

Lesson 18 Alarm Light

18.1 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.

18.2 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:

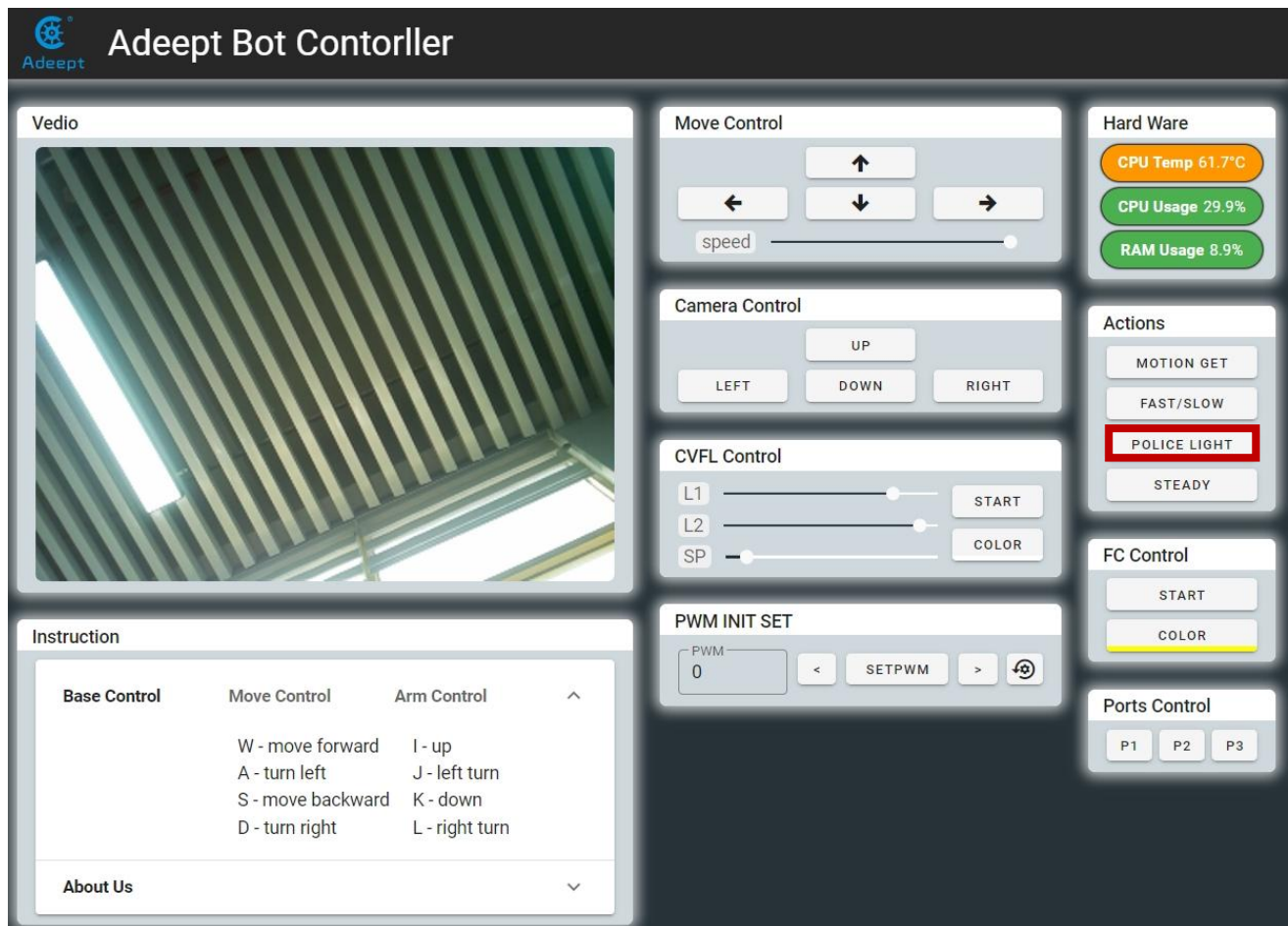
Communication Connection: The SPI communication protocol is used to establish a connection with the WS2812 LED lights. First, the SPI device is initialized and configured. The SPI communication channel is opened according to the set bus and device number, laying the foundation for the subsequent transmission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

18.3 Principle Introduction

1. Start the Adeept Raspclaws Robot. It may take about 30-50s to boot.
2. After Adeept Raspclaws 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 Adeept Raspclaws will flash lights of different colors.

