Chapter 1 LED

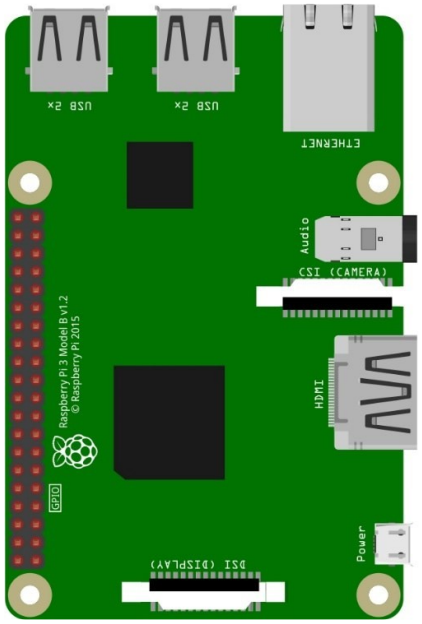
This chapter is the starting point of the journey to explore RPi electronic projects. Let's start with simple "Blink".

Project 1.1 Blink

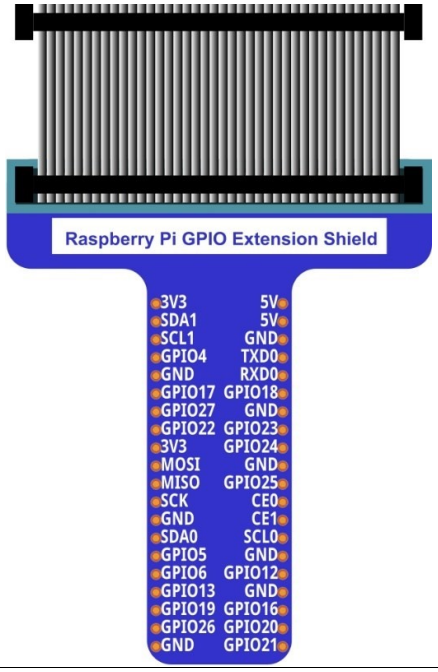
In this project, let's try to use RPi to control LED blinking.

Component List

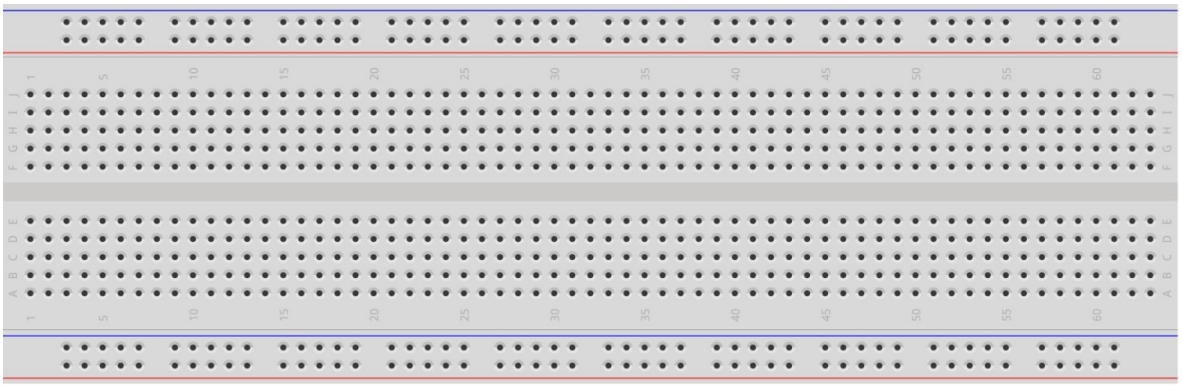
Raspberry Pi 3B x1

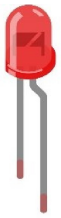


GPIO Extension Board & Wire x1



BreadBoard x1



LED x1	Resistor 220Ω x1	Jumper
		

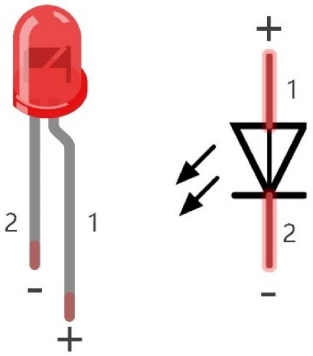
In the components list, 3B GPIO, Extension Shield Raspberry and Breadboard are necessary for each project. They will be listed only in text form later.

