5.7 Ultrasonic following

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.

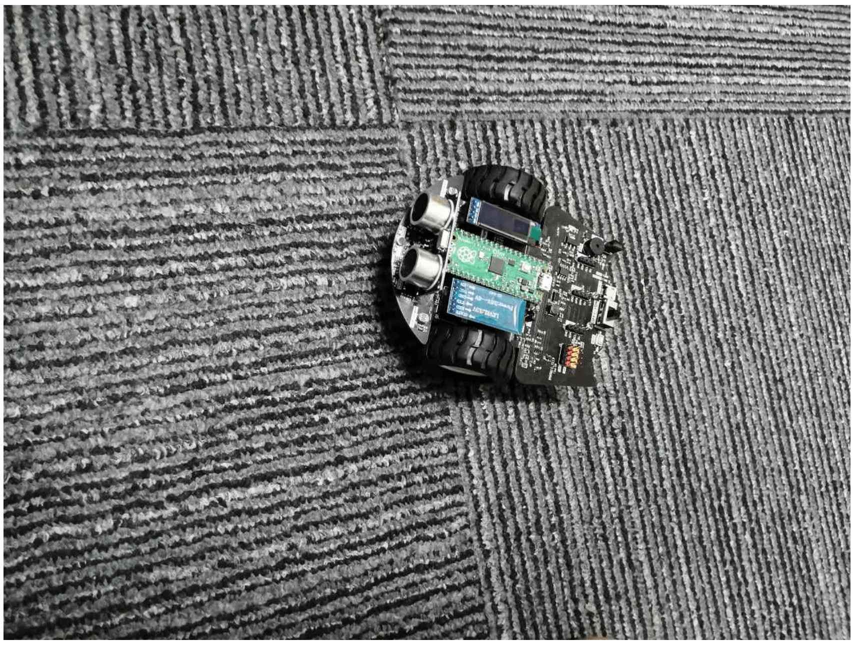
1. Learning Objectives

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

2. About Hardware

We need to use motor, OLED, ultrasonic sensor, programmable RGB light and buzzer on Pico robot.

The car need to be placed on the open ground. If there are obstacles, it may affect the following effect.



3. About Code

Code path: Code -> 3.Robotics course -> 7.Ultrasonic following.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]
song =
[CM[1],CM[1],CM[5],CM[5],CM[6],CM[6],CM[5],CM[4],CM[4],CM[3],CM[3],CM[2],CM[2]
,CM[1],]
beat = [ 0.5,0.5,0.5,0.5,0.5,0.5,1,0.5,0.5,0.5,0.5,0.5,0.5,1,]
#initialization ultrasonic
ultrasonic = ultrasonic()
#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C(128, 32, i2c)
#Define variables
global time_ul, music_i, t_turn, i_turn, t_turn, music_back
music_i = 0
time_ul = 0
i_run = 0
i_turn = 0
t_turn = 0
music_back = 0

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 < 9:
        for i in range(num_leds):
            pixels.set_pixel(i,255,0,0)
        pixels.show()
        Motor.Car_Back(100,100)
        if music_back < 5:
            BZ.duty_u16(500)
            BZ.freq(624)
        else:
            BZ.duty_u16(0)
    elif distance >= 9 and distance < 20:
        if i_run == 0:
            pixels.set_pixel(6,0,0,0)
            pixels.set_pixel(7,0,0,0)
            pixels.set_pixel(2,150,0,150)
            pixels.set_pixel(3,150,0,150)
            i_run = 1
        elif i_run == 1:
            pixels.set_pixel(2,0,0,0)
            pixels.set_pixel(3,0,0,0)
            pixels.set_pixel(1,150,0,150)
            pixels.set_pixel(4,150,0,150)
            i_run = 2
        elif i_run == 2:
            pixels.set_pixel(1,0,0,0)
            pixels.set_pixel(4,0,0,0)
            pixels.set_pixel(0,150,0,150)
            pixels.set_pixel(5,150,0,150)
            i_run = 3
        elif i_run == 3:
            pixels.set_pixel(0,0,0,0)
            pixels.set_pixel(5,0,0,0)
            pixels.set_pixel(6,150,0,150)
            pixels.set_pixel(7,150,0,150)
            i_run = 4
        elif i_run == 4:
            pixels.set_pixel(0,0,0,0)
            pixels.set_pixel(5,0,0,0)
            pixels.set_pixel(6,150,0,150)
            pixels.set_pixel(7,150,0,150)
            i_run = 0
        pixels.show()
        Motor.Car_Run(100,100)
        BZ.duty_u16(500)
        BZ.freq(song[music_i])
        time.sleep(beat[music_i]/2)
        BZ.duty_u16(0)

