

Lesson 19 Introduction of Warning Light

In this lesson, we will learn how to use RaspTank's warning light function.

19.1 Function Overview

This tutorial is about how to use multi-threading to achieve some effects related to WS2812 LED lights. Multi-threading is common in robot projects, since robots have high requirements for real-time response. For each task performing, try not to block the main thread communication.

Multi-threading is similar to executing multiple different programs or tasks at the same time. Multi-threaded operation has the following advantages:

1. Using threads to put time-consuming tasks in the background for processing.
2. Improving the efficiency of the program. In the subsequent real-time video and OpenCV processing video frames, multi-threading is used to greatly increase the frame rate.
3. It's more convenient to call an encapsulated multi-threaded task, similar to the non-blocking control method – in other words, the control of the servo is encapsulated by multi-threading.

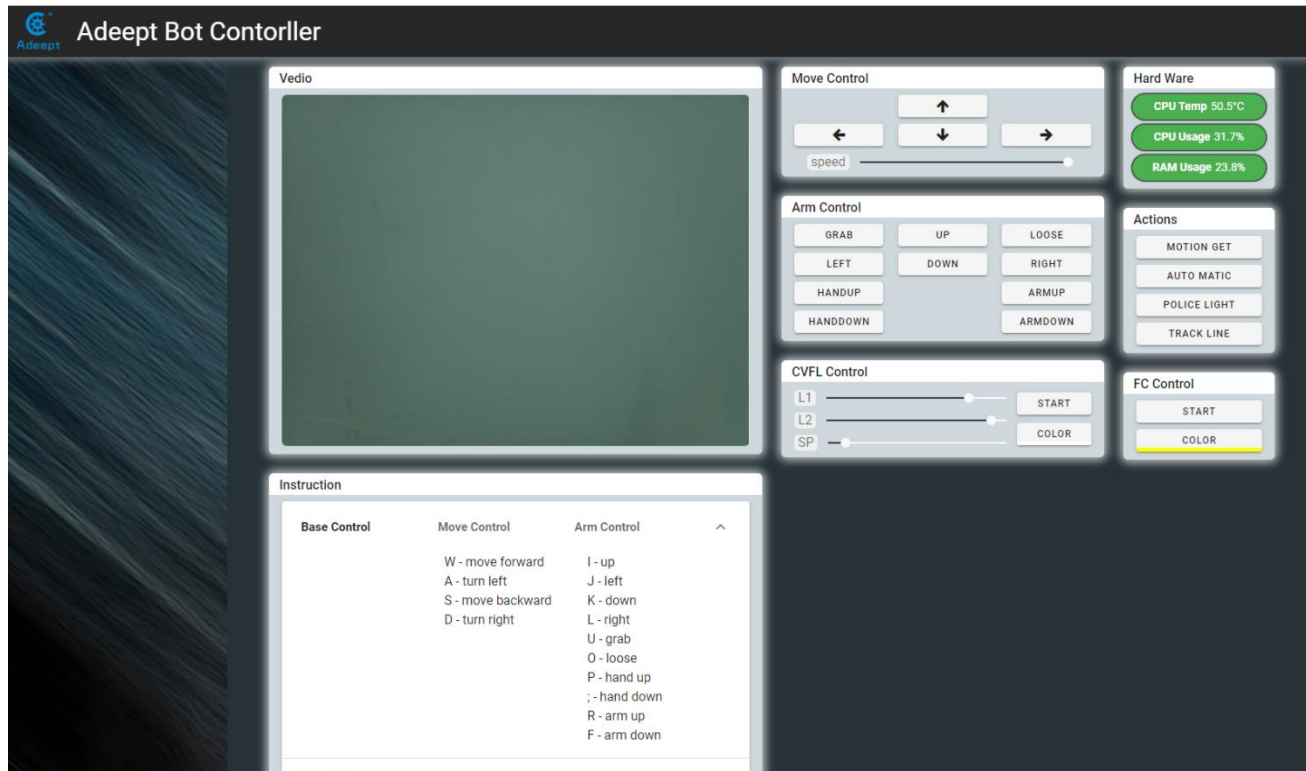
We use Python's threading library to provide thread-related work, and thread is the smallest unit of work in an application. For the current version of Python, there are no priorities, no thread groups, and threads cannot be stopped, suspended, resumed, or interrupted.

