# Lesson 2 Introducing the Warning Light Function

In this lesson, we will learn how to use PiCar-b'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.

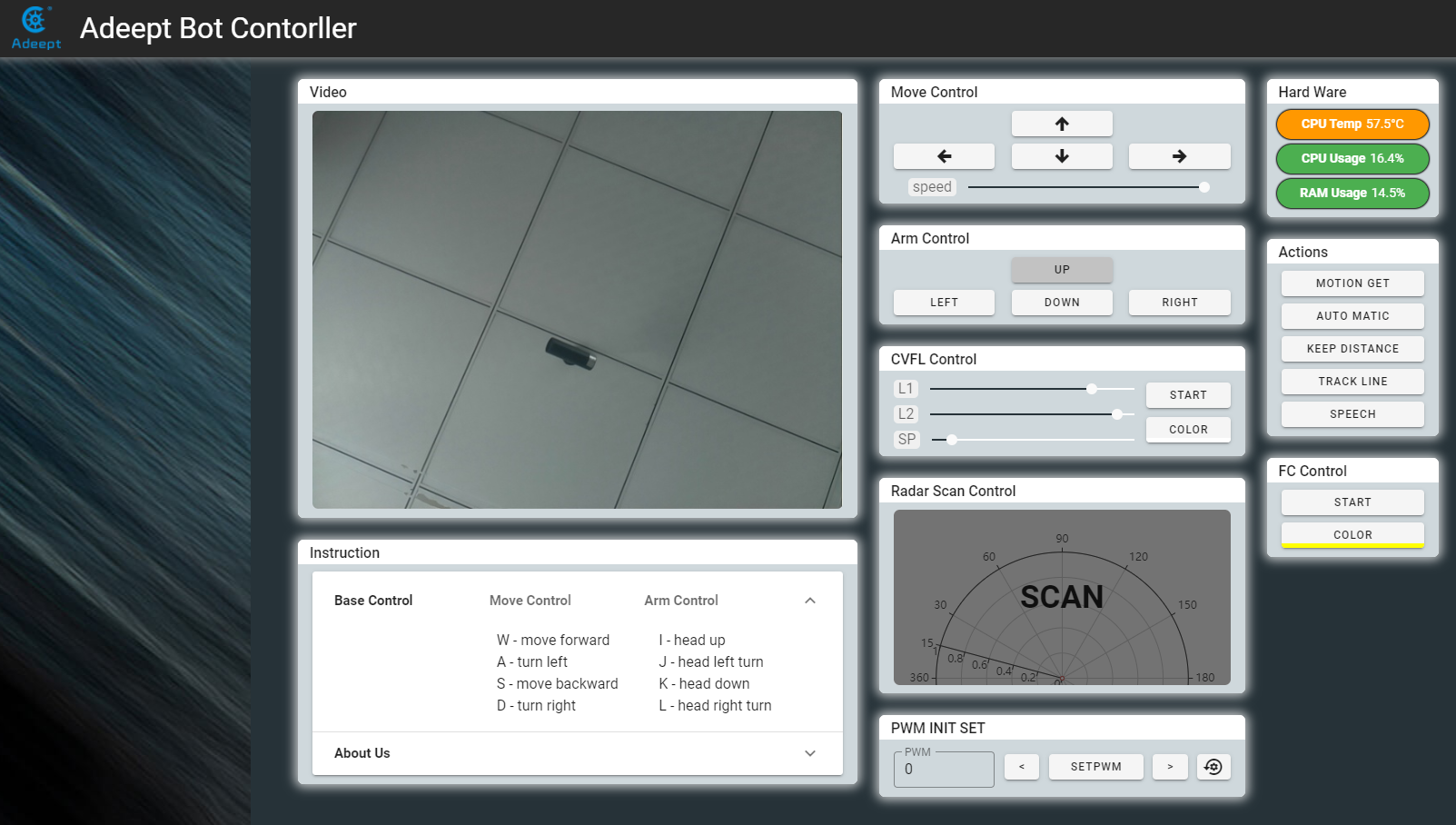
3. It's more convenient to call an encapsulated multi-threaded task, similar to the non-blocking control method – in other words, the control of the servo is encapsulated by multi-threading.

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 PiCar-b Robot. It may take about 30-50s to boot.

2. After PiCar-b 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 PiCar-b 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  7. ''''' 8. Use the Threading module to create threads, inherit directly from threading.Thread, and then override 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. self.LED\_FREQ\_HZ    = 800000  # LED signal frequency in hertz (usually 800khz) 18. self.LED\_DMA        = 10      # DMA channel to use for generating signal (try 10) 19. self.LED\_BRIGHTNESS = 255    # Set to 0 for darkest and 255 for brightest 20. self.LED\_INVERT  = False   # True to invert the signal (when using NPN transistor level shift) 21. self.LED\_CHANNEL    = 0    # set to '1' for GPIOs 13, 19, 41, 45 or 53 23. ''''' 24. Set the brightness of the three RGB color channels, no need to change here, these values will be automatically set after the subsequent call of the breathing light function 25. ''' 26. self.colorBreathR = 0 27. self.colorBreathG = 0 28. self.colorBreathB = 0 29. self.breathSteps = 10 31. ''''' 32. The mode variable, 'none' will make the thread block and hang, the light will not change; 33. 'police' is a police light mode, red and blue flash alternately; 34. 'breath' breathing light, you can set the specified color. 35. ''' 36. self.lightMode = 'none'     #'none' 'police' 'breath' 38. # Create NeoPixel object with appropriate configuration. 39. self.strip = Adafruit\_NeoPixel(self.LED\_COUNT, self.LED\_PIN, self.LED\_FREQ\_HZ, 40. self.LED\_DMA, self.LED\_INVERT, self.LED\_BRIGHTNESS, 41. self.LED\_CHANNEL) 42. # Intialize the library (must be called once before other functions). 43. self.strip.begin() 45. super(RobotLight, self).\_\_init\_\_(\*args, \*\*kwargs) 46. self.\_\_flag = threading.Event() 47. self.\_\_flag.clear() 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() 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() 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() 77. **def** resume(self): 78. ''''' 79. Call this function, set \_\_flag to True to start the thread 80. ''' 81. self.\_\_flag.set() 83. **def** police(self): 84. ''''' 85. Call this function to turn on the police light mode 86. ''' 87. self.lightMode = 'police' 88. self.resume()  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) 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 parameters, namely the brightness of the RGB three color channels, as the color when the brightness 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() 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) 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() 164. **def** run(self): 165. ''''' 166. Functions for multi-threaded tasks 167. ''' 168. **while** 1: 169. self.\_\_flag.wait() 170. self.lightChange() 171. **pass** 173. **if** \_\_name\_\_ == '\_\_main\_\_': 174. RL=RobotLight() # Instantiate the object that controls the LED light 175. RL.start()      # Start thread 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() 184. ''''' 185. Pause for 2 seconds 186. ''' 187. time.sleep(2) 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 3. RL=robotLight.RobotLight()  # Instantiate the object that controls the LED light 4. RL.start()      # Start thread 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() 13. ''''' 14. Pause for 2 seconds 15. ''' 16. time.sleep(2) 18. ''''' 19. Start the police light mode and stop after 15 seconds 20. ''' 21. RL.police() 22. time.sleep(15) 23. RL.pause() |