    else:
        BZ.duty_u16(0)
        if i_turn == 0:
            pixels.set_pixel(7,0,0,0)
            pixels.set_pixel(0,0,150,150)
        else:
            pixels.set_pixel(i_turn-1,0,0,0)
            pixels.set_pixel(i_turn,0,150,150)
        pixels.show()
        if t_turn < 5:
            Motor.Car_Right(120,120)
        elif t_turn >= 5 and t_turn < 10:
            Motor.Car_Left(120,120)

    time_ul = time_ul + 1
    music_i = music_i + 1
    if music_i >= len(song):
        music_i = 0
    i_turn = i_turn + 1
    if i_turn == 8:
        i_turn = 0
    t_turn = t_turn + 1
    if t_turn >= 10:
        t_turn = 0
    music_back = music_back + 1
    if music_back >= 10:
        music_back = 0
    time.sleep(0.01)
```

In this program, RGB lighting effects are made. When moving forward, it will play music while running water lights. When rotating to find and follow the target, there is also a running water light effect. In a program that can only be executed sequentially in a single thread, it is necessary to run multiple effects at the same time. , it is necessary to cut the music, running lights, and car running into small pieces, each of which contains fragments of music, running lights, and the running of the car, so we made multiple variables in the program to achieve this. an effect.

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 100, the parameters are (left motor speed, right motor speed), and the speed range is 0-255.

Motor.Car_Back(100,100)

Control the car to back up.

Motor.Car_Left(120,120)

Control the car to turn left.

Motor.Car_Right(120,120)

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

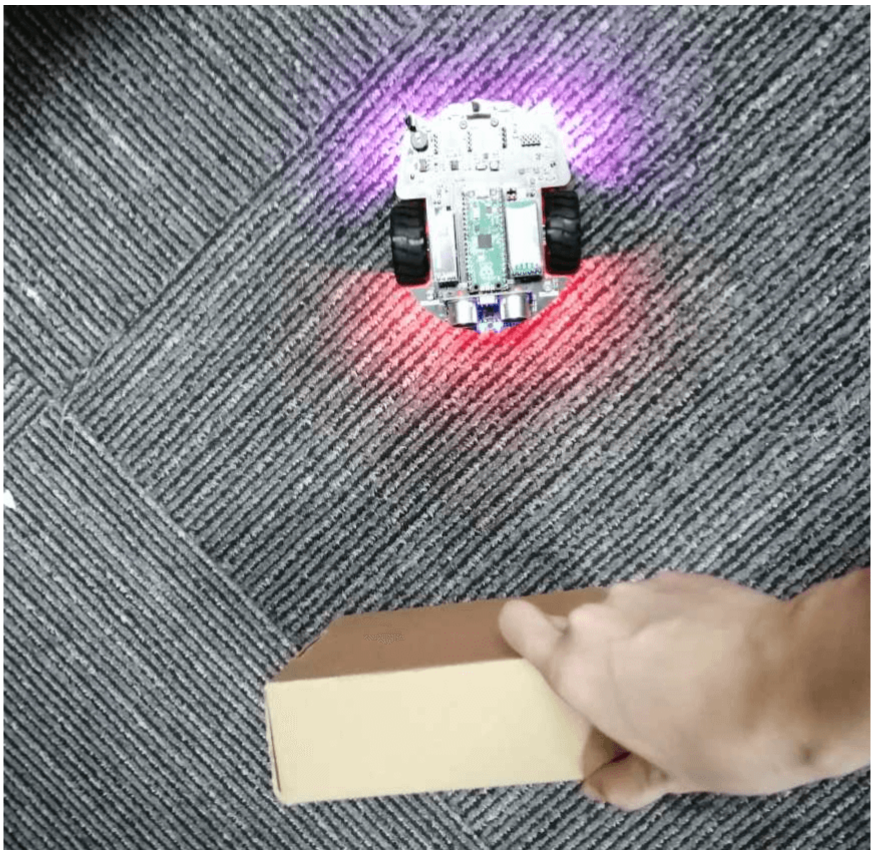
After the code is downloaded, we can see that the OLED displays 'distance: ' and the measured distance. The value will change according to the measurement result.

At the same time, the print shell will also display the measured distance.

When the distance is less than 9cm (too close to the following object), RGB light becomes red, car moves back, and the buzzer makes a whistle twice.

When the distance is between 9-20cm, RGB lights will form purple running lights, car moves forward, and music is played at the same time.

When the distance is more than 20cm, RGB light will form cyan running light, car will turn right and left to find the following object.



Note that the shortest ultrasonic measurement distance is 2-3cm.

If the follow-up effect is not very well, you can modify the distance or modify the running speed of the car.