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

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

18.4 Code

RobotLight.py

```
001 #!/usr/bin/env/python
002 # File name : RobotLight.py
003 # Website : www.Adeept.com
```

```

004 # Author      : Adeept
005 # Date        : 2025/04/14
006 import time
007 import sys
008 from gpiozero import PWMOutputDevice as PWM
009 import threading
010 import spidev
011 import numpy
012 from numpy import sin, cos, pi
013
014 def map(x, in_min, in_max, out_min, out_max):
015     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
016
017
018 class Adeept_SPI_LedPixel(threading.Thread):
019     def __init__(self, count = 8, bright = 255, sequence='GRB', bus = 0, device = 0, *args, **kwargs):
020         self.set_led_type(sequence)
021         self.set_led_count(count)
022         self.set_led_brightness(bright)
023         self.led_begin(bus, device)
024         self.lightMode = 'none'
025         self.colorBreathR = 0
026         self.colorBreathG = 0
027         self.colorBreathB = 0
028         self.breathSteps = 10
029         self.set_all_led_color(0,0,0)
030         super(Adeeft_SPI_LedPixel, self).__init__(*args, **kwargs)
031         self.__flag = threading.Event()
032         self.__flag.clear()
033     def led_begin(self, bus = 0, device = 0):
034         self.bus = bus
035         self.device = device
036         try:
037             self.spi = spidev.SpiDev()
038             self.spi.open(self.bus, self.device)
039             self.spi.mode = 0
040             self.led_init_state = 1
041         except OSError:
042             print("Please check the configuration in /boot/firmware/config.txt.")
043             if self.bus == 0:
044                 print("You can turn on the 'SPI' in 'Interface Options' by using 'sudo raspi-config'.")
045                 print("Or make sure that 'dtoverlay=spi=on' is not commented, then reboot the Raspberry Pi. Otherwise spi0 will not be available.")
046             else:
047                 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))
048             self.led_init_state = 0
049
050
051
052     def check_spi_state(self):
053         return self.led_init_state
054
055     def spi_gpio_info(self):
056         if self.bus == 0:
057             print("SPI0-MOSI: GPIO10(WS2812-PIN) SPI0-MISO: GPIO9 SPI0-SCLK: GPIO11 SPI0-CE0:
058 GPIO8 SPI0-CE1: GPIO7")
059         elif self.bus == 1:

```

```

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

```

```

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

```

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

```

228         self.lightMode = 'police'
229         self.resume()
230
231     def breath(self, R_input, G_input, B_input):
232         self.lightMode = 'breath'
233         self.colorBreathR = R_input
234         self.colorBreathG = G_input
235         self.colorBreathB = B_input
236         self.resume()
237
238     def resume(self):
239         self.__flag.set()
240
241
242
243     def pause(self):
244         self.lightMode = 'none'
245         self.set_all_led_color_data(0,0,0)
246         self.__flag.clear()
247
248     def breathProcessing(self):
249         while self.lightMode == 'breath':
250             for i in range(0,self.breathSteps):
251                 if self.lightMode != 'breath':
252                     break
253                 self.set_all_led_color(self.colorBreathR*i/self.breathSteps,
254 self.colorBreathG*i/self.breathSteps, self.colorBreathB*i/self.breathSteps)
255                 #self.show()
256                 time.sleep(0.03)
257             for i in range(0,self.breathSteps):
258                 if self.lightMode != 'breath':
259                     break
260                 self.set_all_led_color(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
261 self.colorBreathG-(self.colorBreathG*i/self.breathSteps), self.colorBreathB-
262 (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
302
303
304 if __name__ == '__main__':
305     import time
306     import os
307     print("spidev version is ", spidev.__version__)
308     print("spidev device as show:")
309
310     led = AdeepT_SPI_LedPixel(8, 255)          # Use MOSI for /dev/spidev0 to drive the lights
311
312     try:
313         if led.check_spi_state() != 0:
314             led.set_led_count(8)
315             led.set_all_led_color_data(255, 0, 0)
316             led.show()
317             time.sleep(0.5)
318             led.set_all_led_rgb_data([0, 255, 0])
319             led.show()
320             time.sleep(0.5)
321             led.set_all_led_color(0, 0, 255)
322             time.sleep(0.5)
323             led.set_all_led_rgb([0, 255, 255])
324             time.sleep(0.5)
325
326             led.set_led_count(12)
327             led.set_all_led_color_data(255, 255, 0)
328             for i in range(255):
329                 led.set_led_brightness(i)
330                 led.show()
331                 time.sleep(0.005)
332             for i in range(255):
333                 led.set_led_brightness(255-i)
334                 led.show()
335                 time.sleep(0.005)
336
337             led.set_led_brightness(20)
338             while True:
339                 for j in range(255):
```



```
340         for i in range(led.led_count):
341             led.set_led_rgb_data(i, led.wheel((round(i * 255 / led.led_count) + j)%256))
342             led.show()
343             time.sleep(0.002)
344         else:
345             led.pause()
346     except KeyboardInterrupt:
347         led.pause()
```