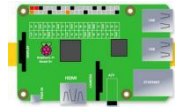
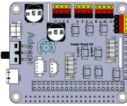



Lesson 7 How to Control Warm Color LED

In this lesson, we will learn how to control warm color LED with Raspberry Pi.

7.1 Components used in this course

Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
warm color LED	1	

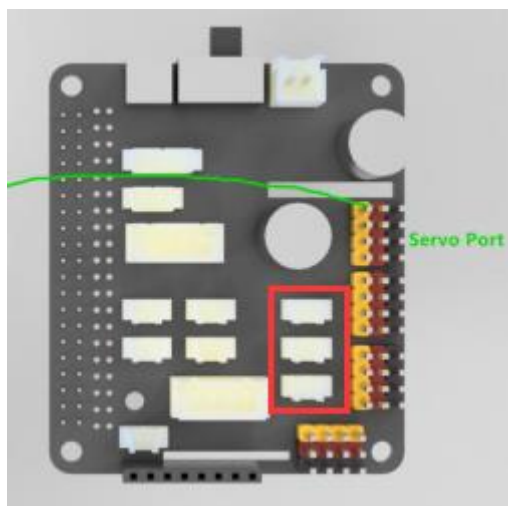
7.2 Introduction of warm color LED

The warm color LED is a LED that can emit warm blue light after being lit(The small LED emits white light). It has a working power of 0.5W, a working voltage of 5V/12V, and a light-emitting angle of 60°. The red line represents the positive pole and the black line represents the negative pole. The small LED needs to be connected to Port1, Port2, and Port3 on the Robot HAT driver board to use



7.3 Wiring diagram (Circuit diagram)

When the warm color LED is used, it needs to be connected to the Port1, Port2, and Port3 on the Robot HAT driver board, as shown in the red box:



7.4.1 Run the program of this course

1. Log in to your Raspberry Pi via SSH (the method to log in to the Raspberry Pi has been introduced in Lesson 3):

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 29 08:17:49 2020 from 192.168.3.208

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

2. Run the command to enter the [adeept_picarpro/server](#) folder. This folder stores the sample code program for controlling the robot. Enter the following command and press Enter:

```
cd adeept_picarpro/server
```

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ cd adeept_picarpro/server/
pi@raspberrypi:~/adeept_picarpro/server $
```

3. Enter the command to view the contents of the current directory:

```
ls
```

```
pi@raspberrypi:~/adeept_picarpro/server $
pi@raspberrypi:~/adeept_picarpro/server $ ls
app.py          findline.py    instruction.txt  OLED.py        servo.py
appserver.py    FPV.py        Kalman_filter.py  PID.py        switch.py
base_camera.py  FPVtest.py    LEDapp.py        __pycache__   ultra.py
camera_opencv.py functions.py    LED.py           robotLight.py  webServer.py
dist            info.py        move.py          RPIservo.py
pi@raspberrypi:~/adeept_picarpro/server $
```

4. [switch.py](#) is a python program, you can run this program on the Raspberry Pi by directly typing the following commands:

```
sudo python3 switch.py
```

```
pi@raspberrypi:~/adeept_picarpro/server $  
pi@raspberrypi:~/adeept_picarpro/server $ sudo python3 switch.py  
Light on...  
Light off...  
Light on...  
Light off...  
█
```

5. After successfully running the program, you will observe that the small warm LED light will be on for 1 second and be off for 1 second. When you want to terminate the running program, you can press the shortcut key "**Ctrl + C**" on the keyboard.

7.5 Main code program

The complete code reference file [switch.py](#).

1. **import** RPi.GPIO as GPIO
2. **import** time

First import the library used to control the Raspberry Pi GPIO, and instantiate it as a GPIO while importing it. Import the time library for code delay.

1. **def** switchSetup():
2. GPIO.setwarnings(False)
3. GPIO.setmode(GPIO.BCM)
4. GPIO.setup(5, GPIO.OUT)
5. GPIO.setup(6, GPIO.OUT)
6. GPIO.setup(13, GPIO.OUT)

The switchSetup function sets the GPIO pin numbers of the Raspberry Pi corresponding to the interface to 5, 6, and 13, here we use BCM coding.

1. **def** switch(port, status):
2. **if** port == 1:
3. **if** status == 1:

```
4.     GPIO.output(5, GPIO.HIGH)
5.     elif status == 0:
6.         GPIO.output(5,GPIO.LOW)
7.     else:
8.         pass
9.     elif port == 2:
10.        if status == 1:
11.            GPIO.output(6, GPIO.HIGH)
12.        elif status == 0:
13.            GPIO.output(6,GPIO.LOW)
14.        else:
15.            pass
16.    elif port == 3:
17.        if status == 1:
18.            GPIO.output(13, GPIO.HIGH)
19.        elif status == 0:
20.            GPIO.output(13,GPIO.LOW)
21.        else:
22.            pass
23.    else:
24.        print('Wrong Command: Example--switch(3, 1)->to switch on port3')
```

The switch function sets the high and low levels of the interface. Port represents ports 1-3. When the status is 0, the corresponding interface light is off, and when the status is 1, the corresponding interface light is on.

```
1. def set_all_switch_off():
2.     switch(1,0)
3.     switch(2,0)
4.     switch(3,0)
```

The set_all_switch_off function sets the level of all interfaces to low level, that is, turns off the lights of all interfaces.

```
1. if __name__ == "__main__":
2.     switchSetup()
3.     while 1:
4.         switch(1,1)
```

```
5.    switch(2,1)
6.    switch(3,1)
7.    print ("Light on....")
8.    time.sleep(1)
9.    set_all_switch_off()
10.   print("Light off....")
11.   time.sleep(1)
```

Instantiating the object and executing the method function **while 1** means always looping, **switchSetup**, **switch**, **set_all_switch_off** means calling the above function. This code keeps the light on for 1 second and then be off for 1 second.

7.6 How to control the LED on Robot HAT driver board

When you study the content of this course, even if you do not use the Small LED module, you can also learn this lesson by controlling the LED on the Robot HAT driver board, because there is a green LED next to each interface: when the interface is turned on, the LED next to the interface will light up, indicating that the interface is on.

You can observe the status of the LED on the Robot HAT driver board by using the method of "7.4 How to control the warm color LED " above.