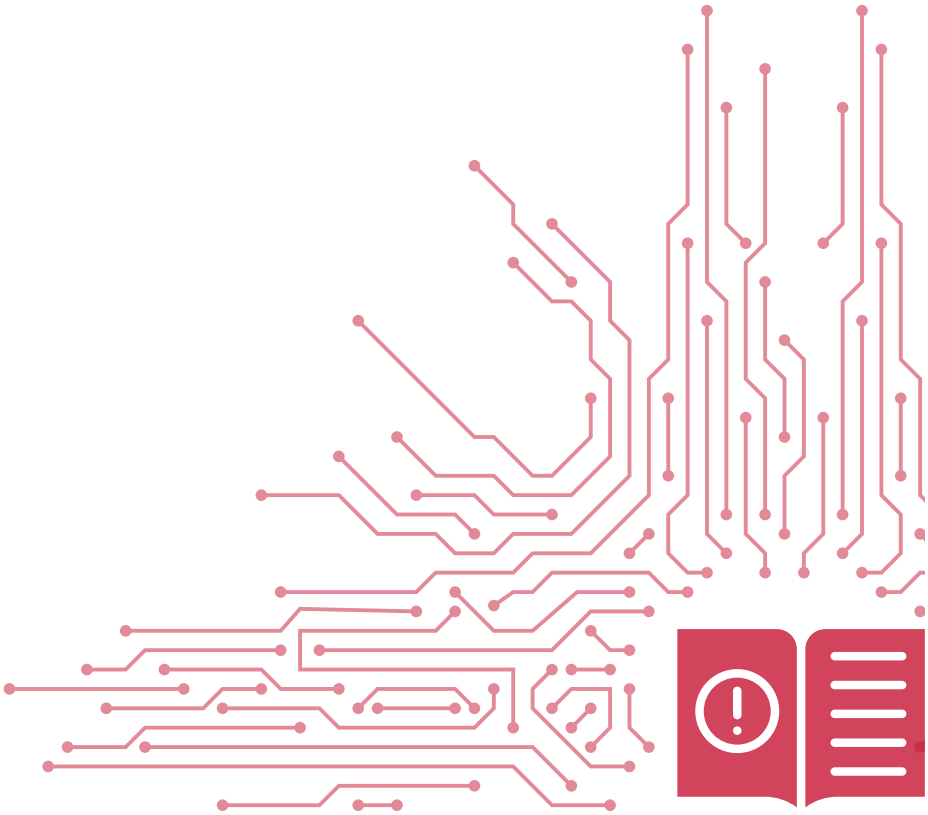
Raspberry Pi Pico

GUIDE



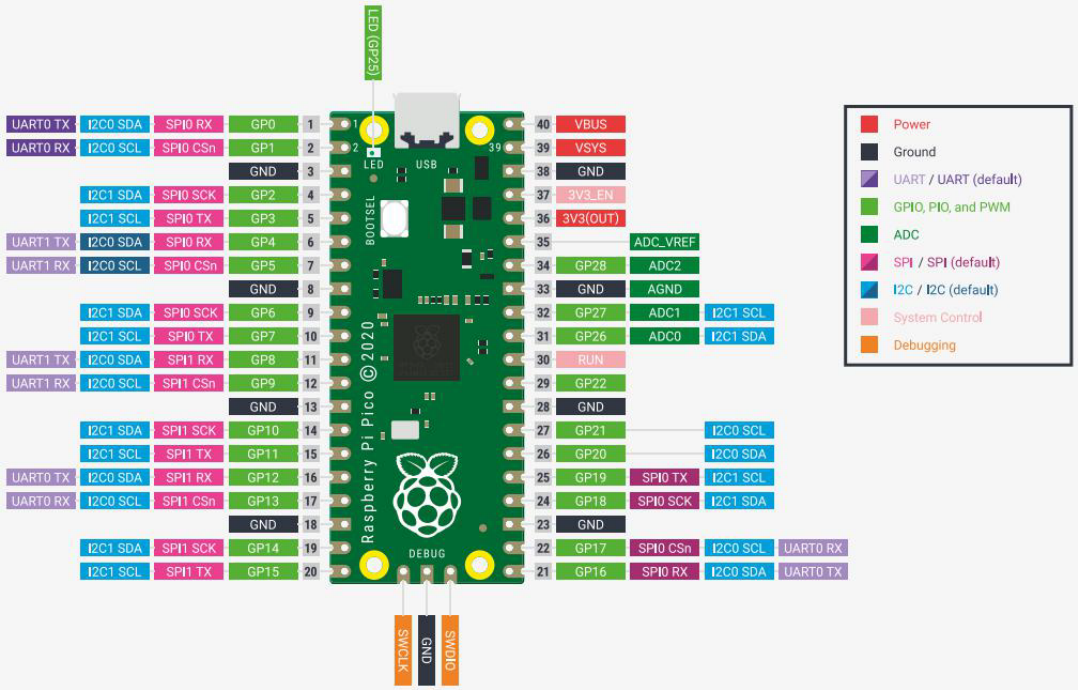
Prepare date: 2021-06-11 Model: Raspberry Pi Pico

CONTENTS

1. Learn about the Raspberry Pi Pico motherboard

On January 21, 2021, the Raspberry Pi Foundation released the Raspberry Pi Pico motherboard, a microprocessor product. The product is based on the Raspberry Pi Foundation's new, self-developed RP2040 processor, which costs just $4. Raspberry Pi Pico is reported as a microprocessor, it specializes in low-latency I/O communication and analog signal input, low power consumption, can make up for the Raspberry Pi motherboard in the microprocessor field of the short board.

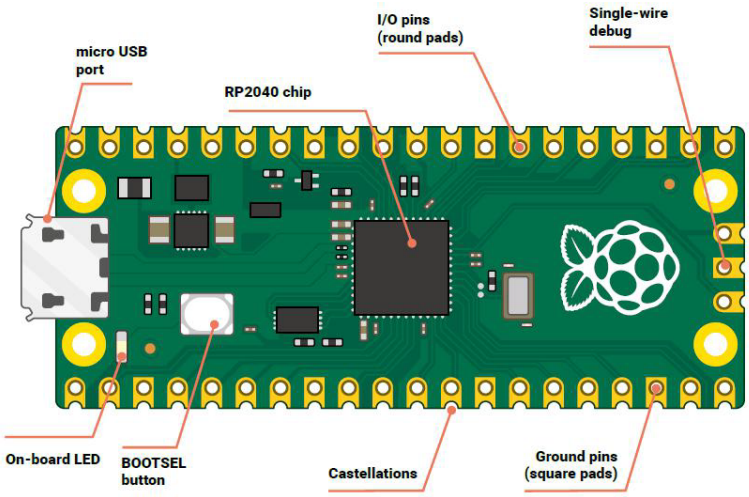
The Raspberry Pi Pico motherboard has become a new favorite of the world's founders, especially based on the new features of the currently hot Micro Python programming, allowing us to learn Python programming at a low cost entirely through the Raspberry Pi Pico motherboard.



1.1 What do you need ?

To build a complete Raspberry Pi Pico learning and development system, you need at least one Raspberry Pico motherboard, Micro USB data cable (optional Pico Block sensor expansion board) and simple line-out IO port and breadboard, in order to better use Raspberry Pi Pico development system, you can also match accessories such as sensor modules, LCD monitors, keyboards and so on, let's go one by one:

1.1.1 Raspberry Pi Pico motherboard



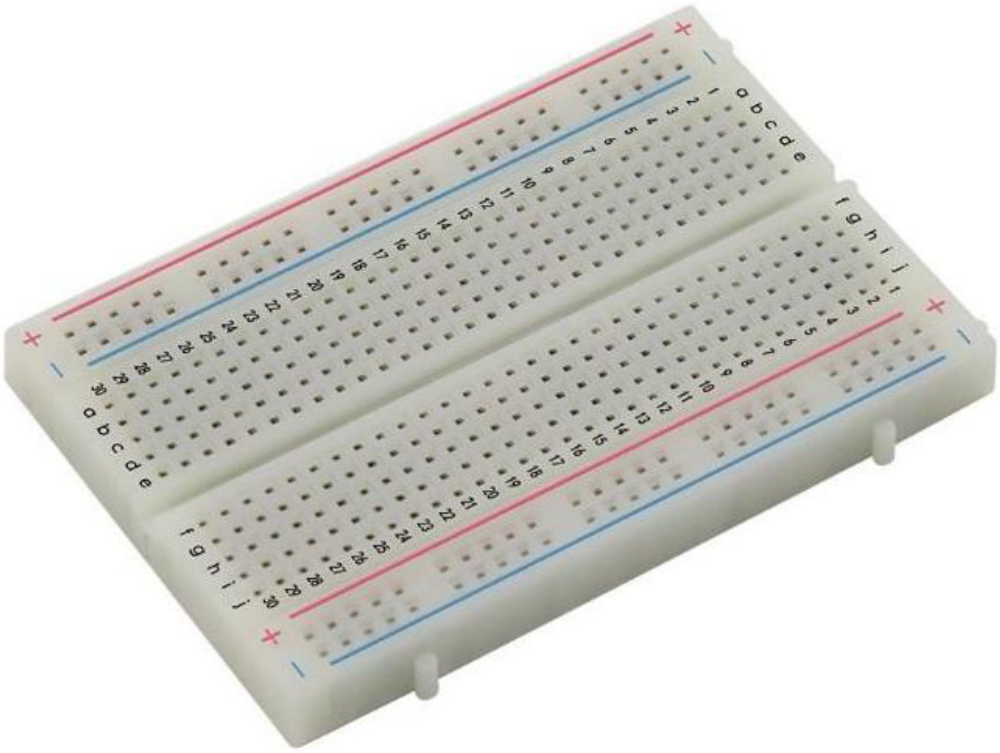
- CPU: 32-bit dual-core ARM Cortex-M0+ at 48MHz, configurable up to 133MHz
- RAM: 264kB of SRAM in six independently configurable banks
- Storage: 2MB external flash RAM
- GPIO: 26 pins
- ADC: 3 × 12-bit ADC pins
- PWM: Eight slices, two outputs per slice for 16 total
- Clock: Accurate on-chip clock and timer with year, month, day, day-of-week, hour, second, and automatic leap-year calculation
- Sensors: On-chip temperature sensor connected to 12-bit ADC channel)
- LEDs: On-board user-addressable LED
- Bus Connectivity: 2 × UART, 2 × SPI, 2 × I2C, Programmable Input/Output (PIO)
- Hardware Debug: Single-Wire Debug (SWD)
- Mount Options: Through-hole and castellated pins (unpopulated) with 4 × mounting holes
- Power: 5 V via micro USB connector, 3.3 V via 3V3 pin, or 2–5V via VSYS pin

1.1.2 Micro USB downloader

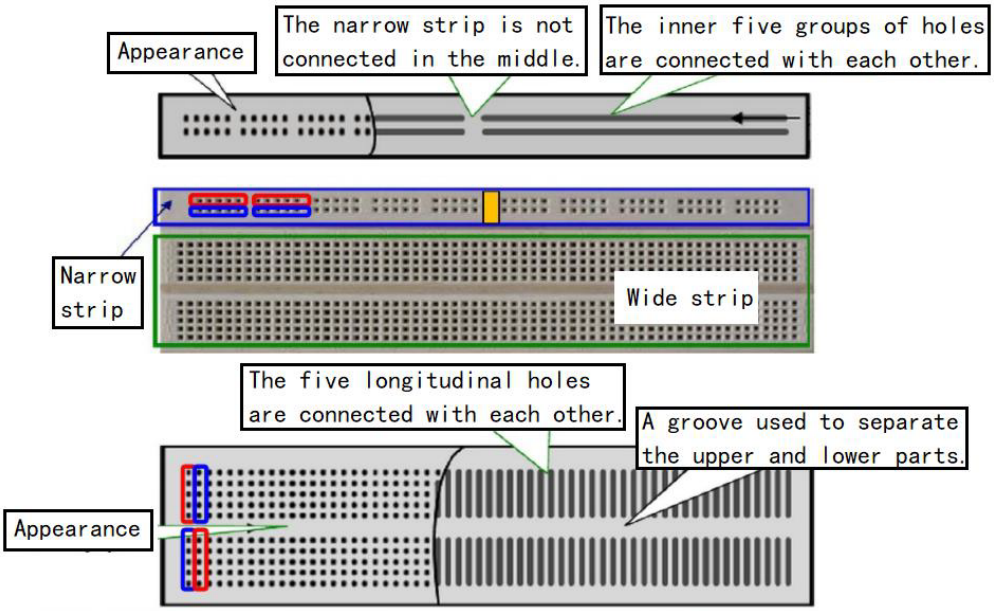
Raspberry Pi Pico motherboard program download and USB power interface for Micro USB interface, so we need to use a better quality Micro USB cable for program download, if the PC side when using, it is best not to use USB Hub extension, because it will lead to unstable download programs, recommended to use the USB with the PC side for program download and power supply, requires USB to provide a stable 5V, In subsequent sensor experiments, sufficient current is required to power the power supply.

1.1.3 Breadboard

The breadboard is made for non-welded experiments in electronic circuits because there are many small jacks on the board. Because a variety of electronic components can be inserted or unplugged as needed, eliminate welding, saving circuit assembly time, and components can be reused, so it is ideal for electronic circuit assembly, commissioning and training.

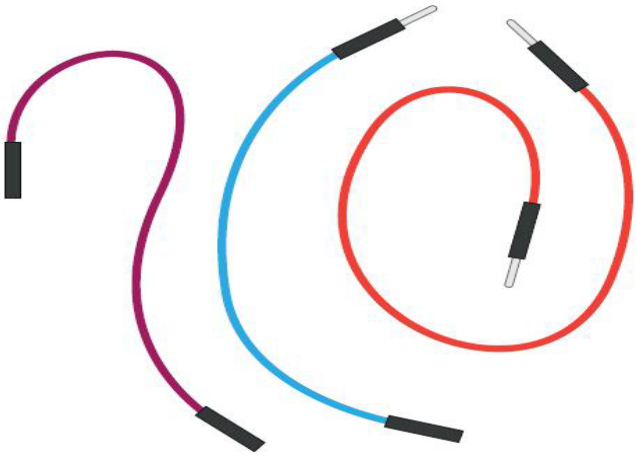


The internal structure of the breadboard



1.1.4 Jumper

There are three types of jumpers: male-to-male, female-to-female, and male-to-male

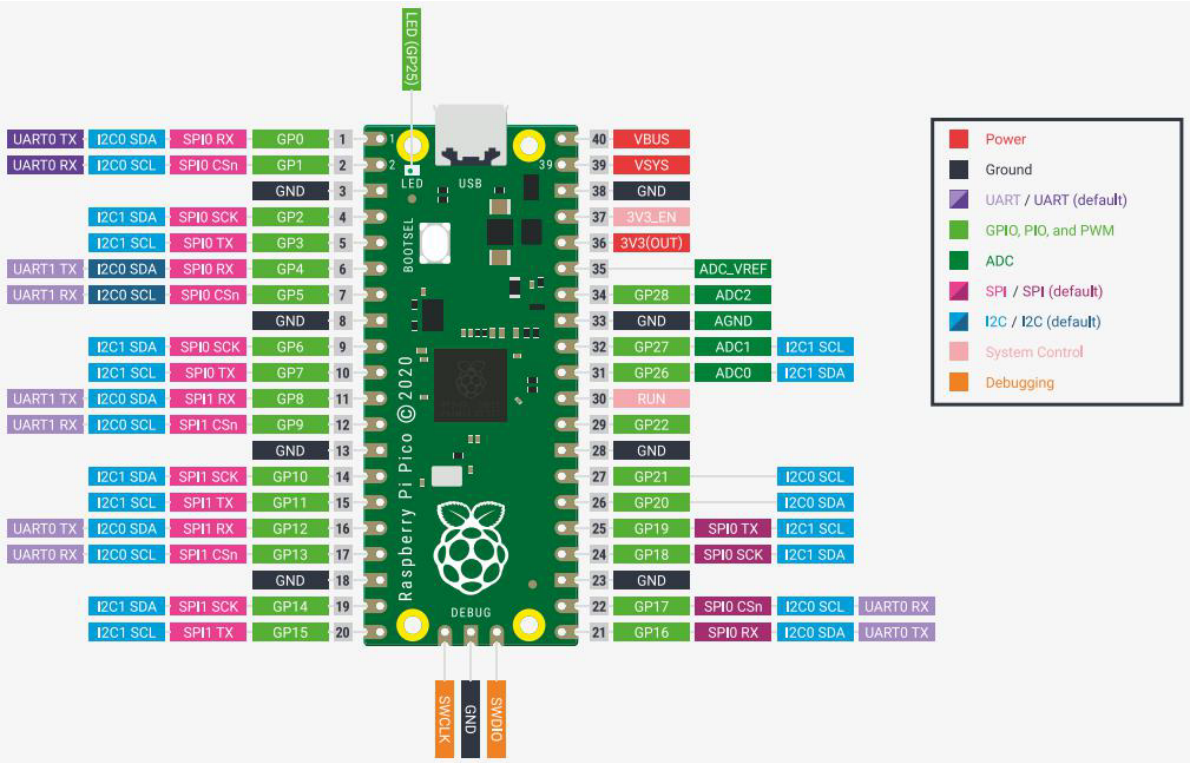


2. Raspberry Pi Pico Learning Readiness

The Raspberry Pi Pico motherboard, which is officially launched by the Raspberry Pico board, is programmed in the Python language and has a Micro python parser installed in the RP2040processor, so we can program using the Python Pico motherboard, and the official programming method is also available in C/C++, an advanced player's choice that requires a lot of software to support it.

