Raspberry Pi Pico Starter Kit

--Micropython

Content

Packing List 1

Raspberry Pi Pico Introduction 2

- Specifications 3
- Pinout 4

Introduction of MicroPython 7

- Install Thonny IDE 7
- Burning Micropython Firmware (important) 14
- Thonny Connected to Raspberry Pi Pico(Important) 18

Your First MicroPython Program 24

- Interactive Mode 24
- Script Mode 25

Project 1 Hello, LED!(important) 31

- Open Code File 31
- Thonny Connected to Raspberry Pi Pico 34
- Run Code 36
- How it works? 37
- What More? 38

Project 2 Button Control LED 39

- Component knowledge 39
- Wiring 40
- Code 40

Project 3 PWM Control LED 43

- PWM signal 43
- Wiring 45
- Code 46
- How it works? 47

Project 4 RGB LED 48

- Wiring 48
- Code 49

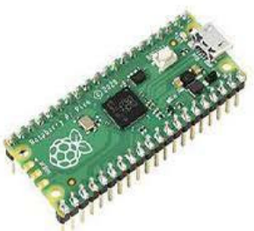
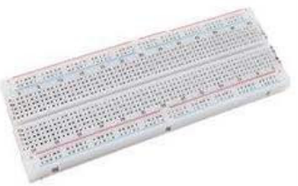
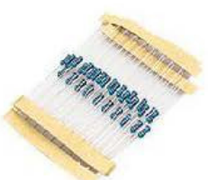
Project 5 Custom Tone 52

Component Knowledge 52
Wiring 53
Code 54
How it works? 55
Project 6 Analog Input 56
Related knowledge 56
Wiring 59
Code 59
How it works? 61
Project 7 Potentiometer Control Servo 62
Wiring 63
Code 64
Project 8 Photoresistor Control LED 66
Wiring 67
Code 67
Project 9 Thermeometer 70
Wiring 70
Code 71
How it works? 72
Project 10 LCD1602 Show Temperature 73
Wiring 73
Upload LCD1602.py to Raspberry Pi Pico(Important) 74
Code 80
Project 11 Intruder Alarm 82
Wiring 83
Code 84
Project 12 RGB LED Strip 86
Wiring 87
Upload ws2812.py to Raspberry Pi Pico(important) 87
Code 93
How it works? 95

Components Introduction 96
- Breadboard 97
- LED 99
- Button 100
- RGB LED 101
- Resistor 103
- Transistor 105
- Buzzer 107
- Potentiometer 109
- IR Proximity Sensor Module 110
- Photoresistor 112
- Thermistor 114
- Tilt Switch 116
- Servo 117
- I2C LCD1602 118
- PIR Motion Sensor 120
- WS2812 RGB 8 LEDs Strip 123

FAQ 125

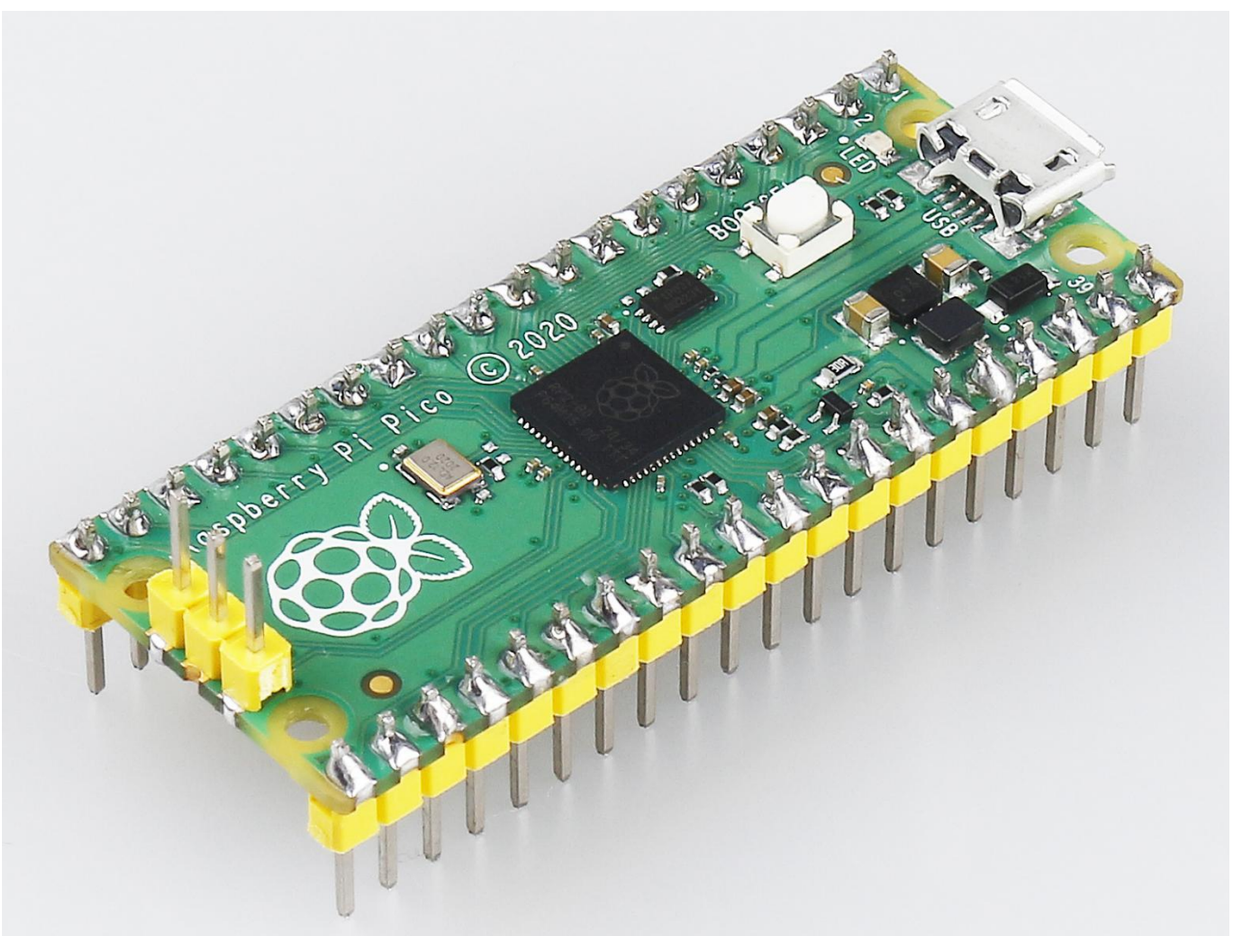
Packing List

 Raspberry Pi Pico x1	 IIC LCD 1602 x1	 830 Tie-Points Breadboard x1	 Obstacle Avoidance Module x1
 HC-SR501 PIR Motion Sensor x1	 SG90 Servo x1	 Potentiometer (10k) x1	 Passive Buzzer x1 Active Buzzer x1
 Resistor-220R/1k/10k x30	 Micro-USB Cable x1	 Button Switch x8	 WS2812 RGB 8 LEDSs Strip x1
 F-M DuPont Cable x10	 65 Jumper Wire x1	 Thermistor x1 Tilt Switch x1	 Photoresistor x2 S8050 Transistor x2
 LED-RGB x5	 LED-Yellow x5	 LED-Green x5	 LED-RGB x1

Raspberry Pi Pico Introduction

The Raspberry Pi Pico is a microcontroller board based on the Raspberry Pi RP2040 microcontroller chip.

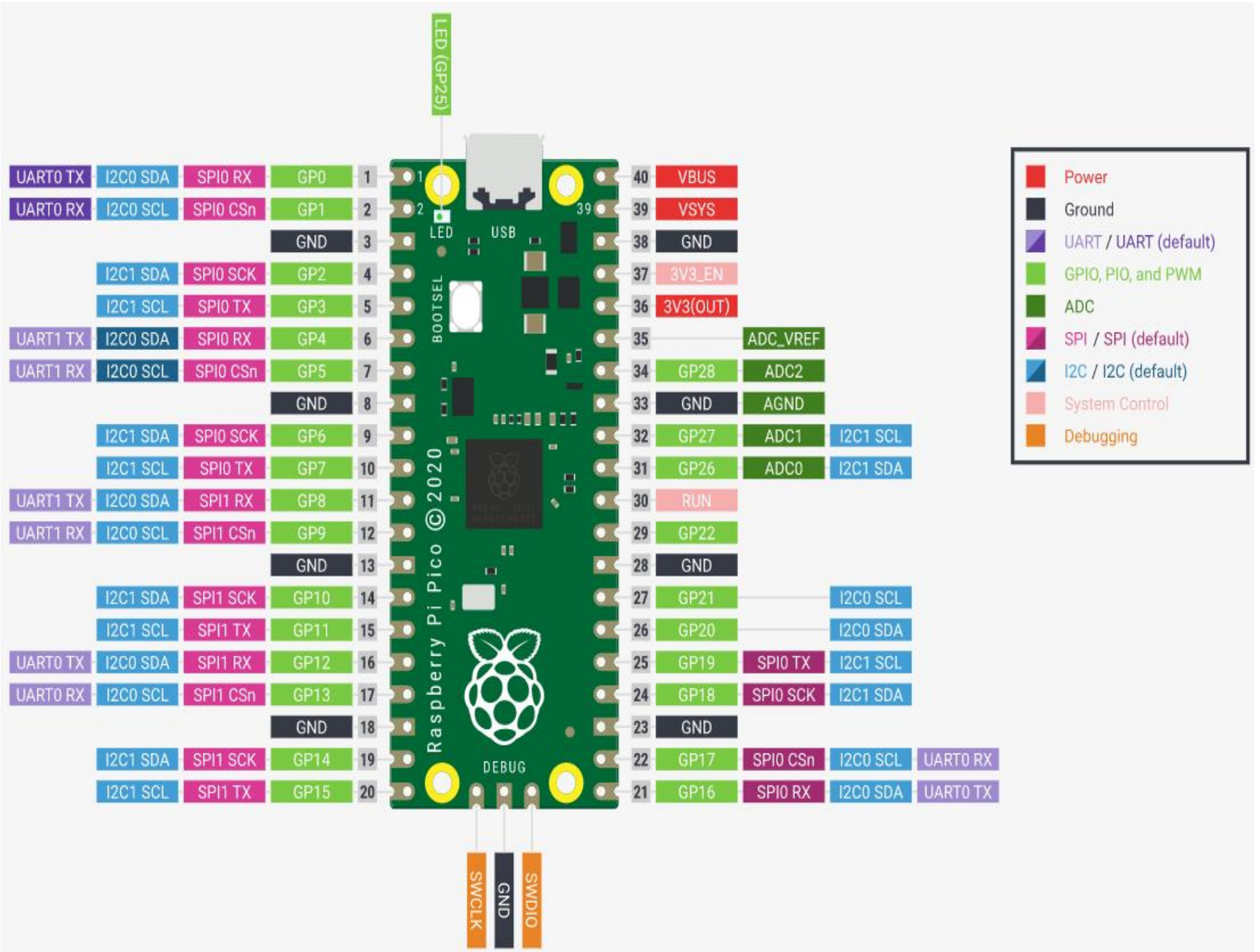
Whether you want to learn the MicroPython programming language, take the first step in physical computing, or want to build a hardware project, Raspberry Pi Pico and its amazing community – will support you every step of the way. In the project, it can control anything, from LEDs and buttons to sensors, motors, and even other microcontrollers.



Specifications

- **21 mm × 51 mm form factor**
- **RP2040 microcontroller chip designed by Raspberry Pi in the UK**
- **Dual-core Arm Cortex-M0+ processor, flexible clock running up to 133 MHz**
- **264KB on-chip SRAM**
- **2MB on-board QSPI Flash**
- **26 multifunction GPIO pins, including 3 analog inputs**
- **2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels**
- **1 × USB 1.1 controller and PHY, with host and device support**
- **8 × Programmable I/O (PIO) state machines for custom peripheral support**
- **Supported input power 1.8–5.5V DC**
- **Operating temperature -20°C to +85°C**
- **Castellated module allows soldering direct to carrier boards**
- **Drag-and-drop programming using mass storage over USB**
- **Low-power sleep and dormant modes**
- **Accurate on-chip clock**
- **Temperature sensor**
- **Accelerated integer and floating-point libraries on-chip**

Pinout


LED (GP25)
Raspberry Pi Pico ©2020
LED
USB
BOOTSEL
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI / SPI (default)
I2C / I2C (default)
System Control
Debugging
1 UART0 TX I2C0 SDA SPI0 RX GP0
2 UART0 RX I2C0 SCL SPI0 CSn GP1
3 GND
4 I2C1 SDA SPI0 SCK GP2
5 I2C1 SCL SPI0 TX GP3
6 UART1 TX I2C0 SDA SPI0 RX GP4
7 UART1 RX I2C0 SCL SPI0 CSn GP5
8 GND
9 I2C1 SDA SPI0 SCK GP6
10 I2C1 SCL SPI0 TX GP7
11 UART1 TX I2C0 SDA SPI1 RX GP8
12 UART1 RX I2C0 SCL SPI1 CSn GP9
13 GND
14 I2C1 SDA SPI1 SCK GP10
15 I2C1 SCL SPI1 TX GP11
16 UART0 TX I2C0 SDA SPI1 RX GP12
17 UART0 RX I2C0 SCL SPI1 CSn GP13
18 GND
19 I2C1 SDA SPI1 SCK GP14
20 I2C1 SCL SPI1 TX GP15
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX

Name	Description	Function
GP0-GP28	General-purpose input/output pins	Act as either input or output and have no fixed purpose of their own
GND	0 volts ground	Several GND pins around Pico to make wiring easier.
RUN	Enables or diables your Pico	Start and stop your Pico from another microcontroller.
GPxx_ADCx	General-purpose input/output or analog input	Used as an analog input as well as a digital input or output – but not both at the same time.
ADC_VREF	Analog-to-digital converter (ADC) voltage reference	A special input pin which sets a reference voltage for any analog inputs.
AGND	Analog-to-digital converter (ADC) 0 volts ground	A special ground connection for use with the ADC_VREF pin.
3V3(O)	3.3 volts power	A source of 3.3V power, the same voltage your Pico runs at internally, generated from the VSYS input.
3v3(E)	Enables or disables the power	Switch on or off the 3V3(O) power, can also switches your Pico off.
VSYS	2-5 volts power	A pin directly connected to your Pico's internal power supply, which cannot be switched off without also switching Pico off.
VBUS	5 volts power	source of 5 V power taken from your Pico's micro USB port, and used to power hardware which needs more than 3.3 V.

The best place to find everything you need to get started with your Raspberry Pi Pico.

Or you can click on the links below:

- Raspberry Pi Pico product brief
- Raspberry Pi Pico datasheet
- Getting started with Raspberry Pi Pico: C/C++ development
- Raspberry Pi Pico C/C++ SDK
- API-level Doxygen documentation for the Raspberry Pi Pico C/C++ SDK
- Raspberry Pi Pico Python SDK
- Raspberry Pi RP2040 datasheet
- Hardware design with RP2040
- Raspberry Pi Pico design files
- Raspberry Pi Pico STEP file

Introduction of MicroPython

MicroPython is a software implementation of a programming language largely compatible with Python 3, written in C, that is optimized to run on a microcontroller.

MicroPython consists of a Python compiler to bytecode and a runtime interpreter of that bytecode. The user is presented with an interactive prompt (the REPL) to execute supported commands immediately. Included are a selection of core Python libraries; MicroPython includes modules which give the programmer access to low-level hardware.

Reference: MicroPython - Wikipedia

Why MicroPython?

While Python has the same advantages, for some Microcontroller boards like the Raspberry Pi Pico, they are small, simple and have little memory to run the Python language at all. That's why MicroPython has evolved, keeping the main Python features and adding a bunch of new ones to work with these Microcontroller boards.

Next you will learn to install MicroPython into the Raspberry Pi Pico.

Install Thonny IDE

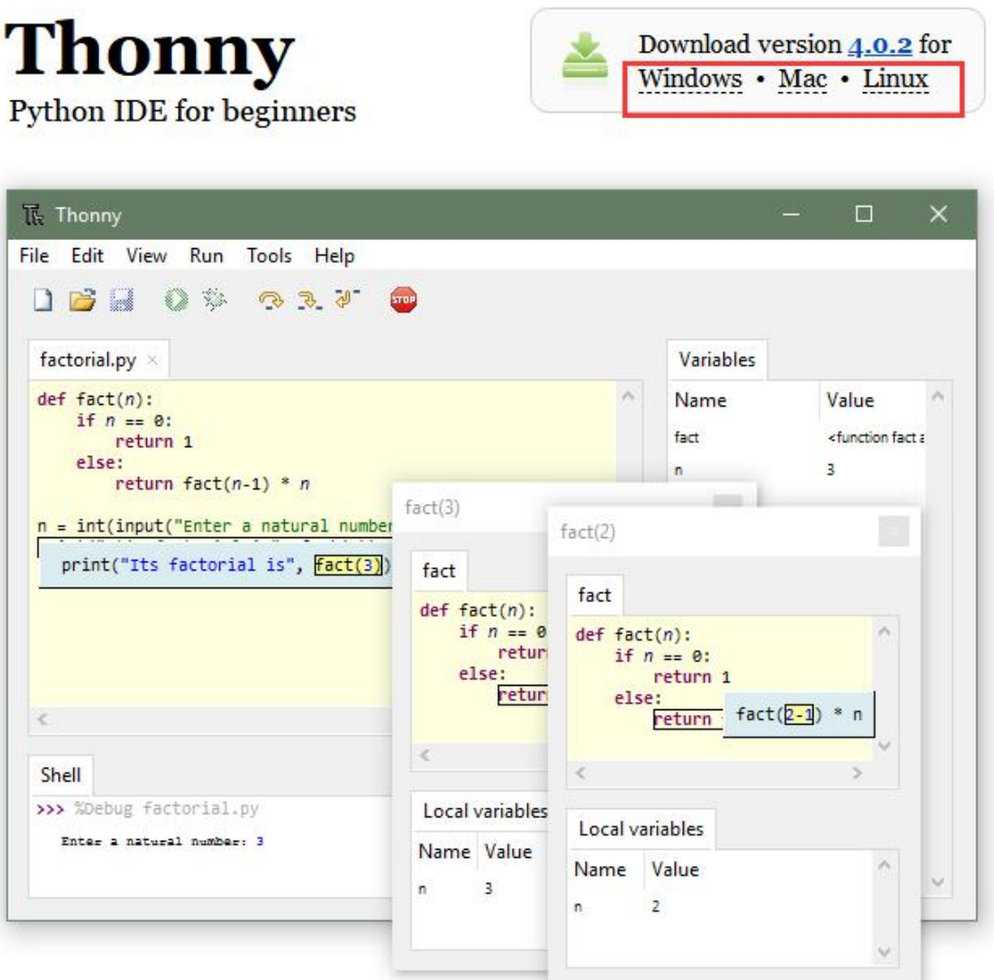
Before you can start to program Pico with MicroPython, you need an integrated development environment (IDE), here we recommend Thonny. Thonny comes with Python 3.7 built in, just one simple installer is needed and you're ready to learn programming.

> **Note**
>
> Since the Raspberry Pi Pico interpreter only works with Thonny version 3.3.3 or later, you can skip this chapter if you have it; otherwise, please update or install it.It is recommended to use the latest version.

Downloading Thonny

You can download it by visiting the Thonny website. Once open the page, you will see a light gray box in the upper right corner, click on the link that applies to your operating system.

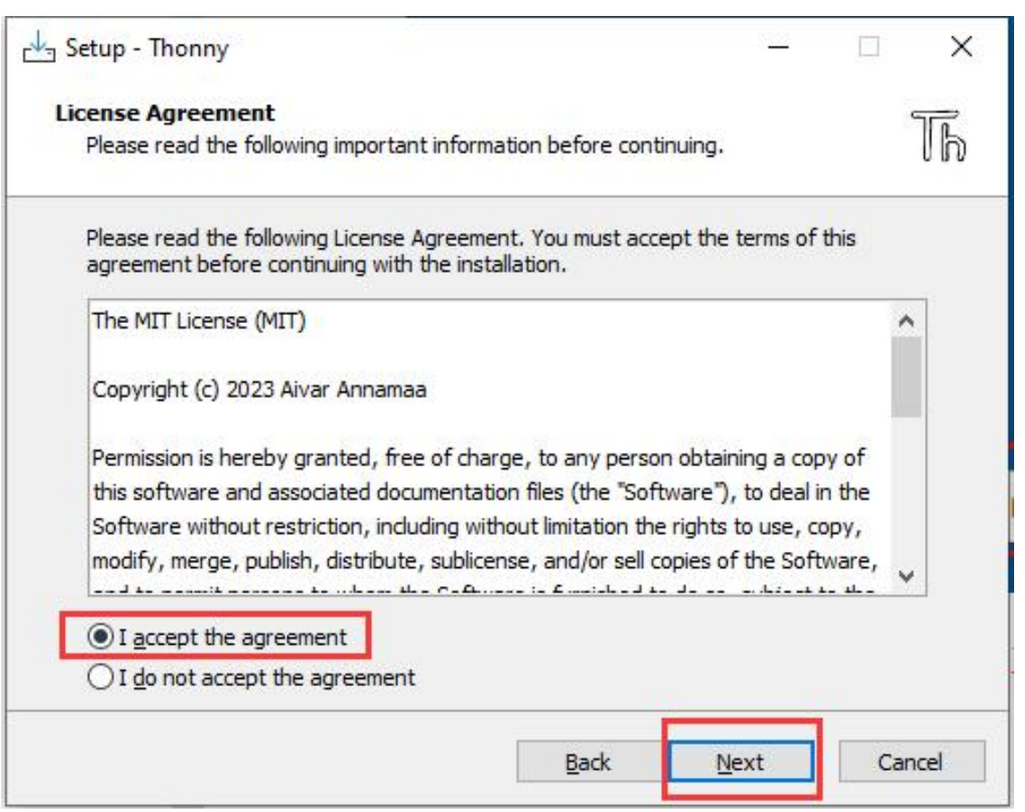
Official website of Thonny: https://thonny.org



Installing on Windows

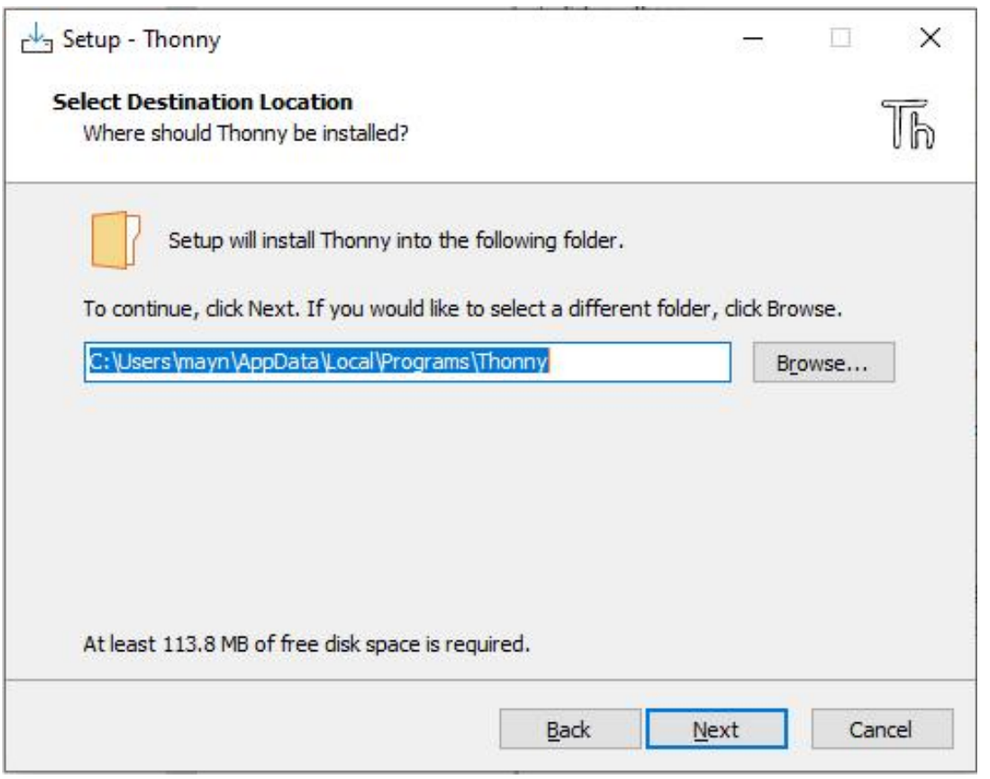
The icon of Thonny after downloading is as below. Double click “thonny-4.0.2.exe”.

