Chapter7: Control GPIO with BST-AI expansion board

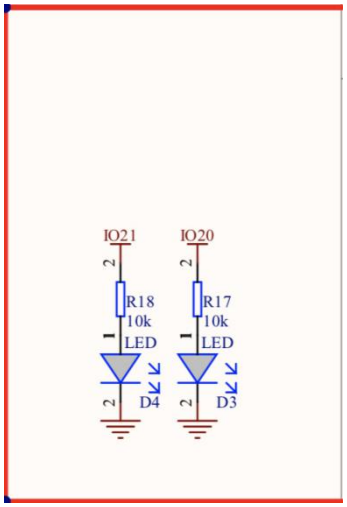


Figure 1-1 Diode

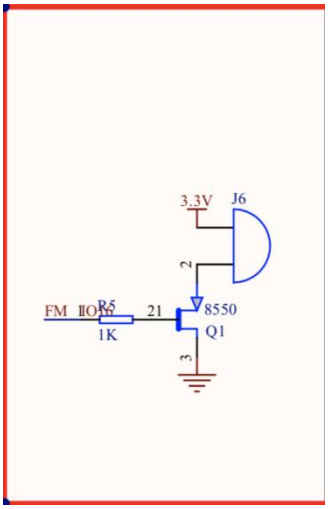


Figure 1-2 Buzzer

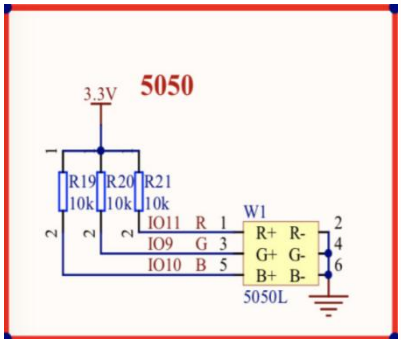


Figure 1-3 RGB

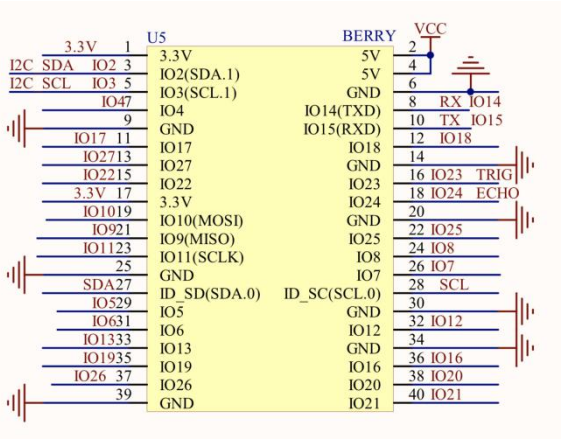


Figure 1-4 Raspberry Pi Pin Diagram

1. The first experiment is to make the blue and red LED lights flash alternately. According to the circuit diagram, we can know that the two LED small lights are controlled by GPIO20, 21.

Introduction to the way Python controls the Raspberry Pi GPIO:

1) we need to introduce the RPi.GPIO library file.

2) we need to- set the working mode of the Raspberry Pi GPIO. The parameters are BCM and BOARD. The BCM represents the GPIO number declaration of the pin, and the BOARD represents the pin number. It is recommended that users use BCM as the mode of declaration of working mode.

3) We need to declare the input and output of the specified pin with the .setup() function.

4) We need to change the level of the output pin to control the LED.

5)Finally, we need to release the working status with GPIO.cleanup(). Otherwise it will affect the next use.

The source code of the program is located at: **/home/pi/yahboom/GPIO/led.py**

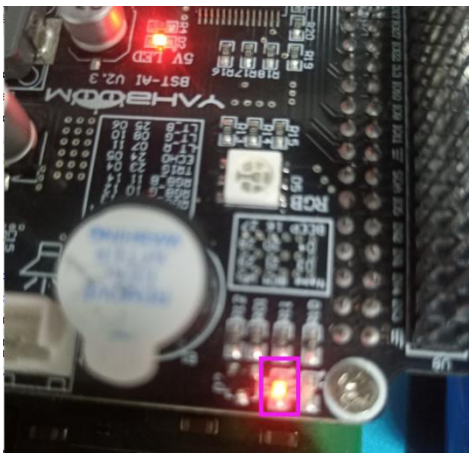
The code as shown in the figure below.

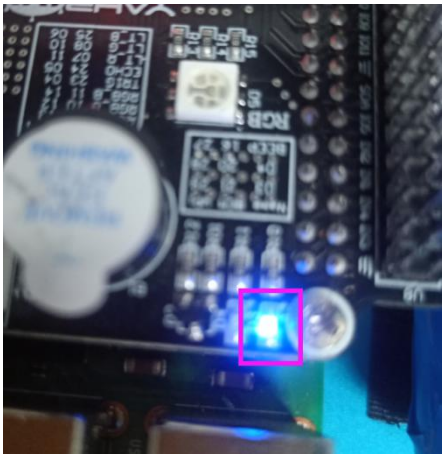
```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file         gpio
    * @version      V1.0
    * @details
    * @par History
    @author: longfuSun
"""


import RPi.GPIO as GPIO

import time
#Set work mode to BCM
GPIO.setmode(GPIO.BCM)
#set pins we will use
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
#run 10 times
for i in range(0,10):
    GPIO.output(20,True)
    time.sleep(0.5)
    GPIO.output(20,False)
    GPIO.output(21,True)
    time.sleep(0.5)
    GPIO.output(21,False)
GPIO.cleanup()
```

The result is as shown in the figure below.





2.In second experiment, we will use the PWM mode of operation to control the LED lights. The program provides a method for controlling the LED lights by pulse modulation and demodulation. The program will realize a breathing light to make the RGB lights, and the color of the gradient between red, green and blue.

The source code of the program is located at: **/home/pi/yahboom/GPIO/BST-AI.py**

The code as shown in the figure below:

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
    Created on Tue Nov  6 01:18:45 2018
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file        led-buzzer
    * @version      V1.0
    * @details
    * @par History

    @author: longfuSun
"""
import RPi.GPIO as GPIO
import time
#Control LED pin is 9,10,11;buzzer pin is 16
R,G,B=9,10,11
buzzer=16
GPIO.setmode(GPIO.BCM)

GPIO.setup(R, GPIO.OUT)
GPIO.setup(G, GPIO.OUT)
GPIO.setup(B, GPIO.OUT)
GPIO.setup(buzzer, GPIO.OUT)
#Make buzzer no sound

GPIO.output(buzzer, False)
time.sleep(2)
GPIO.output(buzzer,True)
#Set work mode of PWM
pwmR = GPIO.PWM(R, 70)
pwmG = GPIO.PWM(G, 70)
pwmB = GPIO.PWM(B, 70)

pwmR.start(0)
pwmG.start(0)
pwmB.start(0)
#Four modes
try:
    t = 0.01
    while True:
        for i in range(0,71):
            pwmG.ChangeDutyCycle(70)
            pwmB.ChangeDutyCycle(i)
            pwmR.ChangeDutyCycle(70-i)
            print(i)
            time.sleep(t)
        for i in range(70,-1,-1):
            pwmG.ChangeDutyCycle(0)
            pwmB.ChangeDutyCycle(i)
            pwmR.ChangeDutyCycle(70-i)
            print(i-1000)
            time.sleep(t)


except KeyboardInterrupt:
    pass
pwmR.stop()
pwmG.stop()
pwmB.stop()
GPIO.cleanup()
```