19.2 Run the Warning Light Program

1. Start the RaspTank Robot. It may take about 30-50s to boot.

2. After RaspTank is turned on, open the Chrome browser on your mobile or computer, enter the IP address of your Raspberry Pi and access port ":5000" into the IP address bar, like this:

192.168.3.44:5000. The web controller will then be displayed on the browser.



3. Click "POLICE LIGHT", and RaspTank will flash lights of different colors.

Note: Currently this module is not compatible with Raspberry Pi 5. You need to wait for the WS1812 official update dependency library to be compatible with Raspberry Pi 5. Some of the latest hardware versions of Raspberry Pi 4 may also have incompatibility issues. This requires waiting for the official update of WS2812.

4. Click "POLICE LIGHT" again to stop the function.

19.3 Code

```
1. #!/usr/bin/env python3
2. import time
```

```
3. import sys
4. from gpiozero import PWMOutputDevice as PWM
5. import threading
6. import spidev
7. import numpy
8. from numpy import sin, cos, pi
9. def check_rpi_model():
10.     _, result = run_command("cat /proc/device-tree/model |awk '{print $3}'")
11.     result = result.strip()
12.     if result == '3':
13.         return 3
14.     elif result == '4':
15.         return 4
16.     elif result == '5':
17.         return 5
18.     else:
19.         return None

20. def run_command(cmd=""):
21.     import subprocess
22.     p = subprocess.Popen(
23.         cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
24.     result = p.stdout.read().decode('utf-8')
25.     status = p.poll()
26.     return status, result

27. def map(x, in_min, in_max, out_min, out_max):
28.     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

29. class Adeept_SPI_LedPixel(threading.Thread):
30.     def __init__(self, count = 8, bright = 255, sequence='GRB', bus = 0, device = 0, *args, **kwargs):
31.         self.set_led_type(sequence)
32.         self.set_led_count(count)
33.         self.set_led_brightness(bright)
34.         self.led_begin(bus, device)
35.         self.lightMode = 'none'
36.         self.colorBreathR = 0
37.         self.colorBreathG = 0
38.         self.colorBreathB = 0
39.         self.breathSteps = 10
40.         #self.spi_gpio_info()
41.         self.set_all_led_color(0,0,0)
```

```

42. super(Adeept_SPI_LedPixel, self).__init__(*args, **kwargs)
43. self.__flag = threading.Event()
44. self.__flag.clear()
45. def led_begin(self, bus = 0, device = 0):
46.     self.bus = bus
47.     self.device = device
48.     try:
49.         self.spi = spidev.SpiDev()
50.         self.spi.open(self.bus, self.device)
51.         self.spi.mode = 0
52.         self.led_init_state = 1
53.     except OSError:
54.         print("Please check the configuration in /boot/firmware/config.txt.")
55.         if self.bus == 0:
56.             print("You can turn on the 'SPI' in 'Interface Options' by using 'sudo raspi-config'.")
57.             print("Or make sure that 'dtoverlay=spi=on' is not commented, then reboot the Raspberry Pi. Otherwise
                    spi0 will not be available.")
58.         else:
59.             print("Please add 'dtoverlay=spi{}-2cs' at the bottom of the /boot/firmware/config.txt, then reboot
                    the Raspberry Pi. otherwise spi{} will not be available.".format(self.bus, self.bus))
60.         self.led_init_state = 0
61.
62.     def check_spi_state(self):
63.         return self.led_init_state
64.
65.     def spi_gpio_info(self):
66.         if self.bus == 0:
67.             print("SPI0-MOSI: GPIO10(WS2812-PIN)  SPI0-MISO: GPIO9  SPI0-SCLK: GPIO11  SPI0-CE0:
                    GPIO8  SPI0-CE1: GPIO7")
68.         elif self.bus == 1:
69.             print("SPI1-MOSI: GPIO20(WS2812-PIN)  SPI1-MISO: GPIO19  SPI1-SCLK: GPIO21  SPI1-CE0:
                    GPIO18  SPI1-CE1: GPIO17  SPI0-CE1: GPIO16")
70.         elif self.bus == 2:
71.             print("SPI2-MOSI: GPIO41(WS2812-PIN)  SPI2-MISO: GPIO40  SPI2-SCLK: GPIO42  SPI2-CE0:
                    GPIO43  SPI2-CE1: GPIO44  SPI2-CE1: GPIO45")
72.         elif self.bus == 3:
73.             print("SPI3-MOSI: GPIO2(WS2812-PIN)  SPI3-MISO: GPIO1  SPI3-SCLK: GPIO3  SPI3-CE0:
                    GPIO0  SPI3-CE1: GPIO24")
74.         elif self.bus == 4:
75.             print("SPI4-MOSI: GPIO6(WS2812-PIN)  SPI4-MISO: GPIO5  SPI4-SCLK: GPIO7  SPI4-CE0:
                    GPIO4  SPI4-CE1: GPIO25")
76.         elif self.bus == 5:

```

```
77. print("SPI5-MOSI: GPIO14(WS2812-PIN) SPI5-MISO: GPIO13 SPI5-SCLK: GPIO15 SPI5-CE0:
    GPIO12 SPI5-CE1: GPIO26")
78. elif self.bus == 6:
79. print("SPI6-MOSI: GPIO20(WS2812-PIN) SPI6-MISO: GPIO19 SPI6-SCLK: GPIO21 SPI6-CE0:
    GPIO18 SPI6-CE1: GPIO27")
80.
81. def led_close(self):
82. self.set_all_led_rgb([0,0,0])
83. self.spi.close()
84.
85. def set_led_count(self, count):
86. self.led_count = count
87. self.led_color = [0,0,0] * self.led_count
88. self.led_original_color = [0,0,0] * self.led_count
89.
90. def set_led_type(self, rgb_type):
91. try:
92. led_type = ['RGB', 'RBG', 'GRB', 'GBR', 'BRG', 'BGR']
93. led_type_offset = [0x06,0x09,0x12,0x21,0x18,0x24]
94. index = led_type.index(rgb_type)
95. self.led_red_offset = (led_type_offset[index]>>4) & 0x03
96. self.led_green_offset = (led_type_offset[index]>>2) & 0x03
97. self.led_blue_offset = (led_type_offset[index]>>0) & 0x03
98. return index
99. except ValueError:
100. self.led_red_offset = 1
101. self.led_green_offset = 0
102. self.led_blue_offset = 2
103. return -1
104.
105. def set_led_brightness(self, brightness):
106. self.led_brightness = brightness
107. for i in range(self.led_count):
108. self.set_led_rgb_data(i, self.led_original_color)
109.
110. def set_ledpixel(self, index, r, g, b):
111. p = [0,0,0]
112. p[self.led_red_offset] = round(r * self.led_brightness / 255)
113. p[self.led_green_offset] = round(g * self.led_brightness / 255)
114. p[self.led_blue_offset] = round(b * self.led_brightness / 255)
115. self.led_original_color[index*3+self.led_red_offset] = r
116. self.led_original_color[index*3+self.led_green_offset] = g
```

```

117. self.led_original_color[index*3+self.led_blue_offset] = b
118. for i in range(3):
119. self.led_color[index*3+i] = p[i]

120. def set_led_color_data(self, index, r, g, b):
121. self.set_ledpixel(index, r, g, b)
122.
123. def set_led_rgb_data(self, index, color):
124. self.set_ledpixel(index, color[0], color[1], color[2])
125.
126. def set_led_color(self, index, r, g, b):
127. self.set_ledpixel(index, r, g, b)
128. self.show()
129.
130. def set_led_rgb(self, index, color):
131. self.set_led_rgb_data(index, color)
132. self.show()
133.
134. def set_all_led_color_data(self, r, g, b):
135. for i in range(self.led_count):
136. self.set_led_color_data(i, r, g, b)
137.
138. def set_all_led_rgb_data(self, color):
139. for i in range(self.led_count):
140. self.set_led_rgb_data(i, color)
141.
142. def set_all_led_color(self, r, g, b):
143. for i in range(self.led_count):
144. self.set_led_color_data(i, r, g, b)
145. self.show()
146.
147. def set_all_led_rgb(self, color):
148. for i in range(self.led_count):
149. self.set_led_rgb_data(i, color)
150. self.show()
151.
152. def write_ws2812_numpy8(self):
153. d = numpy.array(self.led_color).ravel()           #Converts data into a one-dimensional array
154. tx = numpy.zeros(len(d)*8, dtype=numpy.uint8)     #Each RGB color has 8 bits, each represented by a
    uint8 type data
155. for ibit in range(8):                             #Convert each bit of data to the data that the spi
    will send