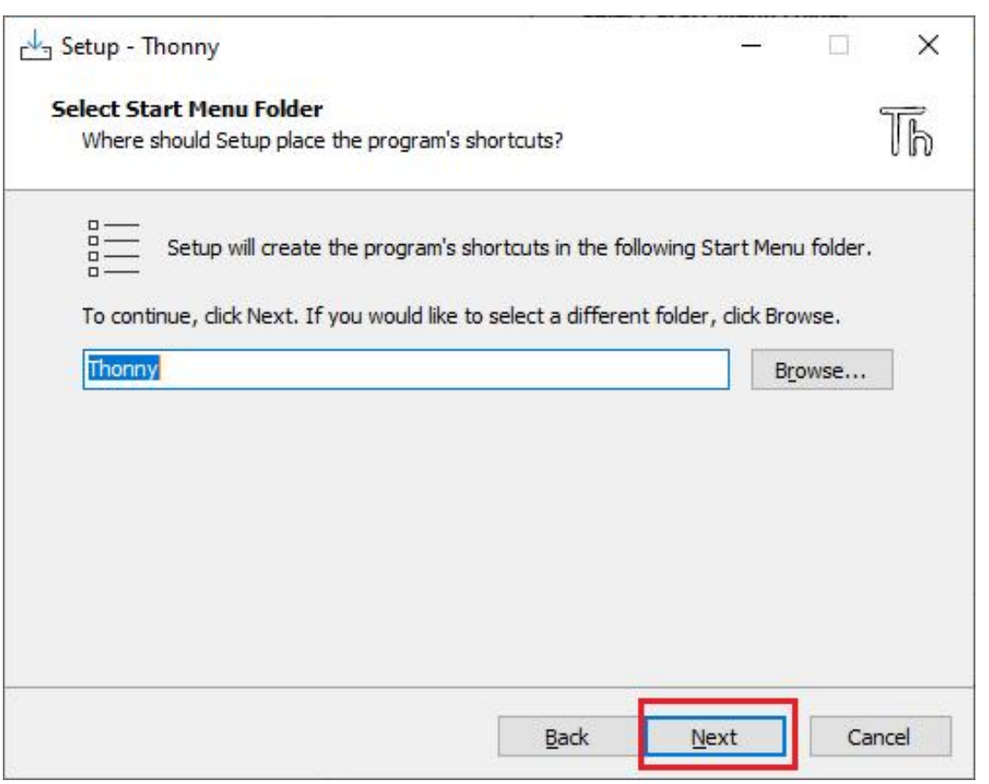
If you’re not familiar with computer software installation, you can simply keep clicking “Next” until the installation completes.



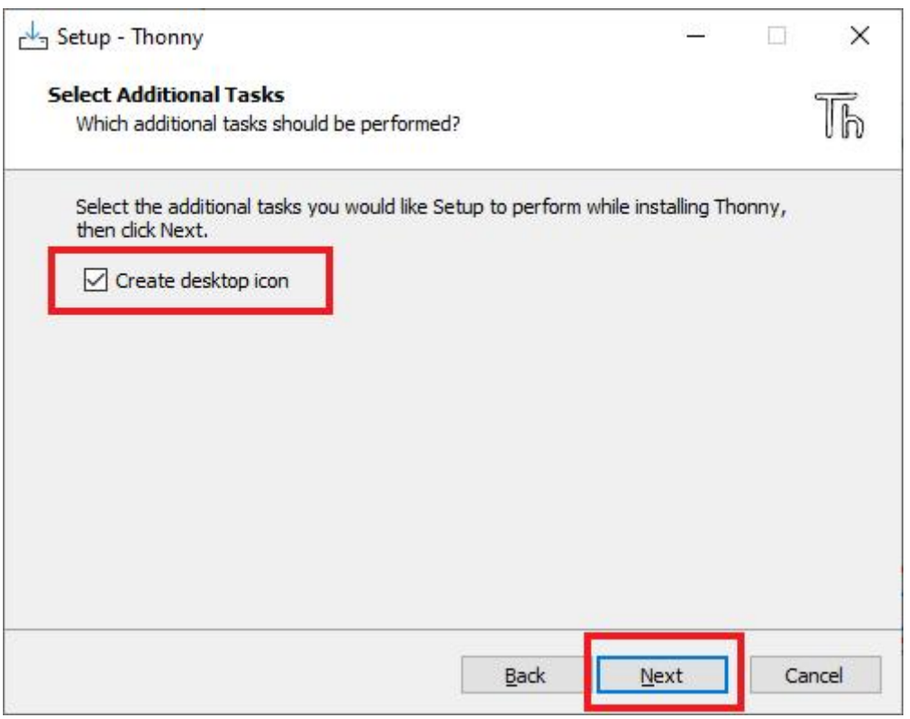
If you want to change Thonny's installation path, you can click "Browse" to modify it. After selecting installation path, click "OK".

If you do not want to change it, just click "Next".

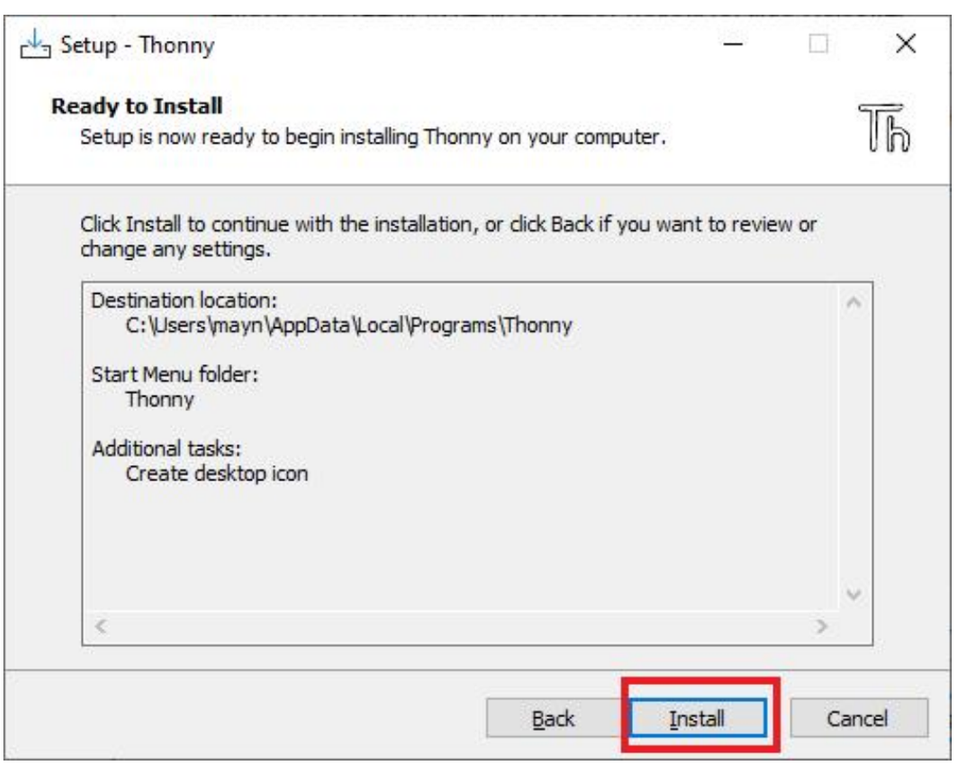




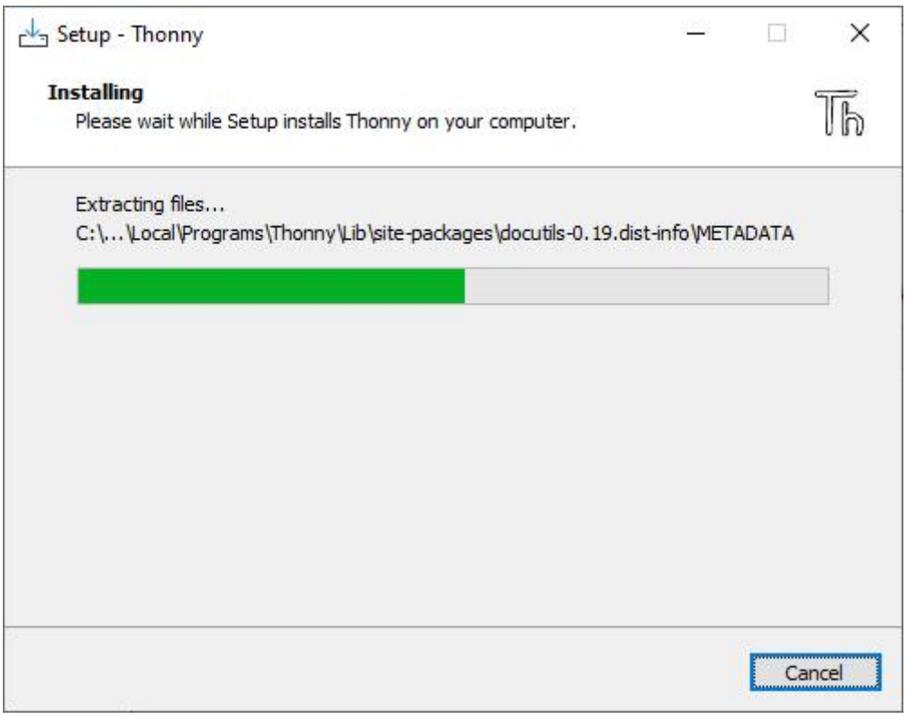
Check “Create desktop icon” and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



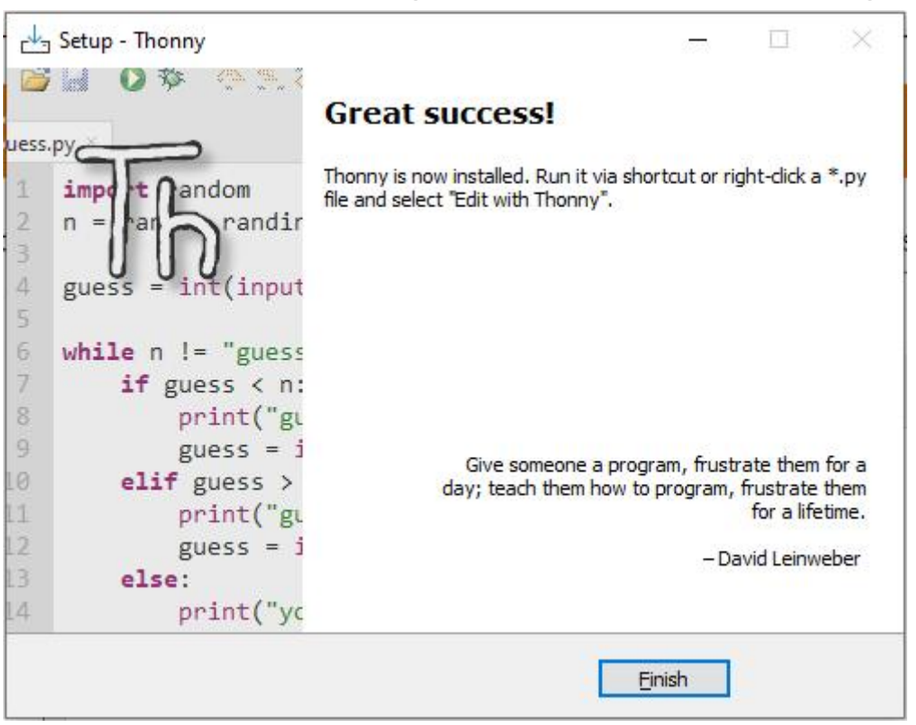
Click “install” to install the software



During the installation process, you only need to wait for the installation to complete, and you msut not click"Cancel", otherwise Thonny will fail to be installed.



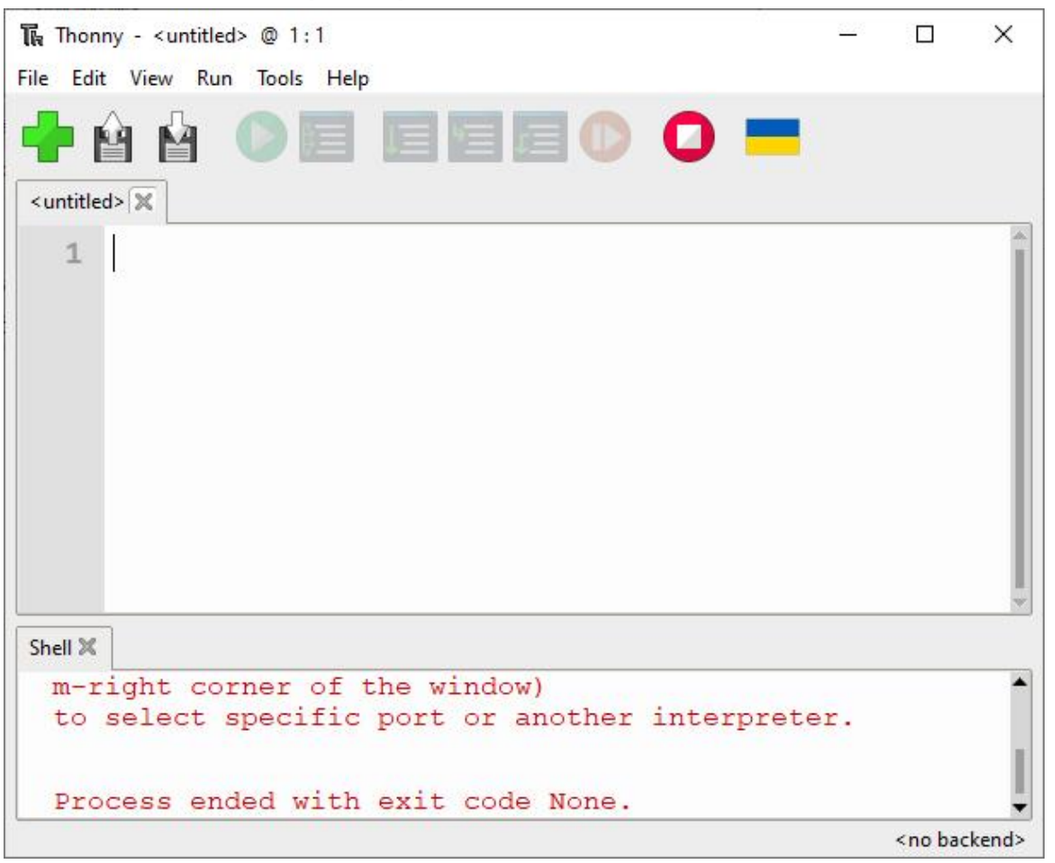
Once you see the interface as below, Thonny has been installed successfully.

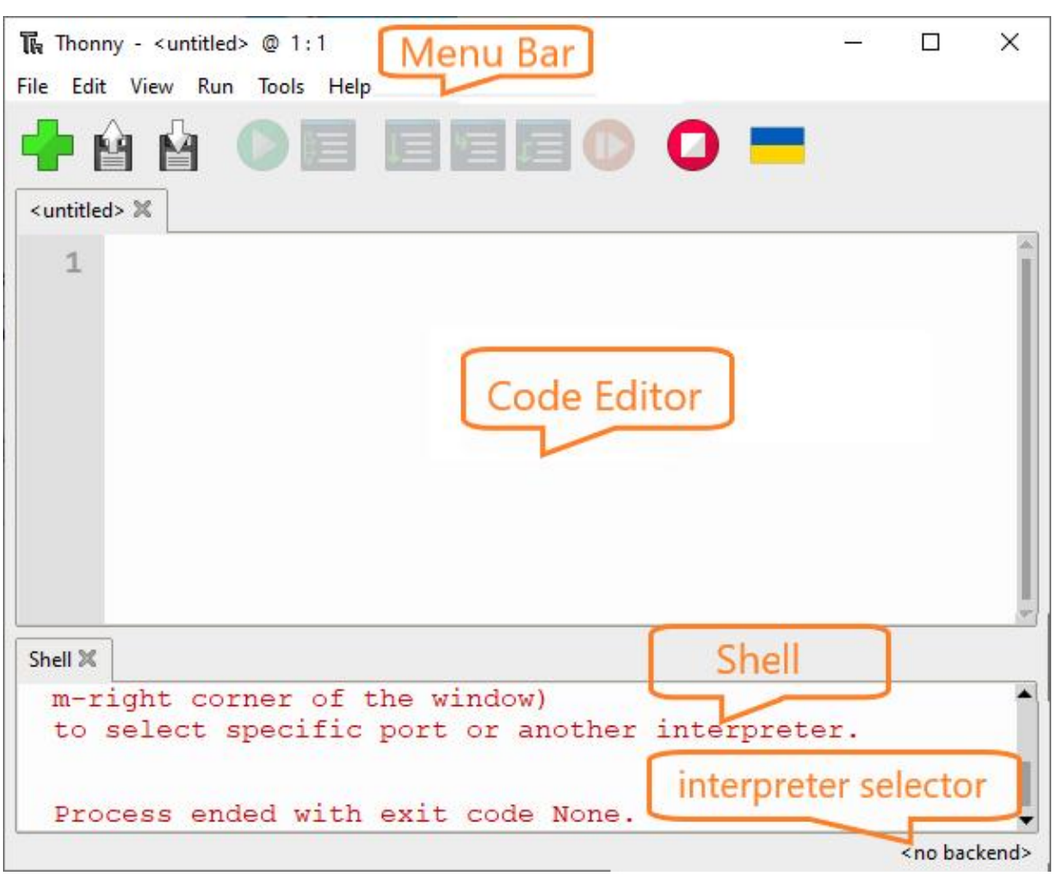


If you've check "Create desktop icon" during the installation process, you can see the below icon on your desktop.

Basic Configuration of Thonny

Click the desktop icon of Thonny and you can see the interface of it as follows:





Burning Micropython Firmware (important)

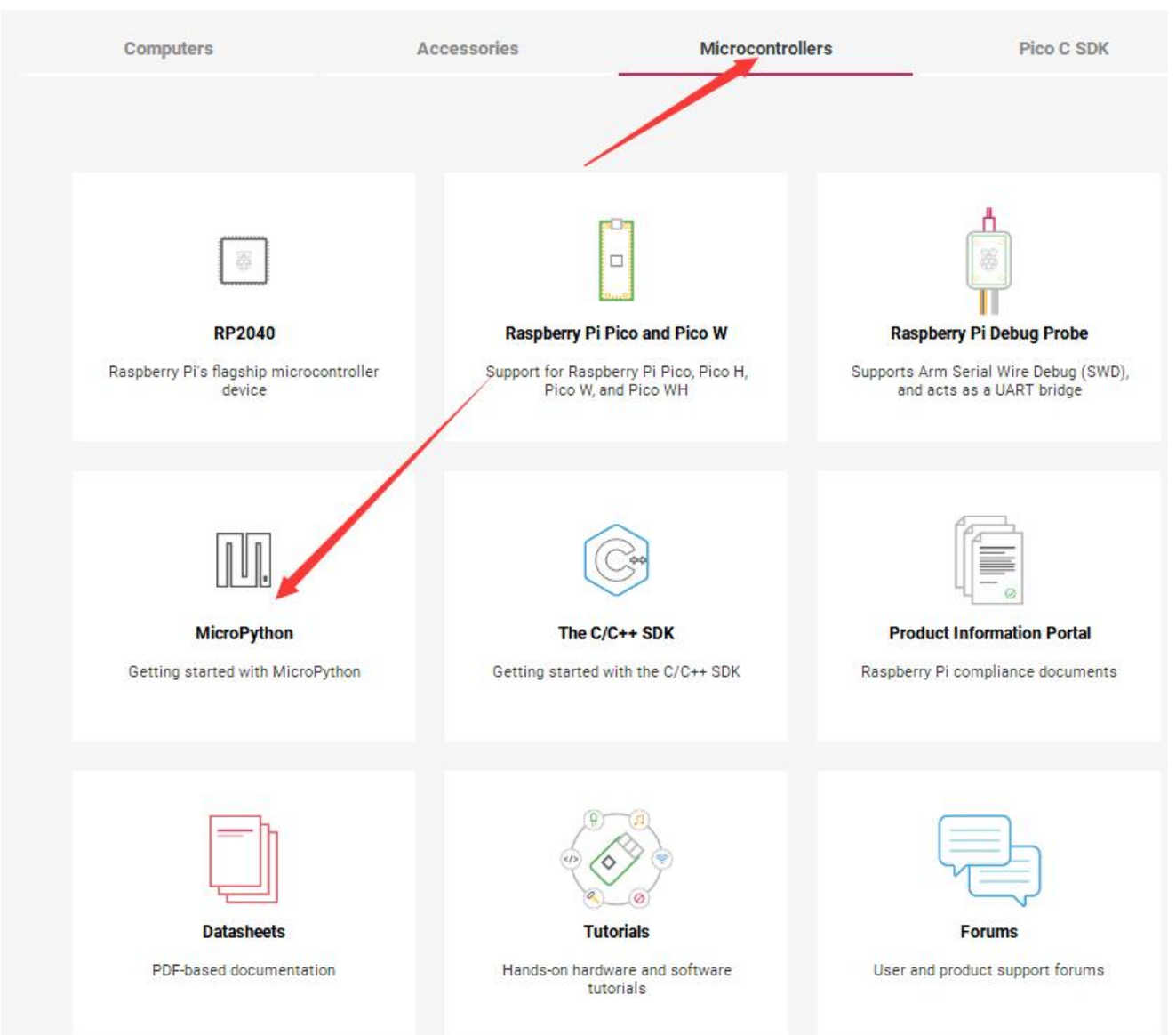
To run Python programs on Raspberry Pi Pico, we need to burn a firmware to Raspberry Pi Pico first.

Downloading Micropython Firmware

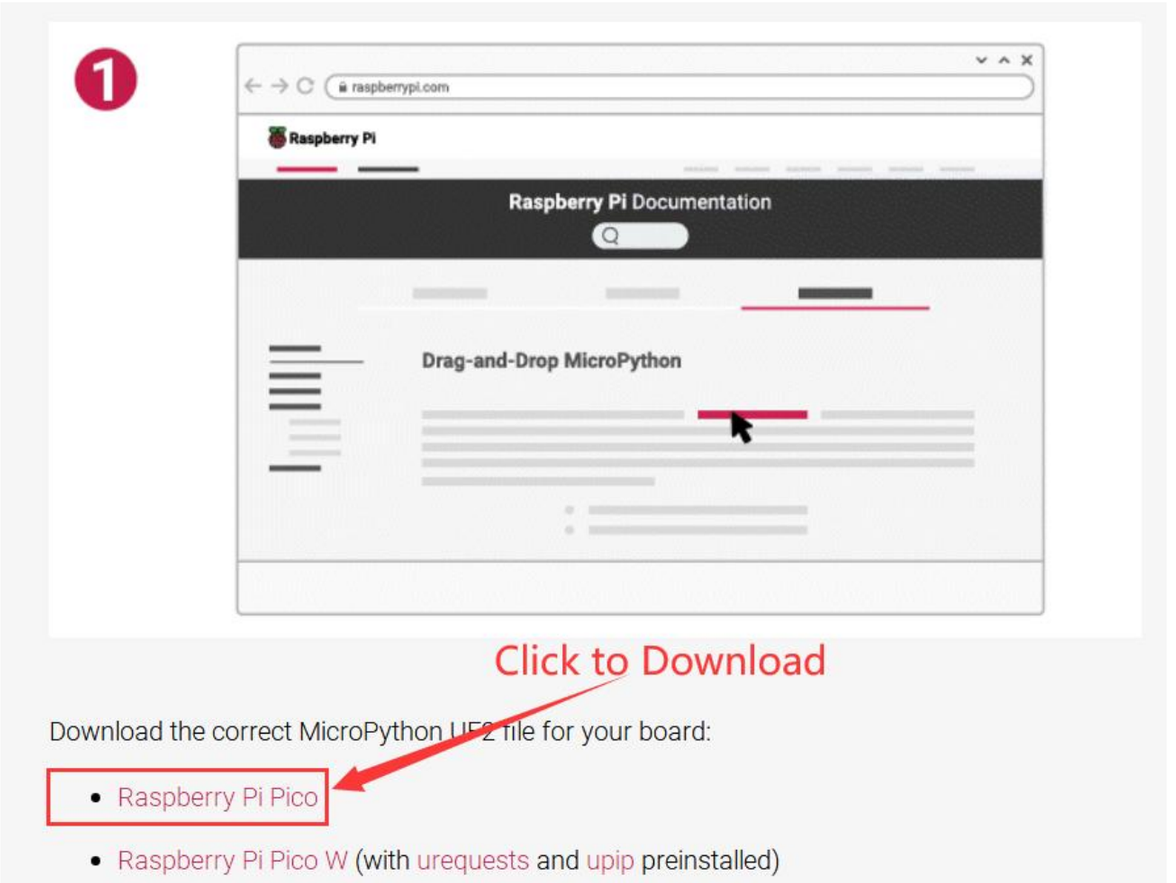
Raspberry Pi Pico official website:

https://www.raspberrypi.com/documentation/microcontrollers/

Click Micropython.