2.1 Introduction to the I/O mouth of Raspberry Pi Pico

Raspberry Pi Pico offers 40Pin ports for use, with 26 multifunction GPIO pins,2 SPIs and 2 I2C controllers,2 UART and 16 PWM control interfaces. 3 12-bit ADC analog acquisition interfaces.
The pin definition is as follows:



2.2 What is Micro Python programming

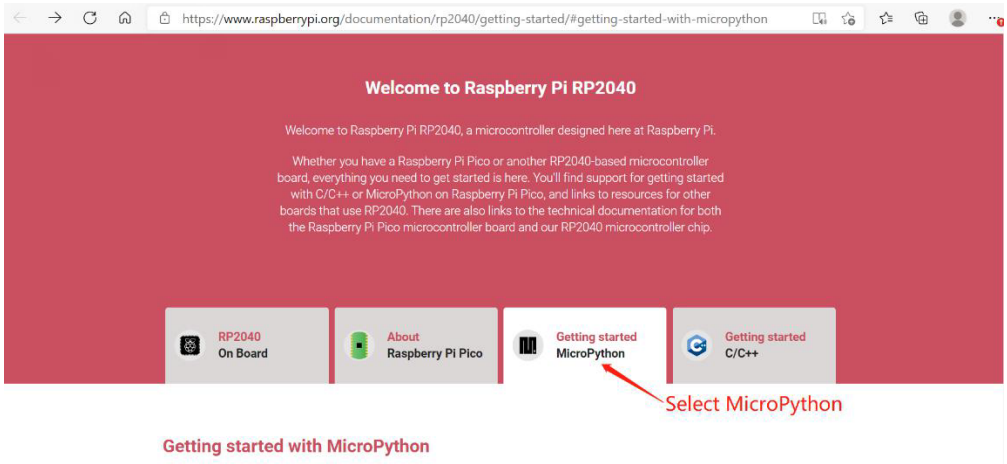
Micro Python is a complete software for the Python3programming language, written in C and run on a microcontroller, and Micro Python is a full Python compiler and operating system running on top of microcontroller hardware. Provides the user with an interactive prompt (REPL) to

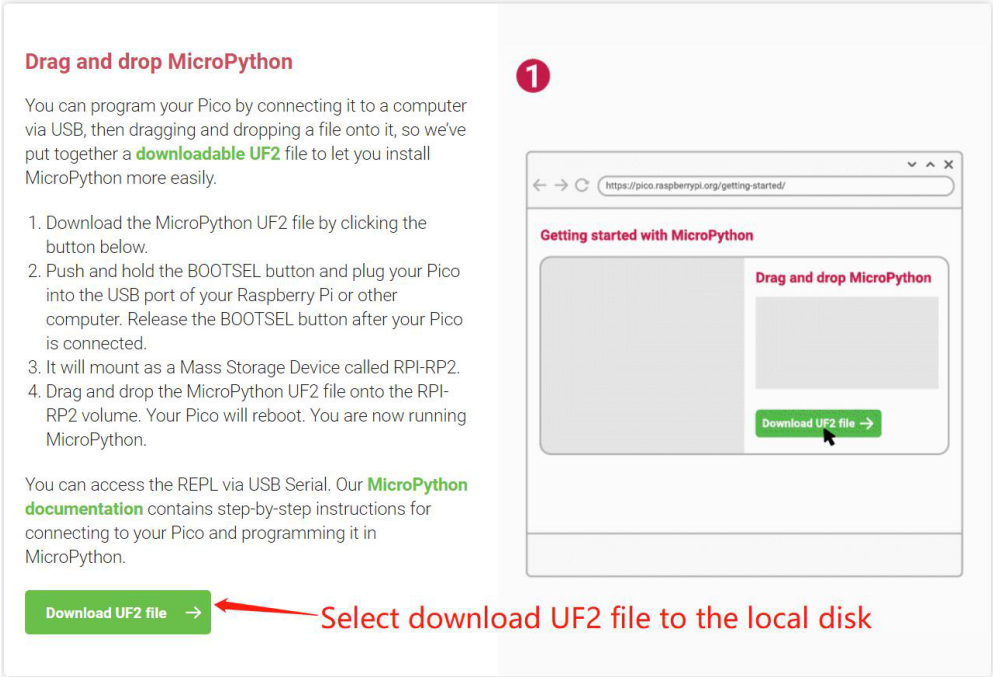
immediately execute the supported commands. In addition to the selected core Python library, Micro Python includes the underlying hardware modules that give programmers access.

Details can be viewed at: https://www.micropython.org/

2.2.1 Raspberry Pi Pico installs Micro Python support firmware

If the Raspberry Pi Pico motherboard supports Micro Python programming, we need to download the UF2 file on the official website and copy it to Pico's USB stick on the computer and go to the official website: https://www.raspberrypi.org/documentation/rp2040/getting-started/





The following image is a obtained UF2 file, which may be officially updated, so the name will change:

rp2-pico-20210205-unstable-v1.14-8-g1f800cac3.uf2

Connect the USB interface of the Pico motherboard with the Micro USB cable and the USB interface on the PC side, if the motherboard without burning firmware will produce a USB stick with the name: RPI-RP2;

Step 1: first hold down the "BOOTSEL", and then insert the computer side USB port;

Step 2: insert the computer end and release the button of "BOOTSEL", you can appear the name RPI-RP2 USB stick, and then the previously downloaded UF2 file copied into the USB stick, the USB stick will automatically disappear;

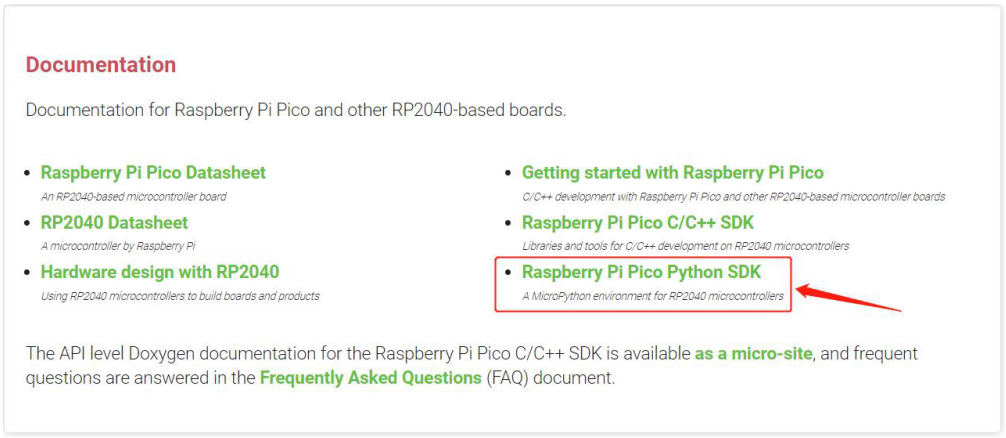
Step 3: Open PC Device Manager and we can find the virtual serial device in the port, as shown below:

Note:

If we need to confirm whether it is COM4shown above, we can unplug the device to see if the port disappears, and if it does, it means that the device is our Pico motherboard and supports programming by Micro Python.

2.3 Introduction to the Raspberry Pi Pico motherboard Micro Python library

The Raspberry Pi Foundation provides a very detailed Python SDK PDF document that details the various built-in functions, as well as the specific usage of the corresponding functions and the case notes provided, which we need to use to query the various built-in function functions for implementation.

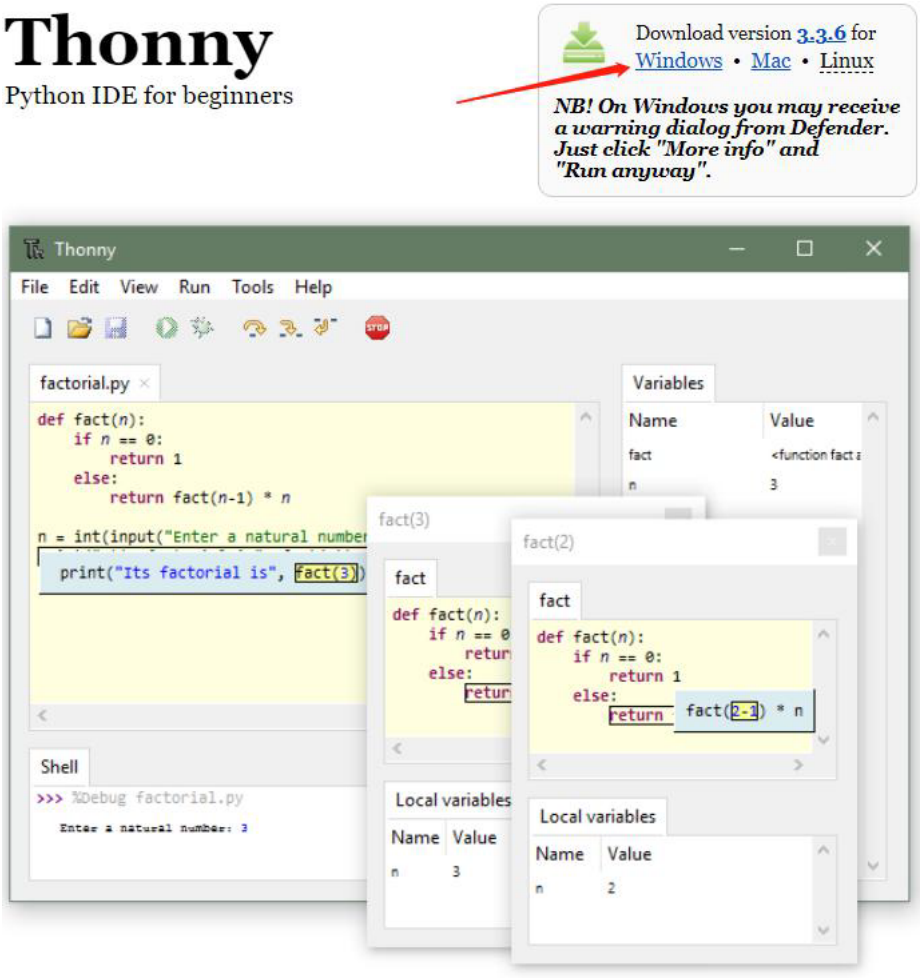


3 Use thonny software for Python development

Python programming has evolved to the present day with a number of very powerful IDE programming software, such as PyCharm IDE, Eclipse, Spyder, and Visual Studio Code, but for beginners, choosing a lightweight, easy-to-use IDE is the best option, and Thonny is what we call a small IDE, at Windows Available on, MAC, or Linux, it supports syntax coloring, code auto-complement, debug, and most importantly, the Python compiler on the Raspberry Pi Pico motherboard.

Official website: https://thonny.org/

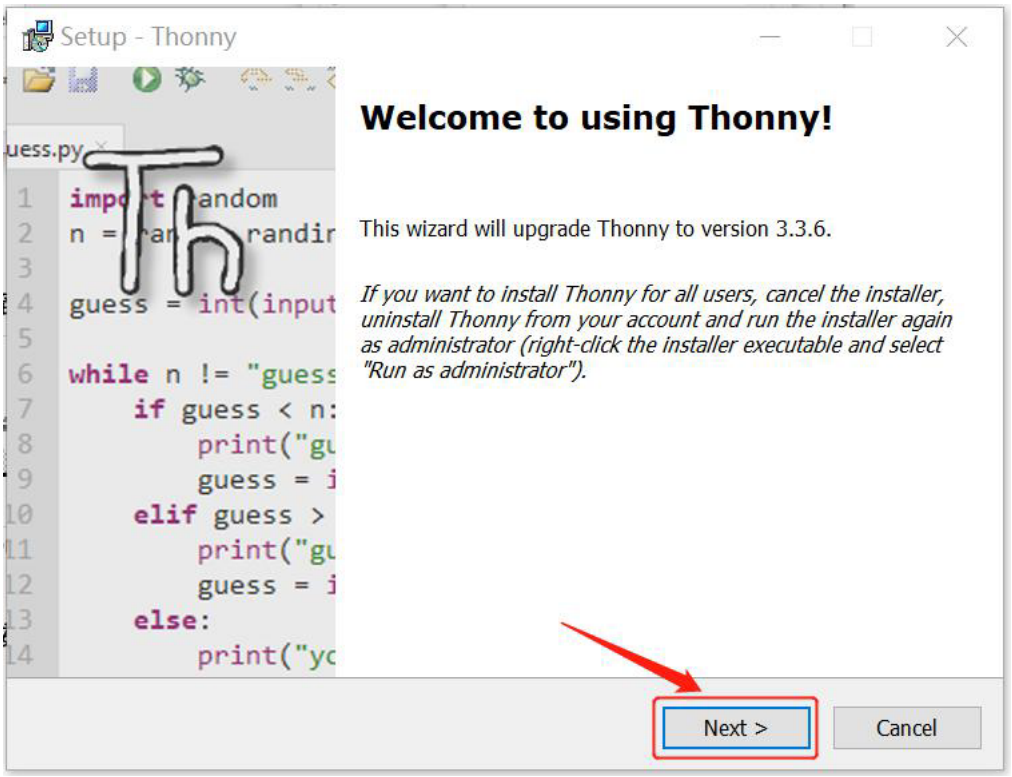
Go to the website to download the latest Thonny software, as follows:

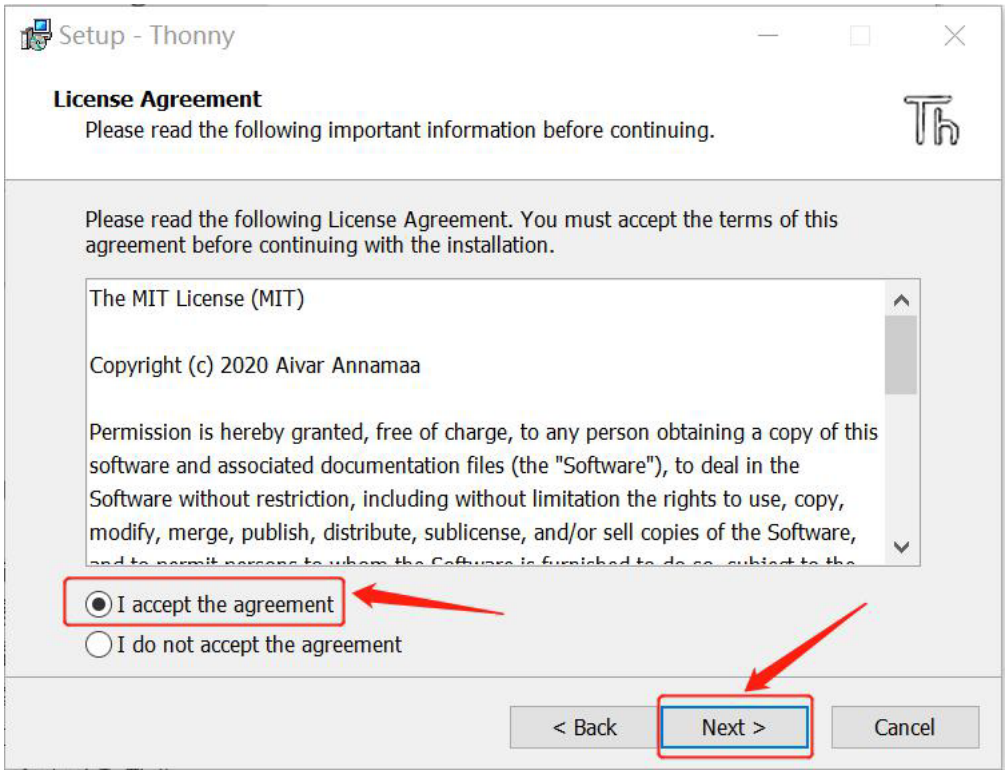


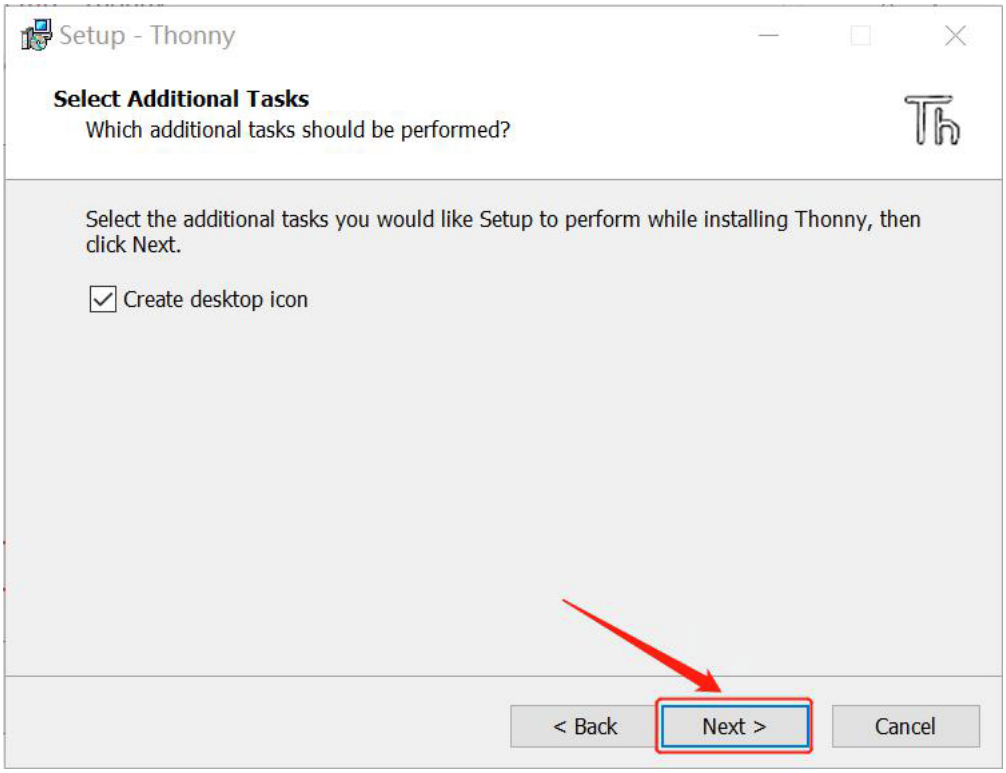
After downloading to the local area, select the corresponding installer, as shown below:

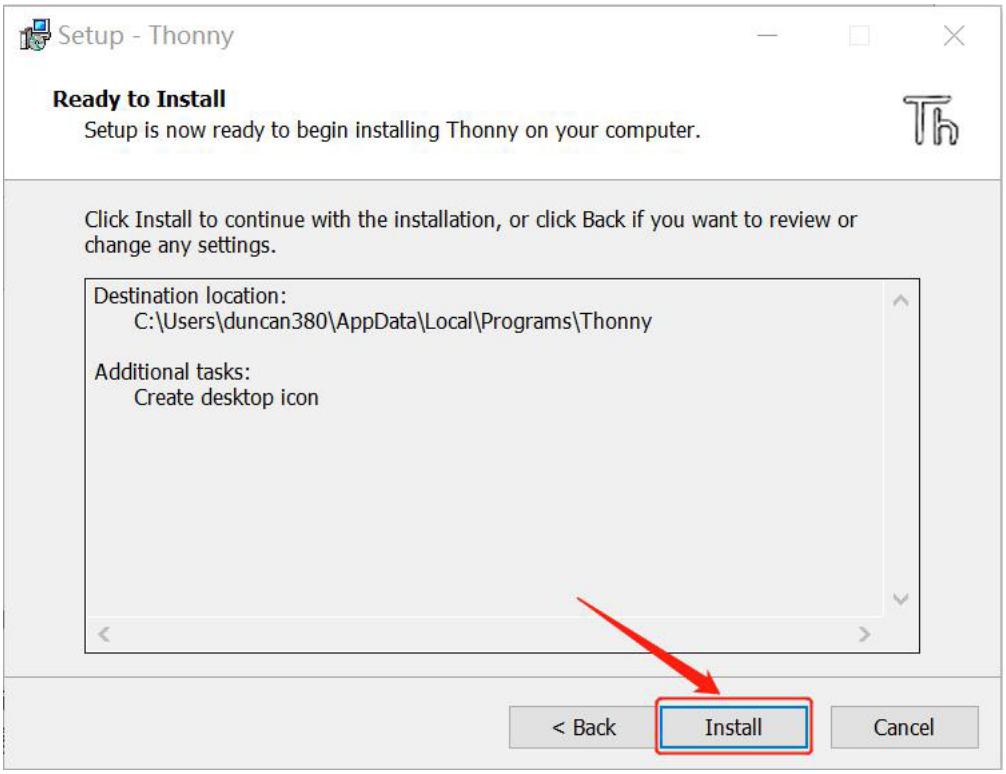
thonny-3.3.6

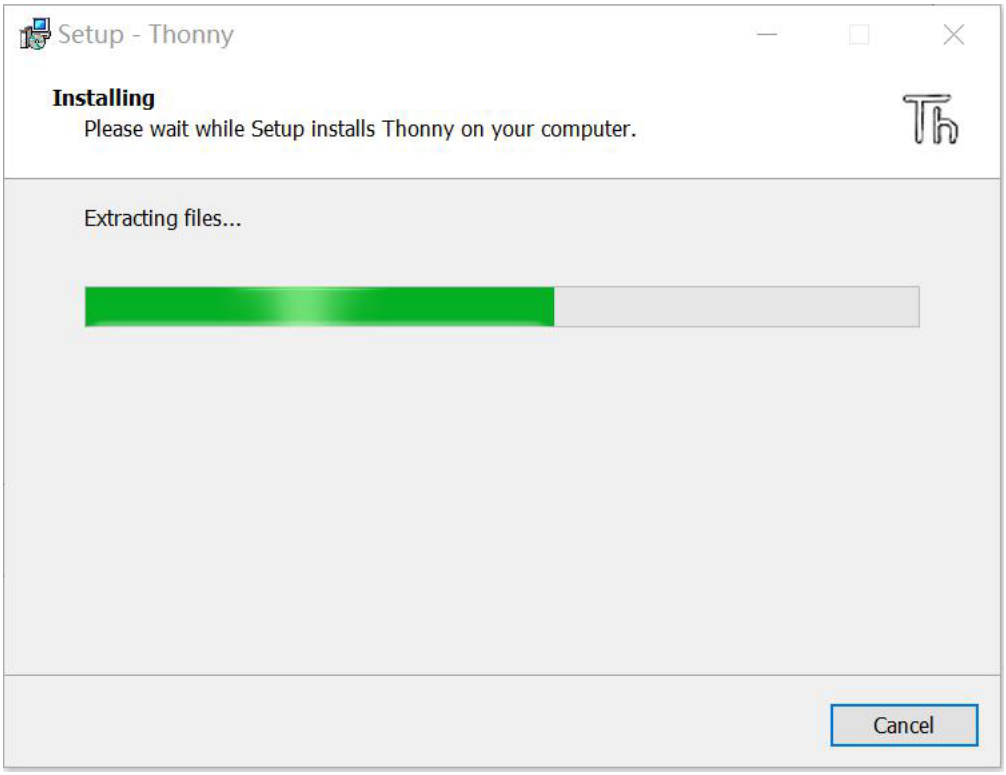
Double-click on the installer and pop up the following installation interface, point NEXT key:

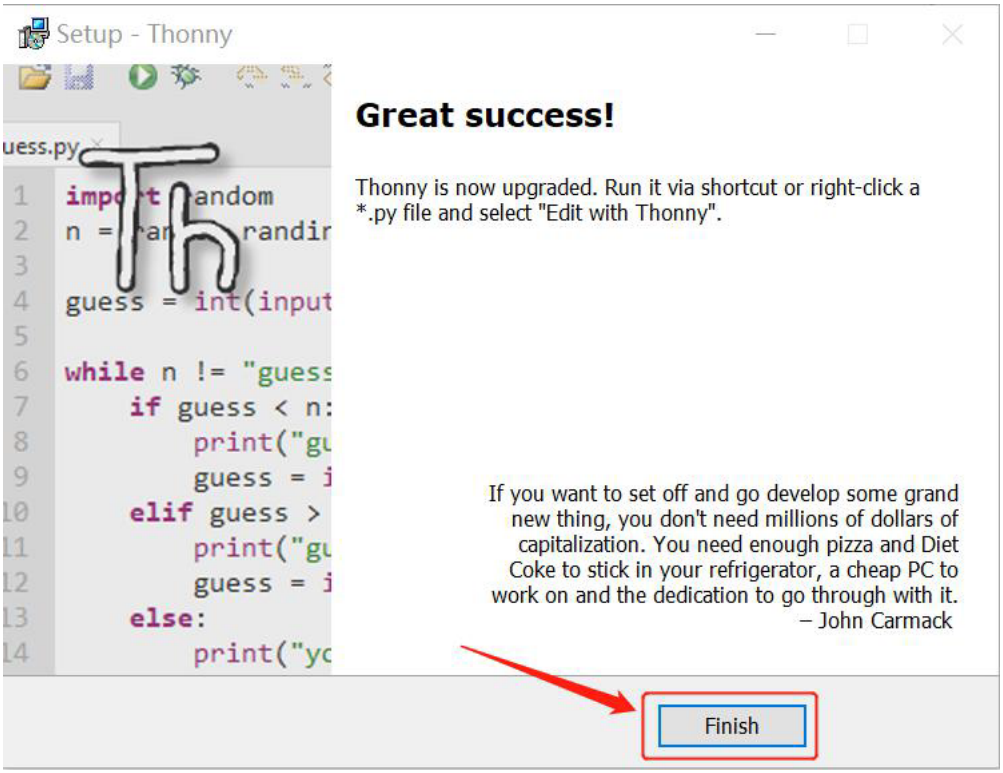




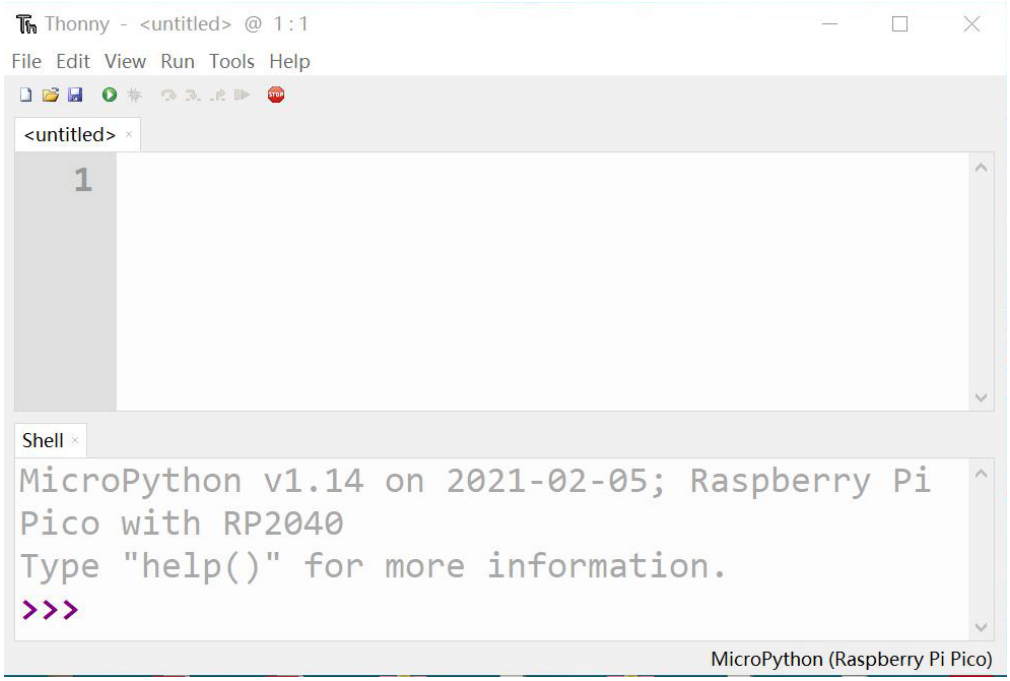

Setup - Thonny
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing Thonny, then click Next.
Create desktop icon
< Back
Next >
Cancel


Setup - Thonny
Ready to Install
Setup is now ready to begin installing Thonny on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Users\duncan380\AppData\Local\Programs\Thonny
Additional tasks:
Create desktop icon
< Back
Install
Cancel



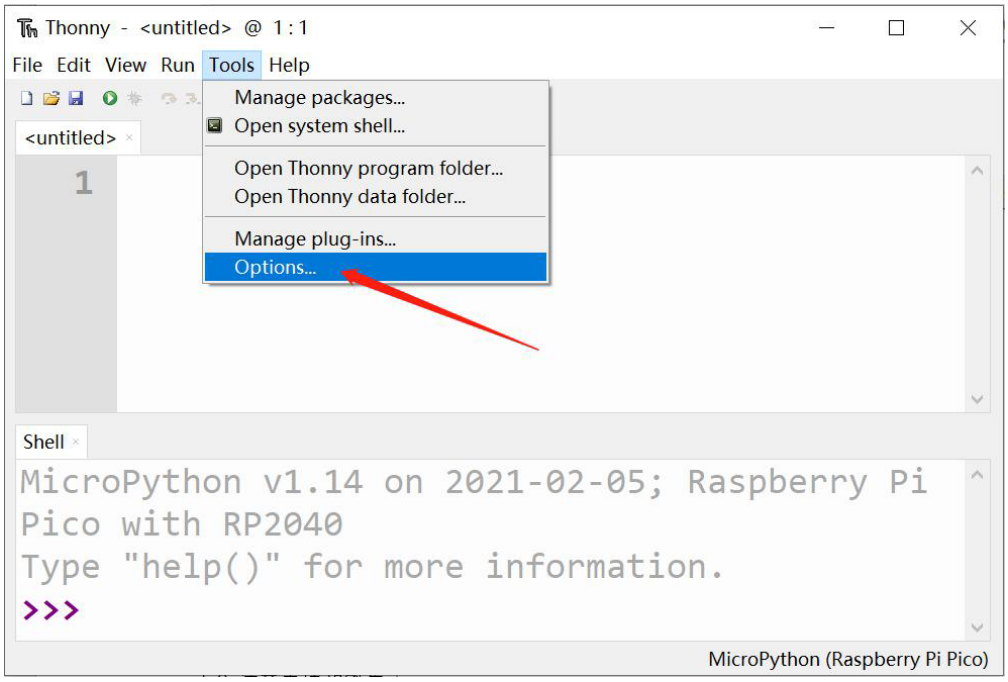


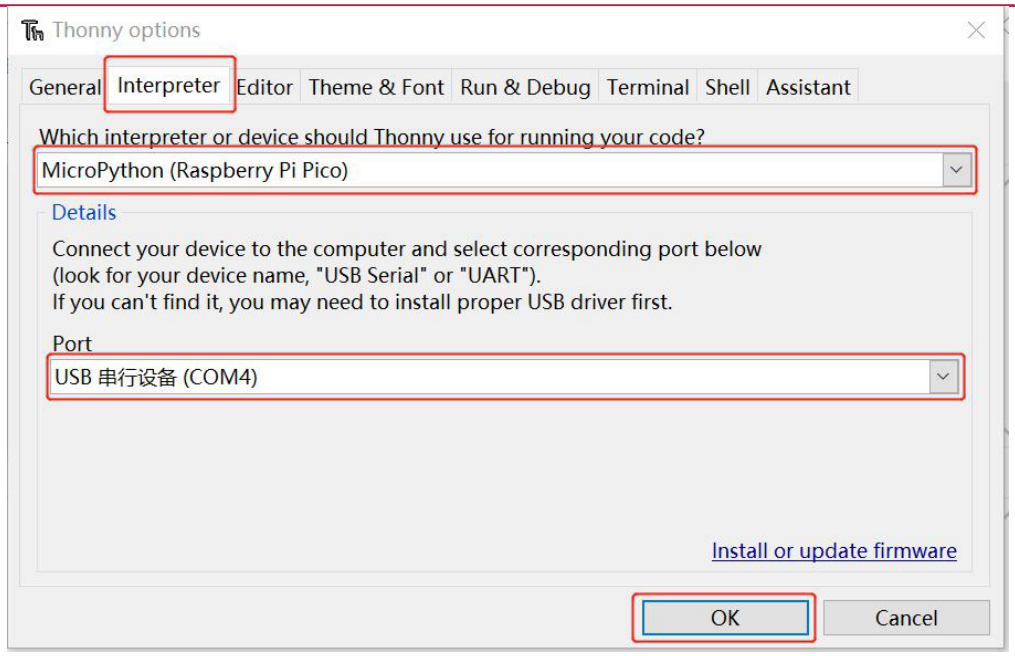
Double-click the following shortcut on the PC desktop to open the software, select the propriate language, and enter the software interface:



3.1 Select the right Raspberry Pi Pico programming

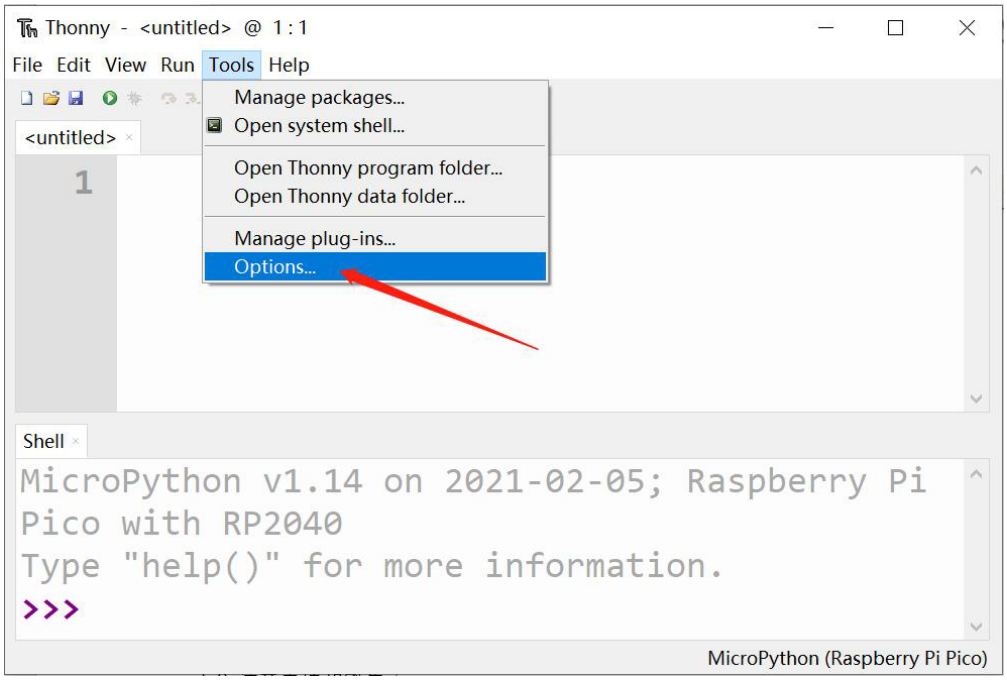
Select Tools/Settings in the menu bar and click on the pop-up settings window:

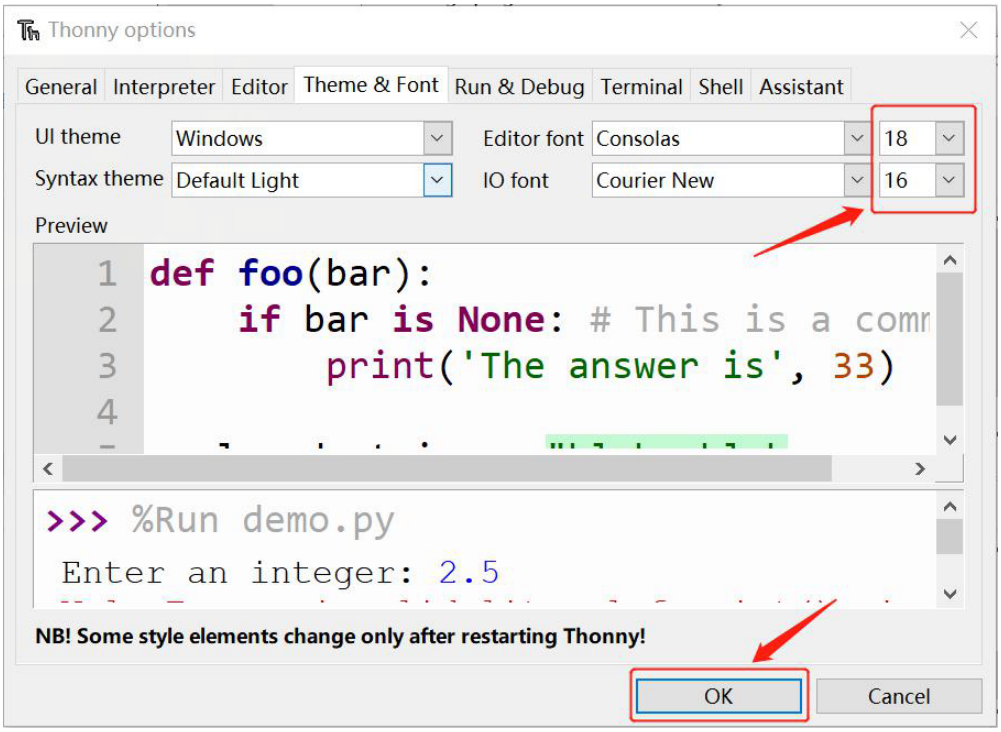




3.2 Modify the font size

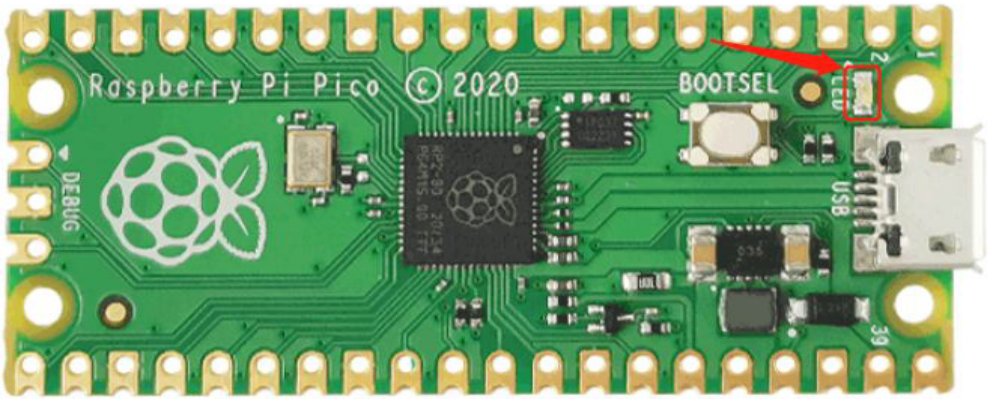
Select Tools/Settings/Subjects and Addresses



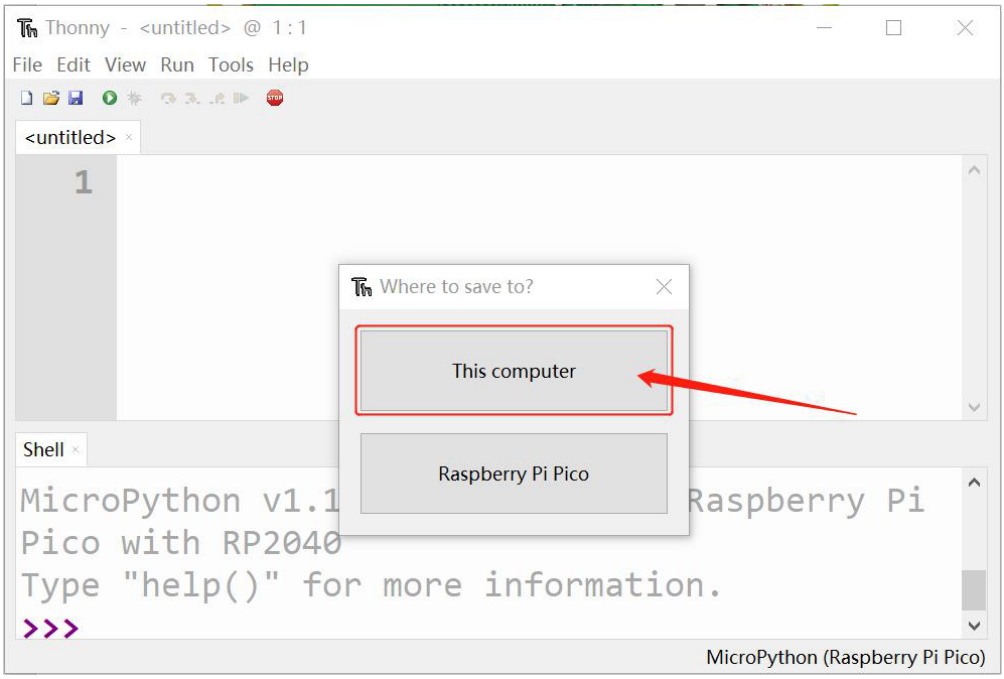


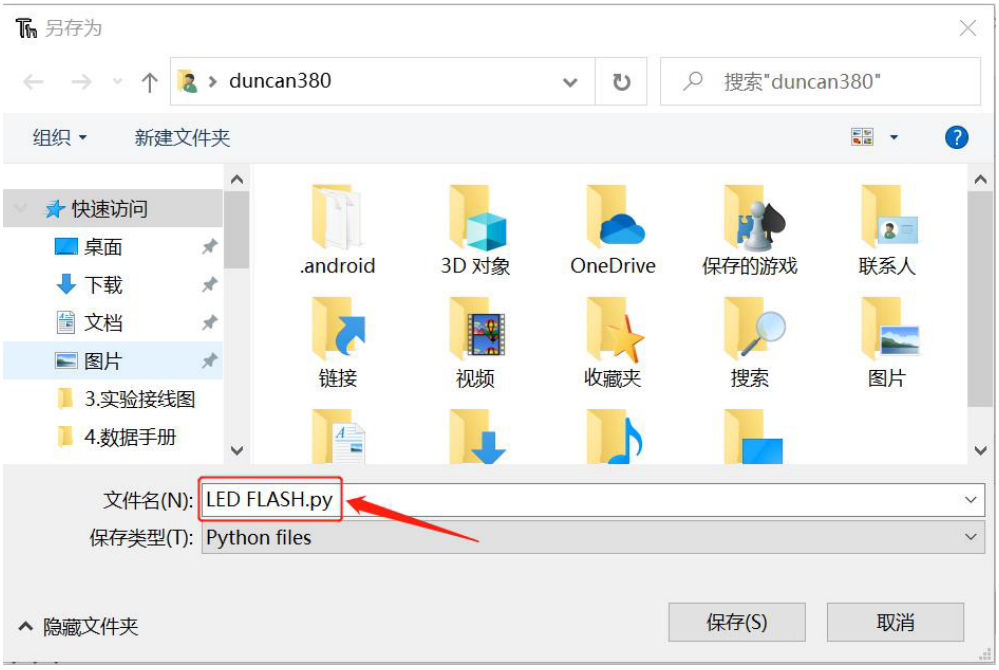
3.3 Write the LED light flashing program using Thonny

