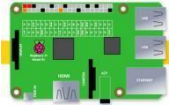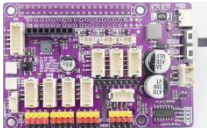# Lesson 9 How to Control WS2812 LED

In this lesson, we will learn how to control WS2812 LED.

9.1 Components Used in This Course

Components	Quantity	Picture
Raspberry Pi	1	
Adeept Robot HAT v3.1	1	
3 pin cable	1	
WS2812 RGB LED	1	

9.2 Introduction of WS2812 LED

WS2812 LED module is a low-power RGB tri-color lamp with integrated current control chip. Its appearance is the same as a 5050LED lamp bead, and each element is a pixel. The pixel contains an intelligent digital interface data latch signal shaping amplifier driving circuit, and also contains a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively guarantees that the color of the pixel light is highly consistent.

WS2812 LED is a very commonly used module on our robot products. There are three WS2812 LEDS on each module. Pay attention to the direction of the signal line when connecting. **The signal line needs to be connected to the "IN" port of WS2812 LED after being led from the Raspberry Pi. When the next WS2812 LED needs to be connected, we connect a signal wire drawn from the "OUT" port of the previous WS2812 LED with the "IN" port of the next WS2812 LED.**

When using the Raspberry Pi to install the driver board Adeept Robot HAT v3.1, the WS2812 LED can be connected to the WS2812 interface on the Adeept Robot HAT v3.1 using a 3pin cable.

We use SPI (Serial Peripheral Interface) communication protocol to control the WS2812 LED. You can learn about it through the following ways.
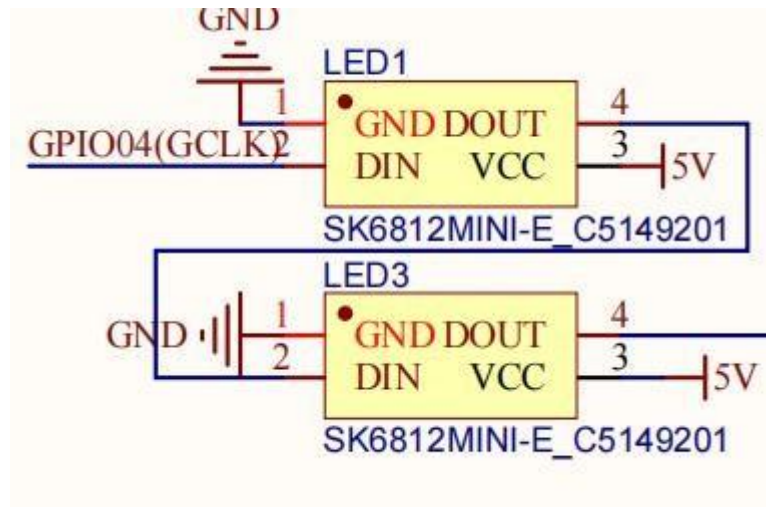
If you connect the WS2812 LED to the WS2812 interface of Adeept Robot HAT v3.1, the signal line is equivalent to connecting to the GPIO 10 of the Raspberry Pi.

9.3 Wiring Diagram (Circuit Diagram)

When the WS2812 LED is in use, the IN port needs to be connected to the WS2812 port on the Adeept Robot HAT v3.1 driver board.

Adeept Robot HAT v3.1 has 2 WS2812 LEDs on board, which are located at the front of the GPIO10 pin. When you use the WS2812 Port interface to connect one or more WS2812 LED

modules, the first and second (0 and 1) in the code control the two WS2812LEDs on the board.

Starting from the 3rd one is used to control the extended WS2812LED light. (2,3,4,...)

9.4 How to Control WS2812 LED

Environment settings

Before you run your python code, check that the spidev library exists.

Enter the following command to install.

```
pip list
```

The spidev is installed on Raspberry PI by default. As shown in the figure below.