Component knowledge

LED

LED is a kind of diode. LED will shine only if the long pin of LED is connected to the positive electrode and the short pin is connected to negative electrode.

This is also the features of the common diode. Diode works only if the voltage of its positive electrode is higher than its negative electrode.



LED	Voltage	Maximum current	Recommended current
Red	1.9-2.2V	20mA	10mA
Green	2.9-3.4V	10mA	5mA
Blue	2.9-3.4V	10mA	5mA
Volt ampere characteristics conform to diode			

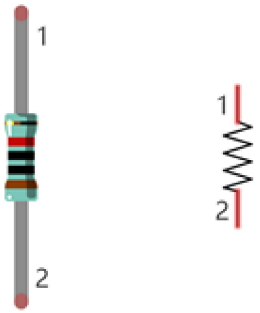
The LED can not be directly connected to power supply, which can damage component. A resistor with certain resistance must be connected in series in the circuit of LED.

Resistor

The unit of resistance(R) is ohm(Ω). 1mΩ=1000kΩ, 1kΩ=1000Ω.

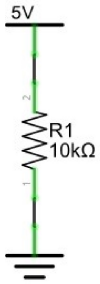
Resistor is an electrical component that limits or regulates the flow of current in an electronic circuit.

The left is the appearance of resistor, and the right is the symbol of resistor represented in circuit.



Color rings attached to the resistor is used to indicate its resistance. For more details of resistor color code, please refer to the appendix of this tutorial.

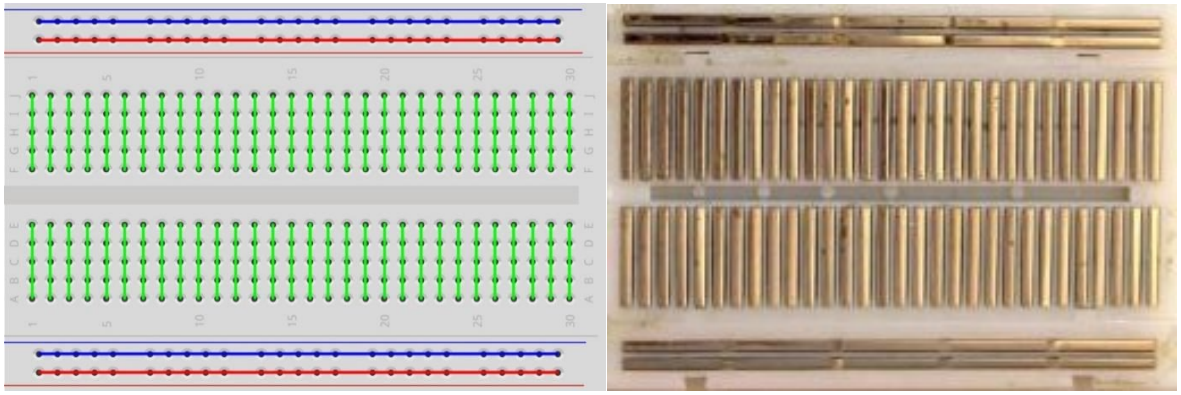
With the same voltage there will be less current with more resistance. And the links among current, voltage and resistance can be expressed by the formula below: I=U/R.
In the following diagram, the current through R1 is: I=U/R=5V/10kΩ=0.0005A=0.5mA.



Do not connect the two poles of power supply with low resistance, which will make the current too high to damage electronic components.
And resistor has no poles.