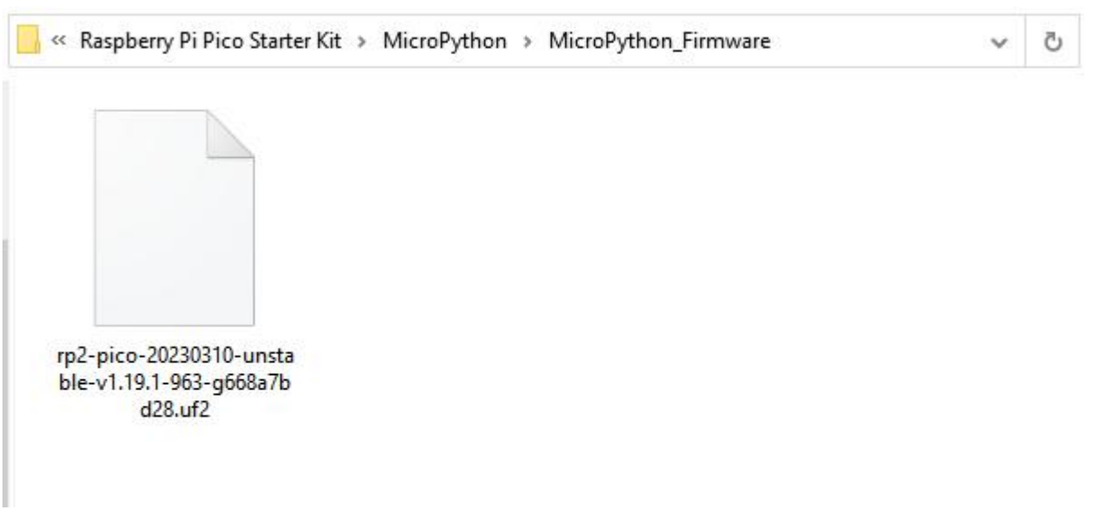


Click Raspberry Pi Pico to download UF2 file.



If you cannot download it due to network issue or other reasons, you can use the one we have prepared,which locates at the following file path:

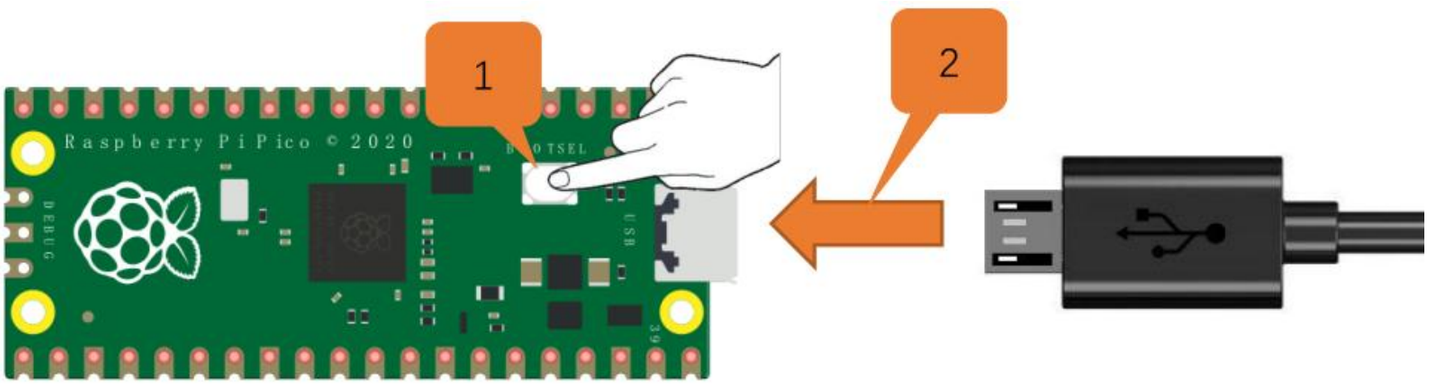
Raspberry Pi Pico Starter Kit/MicroPython/MicroPython_Firmware/rp2-pico-20230310-unstable-v1.19.1-963-g668a7bd28.uf2



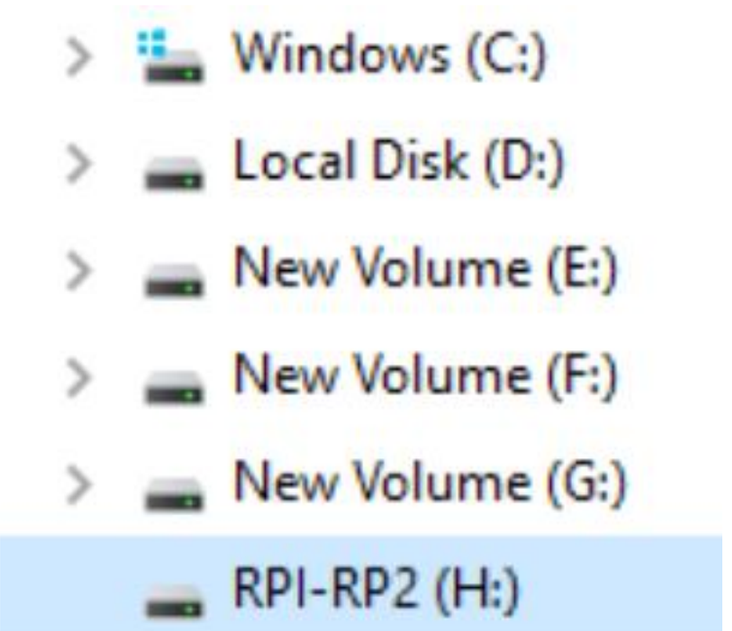
Burning a Micropython Firmware

1. Connect a USB cable to your computer.

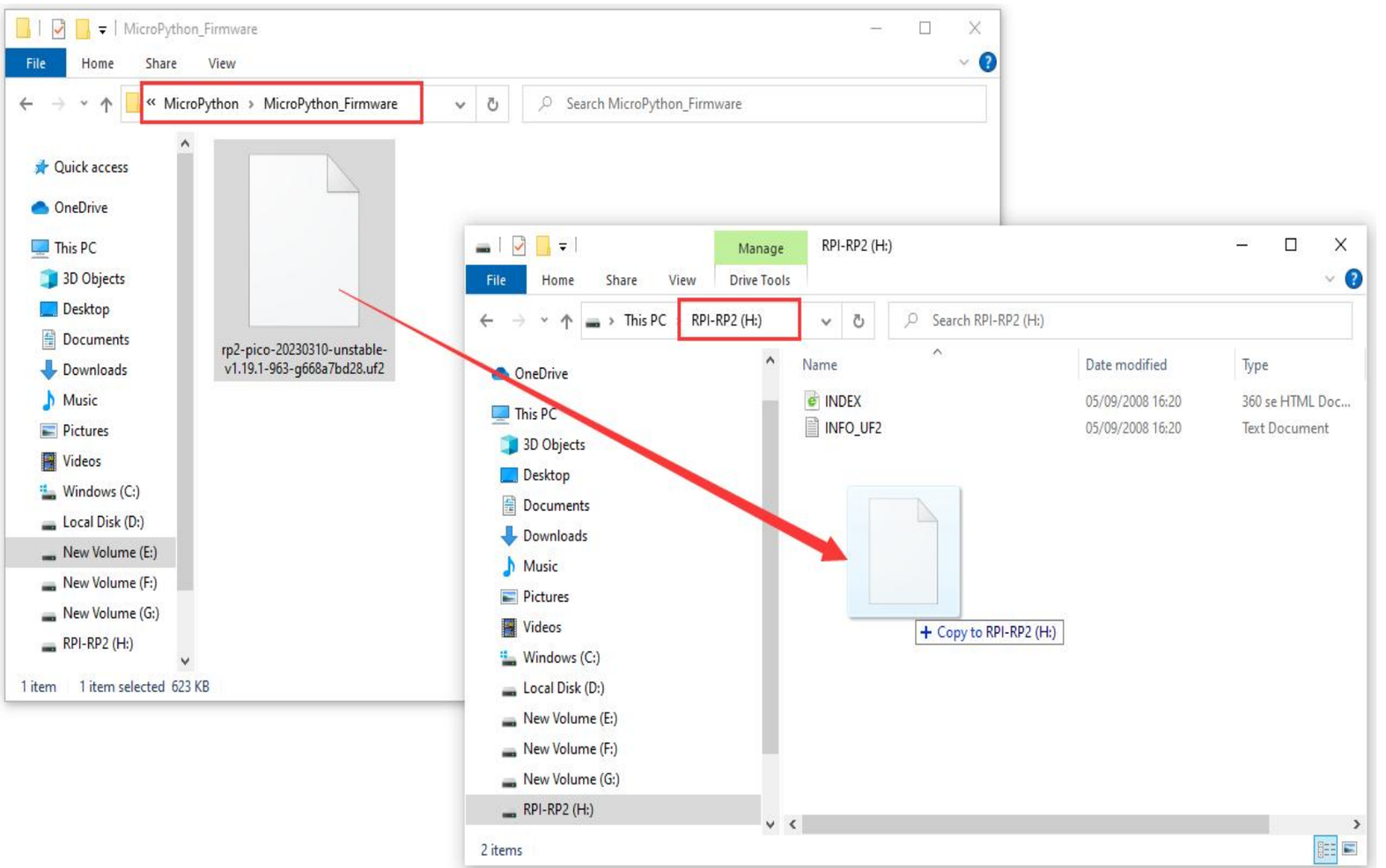
2. Long press BOOTSEL button on Raspberry Pi Pico and connect it to your computer with the USB cable.



3. When the connection succeeds, A new disk called **RPI-RP2(:)** will appear in the file manager.



4. Copy the file(rp2-pico-20230310-unstable-v1.19.1-963-g668a7bd28.uf2) to RPI-RP2 and wait for it to finish, just like copy file to a U disk.

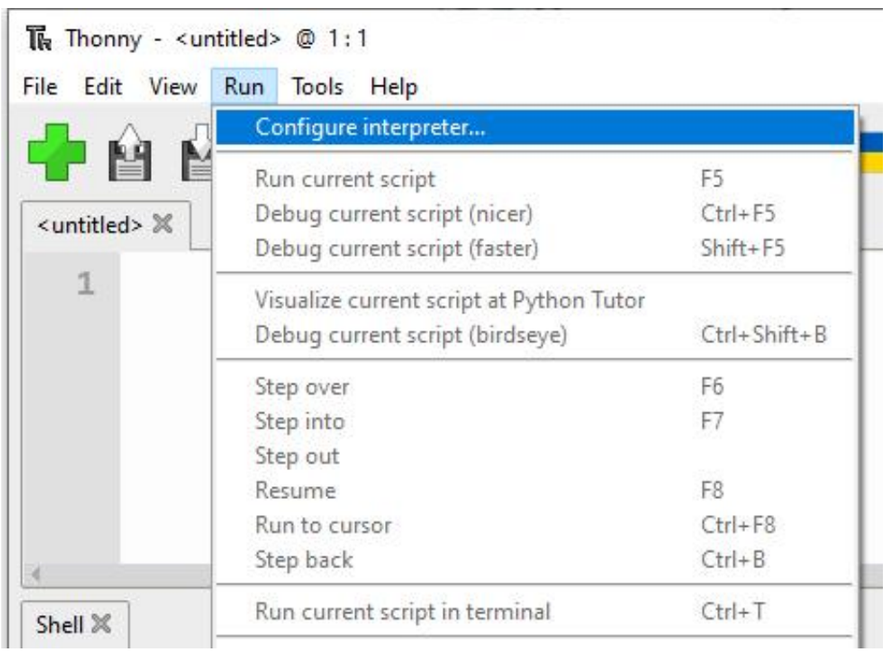


5. When the firmware finishes burning, Raspberry Pi Pico will reboot automatically.After that, you can run Micropython.

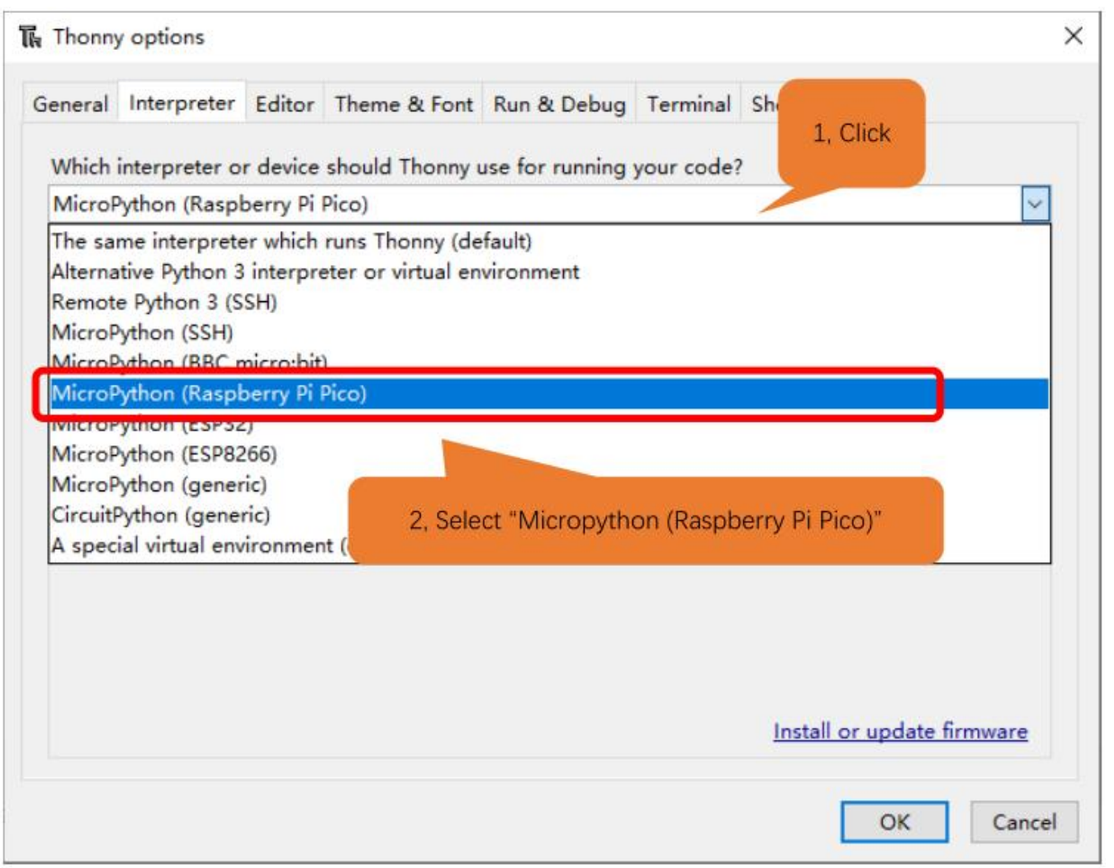
Thonny Connected to Raspberry Pi Pico(Important)

Option 1:

1. Open Thonny, click “run” and select “Select interpreter...”



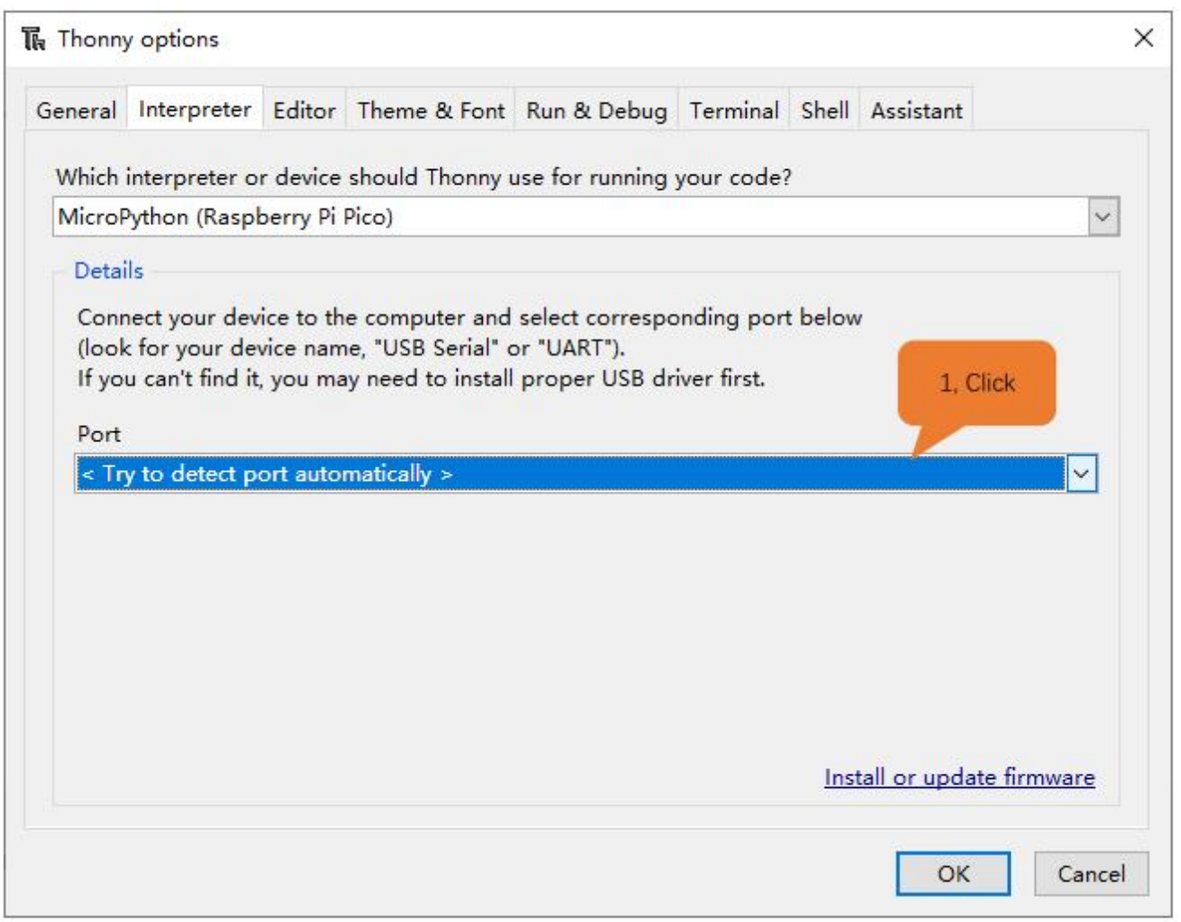
2. Select “Micropython (Raspberry Pi Pico)”.



3. Select “USB Serial Device (COMx)”， The number x of COMx may varifies among different computers. You only need to make sure selecting “USB Serial Device (COMx)”

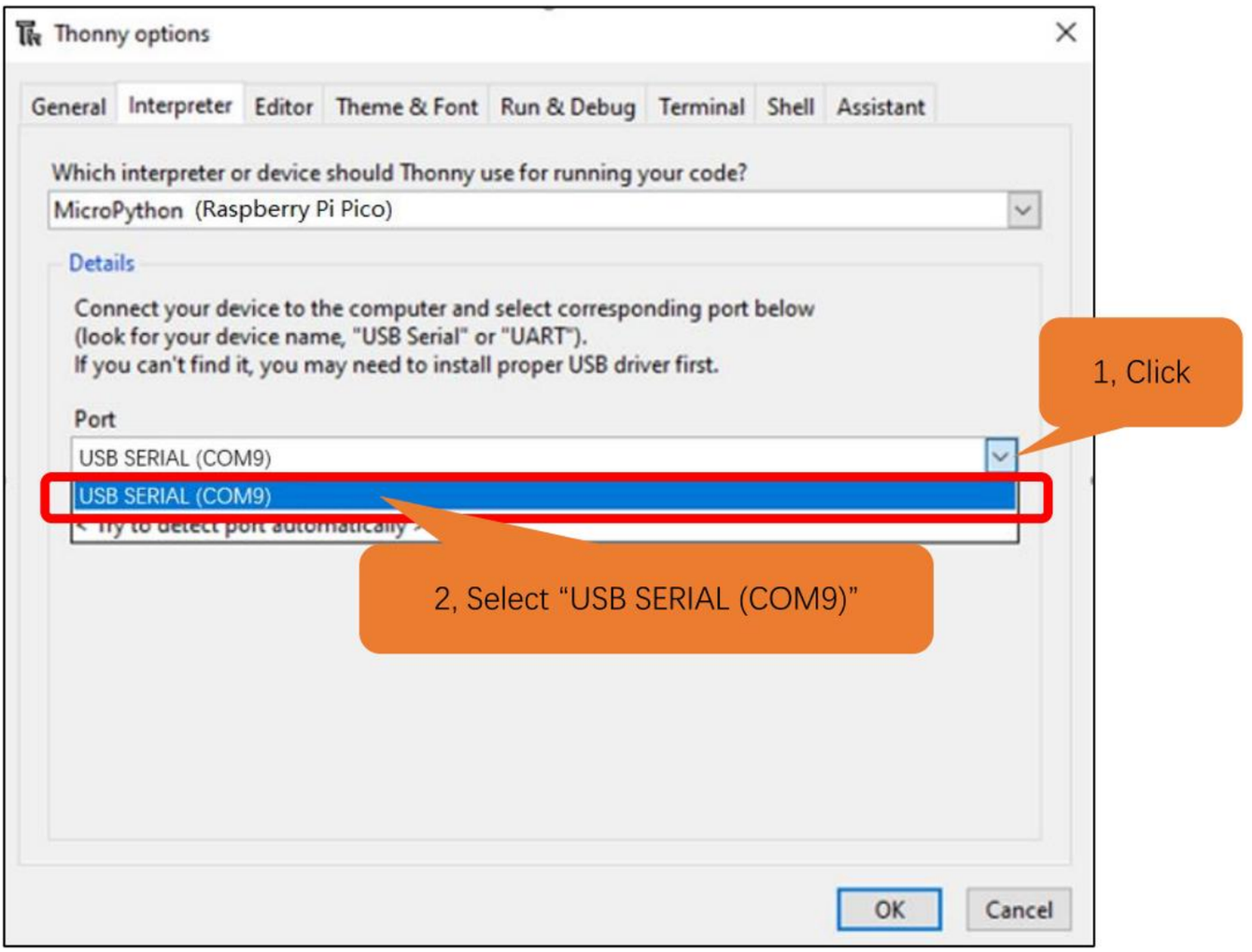
How to determine the port on which your Raspberry Pi Pico communicates with your computer?

Step 1: When Pico doesn’t connect to computer, open Thonny, click “Run”, select “Select interpreter” and then a dialog box will pop up, click “Port” and you can check the ports currently connected to your computer, as shown below:



Step 2: Close the dialog box. Connect Pico to your computer, click ”Run” again and select “Select interpreter”. Click “Port” on the pop-up window and check the current ports. Now there is a newly added port, with which Pico communicates with the computer.

If no USB Serial COM is found, you need to re-program the framework



Note

If no USB Serial COM is found, you need to Burning Micropython Firmware again.

If it’s still not working, some things to try are:

- Try a different USB cable.Some USB cables are charge-only, and some USB cables simply break after a lot of use.
- Other incompatibilities may include anything that connects to a USB serial device such as a SmartBoard or other Robotics platform.

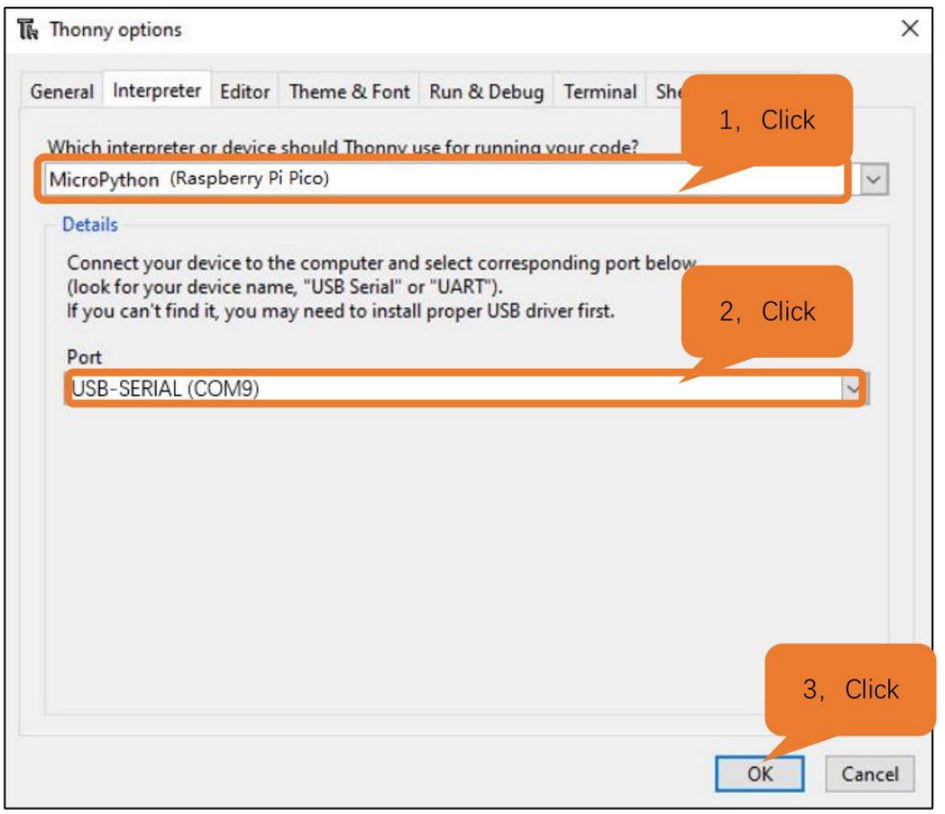
Remove other USB serial devices first.

