Preface

About Sunfounder

As an open source electronics service provider, SUNFOUNDER is committed to developing open source electronics educational products, expanding related knowledge to let hobbyists have a good understanding about open source electronics and providing a platform through which people can learn, develop hands-on skills, and improve their innovative abilities.

About Super Kit

This super kit is suitable for the Raspberry Pi B, model B+ and Raspberry Pi 2 model B.

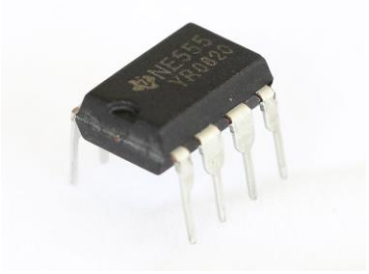
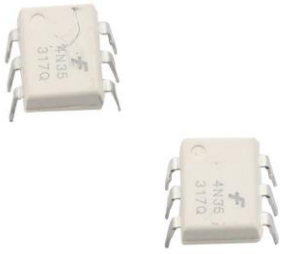
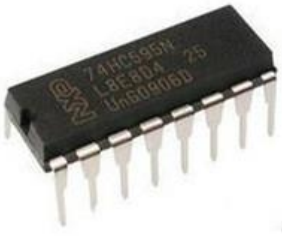
In this book, we will show you circuits with both realistic illustrations and schematic diagrams. All related codes and videos are included on the CD attached. If you cannot play the CD, you can go to our official website at www.sunfounder.com to watch our video tutorials and download related materials. You can also watch these videos on our YouTube channel at https://www.youtube.com/channel/UCf5Q-WNdLaYEx6MfLIPSTgg.

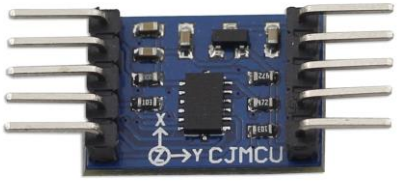
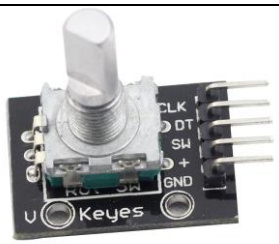
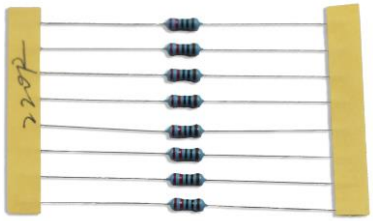
If you have any questions, you can leave a message on our forum. We also welcome you to publish your own projects on our blog to share the fun of creating and making.

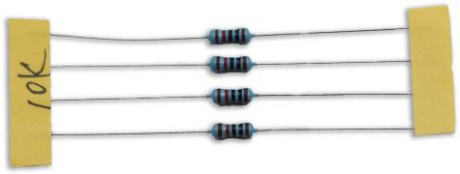
Notes:

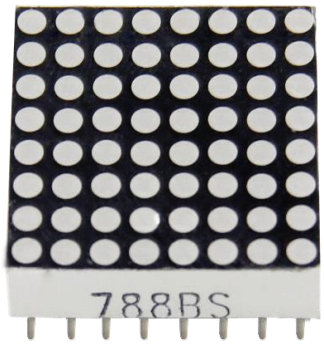
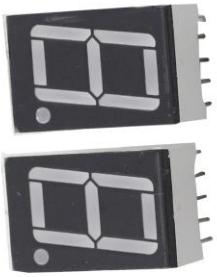
After unpacking, please check that the number of components is correct and that all components are in good condition.

Components List

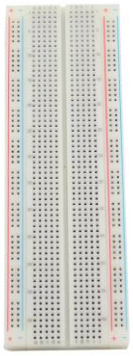
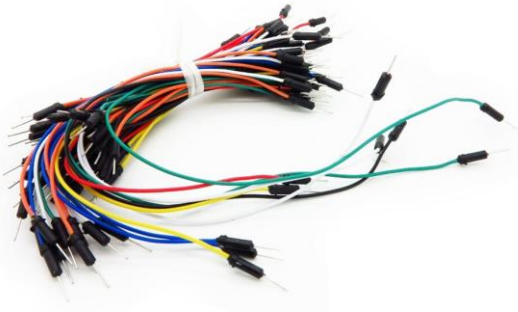
No	Product Name	Quantity	Picture
1	RGB LED	1	
2	Pin Header	40	
3	555 Timer IC	1	
4	Optocoupler (4N35)	2	
5	Shift Register (74HC595)	2	

6	L293D	1	
7	Accelerometer ADXL345	1	
8	Rotary Encoder	1	
9	Button	5	
10	Resistor (220Ω)	8	
11	Resistor (1kΩ)	8	

12	Resistor (10kΩ)	4	
13	Resistor (100kΩ)	4	
14	Resistor (1MΩ)	1	
15	Resistor (5.1MΩ)	1	
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	Transistor (NPN)	2	
28	Transistor (PNP)	2	

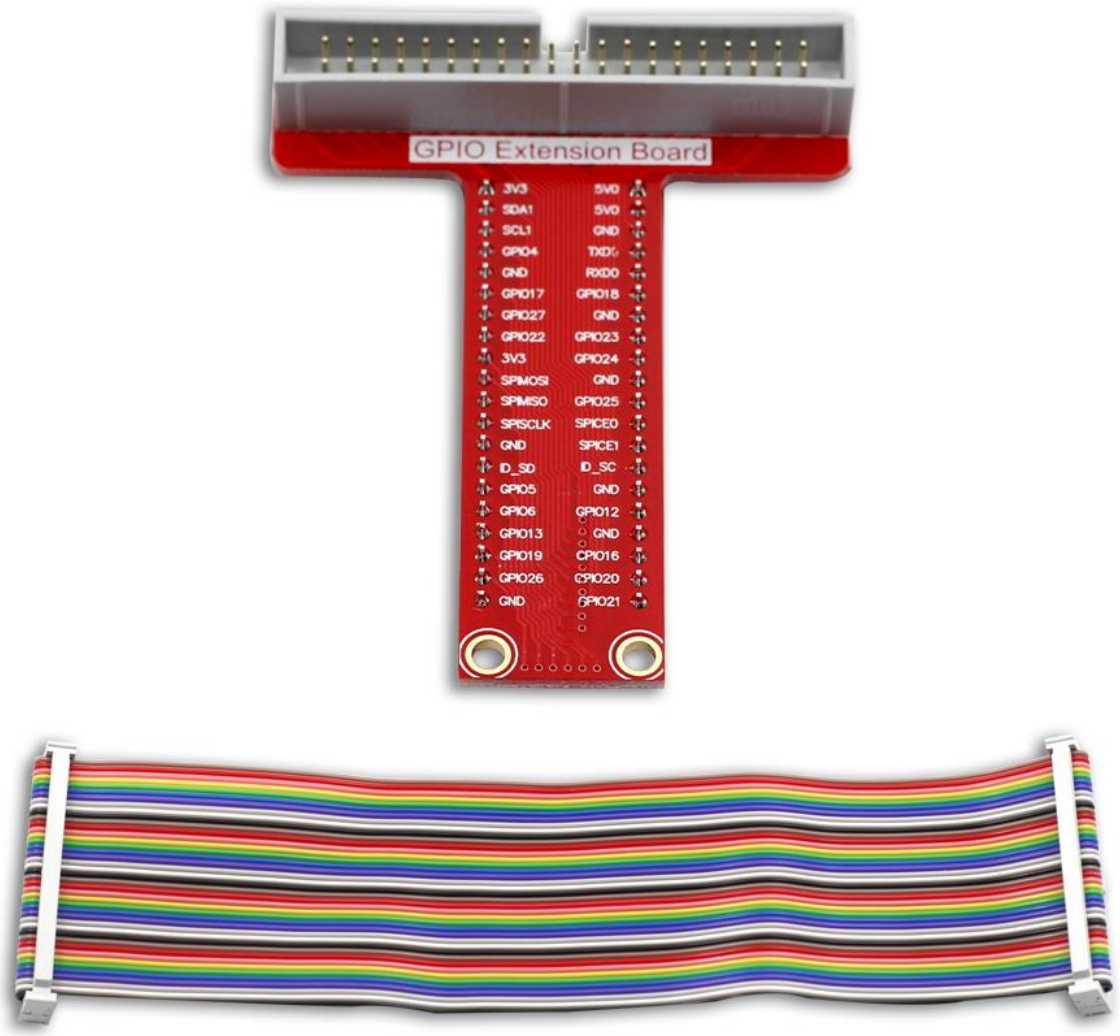
29	Capacitor Ceramic 100nF	4	
30	Capacitor Ceramic 10nF	4	
31	Diode Rectifier	4	
32	Breadboard	1	
33	Male-to-Male Jumper Wire	65	

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

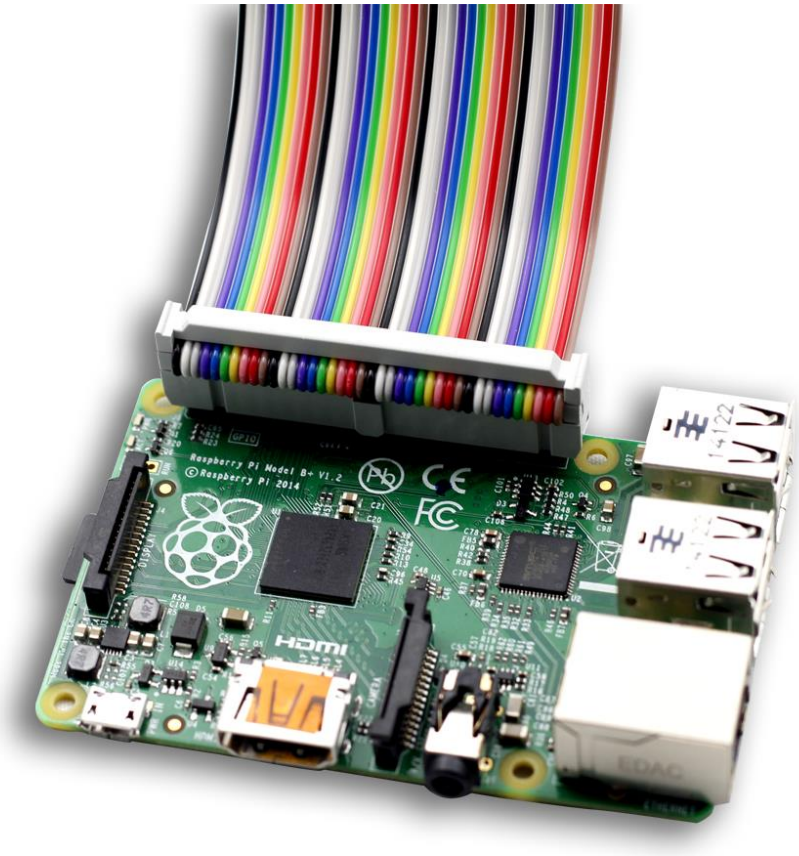
Contents

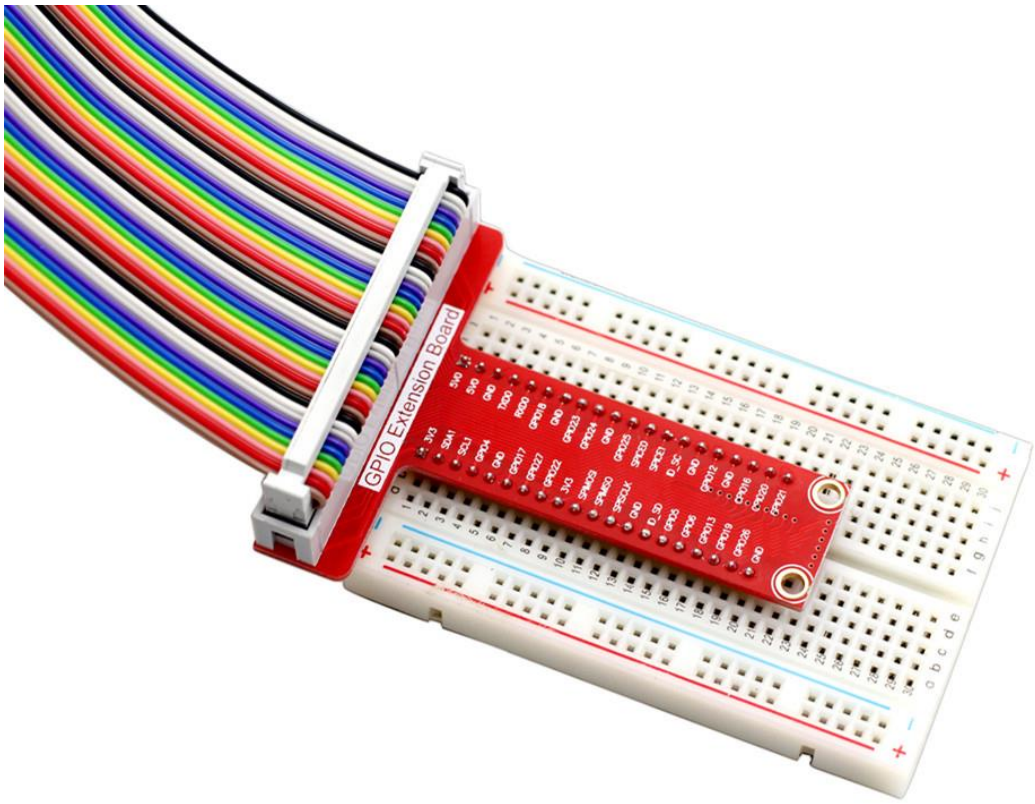
Notice

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

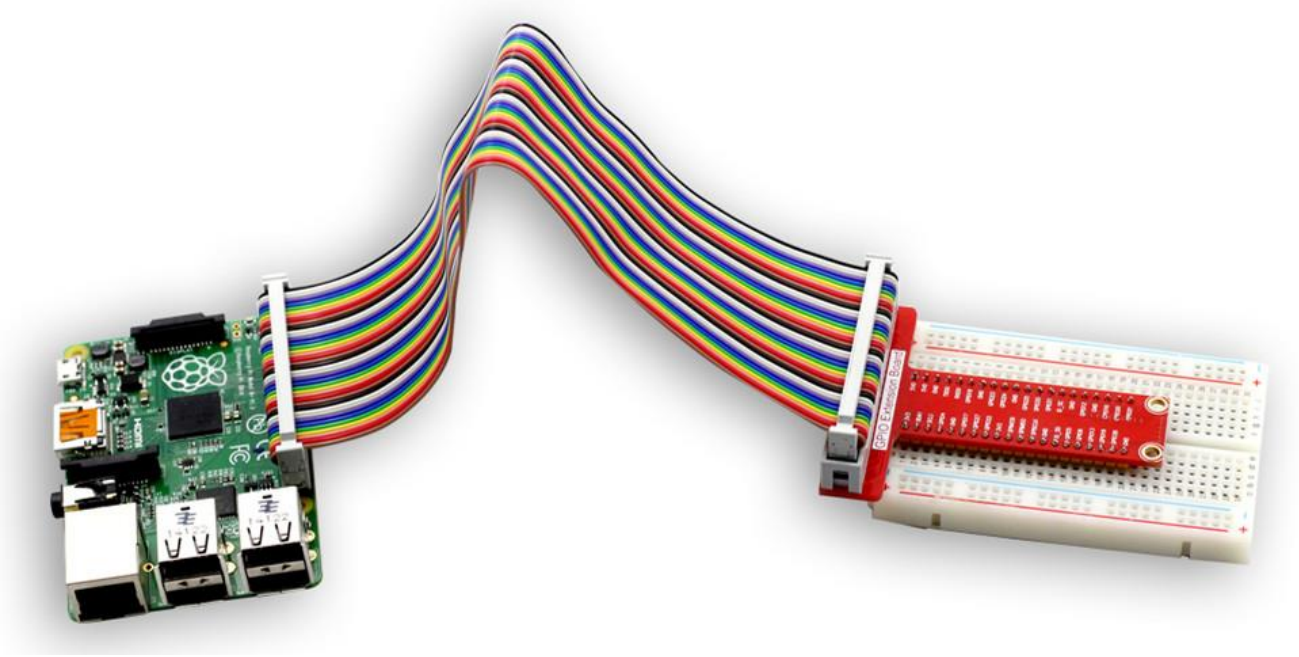


Connect the GPIO cable to the Raspberry Pi B+ like this:


Raspberry Pi Model B+ V1.2
©Raspberry Pi 2014
HDMI

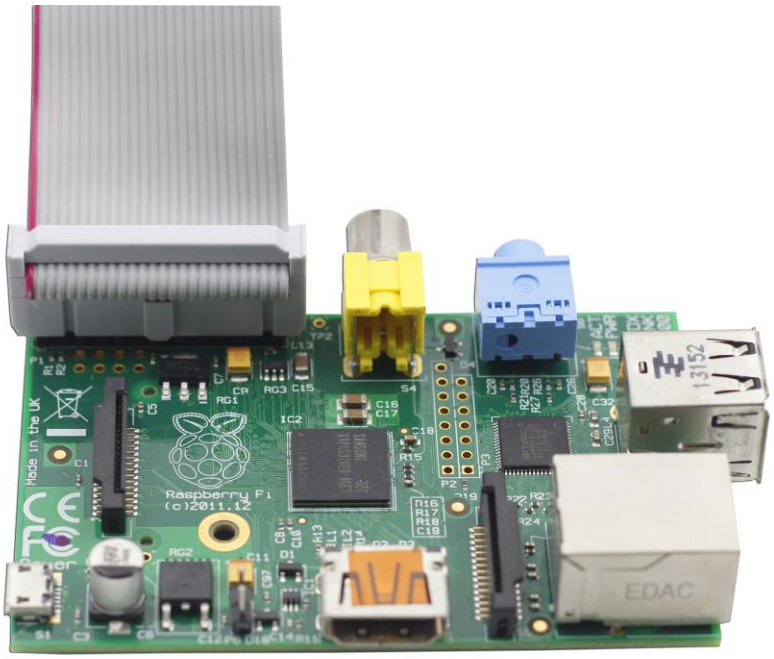

GPIO Extension Board

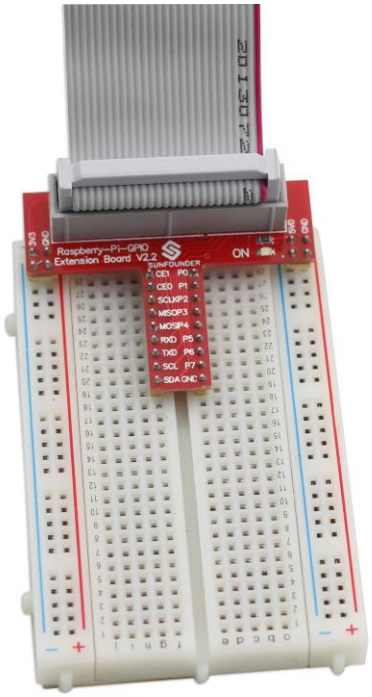
After connection, it is shown as follows:



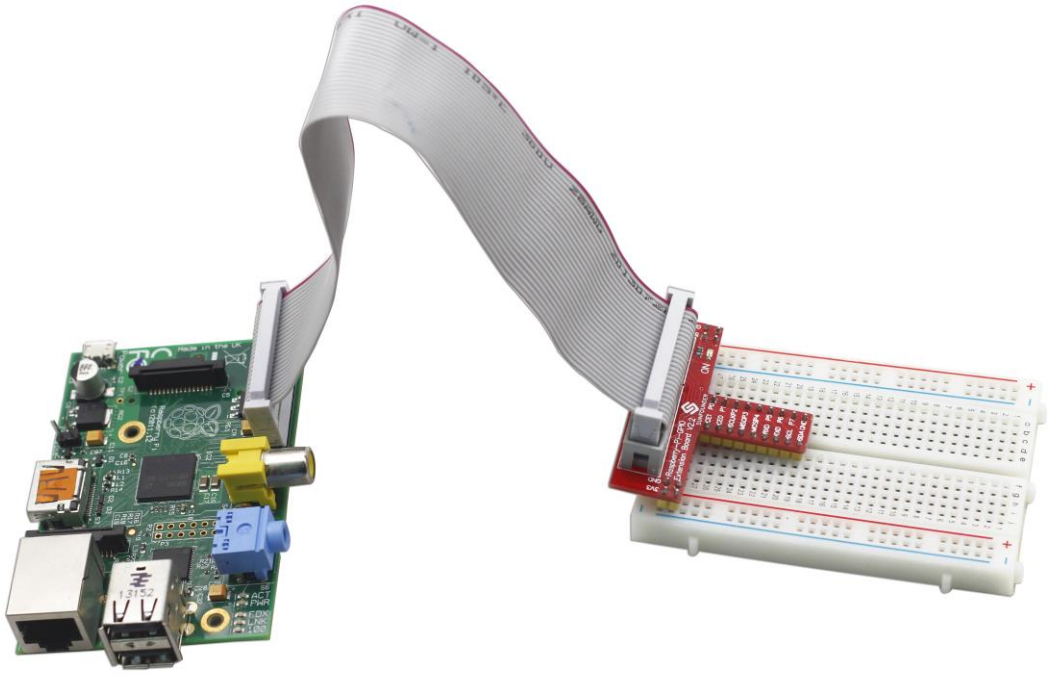
This is our 20-pin GPIO Extension Board and GPIO cable for Raspberry Pi B.

