

Lesson 24 Alarm Light

24.1 Overview

In this lesson, we will explore how to use multi - threading techniques in Python to create specific effects with WS2812 LED lights. By leveraging multi - threading, we can manage the WS2812 LED - related tasks without disrupting the main thread's communication, ensuring smooth operation of the overall system.

24.2 Principle Introduction

The alarm light is based on the SPI interface and multi-threading technology to achieve the control of WS2812 LED lights and the display of various lighting effects. The specific principles are as follows:

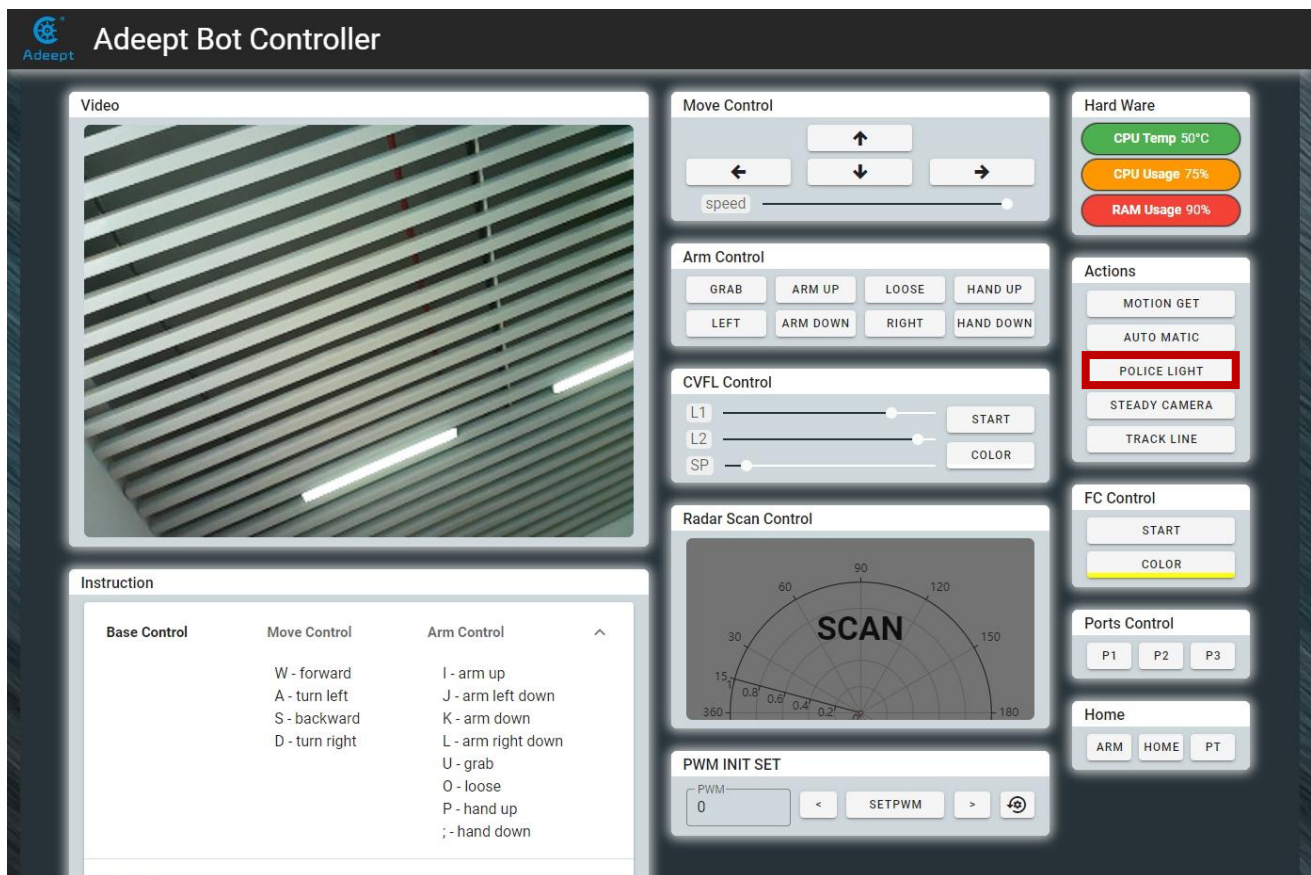
Communication Connection: The SPI communication protocol is used to establish a connection with the WS2812 LED lights. First, the SPI device is initialized and configured. The SPI communication channel is opened according to the set bus and device number, laying the foundation for the subsequent transmission of color data to the LED lights.

