5.6 Ultrasonic avoiding

Note: Motor speed is affected by battery power.
For this course, when the battery power is high (the power value is above 26000), if the battery power is not enough, we need to charge battery in time or modify the motor speed in the code.

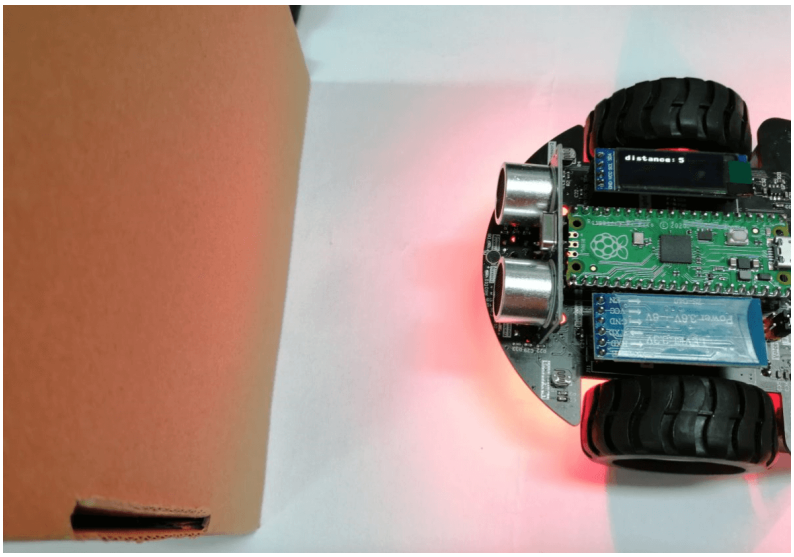
1. Learning Objectives

In this course, we will learn that how to make Pico robot realize ultrasonic avoiding function.

2. About Hardware

We need use motors, unltrasonic sensor, OLED, RGB light, buzzer on Pico robot.

Note: The car can be placed on the ground, and the obstacles on the ground should not be too dense



In the code, we make different actions for different distance values by reading the ultrasonic sensor value.

3. About code

Code path: Code -> 3.Robot course -> 6.Ultrasonic avoiding.py

```
import time
from machine import Pin, I2C, PWM
from pico_car import SSD1306_I2C, ultrasonic, 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()
# set buzzer pin
BZ = PWM(Pin(22))
BZ.freq(1000)
# Initialize music
CM = [0, 330, 350, 393, 441, 495, 556, 624]
```

```
#initialization ultrasonic
ultrasonic = ultrasonic()
#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C(128, 32, i2c)

while True:
    #get distance
    distance = ultrasonic.Distance_accurate()
    print("distance is %d cm"%(distance) )
    #display distance
    oled.text('distance:', 0, 0)
    oled.text(str(distance), 75, 0)
    oled.show()
    oled.fill(0)
    #Control action
    if distance < 10:
        for i in range(num_leds):
            pixels.set_pixel(i,255,0,0)
        pixels.show()
        Motor.Car_Back(150,150)
        BZ.duty_u16(500)
        BZ.freq(CM[7])
        time.sleep(0.2)
        Motor.Car_Right(150,150)
        BZ.duty_u16(500)
        BZ.freq(CM[5])
        time.sleep(0.2)
        BZ.duty_u16(0)
    elif distance >= 10 and distance < 30:
        for i in range(num_leds):
            pixels.set_pixel(i,255,255,0)
        pixels.show()
        Motor.Car_Run(100,100)
    else:
        for i in range(num_leds):
            pixels.set_pixel(i,0,255,0)
        pixels.show()
        Motor.Car_Run(100,100)
    time.sleep(0.1)
```

from pico_car import SSD1306_I2C, ultrasonic, pico_car, ws2812b

Using pico_car's SSD1306_I2C, ultrasonic, pico_car, ws2812b, encapsulates motor driver and RGB lights, OLED, and ultrasonic 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, 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 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(i,255,0,0)

Use a for loop to set all car lights to red.

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.

oled.show ()

Display the set OLED content.

oled.fill (0)

Clear the settings and prepare for the next display.

Motor.Car_Run(100,100)

Control the car to move forward, the speed is set to 150, the parameters are (left motor speed, right motor speed), and the speed range is 0-255.

Motor.Car_Back(150,150)

Control the car to back up.

Motor.Car_Right(150,150)

Control the car to rotate right.

Motor.Car_Stop()

Control the car to stop.