```

```

156. #tx[7-ibit::8]=((d>>ibit)&1)*0x3E + 0xC0    #T0H=2,T0L=6, T1H=7,T1L=1    #0b11111110 mean
      T1(1.09375us), 0b11000000 mean T0(0.3125us)
157. #tx[7-ibit::8]=((d>>ibit)&1)*0x3C + 0xC0    #T0H=2,T0L=6, T1H=6,T1L=2    #0b11111100 mean
      T1(0.9375us), 0b11000000 mean T0(0.3125us)
158. #tx[7-ibit::8]=((d>>ibit)&1)*0x38 + 0xC0    #T0H=2,T0L=6, T1H=5,T1L=3    #0b11111000 mean
      T1(0.78125us), 0b11000000 mean T0(0.3125us)
159. #tx[7-ibit::8]=((d>>ibit)&1)*0x30 + 0xC0    #T0H=2,T0L=6, T1H=4,T1L=4    #0b11110000 mean
      T1(0.625us), 0b11000000 mean T0(0.3125us)
160. #tx[7-ibit::8]=((d>>ibit)&1)*0x20 + 0xC0    #T0H=2,T0L=6, T1H=3,T1L=5    #0b11100000 mean
      T1(0.46875us), 0b11000000 mean T0(0.3125us)
161. #tx[7-ibit::8]=((d>>ibit)&1)*0x7E + 0x80    #T0H=1,T0L=7, T1H=7,T1L=1    #0b11111110 mean
      T1(0.09375us), 0b10000000 mean T0(0.15625us)
162. #tx[7-ibit::8]=((d>>ibit)&1)*0x78 + 0x80    #T0H=1,T0L=7, T1H=5,T1L=3    #0b11111000 mean
      T1(0.78125us), 0b10000000 mean T0(0.15625us)
163. #tx[7-ibit::8]=((d>>ibit)&1)*0x70 + 0x80    #T0H=1,T0L=7, T1H=4,T1L=4    #0b11110000 mean
      T1(0.625us), 0b10000000 mean T0(0.15625us)
164. #tx[7-ibit::8]=((d>>ibit)&1)*0x60 + 0x80    #T0H=1,T0L=7, T1H=3,T1L=5    #0b11100000 mean
      T1(0.46875us), 0b10000000 mean T0(0.15625us)
165. tx[7-ibit::8]=((d>>ibit)&1)*0x78 + 0x80    #T0H=1,T0L=7, T1H=5,T1L=3    #0b11111000 mean
      T1(0.78125us), 0b10000000 mean T0(0.15625us)
166. if self.led_init_state != 0:
167. if self.bus == 0:
168. self.spi.xfer(tx.tolist(), int(8/1.25e-6))    #Send color data at a frequency of 6.4Mhz
169. else:
170. self.spi.xfer(tx.tolist(), int(8/1.0e-6))    #Send color data at a frequency of 8Mhz
171.
172. def write_ws2812_numpy4(self):
173. d=numpy.array(self.led_color).ravel()
174. tx=numpy.zeros(len(d)*4, dtype=numpy.uint8)
175. for ibit in range(4):
176. tx[3-ibit::4]=((d>>(2*ibit+1))&1)*0x60 + ((d>>(2*ibit+0))&1)*0x06 + 0x88
177. if self.led_init_state != 0:
178. if self.bus == 0:
179. self.spi.xfer(tx.tolist(), int(4/1.25e-6))
180. else:
181. self.spi.xfer(tx.tolist(), int(4/1.0e-6))
182.
183. def show(self, mode = 1):
184. if mode == 1:
185. write_ws2812 = self.write_ws2812_numpy8
186. else:
187. write_ws2812 = self.write_ws2812_numpy4