- Try a different computer if you have one available, at least to install the firmware.
- Do not use USB Hub, it may cause insufficient power supply

For more information, refer to the official website.

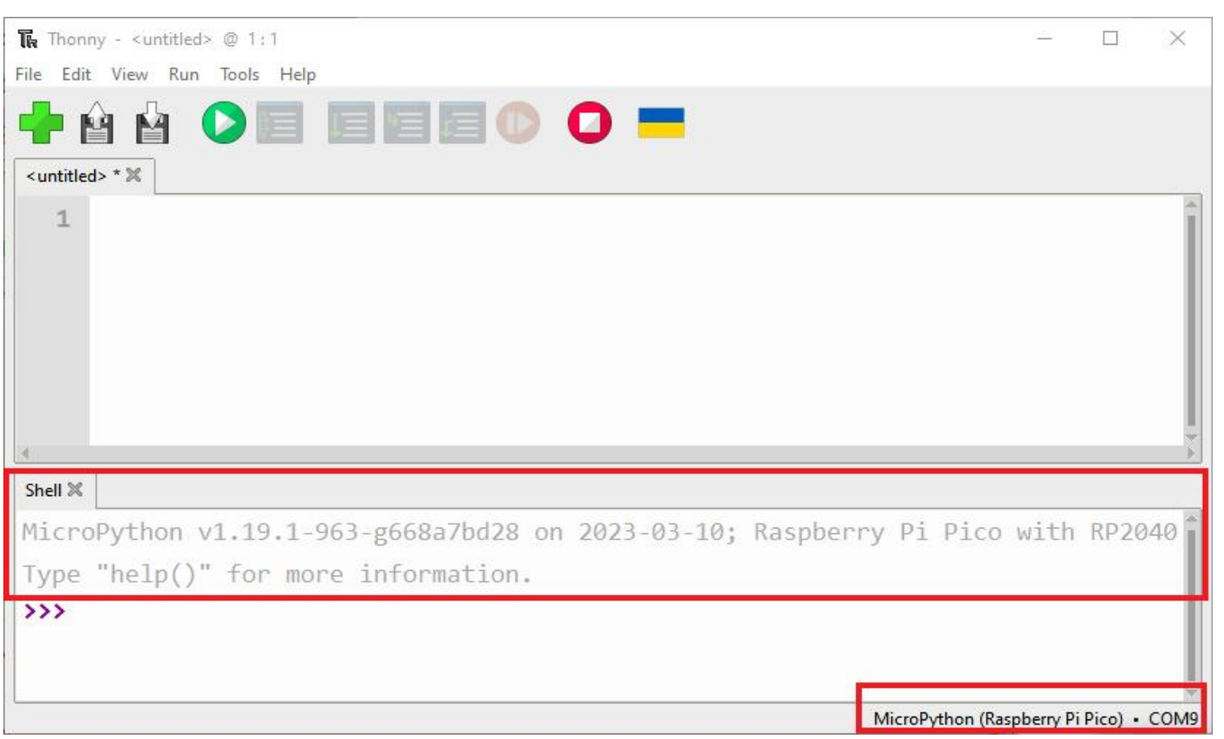
- **FAQ**

4. After selecting “Micropython (Raspberry Pi Pico)” and port, click “OK”



4. When the following message displays on Thonny Shell, it indicates Thonny has successfully connected to Pico.

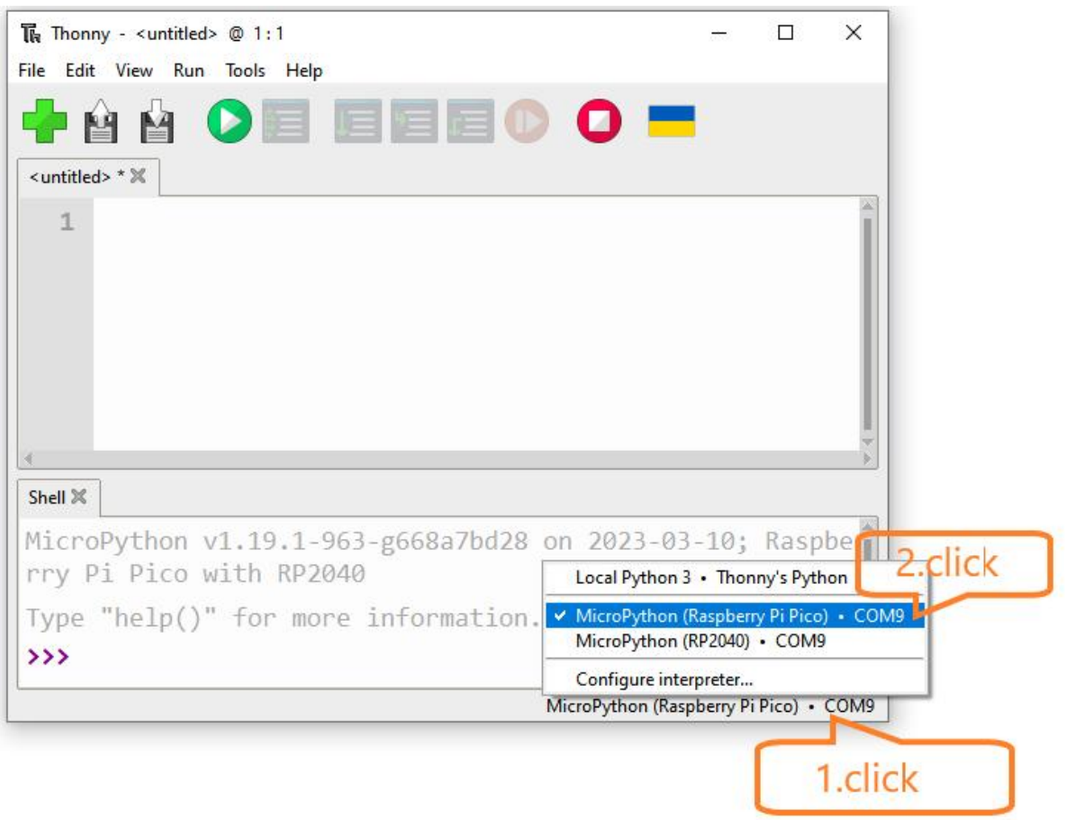
At the same time, the interpreter type is also displayed in the bottom right corner: MicroPython (Raspberry Pi Pico) • COMx



Option 2:

After the firmware finishes burning, Connect Pico to your computer.

left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) • COMx.



Note

If no USB Serial COM is found, you need to Burning Micropython Firmware again.

If it's still not working, some things to try are:

- Try a different USB cable.Some USB cables are charge-only, and some USB cables simply break after a lot of use.
- Other incompatibilities may include anything that connects to a USB serial device such as a SmartBoard or other Robotics platform.

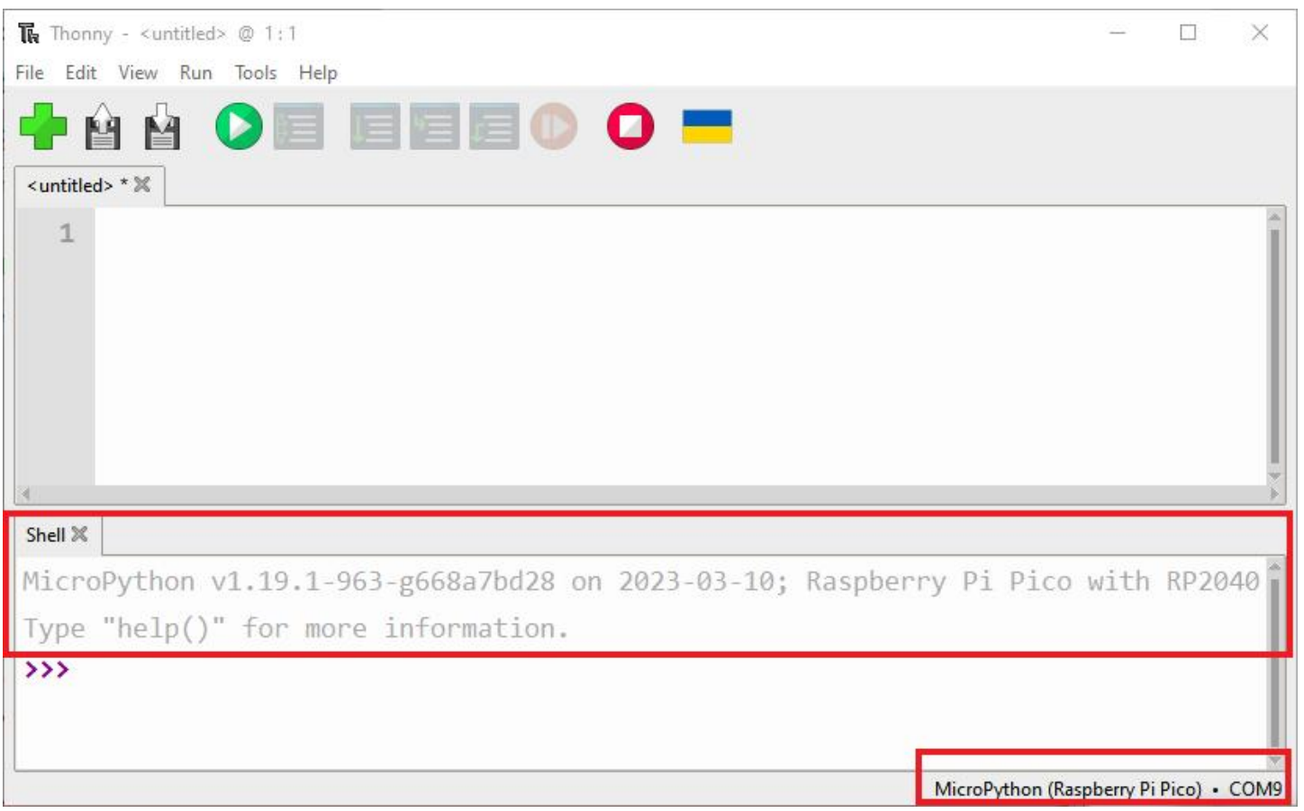
  Remove other USB serial devices first.
- Try a different computer if you have one available, at least to install the firmware.
- Do not use USB Hub, it may cause insufficient power supply

For more information, refer to the official website.

- **FAQ**

When the following message displays on Thonny Shell, it indicates Thonny has successfully connected to Pico.

At the same time, the interpreter type is also displayed in the bottom right corner: MicroPython (Raspberry Pi Pico) • COMx



Your First MicroPython Program

In MicroPython, there are two options/methods for running code:

- Interactive Mode
- Script Mode

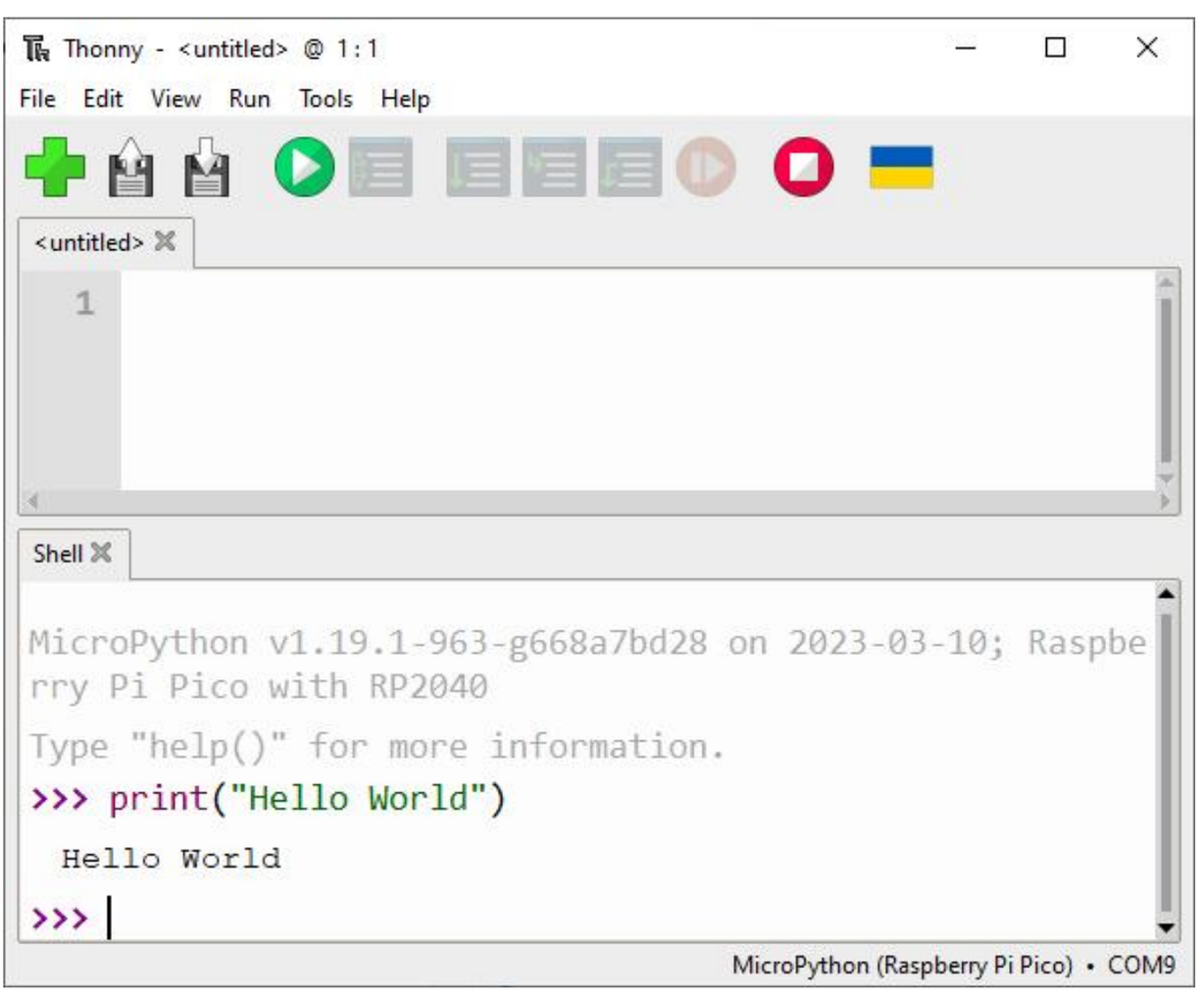
Interactive mode involves running your code directly from the Python shell, which can be accessed from the operating system's terminal. In script mode, you must create a file, save it with .py extension, and then run your code. Interactive mode is suitable when you are running a few lines of code. Script mode is recommended when you need to create large codes and save them for next time.

Interactive Mode

Interactive mode, also known as the REPL provides us with a quick way to run blocks or a single line of MicroPython code via the Python shell.

Click the Python Shell area, type the following command after >>>, and then Enter.

```
print("Hello, World!")
```



Note

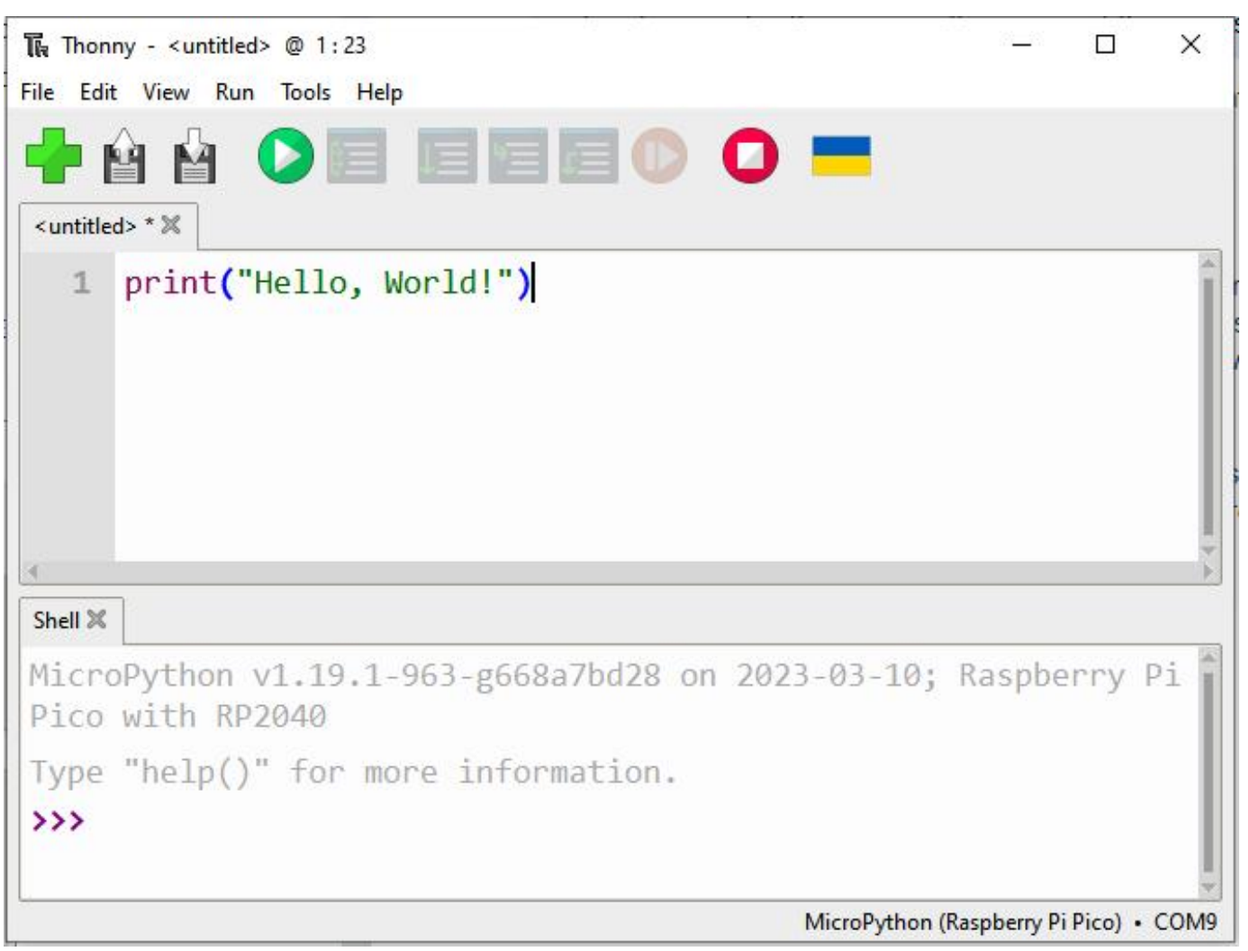
If your program does not run, but prints a “syntax error” message to the Shell area, then there is an error in what you have written.MicroPython needs to write its instructions in a very specific way: miss parentheses or quotation marks, spell print errors, or give it a capital P, or add extra symbols somewhere in the instruction, and it won’t run . Try to type the command again and make sure it is the same as this one, and then press Enter.

You will find that the message hello world will be printed out immediately in the Shell area.

Script Mode

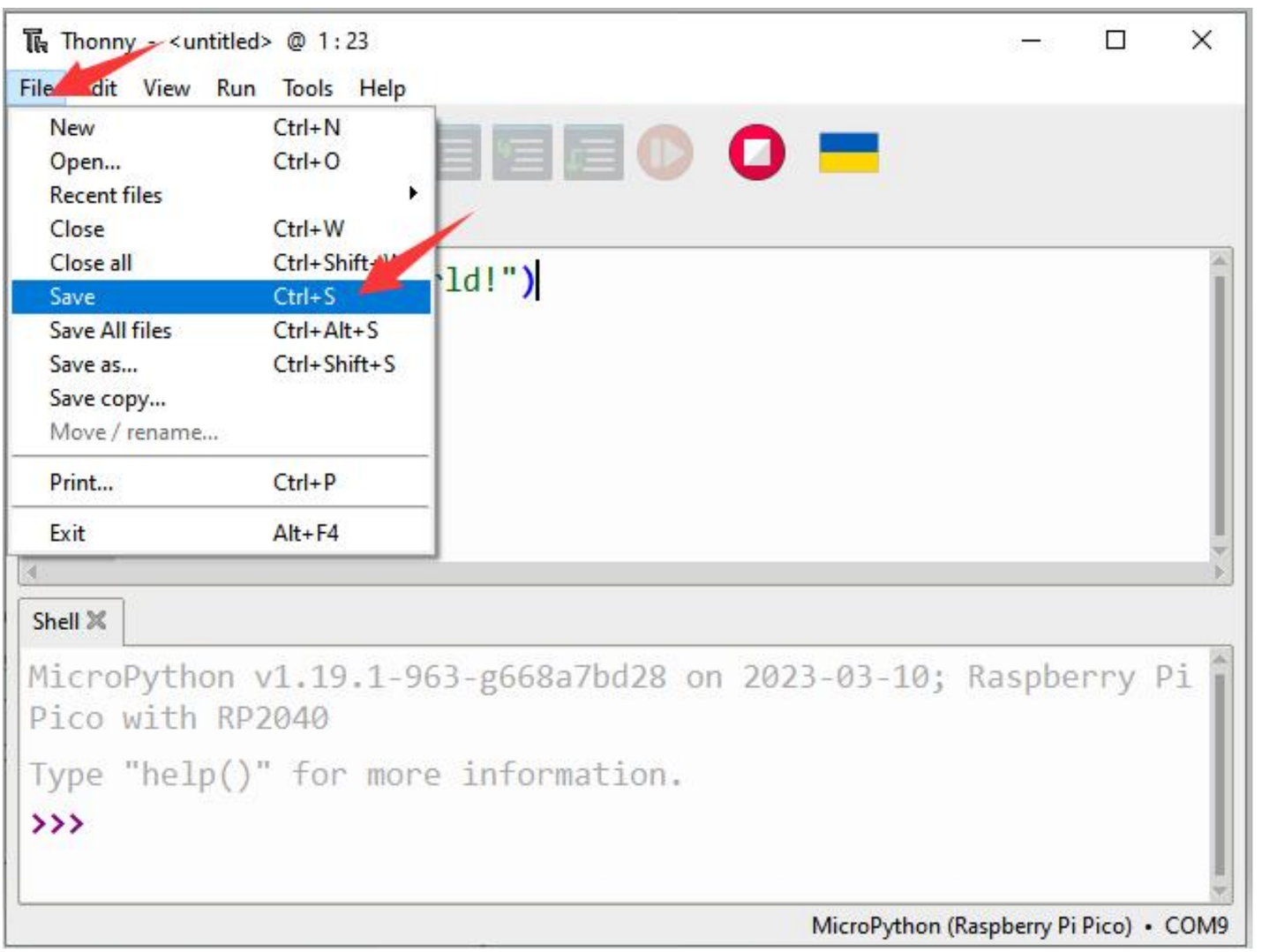
Interactive mode is not recommended if you need to write a long piece of Python code, or if your Python script spans multiple files. In this case, script mode can be used. In script mode, you write your code and save it with a .py extension, which stands for “Python”.