Connect the GPIO cable to the Raspberry Pi B like this:





After connection, it is shown as follows:



Raspberry Pi Pin Number Introduction:

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin
–	–	3.3v	1 | 2	5v	–	–
8	R1:0/R2:2	SDA0	3 | 4	5v	–	–
9	R1:1/R2:3	SCL0	5 | 6	0V	–	–
7	4	GPIO7	7 | 8	TXD	14	15
–	–	0V	9 | 10	RXD	15	16
0	17	GPIO0	11 | 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 | 14	0V	–	–
3	22	GPIO3	15 | 16	GPIO4	23	4
–	–	3.3v	17 | 18	GPIO5	24	5
12	10	MOSI	19 | 20	0V	–	–
13	9	MISO	21 | 22	GPIO6	25	6
14	11	SCLK	23 | 24	CE0	8	10
–	–	0V	25 | 26	CE1	7	11
30	0	SDA. 0	27 | 28	SCL. 0	1	31
21	5	GPIO. 21	29 | 30	0V	–	–
22	6	GPIO. 22	31 | 32	GPIO. 26	12	26
23	13	GPIO. 23	33 | 34	0V	–	–
24	19	GPIO. 24	35 | 36	GPIO. 27	16	27
25	26	GPIO. 25	37 | 38	GPIO. 28	20	28
		0V	39 | 40	GPIO. 29	21	29
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin

For RPi B

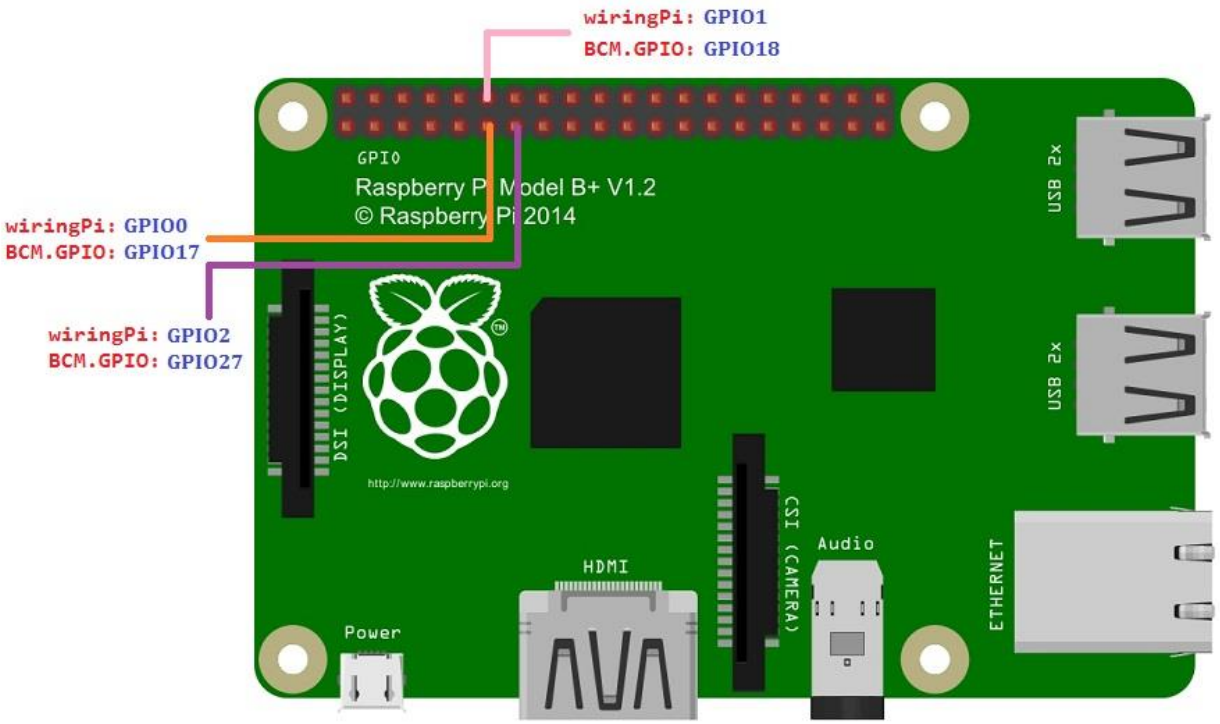
For RPi B+/ 2 model B

Currently, there are three methods of pin numbering for Raspberry Pi.

1. Numbering based on the physical location of pins
2. Numbering appointed by C language GPIO library wiringPi
3. Numbering appointed by BCM2835 SOC

If we want to operate Raspberry Pi GPIOs in C language based on the wiringPi library, choose numbering appointed by wiringPi. You can see from the above diagram that GPIO0 in wiringPi corresponds to pin 11 numbered by physical location, and GPIO30, to pin 27.

The picture below demonstrates the physical numbers of the three pins 11, 12, and 13 in detail:



If you are a C user, please install wiringPi.

WiringPi introduction:

WiringPi is a GPIO library function 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.

How to install

Step 1: Get the source code of wiringPi

```
git clone git://git.drogon.net/wiringPi
```

Step 2: Install wiringPi

```
cd wiringPi
git pull origin
./build
```

After you press Enter, with the script *build,* the source code of wiringPi will be compiled automatically and installed to the appropriate directory of Raspberry Pi OS.

Step 3: Test whether wiringPi is installed successfully or not

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

```
root@raspberrypi:/home/wiringPi# gpio -v
gpio version: 2.21
Copyright (c) 2012-2014 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Model B, Revision: 2, Memory: 512MB, Maker: Sony
```

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

gpio readall

```
root@raspberrypi:/home/wiringPi# gpio readall
 +-----+-----+---------+------+---+-Model B2-+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 1 | 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 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | ALT0 | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | ALT0 | CE1     | 11  | 7   |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |  28 |  17 | GPIO.17 |   IN | 0 | 51 || 52 | 0 | IN   | GPIO.18 | 18  | 29  |
 |  30 |  19 | GPIO.19 |   IN | 0 | 53 || 54 | 0 | IN   | GPIO.20 | 20  | 31  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+-Model B2-+---+------+---------+-----+-----+
```

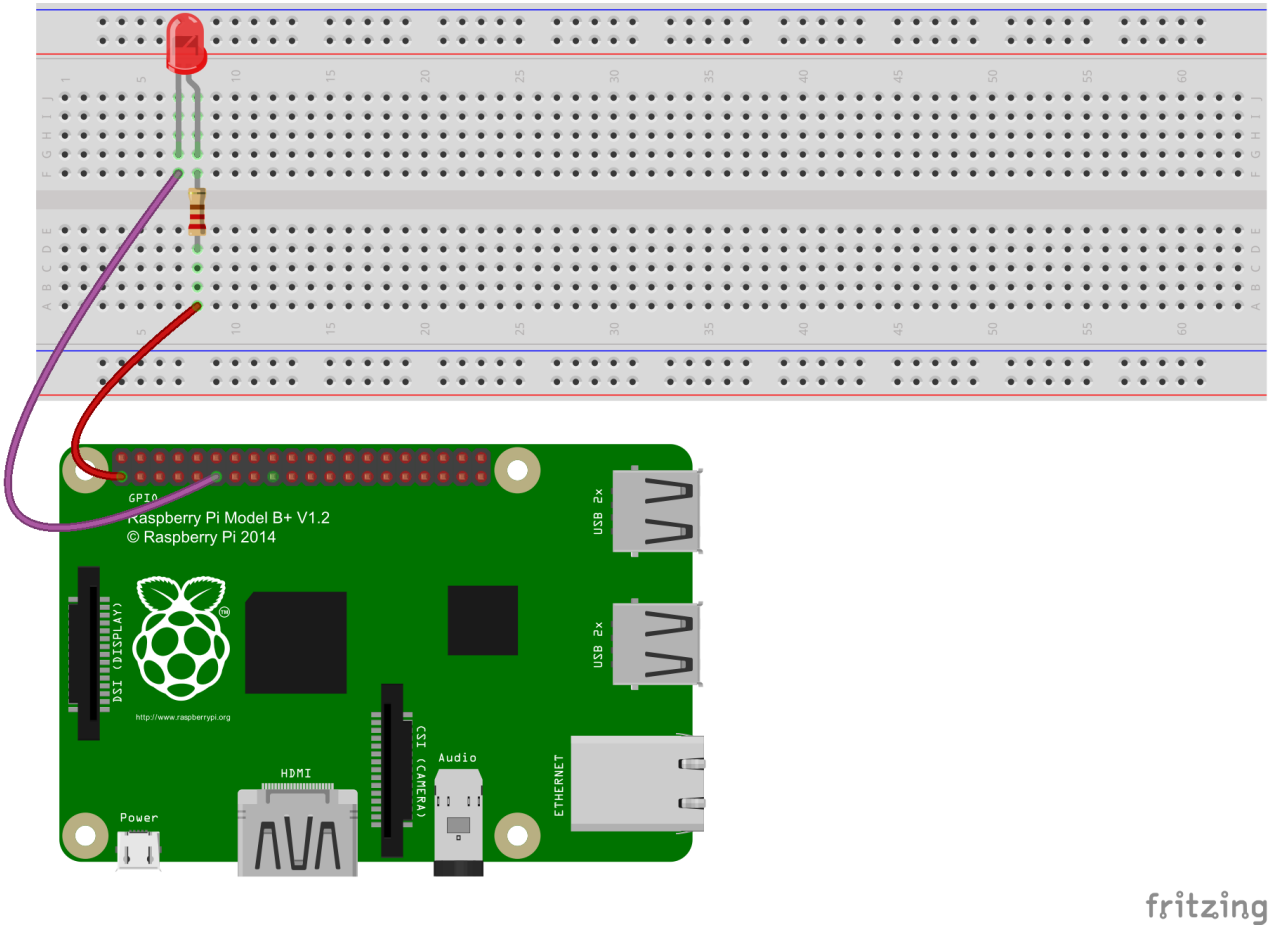
If you are a Python user, you can program GPIOs with API provided by RPi.GPIO.

RPi.GPIO Introduction:

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/

Raspbian OS image of Raspberry Pi installs RPi.GPIO by default, so you can use it directly.

Next, use C and Python language to program. You will learn how to use wiringPi and RPi.GPIO module to control Raspberry Pi GPIOs. Take blinking LED for example. First, connect circuit according to the following diagram:



Note: The resistor here is 220Ω. If the resistance is too high, the LED will be too dim or even won't light up. .

For C language users (based on the wiringPi library):

Step 1: Create a C file named *led.c*

```
touch led.c
```

Step 2: Use nano or other code edit tools to open *led.c*, write down the following code and save it.

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

#define  LedPin    0

int main(void)
{
    if(wiringPiSetup() == -1){//when the wiringPi initialization fails,
                              print message to the screen.
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(LedPin, OUTPUT);

    while(1){
        digitalWrite(LedPin, LOW);    //led on
        printf("led on...\n");
        delay(500);
        digitalWrite(LedPin, HIGH);  //led off
        printf("...led off\n");
        delay(500);
    }
    return 0;
}
```

Step 3: Compile

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

Step 4: Run

```
sudo ./led
```

You should see the LED blinking. Press Ctrl+C to terminate the program.

For Python users (based on the RPi.GPIO python module):

Step 1: Create a Python file named *led.py*

```
touch led.py
```

Step 2: Use nano or other code edit tools to open *led.py*, write down the following code and save it.

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

LedPin = 11    # pin11

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

try:
    while True:
        print '...led on'
        GPIO.output(LedPin, GPIO.LOW)  # led on
        time.sleep(0.5)
        print 'led off...'
        GPIO.output(LedPin, GPIO.HIGH) # led off
        time.sleep(0.5)
except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the following code
                             will be  executed.
    GPIO.output(LedPin, GPIO.HIGH)     # led off
    GPIO.cleanup()                     # Release resource
```

Step 3: Run

```
sudo python led.py
```

You should see the LED blinking. Press Ctrl+C to terminate the program.

SUNFOUNDER's Raspberry Pi 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

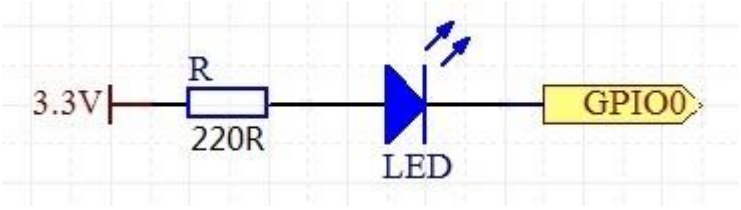
- 1* Raspberry Pi
- 1* Breadboard
- 1* LED
- 1* Resistor (220Ω)
- Jumper wires

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.

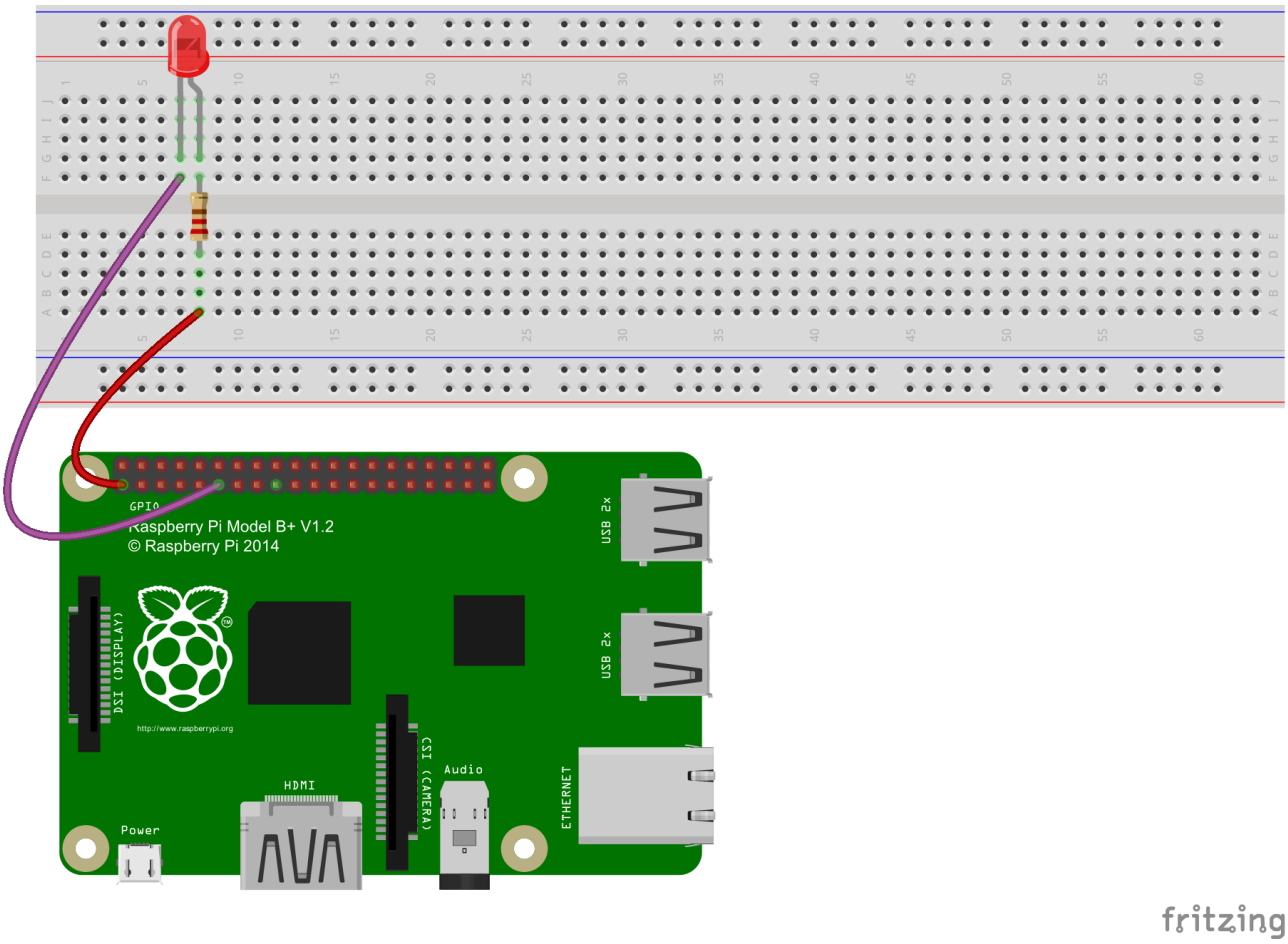
LEDs are diodes too. Hence they have a voltage drop which usually varies from 1V to 3V depending on their types. Likewise, LEDs usually emits light if supplied with 5mA–30mA current, and generally 10mA–20mA is used. So when an LED is used, it is necessary to connect a current-limiting resistor to protect the LED from over-burning.



In this experiment, connect a 220Ω resistor to the anode of the LED and then connect the resistor to 3.3 V power source, and connect the cathode of the LED to GPIO0 (See the table of Raspberry Pi pin number introduction on P12 and the above picture). Write 1 to GPIO0, and the LED will not light up; write 0 to GPIO0, and then the LED will blink according to principle mentioned previously.

Experimental Procedure

Step 1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/01_LED/led.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./led
```

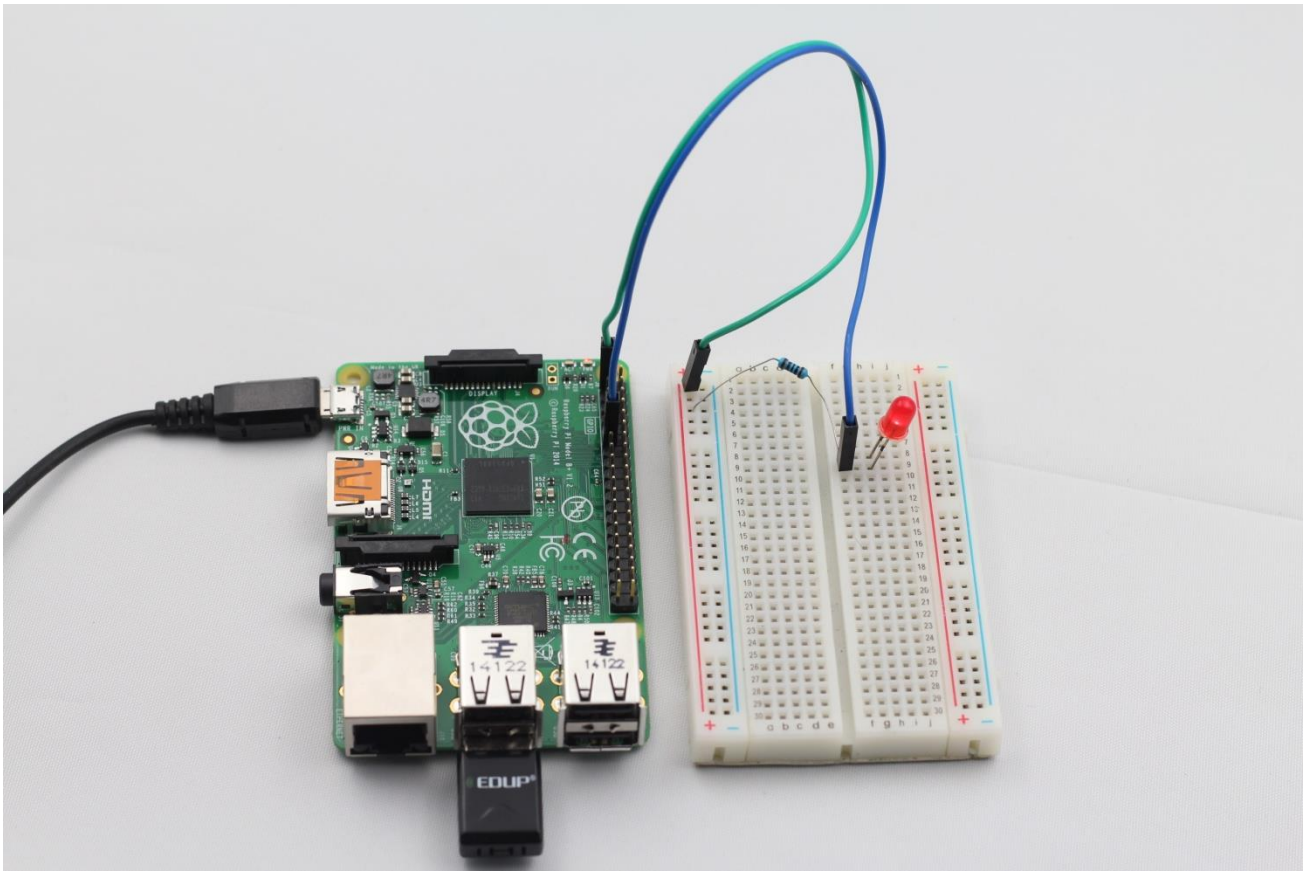
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/01_led_1.py

Step 3: Run

```
sudo python 01_led_1.py
```

Now, you should see the LED blinking.



Further Exploration

If you want the LED to blink faster, just change the delay time. For example, change the time to delay (200), recompile the code and run it, and then you will see the LED blinking 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 apply the Raspberry Pi GPIO0 to making an LED blink. Keep moving to the next contents.

SUNFOUNDER's Raspberry Pi Lesson 2 Controlling an LED by a Button

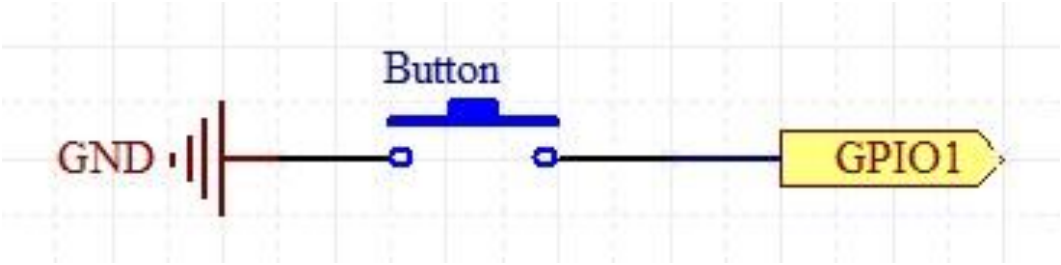
Introduction

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

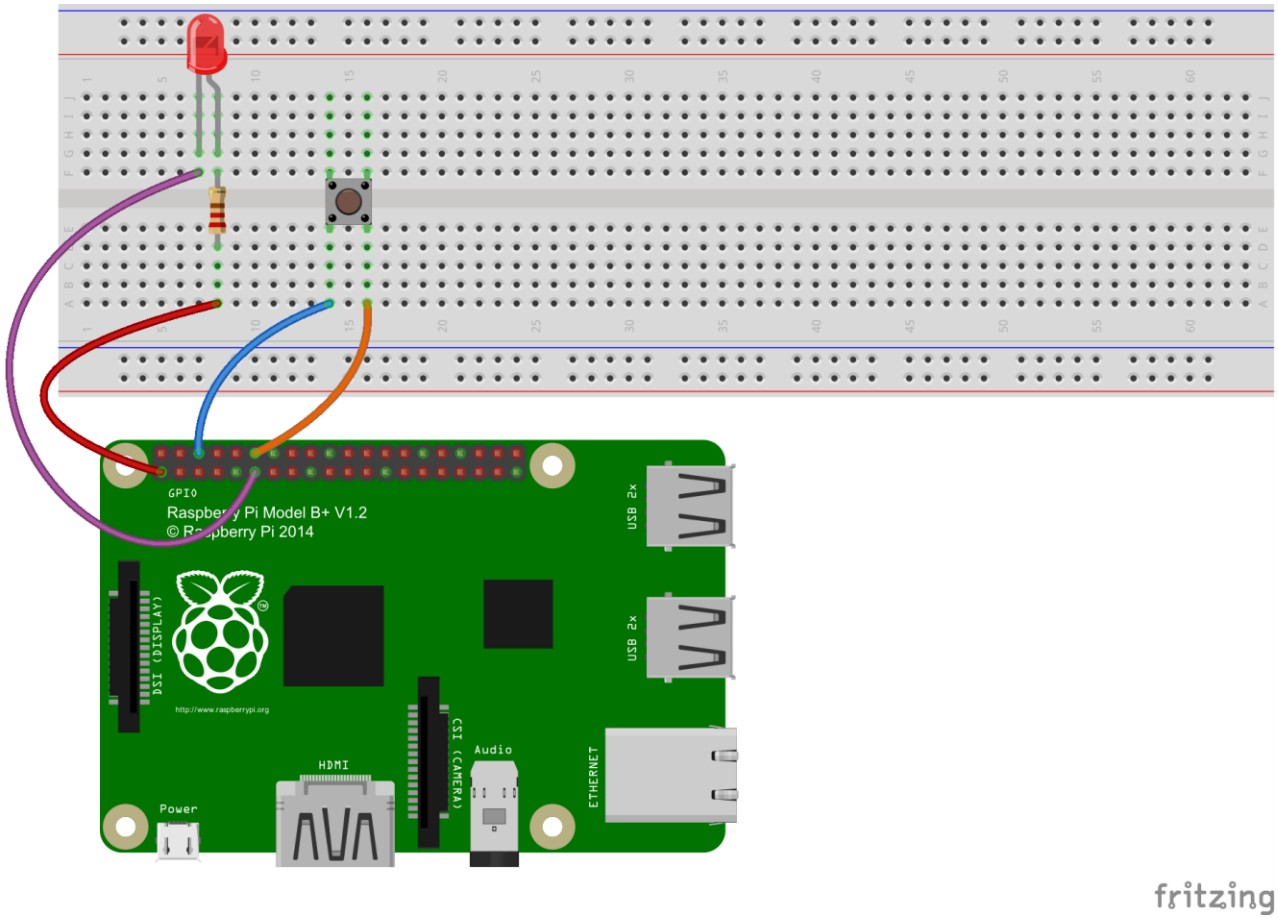
Principle



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: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/02_BtnAndLed/BtnAndLed.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./BtnAndLed
```

