

Lesson 2 Introducing the Warning Light Function

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

2.1 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.

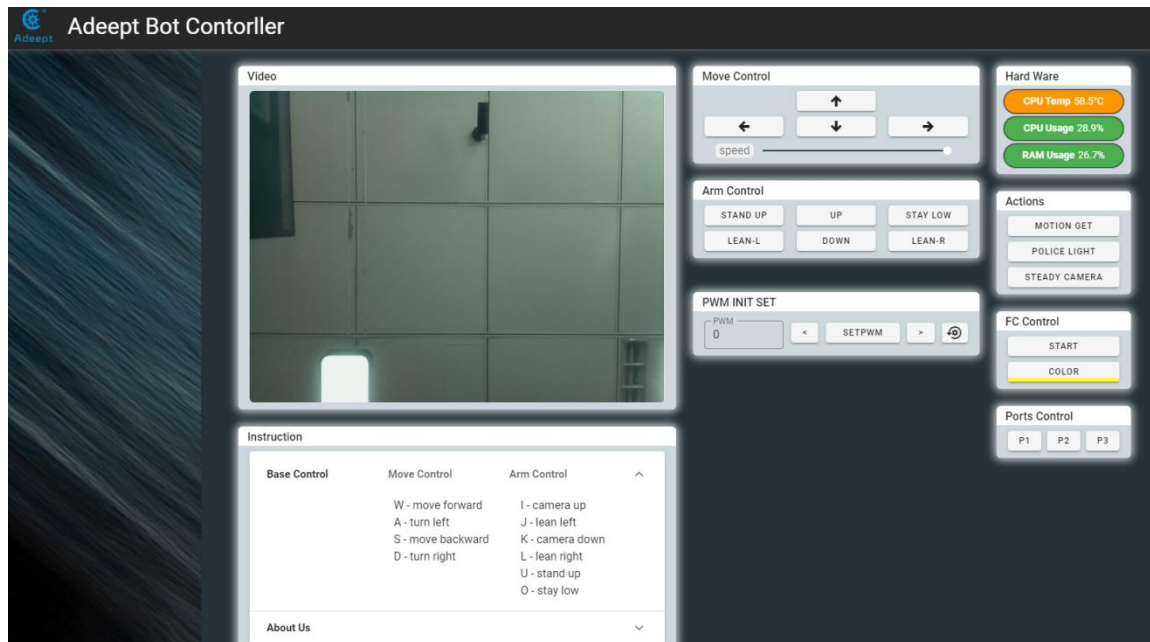
We use Python's threading library to provide thread-related work, and thread is the smallest unit of work in an application. For the current version of Python, there are no priorities, no thread groups, and threads cannot be stopped, suspended, resumed, or interrupted.

2.2 Running the Warning Light Program

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

2. After 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:

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



3. Click "[POLICE LIGHT](#)", and DarkPaw will flash lights of different colors.

4. Click "[POLICE LIGHT](#)" again to stop the function.

2.3 Main Program

For the complete code, please refer to the file [robotLight.py](#).

```

1. import time
2. import sys
3. from rpi_ws281x import *
4. import threading
5.
6.
7. """
8. Use the Threading module to create threads, inherit directly from threading.Thread, and then o
   verride the __init__ method and the run method
9. """
10. class RobotLight(threading.Thread):
11.     def __init__(self, *args, **kwargs):

```

```

12.      """
13.      Here initialize some settings about LED lights
14.      """
15.      self.LED_COUNT      = 16      # Number of LED pixels.
16.      self.LED_PIN        = 12      # GPIO pin connected to the pixels (18 uses PWM!)
17.
18.      self.LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
19.      self.LED_DMA        = 10      # DMA channel to use for generating signal (try 1
20.      0)
21.      self.LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
22.      self.LED_INVERT     = False    # True to invert the signal (when using NPN transistor le
23.      vel shift)
24.      self.LED_CHANNEL    = 0       # set to '1' for GPIOs 13, 19, 41, 45 or 53
25.
26.      """
27.
28.      Set the brightness of the three RGB color channels, no need to change here, these
29.      values will be automatically set after the subsequent call of the breathing light function
30.
31.      """
32.
33.      The mode variable, 'none' will make the thread block and hang, the light will not c
34.      hange;
35.
36.      'police' is a police light mode, red and blue flash alternately;
37.
38.      'breath' breathing light, you can set the specified color.
39.
40.      """
41.      self.lightMode = 'none'       #'none' 'police' 'breath'
42.
43.
44.      # Create NeoPixel object with appropriate configuration.
45.      self.strip = Adafruit_NeoPixel(self.LED_COUNT, self.LED_PIN, self.LED_FREQ_HZ,
46.                                     self.LED_DMA, self.LED_INVERT, self.LED_BRIGHTN
47.      ESS,
48.                                     self.LED_CHANNEL)
49.
50.      # Intialize the library (must be called once before other functions).
51.      self.strip.begin()
52.
53.
54.      super(RobotLight, self).__init__(*args, **kwargs)
55.      self.__flag = threading.Event()
56.      self.__flag.clear()