Breadboard

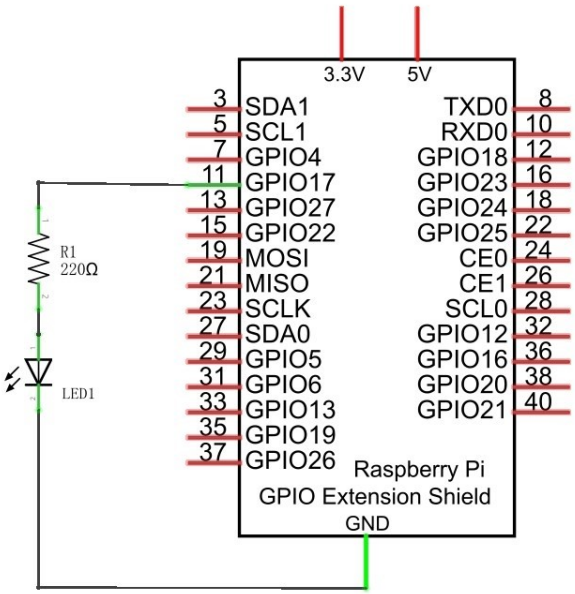
Take a short breadboard as an example to introduce its feature, as below.
The left picture shows the way of connection of pins. The right picture shows the pratical internal structure.



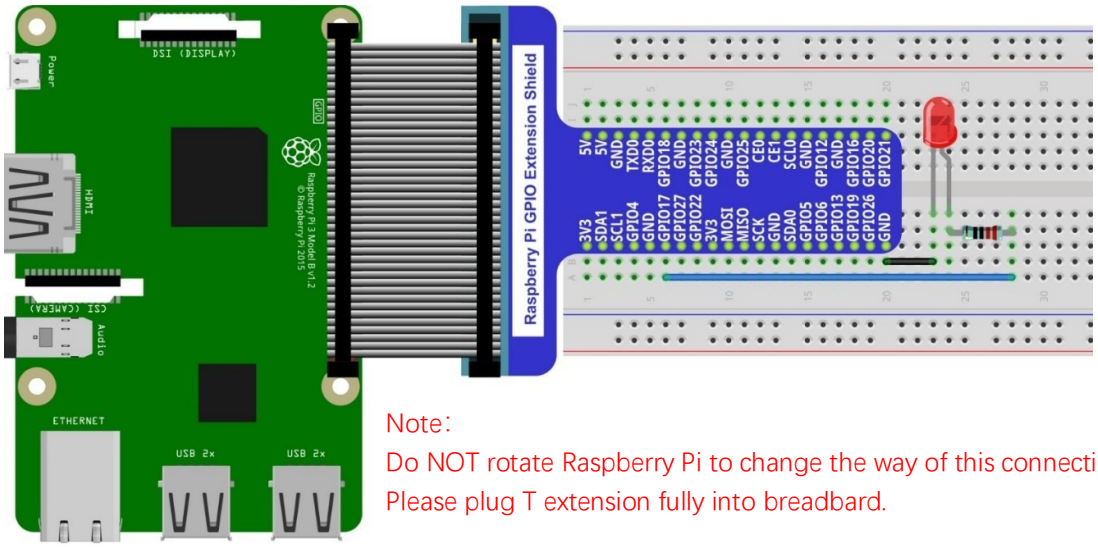
Circuit

Disconnect RPi from GPIO Extension Shield first. Then build the circuit according to the circuit diagram and the hardware connection diagram. After the circuit is built and confirmed, connect RPi to GPIO Extension Shield. In addition, short circuit (especially 5V and GND, 3.3V and GND) should be avoid, because short circuit may cause abnormal circuit work, or even damage to RPi.

Schematic diagram



Hardware connection. If you need any support, pleasefeel free to contact us via: support@freenove.com



Note:
Do NOT rotate Raspberry Pi to change the way of this connection.
Please plug T extension fully into breadbard.

Because Numbering of GPIO Extension Shield is the same as RPi GPIO, later Hardware connection diagram will only show the part of breadboard and GPIO Extension Shield.

Code

According to the circuit, when the GPIO17 of RPi output high level, LED is turned on. Conversely, when the GPIO17 RPi output low level, LED is turned off. Therefore, we can let GPIO17 output high and low level in cycle to make LED blink. We will use both C code and Python code to achieve the target.

C Code 1.1.1 Blink

First, execute command into the terminal one by one. Then observe the project result, and analyze the code.