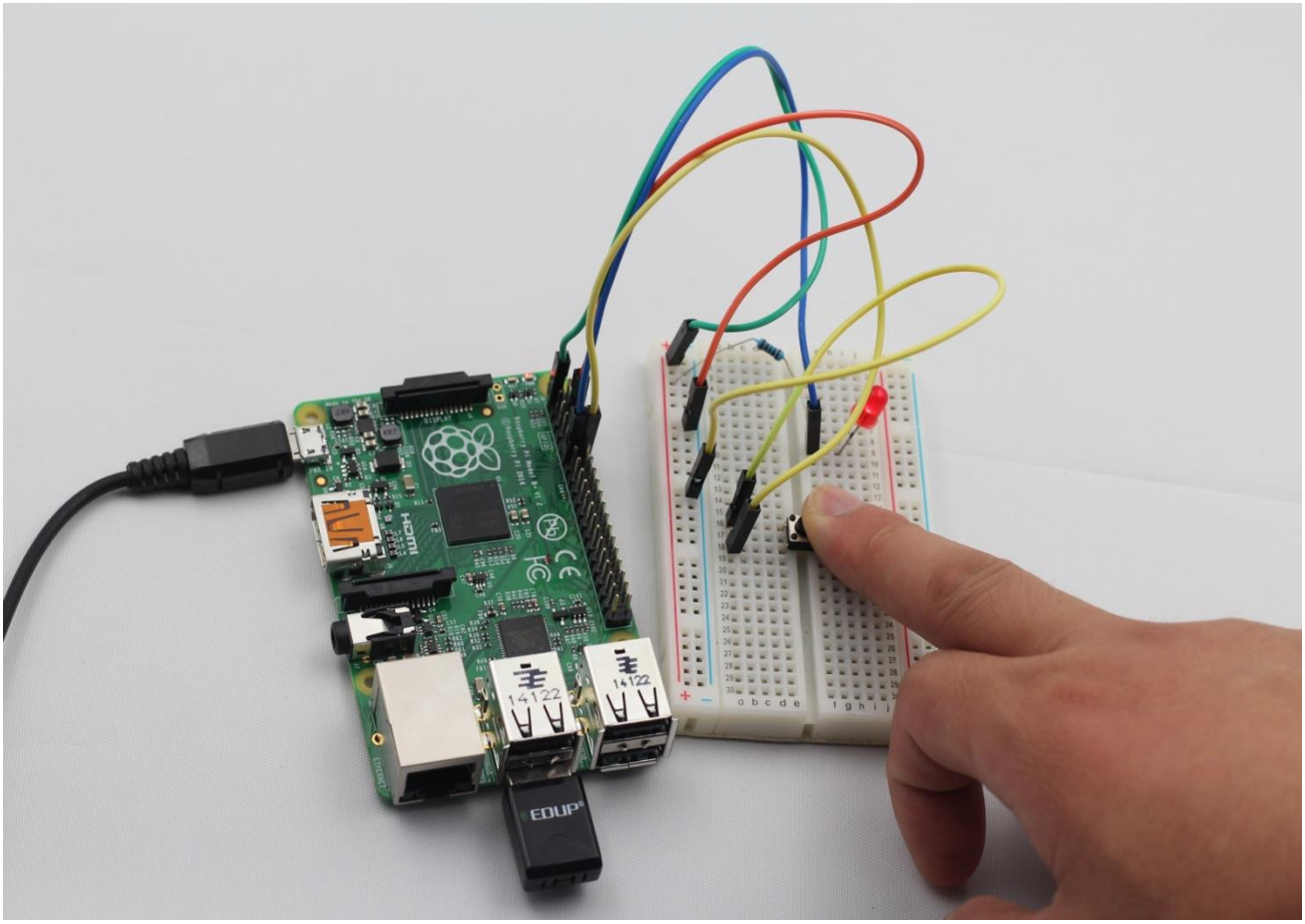
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code /02_btnAndLed_1.py

Step 3: Run

```
sudo python 02_btnAndLed_1.py
```

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



Summary

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

SUNFOUNDER's Raspberry Pi Lesson 3 Flowing LED Lights

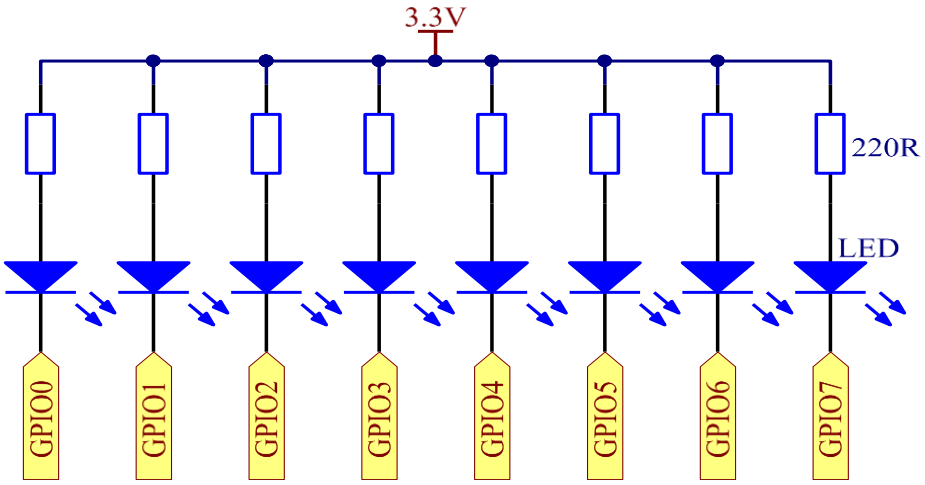
Introduction

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

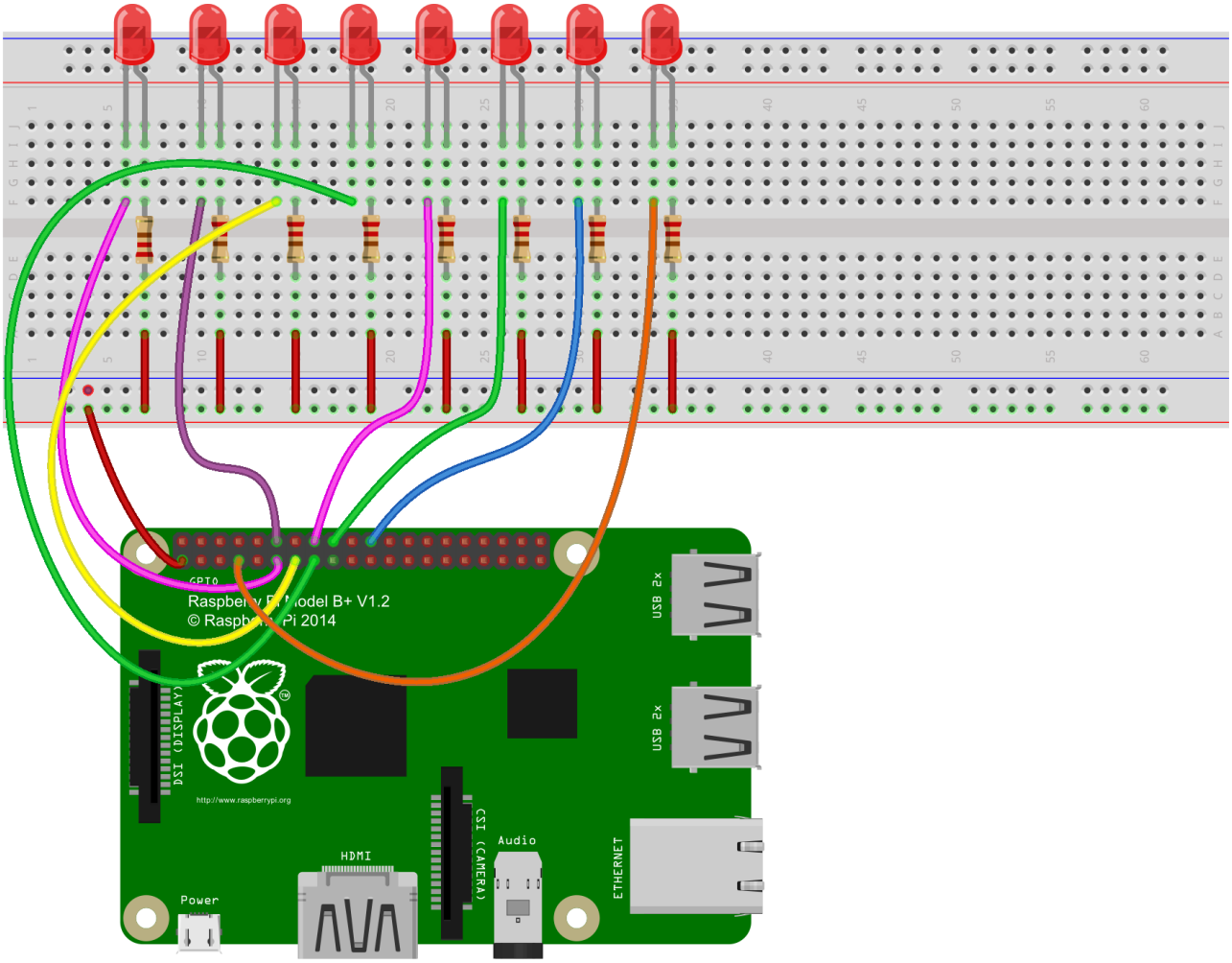
Principle



Set GPIO0-GPIO7 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: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/03_8Led/8Led.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./8Led
```

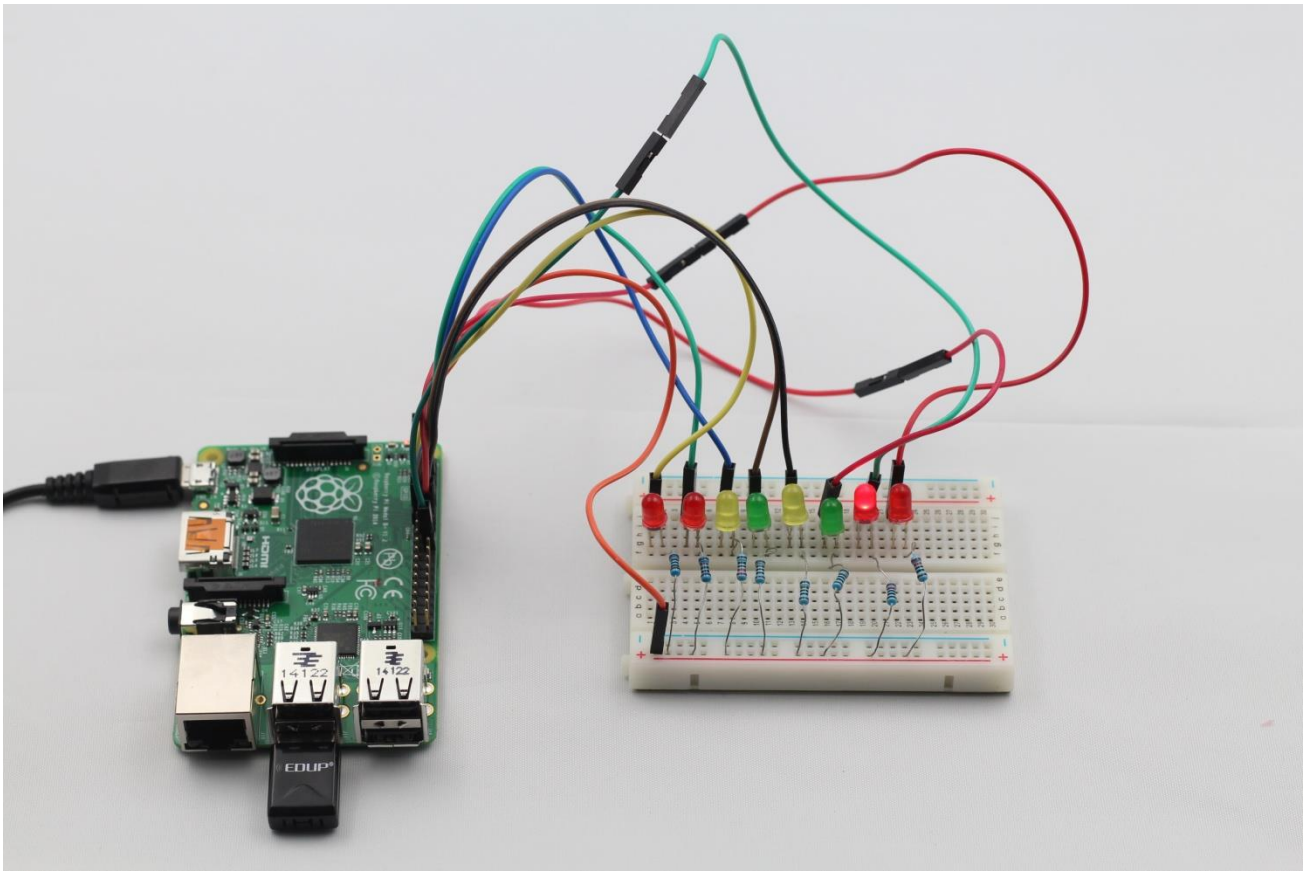
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/03_8Led.py

Step 3: Run

```
sudo python 03_8Led.py
```

Then you will see eight LEDs light up circularly and render different effects.



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.

SUNFOUNDER's Raspberry Pi Lesson 4 Breathing LED

Introduction

In this lesson, we will 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

Before we talk about PWM, let's have a look at the applications of PWM first. PWM has been successfully applied in motor speed regulation, steering angle control, light intensity control and signal output. For example, when PWM is applied to a horn, it will make a sound. After we know about its special functions, let's find out what PWM really is.

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 the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 3.3v controlling the brightness of the LED. (See the PWM description on the official website of Arduino)

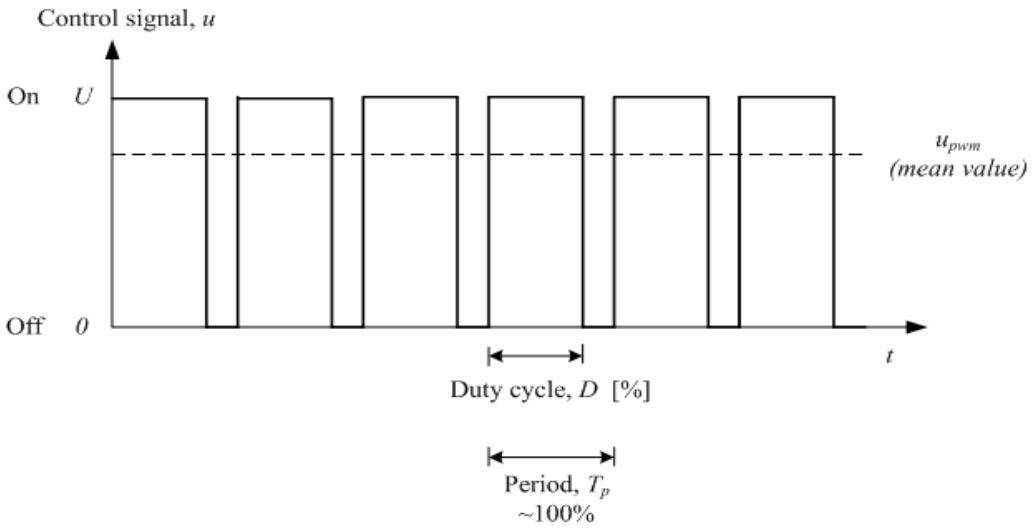
Duty Cycle

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:

$$D = \frac{T}{P} \times 100\%$$

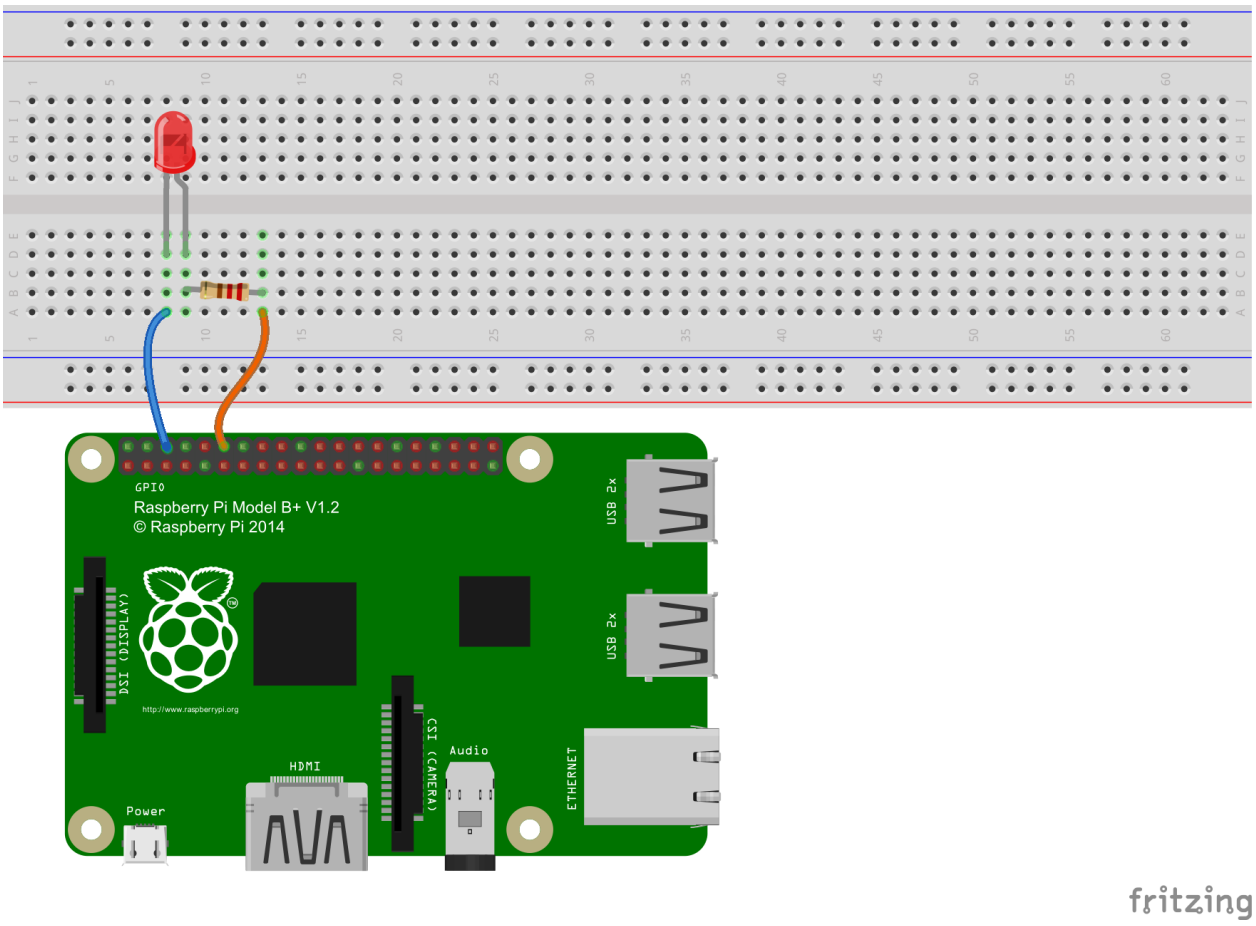
where D is the duty cycle, T is the time the signal is active, and P is the total period of the signal. Thus, a 60% duty cycle means the signal is on 60% of the time but off 40% of the time.

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.



Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/04_PwmLed/PwmLed.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./PwmLed
```

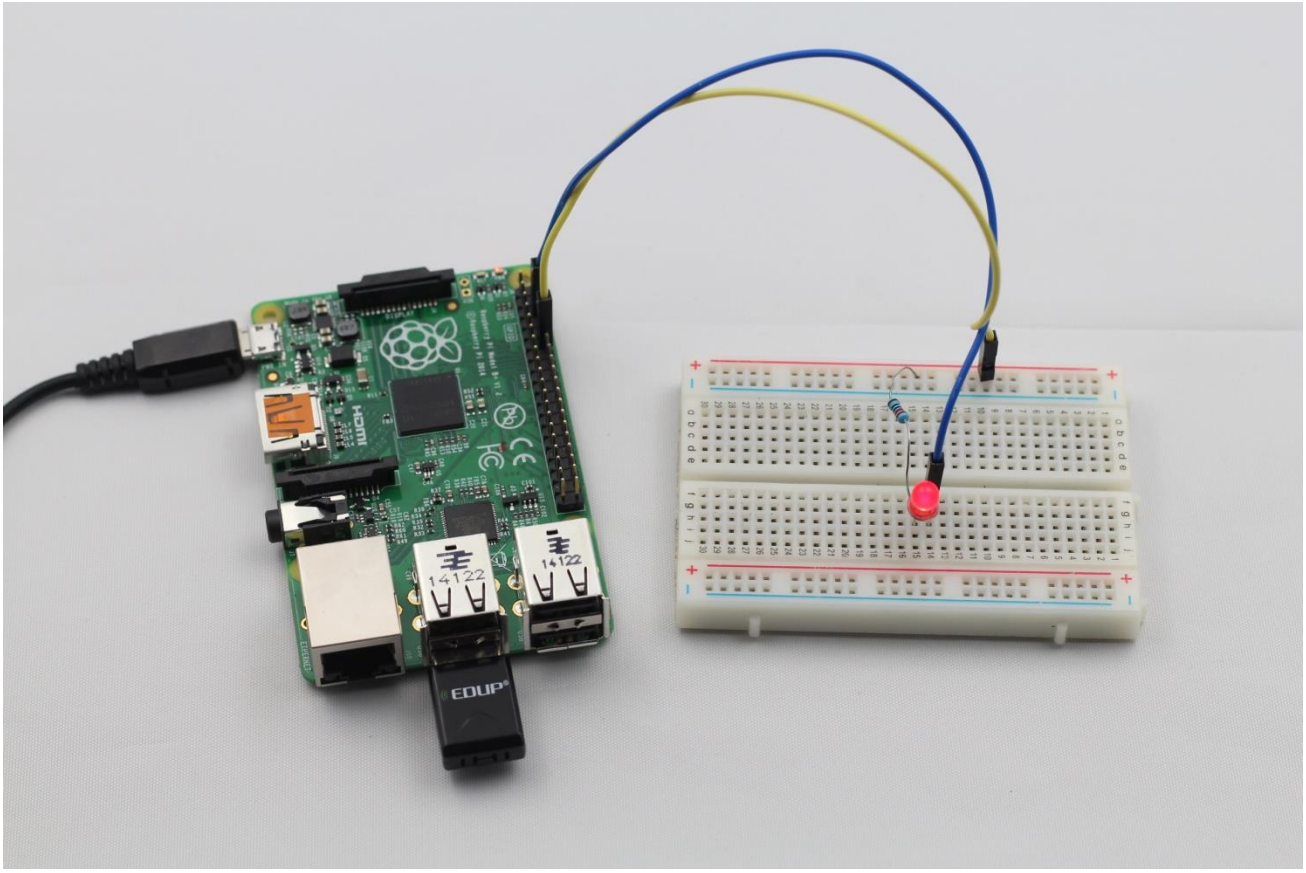
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/04_pwmLed.py

Step 3: Run

```
sudo python 04_pwmLed.py
```

Press Enter and you will see a gradual change of the LED luminance.



