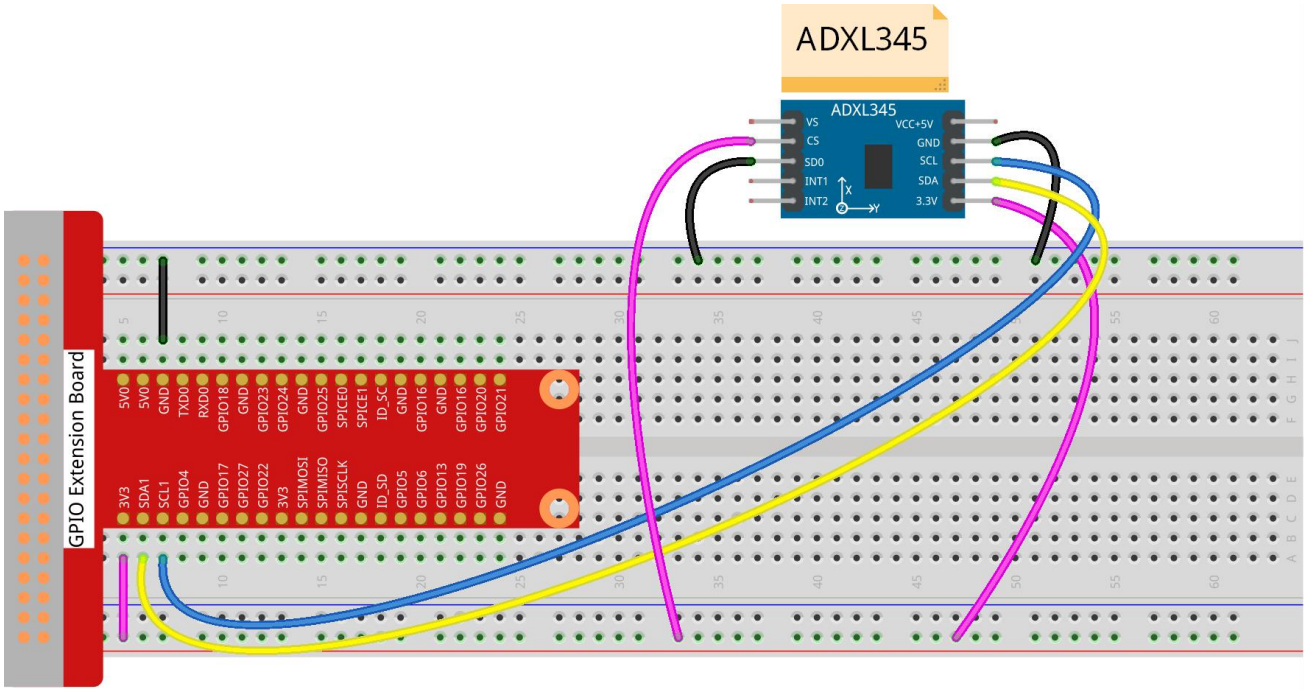
The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit two's complement and is accessible through either an SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables the inclination change measurement by less than 1.0°. And the excellent sensitivity (3.9mg/LSB @2g) provides a high-precision output of up to ±16g. In this experiment, I2C digital interface is used.

Experimental Procedures

Step 1: Build the circuit.

ADXL345 Module		Raspberry Pi
GND	----------	GND
3.3V	----------	3.3V
SCL	----------	SCL1
SDA	----------	SDA1
CS	----------	3.3V
SDO	----------	GND



The I2C interface is used in the following program. Before running the program, please make sure the I2C driver module of Raspberry Pi has loaded normally(Refer to Appendix).

