4.3 Tracking sensor

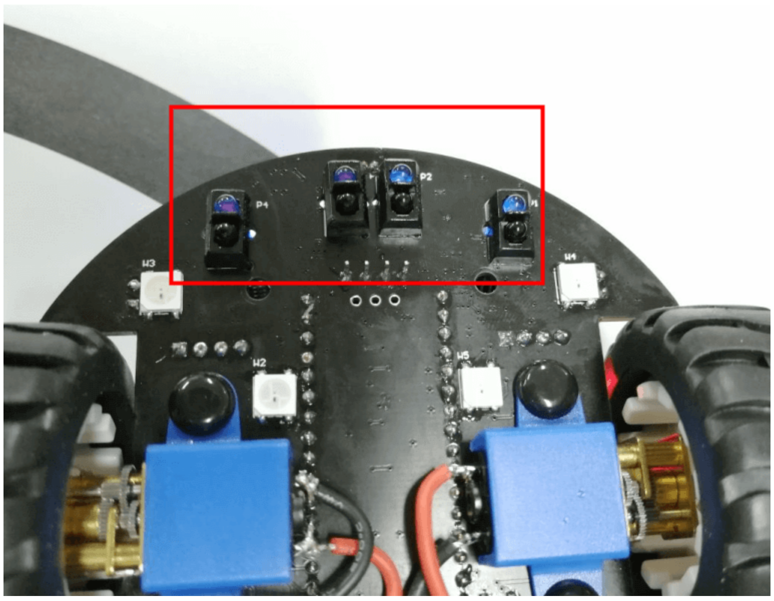
Note: The tracking sensor will be affected by light, please run the program in an indoor environment without sunlight to reduce the interference of sunlight on the tracking sensor. And keep the indoor light enough when tacking.

1. Learning Objectives

In this course, we will learn how to use tracking sensor on Pico robot.

2. About Hardware

We need use tracking sensor and OLED on Pico robot.



The tracking sensor is a sensor that uses infrared rays for data processing. It has the advantages of high sensitivity. There is an infrared transmitting tube and an infrared receiving tube on the sensor. When the ground is black, it absorbs all light, and the resistance of the receiving tube increases. The surface is white, reflecting all the light, the resistance of the receiving tube is reduced, and then the detected state is changed to a value of 0/1 through the voltage comparison circuit on the board.

3. About code

Code path: Code -> 2.Advanced course -> 3.Tracking sensor.py

```
from machine import Pin, I2C
from pico_car import SSD1306_I2C
import time
#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C(128, 32, i2c)
#Define the tracking sensor, 1-4 from left to right
#recognize that black is 0 and white is 1
#Tracing_1 Tracing_2 Tracing_3 Tracing_4
#    2         3        4          5
Tracing_1 = machine.Pin(2, machine.Pin.IN)
Tracing_2 = machine.Pin(3, machine.Pin.IN)
Tracing_3 = machine.Pin(4, machine.Pin.IN)
Tracing_4 = machine.Pin(5, machine.Pin.IN)
```

```python
while True:
    oled.text('T1', 5, 0)
    oled.text('T2', 35, 0)
    oled.text('T3', 65, 0)
    oled.text('T4', 95, 0)
    print("T1: %d T2: %d T3: %d T4: %d "%
(Tracing_1.value(),Tracing_2.value(),Tracing_3.value(),Tracing_4.value()))
    # Tracing1 display
    if Tracing_1.value() == 1:
        oled.text('1', 9, 10)
        for i in range(10):
            for j in range(10):
                oled.pixel(i+8, 20+j, 1)
    elif Tracing_1.value() == 0:
        oled.text('0', 9, 10)
        for i in range(10):
            oled.pixel(i+8, 20, 1)
            oled.pixel(i+8, 29, 1)
        for j in range(8):
            oled.pixel(8, 21+j, 1)
        for j in range(8):
            oled.pixel(17, 21+j, 1)
    # Tracing2 display
    if Tracing_2.value() == 1:
        oled.text('1', 39, 10)
        for i in range(10):
            for j in range(10):
                oled.pixel(i+38, 20+j, 1)
    elif Tracing_2.value() == 0:
        oled.text('0', 39, 10)
        for i in range(10):
            oled.pixel(i+38, 20, 1)
            oled.pixel(i+38, 29, 1)
        for j in range(8):
            oled.pixel(38, 21+j, 1)
        for j in range(8):
            oled.pixel(47, 21+j, 1)
    # Tracing3 display
    if Tracing_3.value() == 1:
        oled.text('1', 69, 10)
        for i in range(10):
            for j in range(10):
                oled.pixel(i+68, 20+j, 1)
    elif Tracing_3.value() == 0:
        oled.text('0', 69, 10)
        for i in range(10):
            oled.pixel(i+68, 20, 1)
            oled.pixel(i+68, 29, 1)
        for j in range(8):
            oled.pixel(68, 21+j, 1)
        for j in range(8):
            oled.pixel(77, 21+j, 1)
    # Tracing4 display
    if Tracing_4.value() == 1:
        oled.text('1', 99, 10)
        for i in range(10):
            for j in range(10):
                oled.pixel(i+98, 20+j, 1)
    elif Tracing_4.value() == 0:
        oled.text('0', 99, 10)
        for i in range(10):
            oled.pixel(i+98, 20, 1)
            oled.pixel(i+98, 29, 1)
        for j in range(8):
            oled.pixel(98, 21+j, 1)
        for j in range(8):
            oled.pixel(107, 21+j, 1)
    oled.show()
    oled.fill(0)
    time.sleep(0.1)
```

from pico_car import SSD1306_I2C

Use SSD1306_I2C of pico_car.

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

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 and I2C libraries here.

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.

Tracing_1 = machine.Pin(2, machine.Pin.IN)

Initialize pin 2 as the pin of line tracking sensor 1 and set it as input.

oled.text ('T1', 5, 0)

Set the OLED to display the character 'T1' at position 5,0.

oled.show ()

Display the set OLED content.

oled.fill (0)

Clear the settings and prepare for the next display.

Tracing_1.value()

The Tracing_1.value() function is used to detect the level of the corresponding port. In the above code, after identifying 0 and 1, draw on the OLED through oled.pixel.

4. Experimental phenomenon

After the code is downloaded, we can see that the first line of the OLED displays 'T1 T2 T3 T4', the second line displays the values of each sensor, and the third line displays the detection of black or white by drawing.

At the same time, the print shell will also print the detection result.