BZ = PWM(Pin(22))

Set IO22 as a PWM output pin to control the buzzer.

BZ.freq(1000)

Set the PWM frequency to 1000.

BZ.duty_u16(0)

When the value is 0, the sound is turned off, and when the value is 500, the sound is turned on.

ultrasonic = ultrasonic()

Initialize ultrasonic ranging.

distance = ultrasonic.Distance_accurate()

Assign the value returned by ultrasonic ranging to the variable distance .

oled.text (str (distance), 75, 0)

Convert the distance to a string to display on the OLED at position 75,0.

4. Experimental Phenomenon

After the code is downloaded, we can see that the OLED displays 'distance: ' and the measured distance. The value will change according to the measurement result. At the same time, the Shell will also display the measured distance. When the distance is less than 10 (encounter obstacles), RGB When the light is red, the car will turn back and turn right, and the buzzer will make a "di-di" sound; when the distance is between 10-30, the RGB light will turn yellow and the car will move forward; when the distance exceeds 30, the RGB light will turn green and the car will move forward.

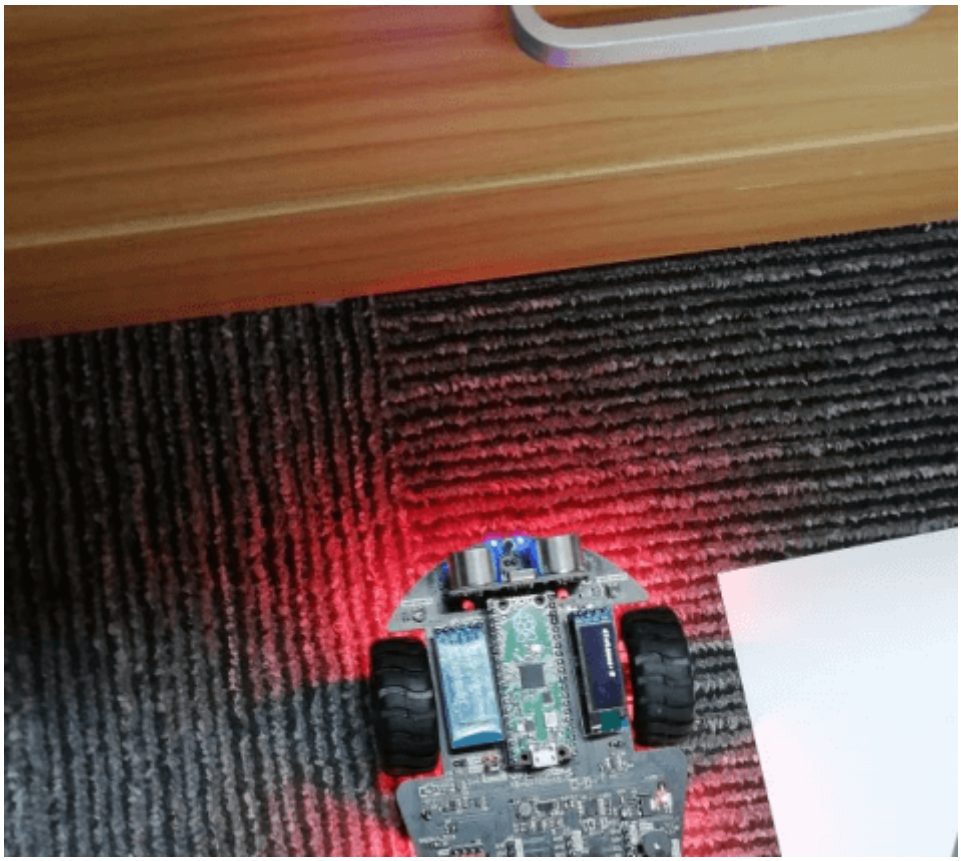
After the code is downloaded, we can see that the OLED displays 'distance: ' and the measured distance.

The value will change according to the measurement result. At the same time, the print shell will also display the measured distance.

When the distance is less than 10cm (encounter obstacles), RGB light will become red, the car will back and spin right, and the buzzer will whistle twice.

When the distance is between 10-30cm, the RGB light will become yellow, the car will move forward.

When the distance exceeds 30cm, the RGB light will become green, the car will move forward.



Note: The shortest ultrasonic measurement distance is 2-3cm, and we uses 10cm to detect obstacles. If the obstacles are too dense, you can modify this value.