Summary

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

SUNFOUNDER's Raspberry Pi Lesson 5 RGB LED

Introduction

In this lesson, we will learn how to use RGB LEDs.

Components

- 1* Raspberry Pi
- 1* Breadboard
- 1* RGB LED
- 3* Resistor (220Ω)
- Several jumper wires

Principle

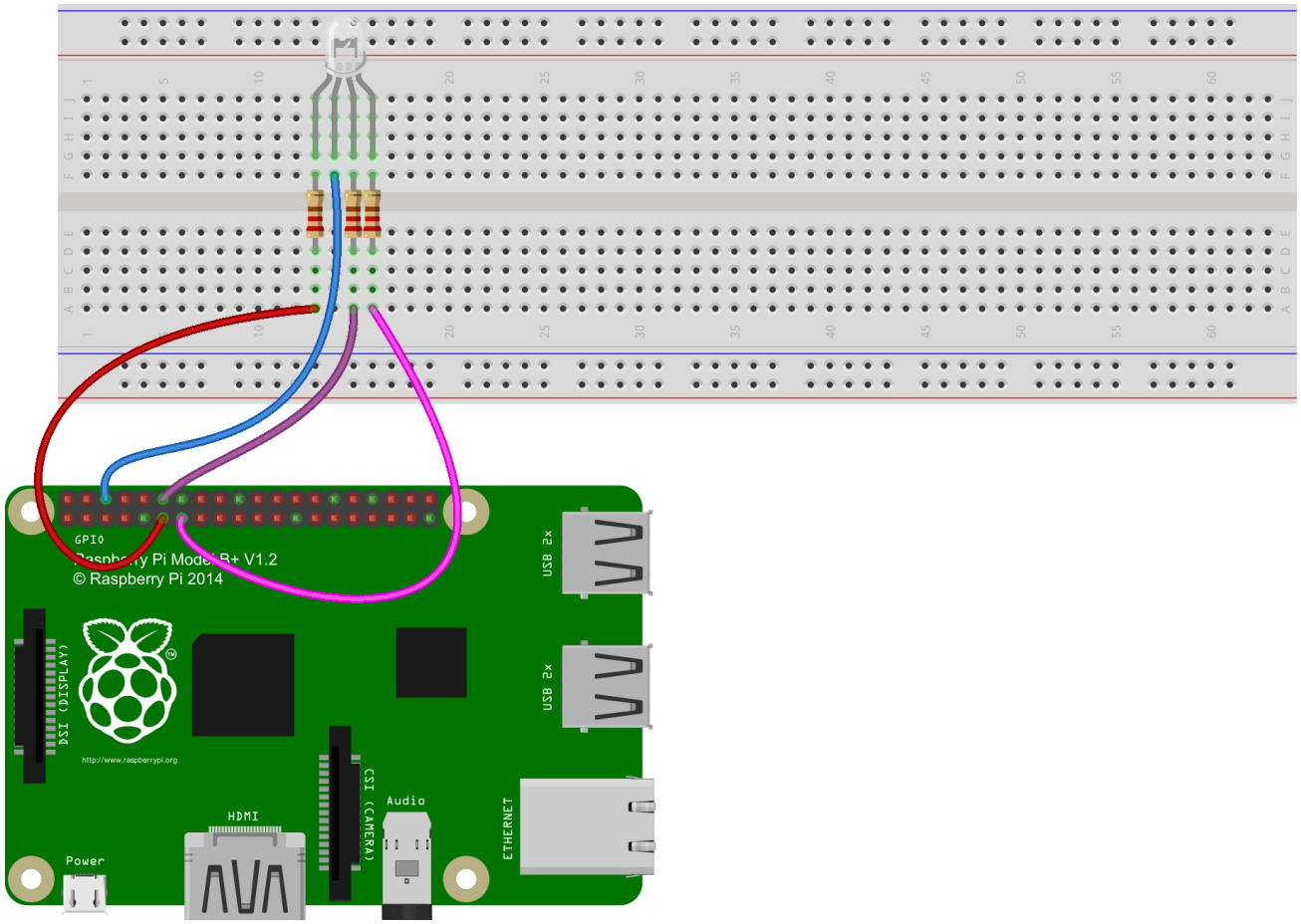
RGB LEDs emit light in various colors. They package three LEDs of red, green, and blue into a transparent or semitransparent plastic shell and have four pins. The three primary colors can be mixed into various colors by brightness. The brightness of LED can be adjusted with PWM. Raspberry Pi has only 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. Thus, you only need to include the header file softPwm.h (for C language users), and then call the API it provided to easily achieve multi-channel PWM output to control the RGB LED so as to display all kinds of colors.

There are two types of packages for RGB LEDs. One is patch type, and the other is dual-in-line type. Their difference is color resolution.

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

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/05_RGB/rgb.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./rgb
```

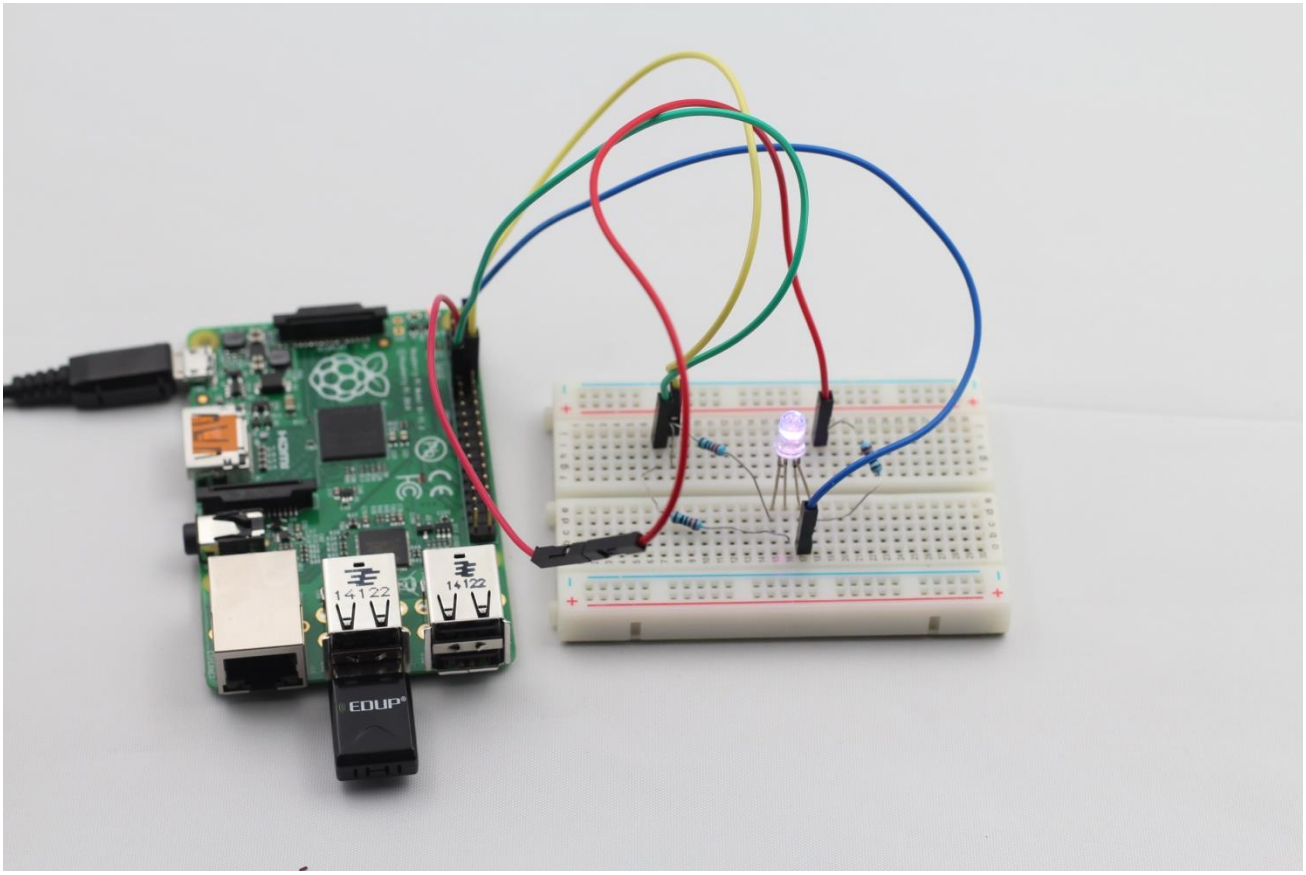
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/ 05_rgb.py

Step 3: Run

```
sudo python 05_rgb.py
```

Here you should see the RGB LED emitting light of different colors in turn.



Further Exploration

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

Experimental Summary

You have learnt how to control RGB LEDs with the softPwm of Raspberry Pi in this experiment. And you can continue to explore softPwm application to DC motor speed regulation.

SUNFOUNDER's Raspberry Pi Lesson 6 Buzzer

Introduction

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

Components

- 1* Raspberry Pi
- 1* Breadboard
- 1* Buzzer (Active)
- 1* PNP transistor (8550)
- 1* Resistor (1KΩ)
- Jumper wires

Principle

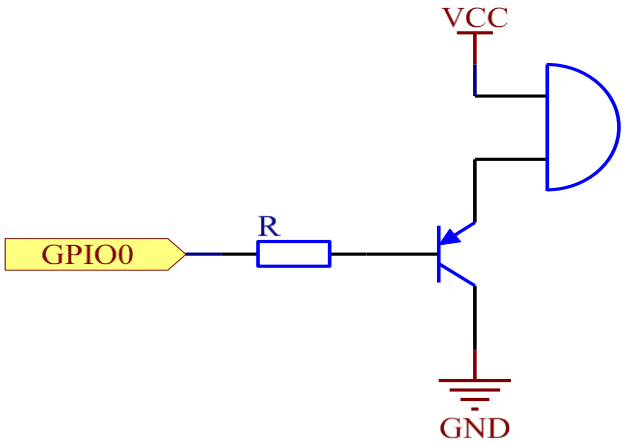
As a type of electronic buzzer with integrated structure, buzzers, which are supplied by DC power, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices. Buzzers can be categorized 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 tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

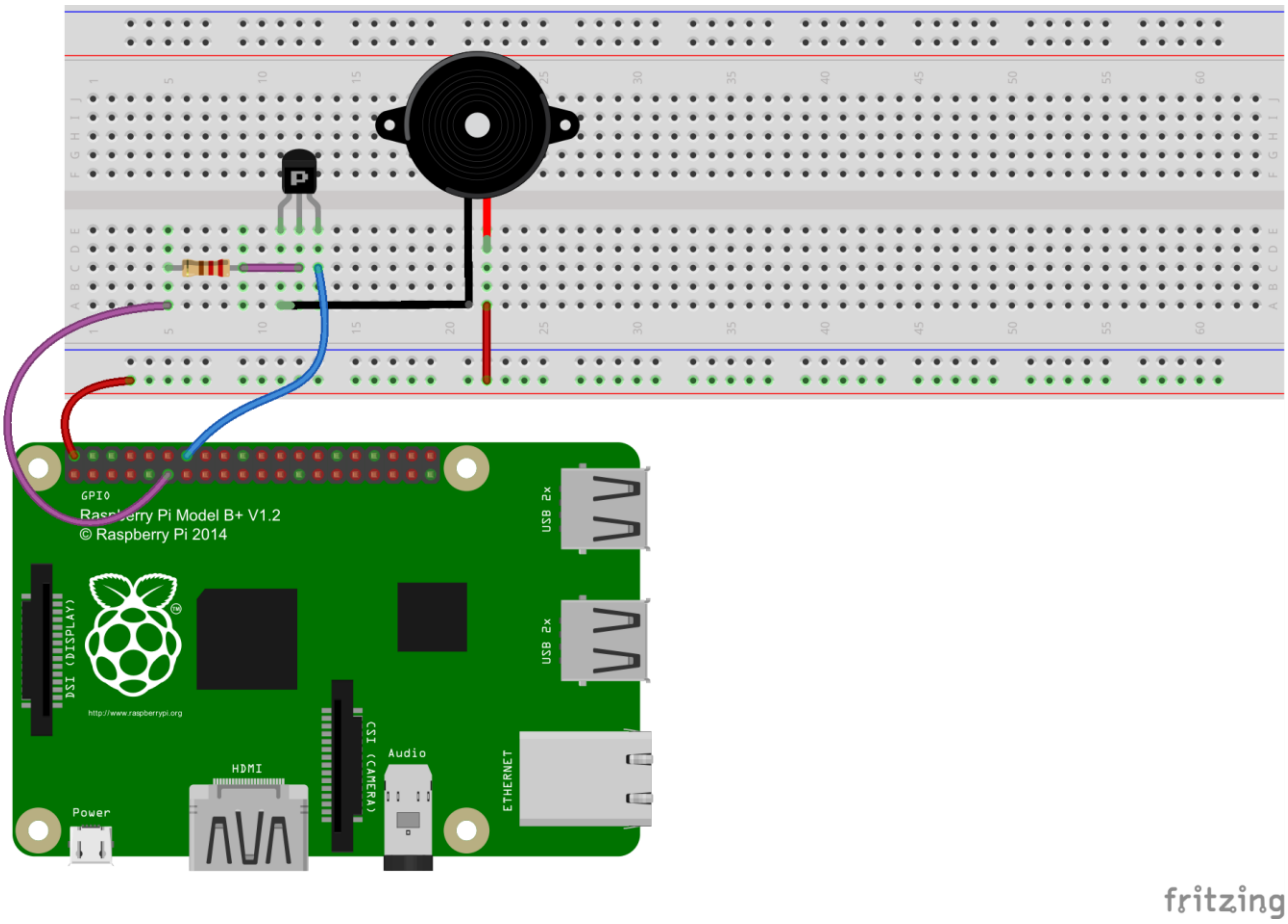
In this experiment, an active buzzer is used. When the GPIO of Raspberry Pi output is supplied with low level (0V) by programming, the transistor will conduct because of current saturation

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: Connect the circuit as shown in the following diagram (Pay attention to the positive and negative poles of the buzzer)



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/06_Beep/beep.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./beep
```

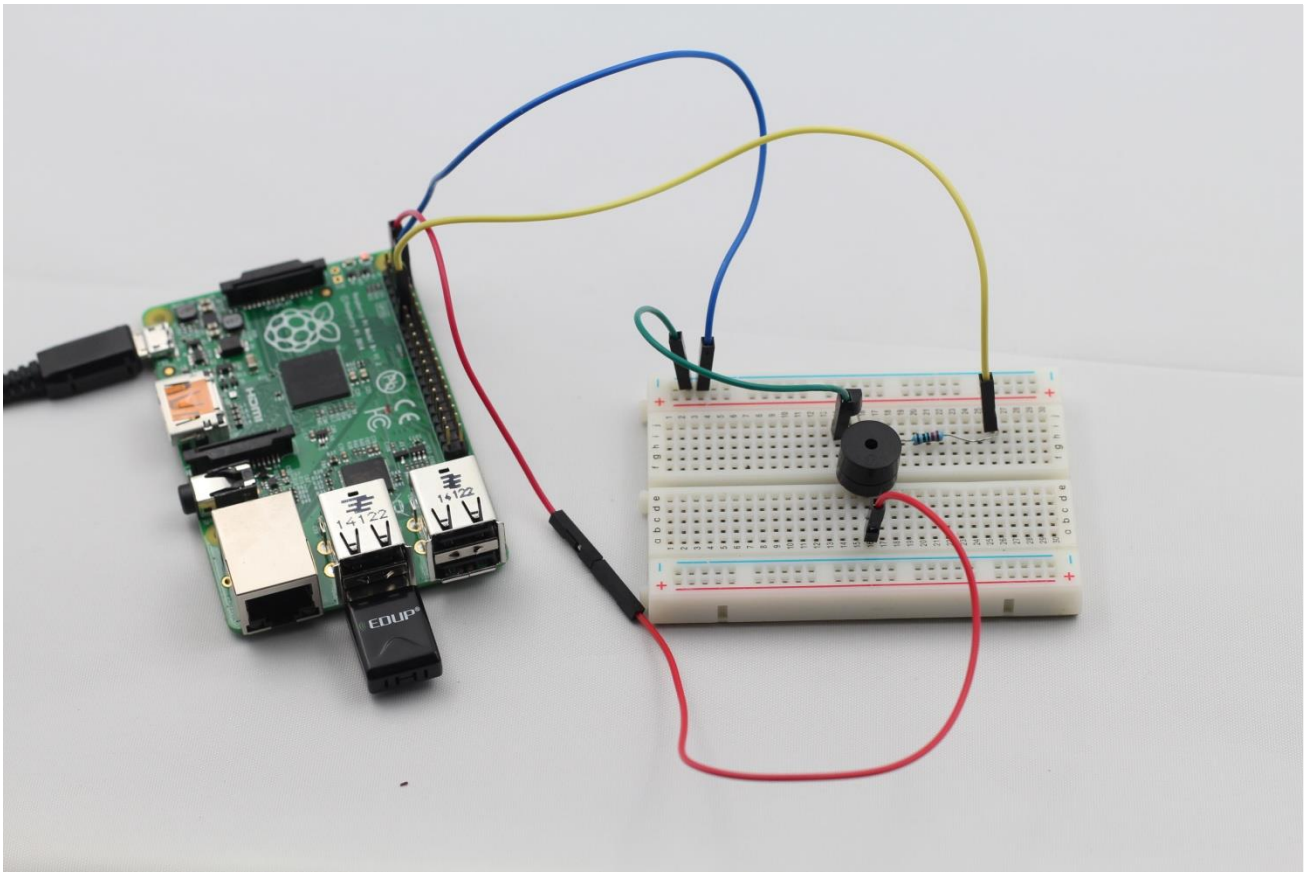
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/06_beep.py

Step 3: Run

```
sudo python 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 some basic knowledge of programming.

SUNFOUNDER's Raspberry Pi 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.

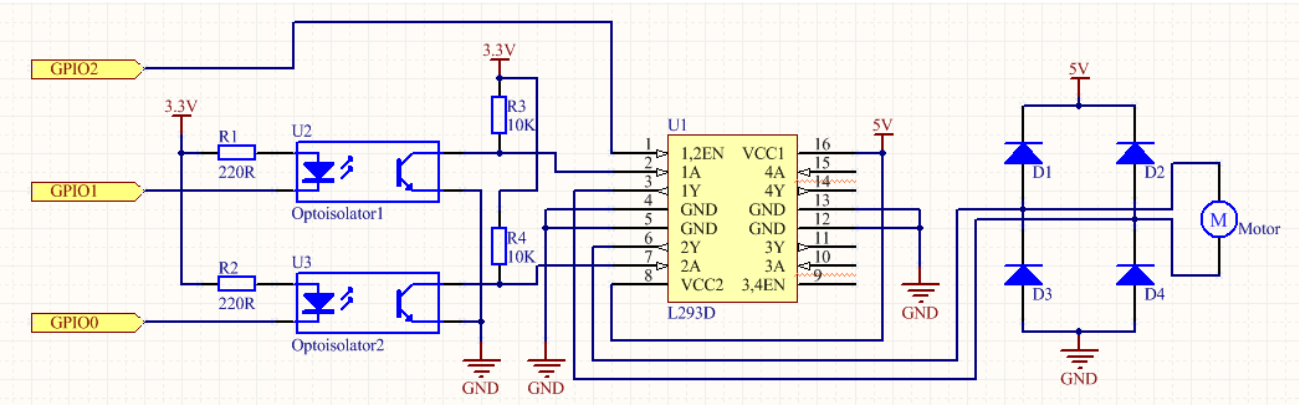
Components

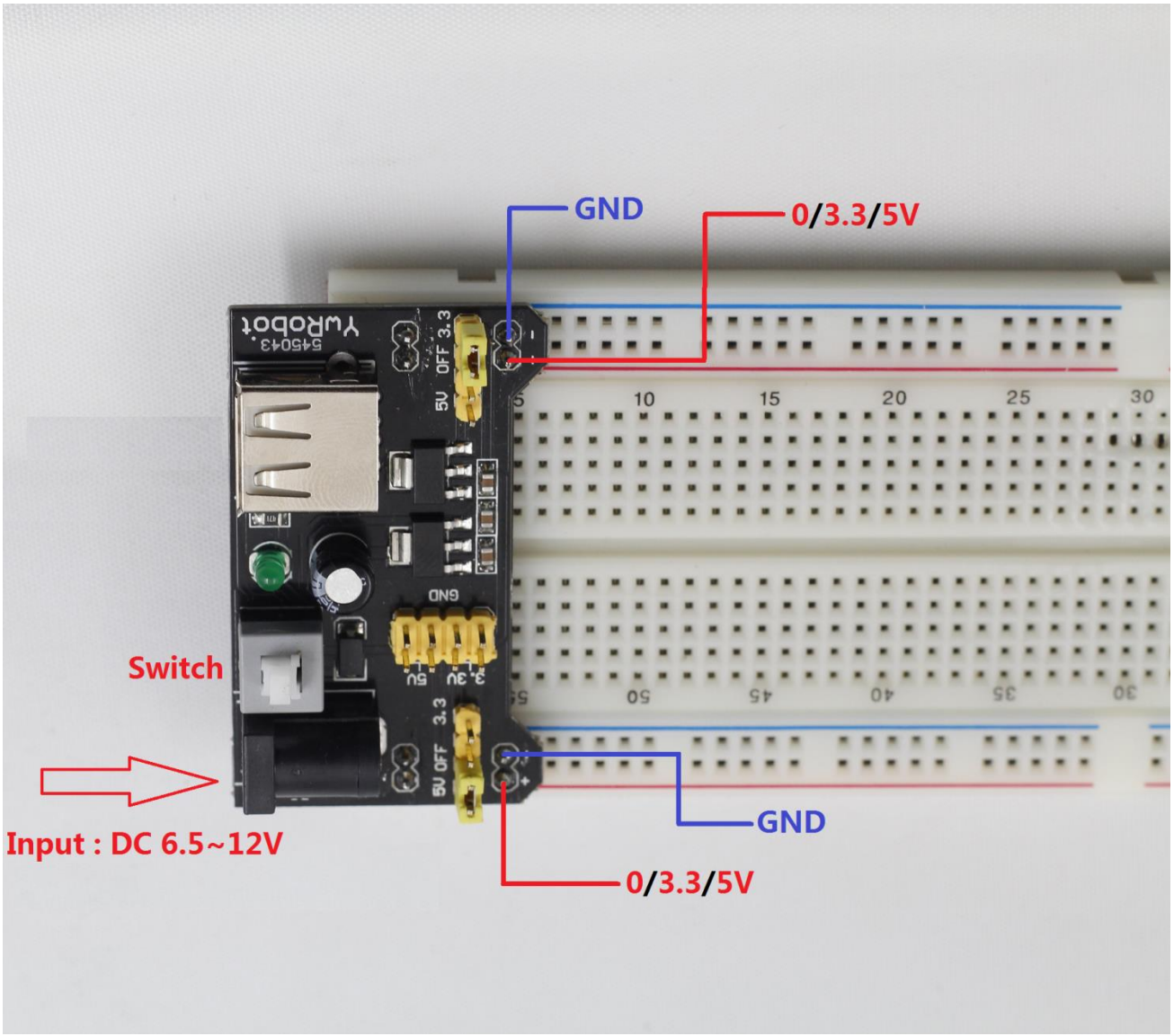
- 1* Raspberry Pi
- 1* Breadboard
- 1* L293D
- 1* DC motor
- 1* Power Module
- 2* Optocoupler (4N35)
- 4* Diode (1N4007)
- 4* Resistor (2*220Ω, 2*10K)
- Jumper wires

Principle

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.

Connect the two IOs of the Raspberry Pi to two optocouplers which isolate the power supply to protect your Raspberry Pi from interference and crashing when the motor works. Connect the output port of the optocoupler to the driver IC (L293D) of the motor. When one of the two IOs of the Raspberry Pi is at high level and the other is low, the motor will rotate clockwise or counterclockwise. By adjusting the pulse width, you can realize DC motor speed regulation as well. A motor is an inductive device. Its four diodes play an important role in freewheeling and suppress the surge voltage of motor coils to prevent irreversible damages to driver ICs.



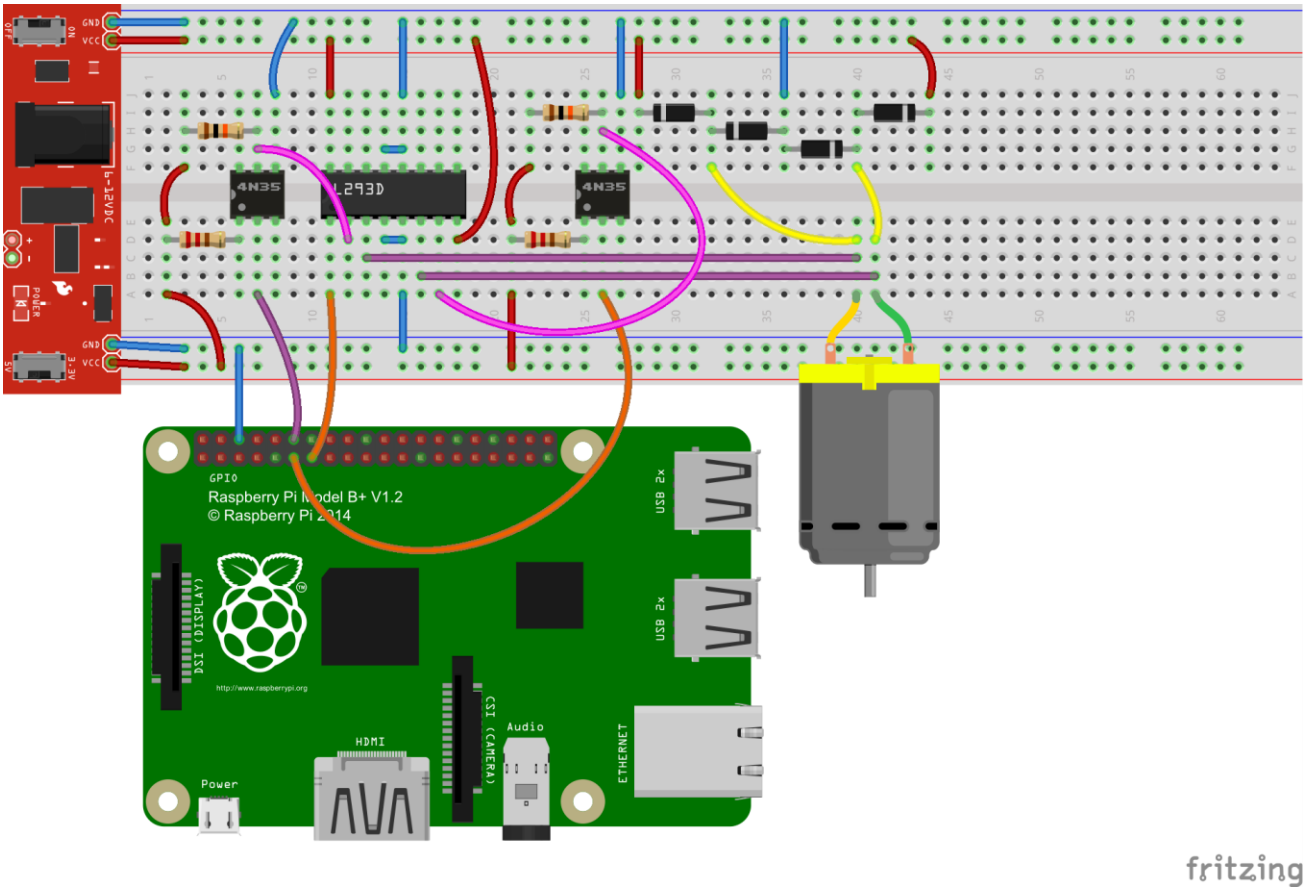


The maximum current allowed for this power supply module is 700mA. You can switch the output voltage (0/3.3/5V) through a yellow jumper cap.

In this experiment, it needs large current 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.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/07_Motor/motor.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./motor
```

