

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:

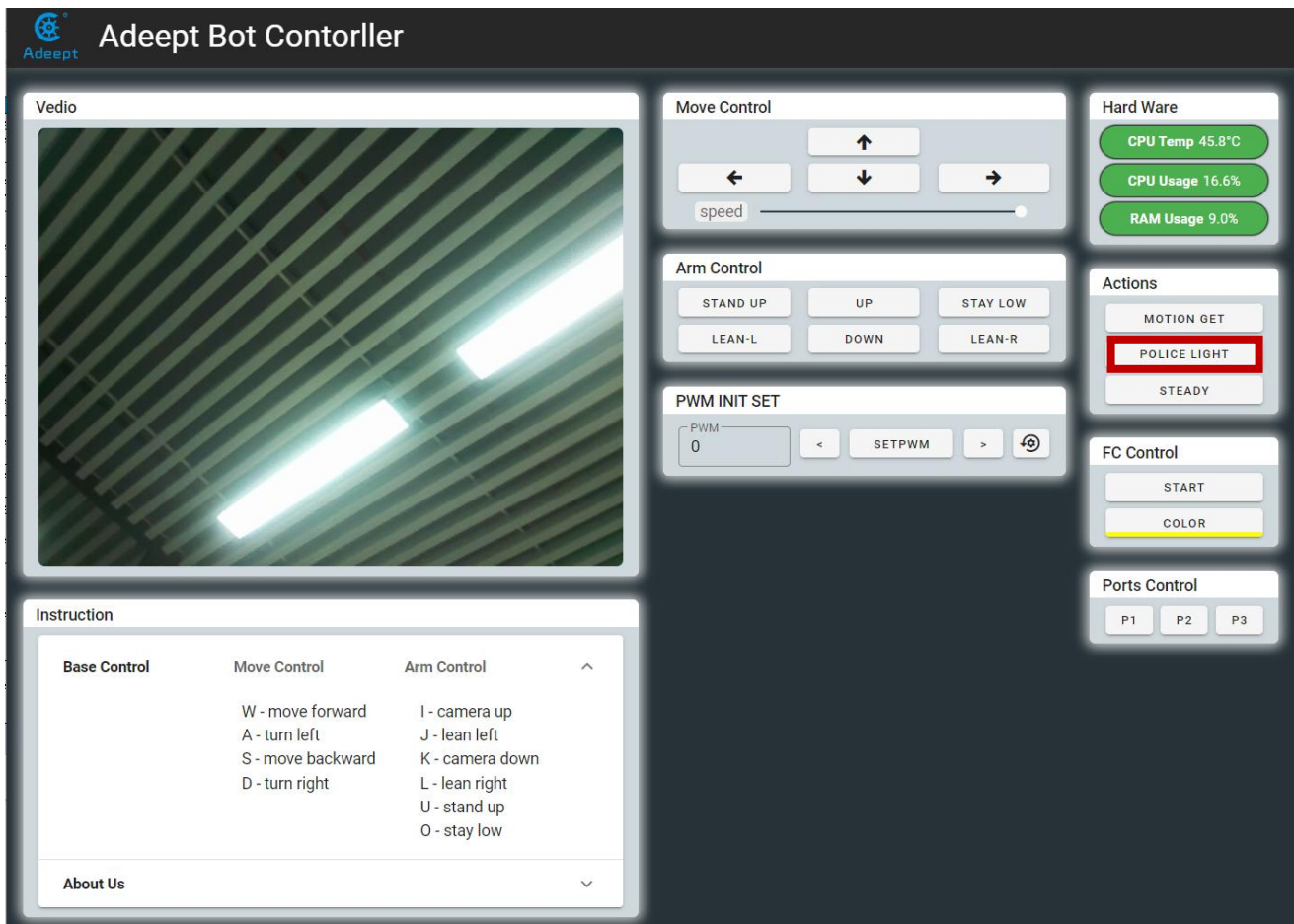
Communication Connection: The SPI communication protocol is used to establish a connection with the WS2812 LED lights. First, the SPI device is initialized and configured. The SPI communication channel is opened according to the set bus and device number, laying the foundation for the subsequent transmission 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_DarkPaw Robot. It may take about 30-50s to boot.
2. After Adeept_DarkPaw 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_DarkPaw 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.