1. Use cd command to enter 01.1.1_Blink directory of C code.

```
cd ~/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/Code/C_Code/01.1.1_Blink
```

2. Use the following command to compile the code "Blink.c" and generate executable file "Blink".

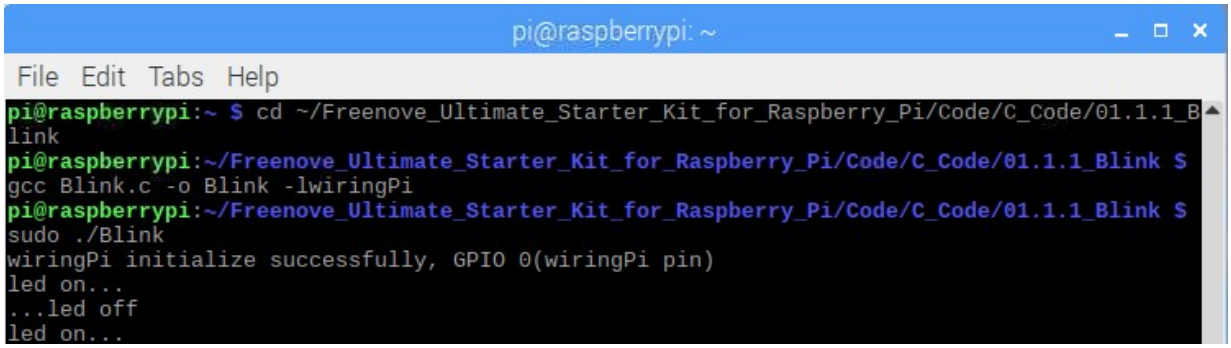
"l" of "lwiringPi" is low case of "L".

```
gcc Blink.c -o Blink -lwiringPi
```

3. Then run the generated file "blink".

```
sudo ./Blink
```

Now, LED start blink.



You can press "Ctrl+C" to end the program. The following is the program code:

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

#define  ledPin    0

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

    pinMode(ledPin, OUTPUT);

    while(1){
            digitalWrite(ledPin, HIGH);  //led on
```

```
19              printf("led on...\n");
20              delay(1000);
21              digitalWrite(ledPin, LOW);  //led off
22              printf("...led off\n");
23              delay(1000);
24      }
25
26      return 0;
27  }
```

GPIO connected to ledPin in the circuit is GPIO17. And GPIO17 is defined as 0 in the wiringPi numbering. So ledPin should be defined as 0 pin. You can refer to the corresponding table in Chapter 0.

```
#define  ledPin    0
```

In the main function main(), initialize wiringPi first, and then print out the initial results. Once the initialization fails, exit the program.

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

After the wiringPi is initialized successfully, set the ledPin to output mode. And then enter the while cycle, which is an endless loop. That is, the program will always be executed in this cycle, unless it is ended outside. In this cycle, use digitalWrite (ledPin, HIGH) to make ledPin output high level, then LED is turned on. After a period of time delay, use digitalWrite(ledPin, LOW) to make ledPin output low level, then LED is turned off, which is followed by a delay. Repeat the cycle, then LED will start blinking.

```
    pinMode(ledPin, OUTPUT);
    while(1){
            digitalWrite(ledPin, HIGH);  //led is turned on
            printf("led on...\n");
            delay(1000);
            digitalWrite(ledPin, LOW);  //led is turned off
            printf("...led off\n");
            delay(1000);
    }
```

Among them, the configuration function for GPIO is shown below as:

void pinMode(int pin, int mode);

This sets the mode of a pin to either INPUT, OUTPUT, PWM_OUTPUT or GPIO_CLOCK. Note that only wiringPi pin 1 (BCM_GPIO 18) supports PWM output and only wiringPi pin 7 (BCM_GPIO 4) supports CLOCK output modes.

This function has no effect when in Sys mode. If you need to change the pin mode, then you can do it with the gpio program in a script before you start your program

void digitalWrite (int pin, int value);

Writes the value HIGH or LOW (1 or 0) to the given pin which must have been previously set as an output.

For more related functions, please refer to http://wiringpi.com/reference/

Python Code 1.1.1 Blink

Net, we will use Python language to make LED blink.

First, observe the project result, and then analyze the code.

1. Use cd command to enter 01.1.1_Blink directory of Python code.

