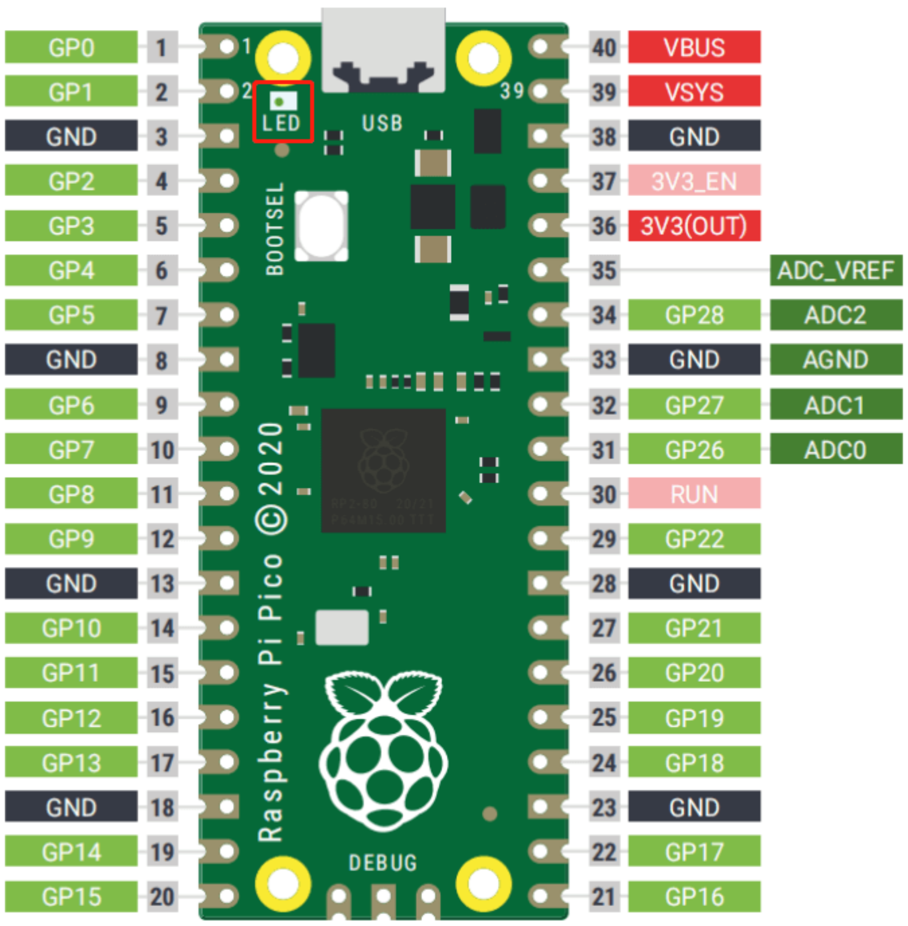
3.1 Control LED light

1. Learning Objectives

In this course, we will learn how to drive LED light on Pico board.

2. About Hardware

This course requires no additional hardware, just use the onboard LED lights on the Raspberry Pi Pico board.



3. About code

Code path: Code -> 1.basic course -> 1.Control LED light.py

```
import machine
import time
led_onboard = machine.Pin(25, machine.Pin.OUT)

while True:
    led_onboard.value(1)
    time.sleep(1)
    led_onboard.value(0)
    time.sleep(1)
```

import machine

The machine library contains all the instructions MicroPython needs to communicate with Pico and other MicroPython-compatible devices, extending the language of physical computing.

import time

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

led_onboard = machine.Pin(25, machine.Pin.OUT)

The first parameter, 25, is the number of pins you are setting; the second parameter, machine.Pin.OUT, tells Pico that the pin should be used as an output rather than an input.

time.sleep(1)

This calls the sleep function from the time library, which makes the program pause for any number of seconds you type - 1 second in this case.

4. Experimental Phenomenon

After the code is downloaded, we can see that the LED lights on the Raspberry Pi Pico board keep flashing every 1 second.

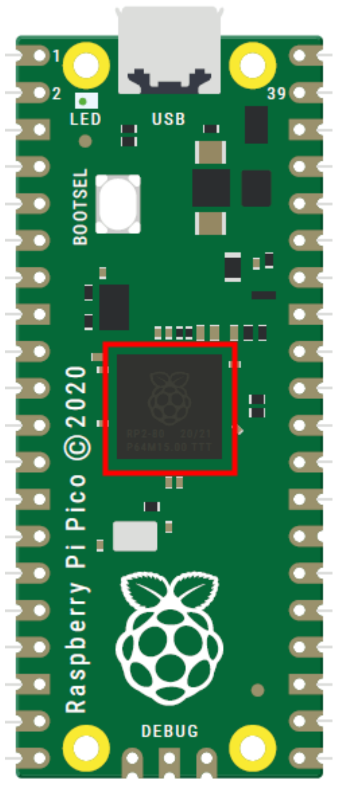
3.2 Onboard temperature sensor

1. Learning Objectives

In this course, we will learn how to read the temperature of the onboard temperature sensor.

2. About Hardware

This course requires no additional hardware and directly utilizes the temperature sensor on the Raspberry Pi Pico board.



The Raspberry Pi Pico's RP2040 microcontroller is a digital device, like all mainstream microcontrollers: it's made up of thousands of transistors, tiny switching devices that either turn on or off. So your PICO can't really understand an analog signal - it can be anything in the spectrum between fully off and fully on - without relying on extra hardware: an analog-to-digital converter (ADC).

An ADC has two key characteristics: its resolution, measured in digital bits, and its channels, or how many analog signals it can accept and convert at a time. The ADC in your PICO has a resolution of 12 bits, which means it can convert an analog signal to a digital signal with numbers ranging from 0 to 4095 - although this is handled in MicroPython to convert from 0 to 65 , a 16-bit number of 535, so it behaves like ADCs on other MicroPython microcontrollers. It has three channels brought to the GPIO pins: GP26, GP27, and GP28, which are also called GP26_ADC0, GP27_ADC1, and GP28_ADC2 for analog channels 0, 1, and 2. There is also a fourth ADC channel, which is connected to a temperature sensor built into the RP2040.