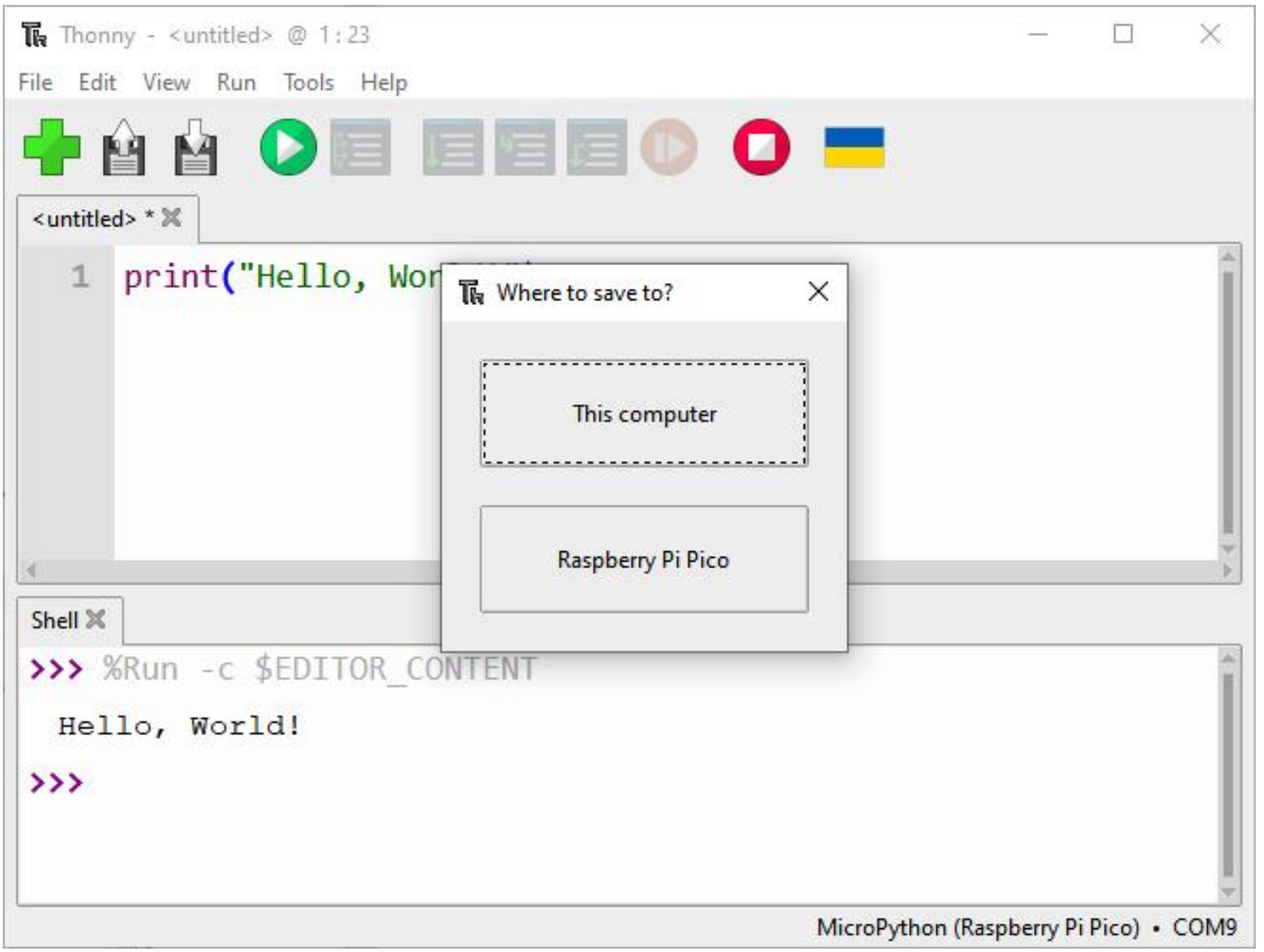
Enter the same command print("Hello, World!") in the script area, when you press the ENTER key, the program will not run, only one more blank line in the script area.



Save Code File

Click **File>>Save**,a window will pop up asking to save to This computer or Raspberry Pi Pico ?(if your Pico is already plugged into the computer and connect to Thonny) Thonny Connected to Raspberry Pi Pico

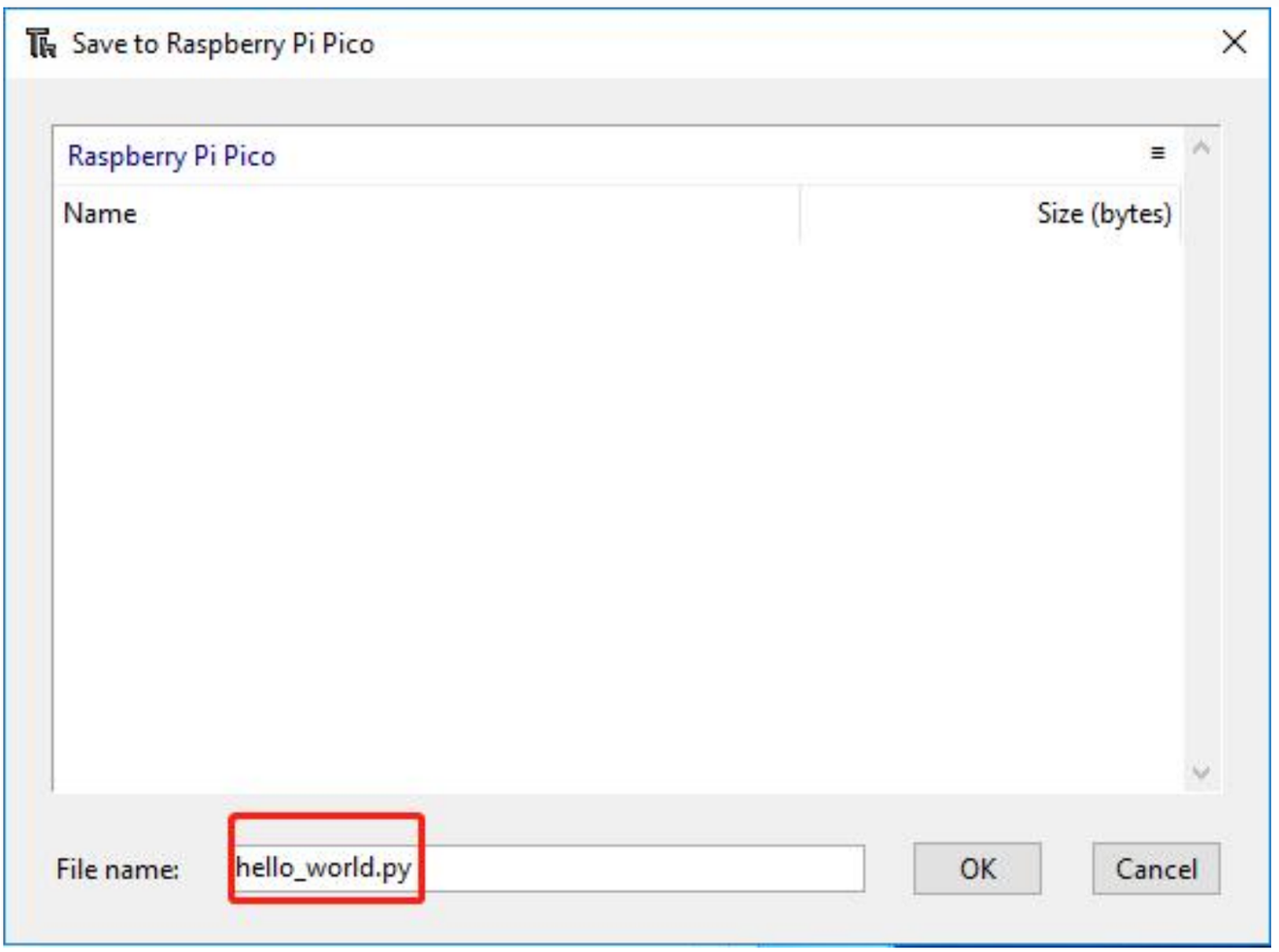




Note

You can choose one of the storage.When you tell Thonny to save your program on the Raspberry Pi Pico, if you unplug the Pico and plug it into someone else's computer, your program is still saved on the Pico.

Choose the location you want to save, then enter the file name hello_world and the extension .py, and then click OK.

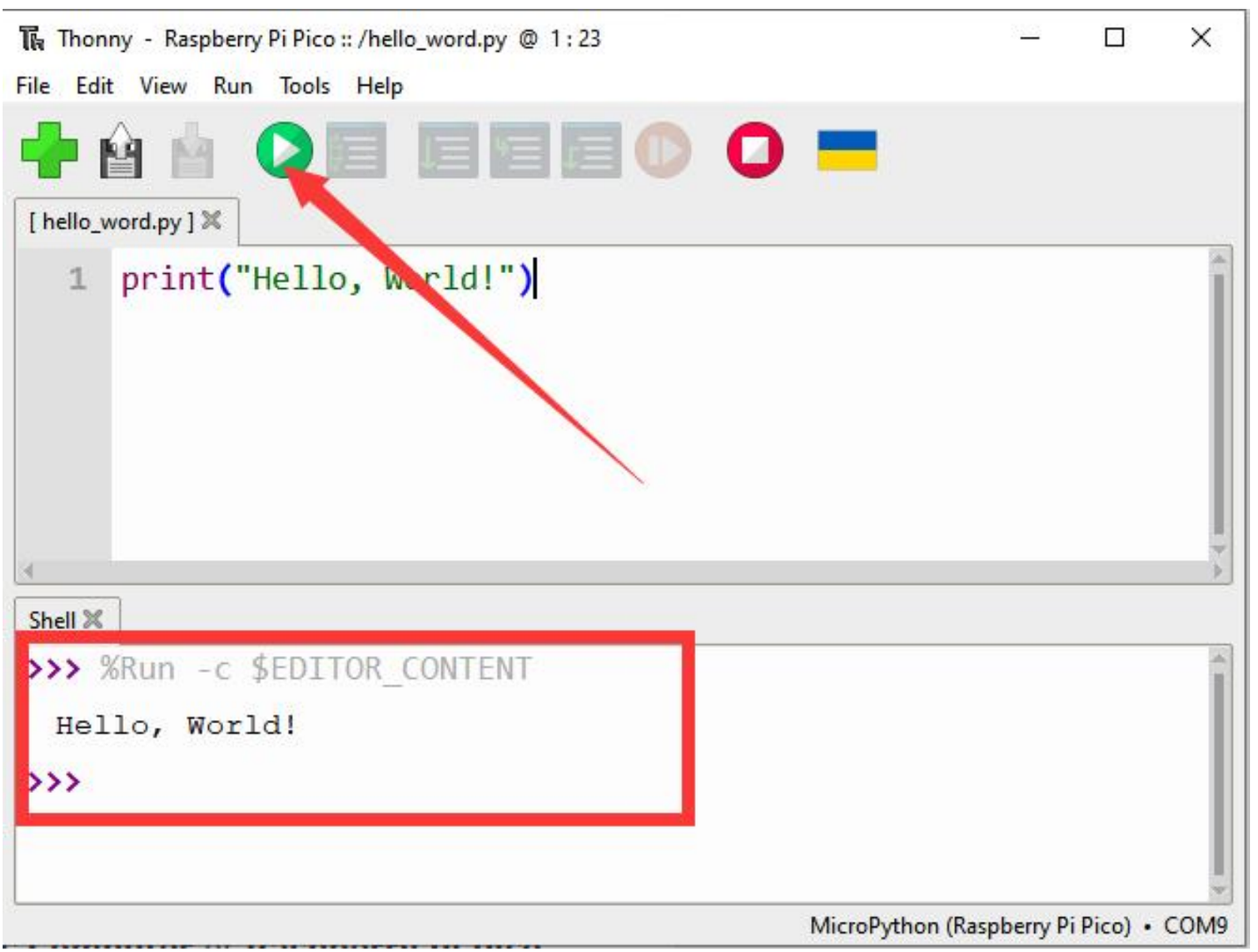


Note

You can save your code under any name, but it's best to describe what kind of code it is, and don't name it with meaningless names such as abc.py. It is important to note that if you save the code file name as main.py, it will run automatically when the power is turned on.

Run Code

You need click **Run Current Script** or simply press F5 to run it. you will see 2 lines of information in the Shell area.




```
>>> %Run -c $EDITOR_CONTENT
Hello, World!
```

`%Run -c $EDITOR_CONTENT` is an instruction from Thonny telling the MicroPython interpreter on your Pico to run the contents of the script area – the 'EDITOR_CONTENT'.

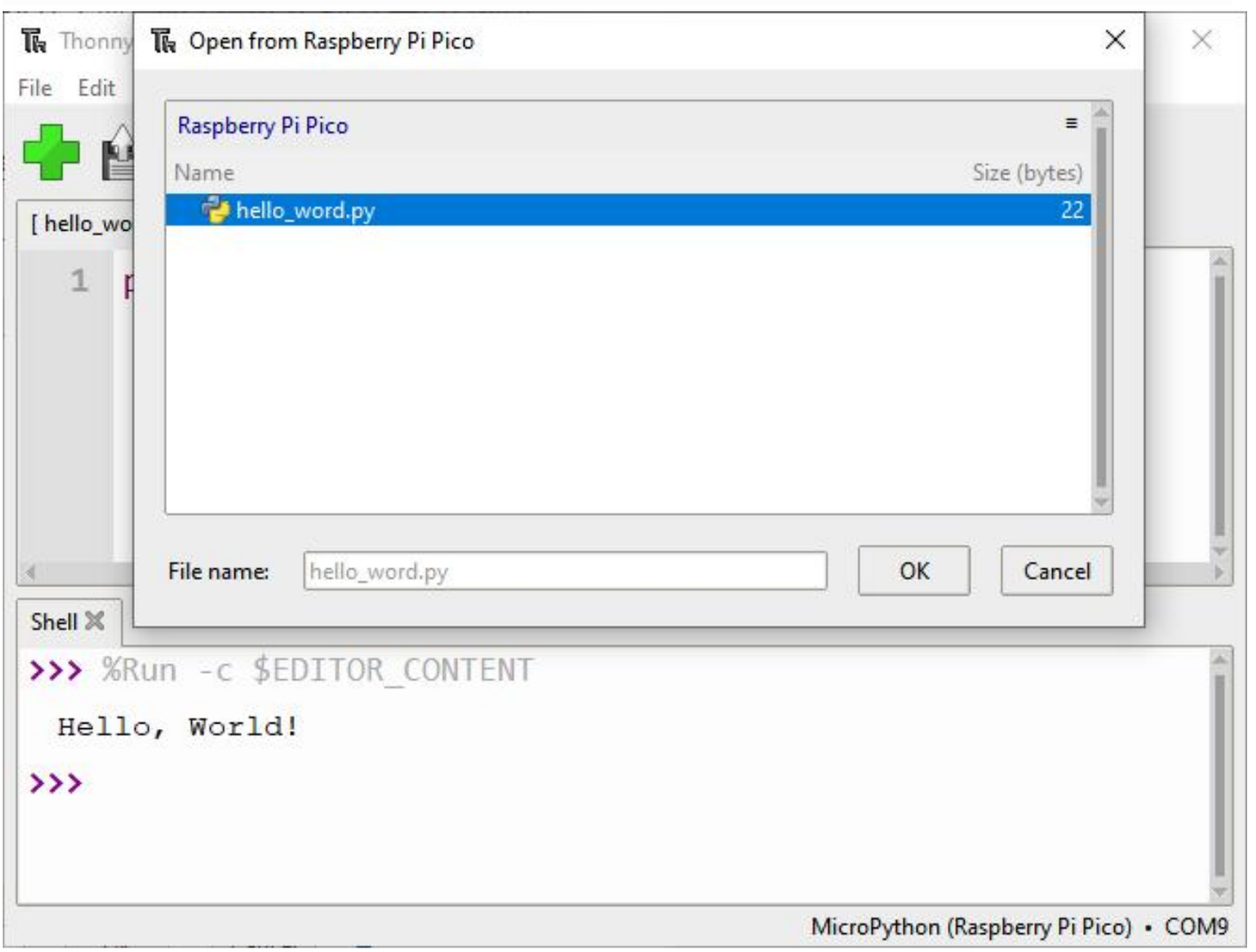
`Hello, World!` is the output of the program – the message you told MicroPython to print.

Open Code File

In Script mode, it is very easy for you to open the previously saved code again, but the code entered in Interactive Mode will not be saved and can only be re-entered.

Click the **File>>open** icon in the Thonny toolbar, just like when you save the program, you will be asked whether you want to save it to **This Computer** or **Raspberry Pi Pico**, for example, click **Raspberry Pi Pico** and you will see a list of all programs you save to your Pico.

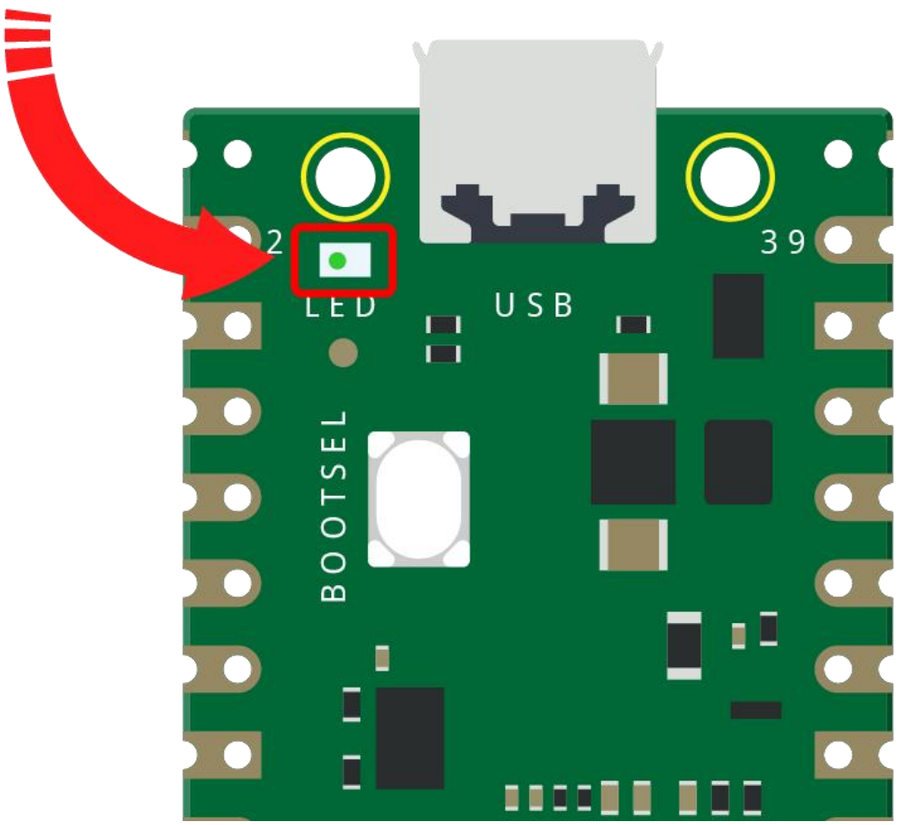
Find hello_world.py in the list, if your Pico is new, it will be the only file there, and click it to select it, then click OK. Your program will be loaded into Thonny, ready to edit or run it again for you.



Project 1 Hello, LED!(important)

Just as printing “Hello, world!” is the first step in learning programming, letting the LED light up is the traditional entry to learning physical programming.

There is a small LED on the top of the Pico. Like other LEDs, it will glow when power is on and go out when power is off.

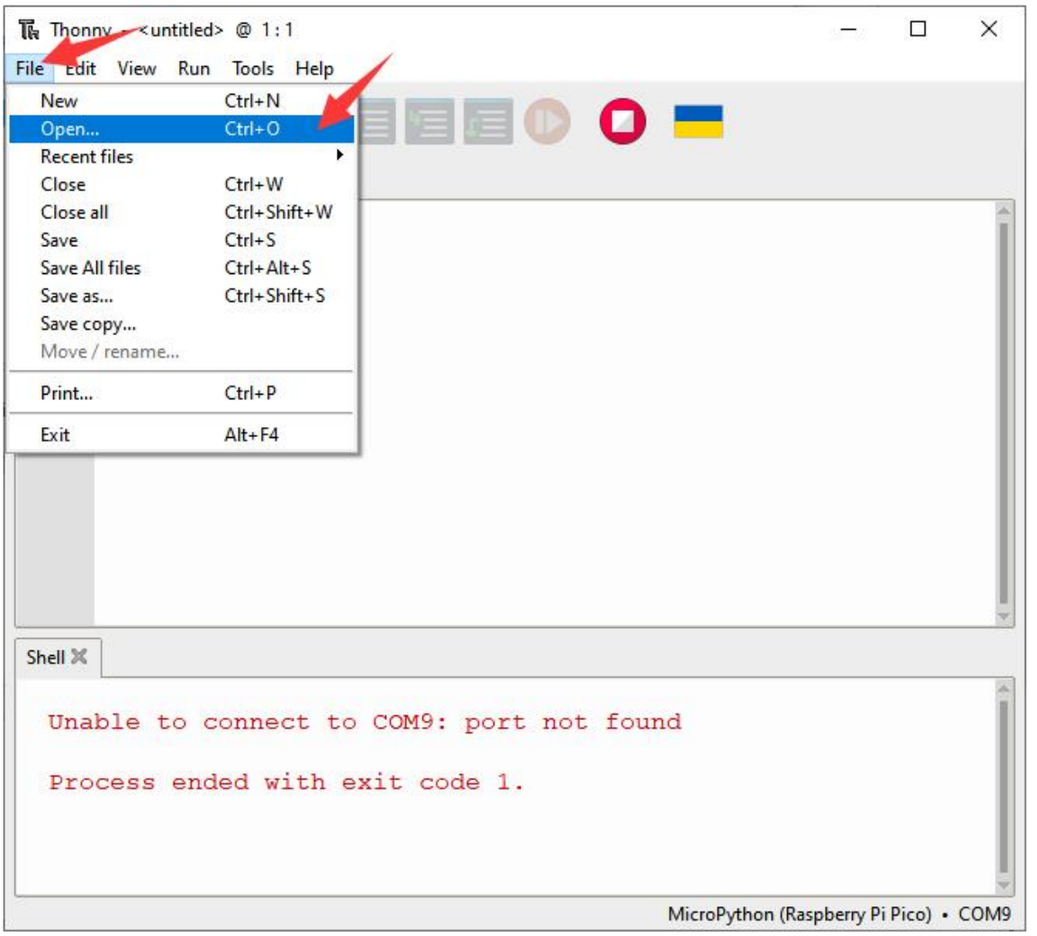


Open Code File

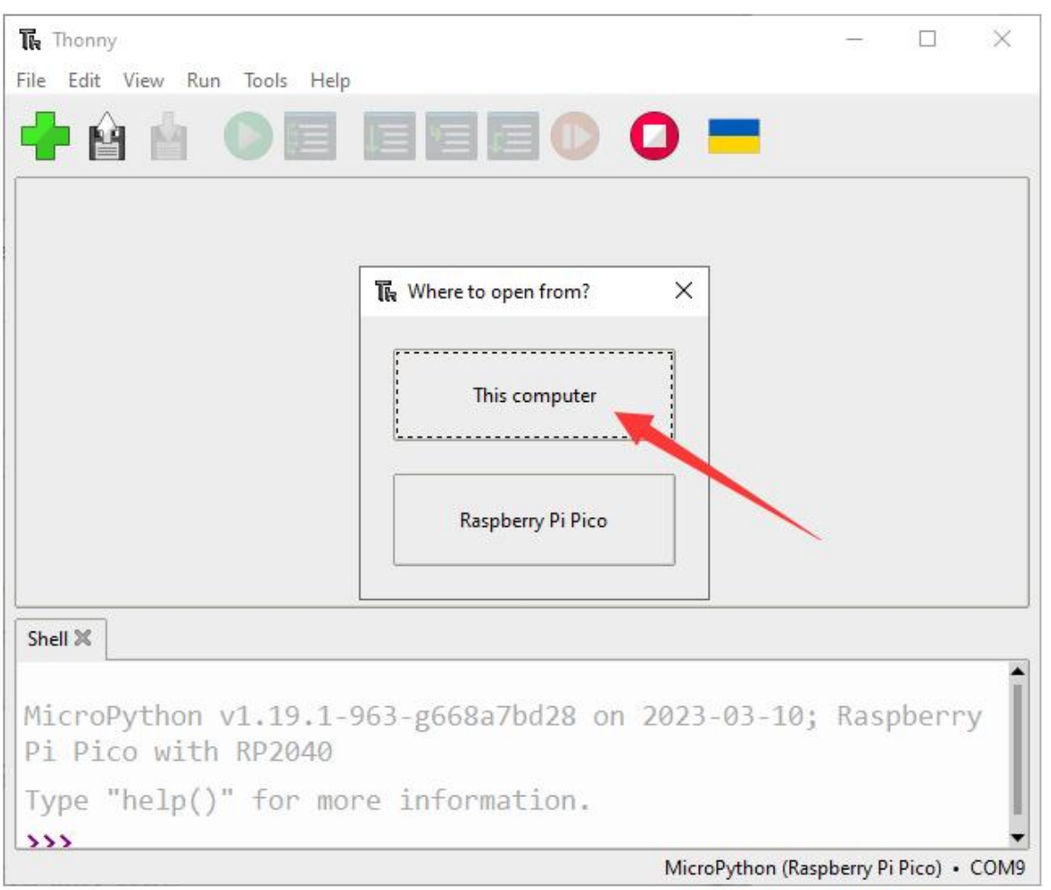
Click the File>>open icon to open Project 1 \ Hello_Led.py file.