For C Language Users:

Step 2: Change directory.

```
cd /home/pi/Sunfounder_SuperKit_C_code_for_RaspberryPi/14_ADXL345/
```

Step 3: Compile.

```
gcc adxl345.c -o adxl345 -lwiringPi
```

Step 4: Run.

```
sudo ./adxl345
```

For Python Users:

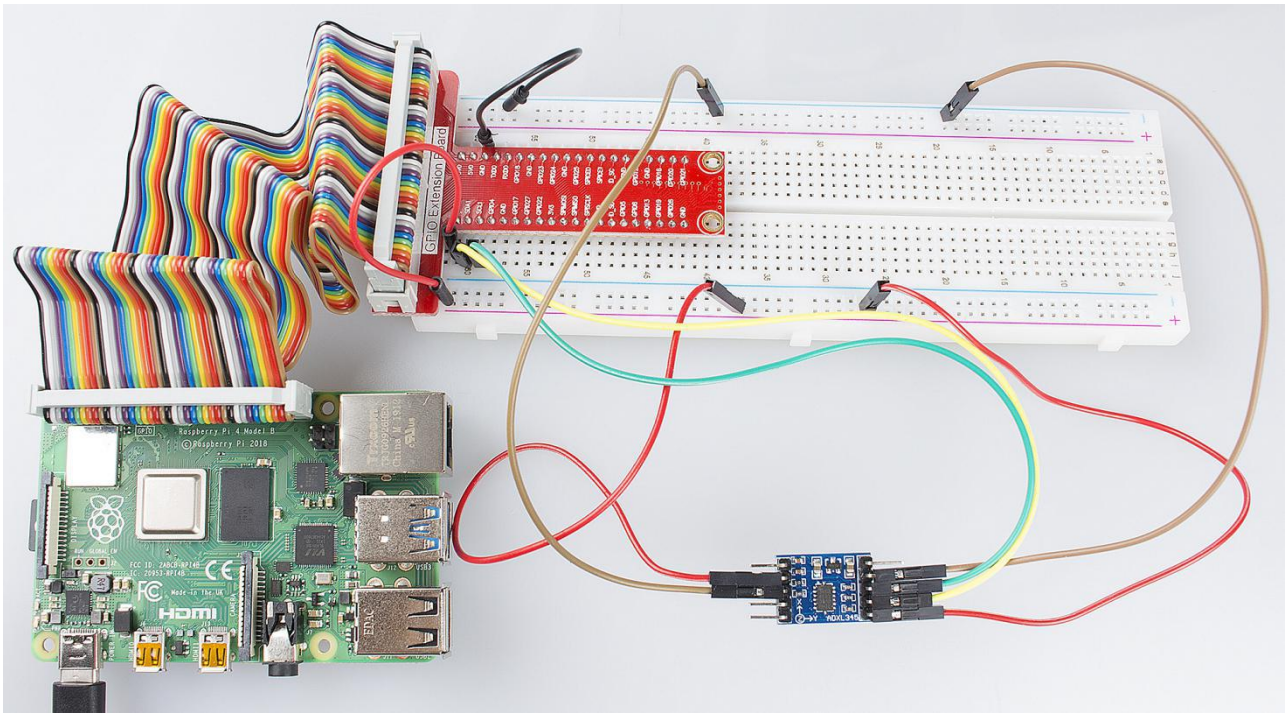
Step 2: Change directory.

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

Step 3: Run.

```
sudo python3 14_ADXL345.py
```

Now, rotate the acceleration sensor, and you should see the values printed on the screen change.