```

```
188.write_ws2812()
189.
190.def wheel(self, pos):
191.if pos < 85:
192.return [(255 - pos * 3), (pos * 3), 0]
193.elif pos < 170:
194.pos = pos - 85
195.return [0, (255 - pos * 3), (pos * 3)]
196.else:
197.pos = pos - 170
198.return [(pos * 3), 0, (255 - pos * 3)]
199.
200.def hsv2rgb(self, h, s, v):
201.h = h % 360
202.rgb_max = round(v * 2.55)
203.rgb_min = round(rgb_max * (100 - s) / 100)
204.i = round(h / 60)
205.diff = round(h % 60)
206.rgb_adj = round((rgb_max - rgb_min) * diff / 60)
207.if i == 0:
208.r = rgb_max
209.g = rgb_min + rgb_adj
210.b = rgb_min
211.elif i == 1:
212.r = rgb_max - rgb_adj
213.g = rgb_max
214.b = rgb_min
215.elif i == 2:
216.r = rgb_min
217.g = rgb_max
218.b = rgb_min + rgb_adj
219.elif i == 3:
220.r = rgb_min
221.g = rgb_max - rgb_adj
222.b = rgb_max
223.elif i == 4:
224.r = rgb_min + rgb_adj
225.g = rgb_min
226.b = rgb_max
227.else:
228.r = rgb_max
229.g = rgb_min
```



```
230.b = rgb_max - rgb_adj
231.return [r, g, b]
232.
233.def police(self):
234.self.lightMode = 'police'
235.self.resume()
236.
237.def breath(self, R_input, G_input, B_input):
238.self.lightMode = 'breath'
239.self.colorBreathR = R_input
240.self.colorBreathG = G_input
241.self.colorBreathB = B_input
242.self.resume()
243.
244.def resume(self):
245.self.__flag.set()
246.
247.def pause(self):
248.self.lightMode = 'none'
249.self.set_all_led_color_data(0,0,0)
250.self.__flag.clear()
251.
251.def breathProcessing(self):
252.while self.lightMode == 'breath':
253.for i in range(0,self.breathSteps):
254.if self.lightMode != 'breath':
255.break
256.self.set_all_led_color(self.colorBreathR*i/self.breathSteps,
    self.colorBreathG*i/self.breathSteps, self.colorBreathB*i/self.breathSteps)
257.#self.show()
258.time.sleep(0.03)
259.for i in range(0,self.breathSteps):
260.if self.lightMode != 'breath':
261.break
262.self.set_all_led_color(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
    self.colorBreathG-(self.colorBreathG*i/self.breathSteps),
    self.colorBreathB-(self.colorBreathB*i/self.breathSteps))
263.#self.show()
264.time.sleep(0.03)
265.def policeProcessing(self):
266.while self.lightMode == 'police':
```

```
267. for i in range(0,3):
268. self.set_all_led_color_data(0,0,255)
269. self.show()
270. time.sleep(0.05)
271. self.set_all_led_color_data(0,0,0)
272. self.show()
273. time.sleep(0.05)
274. if self.lightMode != 'police':
275. break
276. time.sleep(0.1)
277. for i in range(0,3):
278. self.set_all_led_color_data(255,0,0)
279. self.show()
280. time.sleep(0.05)
281. self.set_all_led_color_data(0,0,0)
282. self.show()
283. time.sleep(0.05)
284. time.sleep(0.1)
285.
286.
287. def lightChange(self):
288. if self.lightMode == 'none':
289. self.pause()
290. elif self.lightMode == 'police':
291. self.policeProcessing()
292. elif self.lightMode == 'breath':
293. self.breathProcessing()
294.