Color Processing: Users can set parameters such as the number of LED lights, color sequence, and brightness. The code will process the color data according to these parameters, converting the color values from RGB or HSV format into a format suitable for SPI transmission. Through bit operations and specific algorithms, the color information is transformed into data recognizable by the SPI, ensuring the accurate transmission of color data.

Multi-threading and Lighting Effects: Multi-threading technology is adopted to implement different lighting effects, such as the flashing of the alarm light and the breathing light effect. Each lighting effect has corresponding processing logic and runs in an independent thread. The start, pause, and resume of the threads are realized through a specific control mechanism. In this way, it can prevent the processing of lighting effects from blocking the main thread, ensuring that the program can handle other tasks simultaneously and guaranteeing the real-time performance and smoothness of the system.

24.3 Principle Introduction

1. Start the PiCar-Pro Robot. It may take about 30-50s to boot.
2. After PiCar-Pro 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: 192.168.3.31:5000. The web controller will then be displayed on the browser.



3. Click "POLICE LIGHT", and PiCar-Pro will flash lights of different colors.

Note: Currently, this module requires the use of SPI, and you need to turn on SPI.

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

24.4 Code

[RobotLight.py](#)

```
001 #!/usr/bin/env/python
002 # File name : RobotLight.py
```

```

003 # Website      : www.Adeept.com
004 # Author       : Adeept
005 # Date        : 2025/03/12
006 import time
007 from rpi_ws281x import *
008 import threading
009 import spidev
010 import numpy
011 from numpy import sin, cos, pi
012
013 def map(x, in_min, in_max, out_min, out_max):
014     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
015
016
017 class Adeept_SPI_LedPixel(threading.Thread):
018     def __init__(self, count = 8, bright = 255, sequence='GRB', bus = 0, device = 0, *args, **kwargs):
019         self.set_led_type(sequence)
020         self.set_led_count(count)
021         self.set_led_brightness(bright)
022         self.led_begin(bus, device)
023         self.lightMode = 'none'
024         self.colorBreathR = 0
025         self.colorBreathG = 0
026         self.colorBreathB = 0
027         self.breathSteps = 10
028         self.set_all_led_color(0,0,0)
029         super(Adeept_SPI_LedPixel, self).__init__(*args, **kwargs)
030         self.__flag = threading.Event()
031         self.__flag.clear()
032     def led_begin(self, bus = 0, device = 0):
033         self.bus = bus
034         self.device = device
035         try:
036             self.spi = spidev.SpiDev()
037             self.spi.open(self.bus, self.device)
038             self.spi.mode = 0
039             self.led_init_state = 1
040         except OSError:
041             print("Please check the configuration in /boot/firmware/config.txt.")
042             if self.bus == 0:
043                 print("You can turn on the 'SPI' in 'Interface Options' by using 'sudo raspi-config'.")
044                 print("Or make sure that 'dtparam=spi=on' is not commented, then reboot the Raspberry Pi. Otherwise spi0 will not be available.")
045             else:
046                 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))
047             self.led_init_state = 0
048
049
050
051     def check_spi_state(self):
052         return self.led_init_state
053
054     def spi_gpio_info(self):
055         if self.bus == 0:
056             print("SPI0-MOSI: GPIO10(W52812-PIN) SPI0-MISO: GPIO9 SPI0-SCLK: GPIO11 SPI0-CE0:
057 GPIO8 SPI0-CE1: GPIO7")
058         elif self.bus == 1:

```

```

059         print("SPI1-MOSI: GPIO20(W2812-PIN)  SPI1-MISO: GPIO19  SPI1-SCLK: GPIO21  SPI1-CE0:
060 GPIO18  SPI1-CE1: GPIO17  SPI0-CE1: GPIO16")
061         elif self.bus == 2:
062             print("SPI2-MOSI: GPIO41(W2812-PIN)  SPI2-MISO: GPIO40  SPI2-SCLK: GPIO42  SPI2-CE0:
063 GPIO43  SPI2-CE1: GPIO44  SPI2-CE1: GPIO45")
064         elif self.bus == 3:
065             print("SPI3-MOSI: GPIO2(W2812-PIN)  SPI3-MISO: GPIO1  SPI3-SCLK: GPIO3  SPI3-CE0:
066 GPIO0  SPI3-CE1: GPIO24")
067         elif self.bus == 4:
068             print("SPI4-MOSI: GPIO6(W2812-PIN)  SPI4-MISO: GPIO5  SPI4-SCLK: GPIO7  SPI4-CE0:
069 GPIO4  SPI4-CE1: GPIO25")
070         elif self.bus == 5:
071             print("SPI5-MOSI: GPIO14(W2812-PIN)  SPI5-MISO: GPIO13  SPI5-SCLK: GPIO15  SPI5-CE0:
072 GPIO12  SPI5-CE1: GPIO26")
073         elif self.bus == 6:
074             print("SPI6-MOSI: GPIO20(W2812-PIN)  SPI6-MISO: GPIO19  SPI6-SCLK: GPIO21  SPI6-CE0:
075 GPIO18  SPI6-CE1: GPIO27")
076
077     def led_close(self):
078         self.set_all_led_rgb([0,0,0])
079         self.spi.close()
080
081     def set_led_count(self, count):
082         self.led_count = count
083         self.led_color = [0,0,0] * self.led_count
084         self.led_original_color = [0,0,0] * self.led_count
085
086     def set_led_type(self, rgb_type):
087         try:
088             led_type = ['RGB', 'RBG', 'GRB', 'GBR', 'BRG', 'BGR']
089             led_type_offset = [0x06, 0x09, 0x12, 0x21, 0x18, 0x24]
090             index = led_type.index(rgb_type)
091             self.led_red_offset = (led_type_offset[index]>>4) & 0x03
092             self.led_green_offset = (led_type_offset[index]>>2) & 0x03
093             self.led_blue_offset = (led_type_offset[index]>>0) & 0x03
094             return index
095         except ValueError:
096             self.led_red_offset = 1
097             self.led_green_offset = 0
098             self.led_blue_offset = 2
099             return -1
100
101     def set_led_brightness(self, brightness):
102         self.led_brightness = brightness
103         for i in range(self.led_count):
104             self.set_led_rgb_data(i, self.led_original_color)
105
106     def set_ledpixel(self, index, r, g, b):
107         p = [0,0,0]
108         p[self.led_red_offset] = round(r * self.led_brightness / 255)
109         p[self.led_green_offset] = round(g * self.led_brightness / 255)
110         p[self.led_blue_offset] = round(b * self.led_brightness / 255)
111         self.led_original_color[index*3+self.led_red_offset] = r
112         self.led_original_color[index*3+self.led_green_offset] = g
113         self.led_original_color[index*3+self.led_blue_offset] = b
114         for i in range(3):