To light up the LED light you need to give a high level, turn off to give a low level, and now write a Thonny software that lets the L ED light (GP25) on the Pico board flash, as shown below:

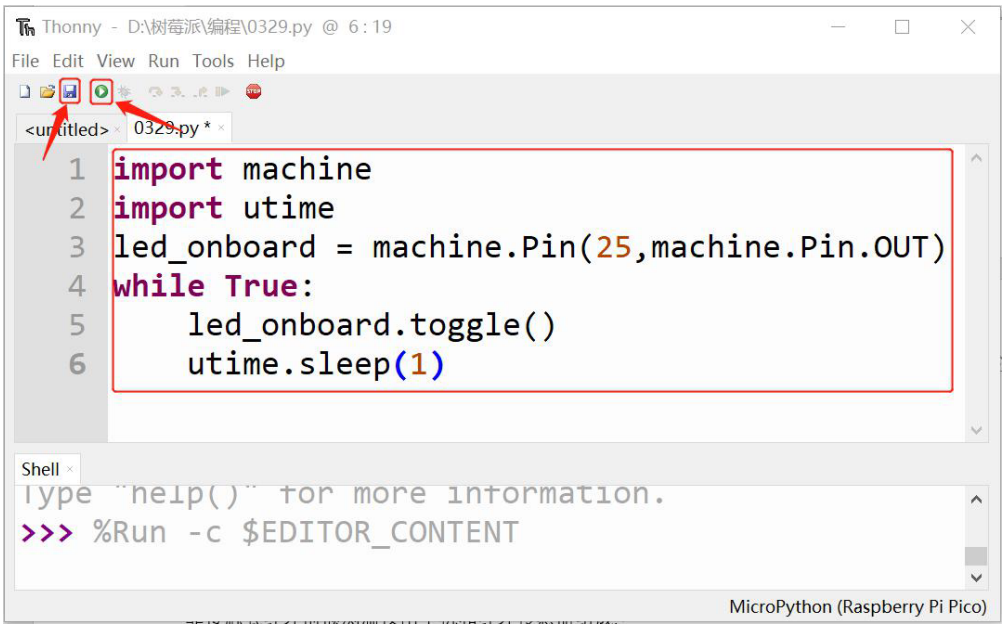


Step 1: Open the Thonny software and create a new file named ".py"**


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
1
Where to save to?
This computer
Raspberry Pi Pico
Shell
MicroPython v1.1
Raspberry Pi
Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico)

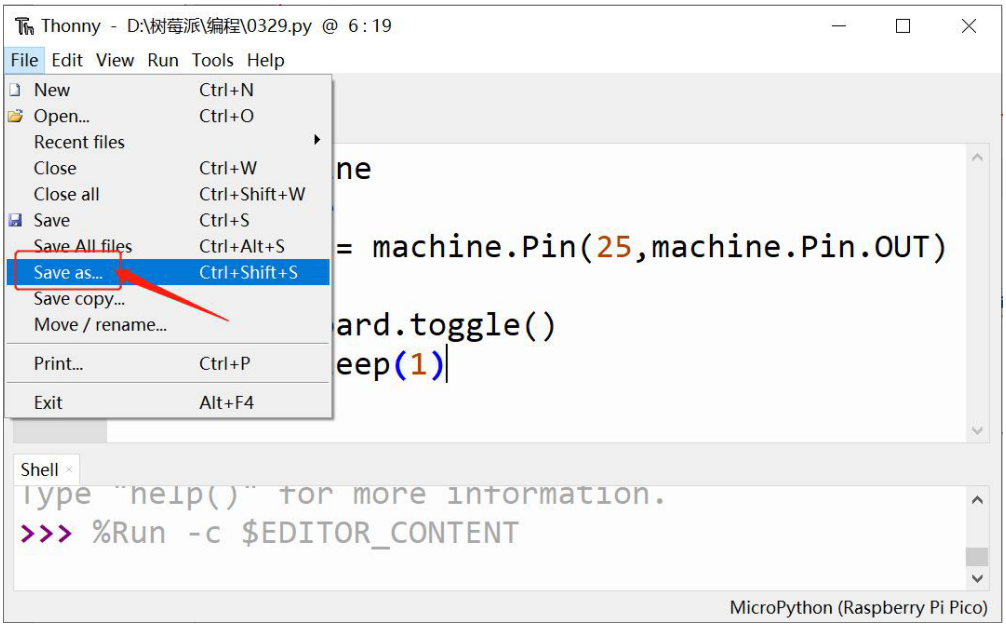

另存为
duncan380
搜索"duncan380"
组织
新建文件夹
快速访问
桌面
下载
文档
图片
3.实验接线图
4.数据手册
.android
3D 对象
OneDrive
保存的游戏
联系人
链接
视频
收藏夹
搜索
图片
文件名(N): LED FLASH.py
保存类型(T): Python files
隐藏文件夹
保存(S)
取消

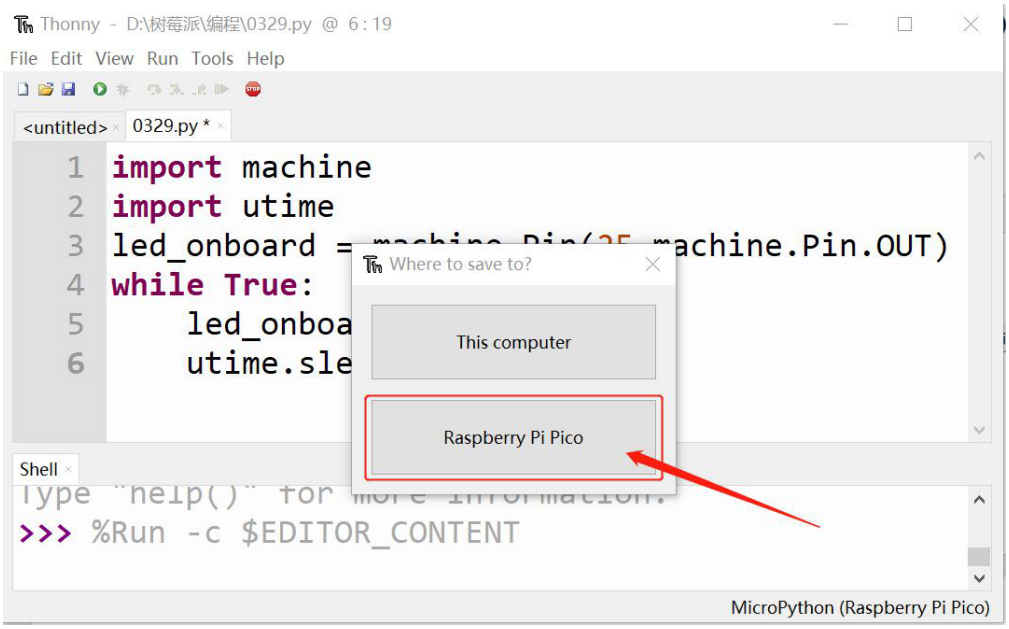
Step 2: After writing the following code, click the SAVE button:

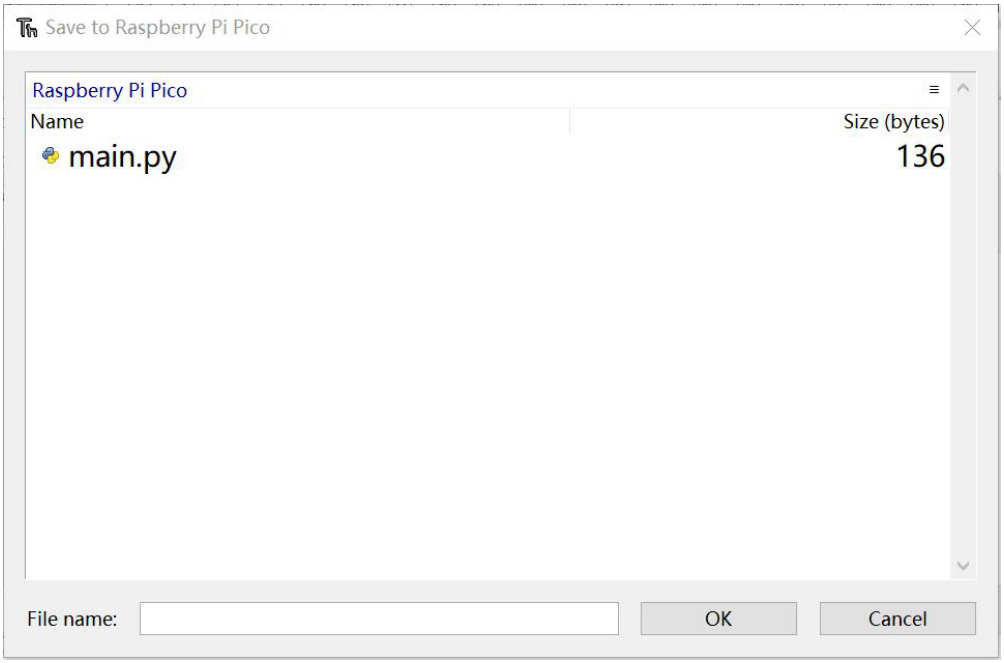


Step 3: Click the RUN button to run the program, and the LED light will flash according to the program.

Note: The program saved according to the above method is on the computer, and if you unplug the USB at this time, the LED will not flash. If you want to cure the program to the Pico motherboard, you can automatically run the program as long as you power it on the device. Here's how:
All you need to do is select Save as and save as m in the file [main.py] menu



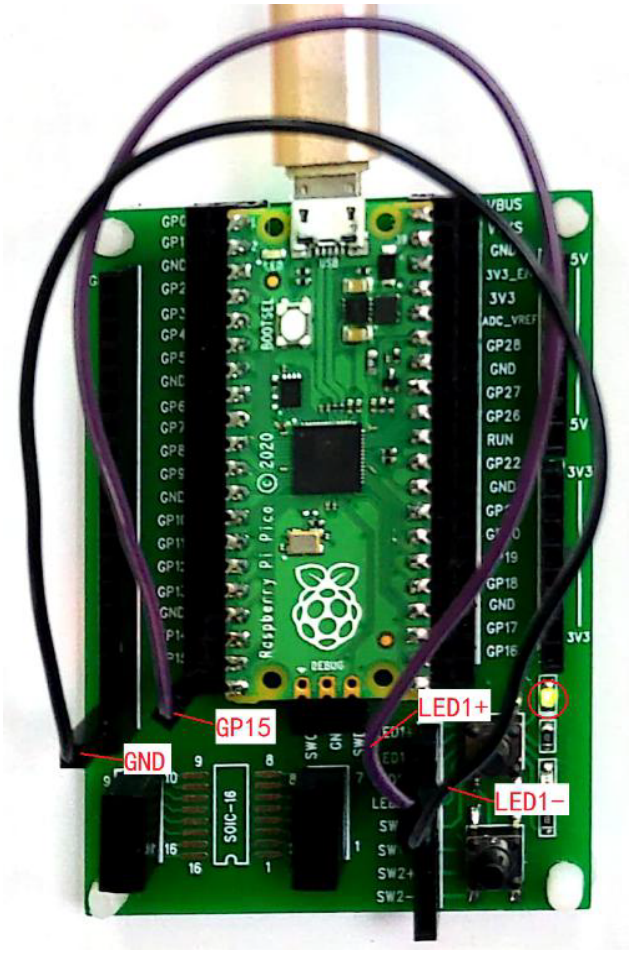

Thonny - D:\树莓派\编程\0329.py @ 6 : 19
File Edit View Run Tools Help
<untitled>
0329.py *
1 import machine
2 import utime
3 led_onboard =
4 while True:
5 led_onboa
6 utime.sle
Where to save to?
This computer
Raspberry Pi Pico
Shell
>>> %Run -c $EDITOR_CONTENT
MicroPython (Raspberry Pi Pico)


Save to Raspberry Pi Pico
Raspberry Pi Pico
Name
Size (bytes)
main.py
136
File name:
OK
Cancel

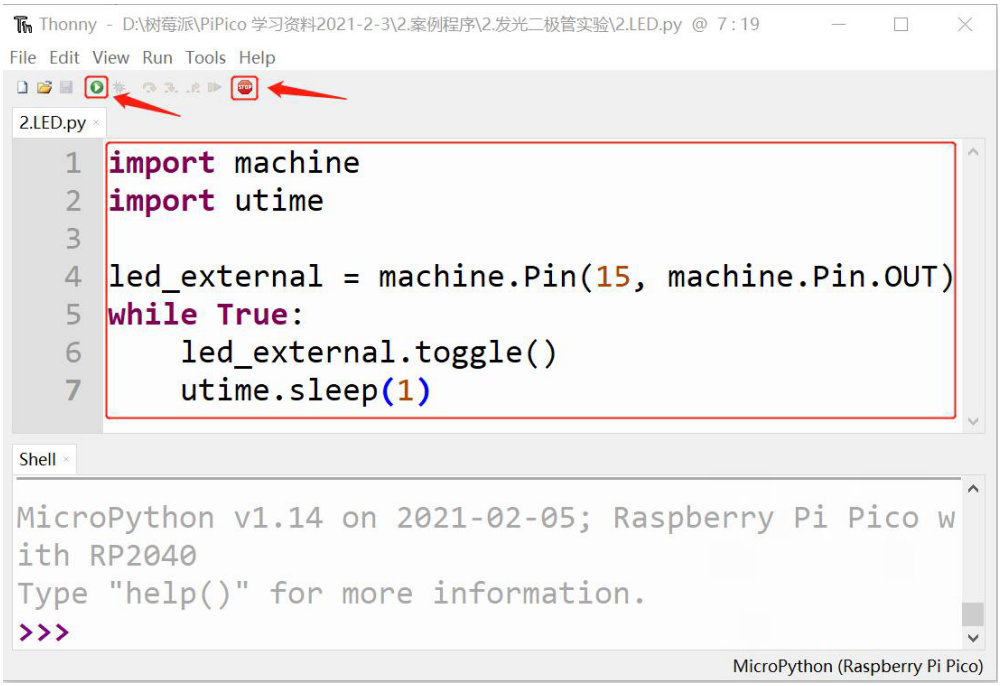
4 Case Programming

4.1 Light-emitting diode experiments

The wiring is as follows:

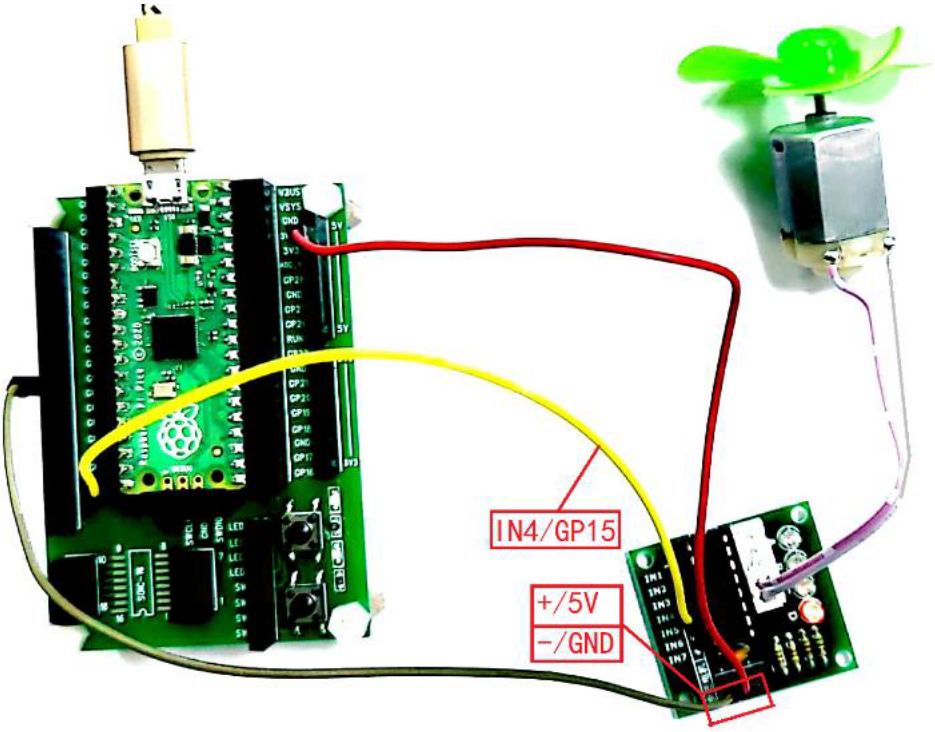


The program code is as follows:



4.2 DC motor fan experiment

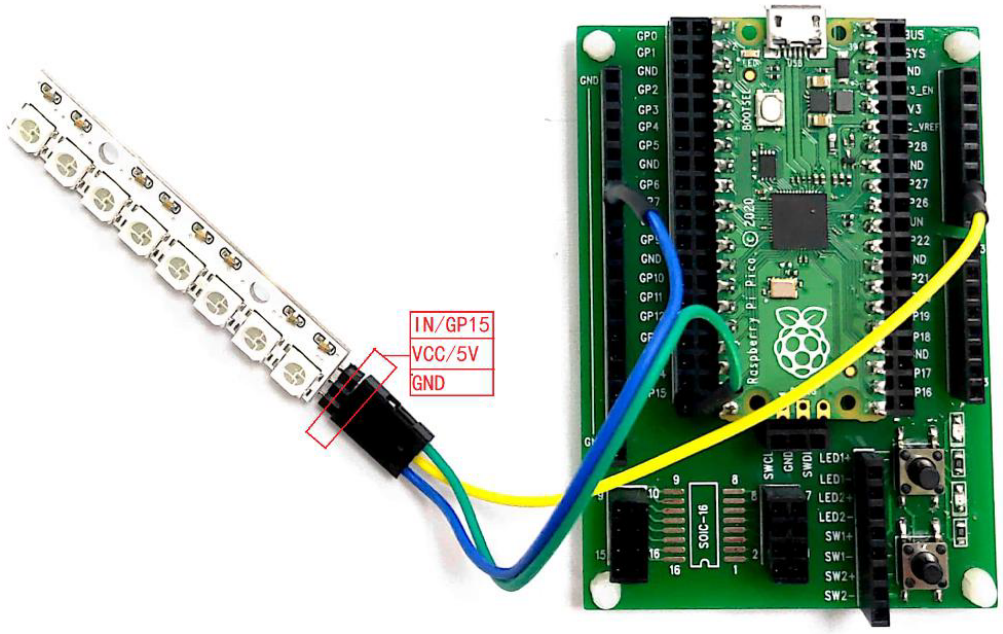
The wiring is as follows:



The program code is as follows:

4.3 LED strip

The wiring is as follows:



The program code is as follows:

Thonny - D:\树莓派\Raspberry-Pi-Pico-Sensor\RGB_LED_ws2812b\rainbow.py @ 21 : 26

File Edit View Run Tools Help

rainbow.py | [ws2812b.py]

```
import time                                          #include delay time
import ws2812b                                       #include WS2812B

numpix = 8                                          #define LED number
pin = 15                                             #define pin function:GP15
strip = ws2812b.ws2812b(numpix, 0,pin)

RED = (255, 0, 0)                                    #define LED brightness(RED,GREEN,BLUE)
ORANGE = (255, 165, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
INDIGO = (75, 0, 130)
VIOLET = (138, 43, 226)
COLORS = (RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, VIOLET)     #define LED color change order

```

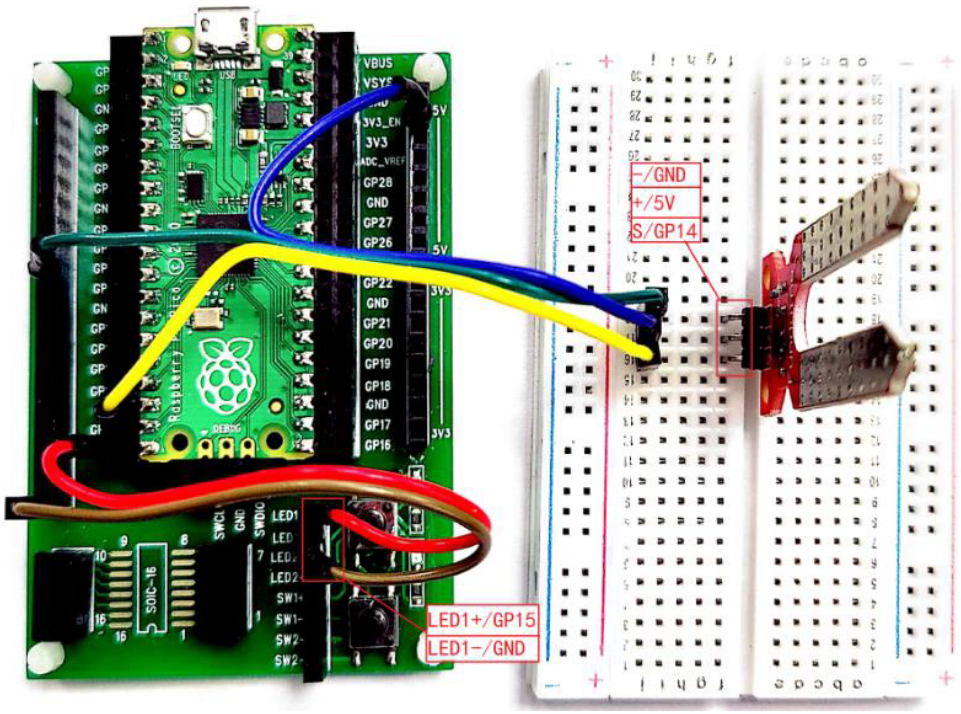
Shell

```
MicroPython v1.13-290-g556ae7914 on 2021-01-21; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

>>>
 Backend terminated or disconnected. Use 'Stop/Restart' to restart.
```

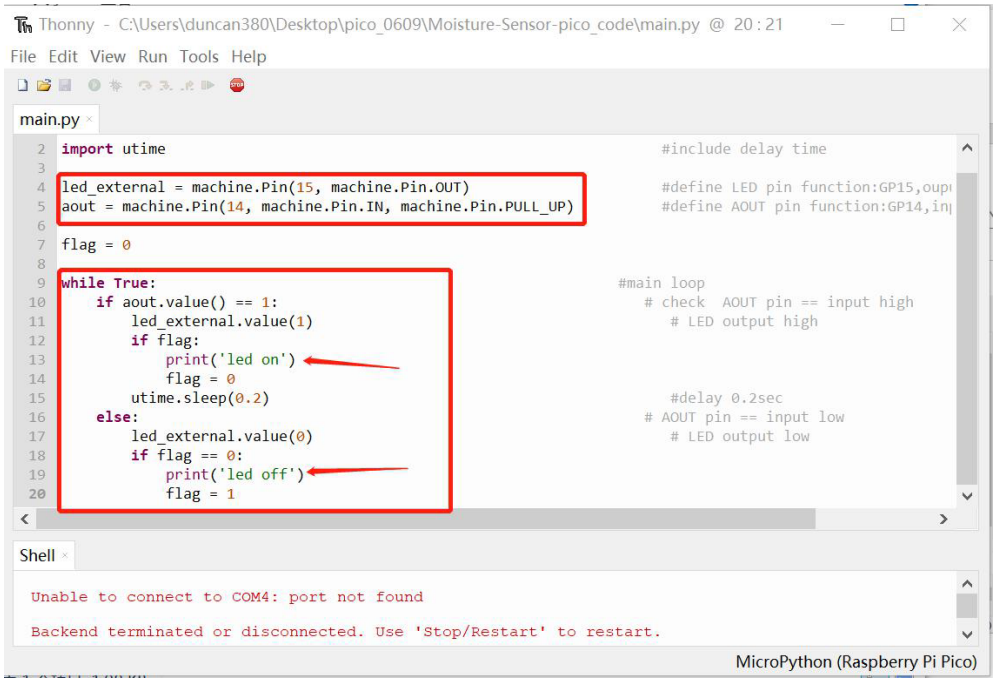
MicroPython (Raspberry Pi Pico)

4.4 Moisture Sensor

The wiring is as follows:

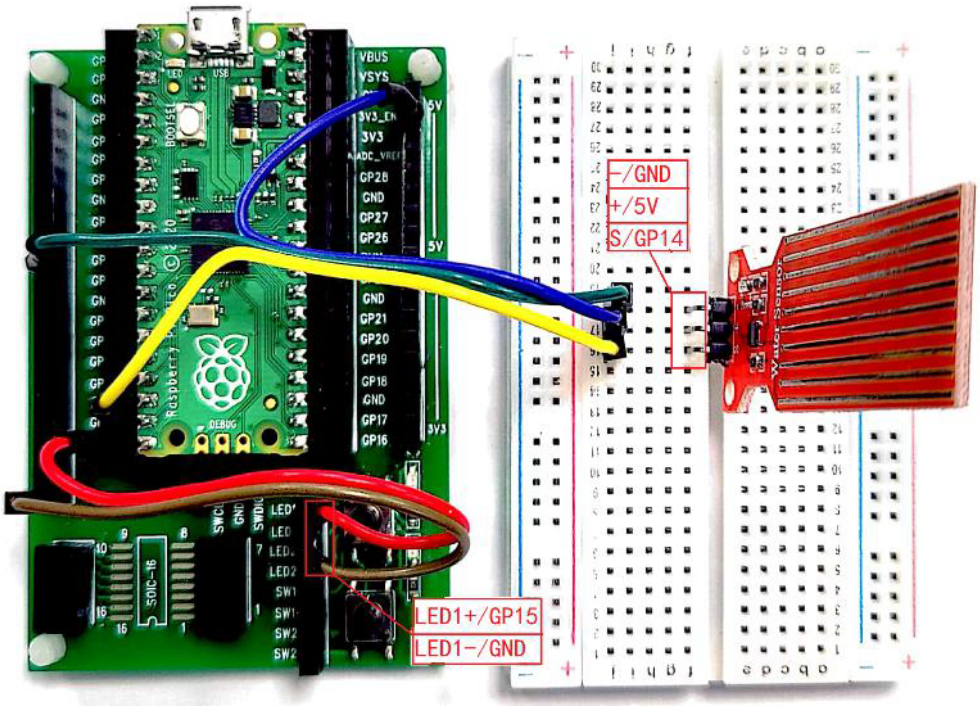


The program code is as follows:



4.5 Liquid Level Sensor

The wiring is as follows:



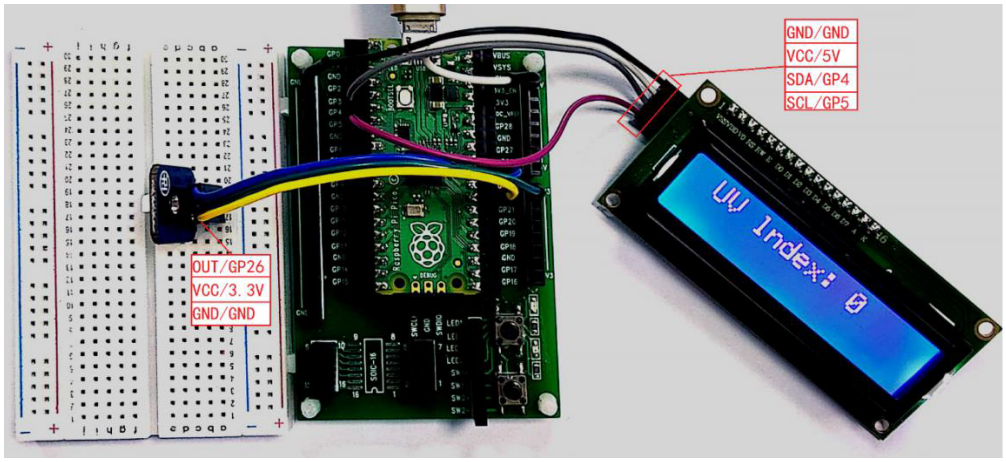
The program code is as follows:

```
import utime                                        #include delay time

led_external = machine.Pin(15, machine.Pin.OUT)     #define LED pin function:GP15,oupu
aout = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)   #define AOUT pin function:GP14,inp

flag = 0

while True:                                         #main loop
    if aout.value() == 1:                           # check  AOUT pin == input high
        led_external.value(1)                       #  LED output high
        if flag:
            print('led on')
            flag = 0
        utime.sleep(0.2)                            #delay 0.2sec
    else:                                           # AOUT pin == input low
        led_external.value(0)                       #  LED output low
        if flag == 0:
            print('led off')
            flag = 1
```

```
Unable to connect to COM4: port not found

Backend terminated or disconnected. Use 'Stop/Restart' to restart.
```

4.6 UV Sensor

The wiring is as follows:



The program code is as follows:

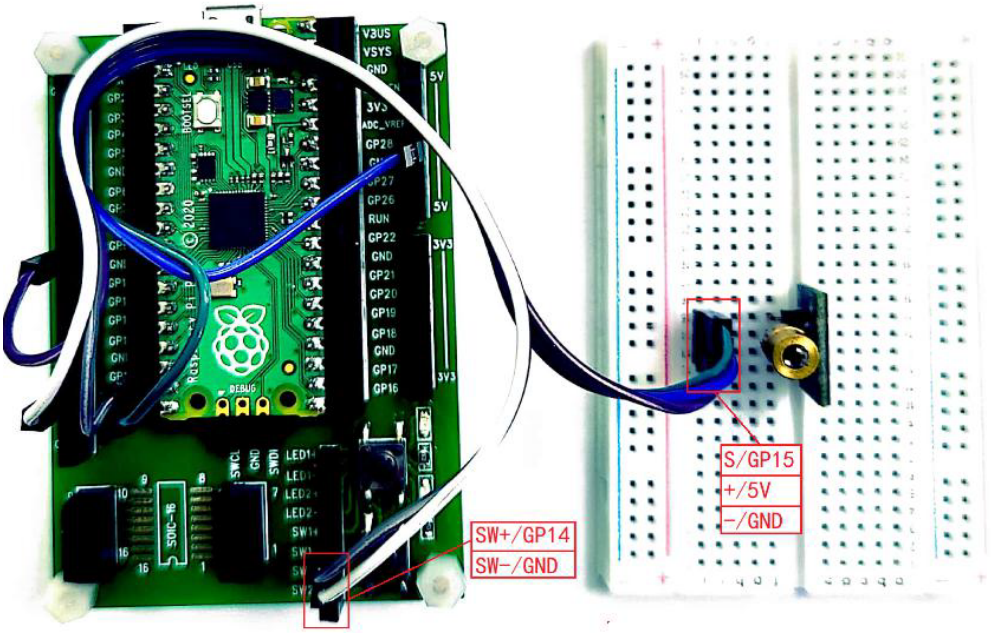
Operation “main.py” First save the "lcd1602_i2c.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

```
import machine
import utime

led_external = machine.Pin(15, machine.Pin.OUT)
aout = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)


while True:
    if aout.value() == 1:
        led_external.value(1)
        utime.sleep(0.2)
    else:
        led_external.value(0)
```

```
Unable to connect to COM4: port not found

Backend terminated or disconnected. Use 'Stop/Restart' to restart.
```

4.7 Laser out Sensor

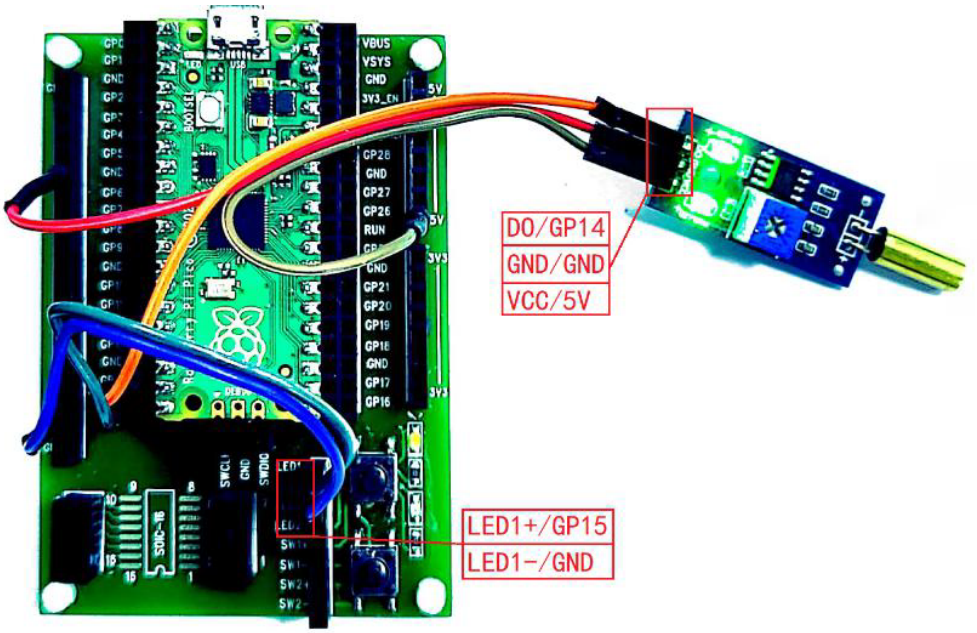
The wiring is as follows:



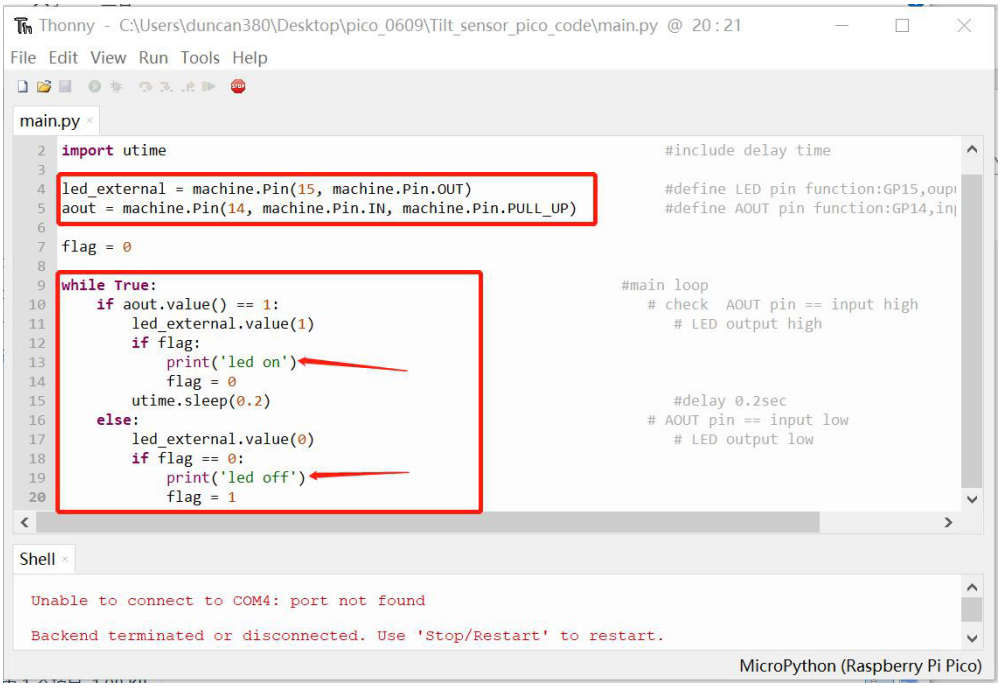
The program code is as follows:

4.8 Tilt Sensor

The wiring is as follows:

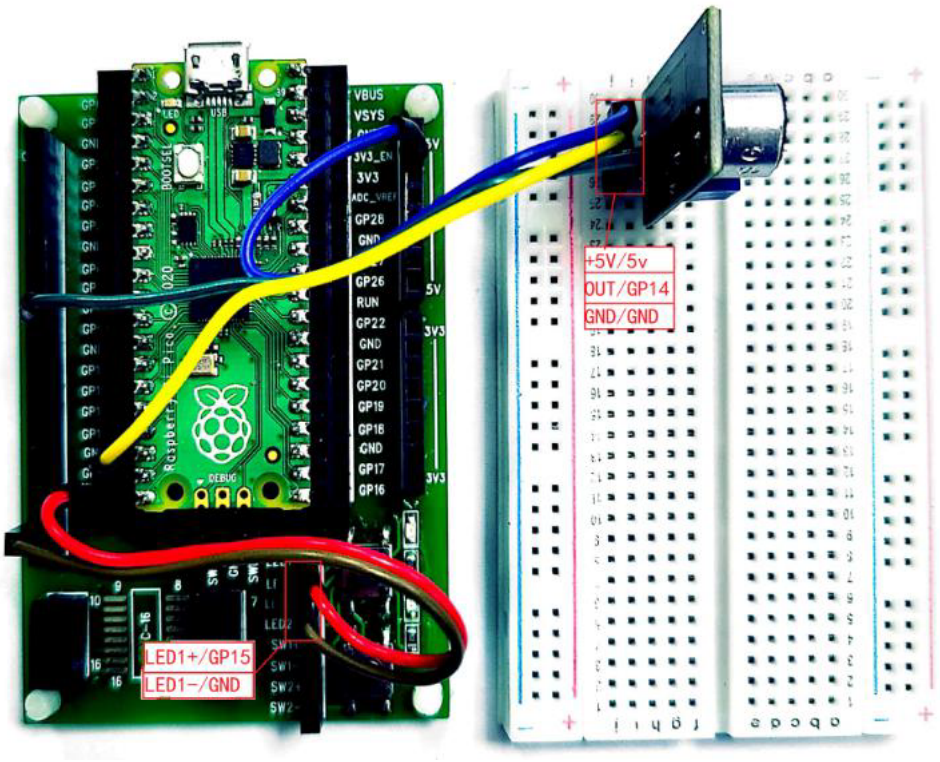


The program code is as follows:

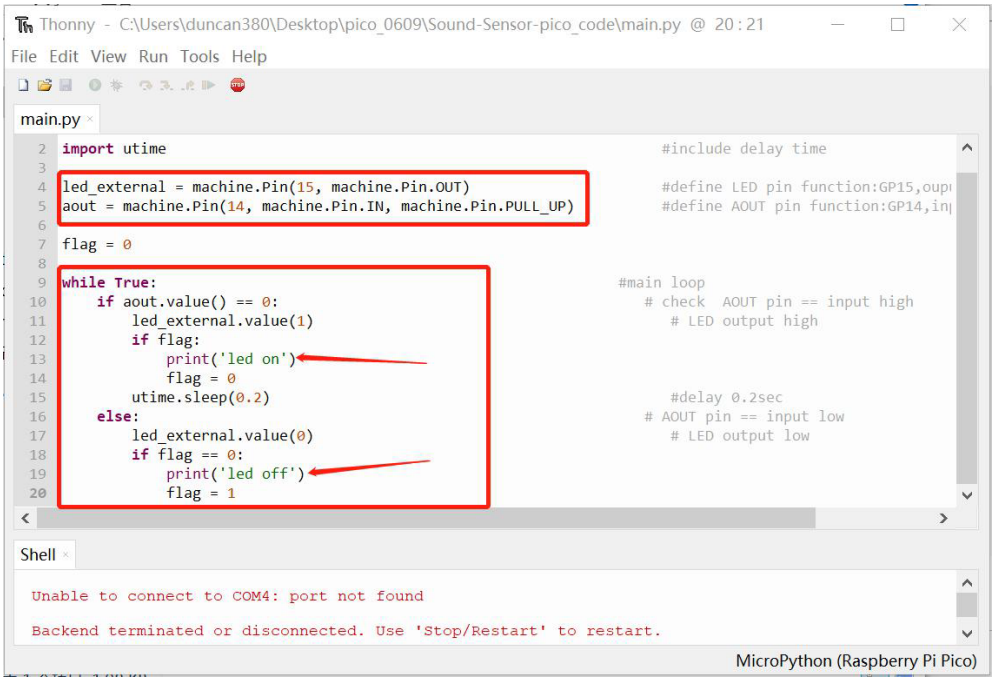


4.9 Sound Sensor

The wiring is as follows:

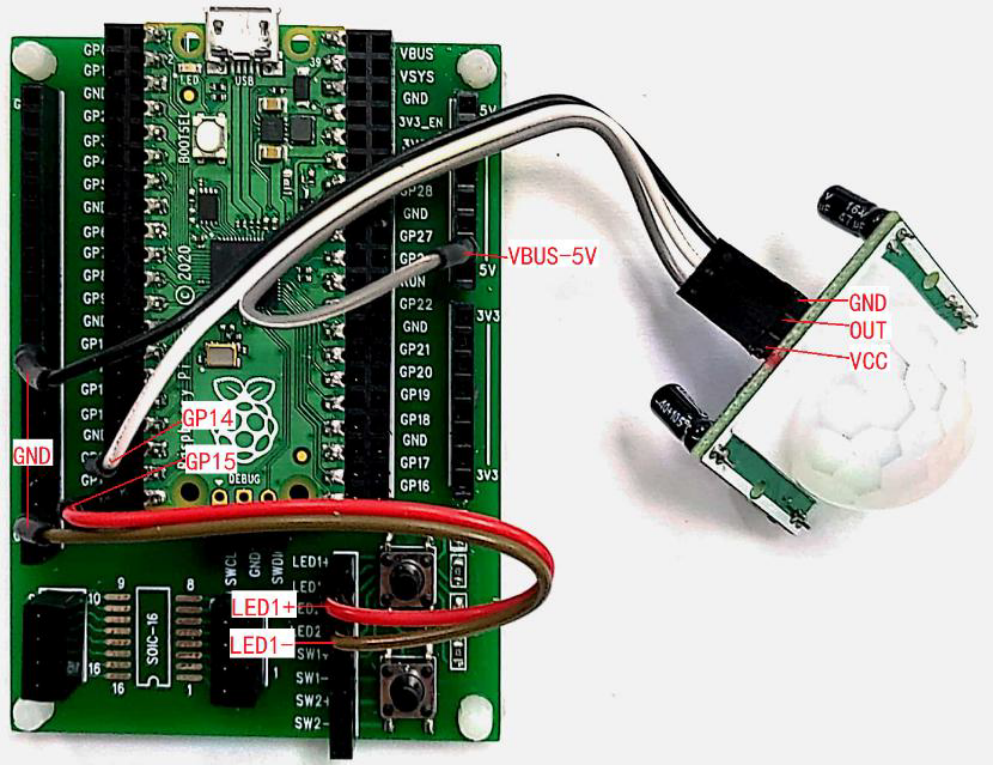


The program code is as follows:

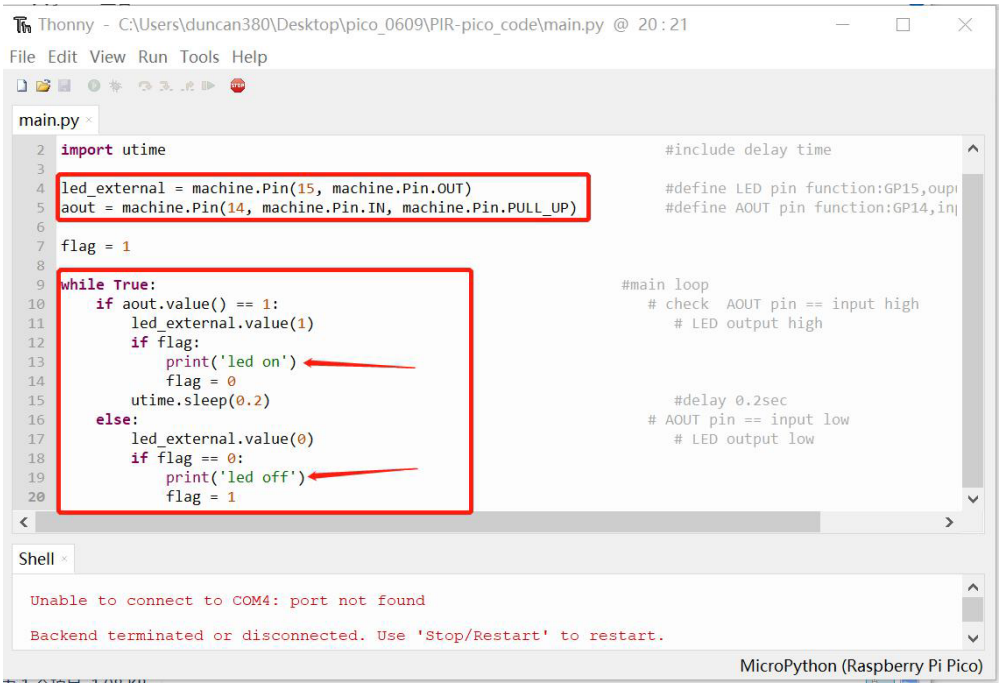


4.10 PIR Sensor

The wiring is as follows:

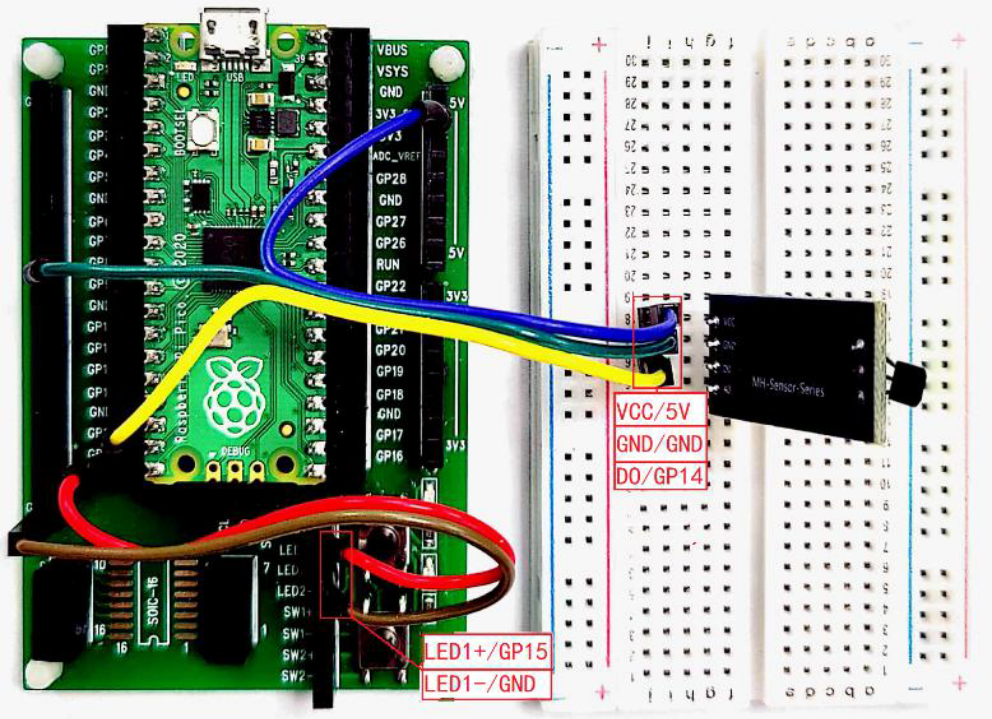


The program code is as follows:

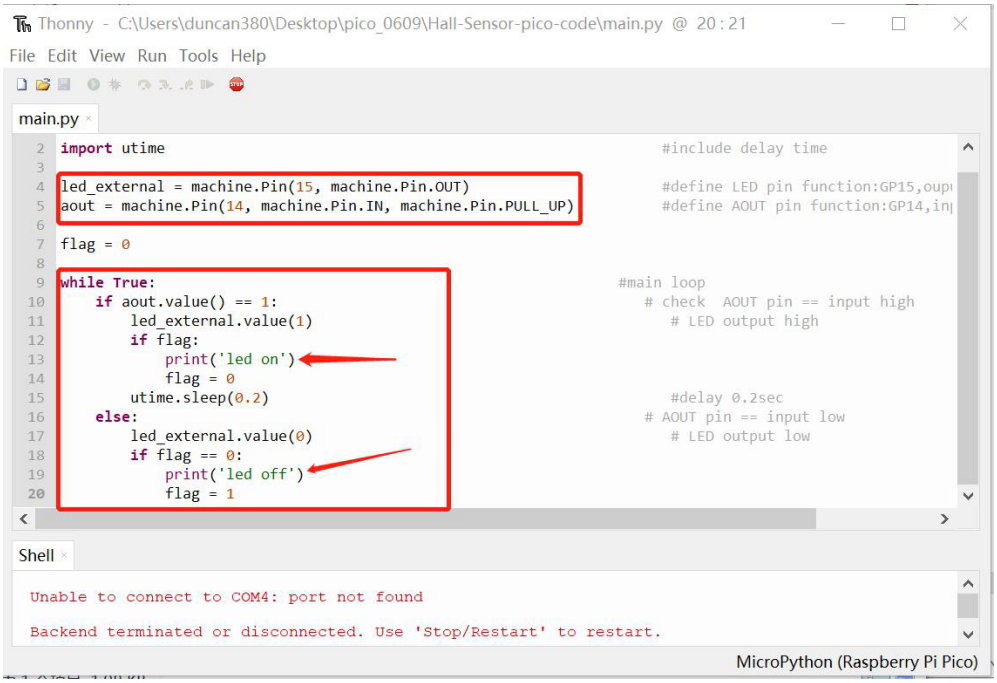


4.11 Hall Sensor

The wiring is as follows:

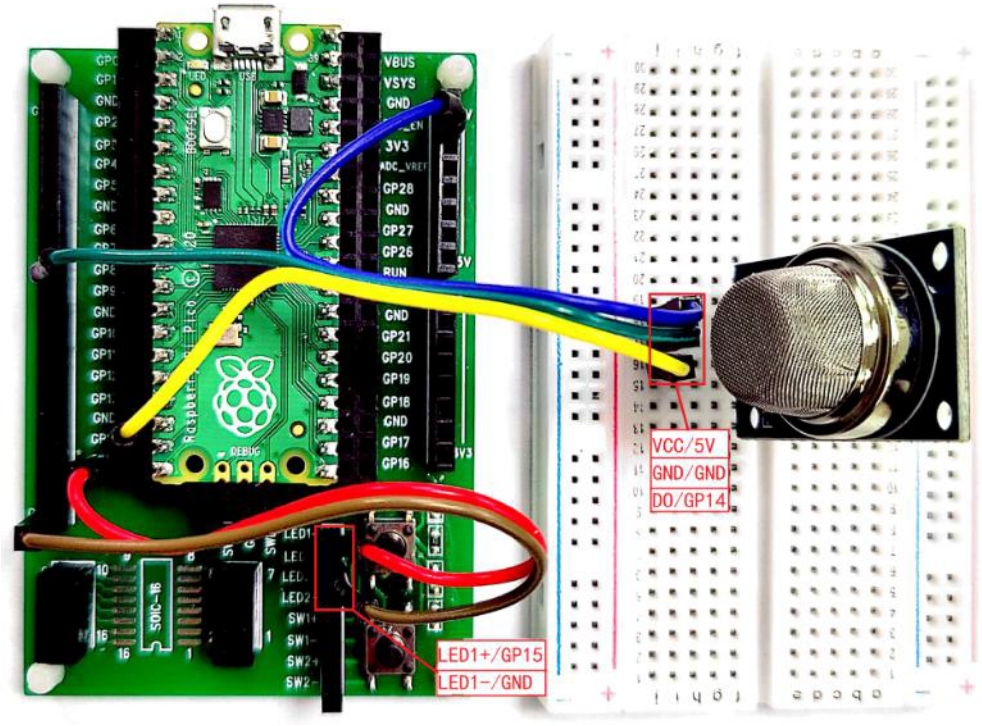


The program code is as follows:

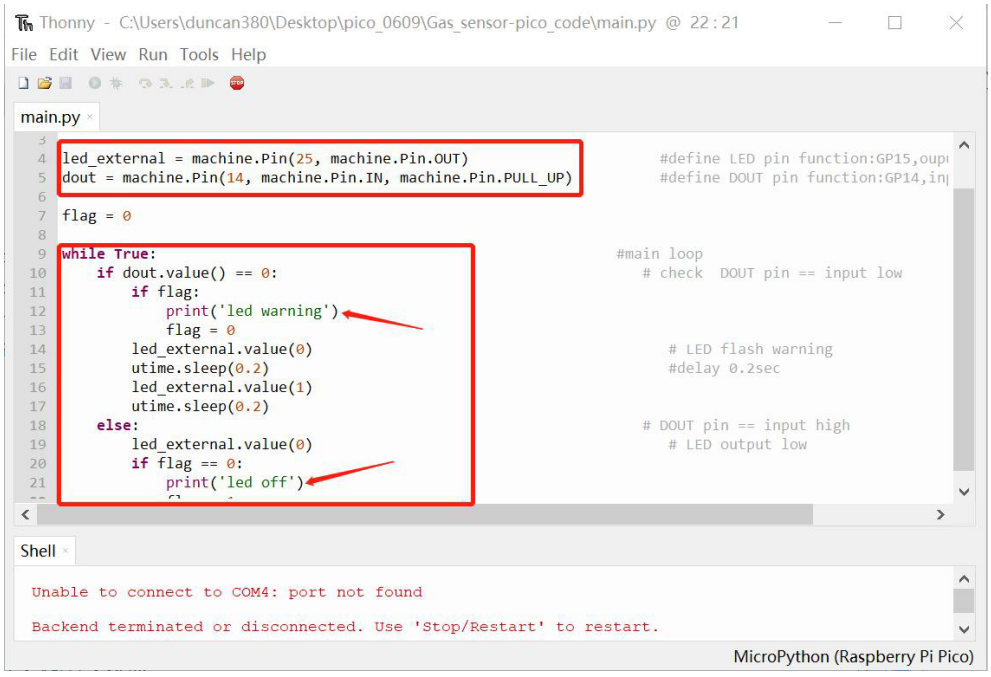


4.12 Gas Sensor

The wiring is as follows:

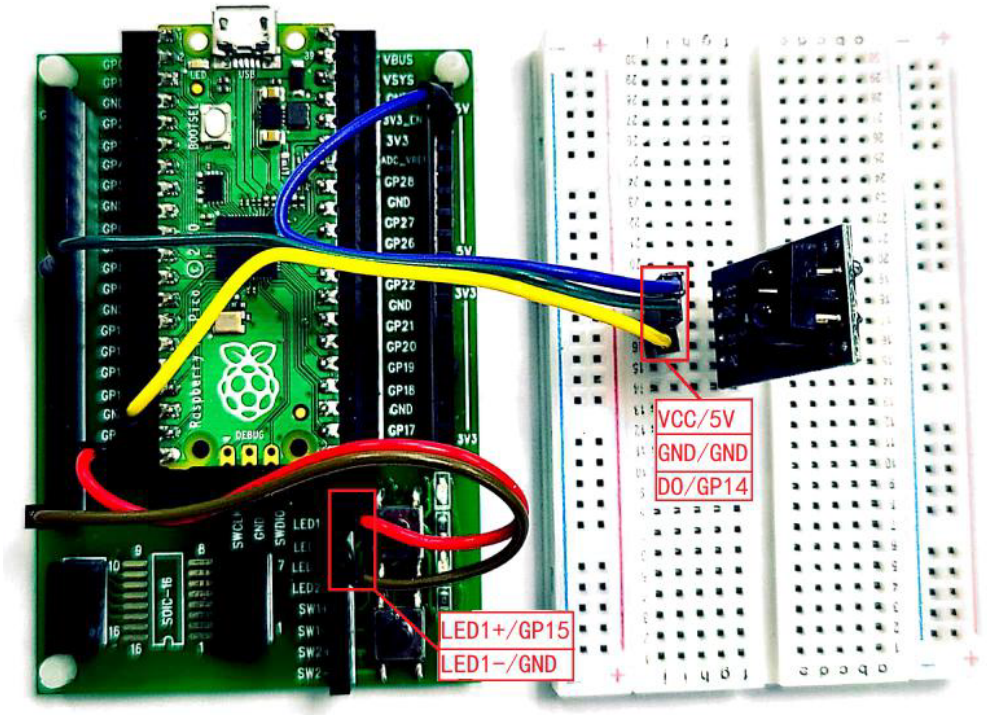


The program code is as follows:



4.13 Infrared Reflective Sensor

The wiring is as follows:

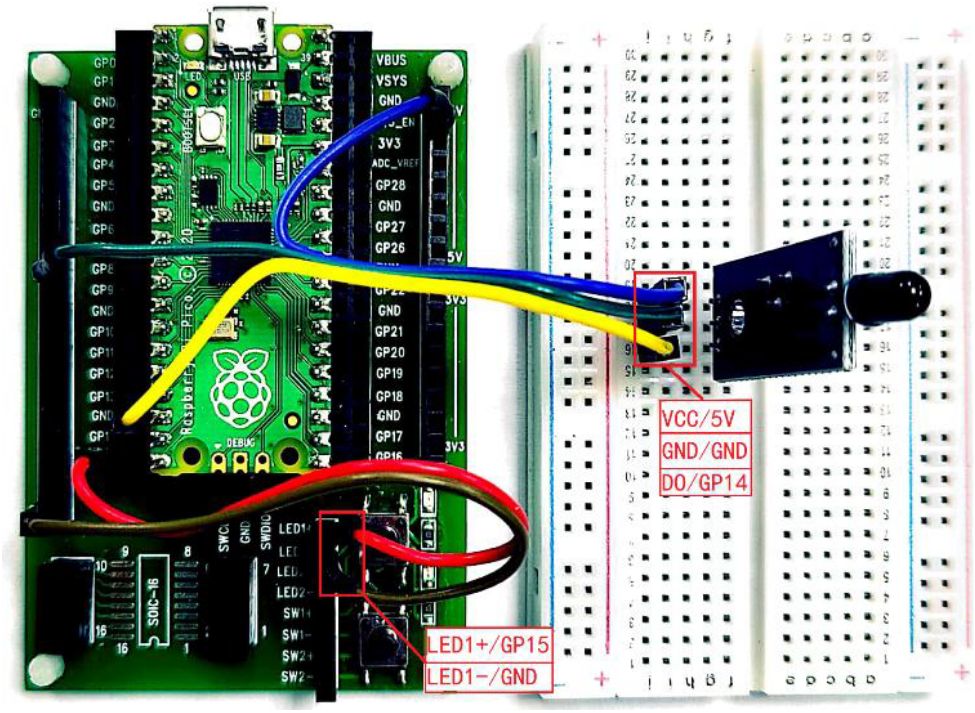


The program code is as follows:

```
Thonny - C:\Users\duncan380\Desktop\pico_0609\Infrared_Reflective_sensor_pico_code\main.py @...
File  Edit  View  Run  Tools  Help
main.py
2  import utime                                          #include delay time
3
4  led_external = machine.Pin(15, machine.Pin.OUT)       #define LED pin function:GP15,oup
5  aout = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)   #define AOUT pin function:GP14,in
6
7  flag = 0
8
9  while True:                                           #main loop
10     if aout.value() == 1:                                # check  AOUT pin == input high
11         led_external.value(1)                               # LED output high
12         if flag:
13             print('led on')
14             flag = 0
15         utime.sleep(0.2)                                 #delay 0.2sec
16     else:                                             # AOUT pin == input low
17         led_external.value(0)                               # LED output low
18         if flag == 0:
19             print('led off')
20             flag = 1
Shell
Unable to connect to COM4: port not found
Backend terminated or disconnected. Use 'Stop/Restart' to restart.
MicroPython (Raspberry Pi Pico)
```

4.14 Flame Sensor

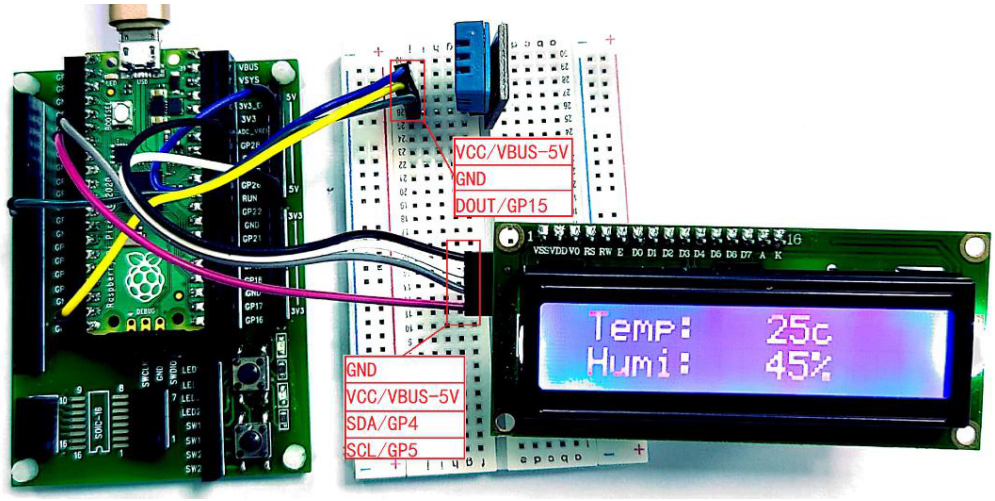
The wiring is as follows:



The program code is as follows:

4.15 Temperature-Humidity Sensor

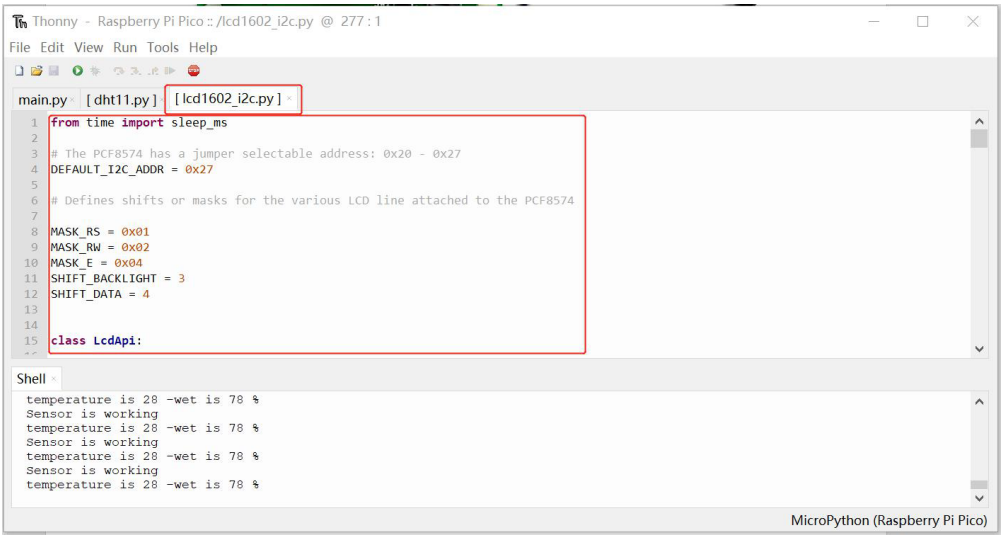
The wiring is as follows:

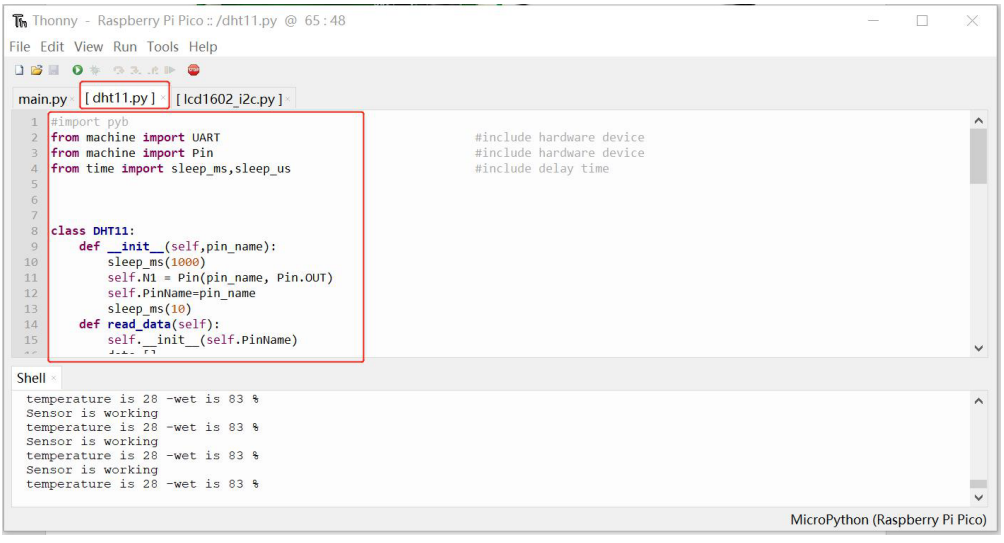


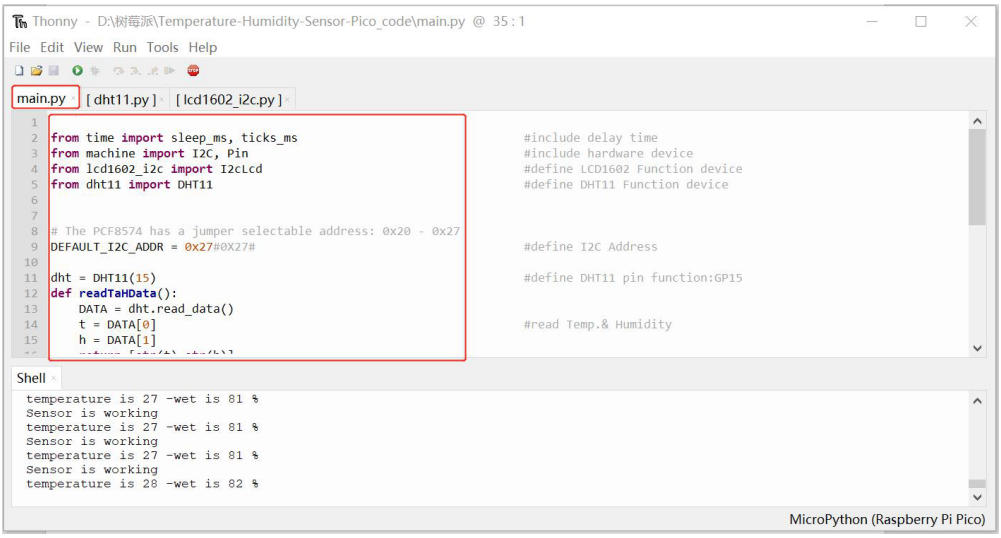
The program code is as follows:

Operation "main.py" First save "dht11.py" and "lcd1602_i2c.py"program to raspberry pi Pico, operation method refer to 3.3/step3.

Note: The file name must be consistent with the name of the program, the case letters and symbols must be consistent.





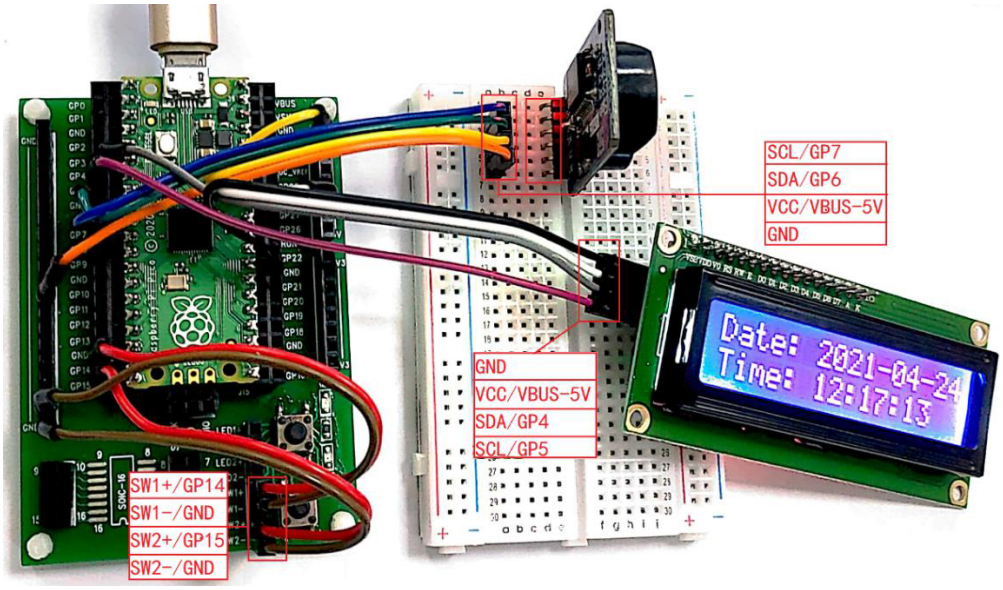


4.16 Clock sensor

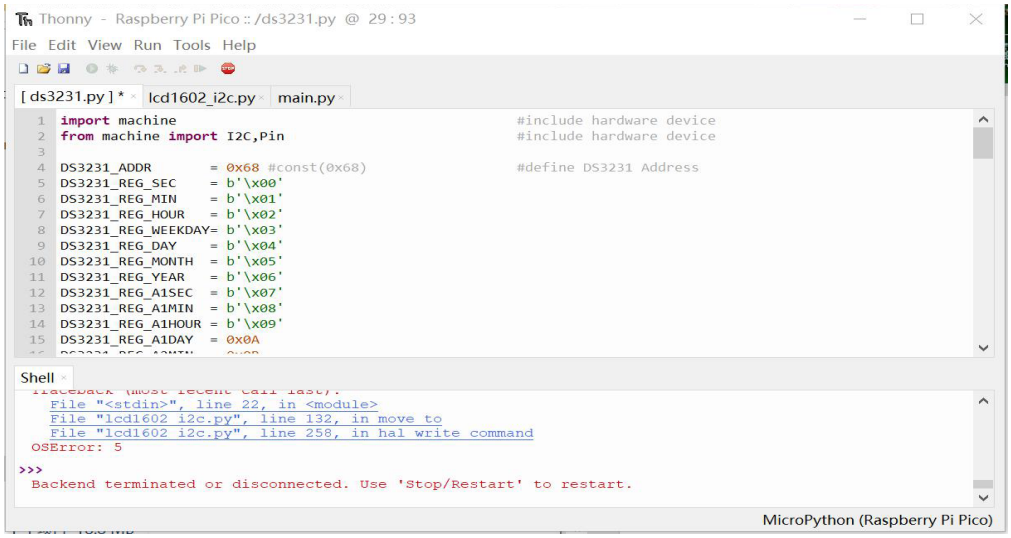
Operation “main.py” First save "DS3231.py " and "lcd1602_i2c.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

Note: The file name must be consistent with the name of the program, the case letters and symbols must be consistent.

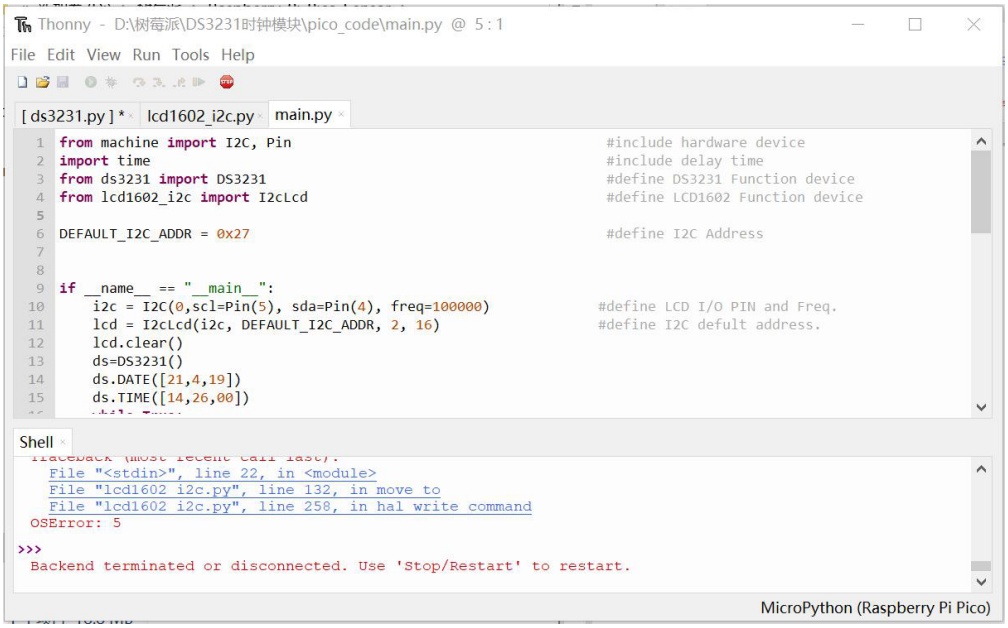
The wiring is as follows:



The program code is as follows:


Thonny - Raspberry Pi Pico :: /ds3231.py @ 29 : 93
File Edit View Run Tools Help
[ds3231.py] *
lcd1602_i2c.py
main.py
import machine #include hardware device
from machine import I2C,Pin #include hardware device
DS3231_ADDR = 0x68 #const(0x68) #define DS3231 Address
DS3231_REG_SEC = b'\x00'
DS3231_REG_MIN = b'\x01'
DS3231_REG_HOUR = b'\x02'
DS3231_REG_WEEKDAY= b'\x03'
DS3231_REG_DAY = b'\x04'
DS3231_REG_MONTH = b'\x05'
DS3231_REG_YEAR = b'\x06'
DS3231_REG_A1SEC = b'\x07'
DS3231_REG_A1MIN = b'\x08'
DS3231_REG_A1HOUR = b'\x09'
DS3231_REG_A1DAY = 0x0A
Shell
File "<stdin>", line 22, in <module>
File "lcd1602_i2c.py", line 132, in move_to
File "lcd1602_i2c.py", line 258, in hal_write_command
OSError: 5
>>>
Backend terminated or disconnected. Use 'Stop/Restart' to restart.
MicroPython (Raspberry Pi Pico)


Thonny - D:\树莓派\DS3231时钟模块\pico_code\lcd1602_i2c.py @ 277 : 1
File Edit View Run Tools Help
[ds3231.py] *
lcd1602_i2c.py
main.py
from time import sleep_ms
# The PCF8574 has a jumper selectable address: 0x20 - 0x27
DEFAULT_I2C_ADDR = 0x27#0x27#
# Defines shifts or masks for the various LCD line attached to the PCF8574
MASK_RS = 0x01
MASK_RW = 0x02
MASK_E = 0x04
SHIFT_BACKLIGHT = 3
SHIFT_DATA = 4
class LcdApi:
Shell
File "<stdin>", line 22, in <module>
File "lcd1602_i2c.py", line 132, in move_to
File "lcd1602_i2c.py", line 258, in hal_write_command
OSError: 5
>>>
Backend terminated or disconnected. Use 'Stop/Restart' to restart.
MicroPython (Raspberry Pi Pico)

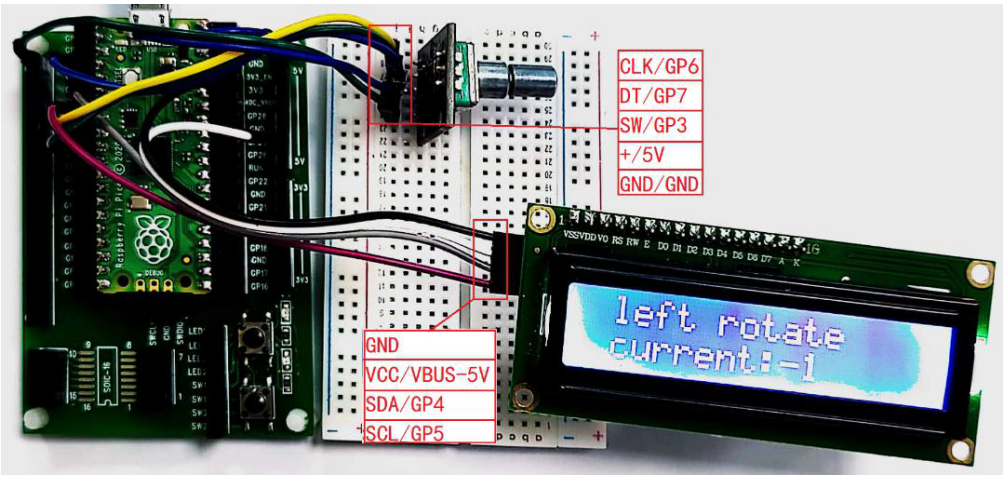

Thonny - D:\树莓派\DS3231时钟模块\pico_code\main.py @ 5 : 1
File Edit View Run Tools Help
[ds3231.py] *
lcd1602_i2c.py
main.py
from machine import I2C, Pin #include hardware device
import time #include delay time
from ds3231 import DS3231 #define DS3231 Function device
from lcd1602_i2c import I2cLcd #define LCD1602 Function device
DEFAULT_I2C_ADDR = 0x27 #define I2C Address
if __name__ == "__main__":
i2c = I2C(0,scl=Pin(5), sda=Pin(4), freq=100000) #define LCD I/O PIN and Freq.
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16) #define I2C defult address.
lcd.clear()
ds=DS3231()
ds.DATE([21,4,19])
ds.TIME([14,26,00])
Shell
File "<stdin>", line 22, in <module>
File "lcd1602_i2c.py", line 132, in move_to
File "lcd1602_i2c.py", line 258, in hal_write_command
OSError: 5
>>>
Backend terminated or disconnected. Use 'Stop/Restart' to restart.
MicroPython (Raspberry Pi Pico)