295. def run(self):
296. while 1:
297. self.__flag.wait()
298. self.lightChange()
299. pass
300.

301. class RobotLight(threading.Thread):
302. def __init__(self, *args, **kwargs):

303. self.left_R = 7
304. self.left_G = 0
305. self.left_B = 8
```

```
306. self.right_R = 5
307. self.right_G = 6
308. self.right_B = 1
309.
310. self.Left_G = PWM(pin=self.left_R, initial_value=1.0, frequency=2000)
311. self.Left_B = PWM(pin=self.left_G, initial_value=1.0, frequency=2000)
312. self.Left_R = PWM(pin=self.left_B, initial_value=1.0, frequency=2000)
313.
314. self.Right_G = PWM(pin=self.right_R, initial_value=1.0, frequency=2000)
315. self.Right_B = PWM(pin=self.right_G, initial_value=1.0, frequency=2000)
316. self.Right_R = PWM(pin=self.right_B, initial_value=1.0, frequency=2000)

317. # super(RobotLight, self).__init__(*args, **kwargs)
318. # self.__flag = threading.Event()
319. # self.__flag.clear()

320. # def setColorLED(self, LED_num, col): # For example : col = 0x112233
321. #     if LED_num == 1 :
322. #         R_val = (col & 0xff0000) >> 16
323. #         G_val = (col & 0x00ff00) >> 8
324. #         B_val = (col & 0x0000ff) >> 0
325.
326. #         R_val = map(R_val, 0, 255, 0, 1.00)
327. #         G_val = map(G_val, 0, 255, 0, 1.00)
328. #         B_val = map(B_val, 0, 255, 0, 1.00)

329. #         self.Left_R.value = 1.0-R_val
330. #         self.Left_G.value = 1.0-G_val
331. #         self.Left_B.value = 1.0-B_val
332. #     elif LED_num == 2:
333. #         R_val = (col & 0xff0000) >> 16
334. #         G_val = (col & 0x00ff00) >> 8
335. #         B_val = (col & 0x0000ff) >> 0
336.
337. #         R_val = map(R_val, 0, 255, 0, 1.00)
338. #         G_val = map(G_val, 0, 255, 0, 1.00)
339. #         B_val = map(B_val, 0, 255, 0, 1.00)

340. #         self.Right_R.value = 1.0-R_val
341. #         self.Right_G.value = 1.0-G_val
342. #         self.Right_B.value = 1.0-B_val
```

```
343. def setRGBColor(self, LED_num, R,G,B):    # For example : (1, 255,0,0)
344. if LED_num ==1 :
345.     R_val = map(R, 0, 255, 0, 1.00)
346.     G_val = map(G, 0, 255, 0, 1.00)
347.     B_val = map(B, 0, 255, 0, 1.00)
348.     self.Left_R.value = 1.0-R_val
349.     self.Left_G.value = 1.0-G_val
350.     self.Left_B.value = 1.0-B_val

351. elif LED_num == 2:
352.     R_val = map(R, 0, 255, 0, 1.00)
353.     G_val = map(G, 0, 255, 0, 1.00)
354.     B_val = map(B, 0, 255, 0, 1.00)
355.     self.Right_R.value = 1.0-R_val
356.     self.Right_G.value = 1.0-G_val
357.     self.Right_B.value = 1.0-B_val

358. def both_on(self,R,G,B):
359.     self.setRGBColor(1, R,G,B)
360.     self.setRGBColor(2, R,G,B)

361. def RGB_left_on(self,R,G,B):
362.     self.setRGBColor(1, R,G,B)
363.     self.setRGBColor(2, 0,0,0)

364. def RGB_right_on(self,R,G,B):
365.     self.setRGBColor(1, 0,0,0)
366.     self.setRGBColor(2, R,G,B)

367. def both_off(self):
