5.8 Light Follow

In Section 4.1, we use the photosensitive sensor to detect the light intensity and display the result on the OLED. In this section, we use this light intensity result to achieve light tracking.

Note that the search light is affected by the ambient light, please use it in a place where the indoor light does not change much or is dark.

The motor speed is affected by the battery power. The routine is when the battery power is high (the power value is above 26000), if the battery power is low, it needs to be charged in time or the motor speed should be modified.

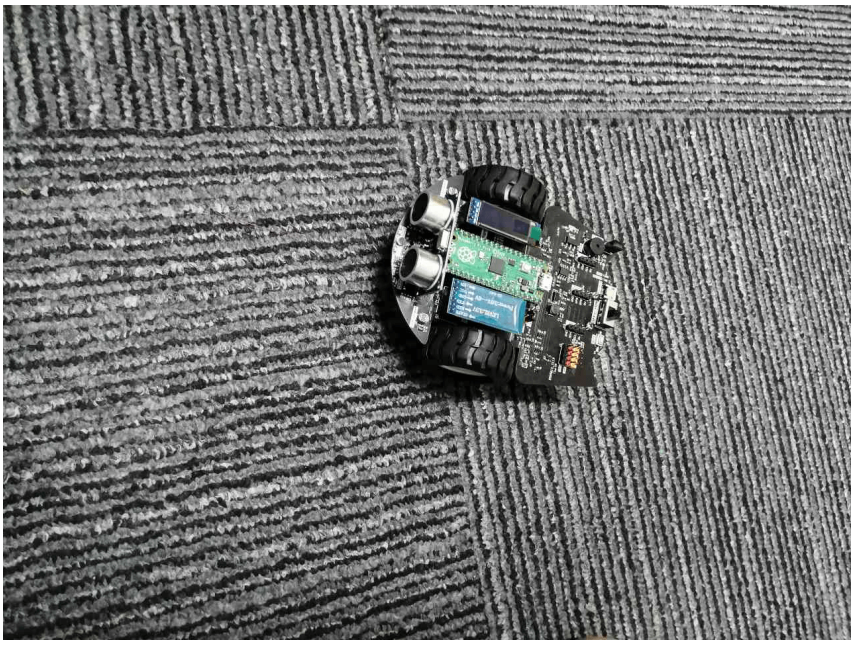
1. Learning Objectives

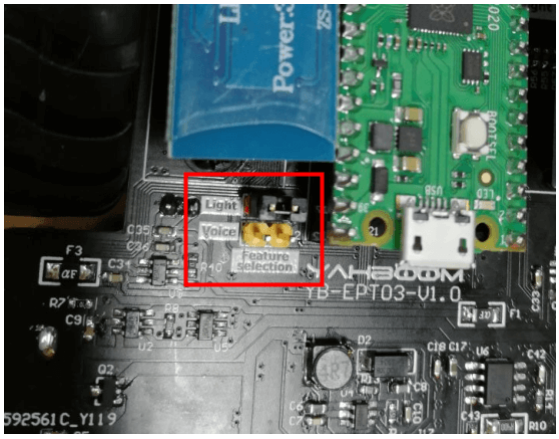
1. Learn the motor, OLED, photosensitive sensor, and programmable RGB lights of the Raspberry Pi Pico motherboard and the car expansion board to realize the car light search.

2. Understand the realization of the car's light seeking.

2. the use of hardware

This course uses the motor, OLED, photosensitive sensor, and programmable RGB lights of the PICO motherboard and the car expansion board. The car is placed in an open space without obstacles in front of it. **Please connect the jumper cap to the Light pin header before running**, have a light source ready (flashlight or your phone's camera light).





In the program, we read the values of the two photosensitive sensors, compare the two values, and judge the position of the light source, thereby controlling the car to search for light.