If "spidev" does not exist, install the "spidev" dependencies with the following command

sudo pip3 install spidev --break-system-packages

Run code

1. Remotely log in to the Raspberry Pi terminal.

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 29 08:17:49 2020 from 192.168.3.208

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $
```

2. Enter the command and press Enter to enter the folder where the program is located:

cd adeept_rasptank2/examples/

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ cd adeept_picar-b2/examples/
pi@raspberrypi:~/adeept_picar-b2/examples $
```

3. View the contents of the current directory file:

ls

```
pi@raspberrypi:~/adeept_rasptank2/examples $ ls
01_LED.py      03_Servo.py   05_Spi_WS2812.py   06_Ultrasonic.py
02_Buzzer.py   04_Motor.py   05_WS2812.py        07_TrackingLine.py
```

4. Enter the command and press Enter to run the program:

sudo killall python3

```
pi@raspberrypi:~/adeept_picar-b/server $
pi@raspberrypi:~/adeept_picar-b/server $ sudo killall python3
```

If an error is reported when running the 06_ws2812.py program, please check the **Q&A** of this tutorial.

sudo python3 05_Spi_WS2812.py

```
pi@raspberrypi:~/adeept_rasptank2/examples $ sudo python3 05_Spi_WS2812.py
spidev version is  3.5
spidev device as show:
/dev/spidev0.0  /dev/spidev0.1
```

5. After running the program successfully, you will observe that the WS2812 alternately flashing lights of different colors.