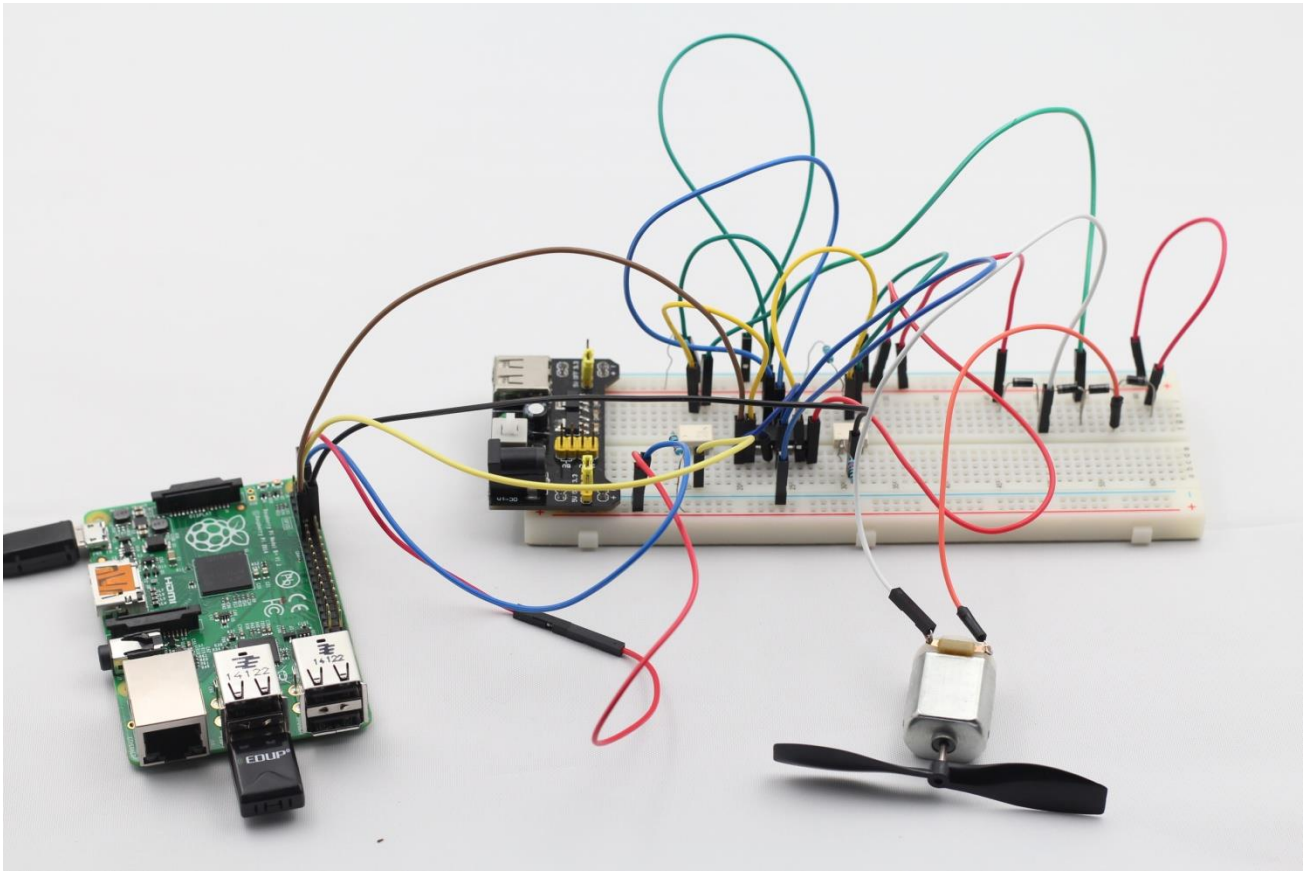
For Python users:

Step 2: See path/RPi_SuperKit_Code/Pyhton_code/07_motor.py

Step 3: Run

```
sudo python 07_motor.py
```

Now, you should see the motor rotating.



Further Exploration

You can use buttons to control the clockwise and counterclockwise rotation of the motor based on the previous contents. Also you can use PWM principle 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.

SUNFOUNDER's Raspberry Pi Lesson 8 Rotary Encoder

Introduction

In this lesson, we will learn how to use rotary encoders.

Components

- 1* Raspberry Pi
- 1* Breadboard
- 1* Rotary Encoder Module
- Jumper wires

Principle

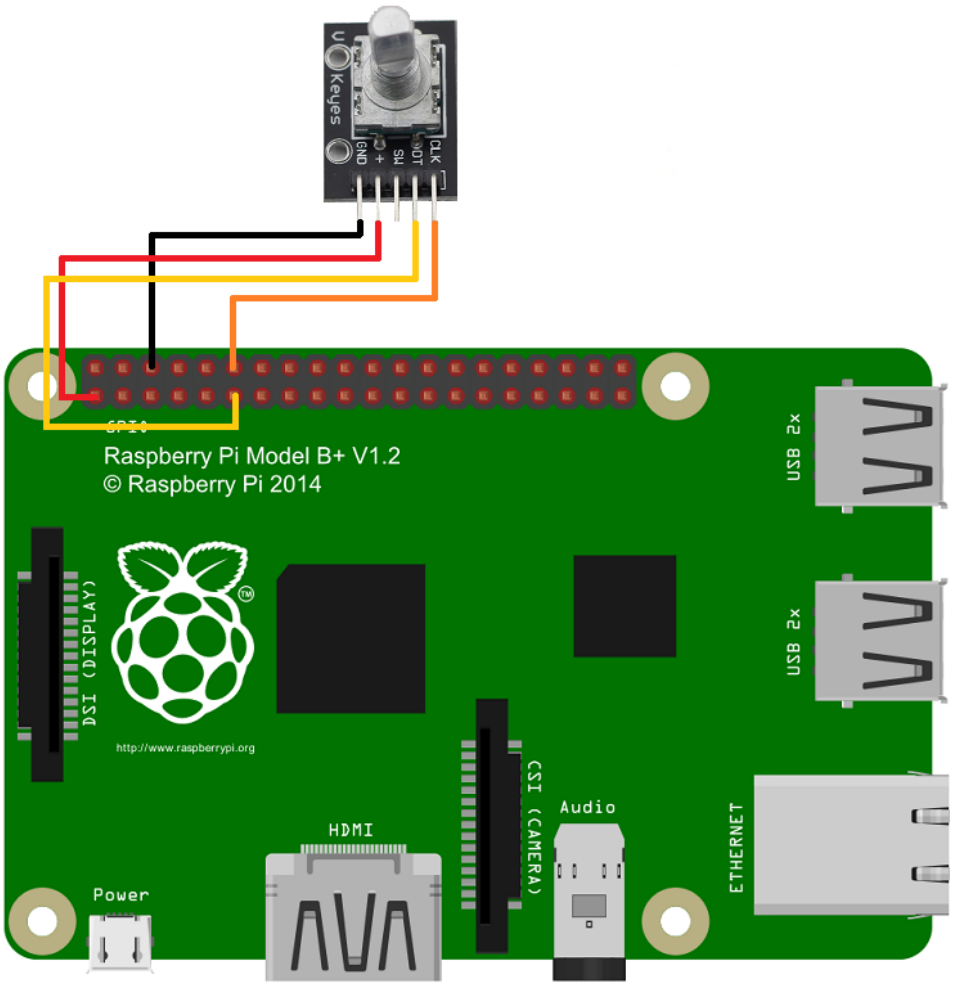
A rotary encoder is an electronic switch with a set of regular pulses with strictly timing sequence. When used with IC, it can achieve increment, decrement, page turning and other operations such as mouse scrolling, menu selection, acoustic sound regulation, frequency regulation, toaster temperature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

Most rotary encoders have 5 pins with three functions of turning left, turning right and pressing down. Pin 4 and pin 5 are switch wiring terminals for pressing down. They are no different from the buttons mentioned previously, so no more details will be provided here. Pin 2 is generally connected to ground. Pin 1 and pin 3 are first connected to a pull-up resistor and then to a microprocessor. In this experiment, they are connected to GPIO0 and GPIO1 of Raspberry Pi. When the encoder rotates 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

Step1: Connect the circuit based on the following method.



Raspberry Pi	Rotary Encoder Module
3.3V	'+'
GND	GND
GPIO0	DT
GPIO1	CLK

For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/08_RotaryEncoder /rotaryEncoder.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./rotaryEncoder
```

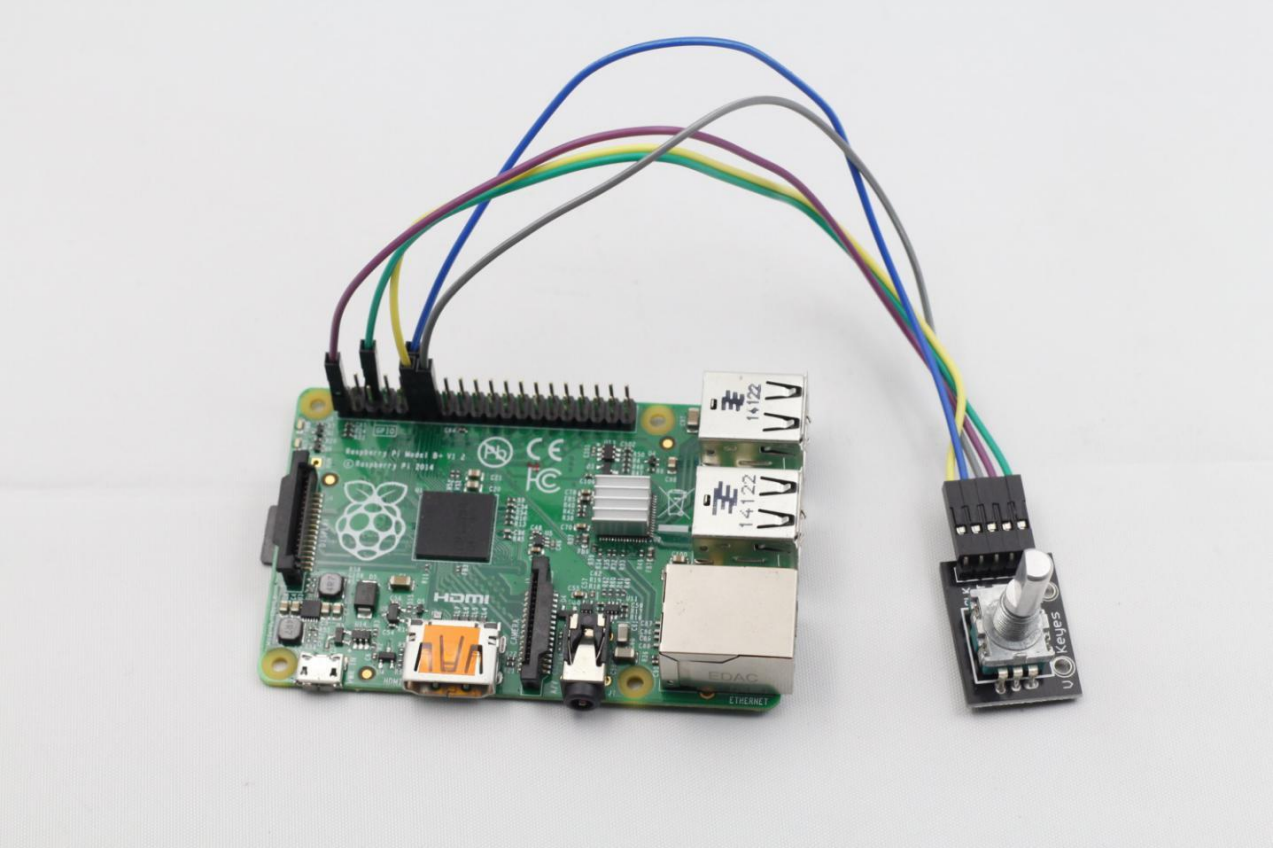
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/08_rotaryEncoder.py

Step 3: Run

```
sudo python 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. You can try this function by yourself. That is, when you press the rotary encoder, the value of the variable will be reset.

SUNFOUNDER's Raspberry Pi Lesson 9 555 Timer

Introduction

In this lesson, we will learn how to use the 555 timer.

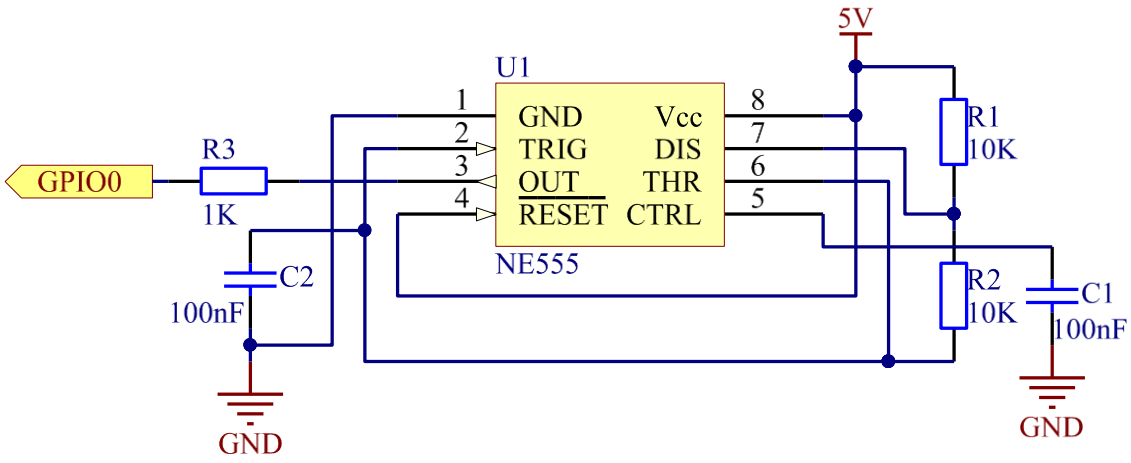
Components

- 1* Raspberry Pi
- 1* Breadboard
- 1* NE555
- 3* Resistor (1*1KΩ, 2*10KΩ)
- 2* Capacitor (100nF)
- Jumper wires

Principle

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.

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



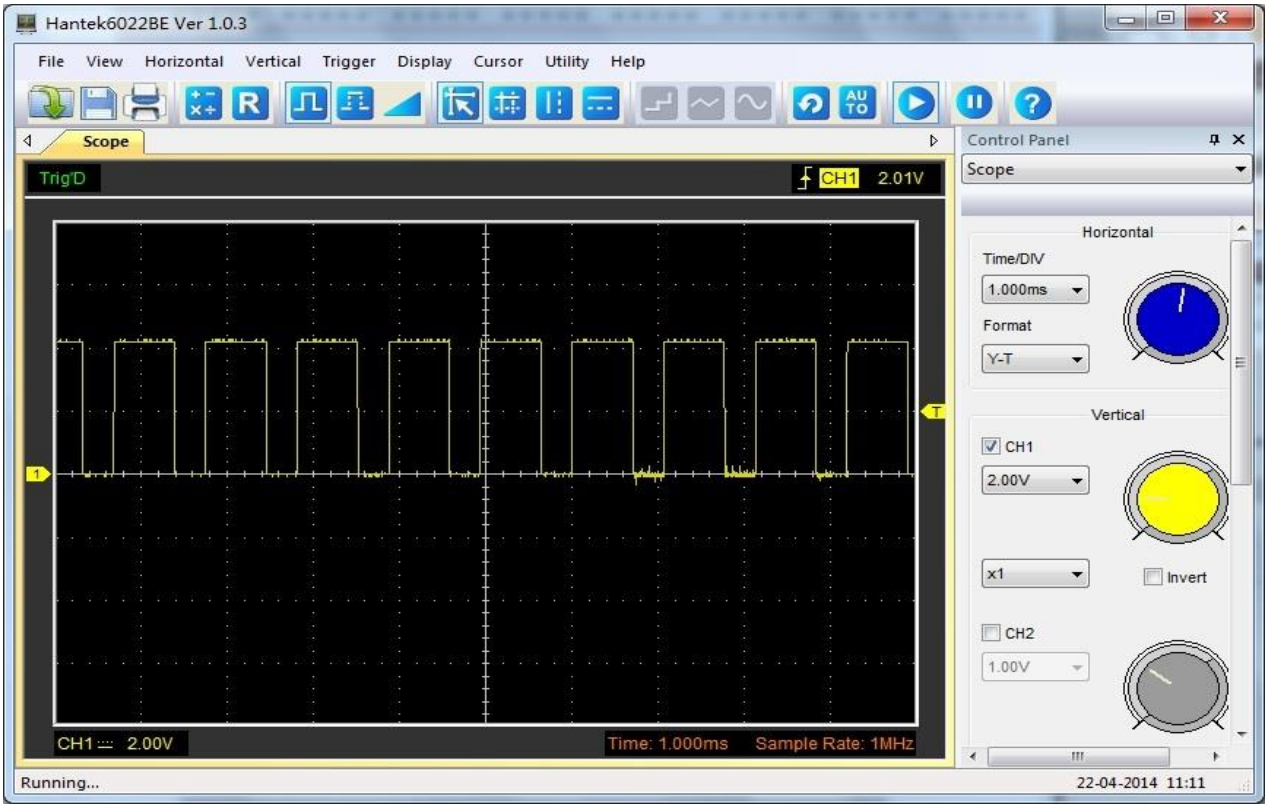
Under the astable mode, the frequency of output waveform of 555 timer is determined by R_1, R_2 and C_2:

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

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

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

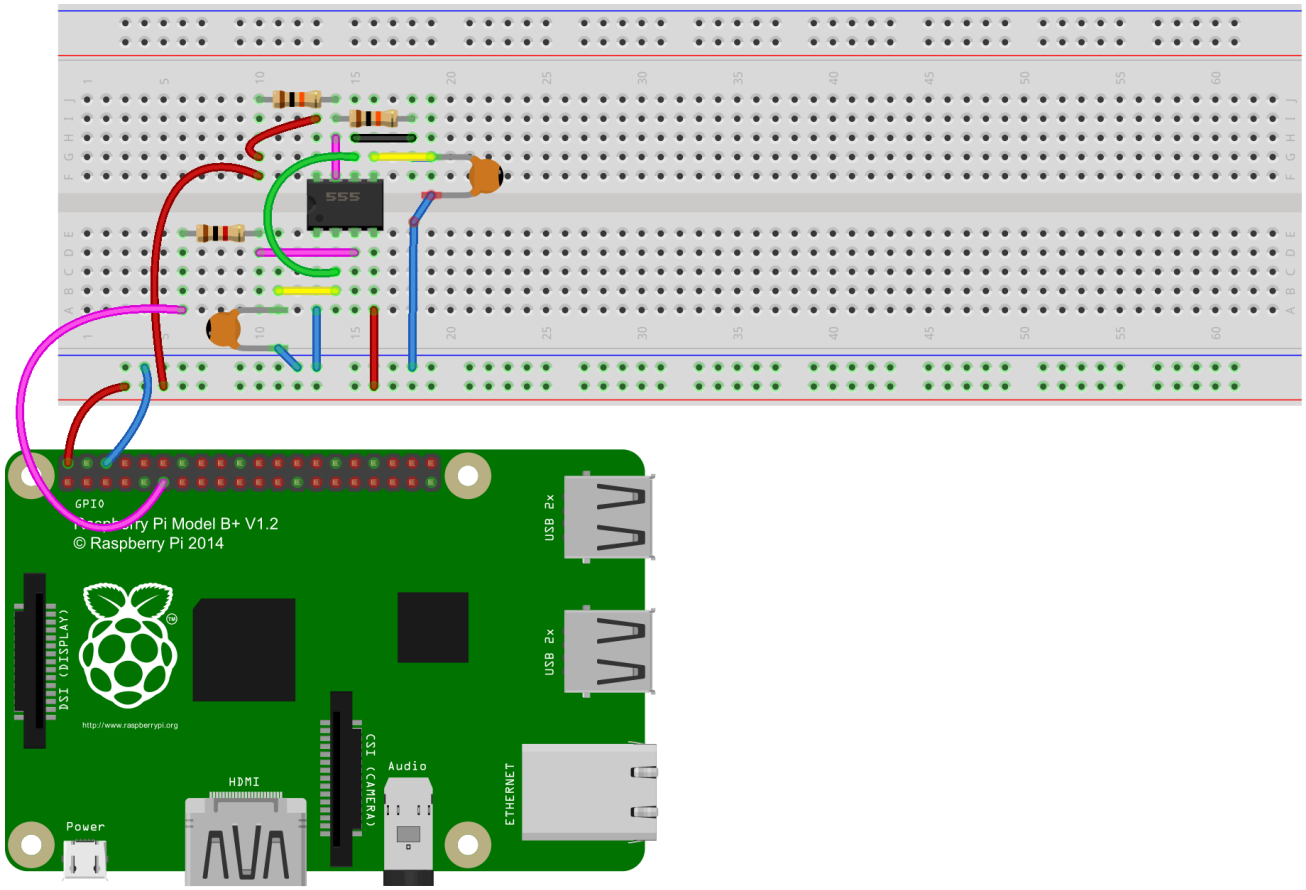
After connecting the circuit according to the schematic diagram, use an oscilloscope to observe the frequency of the output waveform. We can see it is consistent with the above calculated result.