GPIO Extension Board
Raspberry Pi 4 Model B
©Raspberry Pi 2018
HDMI
ADXL345

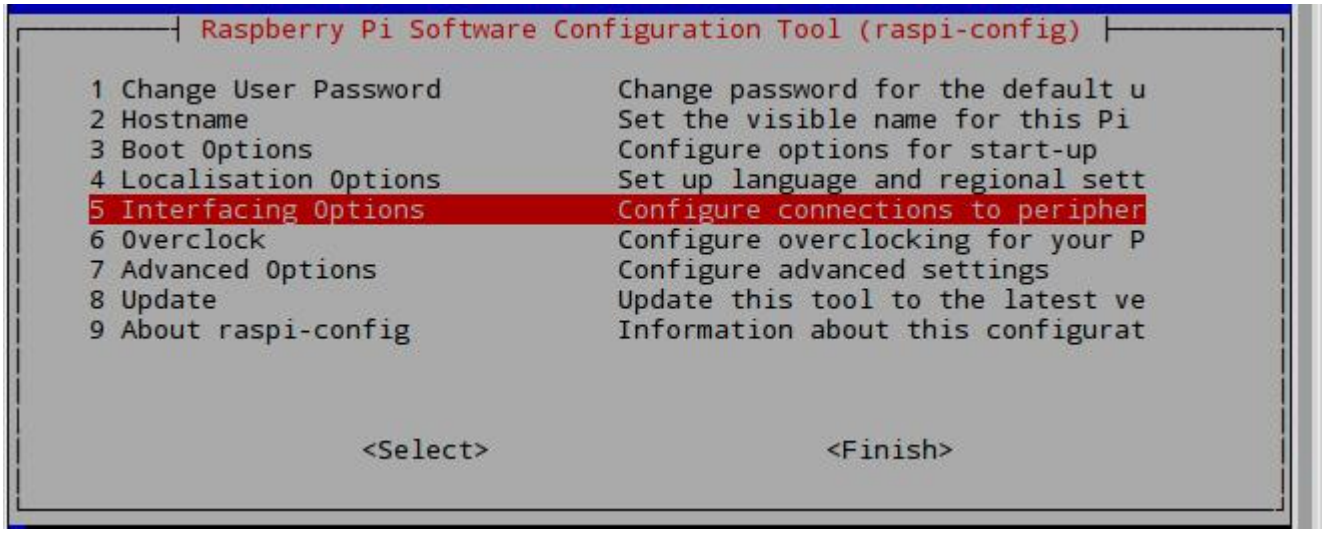
Appendix

I2C Configuration

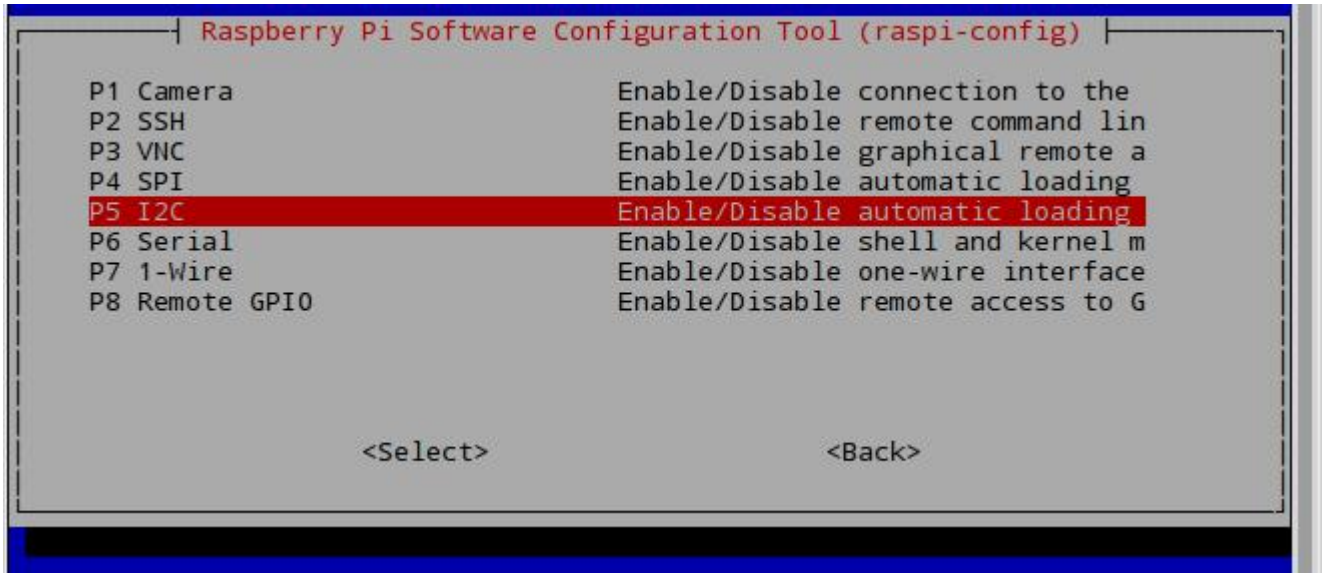
Step 1: Enable the I2C port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