6. When you want to terminate the running program, you can press the shortcut key "Ctrl + C" on the keyboard.

9.5 Code

For the complete code, please refer to the file 05_Spi_WS2812.py.

```
1.   import spidev

2.   import threading

3.   import numpy

4.   from numpy import sin, cos, pi

5.   import time

6.   class Adeept_SPI_LedPixel(threading.Thread):

7.   def __init__(self, count = 8, bright = 255, sequence='GRB', bus = 0, device = 0, *args,
     **kwargs):

8.   self.set_led_type(sequence)

9.   self.set_led_count(count)

10.  self.set_led_brightness(bright)

11.  self.led_begin(bus, device)

12.  self.lightMode = 'none'
```

```
13. self.colorBreathR = 0

14. self.colorBreathG = 0

15. self.colorBreathB = 0

16. self.breathSteps = 10

17. #self.spi_gpio_info()

18. self.set_all_led_color(0,0,0)

19. super(Adeept_SPI_LedPixel, self).__init__(*args, **kwargs)

20. self.__flag = threading.Event()

21. self.__flag.clear()

22. def led_begin(self, bus = 0, device = 0):

23. self.bus = bus

24. self.device = device

25. try:

26. self.spi = spidev.SpiDev()

27. self.spi.open(self.bus, self.device)

28. self.spi.mode = 0

29. self.led_init_state = 1

30. except OSError:

31. print("Please check the configuration in /boot/firmware/config.txt.")

32. if self.bus == 0:

33. print("You can turn on the 'SPI' in 'Interface Options' by using 'sudo raspi-config'.")

34. print("Or make sure that 'dtparam=spi=on' is not commented, then reboot the Raspberry Pi.
    Otherwise spi0 will not be available.")

35. else:

36. 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))

37. self.led_init_state = 0

38.

39. def check_spi_state(self):

40. return self.led_init_state

41.
```

```
42. def spi_gpio_info(self):

43. if self.bus == 0:

44. print("SPI0-MOSI: GPIO10(WS2812-PIN)  SPI0-MISO: GPIO9  SPI0-SCLK: GPIO11  SPI0-CE0:
    GPIO8  SPI0-CE1: GPIO7")

45. elif self.bus == 1:

46. print("SPI1-MOSI: GPIO20(WS2812-PIN)   SPI1-MISO: GPIO19  SPI1-SCLK: GPIO21  SPI1-CE0:
    GPIO18  SPI1-CE1: GPIO17  SPI0-CE1: GPIO16")

47. elif self.bus == 2:

48. print("SPI2-MOSI: GPIO41(WS2812-PIN)   SPI2-MISO: GPIO40  SPI2-SCLK: GPIO42  SPI2-CE0:
    GPIO43  SPI2-CE1: GPIO44  SPI2-CE1: GPIO45")

49. elif self.bus == 3:

50. print("SPI3-MOSI: GPIO2(WS2812-PIN)  SPI3-MISO: GPIO1  SPI3-SCLK: GPIO3  SPI3-CE0:
    GPIO0  SPI3-CE1: GPIO24")

51. elif self.bus == 4:

52. print("SPI4-MOSI: GPIO6(WS2812-PIN)  SPI4-MISO: GPIO5  SPI4-SCLK: GPIO7  SPI4-CE0:
    GPIO4  SPI4-CE1: GPIO25")

53. elif self.bus == 5:

54. print("SPI5-MOSI: GPIO14(WS2812-PIN)  SPI5-MISO: GPIO13  SPI5-SCLK: GPIO15  SPI5-CE0:
    GPIO12  SPI5-CE1: GPIO26")

55. elif self.bus == 6:

56. print("SPI6-MOSI: GPIO20(WS2812-PIN)  SPI6-MISO: GPIO19  SPI6-SCLK: GPIO21  SPI6-CE0:
    GPIO18  SPI6-CE1: GPIO27")

57.

58. def led_close(self):

59. self.set_all_led_rgb([0,0,0])

60. self.spi.close()

61.

62. def set_led_count(self, count):

63. self.led_count = count

64. self.led_color = [0,0,0] * self.led_count

65. self.led_original_color = [0,0,0] * self.led_count
```

```
66.

67. def set_led_type(self, rgb_type):

68. try:

69. led_type = ['RGB','RBG','GRB','GBR','BRG','BGR']

70. led_type_offset = [0x06,0x09,0x12,0x21,0x18,0x24]

71. index = led_type.index(rgb_type)

72. self.led_red_offset = (led_type_offset[index]>>4) & 0x03

73. self.led_green_offset = (led_type_offset[index]>>2) & 0x03

74. self.led_blue_offset = (led_type_offset[index]>>0) & 0x03

75. return index

76. except ValueError:

77. self.led_red_offset = 1

78. self.led_green_offset = 0

79. self.led_blue_offset = 2

80. return -1

81.

82. def set_led_brightness(self, brightness):

83. self.led_brightness = brightness

84. for i in range(self.led_count):

85. self.set_led_rgb_data(i, self.led_original_color)

86.

87. def set_ledpixel(self, index, r, g, b):

88. p = [0,0,0]

89. p[self.led_red_offset] = round(r * self.led_brightness / 255)

90. p[self.led_green_offset] = round(g * self.led_brightness / 255)

91. p[self.led_blue_offset] = round(b * self.led_brightness / 255)

92. self.led_original_color[index*3+self.led_red_offset] = r

93. self.led_original_color[index*3+self.led_green_offset] = g

94. self.led_original_color[index*3+self.led_blue_offset] = b