Attach the output pin (e.g. pin 3) of 555 timer to GPIO0 of Raspberry Pi, configure GPIO0 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

Step1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/09_Timer555/timer555.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./timer555
```

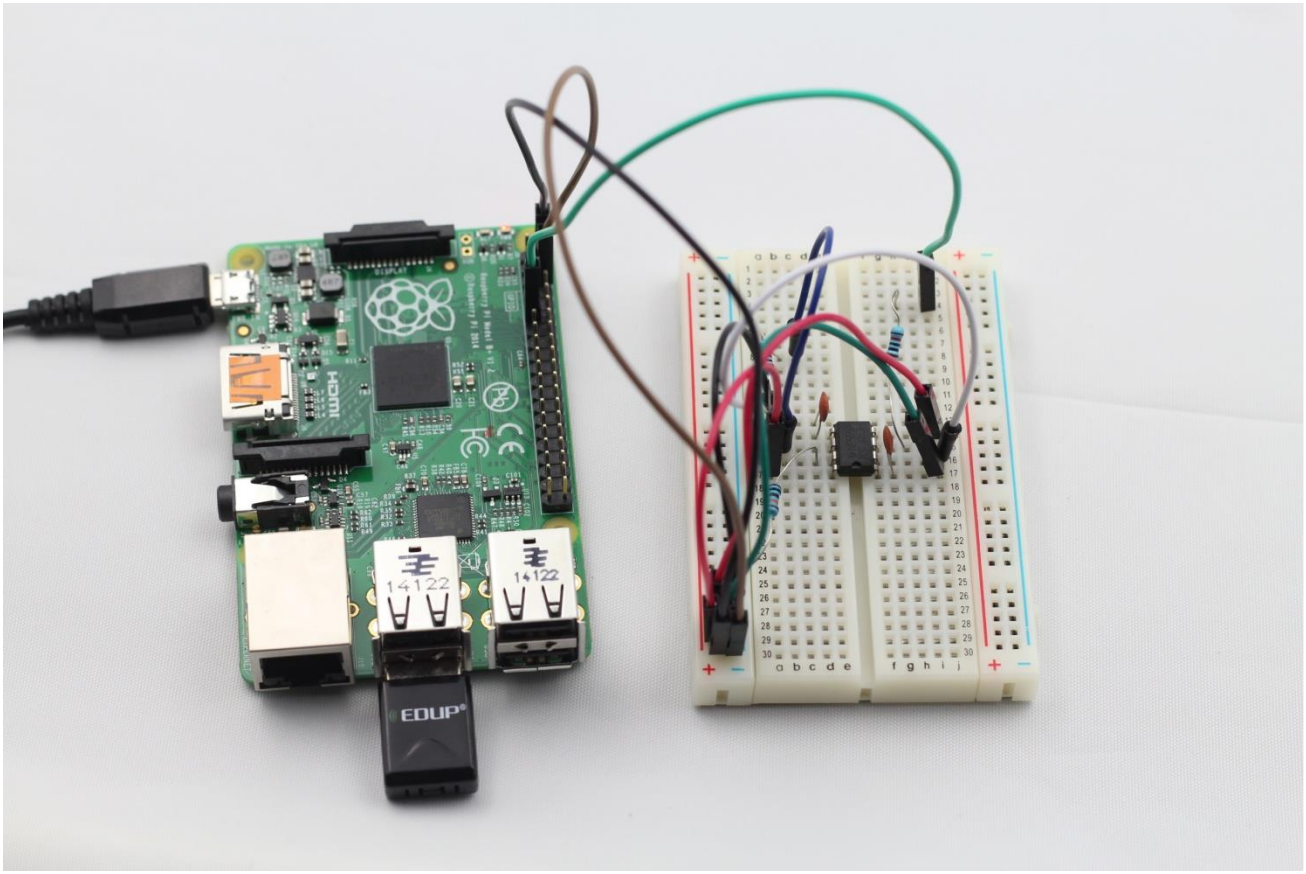
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/09_timer555.py

Step 3: Run

```
sudo python 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.



Further Exploration

The above 555 timer circuit outputs square waves with constant frequency and duty cycle. You can simply change this circuit to adjust the frequency and duty cycle. Thus you can make breathing LED described in Lesson 4 without Raspberry Pi. Now get some hands-on experience.

Summary

Through this lesson, you have mastered the basic knowledge and functions of 555 timer as well as the interrupt programming of Raspberry Pi. In fact, the 555 timer has many other interesting functions. Just explore it if you want to learn more.

SUNFOUNDER's Raspberry Pi Lesson 10 Driving LEDs by 74HC595

Introduction

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

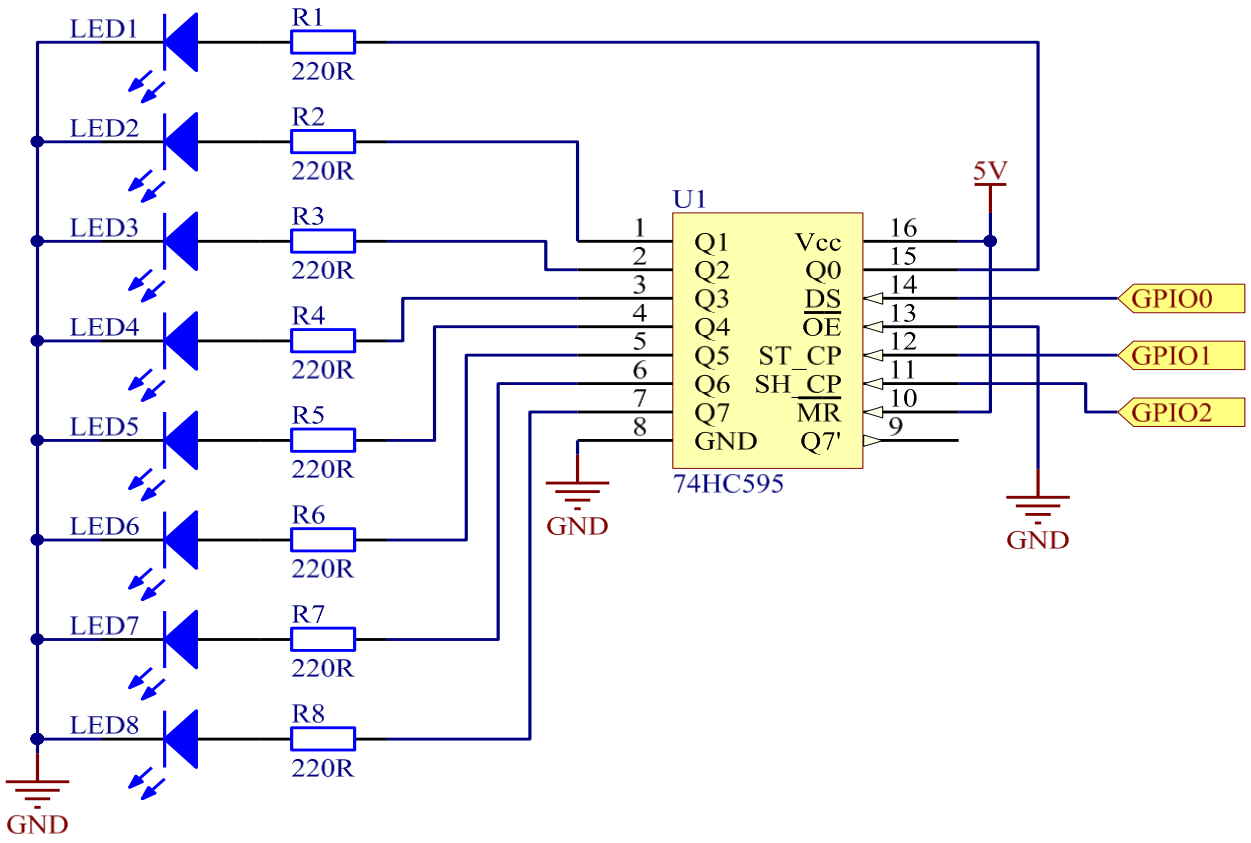
Components

- 1* Raspberry Pi
- 1* Breadboard
- 1* 74HC595
- 8* LED
- 8* Resistor (220Ω)
- Jumper wires

Principle

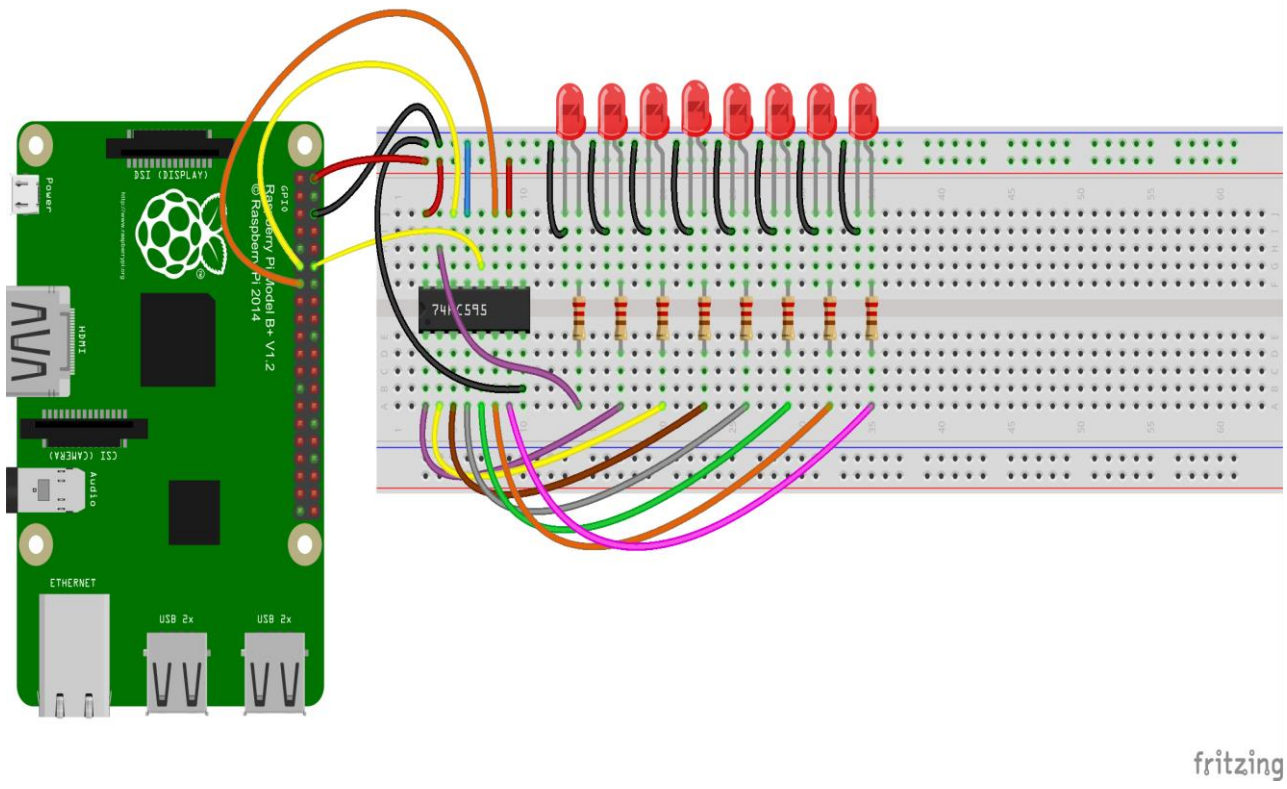
74HC595 is a silicon CMOS device, which has an 8-bit shift register and a memory with three-state output function. Compatible with low voltage TTL circuit, 74HC595 can transform serial input of 8-bit data into parallel output of 8-bit data. So it is often used to extend GPIO for embedded system and drive low power devices.

In this experiment, we attach ST_CP to Raspberry Pi GPIO1, SH_CP to GPIO2, and DS to GPIO0. 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: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/10_74HC595_LED/ 74HC595_LED.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./74HC595_LED
```

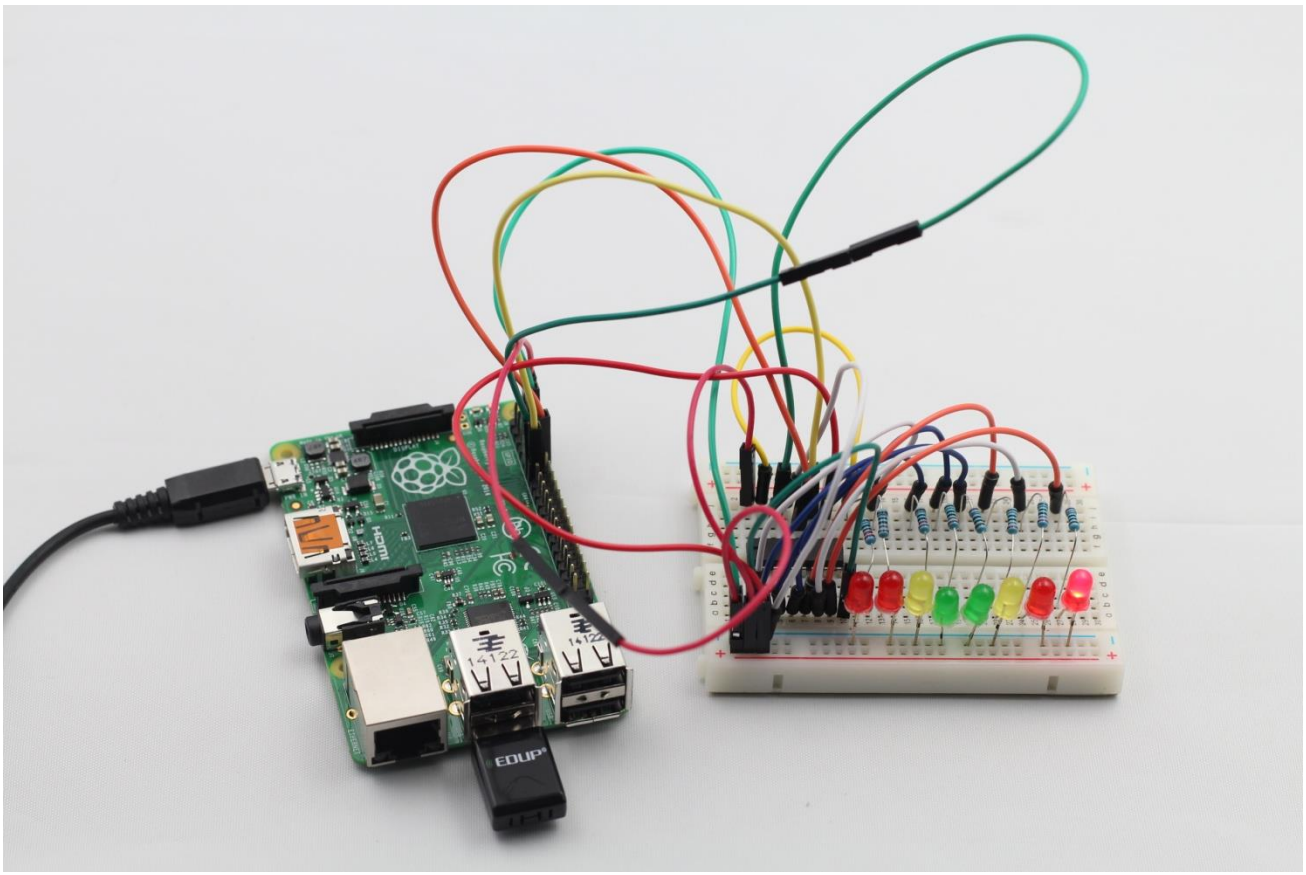
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/10_74HC595_LED.py

Step 3: Run

```
sudo python 10_74HC595_LED.py
```

Here you should see eight LEDs blinking regularly, and realistic effect is shown as follow:



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 like 3 Raspberry Pi IOs to control more peripherals. This function will be explained in subsequent lessons.

SUNFOUNDER's Raspberry Pi Lesson 11 Driving 7-Segment Display by 74HC595

Introduction

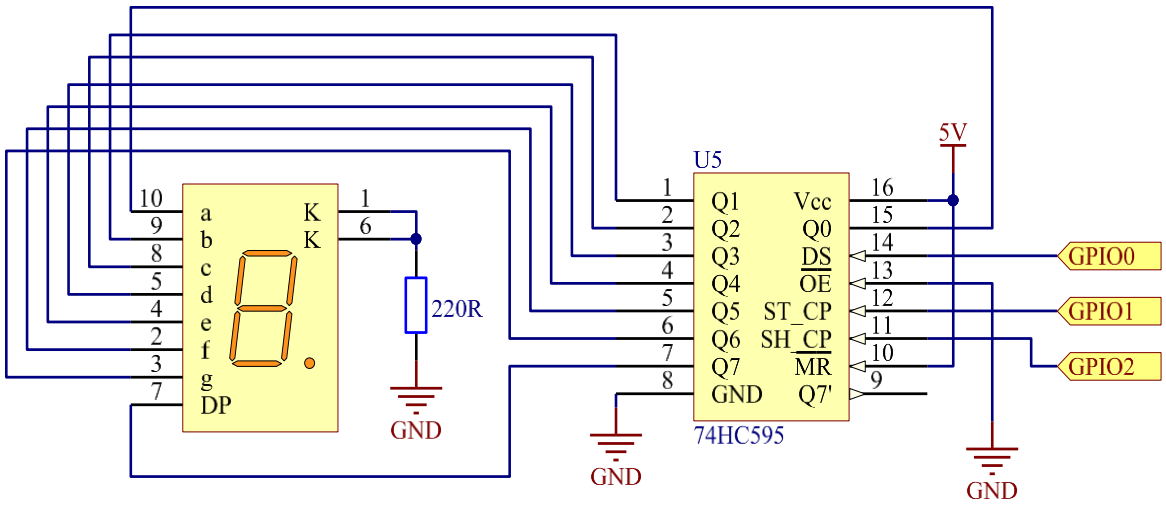
In this lesson, we will learn how to use 74HC595 to drive a 7-segment display to cycle a figure from 0 to 9.

Components

- 1* Raspberry Pi
- 1* Breadboard
- 1* 74HC595
- 1* 7-segment display
- 1* Resistor (1KΩ)
- Jumper wires

Principle

A 7-segment display is an 8-shaped component which packages many LEDs. Its leads are connected internally. External pins are its segments and common electrodes. The segments are respectively labeled by letter a, b, c, d, e, f, g, and dp. 7-segment displays can be categorized into two types: common cathode and common anode, depending on different light-emitting diode connections. They are widely used in electronic appliances especially home appliances such as air conditionings, water heaters, refrigerators and so on, because of its low price, ease of use, high brightness and long service life. Most temperature display units of water heater are 7-segment displays.

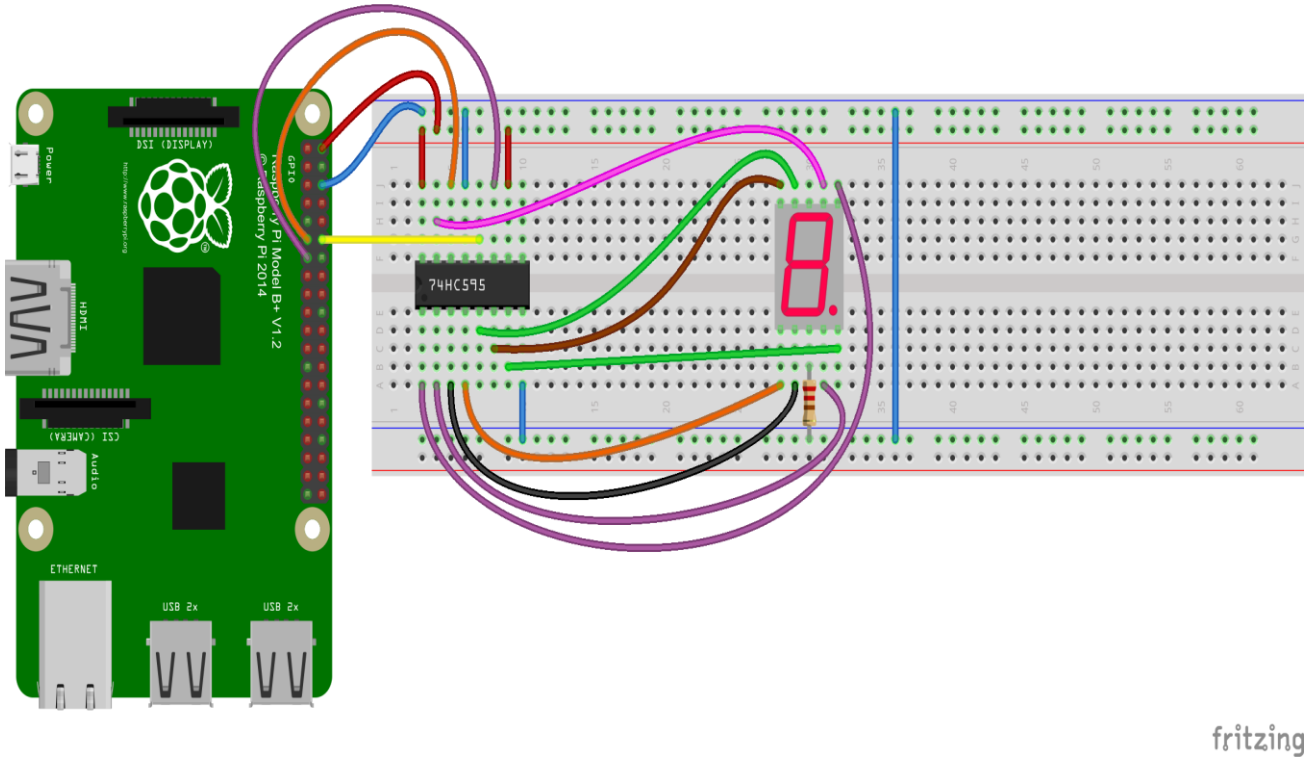


In this experiment, a common cathode 7-segment display is used, which connects all the cathodes of LEDs together to form a common cathode (COM) electrode. It should be connected to ground. When the anode of an LED in a certain segment is at high level, the corresponding segment will light up; when it is at low, the segment will not light up.

We connect pin ST_CP of 74HC595 to Raspberry Pi GPIO1, SH_CP to GPIO2, DS to GPIO0, 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 shift register) is at the rising edge and to memory register when ST_CP (the clock input of memory) is at the rising edge. As a result, we 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 to drive the LED segment display.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/11_Segment/ segment1.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./segment1
```