3. program analysis

Complete program location: Pico Robot Supporting Materials -> Appendix -> Course code -> 3.Robot course -> 8. Lighting follow.py

```
import time
from machine import Pin, I2C, PWM, ADC
from pico_car import SSD1306_I2C, pico_car, ws2812b

Motor = pico_car()
Motor.Car_Stop()
num_leds = 8  # Number of NeoPixels
# Pin where NeoPixels are connected
pixels = ws2812b(num_leds, 0)
pixels.fill(0,0,0)
pixels.show()
#Light1 -> GP27
#Light2 -> GP26
light1 = machine.ADC(27)
light2 = machine.ADC(26)
#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C(128, 32, i2c)
light_down = 0.9
light_up = 1.1

while True:
    #get value
    LightS1 = light1.read_u16()
    LightS2 = light2.read_u16()
    print("light1 is %d"%(LightS1) )
    print("light2 is %d"%(LightS2) )
    #Display sound on OLED
    oled.text('Light1:', 0, 0)
    oled.text(str(LightS1), 60, 0)
    oled.text('Light2:', 0, 10)
    oled.text(str(LightS2), 60, 10)
    #Control action
    if LightS1 > (LightS2*light_down) and LightS1 < (LightS2*light_up):
        Motor.Car_Run(120,120)
        for i in range(num_leds):
            pixels.set_pixel(i,150,150,150)
        pixels.show()
    elif LightS2 > (LightS1*light_down) and LightS2 < (LightS1*light_up):
        Motor.Car_Run(120,120)
        for i in range(num_leds):
            pixels.set_pixel(i,150,150,150)
        pixels.show()
    elif LightS1 > (LightS2*light_up) or LightS2 < (LightS1*light_down):
        Motor.Car_Run(120,0)
        pixels.fill(0,0,0)
        pixels.set_pixel(0,150,0,150)
        pixels.set_pixel(1,150,0,150)
        pixels.set_pixel(2,150,0,150)
        pixels.set_pixel(7,150,0,150)
```

```
            pixels.show()
        elif LightS2 > (LightS1*light_up) or LightS1 < (LightS2*light_down):
            Motor.Car_Run(0,120)
            pixels.fill(0,0,0)
            pixels.set_pixel(3,150,0,150)
            pixels.set_pixel(4,150,0,150)
            pixels.set_pixel(5,150,0,150)
            pixels.set_pixel(6,150,0,150)
            pixels.show()
        else:
            Motor.Car_Stop()
        oled.show()
        oled.fill(0)
        time.sleep(0.01)
```

from pico_car import SSD1306_I2C, pico_car, ws2812b

Using pico_car's SSD1306_I2C, pico_car, ws2812b, encapsulates the motor driver, RGB lights, and OLED libraries.

import time

The "time" library. This library handles everything time related, from measuring it to inserting delays into programs. The unit is seconds.

from machine import Pin, I2C, ADC, PWM

The machine library contains all the instructions that MicroPython needs to communicate with Pico and other MicroPython-compatible devices, extending the language of physical computing, using the Pin, PWM, ADC and I2C libraries here.

Motor= pico_car()

Initialize the motor drive.

pixels = ws2812b(num_leds, 0)

Initialize RGB lights, we have 8 RGB lights, here num_leds is set to 8.

pixels.fill(0,0,0)

Set all lights to 0,0,0, that is, turn off all lights, the parameters are (red, green, blue), and the color brightness is 0-255.

pixels.show()

Display the set lights.

pixels.set_pixel(0,150,0,150)

Set the first headlight to purple.

i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)

Set the IIC 1 pin to SCL 15, SDA 14, and the frequency to 100000.

oled = SSD1306_I2C (128, 32, i2c)

Initialize the size of the OLED to 128*32, and pass in the IIC parameters set earlier.

Motor.Car_Run(120,120)

To control the car to move forward, the speed is set to 120, the parameters are (left motor speed, right motor speed), and the speed range is 0-255.

Motor.Car_Stop()

Control the car to stop.

light1 = machine.ADC(27)

Initialize ADC port 27, a total of two photosensitive sensors, and set pins 27 and 26 respectively.

oled.text (str (LightS1), 60, 0)

Convert the photosensitive value into a string and display it at the 60,0 position of the OLED.

oled.show ()

Display the set OLED content.

oled.fill (0)

Clear the settings and prepare for the next display.