3. About Code

Code path: Code -> 1.Basic course -> 2. On board temperature sensor.py

```
import machine
import time
sensor_temp = machine.ADC(4)
conversion_factor = 3.3 / (65535)
while True:
    reading = sensor_temp.read_u16() * conversion_factor
    temperature = 27 - (reading - 0.706)/0.001721
    print(temperature)
    time.sleep(2)
```

import machine

The machine library contains all the instructions MicroPython needs to communicate with Pico and other MicroPython-compatible devices, extending the language of physical computing.

import time

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

sensor_temp = machine.ADC(4)

Using ADC channel 4, it is connected to a temperature sensor built into the RP2040.

conversion_factor = 3.3 / (65535)

The level of this pin is 3.3V, and the upper limit of the conversion value is 65535, so the ratio of voltage and number is calculated according to this formula.

reading = sensor_temp.read_u16() * conversion_factor

Calculate the voltage value read by the ADC.

temperature = 27 - (reading - 0.706)/0.001721

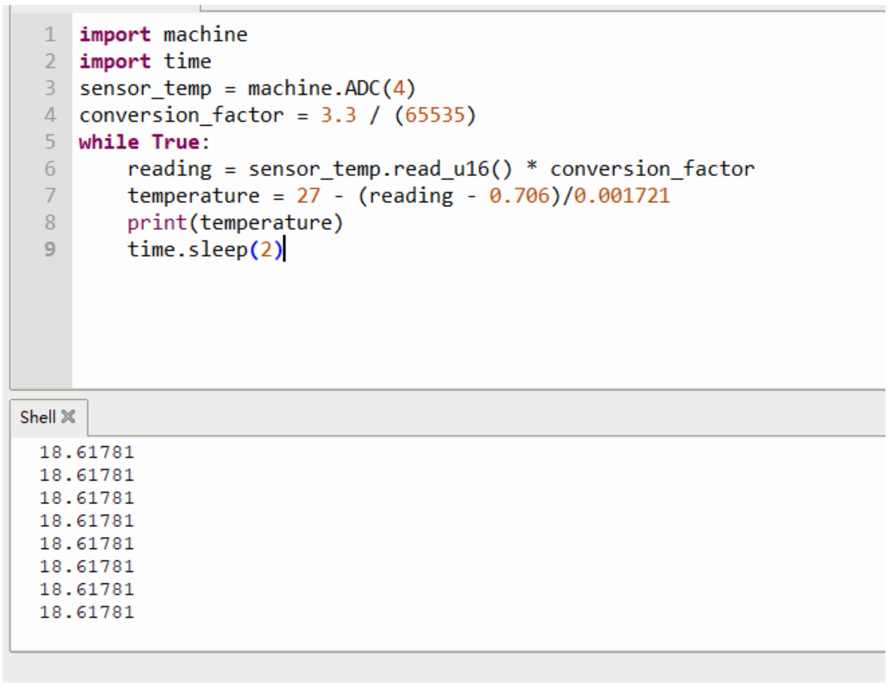
Calculates the temperature value of the built-in temperature sensor based on the voltage value.

time.sleep(2)

This calls the sleep function from the utime library, which makes the program pause for any number of seconds you type - in this case 2 seconds.

4. Experimental Phenomenon

After the code is downloaded, we can observe the temperature value through Thony's Shell window.


```python
import machine
import time
sensor_temp = machine.ADC(4)
conversion_factor = 3.3 / (65535)
while True:
    reading = sensor_temp.read_u16() * conversion_factor
    temperature = 27 - (reading - 0.706)/0.001721
    print(temperature)
    time.sleep(2)
```

```
18.61781
18.61781
18.61781
18.61781
18.61781
18.61781
18.61781
18.61781
```

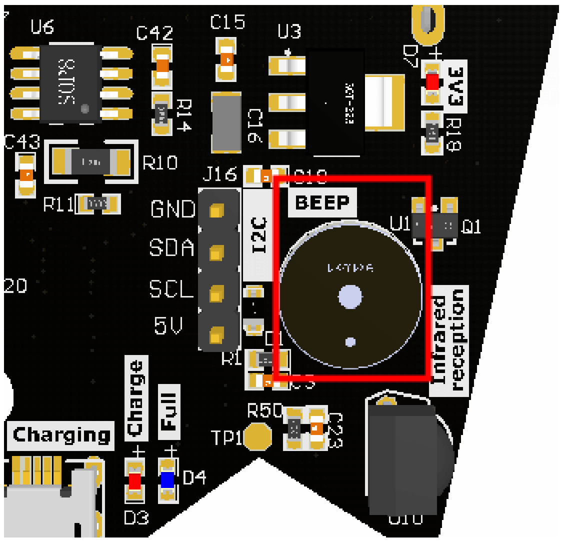
3.3 Passive buzzer

1. Learning Objectives

In this course, we will learn how to use active buzzers.

2. About Hardware

We need use buzzer on Pico robot car.



Passive buzzers use the phenomenon of electromagnetic induction to attract or repel the electromagnet and permanent magnet formed after the voice coil is connected to the alternating current to push the diaphragm to sound. Sound, in the control, we generally use PWM to control the passive buzzer to emit sound.

3. About code

Code path: Code -> 1.Basic course -> 3.Passive buzzer.py

```
from machine import Pin, PWM
import time
# 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,]
# music
def music():
        print('Playing song...')
        for i in range(len(song)):
            BZ.duty_u16(500)
            BZ.freq(song[i])
            time.sleep(beat[i])
            BZ.duty_u16(0)
            time.sleep(0.01)
#play music
music()
print("Ending")
```

from machine import Pin, 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, here only the Pin and PWM libraries are used.

import time

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

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.

music()

By calling the music() function, a for loop is used in the function to play the pre-written sounds of different frequencies one by one, so as to realize the playback of music.

4. Experimental Phenomenon

After the code is downloaded, we can hear the buzzer playing the music star, and the printf shell will printf "Ending" after the music is over.

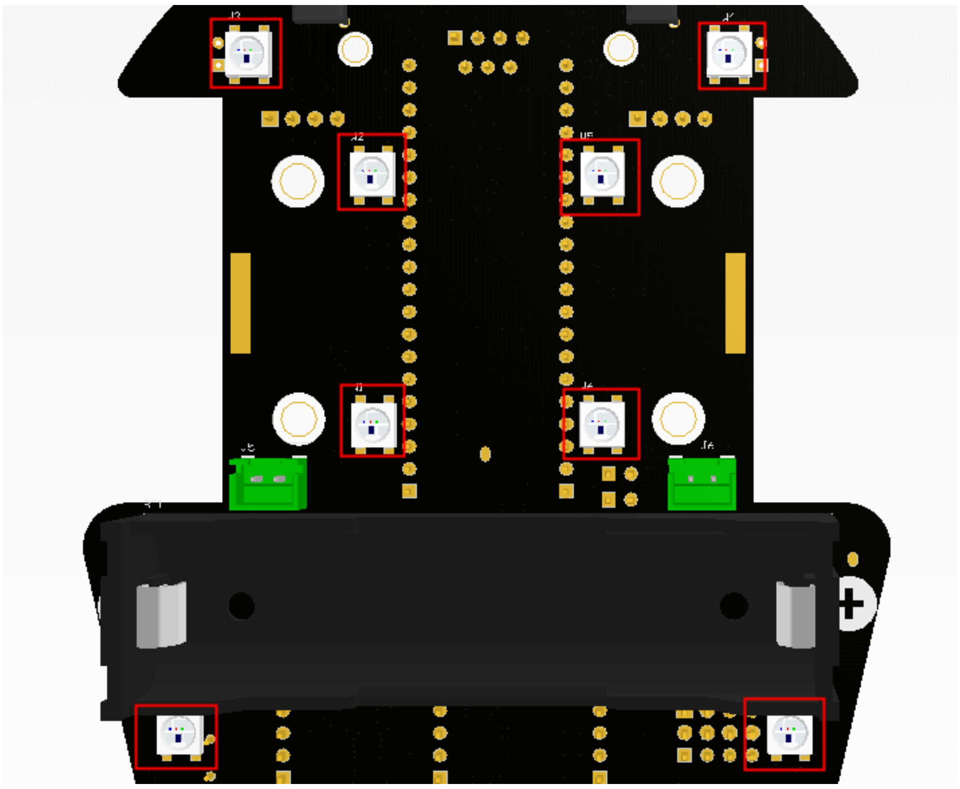
3.4 Breathing light

1. Learning Objectives

In this course, we will learn how to make programmable RGB lights to achieve the effect of breathing lights.