4.17 Rotation Sensor

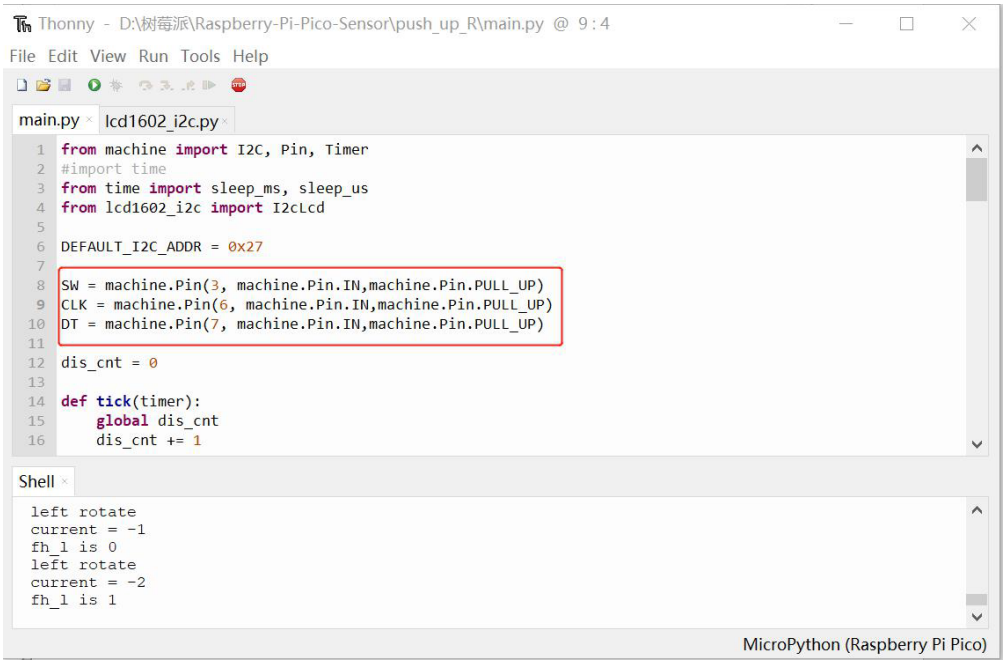
Operation “main.py” First save the "lcd1602_i2c.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

Note: The file name must be consistent with the name of the program, the case letters and symbols must be consistent.

The wiring is as follows:



The program code is as follows:

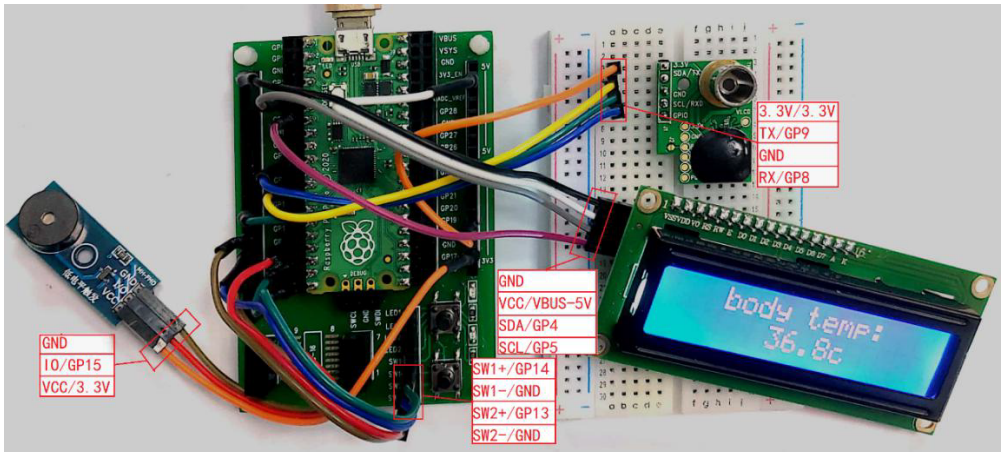


4.18 Temperature Sensor

Operation "main.py" First save the "lcd1602_i2c.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

Note: The file name must be consistent with the name of the program, the case letters and symbols must be consistent.

The wiring is as follows:



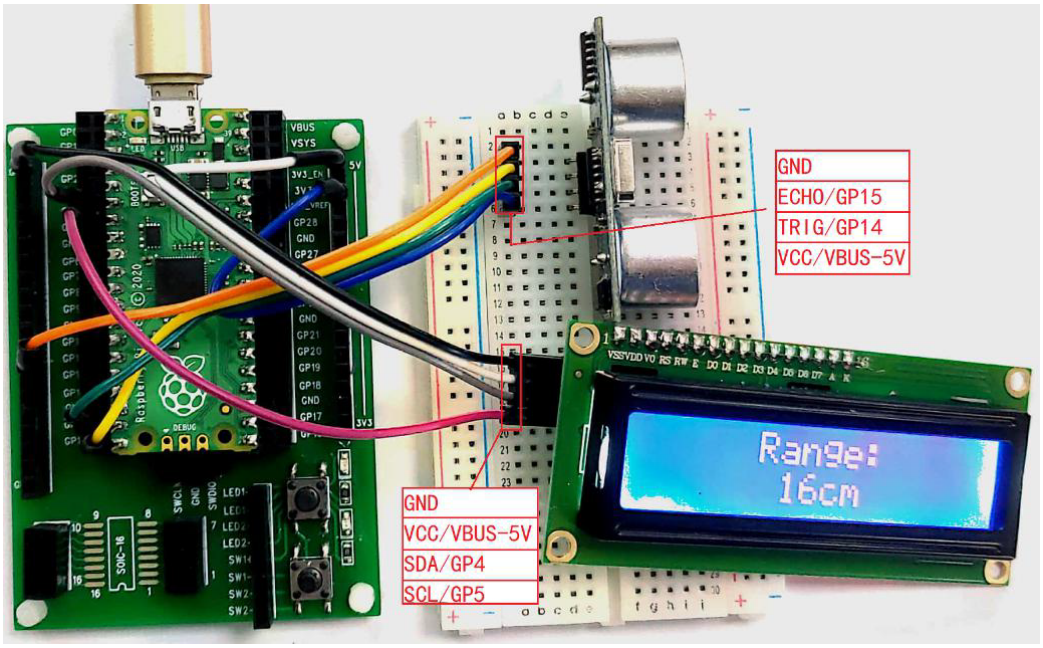
The program code is as follows:

4.19 Ultrasonic sensor

Operation “main.py” First save the "lcd1602_i2c.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

Note: The file name must be consistent with the name of the program, the case letters and symbols must be consistent.

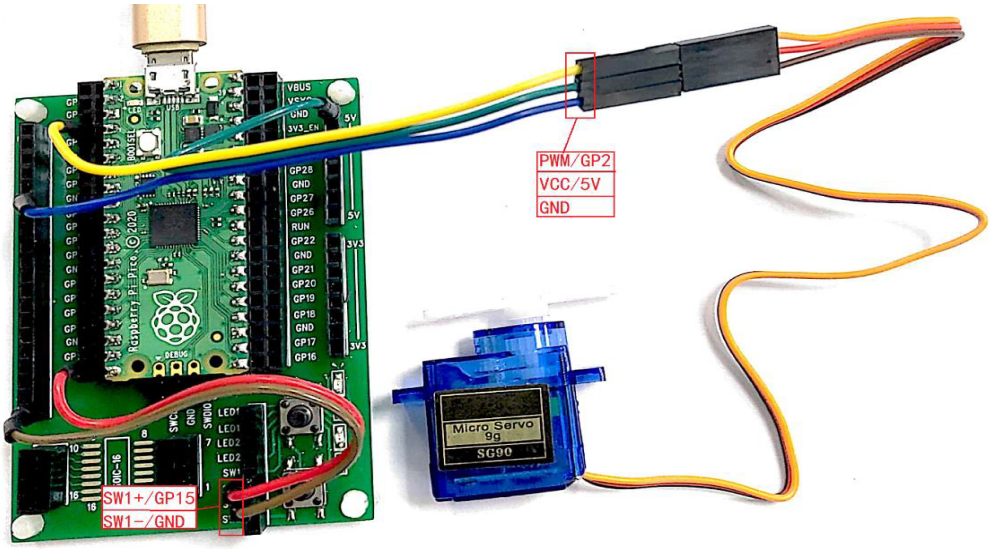
The wiring is as follows:



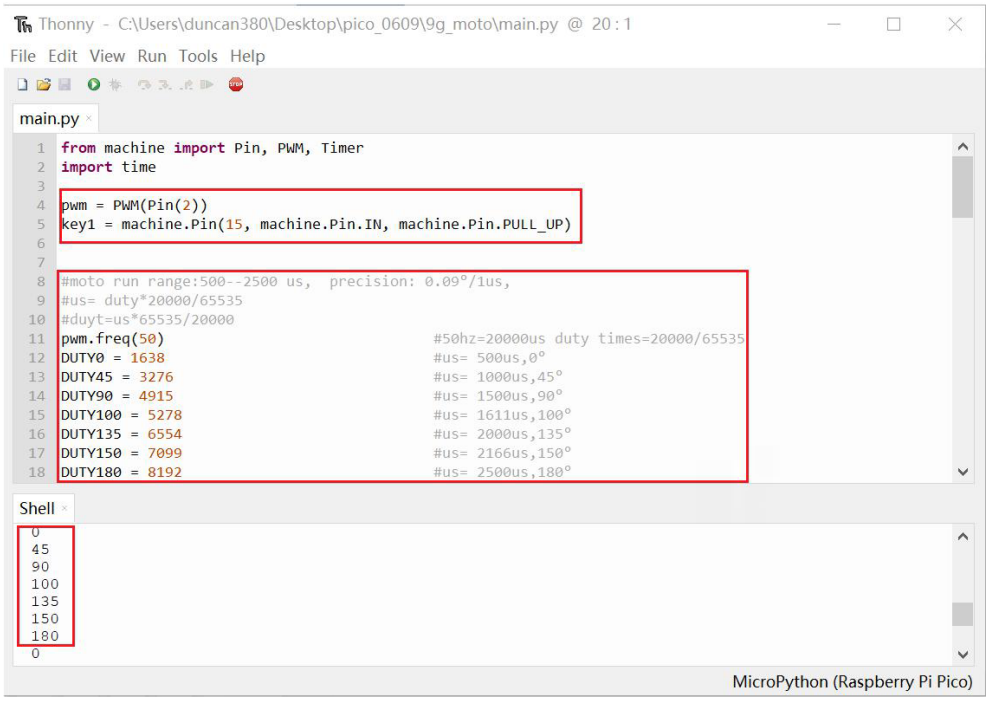
The program code is as follows:

4.20 Steering engine

The wiring is as follows:



The program code is as follows:

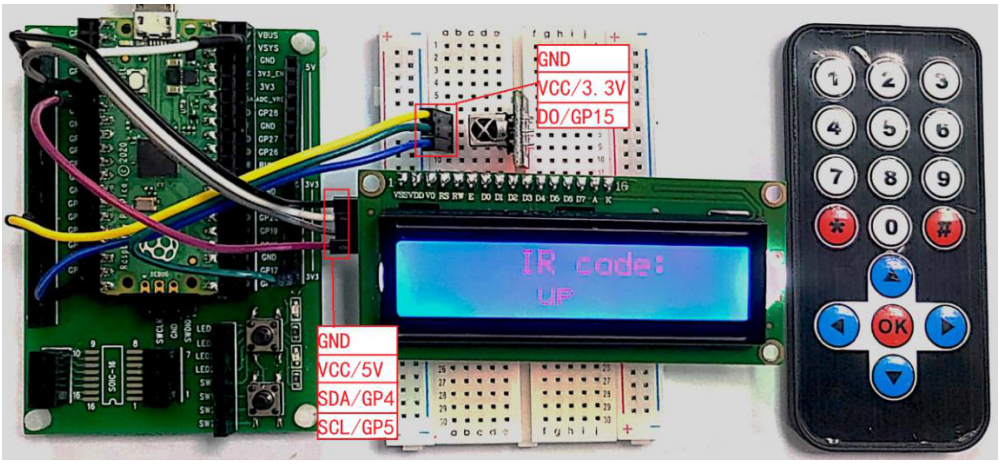


4.21 Infrared remote control

Operation “main.py” First save the "lcd1602_i2c.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

Note: The file name must be consistent with the name of the program, the case letters and symbols must be consistent.

The wiring is as follows:

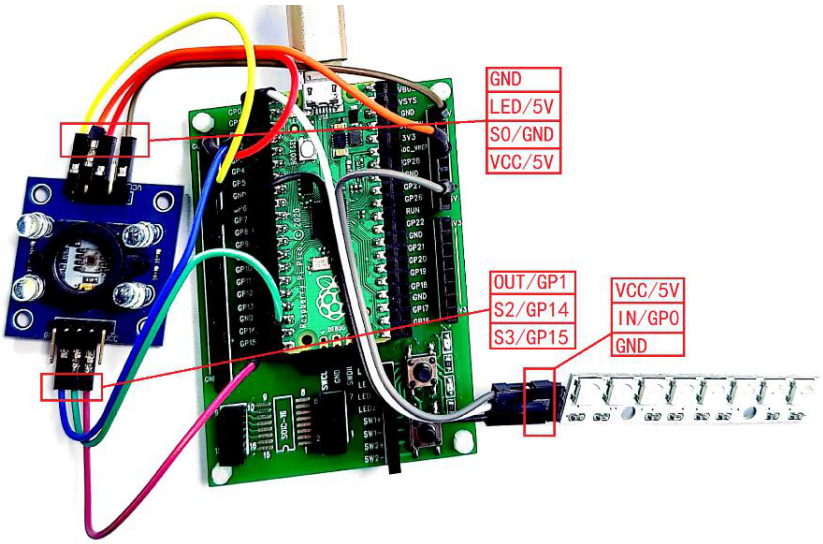


The program code is as follows:

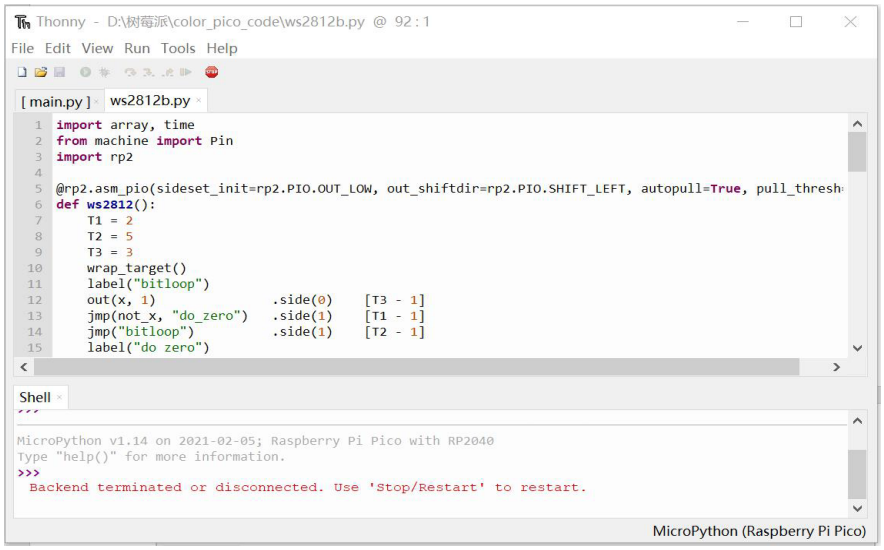
4.22 Color sensor

Operation "main.py" First save the "ws2812b.py "program to raspberry pi Pico, operation method refer to 3.3/step3.

The wiring is as follows:

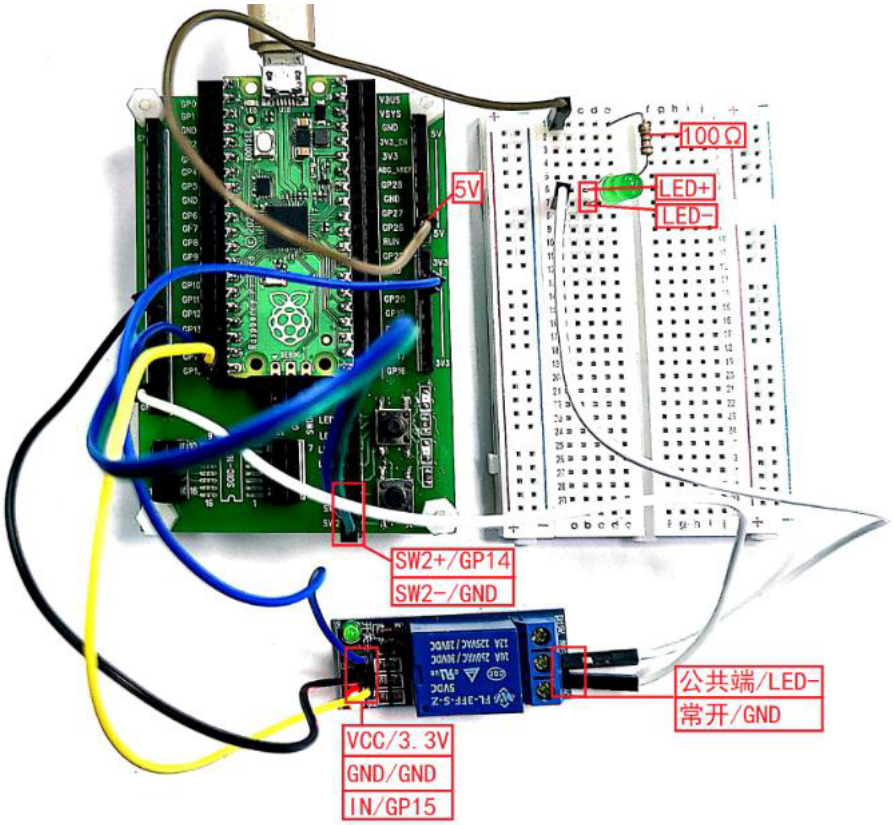


The program code is as follows:



4.23 Relay

The wiring is as follows:



The program code is as follows:

```
File  Edit  View  Run  Tools  Help
main.py
import machine                                          #include hardware devices
import utime                                            #include delay time

led_external = machine.Pin(15, machine.Pin.OUT)         #define LED pin function:GP15,oup
aout = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)   #define AOUT pin function:GP14,in

flag = 0

while True:                                           #main loop
    if aout.value() == 1:                               # check  AOUT pin == input high
        led_external.value(1)                             # LED output high
        if flag:
            print('led on')
            flag = 0
        utime.sleep(0.2)                                  #delay 0.2sec
    else:                                               # AOUT pin == input low
        led_external.value(0)                             # LED output low
        if flag == 0:
            print('led off')
Shell
led on
led off
led on
MicroPython (Raspberry Pi Pico)
```