LightS1 = light1.read_u16()

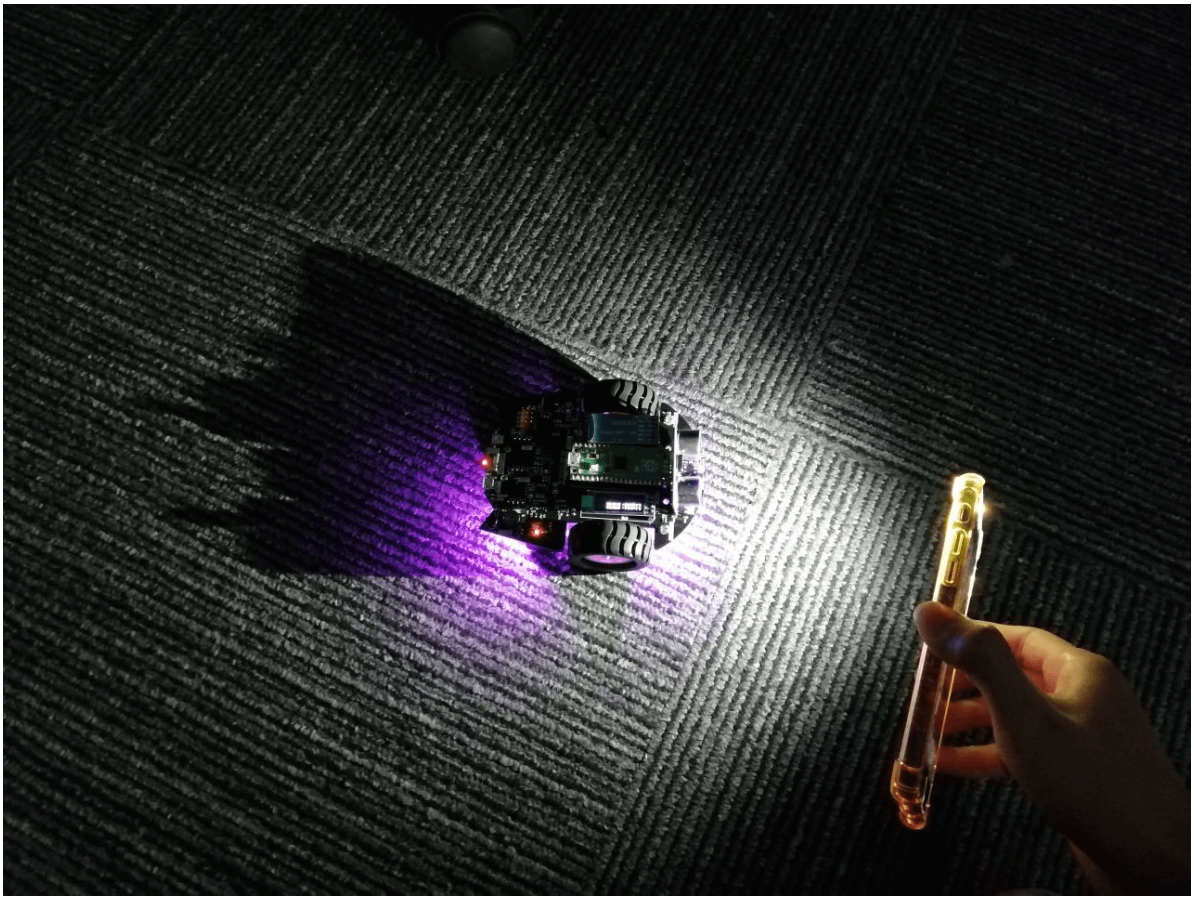
The light1.read_u16() function is used to detect the value of the sound sensor and assign it to the variable LightS1.

4. Experimental phenomenon

After the code is downloaded, we can see that the first line of the OLED displays the value of photosensitive sensor 1, and the second line displays the value of photosensitive sensor 2.

At the same time, the printf shell will also print the value of the photosensitive sensor.

When we shine strong light on the photosensitive sensor, we can control the movement of the car.



Due to the photosensitive sensor is easily affected by ambient light, please use it in a dark place.

If the light-seeking effect is not very well, you can adjust the detection range parameter in code appropriately