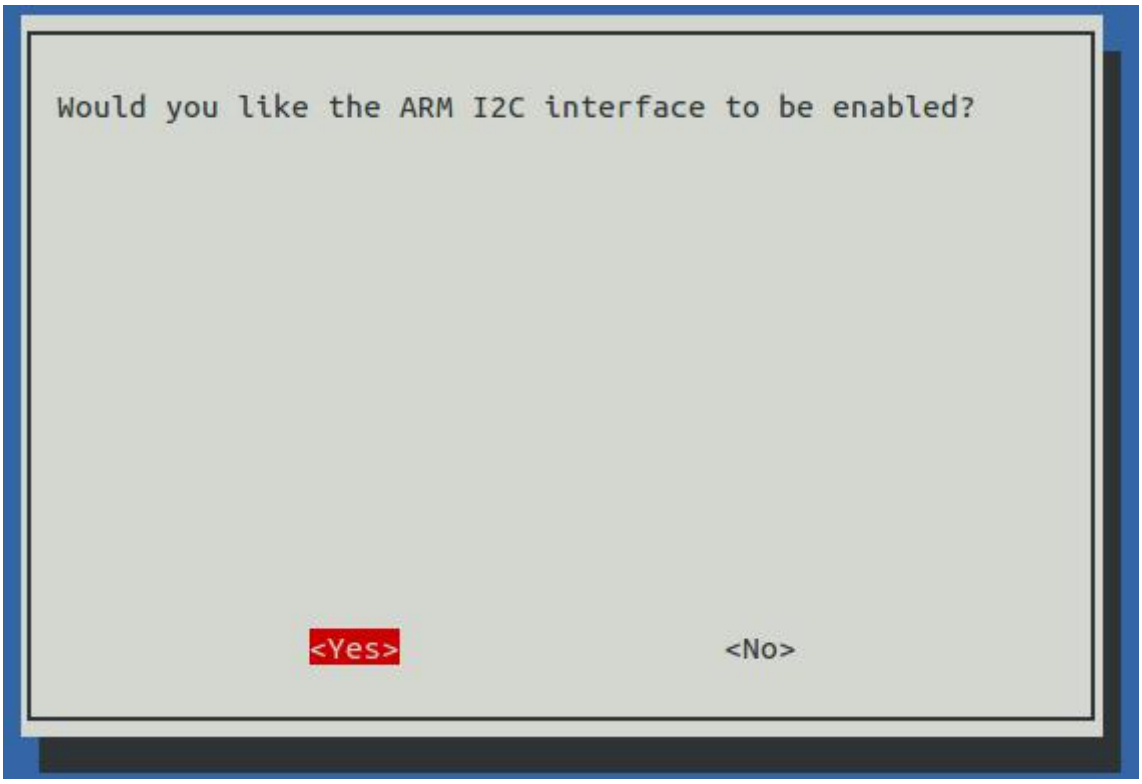
5 Interfacing options



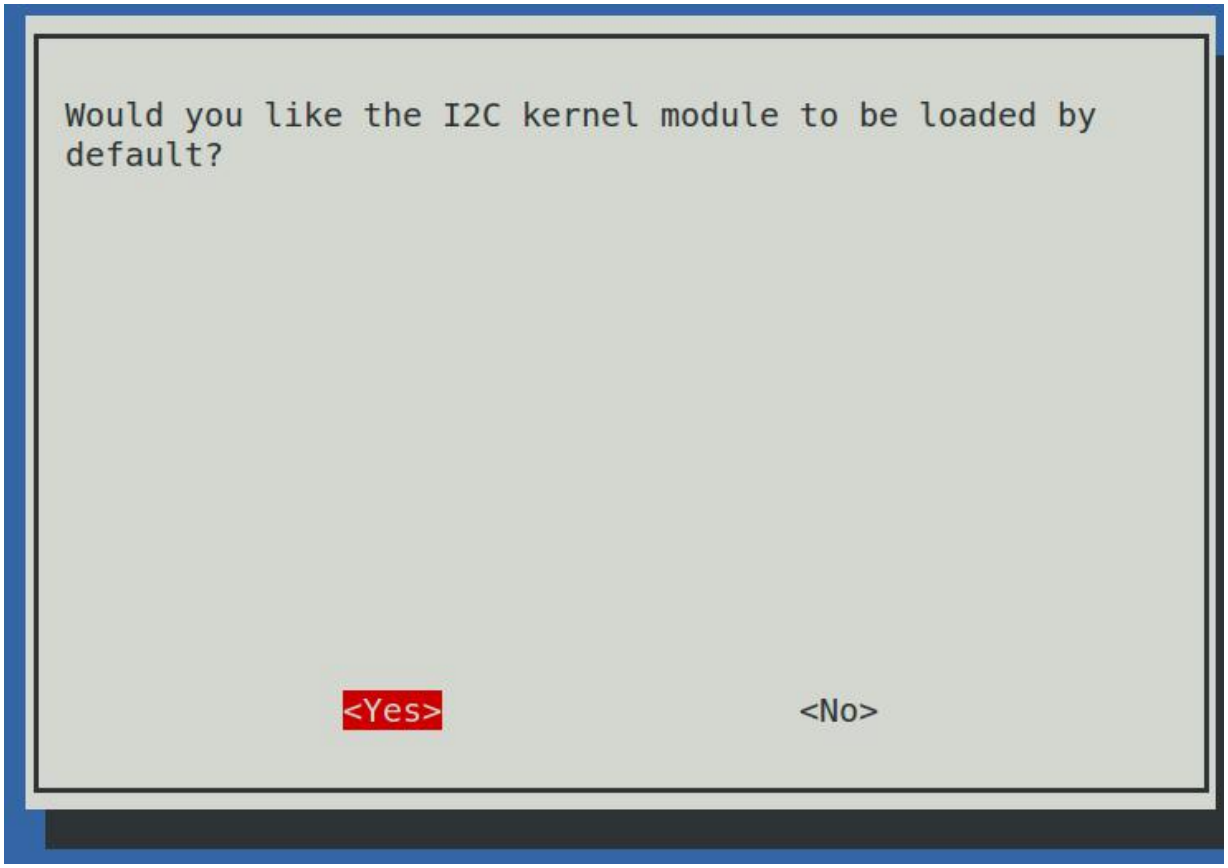
P5 I2C



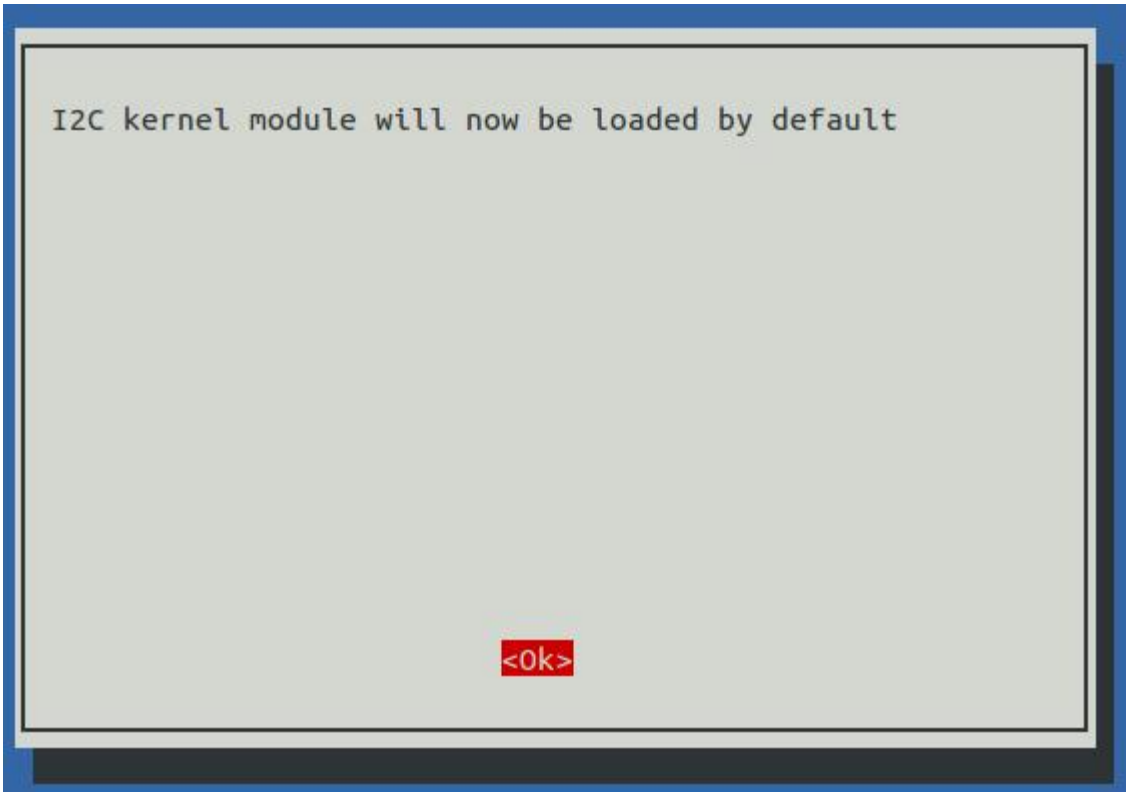
<Yes>


Would you like the ARM I2C interface to be enabled?