2. About Hardware

We need use RGB lights on Pico robot.



Pico robot has 8 programmable RGB lights, which can realize colorful lighting effects.

The 8 programmable lights have built-in ws2812 chips. Only one port is needed to control 8 lights at the same time through timing control.

The timing control function is encapsulated in the library, we just need to call it to set the color of the light.

3. About Code

Code path: Code -> 1.Basic course -> 4.Breathing light.py

```
import time
from pico_car import ws2812b

num_leds = 8 # Number of NeoPixels
# Pin where NeoPixels are connected
pixels = ws2812b(num_leds, 0)
# Set all led off
pixels.fill(0,0,0)
pixels.show()
# Define variables
i = 0
```

```
brightness = 0
fadeAmount = 1
#breathing
while True:
    for i in range(num_leds):
        pixels.set_pixel(i,0,brightness,brightness)
    pixels.show()
    brightness = brightness + fadeAmount
    if brightness <= 0 or brightness >= 200:
        fadeAmount = -fadeAmount
    time.sleep(0.005)
```

from pico_car import ws2812b

Because only the lights are on, only the ws2812b of pico_car is used here.

import time

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

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,0,brightness,brightness)

Use this function to set the color of each light. The parameters are (the number of the light, red, green, blue), the number of the light starts from 0, and the color brightness is 0-255, for example, the first bright red pixels. set_pixel(0,255,0,0).

brightness = 0 fadeAmount = 1

The breathing light effect is achieved by controlling the addition and subtraction of these two values.

4. Experimental Phenomenon

After the code is downloaded, we can see that the cyan RGB lights are constantly turning on and off with breathing effect.

3.6 OLED display

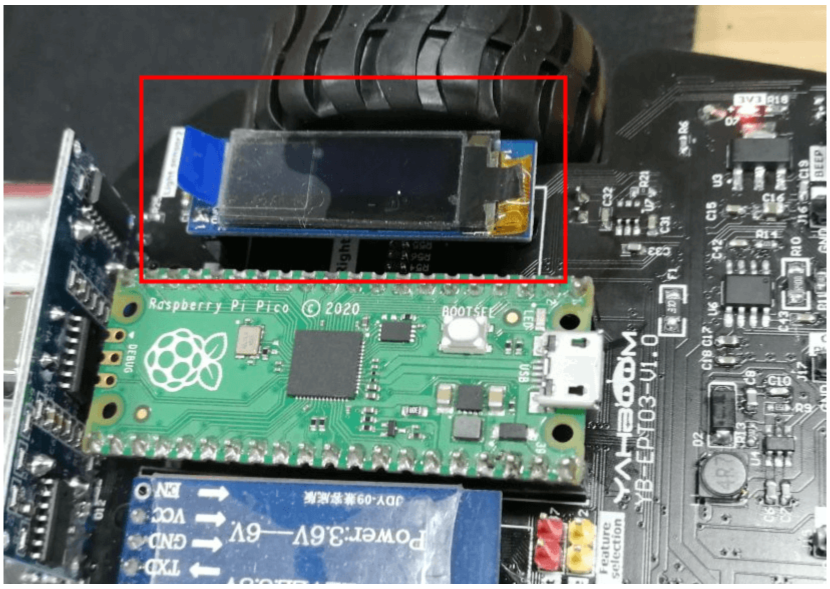
1. Learning Objectives

In this course, we will learn how to drive OLED display.

2. About Hardware

We need to use OLED display on Pico robot.

Note: OLED display must be plugged in before running the routine in this section, otherwise the program will report an error.



OLED display is a display made of organic self-light emitting diodes. Due to the self-luminous organic electroluminescent diodes, no backlight, high contrast ratio, thin thickness, wide viewing angle, fast response speed, wide operating temperature range, simple structure and process are excellent characteristics. The 0.91-inch OLED screen we use uses IIC communication, which saves IO pins and simplifies the control method.

3. About code

Code path: Code -> 1.Basic course -> 6.OLED display.py

```
from machine import Pin, I2C
from pico_car import SSD1306_I2C
import time
# set IIC pin
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
# initialization oled
oled = SSD1306_I2C (128, 32, i2c)
# oled show hello at 0,0
oled.text ('Hello', 0, 0)
oled.show()
oled.fill (0)
time.sleep(1)
# oled showing the World at 0.10
oled.text ('World', 0, 10)
oled.show()
oled.fill (0)
```

```
    time.sleep(1)
    # oled show spot at 100.30
    oled.pixel (100, 30, 1)
    oled.show()
    oled.fill (0)
    time.sleep(1)
```

from pico_car import SSD1306_I2C

Use SSD1306_I2C of pico_car, which is our packaged OLED library.

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. Only the Pin and I2C libraries are used 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.

oled.text ('Hello', 0, 0)

Set the OLED to display 'Hello' at position 0,0.

oled.show ()

Display the set OLED content.

oled.fill (0)

Clear the settings and prepare for the next display.

oled.pixel (100, 30, 1)

Set the OLED to light up the pixel at coordinates 100, 30, if oled.pixel(100, 30, 0), turn off the pixel at coordinates 100, 30.

4. Experimental Phenomenon

After the code is downloaded, we can see that the OLED displays 'Hello' first, then displays 'World' in the lower line, and then displays a point at coordinates 100,30.

3.7 Battery power display

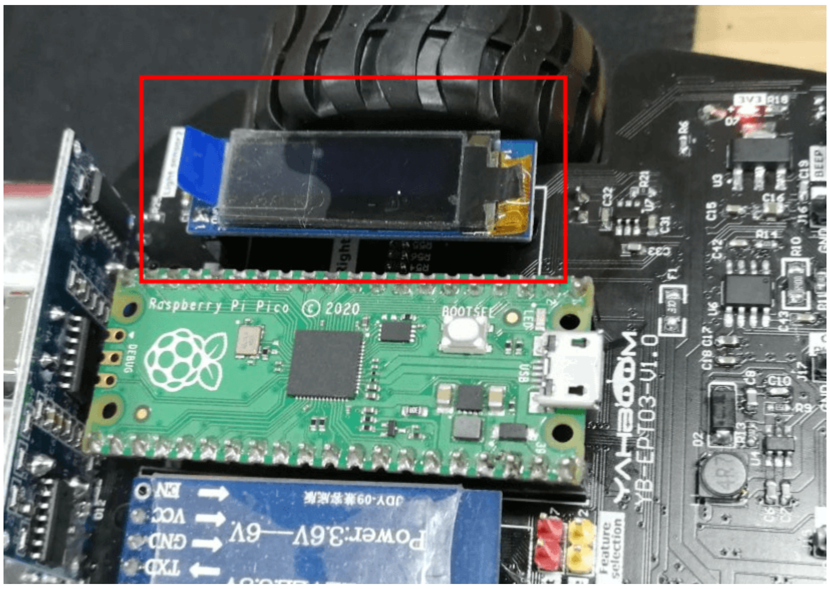
1. Learning Objectives

In this course, we will learn how to detect the current battery level of the car and display it through OLED.

