Lesson 8 Control RGB Light

1. Working Principle

RGB represents red, green and blue three colors and the parameter of three colors ranges from 0 to 255. The greater the value, the darker the color. The color can be changed by setting RGB color parameter.

The source code of program is located in: /home/ubuntu/armpi_pro/src/armpi_pro_demo/expansion_board_demo/RGB_Lighten.py

```
start = True
#Process before closing
def Stop(signum, frame):
    global start

    start = False
    print('Closing...')

#Turn off all lights
Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 0))
Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 0))
Board.RGB.show()

signal.signal(signal.SIGINT, Stop)

while True:
    Board.RGB.setPixelColor(0, Board.PixelColor(255, 0, 0)) # Set RGB1 on expansion board as red
    Board.RGB.setPixelColor(1, Board.PixelColor(255, 0, 0)) # Set RGB2 on expansion board as red
    Board.RGB.show()
    time.sleep(1)

    if not start:
        #Turn off all lights
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 0))
        Board.RGB.show()
        print('Closed')
        break
```

Control RGB light by calling RGB.setPixelColor() function under Board library. Take the code “Board.RGB.setPixelColor(0, Board.PixelColor(255, 0, 0))” as example.

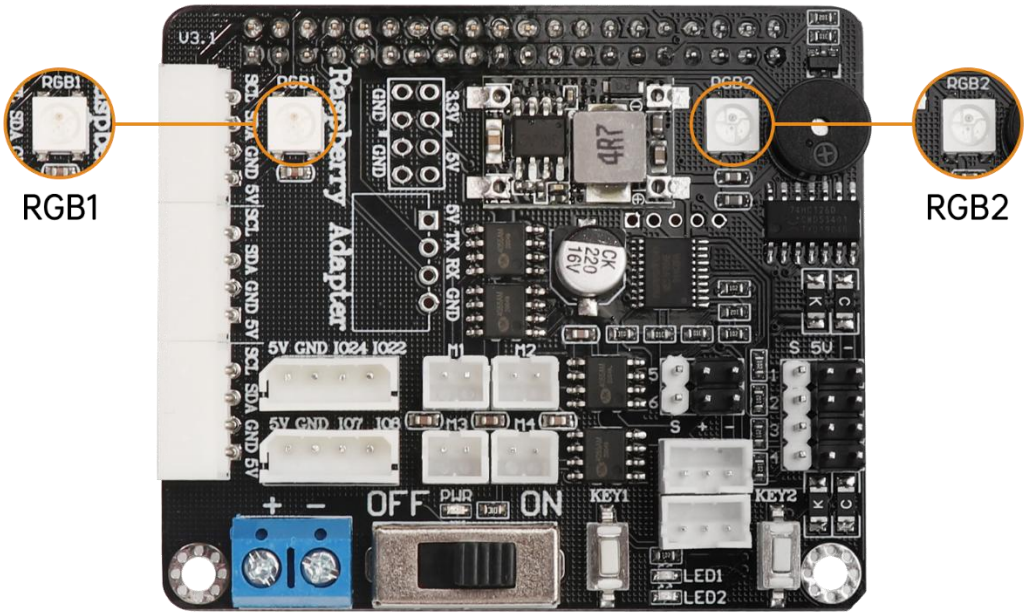
The first parameter “0” represents RGB1. “1” represents RGB2.

The second parameter “**Board.PixelColor(255, 0, 0)**” represents the parameter of color channel set by “**PixelColor()**” function. Among three parameter, the first

parameter “225” represents the value of “R” channel (red component); the second and third parameters are the values of “G” and “B” channels accordingly.

2. Preparation

There are two RGB lights on expansion board, as the figure shown below:



3. Operation Steps

1) Please refer to the tutorial in “6.Raspberry Pi and Expansion Board Lessons/2.Raspberry Pi Expansion Board/Lesson 4 Set Environment Development” to remotely connect through NoMachine.

2) Open the terminal. Click 

in the lower left corner and select 

to enter the terminal.

3) In the opened interface, enter the command "cd armpi_pro/src/armpi_pro_demo/expansion_board_demo/" to access to game programmings directory and then press "Enter".

```
ubuntu@ubuntu:~$ cd armpi_pro/src/armpi_pro_demo/expansion_board_demo/
```

4) Then enter command "sudo python3 BusServo_Speed.py" to start the game and press "Enter".

```
ubuntu@ubuntu:~/armpi_pro/src/armpi_pro_demo/expansion_board_demo$
sudo python3 RGB_Lighten.py
```

5) If want to exit the program, you can press "Ctrl+C". If fail to exit, you can try multiple times.

4. Project Outcome

After running program, two RGB lights on Raspberry Pi expansion board light up in red.

5. Function Extension

In program, RGB lights up red light default. If want to modify its color, we can modify the program. Take changing green light as example.

1) Enter command **"cd armpi_pro/src/armpi_pro_demo/expansion_board_demo/"** and press "Enter" to access to the game programmings directory.

```
ubuntu@ubuntu:~$ cd armpi_pro/src/armpi_pro_demo/expansion_board_demo/
```

2) Enter command "sudo vim RGB_Lighten.py" and press "Enter" to open the program file.

```
ubuntu@ubuntu:~/armpi_pro/src/armpi_pro_demo/expansion_board_demo$
sudo vim RGB_Lighten.py
```

3) Find the code shown in the figure below.

```
34 while True:
35     Board.RGB.setPixelColor(0, Board.PixelColor(255, 0, 0))
36     Board.RGB.setPixelColor(1, Board.PixelColor(255, 0, 0))
```

4) Press “i” key. When “INSERT” word appears, which means it has been switched to the editing mode.

```
46         break
47
-- INSERT --
```

5) We can change color by modifying RGB value. (Board.PixelColor (0, 0, 0) corresponds to RGB value. The first parameter is R, the second parameter is G and the third parameter is B, as the figure shown below.

```
34 while True:
35     Board.RGB.setPixelColor(0, Board.PixelColor(0, 255, 0))
36     Board.RGB.setPixelColor(1, Board.PixelColor(0, 255, 0))
```

6) After modifying, press “Esc”. Then enter “:wq” and press “Enter” to save and exit.

```
46         break
47
:wq
```

7) After saving program, please refer to “3.Operation Steps” to check the outcome.