```

```

115         self.led_color[index*3+i] = p[i]
116
117     def setSomeColor_data(self, index, r, g, b):
118         self.set_ledpixel(index, r, g, b)
119
120     def set_led_rgb_data(self, index, color):
121         self.set_ledpixel(index, color[0], color[1], color[2])
122
123     def setSomeColor(self, index, r, g, b):
124         self.set_ledpixel(index, r, g, b)
125         self.show()
126
127     def set_led_rgb(self, index, color):
128         self.set_led_rgb_data(index, color)
129         self.show()
130
131     def set_all_led_color_data(self, r, g, b):
132         for i in range(self.led_count):
133             self.setSomeColor_data(i, r, g, b)
134
135     def set_all_led_rgb_data(self, color):
136         for i in range(self.led_count):
137             self.set_led_rgb_data(i, color)
138
139     def set_all_led_color(self, r, g, b):
140         for i in range(self.led_count):
141             self.setSomeColor_data(i, r, g, b)
142         self.show()
143
144     def set_all_led_rgb(self, color):
145         for i in range(self.led_count):
146             self.set_led_rgb_data(i, color)
147         self.show()
148
149     def write_ws2812_numpy8(self):
150         d = numpy.array(self.led_color).ravel()          #Converts data into a one-dimensional array
151         tx = numpy.zeros(len(d)*8, dtype=numpy.uint8)    #Each RGB color has 8 bits, each represented by
152         a uint8 type data
153         for ibit in range(8):                            #Convert each bit of data to the data that the
154         spi will send
155             tx[7-ibit::8]=((d>>ibit)&1)*0x78 + 0x80      #T0H=1,T0L=7, T1H=5,T1L=3   #0b11111000 mean
156             T1(0.78125us), 0b10000000 mean T0(0.15625us)
157             if self.led_init_state != 0:
158                 if self.bus == 0:
159                     self.spi.xfer(tx.tolist(), int(8/1.25e-6))    #Send color data at a frequency of
160             6.4Mhz
161                 else:
162                     self.spi.xfer(tx.tolist(), int(8/1.0e-6))      #Send color data at a frequency of
163             8Mhz
164
165     def write_ws2812_numpy4(self):
166         d=numpy.array(self.led_color).ravel()
167         tx=numpy.zeros(len(d)*4, dtype=numpy.uint8)
168         for ibit in range(4):
169             tx[3-ibit::4]=((d>>(2*ibit+1))&1)*0x60 + ((d>>(2*ibit+0))&1)*0x06 + 0x88
170             if self.led_init_state != 0:

```

```
171         if self.bus == 0:
172             self.spi.xfer(tx.tolist(), int(4/1.25e-6))
173         else:
174             self.spi.xfer(tx.tolist(), int(4/1.0e-6))
175
176     def show(self, mode = 1):
177         if mode == 1:
178             write_ws2812 = self.write_ws2812_numpy8
179         else:
180             write_ws2812 = self.write_ws2812_numpy4
181         write_ws2812()
182
183     def wheel(self, pos):
184         if pos < 85:
185             return [(255 - pos * 3), (pos * 3), 0]
186         elif pos < 170:
187             pos = pos - 85
188             return [0, (255 - pos * 3), (pos * 3)]
189         else:
190             pos = pos - 170
191             return [(pos * 3), 0, (255 - pos * 3)]
192
193     def hsv2rgb(self, h, s, v):
194         h = h % 360
195         rgb_max = round(v * 2.55)
196         rgb_min = round(rgb_max * (100 - s) / 100)
197         i = round(h / 60)
198         diff = round(h % 60)
199         rgb_adj = round((rgb_max - rgb_min) * diff / 60)
200         if i == 0:
201             r = rgb_max
202             g = rgb_min + rgb_adj
203             b = rgb_min
204         elif i == 1:
205             r = rgb_max - rgb_adj
206             g = rgb_max
207             b = rgb_min
208         elif i == 2:
209             r = rgb_min
210             g = rgb_max
211             b = rgb_min + rgb_adj
212         elif i == 3:
213             r = rgb_min
214             g = rgb_max - rgb_adj
215             b = rgb_max
216         elif i == 4:
217             r = rgb_min + rgb_adj
218             g = rgb_min
219             b = rgb_max
220         else:
221             r = rgb_max
222             g = rgb_min
223             b = rgb_max - rgb_adj
224         return [r, g, b]
225
226     def police(self):
```

```
227         self.lightMode = 'police'
228         self.resume()
229
230     def breath(self, R_input, G_input, B_input):
231         self.lightMode = 'breath'
232         self.colorBreathR = R_input
233         self.colorBreathG = G_input
234         self.colorBreathB = B_input
235         self.resume()
236
237     def resume(self):
238         self.__flag.set()
239
240     def pause(self):
241         self.lightMode = 'none'
242         self.set_all_led_color_data(0,0,0)
243         self.__flag.clear()
244
245     def breathProcessing(self):
246         while self.lightMode == 'breath':
247             for i in range(0,self.breathSteps):
248                 if self.lightMode != 'breath':
249                     break
250                 self.set_all_led_color(self.colorBreathR*i/self.breathSteps,
251 self.colorBreathG*i/self.breathSteps, self.colorBreathB*i/self.breathSteps)
252                 #self.show()
253                 time.sleep(0.03)
254             for i in range(0,self.breathSteps):
255                 if self.lightMode != 'breath':
256                     break
257                 self.set_all_led_color(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
258 self.colorBreathG-(self.colorBreathG*i/self.breathSteps), self.colorBreathB-
259 (self.colorBreathB*i/self.breathSteps))
260                 #self.show()
261                 time.sleep(0.03)
262     def policeProcessing(self):
263         while self.lightMode == 'police':
264             for i in range(0,3):
265                 self.set_all_led_color_data(0,0,255)
266                 self.show()
267                 time.sleep(0.05)
268                 self.set_all_led_color_data(0,0,0)
269                 self.show()
270                 time.sleep(0.05)
271             if self.lightMode != 'police':
272                 break
273             time.sleep(0.1)
274             for i in range(0,3):
275                 self.set_all_led_color_data(255,0,0)
276                 self.show()
277                 time.sleep(0.05)
278                 self.set_all_led_color_data(0,0,0)
279                 self.show()
280                 time.sleep(0.05)
281             time.sleep(0.1)
282
```

```
283
284     def lightChange(self):
285         if self.lightMode == 'none':
286             self.pause()
287         elif self.lightMode == 'police':
288             self.policeProcessing()
289         elif self.lightMode == 'breath':
290             self.breathProcessing()
291
292     def run(self):
293         while 1:
294             self.__flag.wait()
295             self.lightChange()
296             pass
297
298
299 if __name__ == '__main__':
300     import time
301     import os
302     print("spidev version is ", spidev.__version__)
303     print("spidev device as show:")
304
305     led = AdeepT_SPI_LedPixel(8, 255)          # Use MOSI for /dev/spidev0 to drive the lights
306
307     try:
308         if led.check_spi_state() != 0:
309             led.set_led_count(8)
310             led.set_all_led_color_data(255, 0, 0)
311             led.show()
312             time.sleep(0.5)
313             led.set_all_led_rgb_data([0, 255, 0])
314             led.show()
315             time.sleep(0.5)
316             led.set_all_led_color(0, 0, 255)
317             time.sleep(0.5)
318             led.set_all_led_rgb([0, 255, 255])
319             time.sleep(0.5)
320
321             led.set_led_count(12)
322             led.set_all_led_color_data(255, 255, 0)
323             for i in range(255):
324                 led.set_led_brightness(i)
325                 led.show()
326                 time.sleep(0.005)
327             for i in range(255):
328                 led.set_led_brightness(255-i)
329                 led.show()
330                 time.sleep(0.005)
331
332             led.set_led_brightness(20)
333             while True:
334                 for j in range(255):
335                     for i in range(led.led_count):
336                         led.set_led_rgb_data(i, led.wheel((round(i * 255 / led.led_count) + j)%256))
337                         led.show()
338                         time.sleep(0.002)
```



```
    else:  
        led.led_close()  
except KeyboardInterrupt:  
    led.led_close()
```