2. About Hardware

We need to use detection circuit and OLED on Pico robot.

Note: OLED display must be plugged in before running the routine in this section, otherwise the program will report an error.



The basic principle of the power detection hardware is to adjust the battery voltage to an appropriate value through a resistor and input it to the main control.

Raspberry Pi Pico detects the voltage through the ADC to determine the power.

3. About code

Code path: Code -> 1.basic course -> 7.Battery power display.py

```
from machine import Pin, I2C, ADC
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)
#initialization ADC
Quantity_of_electricity = machine.ADC(28)

while True:
    #Display power on OLED
    #Under 20000, there is no power at all
    oled.text ('Battery:', 0, 0)
    oled.text(str(Quantity_of_electricity.read_u16()), 65, 0)
    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, ADC

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, I2C and ADC 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.

Quantity_of_electricity = machine.ADC(28)

Initialize ADC port 28.

oled.text(str(Quantity_of_electricity.read_u16()), 65, 0)

Set the OLED to display the value of the battery power at the position of 65,0. We read the value of the ADC through Quantity_of_electricity.read_u16(), convert the value into a string through the str() function, and then display it by the OLED.

oled.show ()

Display the set OLED content.

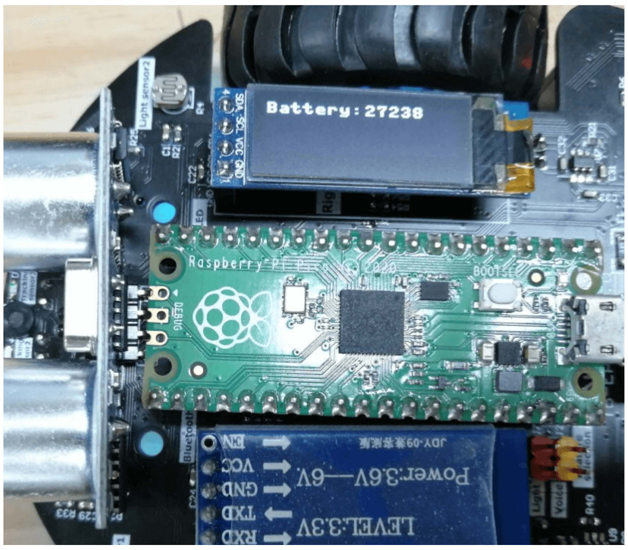
oled.fill (0)

Clear the settings and prepare for the next display.

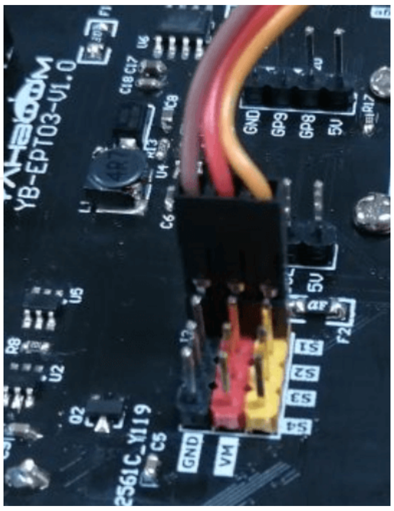
4. Experimental Phenomenon

After the code is downloaded, we can see that the OLED displays 'Battery: battery level', which will change with the voltage.

After testing, when the value is lower than 20000, which means battery at low battery state, and if it is less than 25000, which means battery at medium battery state. .



3.8 Servo control



1. Learning Objectives

In this course, we will learn how to drive PWM servo.

2. About Hardware

We need use servo interface and PWM servo on Pico robot.

The basic control principle of PWM steering gear is to use PWM signal to generate a signal with a period of 20ms and a duty cycle of 0.5~2.5ms to control the angle of the servo.

3. About Code

Code path: Code -> 1.Basic course -> 8.Servo control.py

```
from pico_car import pico_car
Servo = pico_car()