<Yes>
<No>

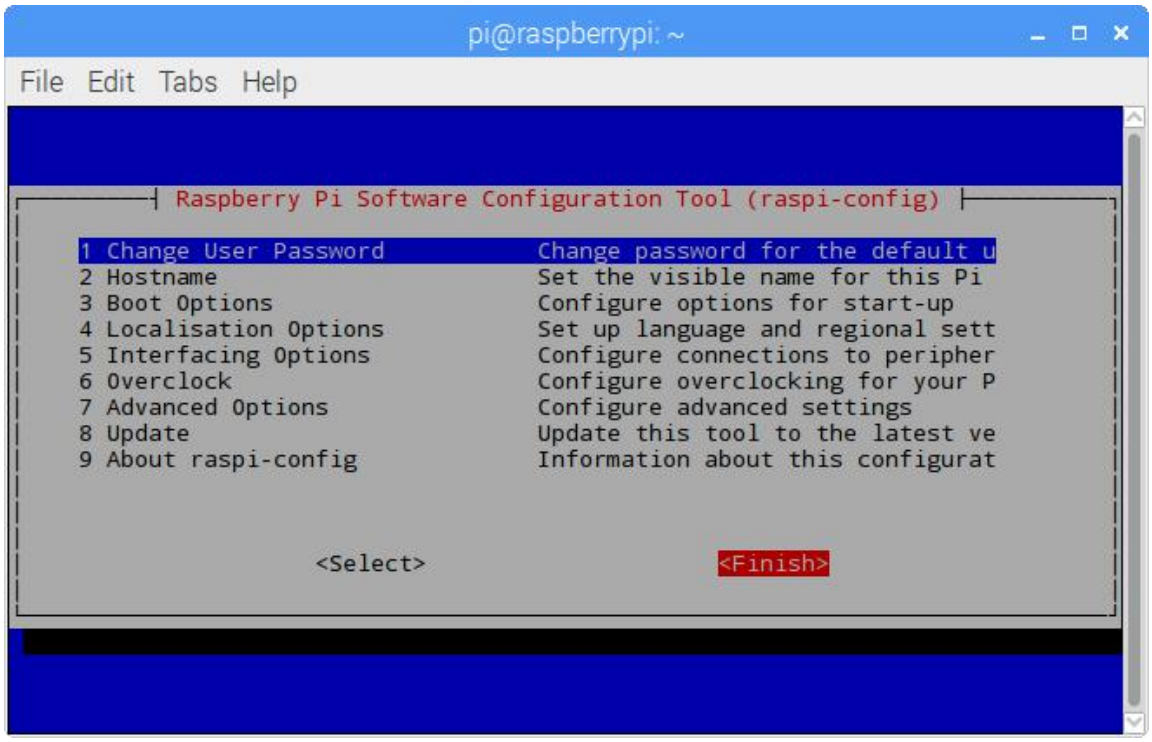
<Yes>


Would you like the I2C kernel module to be loaded by default?
<Yes>
<No>

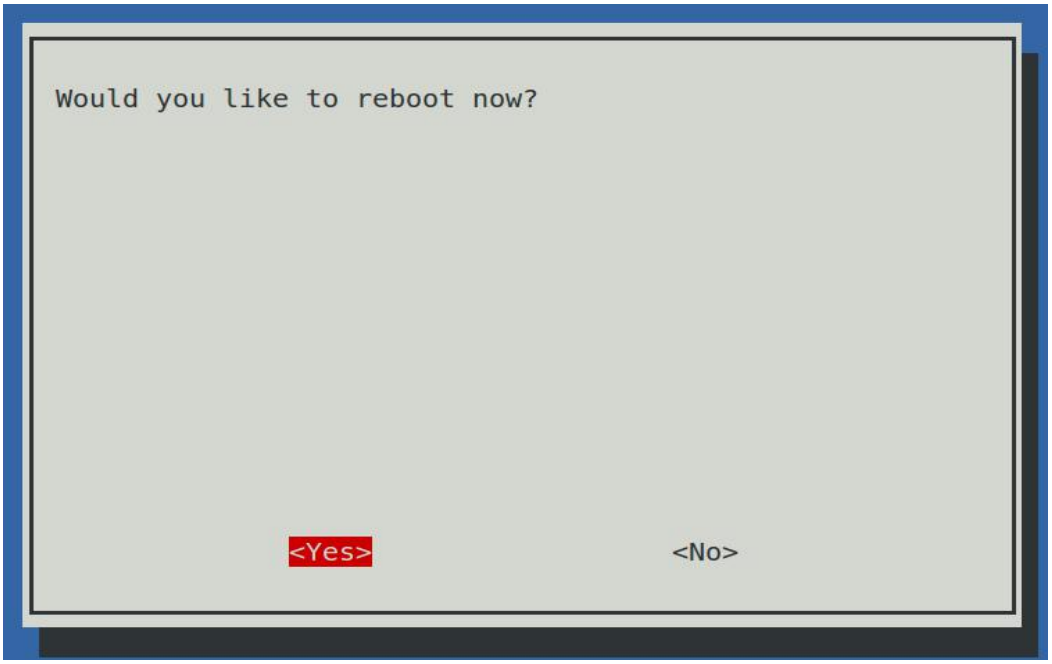
<Ok>



<Finish>



<Yes> (If you do not see this page, continue to the next step)



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's an I2C device connected, the results will be similar as shown above - since the address of the device is 0x48, 48 is printed.

Step 5:

For C language users: Install libi2c-dev.

```
sudo apt-get install libi2c-dev
```

For Python users: Install smbus for I2C.

```
sudo apt-get install python3-smbus
```

Copyright Notice