```
cd ~/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/Code/Python_Code/01.1.1_Blink
```

2. Use python command to execute python code blink.py.

```
python Blink.py
```

Now, LED start blinking.

```
pi@raspberrypi:~ $ cd ~/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/Code/Python_Code/01.1.1_Blink
pi@raspberrypi:~/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/Code/Python_Code/01.1.1_Blink $ python Blink.py
```

You can press "Ctrl+C" to end the program. The following is the program code:

```
1  import RPi.GPIO as GPIO
2  import time
3
4  ledPin = 11    # RPI Board pin11
5
6  def setup():
7      GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
8      GPIO.setup(ledPin, GPIO.OUT)   # Set ledPin's mode is output
9      GPIO.output(ledPin, GPIO.LOW) # Set ledPin low to off led
10     print ('using pin%d'%ledPin)
11
12 def loop():
13     while True:
14         GPIO.output(ledPin, GPIO.HIGH)  # led on
15         print ('...led on')
16         time.sleep(1)                   # delay 1 second
17         GPIO.output(ledPin, GPIO.LOW) # led off
18         print ('led off...')
19         time.sleep(1)
20 def destroy():
21     GPIO.output(ledPin, GPIO.LOW)     # led off
22     GPIO.cleanup()                     # Release resource
23
24 if __name__ == '__main__':     # Program start from here
25     setup()
26     try:
27         loop()
28     except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the subprogram destroy() will
29 be  executed.
30         destroy()
31
```

In subfunction setup(), GPIO.setmode (GPIO.BOARD) is used to set the serial number for GPIO based on physical location of the pin. GPIO17 use pin 11 of the board, so define ledPin as 11 and set ledPin to output

mode (output low level).

```
ledPin = 11    # RPi Board pin11
def setup():
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(ledPin, GPIO.OUT)    # Set ledPin to output mode
    GPIO.output(ledPin, GPIO.LOW) # Set ledPin to low level to turn off led
    print ('using pin%d'%ledPin)
```

In loop(), there is a while cycle, which is an endless loop. That is, the program will always be executed in this cycle, unless it is ended outside. In this cycle, set ledPin output high level, then LED is turned on. After a period of time delay, set ledPin output low level, then LED is turned off, which is followed by a delay. Repeat the cycle, then LED will start blinking.

```
def loop():
    while True:
        GPIO.output(ledPin, GPIO.HIGH)  # led on
        print ('...led on')
        time.sleep(1)
        GPIO.output(ledPin, GPIO.LOW) # led off
        print ('led off...')
        time.sleep(1)
```

Finally, when the program is terminated, subfunction will be executed, the LED will be turned off and then the IO port will be released. If close the program terminal directly, the program will be terminated too, but destroy() function will not be executed. So, GPIO resources won't be released, in the warning message may appear next time you use GPIO. So, it is not a good habit to close the program terminal directly.

```
def destroy():
    GPIO.output(ledPin, GPIO.LOW)     # led is turned off
    GPIO.cleanup()                     # Release resource
```

About RPi.GPIO:

RPi.GPIO
This is a Python module to control the GPIO on a Raspberry Pi. It includes basic output function and input function of GPIO, and function used to generate PWM.
GPIO.setmode(mode)
Set the mode for pin serial number of GPIO. mode=GPIO.BOARD, which represents the GPIO pin serial number is based on physical location of RPi. mode=GPIO.BCM, which represents the pin serial number is based on CPU of BCM chip.
GPIO.setup(pin,mode)
Set pin to input mode or output mode. "pin" for the GPIO pin, "mode" for INPUT or OUTPUT.
GPIO.output(pin,mode)
Set pin to output mode. "pin" for the GPIO pin, "mode" for HIGH (high level) or LOW (low level).

For more functions related to RPi.GPIO, please refer to:

https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/

"import time" time is a module of python.

https://docs.python.org/2/library/time.html?highlight=time%20time#module-time