#180 servo S1 angle 0
#the parameters are (steering gear number, steering gear angle)
Servo.servo180 (1.0)
#270 servo
Servo.servo270(2,90)
#360 servo
Servo.servo360 (3,360)
```

from pico_car import pico_car

Use pico_car of pico_car, which encapsulates the servo driver library.

Servo = pico_car()

Initialize the servo.

Servo.servo180 (1.0)

Set the 180-degree PWM servo S1 angle to 0 degrees.

Servo.servo270(2,90)

Set the angle of the 270-degree PWM servo S2 to 90 degrees.

Servo.servo360 (3,360)

Set the angle of the 360-degree PWM servo S3 to 360 degrees.

4. Experimental Phenomen

After the code is downloaded, we can see that the 180° servo, 270° servo, and 360° servo plugged into S1-S3 are at 0°, 90°, and 360° respectively.

4.1 Light sensor

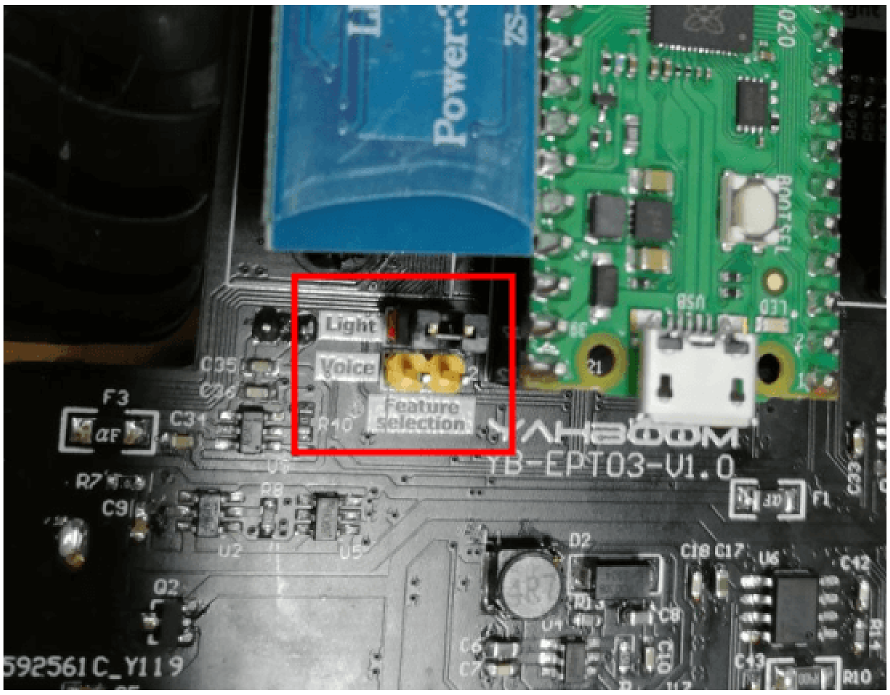
1. Learning Objectives

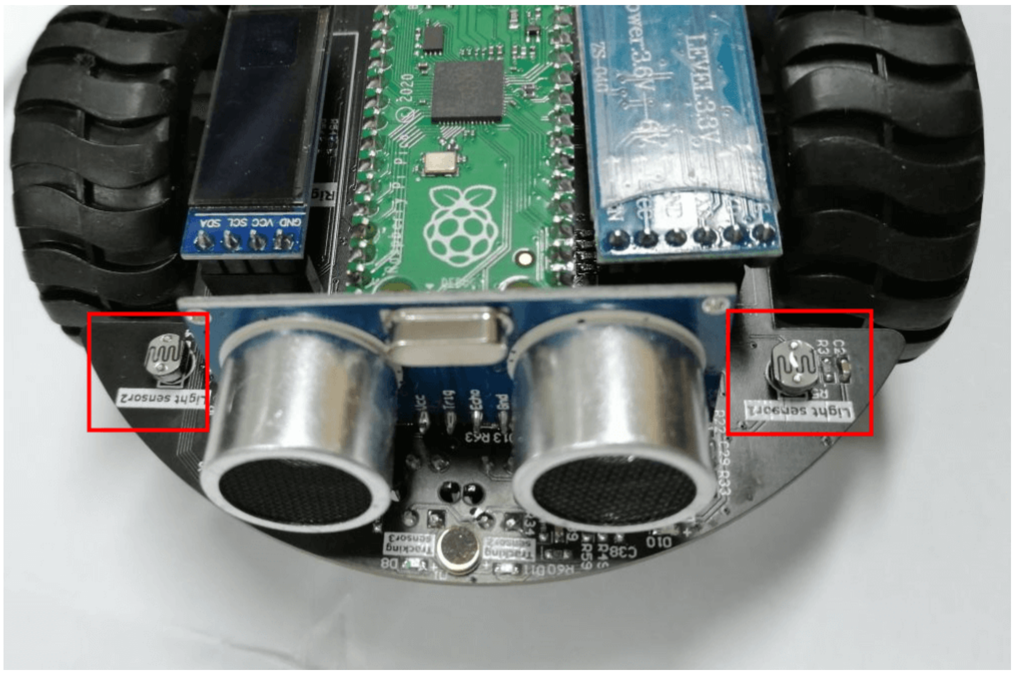
In this course, we will learn how to use the light sensor and OLED on Raspberry Pi Pico robot.

2. About hardware

We need use the photosensitive sensor and OLED on Pico robot expansion board.

Please connect the jumper cap to the Light pin before using. As shown below.





Photoresistors are special resistors made of semiconductor materials such as cadmium sulfide or cadmium selenide, and their working principle is based on the internal photoelectric effect. The stronger the light, the lower the resistance value, and as the light intensity increases, the resistance value decreases rapidly. The sensitivity of the photoresistor to light (that is, the spectral characteristics) is very close to the response of the human eye to visible light (0.4~0.76) μm. As long as the light that the human eye can perceive, it will cause its resistance to change.

3. About code

Code path: Code -> 2.Advanced course -> 1.Photosensitive sensor.py

```
from pico_car import ds, SSD1306_I2C
from machine import Pin, I2C, ADC
import time

#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C (128, 32, i2c)
#Light1 -> GP27
#Light2 -> GP26
light1 = machine.ADC(27)
light2 = machine.ADC(26)

while True:
    #get value
    LightS1 = light1.read_u16()
    LightS2 = light2.read_u16()
    print("light1 is %d"%(LightS1) )
    print("light2 is %d"%(LightS2) )
    #Display sound is OLED
    oled.text ('Light1:', 0, 0)
    oled.text (str (LightS1), 60, 0)
    oled.text ('Light2:', 0, 10)
    oled.text (str (LightS2), 60, 10)
    oled.show()
    oled.fill (0)
    time.sleep(0.5)
```

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, ADC

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, ADC 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.

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 print windows will also print the value of the photosensitive sensor.

If you block the photosensitive sensor with your hand, the value will change.

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

4.2 Sound Sensor

Note: If the recognition effect is not good, we can modify the detection reference value appropriately in the code.
In addition to making sounds, we can also trigger sound sensors by blowing air.

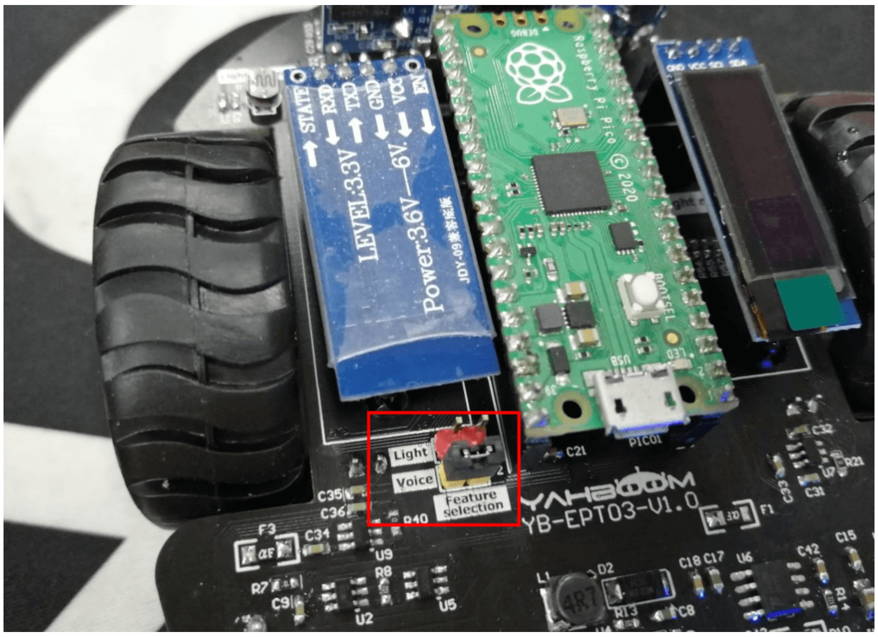
1. Learning Objectives

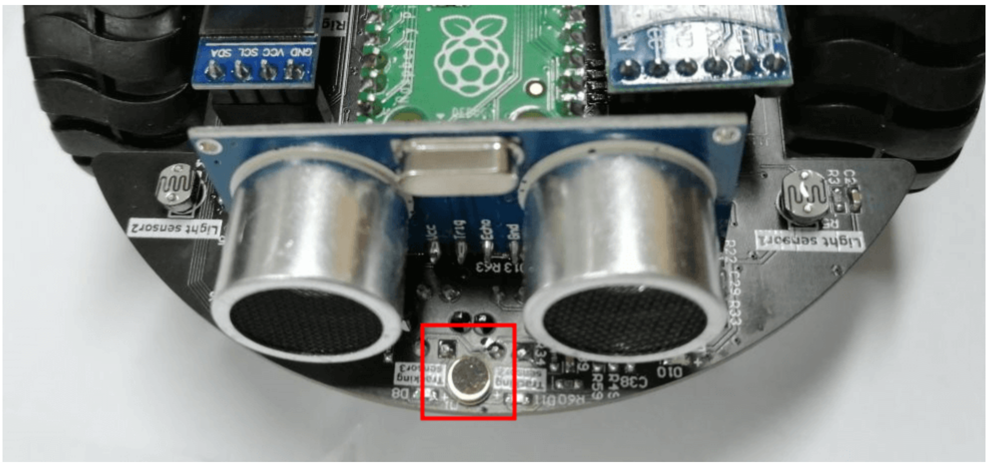
In this course, we will learn how to use the sound sensor and OLED of the Raspberry Pi Pico robot.

2. About Hardware

We need use sound sensor and OLED on Pico robot expansion board.

Please connect the jumper cap to the Voice pin before using. As shown below.





Sound sensor has a built-in condenser electret microphone that is sensitive to sound. The sound wave makes the electret film in the microphone vibrate, resulting in a change in capacitance and a small voltage corresponding to the change. This voltage is converted into a suitable The voltage is sent to the PICO development board. Because there is no ability to detect noise, try to use it in a quieter environment.

3. About code

Code path: Code -> 2.Sensor advanced course -> 2. Sound sensor.py