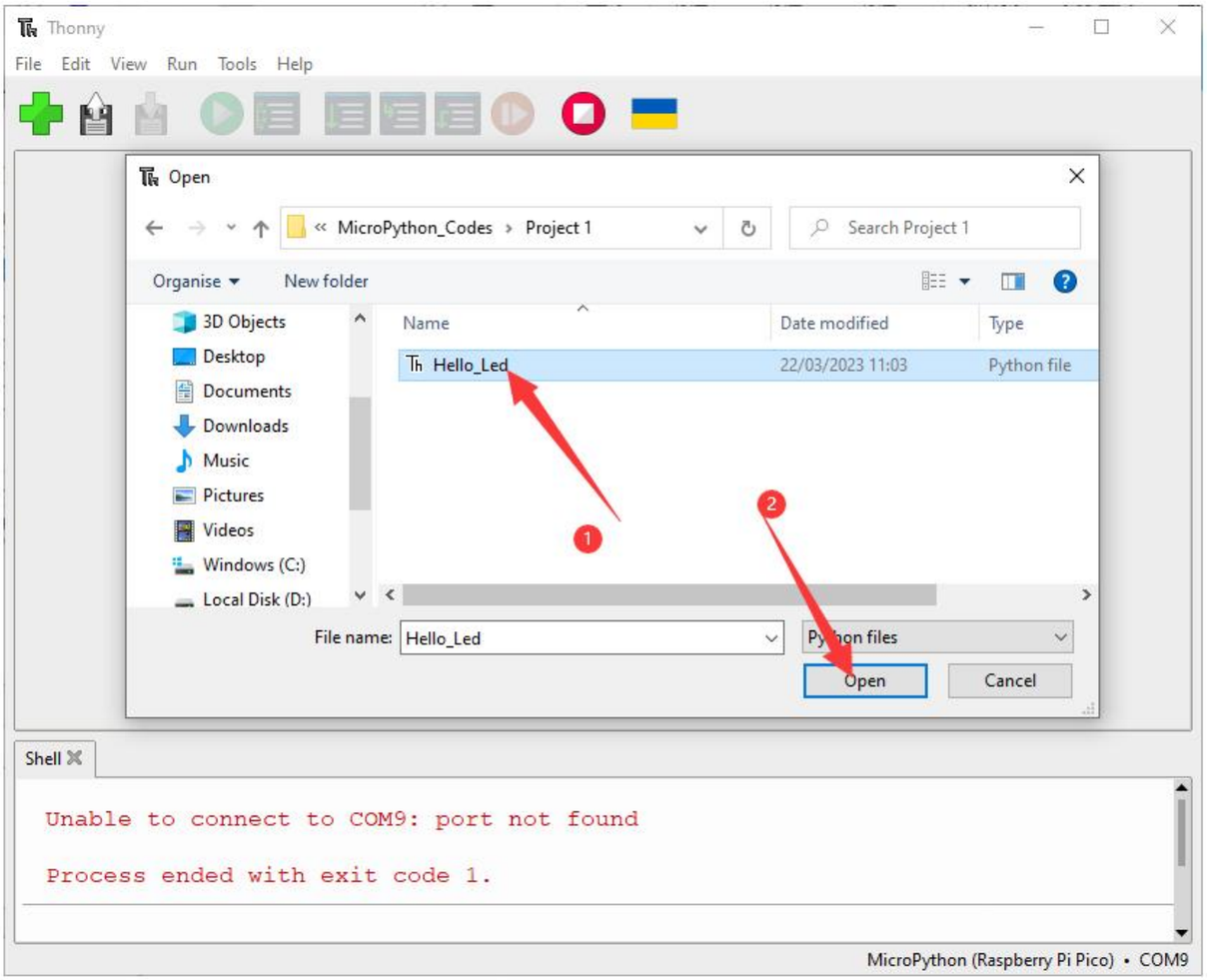
If you have downloaded the tutorial. You can find the .py file.

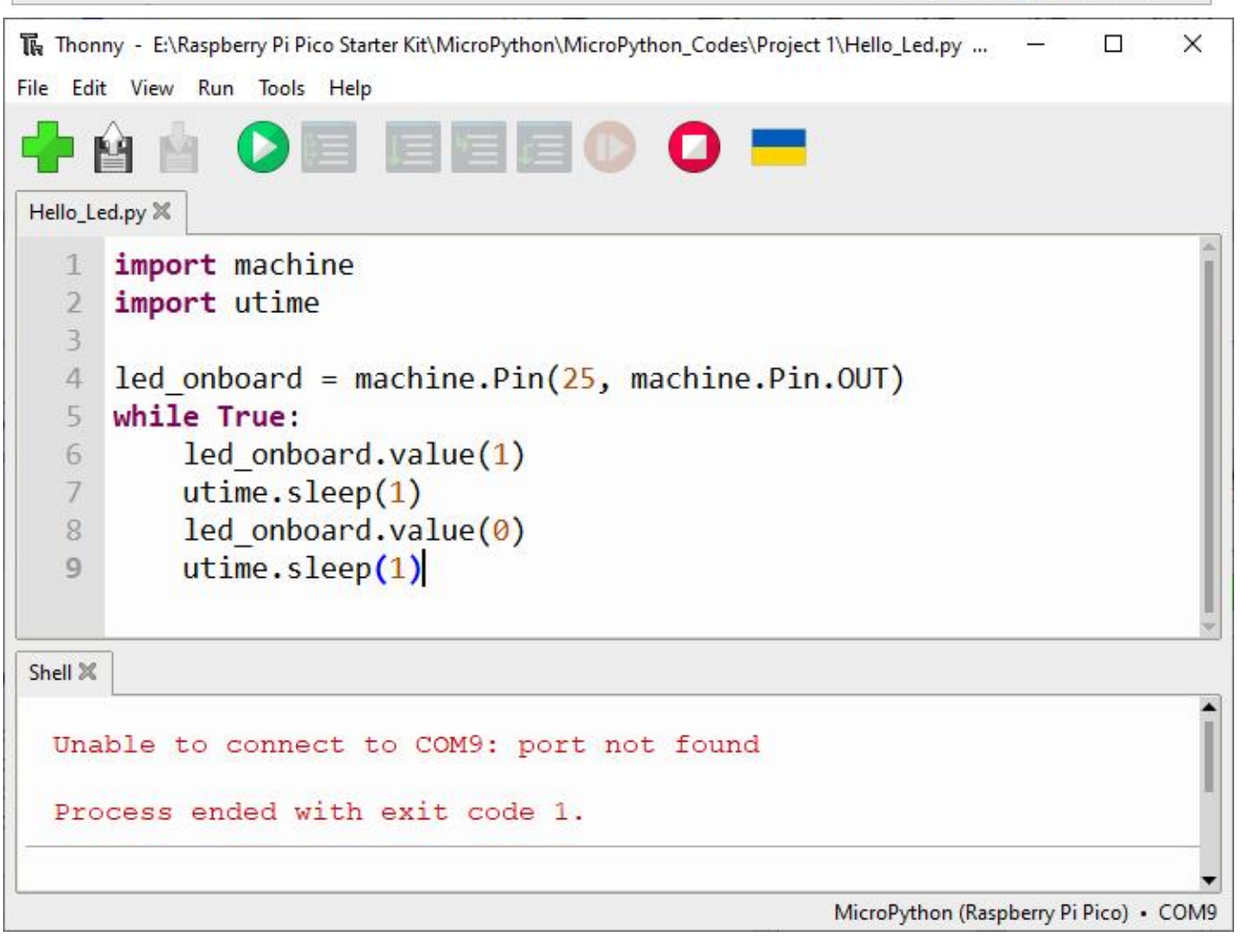
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes



If the pico is connected to Thonny, it will ask: where to open from? Click **This computer**.



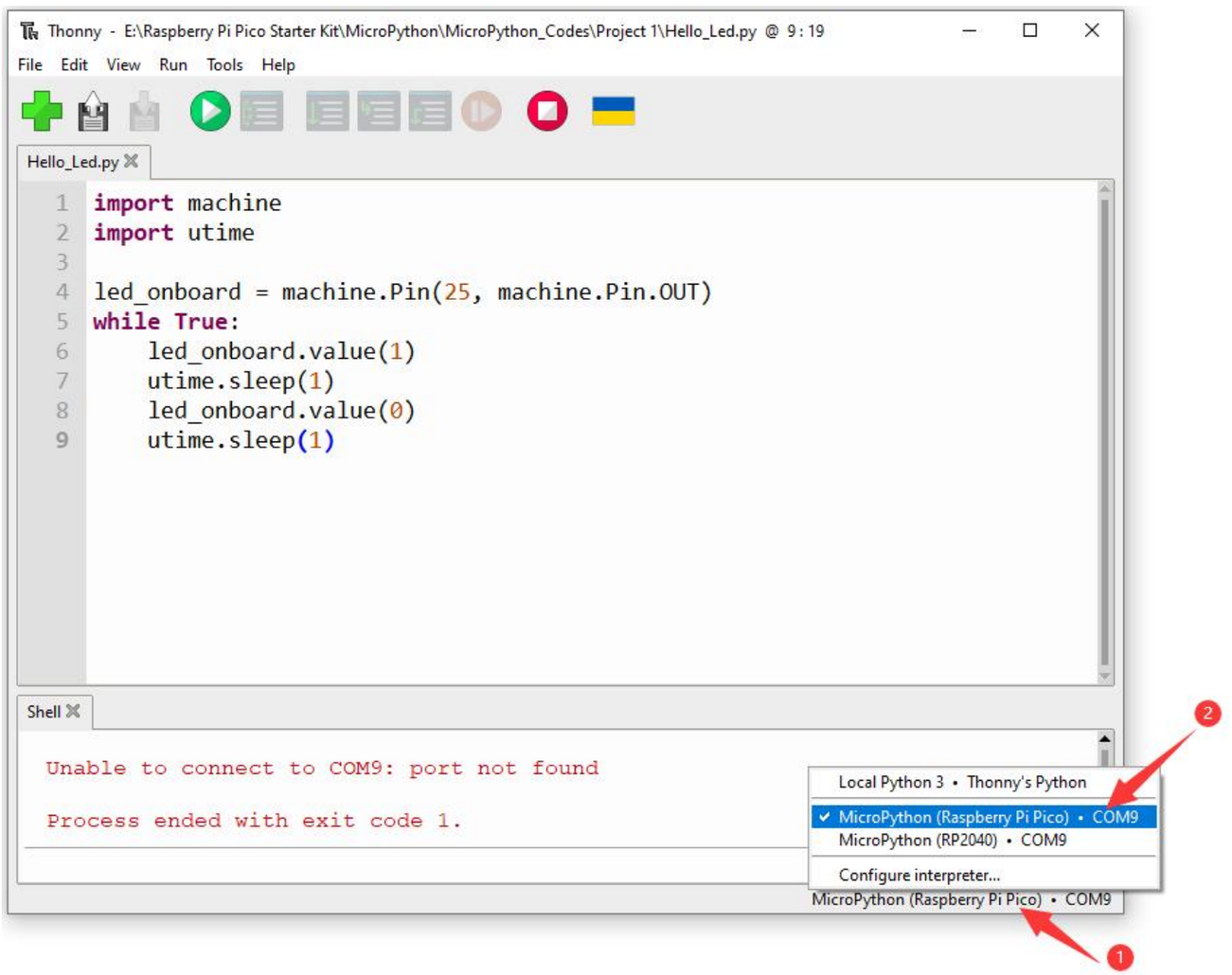

Thonny
File Edit View Run Tools Help
Open
MicroPython_Codes › Project 1
Search Project 1
Organise
New folder
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Local Disk (D:)
Name
Date modified
Type
Hello_Led
22/03/2023 11:03
Python file
1
2
File name: Hello_Led
Python files
Open
Cancel
Shell
Unable to connect to COM9: port not found
Process ended with exit code 1.
MicroPython (Raspberry Pi Pico) • COM9


Thonny - E:\Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes\Project 1\Hello_Led.py ...
File Edit View Run Tools Help
Hello_Led.py
1 import machine
2 import utime
3
4 led_onboard = machine.Pin(25, machine.Pin.OUT)
5 while True:
6 led_onboard.value(1)
7 utime.sleep(1)
8 led_onboard.value(0)
9 utime.sleep(1)
Shell
Unable to connect to COM9: port not found
Process ended with exit code 1.
MicroPython (Raspberry Pi Pico) • COM9

Thonny Connected to Raspberry Pi Pico

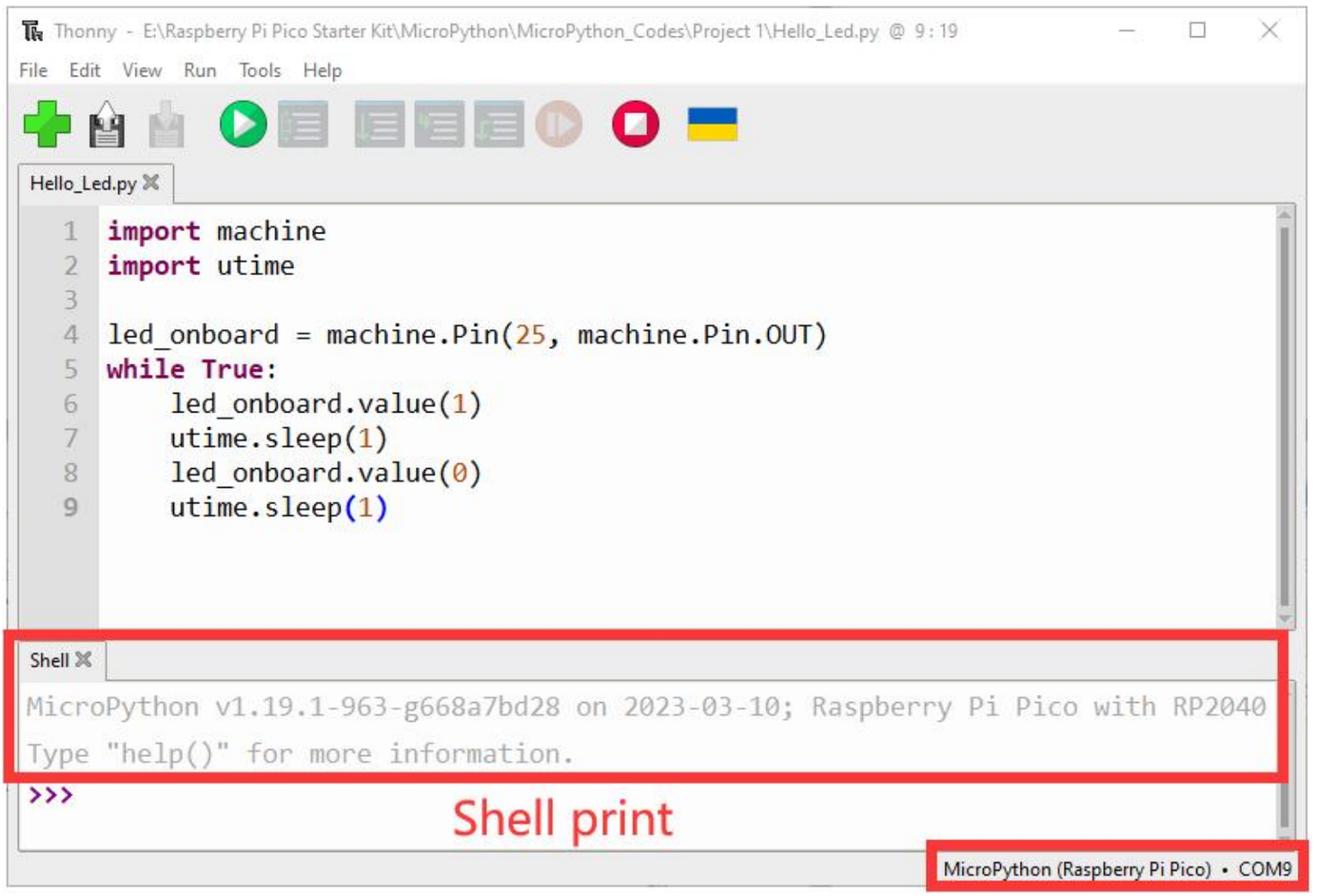
After the firmware finishes burning, Connect Pico to your computer.

left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) • COMx. Have Question?



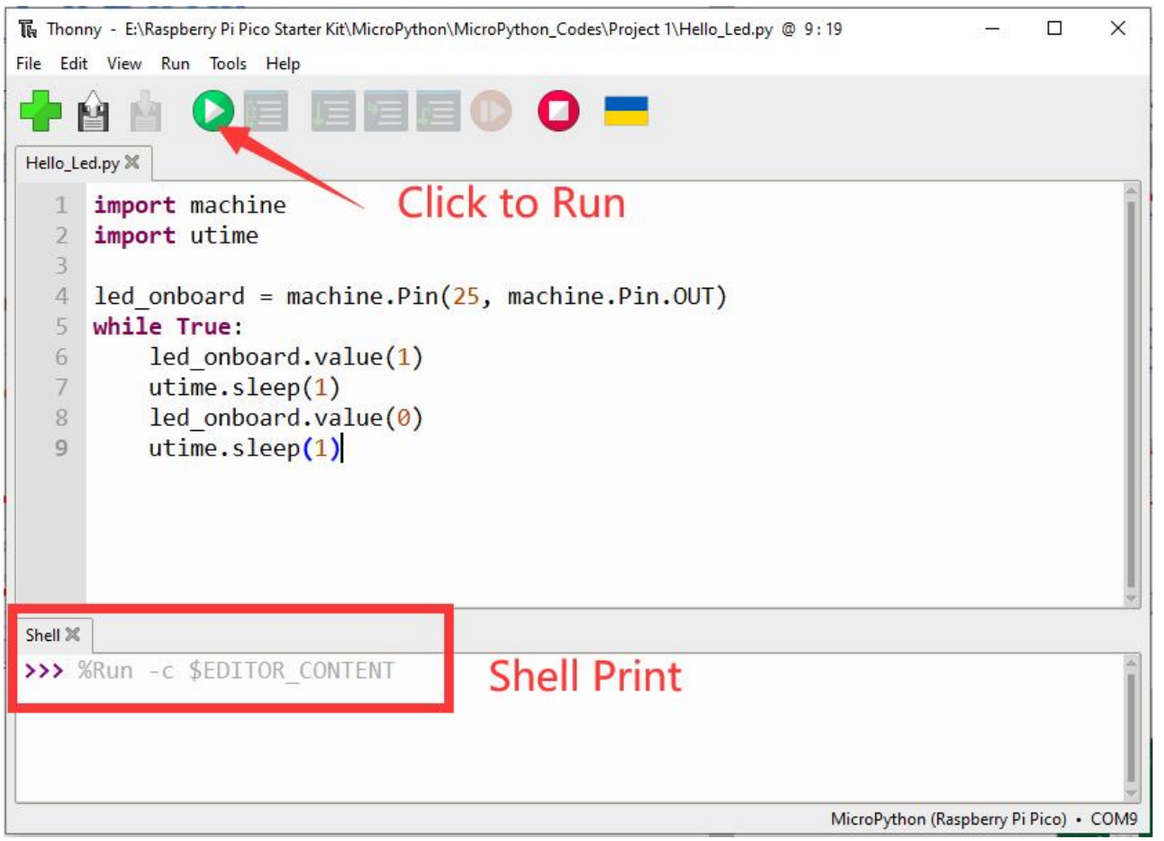
When the following message displays on Thonny Shell, it indicates Thonny has successfully connected to Pico.

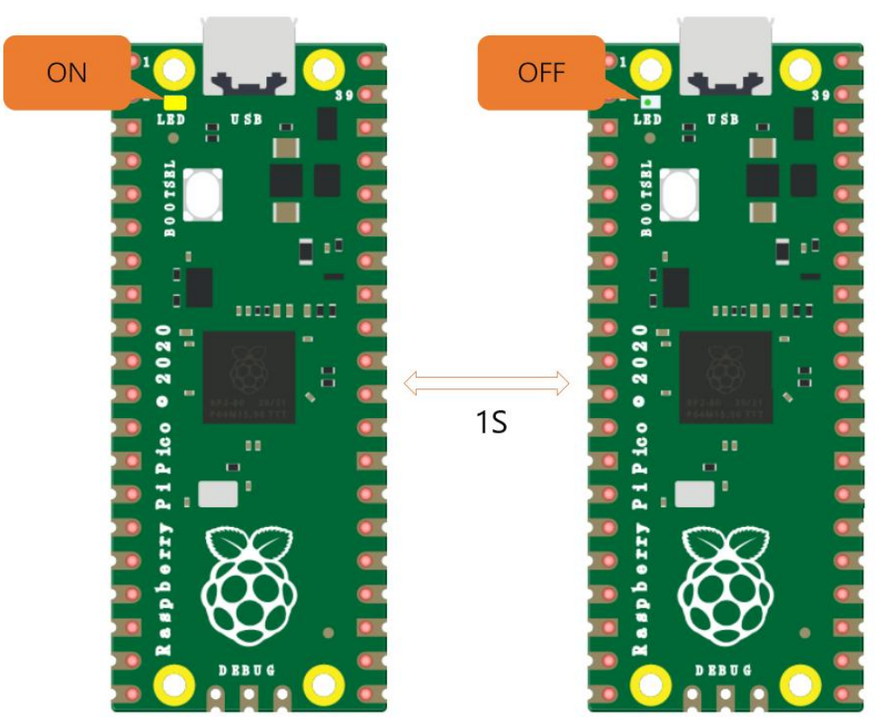
At the same time, the interpreter type is also displayed in the bottom right corner: MicroPython (Raspberry Pi Pico) • COMx.


Thonny - E:\Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes\Project 1\Hello_Led.py @ 9 : 19
File Edit View Run Tools Help
Hello_Led.py
1 import machine
2 import utime
3
4 led_onboard = machine.Pin(25, machine.Pin.OUT)
5 while True:
6 led_onboard.value(1)
7 utime.sleep(1)
8 led_onboard.value(0)
9 utime.sleep(1)
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
Shell print
MicroPython (Raspberry Pi Pico) • COM9

Run Code

click “Run Current Script” or simply press F5 to run it to make the LED blink!





How it works?

The onboard LED is connected to the GP25 pin, if you carefully observe the Pico pinout, you will find that GP25 is one of the hidden pins, which means that we cannot use this pin (even if GP25 is used in exactly the same way as other pins). The advantage of this design is that even if you don't connect any external components, you can still have an OUTPUT to test the program.

The machine library is required to use GPIO.

```
import machine
```

This library contains all the instructions needed to communicate between MicroPython and Pico. Without this line of code, we will not be able to control any GPIOs (Of course including GP25).

The next thing to notice is this line:

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

An object named led_onboard is defined here. Technically, it can be any name, it can be x, y, banana, Micheal_Jackson, or any character, but it is best to use a name that describes the purpose to ensure that the program is easy to read.

The second part of this line (the part after the equal sign) calls the Pin function in the machine library. It is used to tell Pico's GPIO pins what to do. The Pin function has two parameters: the first parameter (25) means the pin you want to set; the second parameter (machine.Pin.OUT) tells that the pin should be used as an output instead of an input.

The above code has "set" the pin, but it will not light up the LED. To do this, we also need to "use" the pin.

```
led_onboard.value(1)
```

We have set up the GP25 pin before and named it led_onboard. The function of this statement is to set the value of led_onboard to 1 to turn the on-board LED on.

All in all, to use GPIO, these steps are necessary:

- **import machine library**: This is necessary, and it is only executed once in the entire program.
- **Set GPIO**: Each pin should be set before use.
- **Use**: Assign a value to the pin, each assignment will change the working state of the pin.

```
time.sleep(1)
```

Like machine, the time library is introduced here, which handles all time-related things, including the delay we need to use. Let's insert a delay sentence between led_onboard.value(1) and led_onboard.value(0), let them be separated by 1 seconds:

What More?

Usually, the library will have a corresponding API (Application Programming Interface) file. This is a concise reference manual that contains all the information needed to use this library, detailed introduction to functions, classes, return types, parameters, etc., and even comes with a tutorial.

In this article, we used MicroPython's machine and time libraries, we can find more ways to use them here.

- machine.Pin
- time

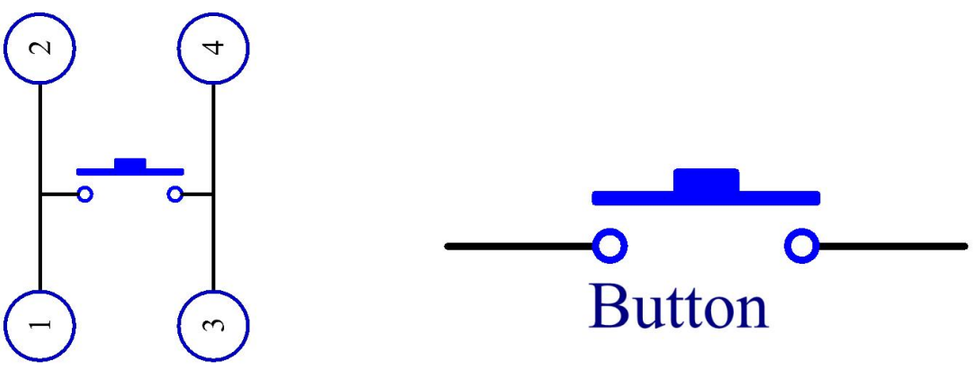
Project 2 Button Control LED

In the project, we will control the LED state through a Push Button Switch. When the button is pressed, our LED will turn ON, and when it is released, the LED will turn OFF. This describes a Momentary Switch.

