Program Analysis

1. File Path

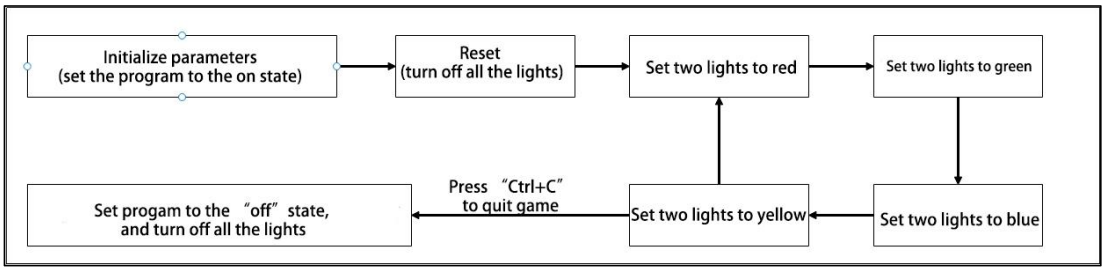
The program corresponding to this lesson is stored in **/home/pi/MasterPi/HiwonderSDK/RGBControlDemo.py**

2. Performance

After the program is started, the two RGB lights on Raspberry Pi expansion board light up in a cyclic pattern of red, green, blue, and yellow.

3. Program Analysis

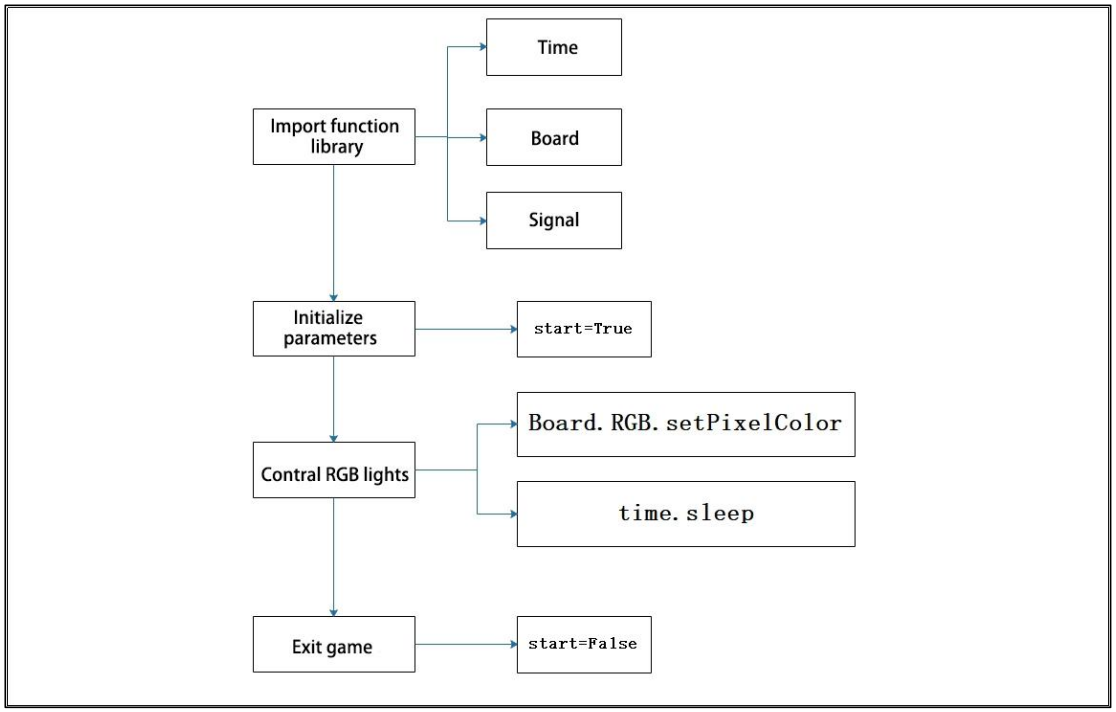
3.1 Program Logic



From the above picture, we can know that this game will first perform initialization and modify the parameters in program. Turn off all the lights, then control the RGB lights on expansion board to light up in a cyclic pattern of red, green, blue, and yellow.

When pushing “Ctrl+C” to close the game, the lights on expansion board will go off and program will disable the parameter settings.

3.2 Program Analysis



From the above flow diagram, the program primarily divides into four parts, namely importing the function library, initializing parameters, controlling RGB lights and closing the game. The following content will provide an explanation based on the logic of the program as depicted in the diagram.

- **Import Function Library**

Firstly, import library functions including Time, Board and Signal. The below table will explain the library files.

```
1 import time
2 import Board
3 import signal
```

Library File	Function
Time	The expression for handling time within a program refers to the method of obtaining the system time and formatting its output
Board	control sensors for executing control operations.

Signal	Use for receiving and processing signal

- **Initialize Parameters**

When executing a program, it is generally necessary to perform initialization within the program to facilitate the realization of the sequence functionalities. In this case, initialization involves setting the “Start” parameter in the program to True, indicating that the program is running. Also, all RGB lights are set to the “OFF” state.

```
18  start = True
```

```
26  #先将所有灯关闭
27  Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 0))
28  Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 0))
29  Board.RGB.show()
30
31  signal.signal(signal.SIGINT, Stop)
```

- **Control RGB Light**

After the initialization is completed, proceed with controlling RGB lights and its color. The Board.RGB.setPixelColor(), Board.RGB.show() and time.sleep() functions are used in this case.

The Board.RGB.setPixelColor() function can be filled in two parameters in parentheses, for example Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 0)), the first parameter corresponds to the serial number of RGB light and the second one is used to control the light colors. In this case, (0,0,0) represents the “off” state of light.

The Board.RGB.show() function is used to display the set RGB light states.

The time.sleep() function is used to control the duration of RDG light.

```
34      #设置2个灯为红色
35      Board.RGB.setPixelColor(0, Board.PixelColor(255, 0, 0))
36      Board.RGB.setPixelColor(1, Board.PixelColor(255, 0, 0))
37      Board.RGB.show()
38      time.sleep(1)
39
```

- **Quid Game**

The “start” parameter is set to “False” to quit the game.

```
#关闭前处理
def Stop(signum, frame):
    global start

    start = False
    print('关闭中...')
```

Then, there will be a trigger for the judgment to turn off all RGB lights.

```
    if not start:
        #所有灯关闭
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 0))
        Board.RGB.show()
        print('已关闭')
        break
```