```python
from pico_car import SSD1306_I2C
from machine import Pin, I2C, ADC
import time
#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C(128, 32, i2c)
#initialization ADC
Sound = machine.ADC(27)

while True:
    #get value
    sounds = Sound.read_u16()
    print(sounds)
    oled.text('Sound:', 0, 0)
    oled.text(str(sounds), 50, 0)
    #Display sound on OLED
    for i in range(10):
        oled.pixel(i, 30, 1)
        oled.pixel(i, 29, 1)
    if sounds > 5000:
        for i in range(10):
            for j in range(4):
                oled.pixel(i+10, 27+j, 1)
    if sounds > 10000:
        for i in range(10):
            for j in range(10):
                oled.pixel(i+20, 21+j, 1)
    if sounds > 20000:
        for i in range(10):
            for j in range(20):
                oled.pixel(i+30, 11+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, ADC

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, ADC 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.

Sound = machine.ADC(27)

Initialize ADC port 27.

oled.text (str (sounds), 50, 0)

Convert the value of the sound into a string and display it at the 50,0 position of the OLED.

oled.pixel (i, 30, 1)

Display a point at the position of i, 30, and display it on the OLED through the for loop drawing.

oled.show ()

Display the set OLED content.

oled.fill (0)

Clear the settings and prepare for the next display.

sounds = Sound.read_u16()

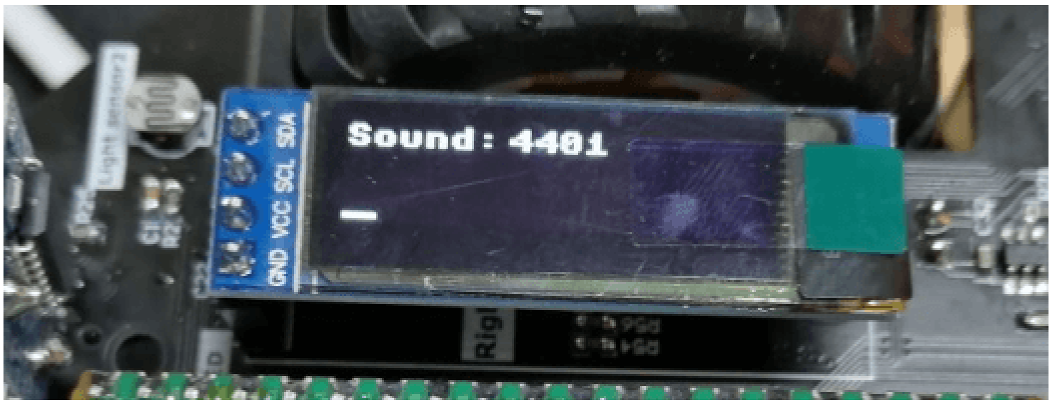
The Sound.read_u16() function is used to detect the value of the sound sensor and assign it to the variable sounds.

4. Experimental phenomenon

After the code is downloaded, we can see that the first line of the OLED displays the value of Sound, and the second line displays the volume of the sound.

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

If you block the photosensitive sensor with your hand, the value will change and the volume will fluctuate.



The performance of the sound sensor will be affected by the battery level.
We recommend adjusting comparison values in code after user testing.
When testing the sensor, it is recommended to turn off high-power loads such as the motor and RGB lights of the car.

4.3 Tracking sensor

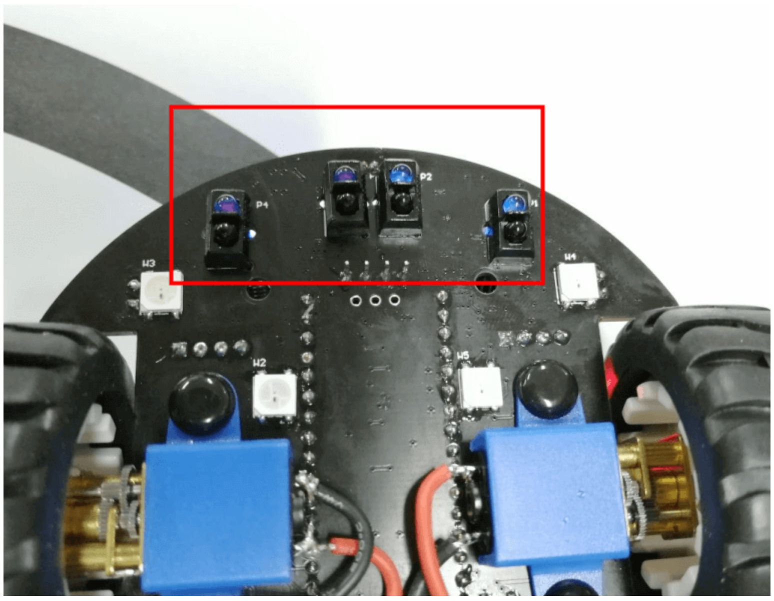
Note: The tracking sensor will be affected by light, please run the program in an indoor environment without sunlight to reduce the interference of sunlight on the tracking sensor. And keep the indoor light enough when tacking.

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)
```

```
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.