18.4 Code

RobotLight.py

```
001 #!/usr/bin/env/python3
002 # File name : RobotLight.py
003 # Website : www.adeept.com
```

```

004 # Author      : Adeept
005 # Date        : 2025/04/25
006 import spidev
007 import threading
008 import numpy
009 from numpy import sin, cos, pi
010 import time
011 class Adeept_SPI_LedPixel(threading.Thread):
012     def __init__(self, count = 16, bright = 255, sequence='GRB', bus = 0, device = 0, *args, **kwargs):
013         self.set_led_type(sequence)
014         self.set_led_count(count)
015         self.set_led_brightness(bright)
016         self.led_begin(bus, device)
017         self.lightMode = 'none'
018         self.colorBreathR = 0
019         self.colorBreathG = 0
020         self.colorBreathB = 0
021         self.breathSteps = 10
022         self.set_all_led_color(0,0,0)
023         super(Adeeps_SPI_LedPixel, self).__init__(*args, **kwargs)
024         self.__flag = threading.Event()
025         self.__flag.clear()
026     def led_begin(self, bus = 0, device = 0):
027         self.bus = bus
028         self.device = device
029         try:
030             self.spi = spidev.SpiDev()
031             self.spi.open(self.bus, self.device)
032             self.spi.mode = 0
033             self.led_init_state = 1
034         except OSError:
035             print("Please check the configuration in /boot/firmware/config.txt.")
036             if self.bus == 0:
037                 print("You can turn on the 'SPI' in 'Interface Options' by using 'sudo raspi-config'.")
038                 print("Or make sure that 'dtoverlay=spi=on' is not commented, then reboot the Raspberry
039 Pi. Otherwise spi0 will not be available.")
040             else:
041                 print("Please add 'dtoverlay=spi{}-2cs' at the bottom of the /boot/firmware/config.txt,
042 then reboot the Raspberry Pi. otherwise spi{} will not be available.".format(self.bus, self.bus))
043             self.led_init_state = 0
044
045     def check_spi_state(self):
046         return self.led_init_state
047
048     def spi_gpio_info(self):
049         if self.bus == 0:
050             print("SPI0-MOSI: GPIO10(W52812-PIN) SPI0-MISO: GPIO9 SPI0-SCLK: GPIO11 SPI0-CE0:
051 GPIO8 SPI0-CE1: GPIO7")
052         elif self.bus == 1:
053             print("SPI1-MOSI: GPIO20(W52812-PIN) SPI1-MISO: GPIO19 SPI1-SCLK: GPIO21 SPI1-CE0:
054 GPIO18 SPI1-CE1: GPIO17 SPI1-CE1: GPIO16")
055         elif self.bus == 2:
056             print("SPI2-MOSI: GPIO41(W52812-PIN) SPI2-MISO: GPIO40 SPI2-SCLK: GPIO42 SPI2-CE0:
057 GPIO43 SPI2-CE1: GPIO44 SPI2-CE1: GPIO45")
058         elif self.bus == 3:
059             print("SPI3-MOSI: GPIO2(W52812-PIN) SPI3-MISO: GPIO1 SPI3-SCLK: GPIO3 SPI3-CE0:

```

```

060 GPIO0 SPI3-CE1: GPIO24")
061     elif self.bus == 4:
062         print("SPI4-MOSI: GPIO6(W52812-PIN) SPI4-MISO: GPIO5 SPI4-SCLK: GPIO7 SPI4-CE0:
063 GPIO4 SPI4-CE1: GPIO25")
064     elif self.bus == 5:
065         print("SPI5-MOSI: GPIO14(W52812-PIN) SPI5-MISO: GPIO13 SPI5-SCLK: GPIO15 SPI5-CE0:
066 GPIO12 SPI5-CE1: GPIO26")
067     elif self.bus == 6:
068         print("SPI6-MOSI: GPIO20(W52812-PIN) SPI6-MISO: GPIO19 SPI6-SCLK: GPIO21 SPI6-CE0:
069 GPIO18 SPI6-CE1: GPIO27")
070
071     def led_close(self):
072         self.set_all_led_rgb([0,0,0])
073         self.spi.close()
074
075     def set_led_count(self, count):
076         self.led_count = count
077         self.led_color = [0,0,0] * self.led_count
078         self.led_original_color = [0,0,0] * self.led_count
079
080     def set_led_type(self, rgb_type):
081         try:
082             led_type = ['RGB','RBG','GRB','GBR','BRG','BGR']
083             led_type_offset = [0x06,0x09,0x12,0x21,0x18,0x24]
084             index = led_type.index(rgb_type)
085             self.led_red_offset = (led_type_offset[index]>>4) & 0x03
086             self.led_green_offset = (led_type_offset[index]>>2) & 0x03
087             self.led_blue_offset = (led_type_offset[index]>>0) & 0x03
088             return index
089         except ValueError:
090             self.led_red_offset = 1
091             self.led_green_offset = 0
092             self.led_blue_offset = 2
093             return -1
094
095     def set_led_brightness(self, brightness):
096         self.led_brightness = brightness
097         for i in range(self.led_count):
098             self.set_led_rgb_data(i, self.led_original_color)
099
100     def set_ledpixel(self, index, r, g, b):
101         p = [0,0,0]
102         p[self.led_red_offset] = round(r * self.led_brightness / 255)
103         p[self.led_green_offset] = round(g * self.led_brightness / 255)
104         p[self.led_blue_offset] = round(b * self.led_brightness / 255)
105         self.led_original_color[index*3+self.led_red_offset] = r
106         self.led_original_color[index*3+self.led_green_offset] = g
107         self.led_original_color[index*3+self.led_blue_offset] = b
108         for i in range(3):
109             self.led_color[index*3+i] = p[i]
110
111     def set_led_color_data(self, index, r, g, b):
112         self.set_ledpixel(index, r, g, b)
113
114     def set_led_rgb_data(self, index, color):
115         self.set_ledpixel(index, color[0], color[1], color[2])

```

```

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

```

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

```

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

```

```

284         elif self.lightMode == 'breath':
285             self.breathProcessing()
286
287     def run(self):
288         while 1:
289             self.__flag.wait()
290             self.lightChange()
291             pass
292
293
294
295 if __name__ == '__main__':
296     import time
297     import os
298     print("spidev version is ", spidev.__version__)
299     print("spidev device as show:")
300     os.system("ls /dev/spi*")
301
302     led = Adeept_SPI_LedPixel(8, 255)          # Use MOSI for /dev/spidev0 to drive the lights
303     try:
304         if led.check_spi_state() != 0:
305             led.set_led_count(8)
306             led.set_all_led_color_data(255, 0, 0)
307             led.show()
308             time.sleep(0.5)
309             led.set_all_led_rgb_data([0, 255, 0])
310             led.show()
311             time.sleep(0.5)
312             led.set_all_led_color(0, 0, 255)
313             time.sleep(0.5)
314             led.set_all_led_rgb([0, 255, 255])
315             time.sleep(0.5)
316
317             led.set_led_count(12)
318             led.set_all_led_color_data(255, 255, 0)
319             for i in range(255):
320                 led.set_led_brightness(i)
321                 led.show()
322                 time.sleep(0.005)
323             for i in range(255):
324                 led.set_led_brightness(255-i)
325                 led.show()
326                 time.sleep(0.005)
327
328             led.set_led_brightness(20)
329             while True:
330                 for j in range(255):
331                     for i in range(led.led_count):
332                         led.set_led_rgb_data(i, led.wheel((round(i * 255 / led.led_count) + j)%256))
333                         led.show()
334                         time.sleep(0.002)
335             else:
336                 led.led_close()
337     except KeyboardInterrupt:
338         led.led_close()
339

```