368.     self.setRGBColor(1, 0,0,0)
369.     self.setRGBColor(2, 0,0,0)

370. # def pause(self):
371. #     self.lightMode = 'none'
372. #     self.setColor(0,0,0)
373. #     self.__flag.clear()

374. # def resume(self):
375. #     self.__flag.set()
376.
```

```
377.# def frontLight(self, switch):
378.#     if switch == 'on':
379.#         GPIO.output(6, GPIO.HIGH)
380.#         GPIO.output(13, GPIO.HIGH)
381.#     elif switch == 'off':
382.#         GPIO.output(5,GPIO.LOW)
383.#         GPIO.output(13,GPIO.LOW)

384.# def switch(self, port, status):
385.#     if port == 1:
386.#         if status == 1:
387.#             GPIO.output(5, GPIO.HIGH)
388.#         elif status == 0:
389.#             GPIO.output(5,GPIO.LOW)
390.#         else:
391.#             pass
392.#     elif port == 2:
393.#         if status == 1:
394.#             GPIO.output(6, GPIO.HIGH)
395.#         elif status == 0:
396.#             GPIO.output(6,GPIO.LOW)
397.#         else:
398.#             pass
399.#     elif port == 3:
400.#         if status == 1:
401.#             GPIO.output(13, GPIO.HIGH)
402.#         elif status == 0:
403.#             GPIO.output(13,GPIO.LOW)
404.#         else:
405.#             pass
406.#     else:
407.#         print('Wrong Command: Example--switch(3, 1)->to switch on port3')

408.# def set_all_switch_off(self):
409.#     self.switch(1,0)
410.#     self.switch(2,0)
411.#     self.switch(3,0)

412.# def headLight(self, switch):
413.#     if switch == 'on':
414.#         GPIO.output(5, GPIO.HIGH)
415.#     elif switch == 'off':
```

```
416. #         GPIO.output(5,GPIO.LOW)
417.

418. if __name__ == '__main__':
419. import time
420. import os
421. print("spidev version is ", spidev.__version__)
422. print("spidev device as show:")
423. #     os.system("ls /dev/spi*")
424.
425. led = Adeept_SPI_LedPixel(8, 255)                # Use MOSI for /dev/spidev0 to drive the lights

426. try:
427. if led.check_spi_state() != 0:
428. led.set_led_count(8)
429. led.set_all_led_color_data(255, 0, 0)
430. led.show()
431. time.sleep(0.5)
432. led.set_all_led_rgb_data([0, 255, 0])
433. led.show()
434. time.sleep(0.5)
435. led.set_all_led_color(0, 0, 255)
436. time.sleep(0.5)
437. led.set_all_led_rgb([0, 255, 255])
438. time.sleep(0.5)

439. led.set_led_count(12)
440. led.set_all_led_color_data(255, 255, 0)
441. for i in range(255):
442. led.set_led_brightness(i)
443. led.show()
444. time.sleep(0.005)
445. for i in range(255):
446. led.set_led_brightness(255-i)
447. led.show()
448. time.sleep(0.005)
449.
450. led.set_led_brightness(20)
451. while True:
452. for j in range(255):
453. for i in range(led.led_count):
454. led.set_led_rgb_data(i, led.wheel((round(i * 255 / led.led_count) + j)%256))
```

```
455. led.show()
456. time.sleep(0.002)
457. else:
458. led.led_close()
459. except KeyboardInterrupt:
460. led.led_close()
```