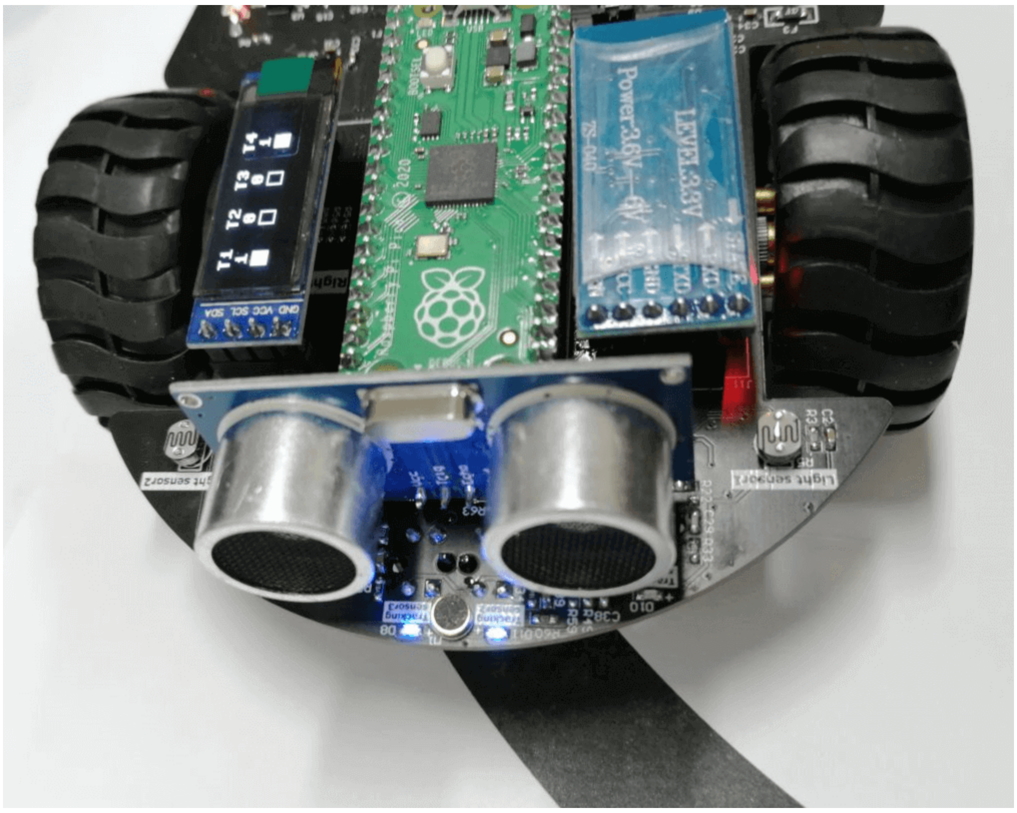
Tracing_1.value()

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.



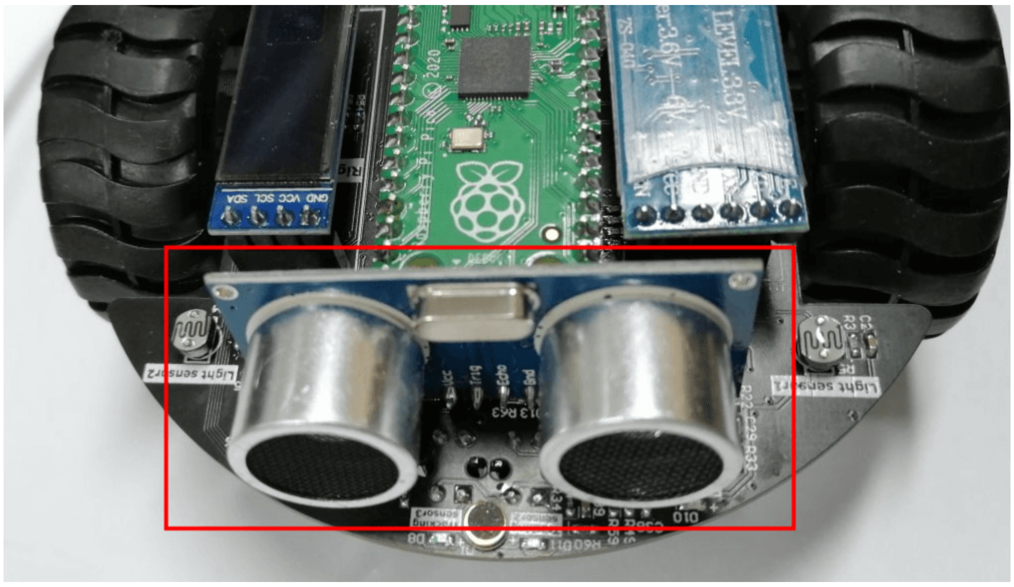
4.4 Ultrasonic sensor

1. Learning Objectives

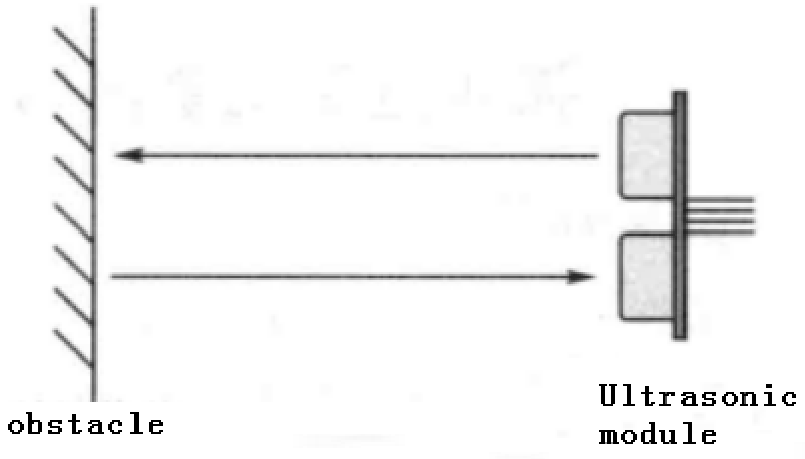
In this course, we will learn to experiment with the combination of ultrasonic sensors and OLEDs on the Raspberry Pi Pico robot.

2. About Hardware

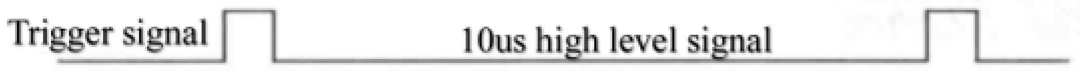
This course uses the ultrasonic sensor and OLED on Pico robot.



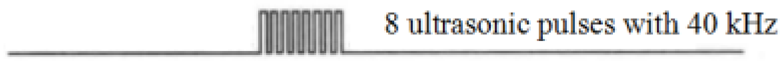
The ultrasonic module is a sensor that uses ultrasonic characteristics to detect the distance. It has two ultrasonic probes for transmitting and receiving ultrasonic waves. The range of measurement is 3-450 cm.



(1) You need to input a high level signal of at least 10us to the Trig pin to trigger the ranging function of the ultrasonic module.



(2) After the ranging function is triggered, the module will automatically send out 8 ultrasonic pulses with 40 kHz and automatically detect whether there is a signal return. This step is done internally by the module.



(3) When the module detects an echo signal, the ECHO pin will output a high level. The high level duration is the time from when the ultrasonic wave is sent to when it returns. You can calculate the distance by using the time function to calculate the high level duration.

Formula: Distance = High level duration * Speed of sound(340M/S)/2.

3. About code

Code path: Code -> 2.Advanced course -> 4. Ultrasonic sensor.py

```
import time
from machine import Pin, I2C
from pico_car import SSD1306_I2C, ultrasonic
#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)
    time.sleep(1)
```

from pico_car import SSD1306_I2C, ultrasonic

Use SSD1306_I2C and ultrasonic from pico_car, which is our packaged OLED and ultrasonic library.

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.

ultrasonic = ultrasonic()

Initialize ultrasonic ranging.

distance = ultrasonic.Distance_accurate()

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

oled.show()

Display the set OLED content.

oled.fill(0)

Clear the settings and prepare for the next display.

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 windows will also print the measured distance.