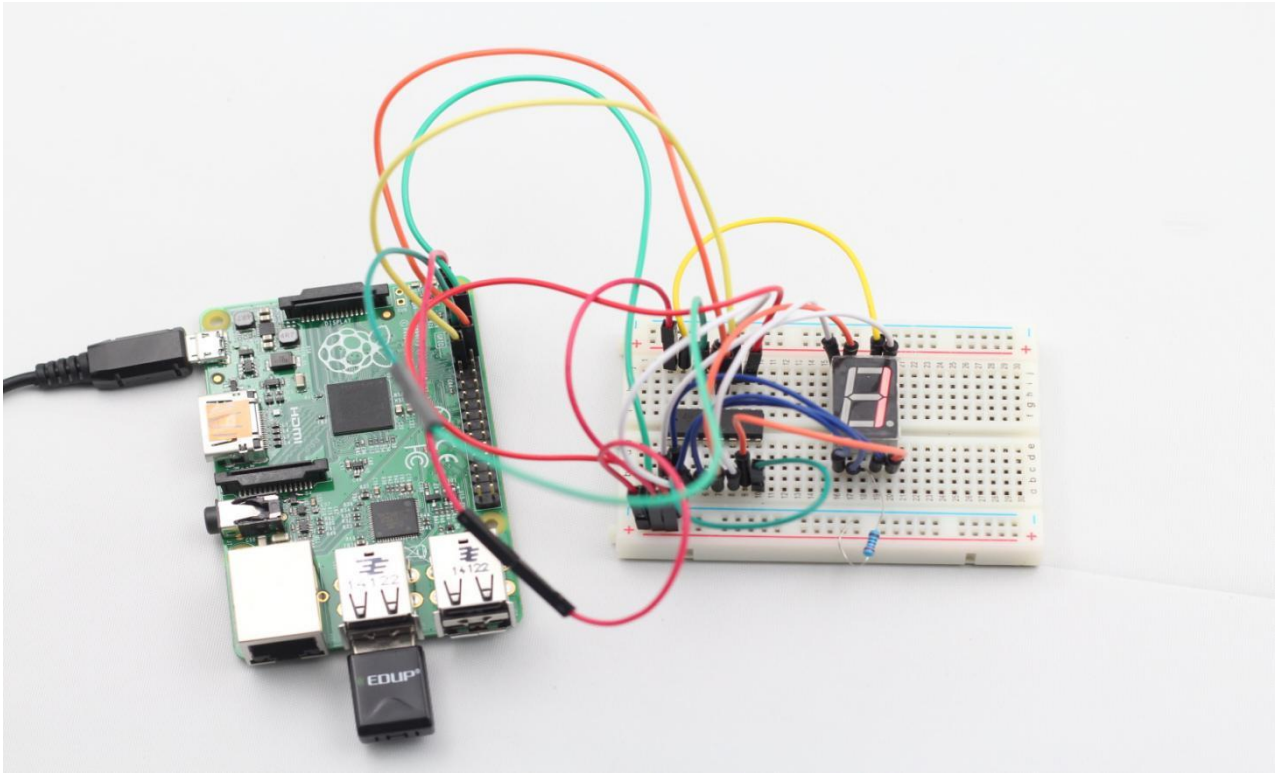
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/11_segment.py

Step 3: Run

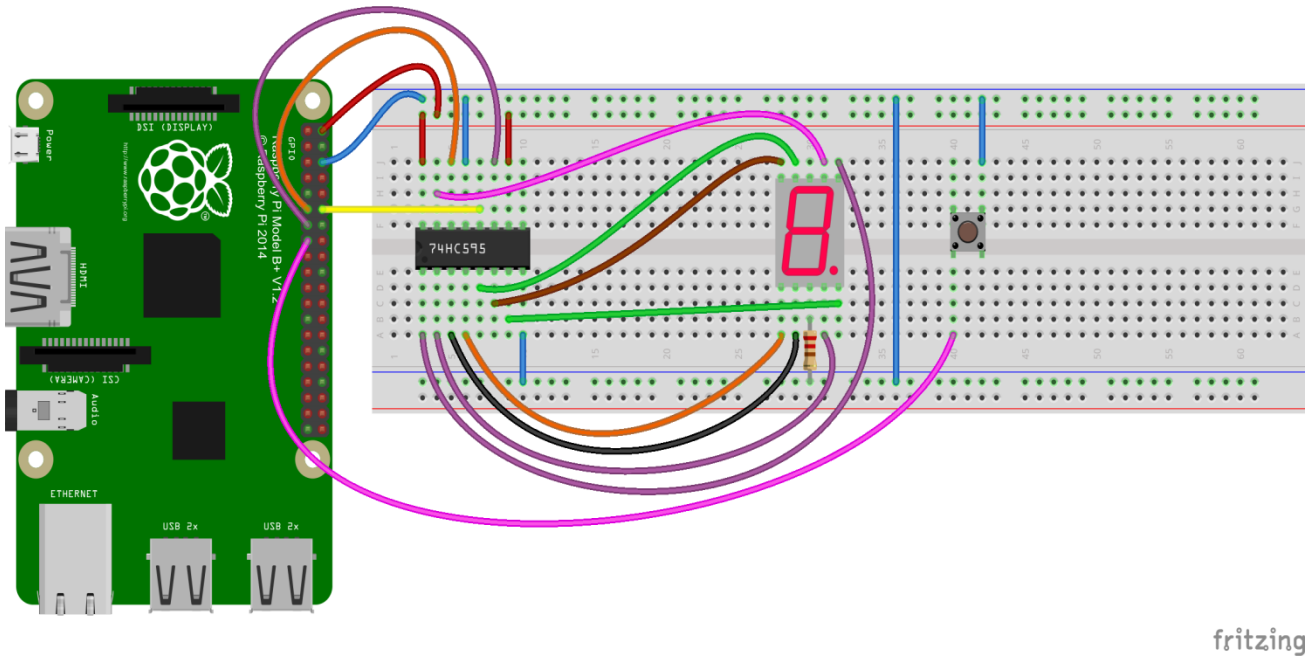
```
sudo python 11_segment.py
```

You should see the 7-segment display cycle from 0 to 9, and A to F. The practical effect is shown as below:



Further Exploration

You can slightly modify the hardware and software based on this experiment to make a dice. For hardware, you add a button to the original board. Then the modified circuit is as follows:



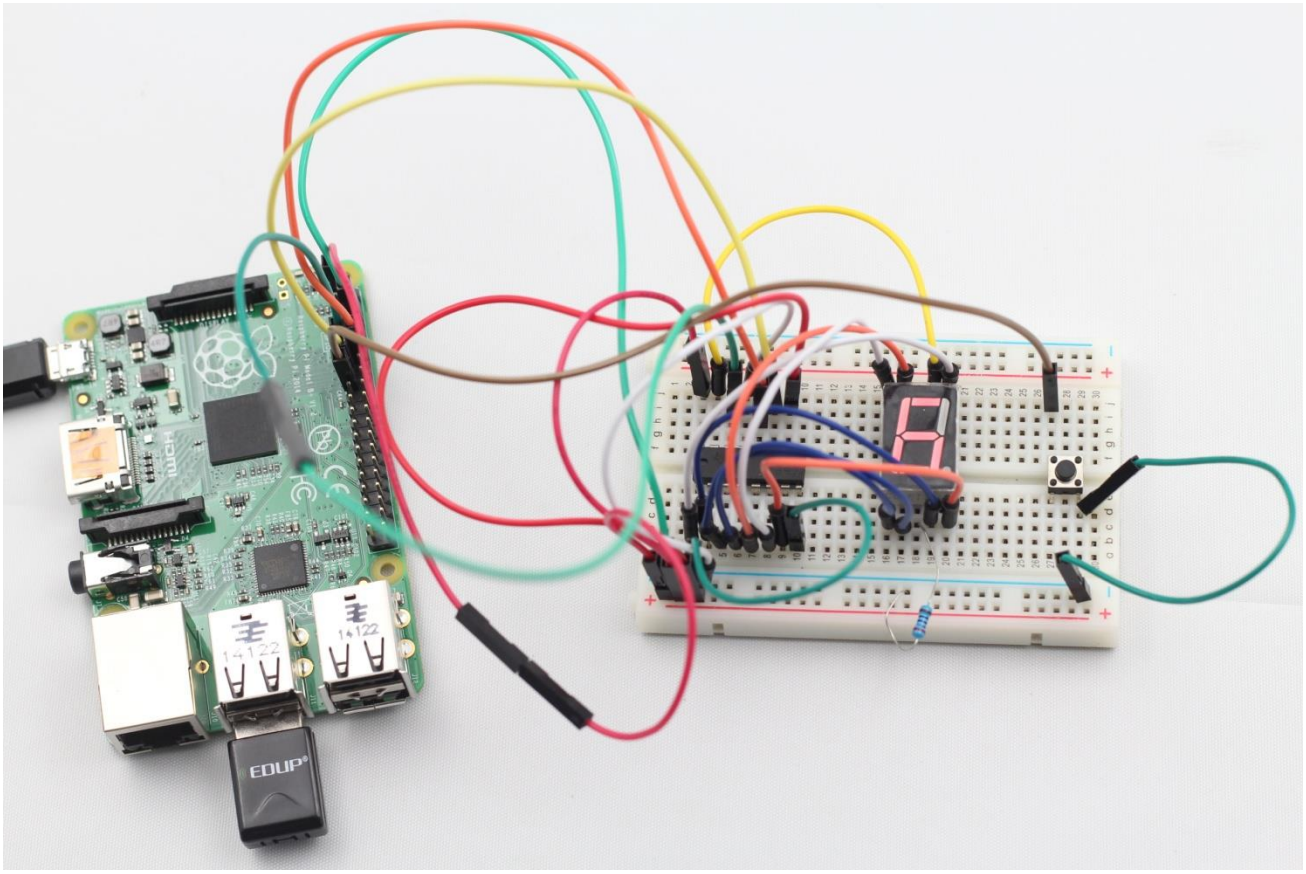
Next, enter into the directory 11_Segment, and compile *dice.c*

```
gcc dice.c -o dice -lwiringPi
```

Run

```
sudo ./dice
```

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



Summary

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

SUNFOUNDER's Raspberry Pi Lesson 12 Driving Dot-Matrix by 74HC595

Introduction

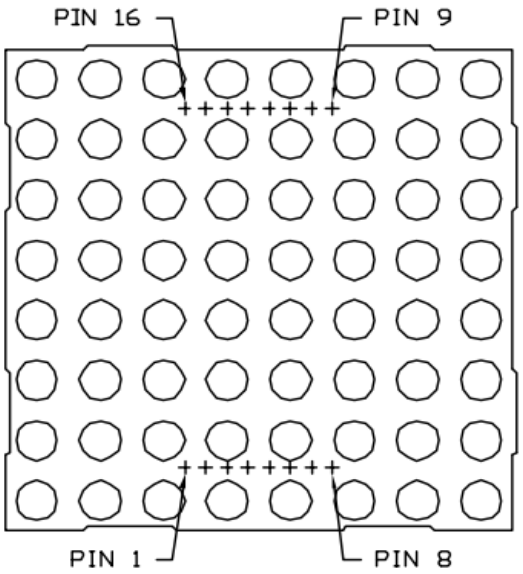
In this lesson, we will learn how to use 74HC595 to drive an LED dot-matrix.

Components

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

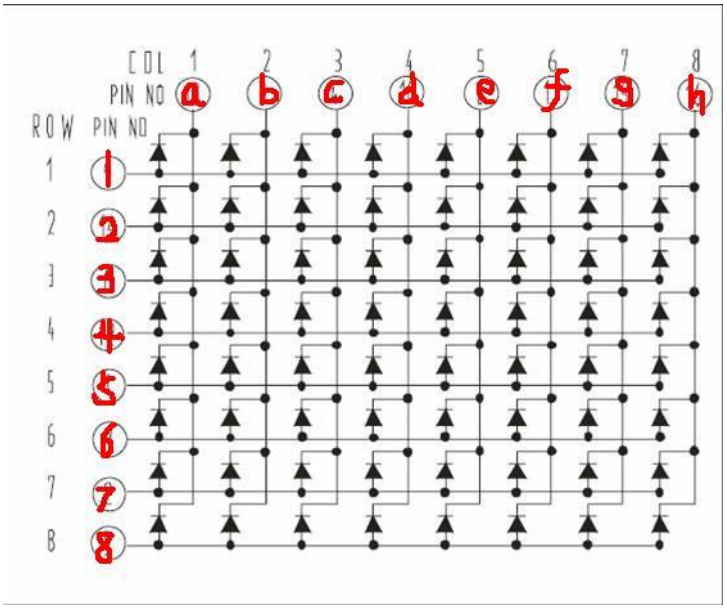
Principle

The external view of a dot-matrix is shown as follow:

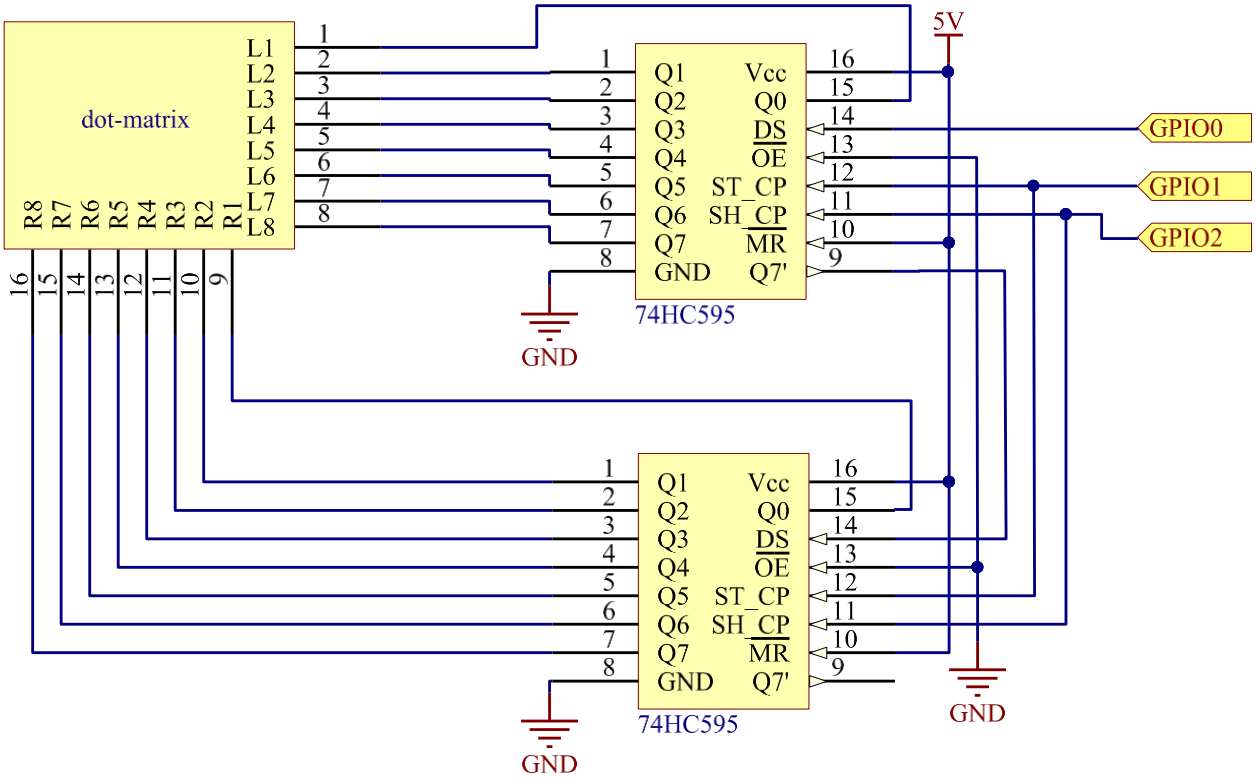


The principle of an 8*8 LED dot-matrix:

An 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 1 and that of a certain column is 0, then the LED at their cross point will light up. For example, if you want to light the LED on the first dot, set PIN 1 to high level and PIN a to low, and then the LED will light up; if you want to light all LEDs on the first row, set PIN 1 to high level and PIN (a, b, c, d, e, f, g, h) to low, and then all the LEDs on the first row will light up; if you want to light all the LEDs on the first column, set PIN a to low level and PIN (1, 2, 3, 4, 5, 6, 7, 8) to high, and then all the LEDs on the first column will light up.

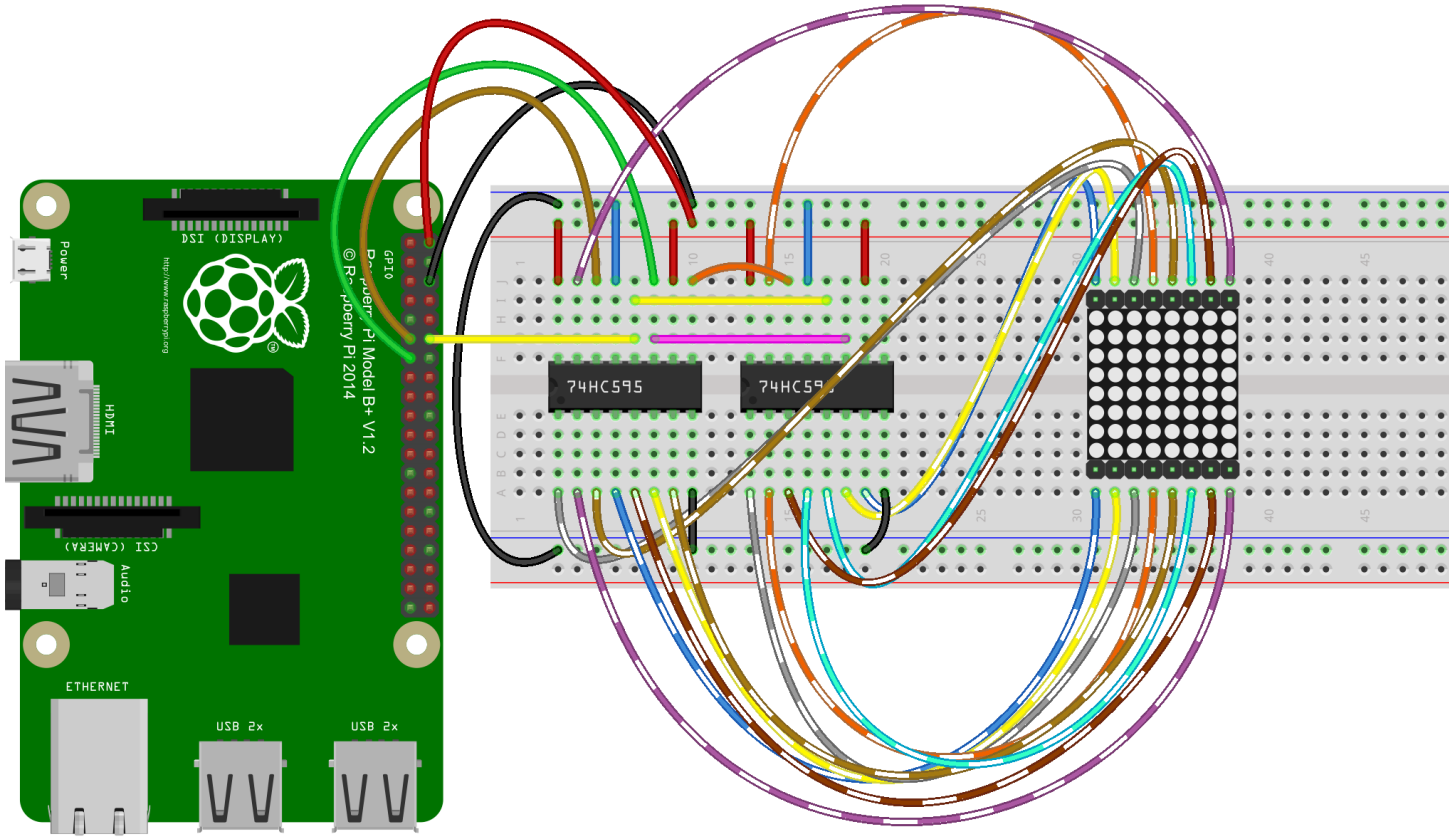


The principle of 74HC595 has been illustrated previously. One chip is used to control the rows of the dot-matrix while the other, the columns.



Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/12_DotMatrix/ dotMatrix.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./dotMatrix
```

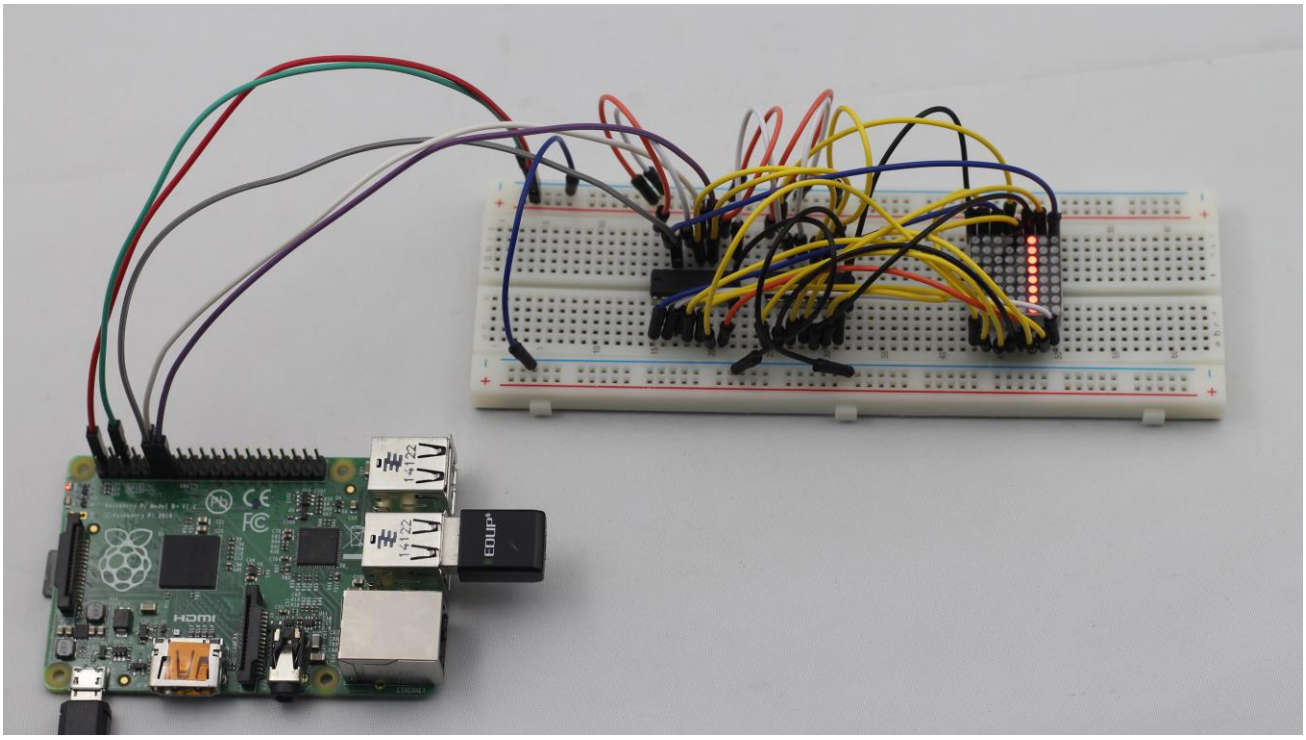
For Python users:

Step 2: See path/RPi_SuperKit_Code/Python_code/12_DotMatrix.py

Step 3: Run

```
sudo python 12_DotMatrix.py
```

You should see LEDs light up as you want.



Summary

Through this lesson, you have mastered the basic principle of LED dot-matrix and how to program Raspberry Pi to drive an LED dot-matrix based on 74HC595 cascade.

SUNFOUNDER's Raspberry Pi Lesson 13 LCD1602

Introduction

In this lesson, we will learn how to use LCD1602 to display character 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 positive power supply;

Pin 3 (Vo): used to adjust the contrast of LCD1602; the level is lowest when it's connected to 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): for register selection; it selects data register when supplied with high level (1), and instruction register when supplied with low level (0);

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

Pin 6 (E): An enable pin; when supplied with low level, the LCD module will execute relevant instructions;

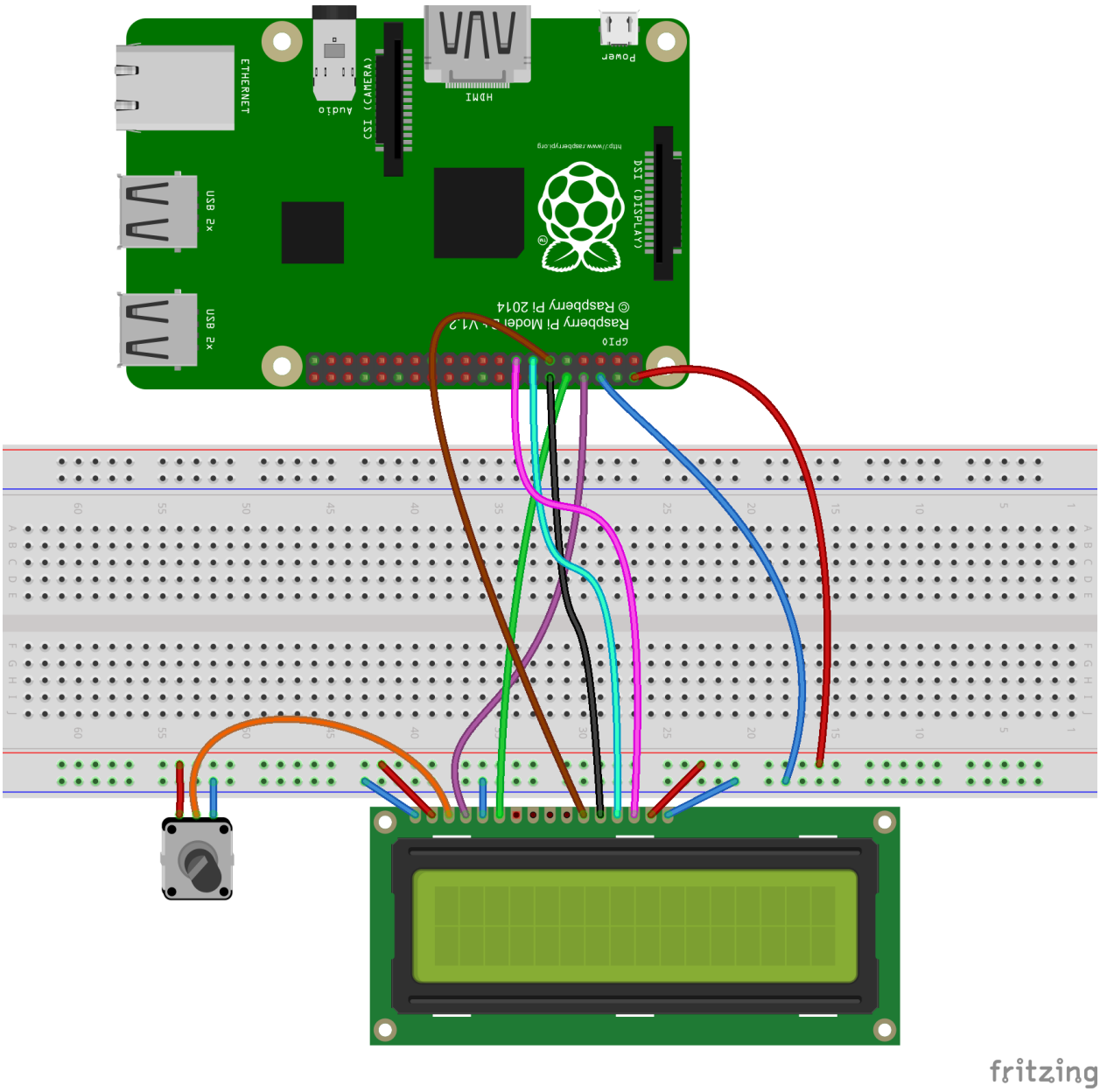
Pin 7 (D0-D7): pins that read and write data;

A and K: LCD backlight power source.

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.

Experimental Procedures

Step1: Connect the circuit as shown in the following diagram



For C language users:

Step 2: Edit and save the code (see path/RPi_SuperKit_Code/C_code/13_LCD1602/lcd1602_2.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./lcd0602_2
```

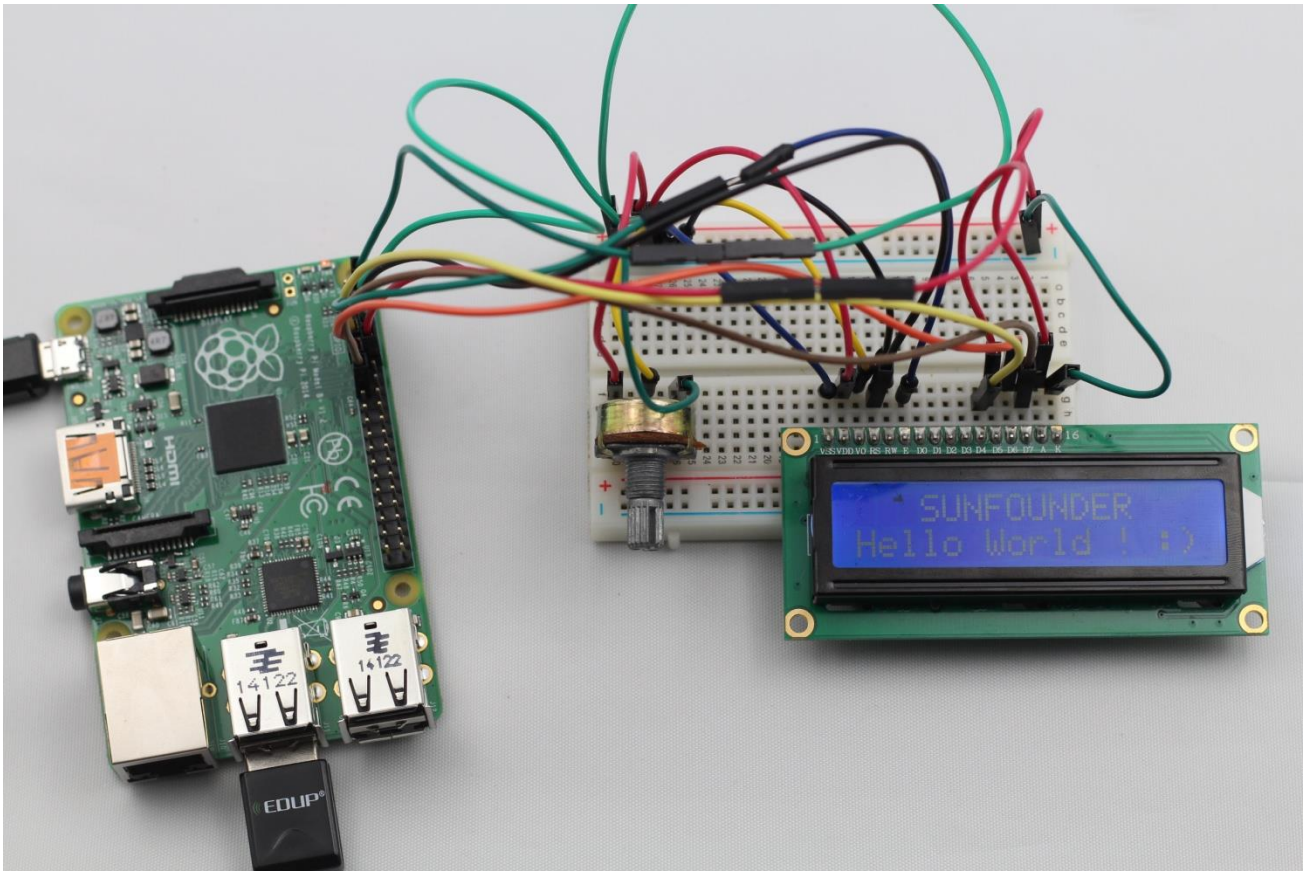
For Python users

Step 2: See path/RPi_SuperKit_Code/Python_code/13_ lcd1602.py

Step 3: Run

```
sudo python 13_lcd1602.py
```

You should see two lines of characters displayed on the LCD1602. The realistic effect is as follows:



Further Exploration

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

SUNFOUNDER's Raspberry Pi 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
An 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°.

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: Connect the circuit based on the following method

ADXL345 Module		Raspberry Pi
GND	----------	GND
3.3V	----------	3.3V
SCL0	----------	SCL
SDA0	----------	SDA
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.

For C language users:

Step 2: Edit and save the code with Vim (see path/RPi_SuperKit_Code/C_code/14_ADXL345/ adxl345.c)

Step 3: Compile

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

Step 4: Run

```
sudo ./adxl345
```

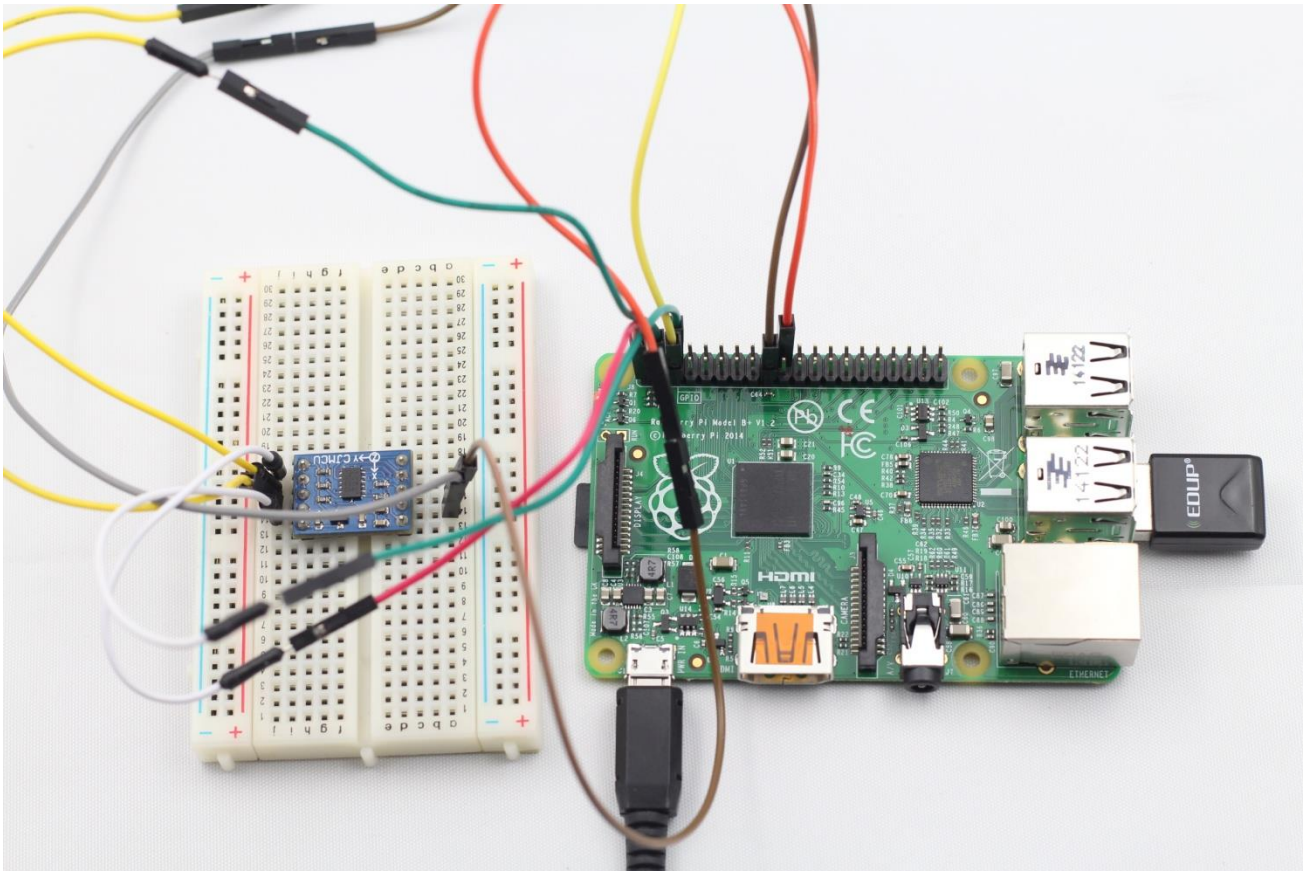
For Python users :

Step 2: See path/RPi_SuperKit_Code/Python_code/14_ADXL345/Adafruit_ADXL345/ Adafruit_ADXL345.py

Step 3: Run

```
sudo python Adafruit_ADXL345.py
```

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



Advanced application of Raspberry Pi – Controlling an LED Based on Web

Introduction

The reason why Raspberry Pi is so amazing is the networking function and support by numerous open source web applications. With it, your own Internet of Things system can be constructed conveniently. In this lesson, you will control an LED based on the web with Raspberry Pi. You can use your mobile phone, tablet pc or computer to control the LED too, as long as it is in the same Local Area Network (LAN) with your Raspberry Pi.

Components

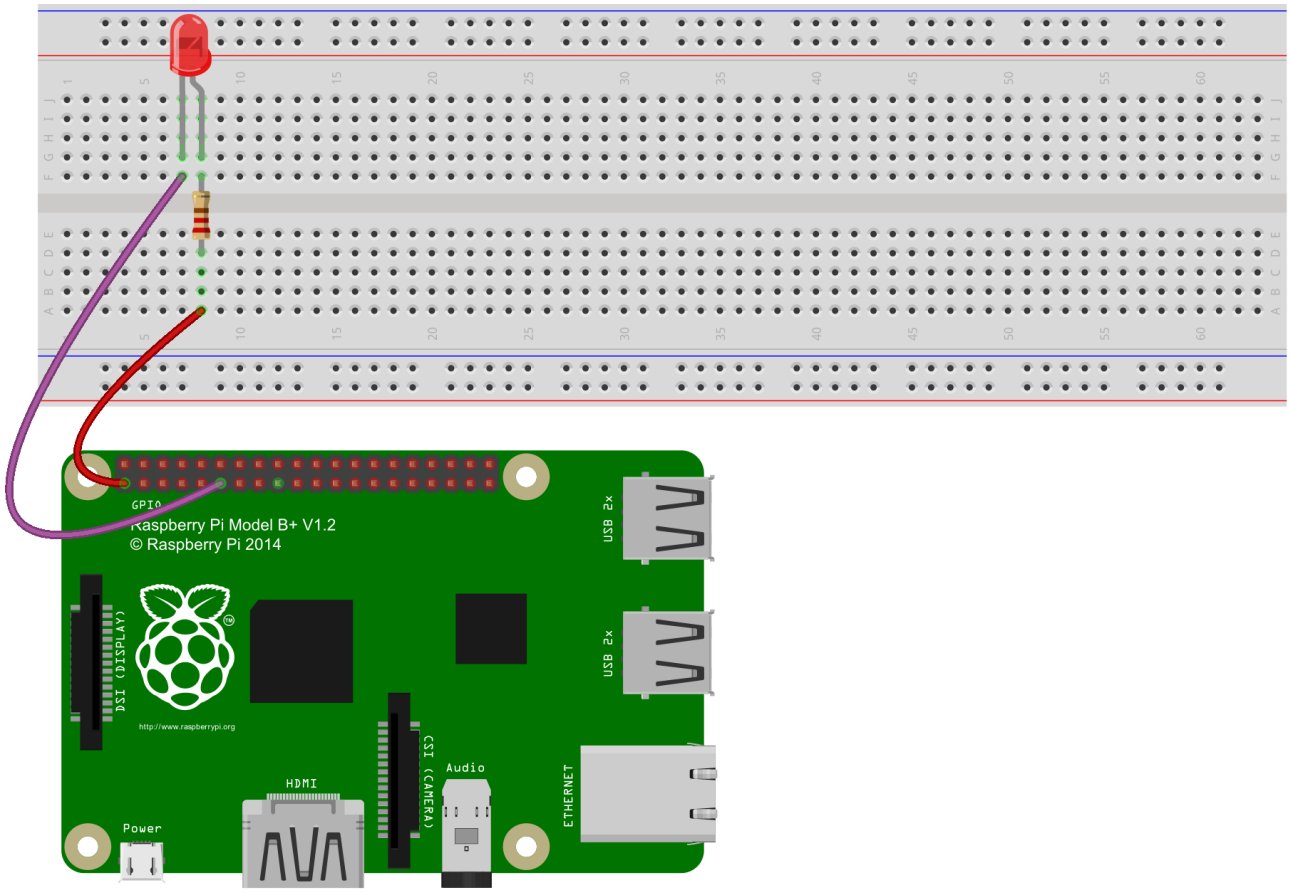
- 1* Raspberry Pi
- 1* Breadboard
- 1* LED
- 1* Resistor (220Ω)
- Jumper wires

Principle

WebIOPi is a fully integrated Internet of Things framework for the Raspberry Pi. It is a lightweight web server program on the Raspberry Pi platform which provides users a solution for remote access Raspberry Pi. That is, you can remotely operate your Raspberry Pi and control the hardware connected to it by a browser or an APP.

Experimental Procedures

1. Connect the circuit as shown in the following diagram



2. Install *WebIOPi*

```
wget http://webiopi.googlecode.com/files/WebIOPi-0.5.3.tar.gz
tar xvzf WebIOPi-0.5.3.tar.gz
cd WebIOPi-0.5.3
sudo ./setup.sh
```

3. Set or modify the password of *WebIOPi*

```
sudo webiopi-passwd
```

Username: webiopi

The default password is raspberry

4. Start *WebIOPi*

```
sudo python -m webiopi 8000
```

You can change the port by yourself. The default port is 8000.

5. Run *WebIOPi* in the back-end system, or it will be killed when you press Ctrl+C

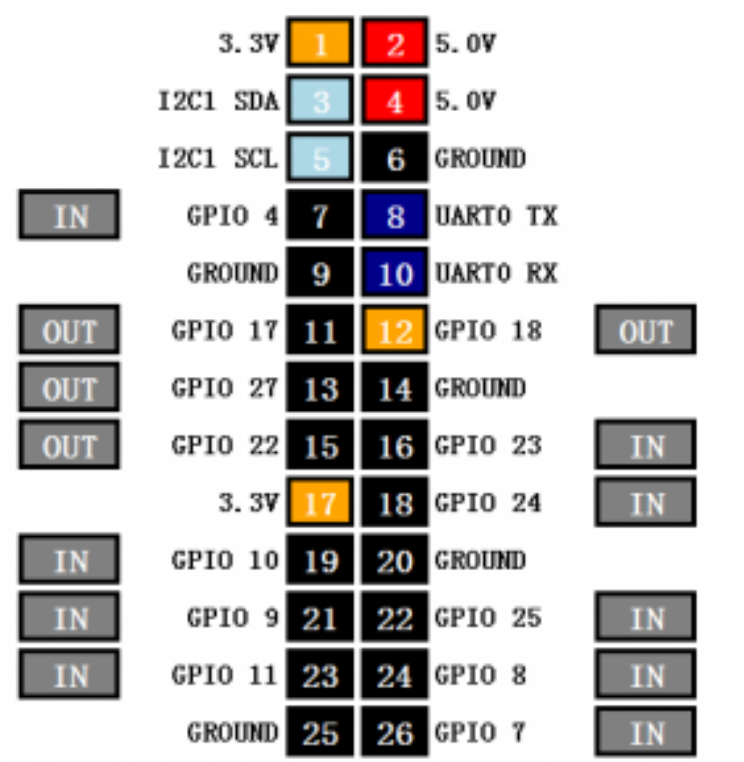
```
sudo /etc/init.d/webiopi start
and
```

```
sudo /etc/init.d/webiopi stop
```

6. Set *WebIOPi* to start with the system

```
sudo update-rc.d webiopi defaults
```

7. Open the interface of IP address access management with a browser
   http://192.168.1.106:8000/webiopi/



Control method:

1) Click outside IN/OUT to switch between the input and output modes of GPIOs
2) In the output mode, click the inside figures to switch between high and low outputs. After connecting the LED to GPIO17, click block 11, and you will see the state of the LED changed
3) In the input mode, the inside figures represent the input states of GPIO

Summary

In this lesson, you have learnt how to build the Internet of Things for *WebIOPi* platform based on Raspberry Pi and how to control your device remotely on the platform. You can learn WebIOPi in depth and make full use of your knowledge to make more fun things.

Postscript

If you have any questions, please send an email to support@sunfounder.com. We will reply to you ASAP!

For more tutorials, please visit our website www.sunfounder.com.

If you've created something you would like to share, you're welcome to post on our website.

Thanks!

SUNFOUNDER Technical Support Team