Component knowledge

Push button

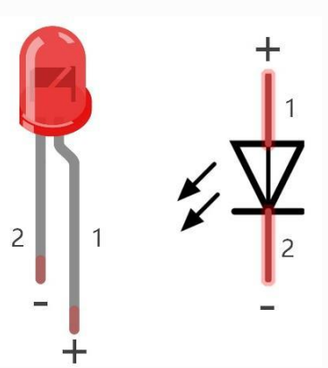
This type of Push Button Switch has 4 pins (2 Pole Switch). Two pins on the left are connected, and both left and right sides are the same per the illustration:



When the button on the switch is pressed, the circuit is completed (your project is Powered ON).

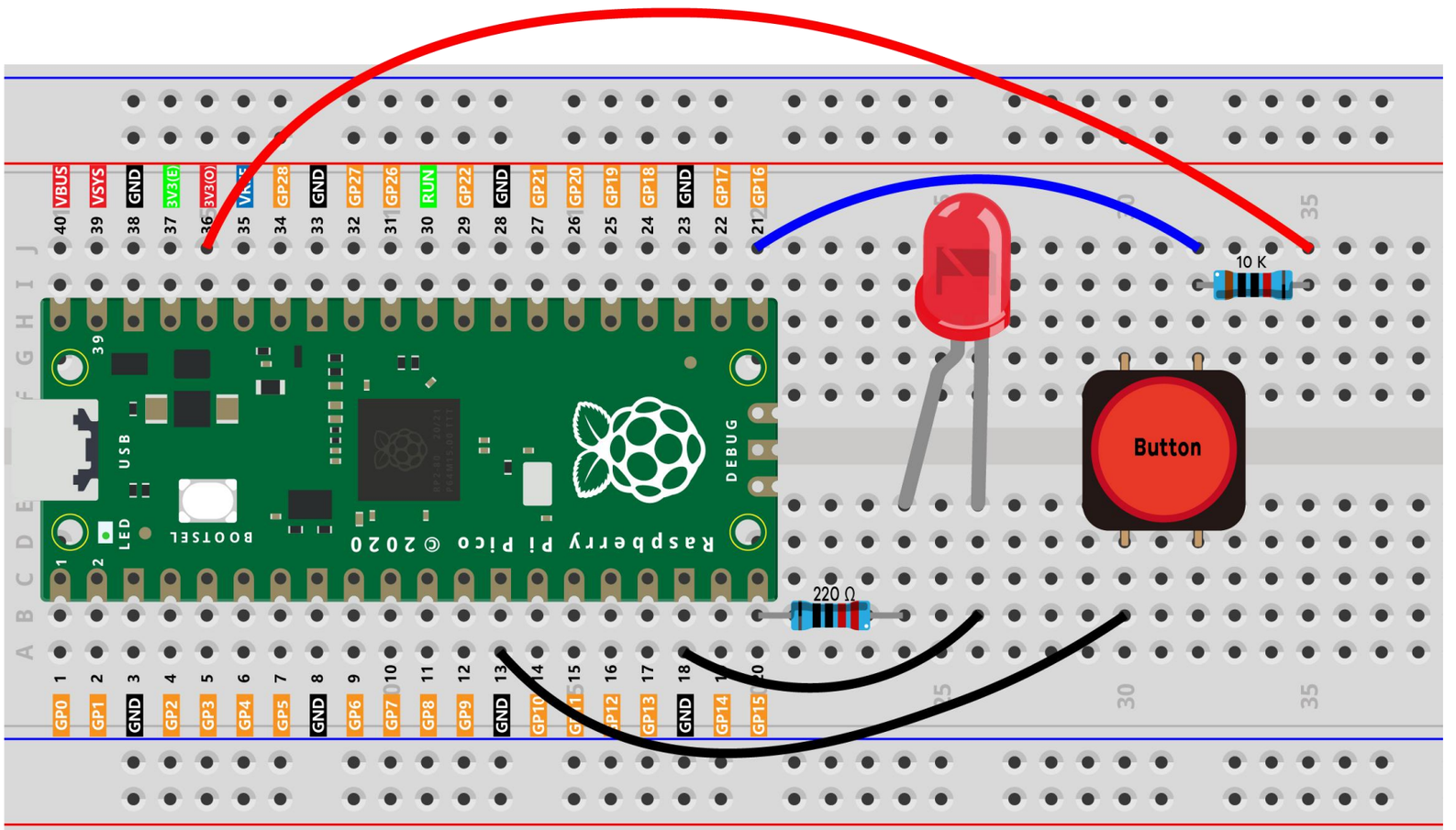
LED

An LED is a type of diode. All diodes only work if current is flowing in the correct direction and have two Poles.



Tip:Learn more about Button LED

Wiring



Code

This project is designed to learn to control an LED with a push button switch. First, we need to read the state of the switch and then decide whether the LED is turned on or not based on it.

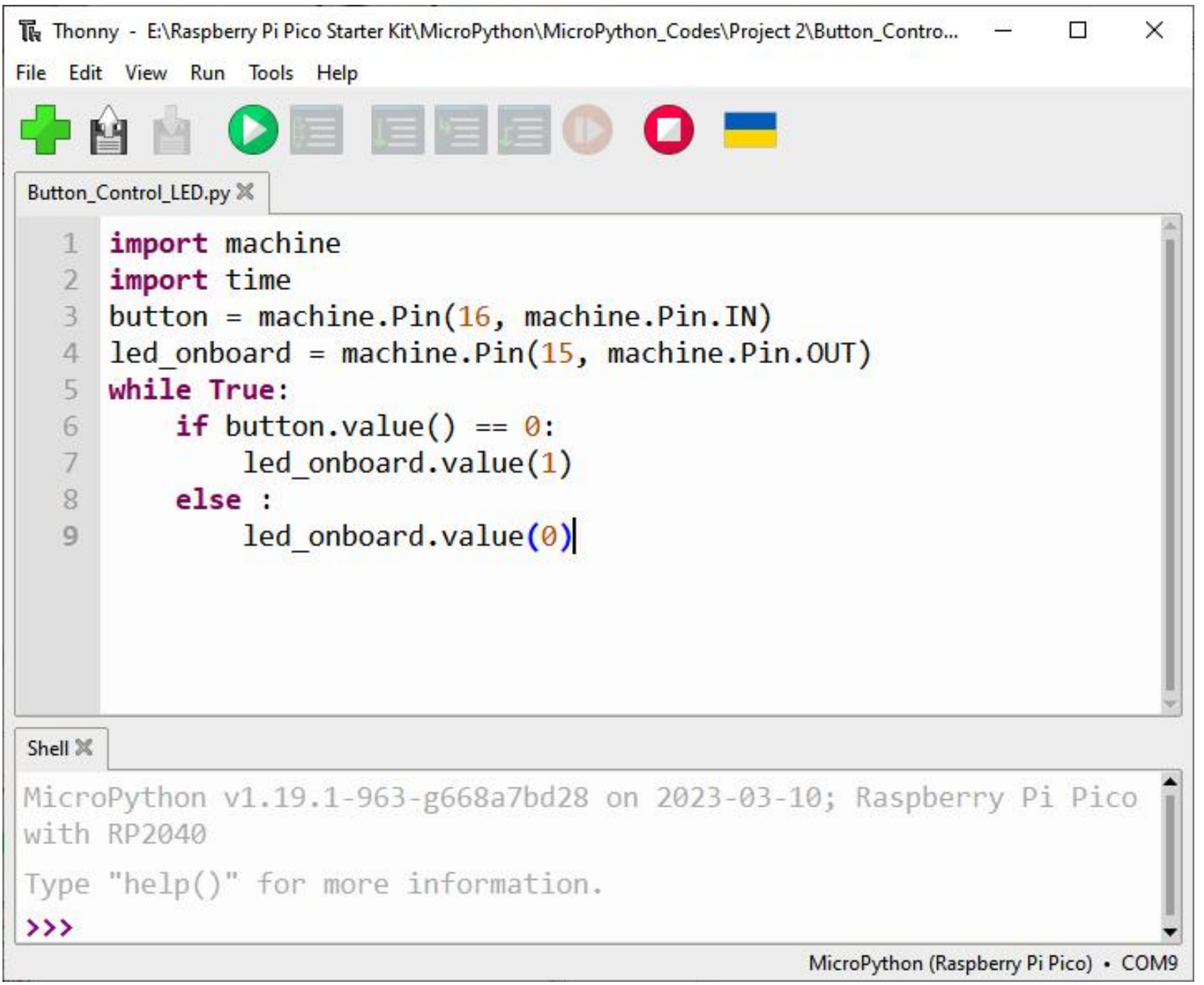
Click the File>>open icon to open Project 2 \ Button_Control_LED.py file.

If you have downloaded the tutorial. You can find the .py file.

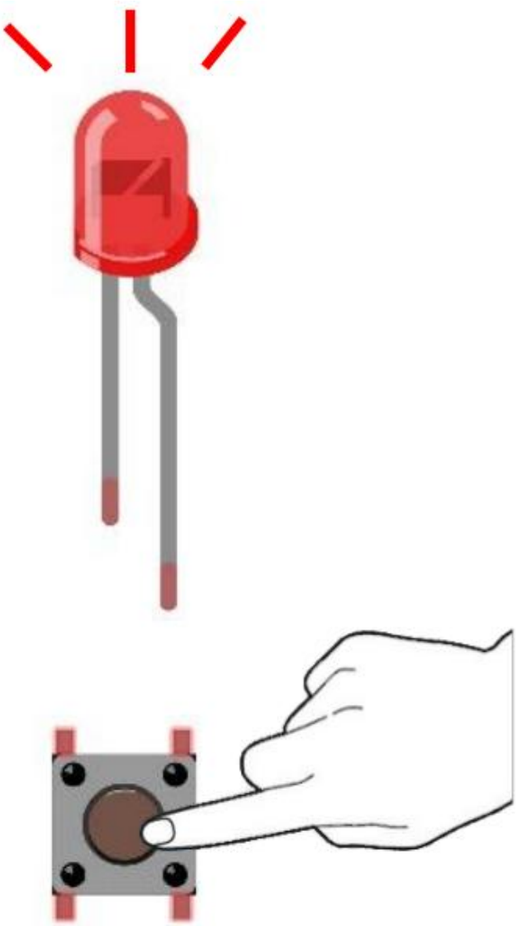
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

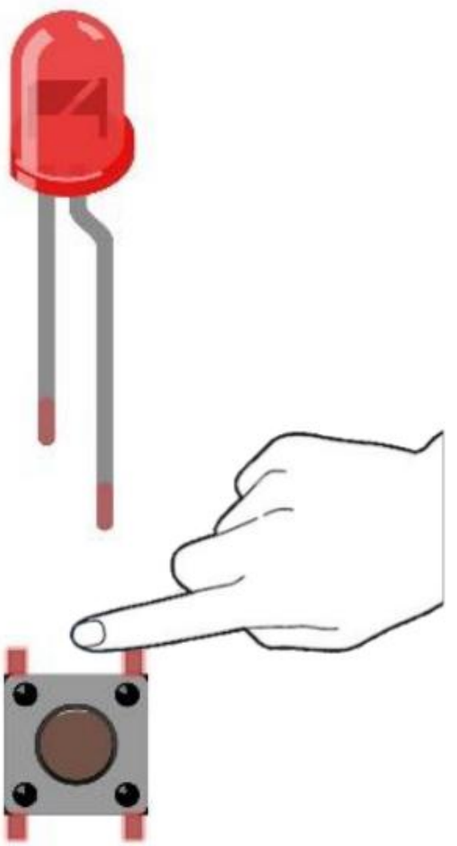
Note:

Don’t forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click “Run current script”  shown in the box of the above illustration, press the push button switch, LED turns ON; release the switch, LED turns OFF. Press Ctrl+C or click “Stop/Restart backend” to exit program.





Project 3 PWM Control LED

So far, we have used only two output signals: high level and low level (or called 1 & 0, ON & OFF), which is called **digital output**. However, in actual use, many devices do not simply ON/OFF to work, for example, adjusting the speed of the motor, adjusting the brightness of the desk lamp, and so on. In the past, a slider that can adjust the resistance was used to achieve this goal, but this is always unreliable and inefficient. Therefore, Pulse width modulation (PWM) has emerged as a feasible solution to such complex problems.

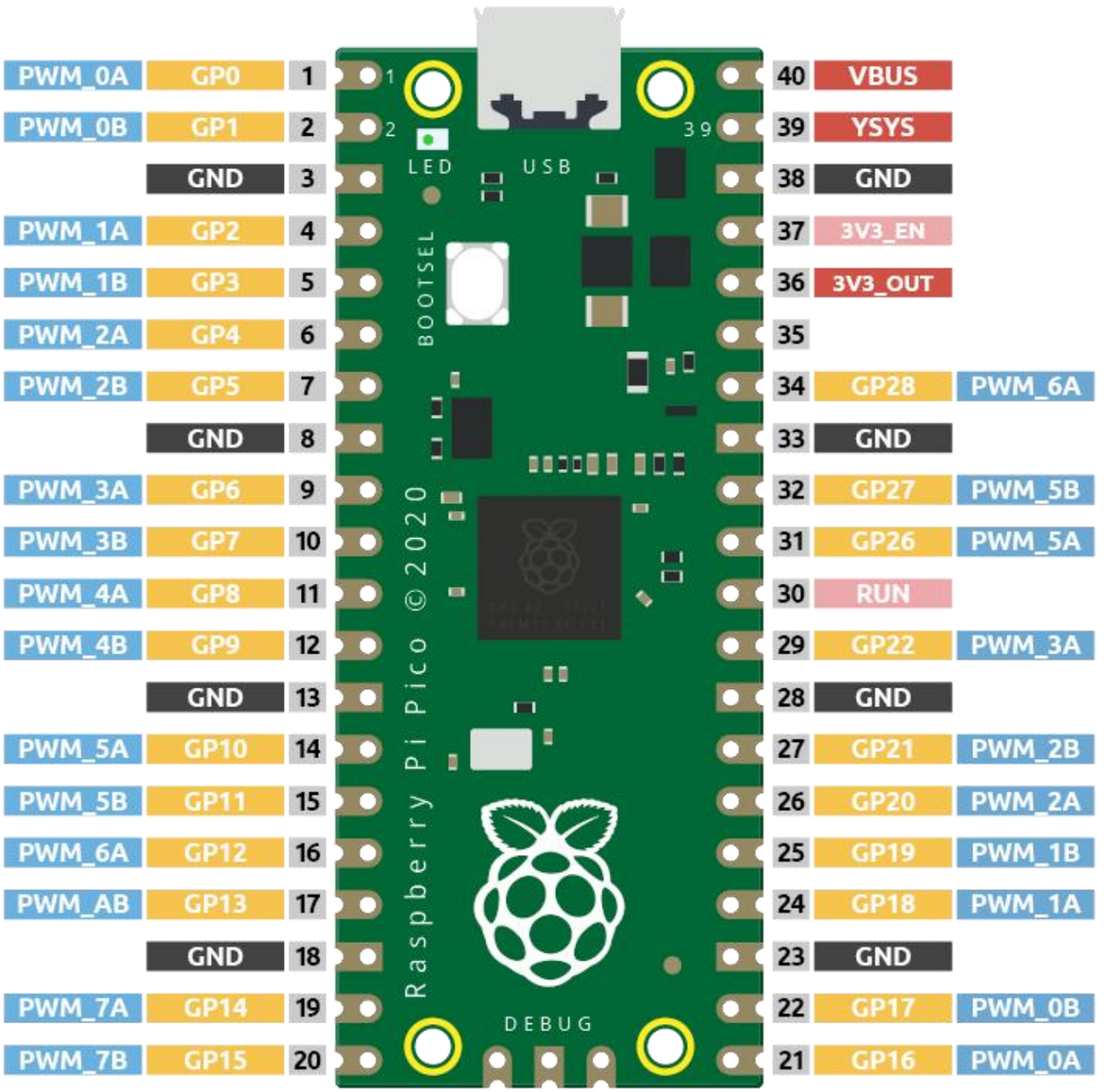
PWM signal

A digital output composed of a high level and a low level is called a pulse. The pulse width of these pins can be adjusted by changing the ON/OFF speed.Common processors cannot directly output analog signals. PWM technology makes it very convenient to achieve this conversion **(translation of digital to analog signals)**

Simply put, when we are in a short period (such as 20ms, most people's visual retention time), Let the LED turn on, turn off, and turn on again, we won't see it has been turned off, but the brightness of the light will be slightly weaker. During this period, the more time the LED is turned on, the higher the brightness of the LED. In other words, in the cycle, the wider the pulse, the greater the "electric signal strength" output by the microcontroller. This is how PWM controls LED brightness (or motor speed).

- Pulse-width modulation - Wikipedia

There are some points to pay attention to when Pico uses PWM. Let's take a look at this picture.

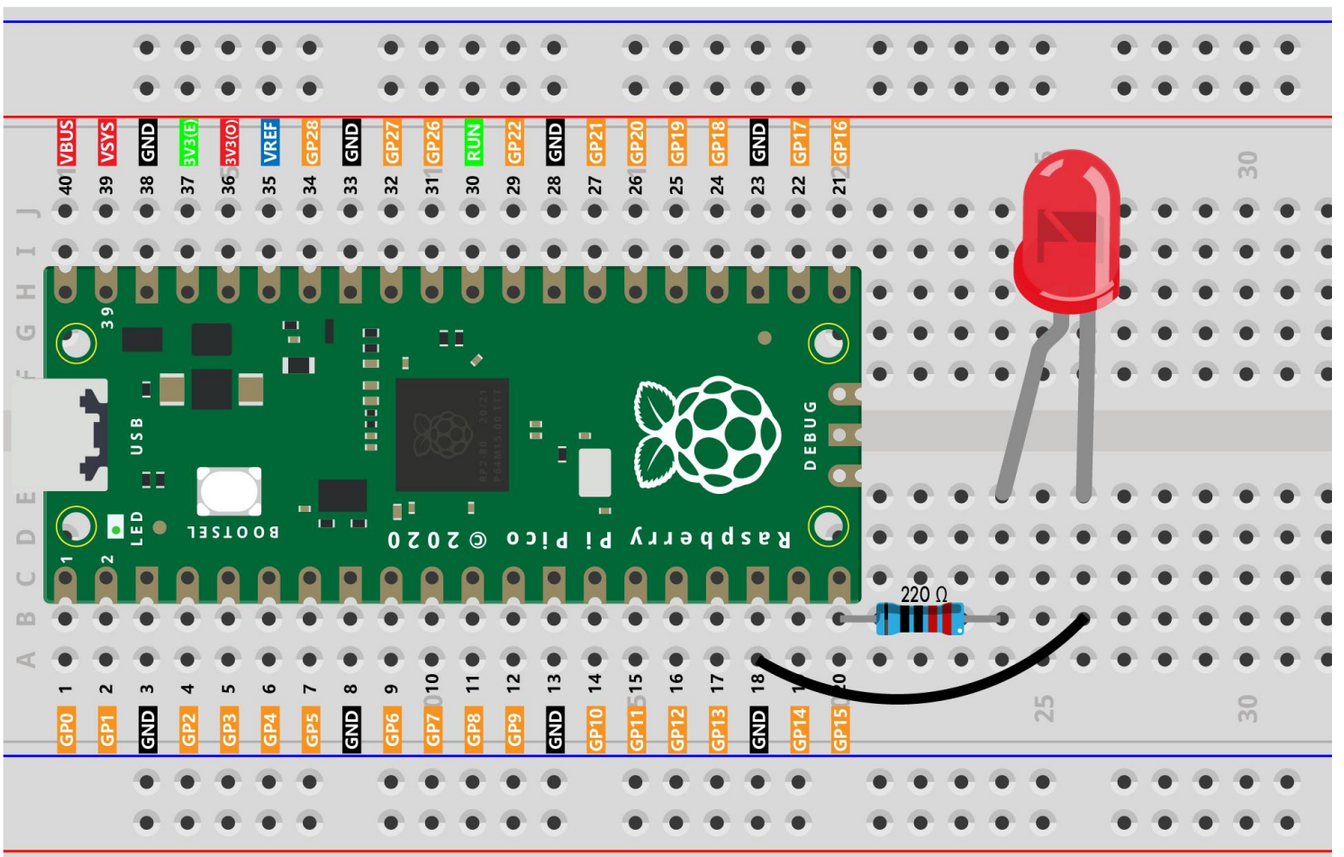


Each GPIO pin of Pico supports PWM, but it actually has a total of 16 independent **PWM outputs** (instead of 30), distributed between GP0 to GP15 on the left, and the PWM output of the right GPIO is equivalent to the left copy.

What we need to pay attention to is to avoid setting the same PWM channel for different purposes during programming. (For example, GP0 and GP16 are both PWM_0A)

After understanding this knowledge, let us try to achieve the effect of Fading LED.

Wiring



1. Here we use the GP15 pin of the Pico board.
2. Connect one end (either end) of the 220 ohm resistor to GP15, and insert the other end into the free row of the breadboard.
3. Insert the anode lead of the LED into the same row as the end of the 220Ω resistor, and connect the cathode lead across the middle gap of the breadboard to the same row.
4. Connect the LED cathode to the negative power bus of the breadboard.
5. Connect the negative power bus to the GND pin of Pico.

Note

The color ring of the 220 ohm resistor is red, red, black, black and brown.

Code

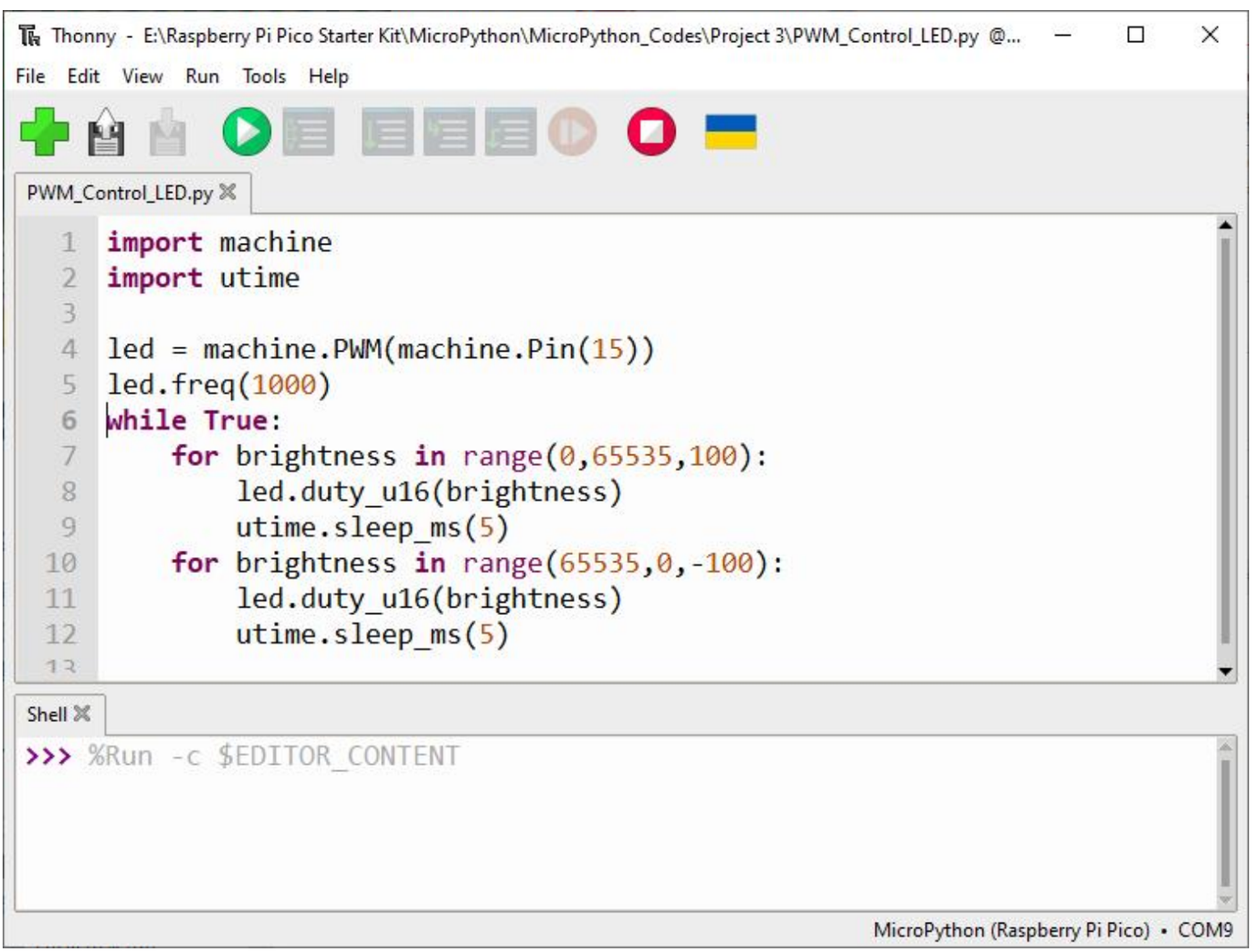
Click the File>>open icon to open Project 3 \ PWM_Control_LED.py file.