Note: The shortest ultrasonic measurement distance is 2-3 cm.

4.5 Infrared Remote Control Display

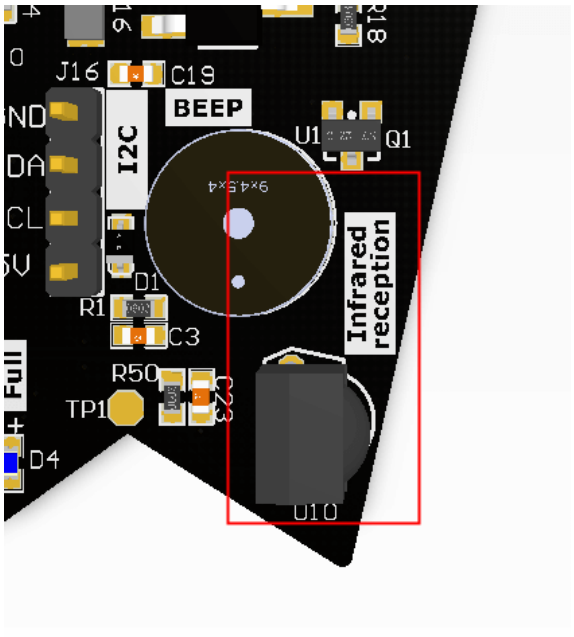
Note: The infrared receiver and remote controller will be affected by light, please run this code in an indoor environment without sunlight to reduce the interference of sunlight on the infrared sensor.

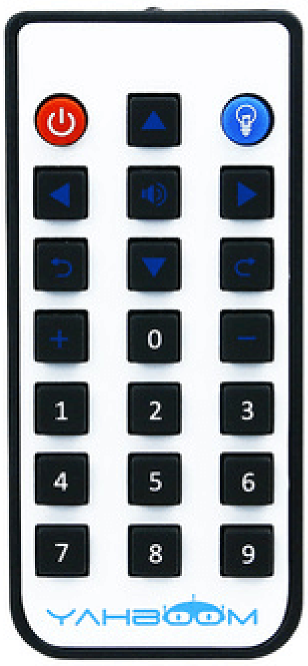
1. Learning Objectives

In this course, we will learn the experiment of infrared remote control and reception.

2. About Hardware

This course uses the infrared receiver and OLED on Pico robot. And we also need to use the infrared remote controller.





The spectrum of infrared light is outside the red light, and the wavelength is 0.76-1.5 μm, which is longer than the wavelength of red light. Infrared remote control is a control method that uses infrared to transmit information. Infrared remote control has the advantages of anti-interference, simple circuit, easy encoding and decoding, low power consumption and low cost. Infrared remote control is suitable for the control of almost all home appliances. The infrared receiving head has built-in photoelectric elements, which can receive infrared light of corresponding wavelength, convert it into digital signal, and judge different remote control buttons by reading the signal value.

The code value of the infrared remote control included in the car kit is shown in the table below.

Keys	Shell prints key values
Power	0
Up	1
Light	2
Left	4
Sound	5
Right	6
Turn Left	8
Down	9
Turn Right	10
+	12
0	13
-	14
1	16
2	17
3	18
4	20
5	21
6	22
7	24
8	25
9	26

3. About code

Code path: Code -> 2.Advanced course -> 5. IR control display.py

```python
import time
from machine import Pin, I2C
from pico_car import SSD1306_I2C, ir
#initialization ir
Ir = ir()
#initialization oled
i2c=I2C(1, scl=Pin(15),sda=Pin(14), freq=100000)
oled = SSD1306_I2C(128, 32, i2c)

while True:
    #get value
    value = Ir.Getir()
    time.sleep(0.01)
    if value != None:
        print(value)
        #display press
        if value == 0:
            while value == 0:
                value = Ir.Getir()
            oled.text('Press:Power', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 1:
            while value == 1:
                value = Ir.Getir()
            oled.text('Press:Up', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 2:
            while value == 2:
                value = Ir.Getir()
            oled.text('Press:Light', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 4:
            while value == 4:
                value = Ir.Getir()
            oled.text('Press:Left', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 5:
            while value == 5:
                value = Ir.Getir()
            oled.text('Press:Sound', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 6:
            while value == 6:
                value = Ir.Getir()
            oled.text('Press:Right', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 8:
            while value == 8:
                value = Ir.Getir()
            oled.text('Press:Turn Left', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 9:
            while value == 9:
                value = Ir.Getir()
            oled.text('Press:Down', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 10:
            while value == 10:
                value = Ir.Getir()
            oled.text('Press:Turn Right', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 12:
            while value == 12:
                value = Ir.Getir()
            oled.text('Press:+', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 13:
            while value == 13:
                value = Ir.Getir()
            oled.text('Press:0', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 14:
            while value == 14:
                value = Ir.Getir()
            oled.text('Press:-', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 16:
            while value == 16:
                value = Ir.Getir()
            oled.text('Press:1', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 17:
            while value == 17:
                value = Ir.Getir()
            oled.text('Press:2', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 18:
            while value == 18:
                value = Ir.Getir()
            oled.text('Press:3', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 20:
            while value == 20:
                value = Ir.Getir()
            oled.text('Press:4', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 21:
            while value == 21:
                value = Ir.Getir()
            oled.text('Press:5', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 22:
            while value == 22:
                value = Ir.Getir()
            oled.text('Press:6', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 24:
            while value == 24:
                value = Ir.Getir()
            oled.text('Press:7', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 25:
            while value == 25:
                value = Ir.Getir()
            oled.text('Press:8', 0, 0)
            oled.show()
            oled.fill(0)
        elif value == 26:
            while value == 26:
                value = Ir.Getir()
            oled.text('Press:9', 0, 0)
            oled.show()
            oled.fill(0)
        value = None
```

We use `while value == key value:` to achieve the effect of pressing and releasing the remote control and then executing.

from pico_car import SSD1306_I2C, ir

Use SSD1306_I2C and ir from pico_car, which is our packaged OLED and IR receiver library.

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.

Ir = ir ()

Initialize the infrared remote control.

value = Ir.Getir ()

Read the infrared remote control value and assign it to the variable value.

oled.show ()

Display the set OLED content.

oled.fill (0)

Clear the settings and prepare for the next display.

oled.text ('Press: Power', 0, 0)

isplay the corresponding key on the OLED, for example, press the power key to display 'Press: Power'.

4. Experimental Phenomenon

After the code is downloaded, when we press the button on IR controller, the corresponding button name will be displayed on the OLED.

At the same time, the print shell will also print corresponding button value.

As shown below.

Keys	Shell prints key values	OLED display
Power	0	Press:Power
Up	1	Press:Up
Light	2	Press:Light
Left	4	Press:Left
Sound	5	Press:Sound
Right	6	Press:Right
Turn Left	8	Press:Turn Left
Down	9	Press:Down
Turn Right	10	Press:Turn Right
+	12	Press:+
0	13	Press:0
-	14	Press:-
1	16	Press:1
2	17	Press:2
3	18	Press:3
4	20	Press:4
5	21	Press:5
6	22	Press:6
7	24	Press:7
8	25	Press:8
9	26	Press:9