```

```
48.
49.     # Define functions which animate LEDs in various ways.
50.     def setColor(self, R, G, B):
51.         """
52.         Set the color of all lights
53.         """
54.         color = Color(int(R),int(G),int(B))
55.         for i in range(self.strip.numPixels()):
56.             self.strip.setPixelColor(i, color)
57.             self.strip.show()
58.
59.     def setSomeColor(self, R, G, B, ID):
60.         """
61.         Set the color of some lamps, the ID is the array of the serial number of this lamp
62.         """
63.         color = Color(int(R),int(G),int(B))
64.         #print(int(R),' ',int(G),' ',int(B))
65.         for i in ID:
66.             self.strip.setPixelColor(i, color)
67.             self.strip.show()
68.
69.     def pause(self):
70.         """
71.         Call this function, set __flag to False, block the thread
72.         """
73.         self.lightMode = 'none'
74.         self.setColor(0,0,0)
75.         self.__flag.clear()
76.
77.     def resume(self):
78.         """
79.         Call this function, set __flag to True to start the thread
80.         """
81.         self.__flag.set()
82.
83.     def police(self):
84.         """
85.         Call this function to turn on the police light mode
86.         """
87.         self.lightMode = 'police'
88.         self.resume()
89.
```

```

90.
91.     def policeProcessing(self):
92.         """
93.             The specific realization of the police light mode
94.         """
95.         while self.lightMode == 'police':
96.             """
97.             Blue flashes 3 times
98.             """
99.             for i in range(0,3):
100.                 self.setSomeColor(0,0,255,[0,1,2,3,4,5,6,7,8,9,10,11])
101.                 time.sleep(0.05)
102.                 self.setSomeColor(0,0,0,[0,1,2,3,4,5,6,7,8,9,10,11])
103.                 time.sleep(0.05)
104.             if self.lightMode != 'police':
105.                 break
106.             time.sleep(0.1)
107.             """
108.             Red flashes 3 times
109.             """
110.             for i in range(0,3):
111.                 self.setSomeColor(255,0,0,[0,1,2,3,4,5,6,7,8,9,10,11])
112.                 time.sleep(0.05)
113.                 self.setSomeColor(0,0,0,[0,1,2,3,4,5,6,7,8,9,10,11])
114.                 time.sleep(0.05)
115.             time.sleep(0.1)
116.
117.     def breath(self, R_input, G_input, B_input):
118.         """
119.             Call this function to turn on the breathing light mode, you need to enter three pa
rameters, namely the brightness of the RGB three color channels, as the color when the bright
ness of the breathing lamp is maximum
120.         """
121.         self.lightMode = 'breath'
122.         self.colorBreathR = R_input
123.         self.colorBreathG = G_input
124.         self.colorBreathB = B_input
125.         self.resume()
126.
127.     def breathProcessing(self):
128.         """
129.             Specific realization method of breathing lamp

```

```

130.         """
131.         while self.lightMode == 'breath':
132.             """
133.             All lights gradually brighten
134.             """
135.             for i in range(0,self.breathSteps):
136.                 if self.lightMode != 'breath':
137.                     break
138.                 self.setColor(self.colorBreathR*i/self.breathSteps,
139.                               self.colorBreathG*i/self.breathSteps,
140.                               self.colorBreathB*i/self.breathSteps)
141.                 time.sleep(0.03)
142.             """
143.             All lights are getting darker
144.             """
145.             for i in range(0,self.breathSteps):
146.                 if self.lightMode != 'breath':
147.                     break
148.                 self.setColor(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
149.                               self.colorBreathG-(self.colorBreathG*i/self.breathSteps),
150.                               self.colorBreathB-(self.colorBreathB*i/self.breathSteps))
151.                 time.sleep(0.03)
152.
153.     def lightChange(self):
154.         """
155.         This function is used to select the task to perform
156.         """
157.         if self.lightMode == 'none':
158.             self.pause()
159.         elif self.lightMode == 'police':
160.             self.policeProcessing()
161.         elif self.lightMode == 'breath':
162.             self.breathProcessing()
163.
164.     def run(self):
165.         """
166.         Functions for multi-threaded tasks
167.         """
168.         while 1:
169.             self.__flag.wait()
170.             self.lightChange()
171.         pass

```

```

172.
173. if __name__ == '__main__':
174.     RL=RobotLight() # Instantiate the object that controls the LED light
175.     RL.start()      # Start thread
176.
177.     """
178.     Start breathing light mode and stop after 15 seconds
179.     """
180.     RL.breath(70,70,255)
181.     time.sleep(15)
182.     RL.pause()
183.
184.     """
185.     Pause for 2 seconds
186.     """
187.     time.sleep(2)
188.
189.     """
190.     Start the police light mode and stop after 15 seconds
191.     """
192.     RL.police()
193.     time.sleep(15)
194.     RL.pause()

```

2.4 Warning Lights or Breathing Lights in Other Projects

When your project needs to use LED lights for warning lights or breathing lights, you don't need to completely rewrite the code above but just copy *robotLight.py* in the *server* folder in the robot's package to the same folder of your project, and use the following code to make a warning light or breathing light:

```

1. import robotLight
2.
3. RL=robotLight.RobotLight() # Instantiate the object that controls the LED light
4. RL.start()      # Start thread
5.
6. """

```

```
7. Start breathing light mode and stop after 15 seconds
8. '''
9. RL.breath(70,70,255)
10. time.sleep(15)
11. RL.pause()
12.
13. ....
14. Pause for 2 seconds
15. '''
16. time.sleep(2)
17.
18. ....
19. Start the police light mode and stop after 15 seconds
20. '''
21. RL.police()
22. time.sleep(15)
23. RL.pause()
```