95. for i in range(3):

96. self.led_color[index*3+i] = p[i]
```

```
97. def set_led_color_data(self, index, r, g, b):

98. self.set_ledpixel(index, r, g, b)

99.

100. def set_led_rgb_data(self, index, color):

101. self.set_ledpixel(index, color[0], color[1], color[2])

102.

103. def set_led_color(self, index, r, g, b):

104. self.set_ledpixel(index, r, g, b)

105. self.show()

106.

107. def set_led_rgb(self, index, color):

108. self.set_led_rgb_data(index, color)

109. self.show()

110.

111. def set_all_led_color_data(self, r, g, b):

112. for i in range(self.led_count):

113. self.set_led_color_data(i, r, g, b)

114.

115. def set_all_led_rgb_data(self, color):

116. for i in range(self.led_count):

117. self.set_led_rgb_data(i, color)

118.

119. def set_all_led_color(self, r, g, b):

120. for i in range(self.led_count):

121. self.set_led_color_data(i, r, g, b)

122. self.show()

123.

124. def set_all_led_rgb(self, color):

125. for i in range(self.led_count):

126. self.set_led_rgb_data(i, color)
```

```
127.self.show()

128.

129.def write_ws2812_numpy8(self):

130.d = numpy.array(self.led_color).ravel()        #Converts data into a one-dimensional array

131.tx = numpy.zeros(len(d)*8, dtype=numpy.uint8)  #Each RGB color has 8 bits, each represented
    by a uint8 type data

132.for ibit in range(8):                          #Convert each bit of data to the data that
    the spi will send

133.#tx[7-ibit::8]=((d>>ibit)&1)*0x3E + 0xC0   #T0H=2,T0L=6, T1H=7,T1L=1   #0b11111110 mean
    T1(1.09375us), 0b11000000 mean T0(0.3125us)

134.#tx[7-ibit::8]=((d>>ibit)&1)*0x3C + 0xC0   #T0H=2,T0L=6, T1H=6,T1L=2   #0b11111100 mean
    T1(0.9375us),  0b11000000 mean T0(0.3125us)

135.#tx[7-ibit::8]=((d>>ibit)&1)*0x38 + 0xC0   #T0H=2,T0L=6, T1H=5,T1L=3   #0b11111000 mean
    T1(0.78125us), 0b11000000 mean T0(0.3125us)

136.#tx[7-ibit::8]=((d>>ibit)&1)*0x30 + 0xC0   #T0H=2,T0L=6, T1H=4,T1L=4   #0b11110000 mean
    T1(0.625us),   0b11000000 mean T0(0.3125us)

137.#tx[7-ibit::8]=((d>>ibit)&1)*0x20 + 0xC0   #T0H=2,T0L=6, T1H=3,T1L=5   #0b11100000 mean
    T1(0.46875us), 0b11000000 mean T0(0.3125us)

138.#tx[7-ibit::8]=((d>>ibit)&1)*0x7E + 0x80   #T0H=1,T0L=7, T1H=7,T1L=1   #0b11111110 mean
    T1(0.09375us), 0b10000000 mean T0(0.15625us)

139.#tx[7-ibit::8]=((d>>ibit)&1)*0x7C + 0x80   #T0H=1,T0L=7, T1H=6,T1L=2   #0b11111100 mean
    T1(0.9375us),  0b10000000 mean T0(0.15625us)

140.#tx[7-ibit::8]=((d>>ibit)&1)*0x78 + 0x80   #T0H=1,T0L=7, T1H=5,T1L=3   #0b11111000 mean
    T1(0.78125us), 0b10000000 mean T0(0.15625us)

141.#tx[7-ibit::8]=((d>>ibit)&1)*0x70 + 0x80   #T0H=1,T0L=7, T1H=4,T1L=4   #0b11110000 mean
    T1(0.625us),   0b10000000 mean T0(0.15625us)

142.#tx[7-ibit::8]=((d>>ibit)&1)*0x60 + 0x80   #T0H=1,T0L=7, T1H=3,T1L=5   #0b11100000 mean
    T1(0.46875us), 0b10000000 mean T0(0.15625us)

143.tx[7-ibit::8]=((d>>ibit)&1)*0x78 + 0x80    #T0H=1,T0L=7, T1H=5,T1L=3   #0b11111000 mean
    T1(0.78125us), 0b10000000 mean T0(0.15625us)

144.if self.led_init_state != 0:
```

```
145. if self.bus == 0:
146. self.spi.xfer(tx.tolist(), int(8/1.25e-6))        #Send color data at a frequency of 6.4Mhz
147. else:
148. self.spi.xfer(tx.tolist(), int(8/1.0e-6))          #Send color data at a frequency of 8Mhz
149.
150. def write_ws2812_numpy4(self):
151. d=numpy.array(self.led_color).ravel()
152. tx=numpy.zeros(len(d)*4, dtype=numpy.uint8)
153. for ibit in range(4):
154. tx[3-ibit::4]=((d>>(2*ibit+1))&1)*0x60 + ((d>>(2*ibit+0))&1)*0x06 + 0x88
155. if self.led_init_state != 0:
156. if self.bus == 0:
157. self.spi.xfer(tx.tolist(), int(4/1.25e-6))
158. else:
159. self.spi.xfer(tx.tolist(), int(4/1.0e-6))
160.
161. def show(self, mode = 1):
162. if mode == 1:
163. write_ws2812 = self.write_ws2812_numpy8
164. else:
165. write_ws2812 = self.write_ws2812_numpy4
166. write_ws2812()
167.
168. def wheel(self, pos):
169. if pos < 85:
170. return [(255 - pos * 3), (pos * 3), 0]
171. elif pos < 170:
172. pos = pos - 85
173. return [0, (255 - pos * 3), (pos * 3)]
174. else:
175. pos = pos - 170
```

```
176. return [(pos * 3), 0, (255 - pos * 3)]
177.
178. def hsv2rgb(self, h, s, v):
179. h = h % 360
180. rgb_max = round(v * 2.55)
181. rgb_min = round(rgb_max * (100 - s) / 100)
182. i = round(h / 60)
183. diff = round(h % 60)
184. rgb_adj = round((rgb_max - rgb_min) * diff / 60)
185. if i == 0:
186. r = rgb_max
187. g = rgb_min + rgb_adj
188. b = rgb_min
189. elif i == 1:
190. r = rgb_max - rgb_adj
191. g = rgb_max
192. b = rgb_min
193. elif i == 2:
194. r = rgb_min
195. g = rgb_max
196. b = rgb_min + rgb_adj
197. elif i == 3:
198. r = rgb_min
199. g = rgb_max - rgb_adj
200. b = rgb_max
201. elif i == 4:
202. r = rgb_min + rgb_adj
203. g = rgb_min
204. b = rgb_max
205. else:
206. r = rgb_max
```

```
207.g = rgb_min

208.b = rgb_max - rgb_adj

209.return [r, g, b]

210.

211.def police(self):

212.self.lightMode = 'police'

213.self.resume()

214.

215.def breath(self, R_input, G_input, B_input):

216.self.lightMode = 'breath'

217.self.colorBreathR = R_input

218.self.colorBreathG = G_input

219.self.colorBreathB = B_input

220.self.resume()

221.

222.def resume(self):

223.self.__flag.set()


224.def breathProcessing(self):

225.while self.lightMode == 'breath':

226.for i in range(0,self.breathSteps):

227.if self.lightMode != 'breath':

228.break

229.self.set_all_led_color(self.colorBreathR*i/self.breathSteps,
    self.colorBreathG*i/self.breathSteps, self.colorBreathB*i/self.breathSteps)

230.#self.show()

231.time.sleep(0.03)

232.for i in range(0,self.breathSteps):

233.if self.lightMode != 'breath':

234.break

235.self.set_all_led_color(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
```

```
       self.colorBreathG-(self.colorBreathG*i/self.breathSteps), self.colorBreathB-

       (self.colorBreathB*i/self.breathSteps))
236. #self.show()
237. time.sleep(0.03)
238. def policeProcessing(self):
239. while self.lightMode == 'police':
240. for i in range(0,3):
241. self.set_all_led_color_data(0,0,255)
242. self.show()
243. time.sleep(0.05)
244. self.set_all_led_color_data(0,0,0)
245. self.show()
246. time.sleep(0.05)
247. if self.lightMode != 'police':
248. break
249. time.sleep(0.1)
250. for i in range(0,3):
251. self.set_all_led_color_data(255,0,0)
252. self.show()
253. time.sleep(0.05)
254. self.set_all_led_color_data(0,0,0)
255. self.show()
256. time.sleep(0.05)
257. time.sleep(0.1)
258.
259.
260. def lightChange(self):
261. if self.lightMode == 'none':
262. self.pause()
263. elif self.lightMode == 'police':
264. self.policeProcessing()
```

```
265. elif self.lightMode == 'breath':
266. self.breathProcessing()
267.
268. def run(self):
269. while 1:
270. self.__flag.wait()
271. self.lightChange()
272. pass
273.
274.
275.
276. if __name__ == '__main__':
277. import time
278. import os
279. print("spidev version is ", spidev.__version__)
280. print("spidev device as show:")
281. os.system("ls /dev/spi*")
282.
283. led = Adeept_SPI_LedPixel(8, 255)              # Use MOSI for /dev/spidev0 to drive the
     lights
284. #led = Adeept_SPI_LedPixel(8, 255, 'GRB', 0)   # Use MOSI for /dev/spidev0 to drive the
     lights
285. #led = Adeept_SPI_LedPixel(8, 255, 'GRB', 1)   # Use MOSI for /dev/spidev1 to drive the
     lights
286. #led = Adeept_SPI_LedPixel(8, 255, 'GRB', 3)   # Use MOSI for /dev/spidev3 to drive the
     lights
287. #led = Adeept_SPI_LedPixel(8, 255, 'GRB', 5)   # Use MOSI for /dev/spidev5 to drive the
     lights
288. #led = Adeept_SPI_LedPixel(8, 255, 'GRB', 6)   # Use MOSI for /dev/spidev6 to drive the
     lights
289. try:
```

```
290.if led.check_spi_state() != 0:
291.led.set_led_count(8)
292.led.set_all_led_color_data(255, 0, 0)
293.led.show()
294.time.sleep(0.5)
295.led.set_all_led_rgb_data([0, 255, 0])
296.led.show()
297.time.sleep(0.5)
298.led.set_all_led_color(0, 0, 255)
299.time.sleep(0.5)
300.led.set_all_led_rgb([0, 255, 255])
301.time.sleep(0.5)

302.led.set_led_count(12)
303.led.set_all_led_color_data(255, 255, 0)
304.for i in range(255):
305.led.set_led_brightness(i)
306.led.show()
307.time.sleep(0.005)
308.for i in range(255):
309.led.set_led_brightness(255-i)
310.led.show()
311.time.sleep(0.005)
312.
313.led.set_led_brightness(20)
314.while True:
315.for j in range(255):
316.for i in range(led.led_count):
317.led.set_led_rgb_data(i, led.wheel((round(i * 255 / led.led_count) + j)%256))
318.led.show()
319.time.sleep(0.002)
```

```
320.else:

321.led.led_close()

322.except KeyboardInterrupt:

323.led.led_close()

324.
```