If you have downloaded the tutorial. You can find the .py file.

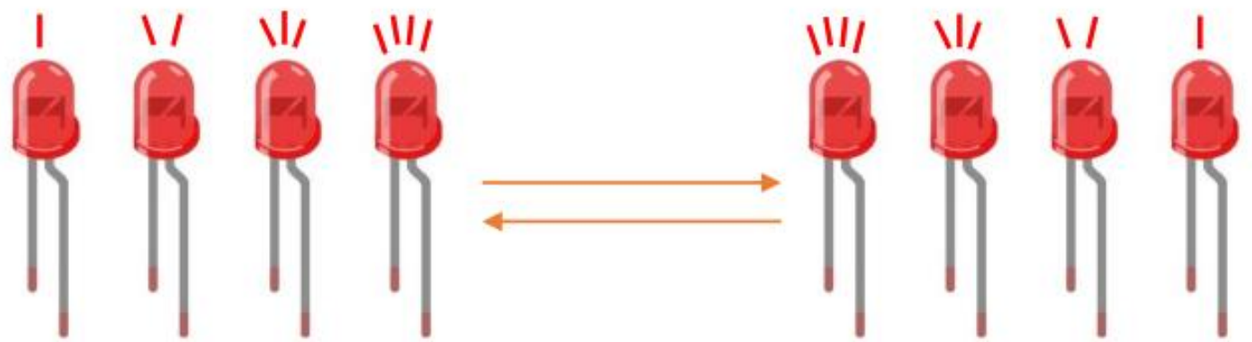
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

Don't forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click "Run current script"  , and you'll see that LED is turned from ON to OFF and then back from OFF to ON gradually like breathing. Press Ctrl+C or click "Stop/Restart backend" to exit program.



How it works?

Here, we change the brightness of the LED by changing the duty cycle of the GP15's PWM output. Let's take a look at these lines.

```
import machine
import utime
led = machine.PWM(machine.Pin(15))
led.freq(1000)
While True:
    for brightness in range(0,65535,100):
        led.duty_u16(brightness)
        utime.sleep_ms(5)
    for brightness in range(65535,0,-100):
        led.duty_u16(brightness)
        utime.sleep_ms(5)
```

- led = machine.PWM(machine.Pin(15)) sets the GP15 pin as PWM output.
- The line led.freq(1000) is used to set the PWM frequency, here it is set to 1000Hz, which means 1ms (1/1000) is a cycle. The PWM frequency can be adjusted, for example, the steering wheel needs to work at 50Hz, the passive buzzer can change the tone by changing the PWM frequency. However, there is no limit when using LEDs alone, we set it to 1000Hz.
- The led.duty_u16() line is used to set the duty cycle, which is a 16-bit interger(2^16=65536). When we assign a 0 to this function, the duty cycle is 0%, and each cycle has 0% of the time to output a high level, in other words, turn off all pulses. When the value is 65535, the duty cycle is 100%, that is, the complete pulse is turned on, and the result is equal to '1' as a digital output. If it is 32768, it will turn on half a pulse, and the brightness of the LED will be half of that when it is fully turned on.

Project 4 RGB LED

As we know, light can be superimposed. For example, mix blue light and green light give cyan light, red light and green light give yellow light. This is called “The additive method of color mixing”.

- Additive color - Wikipedia

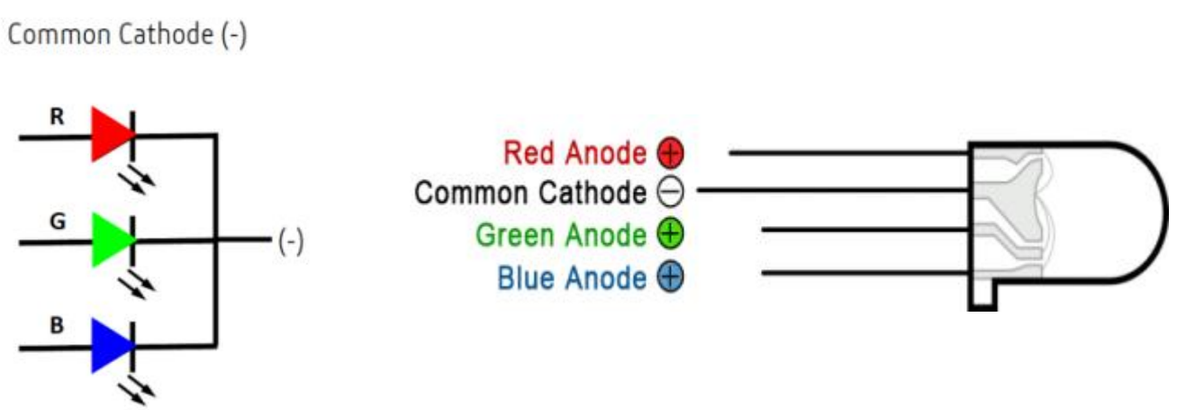
Based on this method, we can use the three primary colors to mix the visible light of any color according to different specific gravity. For example, orange can be produced by more red and less green.

In this chapter, we will use RGB LED to explore the mystery of additive color mixing!

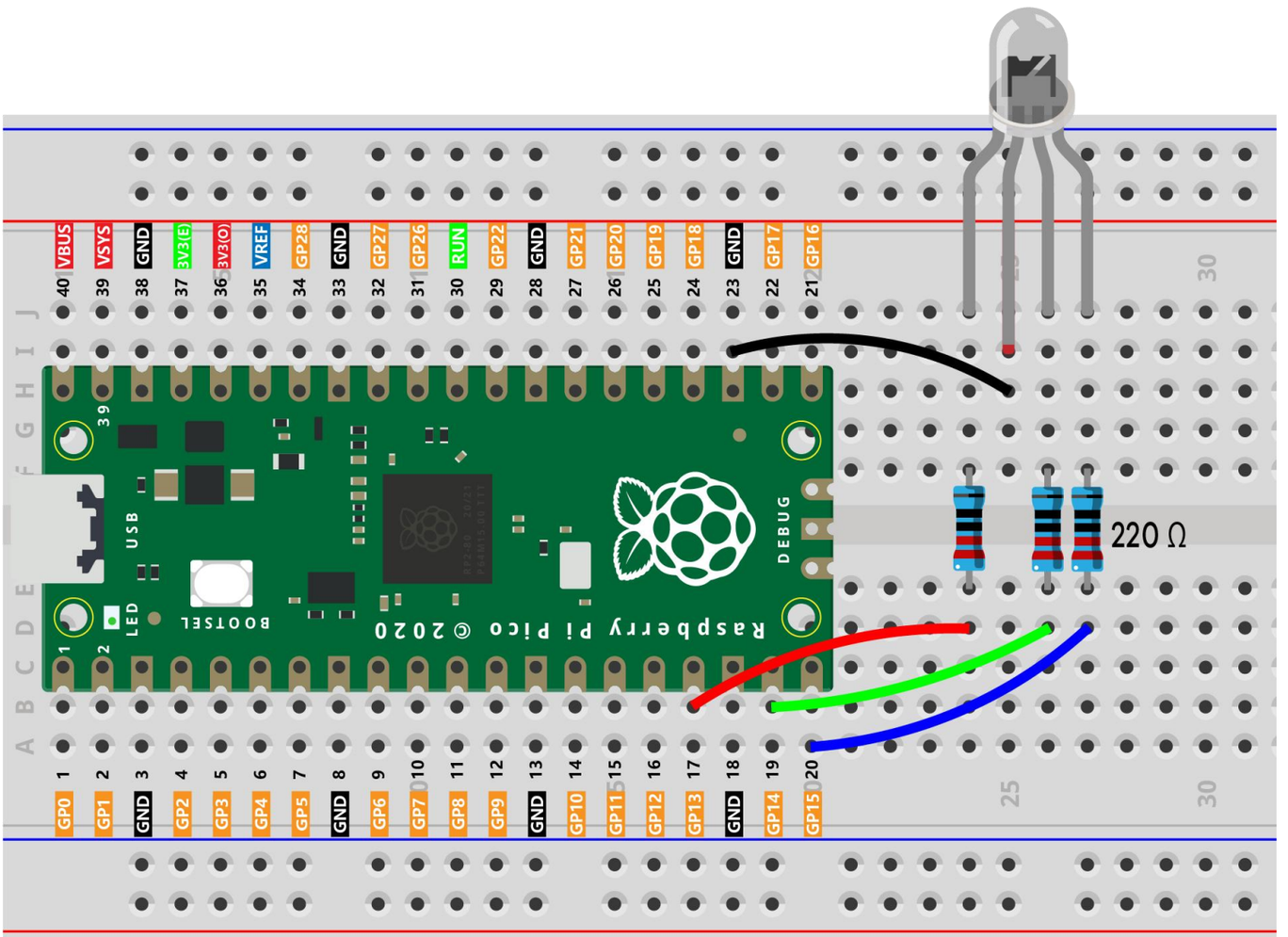
RGB LED is equivalent to encapsulating Red LED, Green LED, Blue LED under one lamp cap, and the three LEDs share one cathode pin. Since the electric signal is provided for each anode pin, the light of the corresponding color can be displayed. By changing the electrical signal intensity of each anode, it can be made to produce various colors.

Wiring

Put the RGB LED flat on the table, we can see that it has 4 leads of different lengths. Find the longest one (GND) and turn it sideways to the left. Now, the order of the four leads is Red, GND, Green, Blue from left to right.



Tip:Learn more about RGB LED



Code

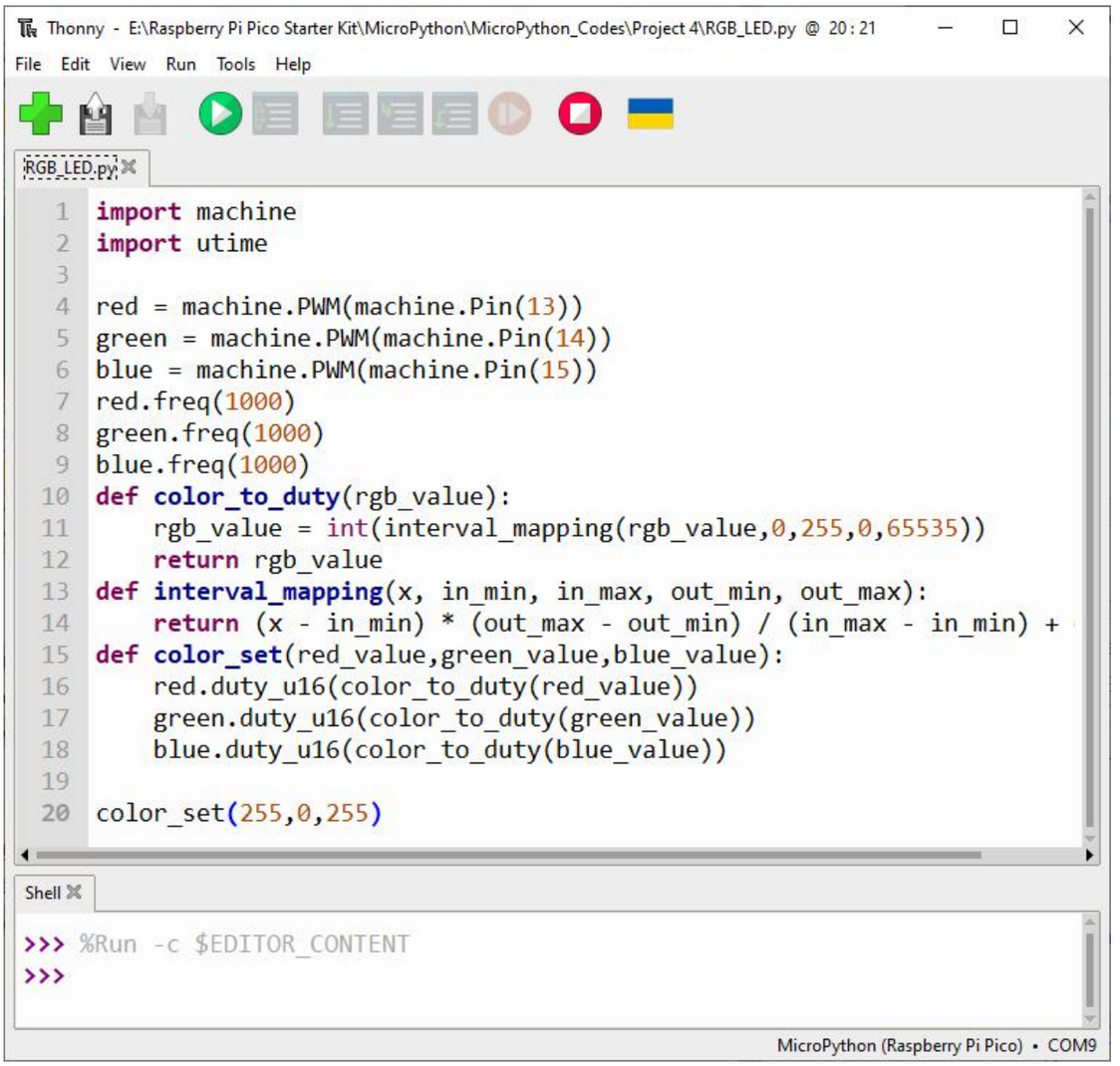
Click the File>>open icon to open Project 4 \ RGB_LED.py file.

If you have downloaded the tutorial. You can find the .py file.

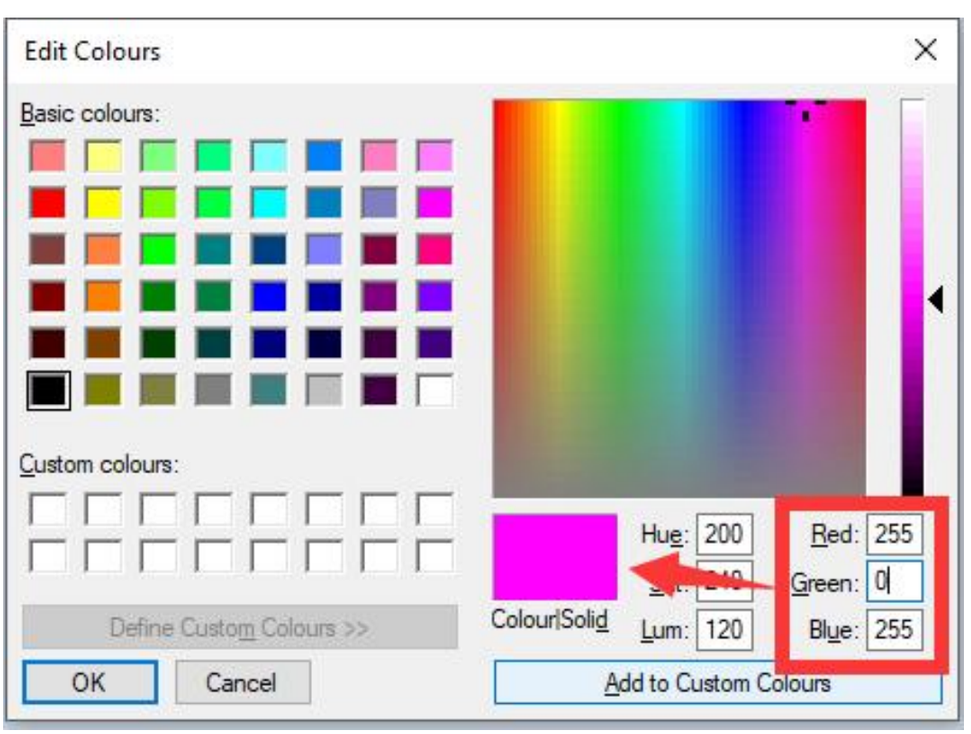
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

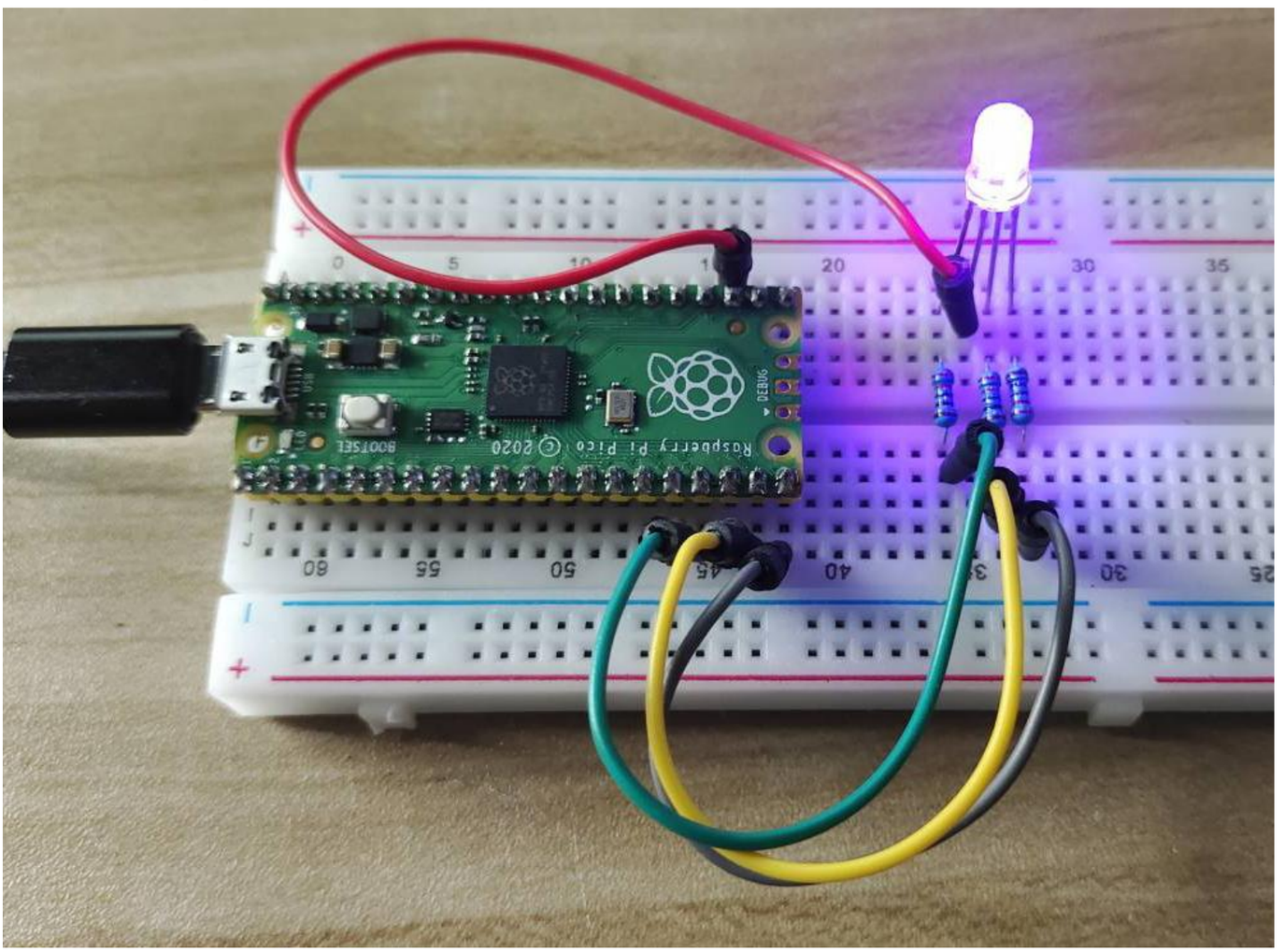
Don’t forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Here, we can choose our favorite color in drawing software (such as paint) and display it with RGB LED.



Write the RGB value into color_set(255,0,255),Click “Run current script” . you will be able to see the RGB light up the colors-Purple.



Project 5 Custom Tone

In this project I use a passive buzzer to make different note sounds.

Component Knowledge

Buzzer

the passive buzzer also uses the phenomenon of electromagnetic induction to work. The difference is that a passive buzzer does not have oscillating source, so it will not beep if DC signals are used. But this allows the passive buzzer to adjust its own oscillation frequency and can emit different notes such as “doh, re, mi, fa, sol, la, ti”.

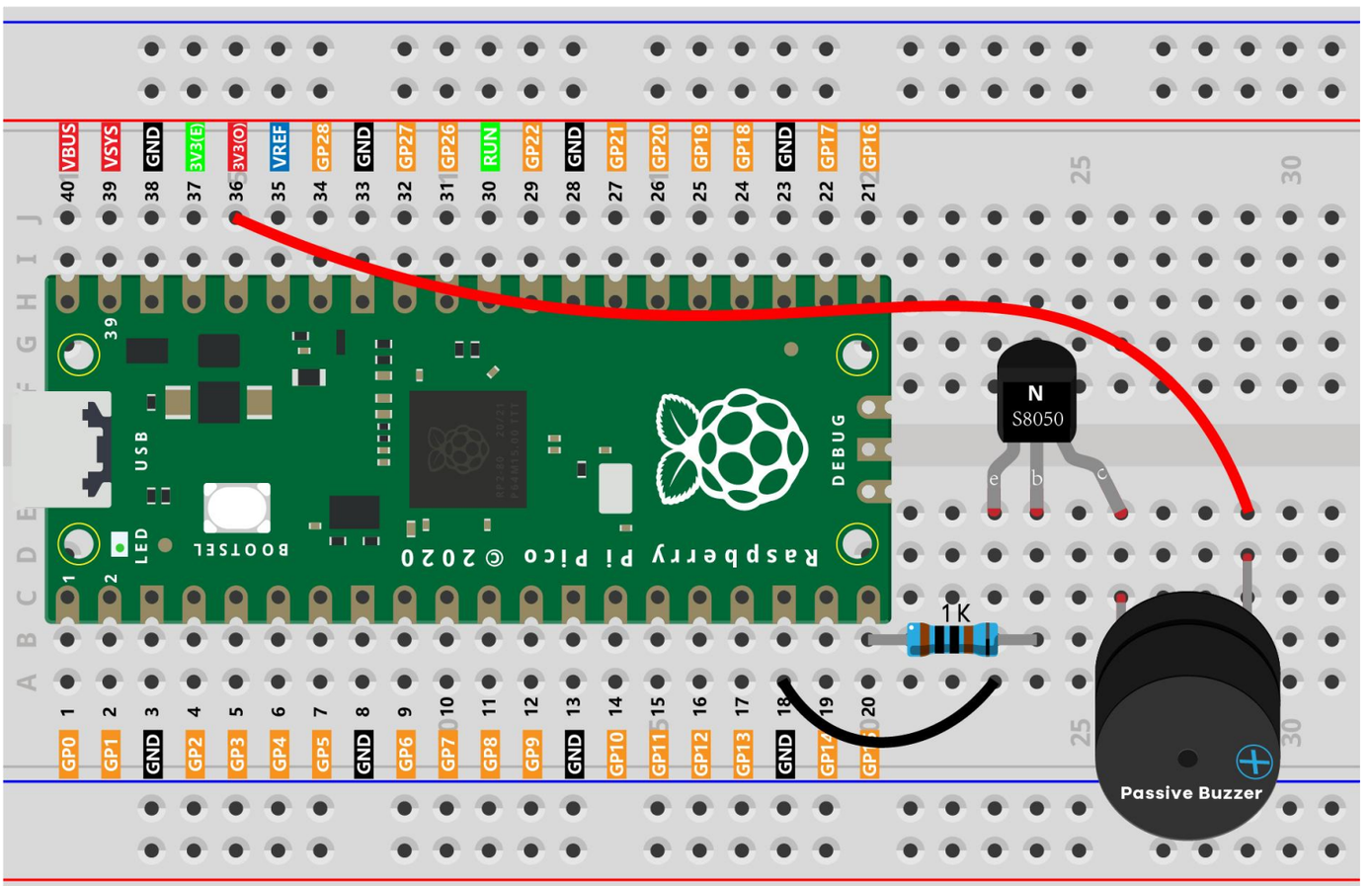
Tip:Learn more about Buzzer

Transistor

In simple words, some components use high-current (such as Buzzer). If the power is directly supplied from the GPIO of the microcontroller, the power may be insufficient or the microcontroller may be damaged. Then, the transistor has played a “dam” role here. Transistor receives the weak electrical signal from the GPIO pin to control the turn-on and turn-off of the large current (from VCC to GND). In this way, high-current components can be driven and the microcontroller can be protected.

Tip:Learn more about Transistor

Wiring



Note:

①Two buzzers are included in the kit, we use the one with exposed PCB behind.

②The buzzer needs a transistor to work, and here we use S8050.

③The 1K resistor between the NPN transistor and GP15 is used for current limiting when the transistor is energized.

Code

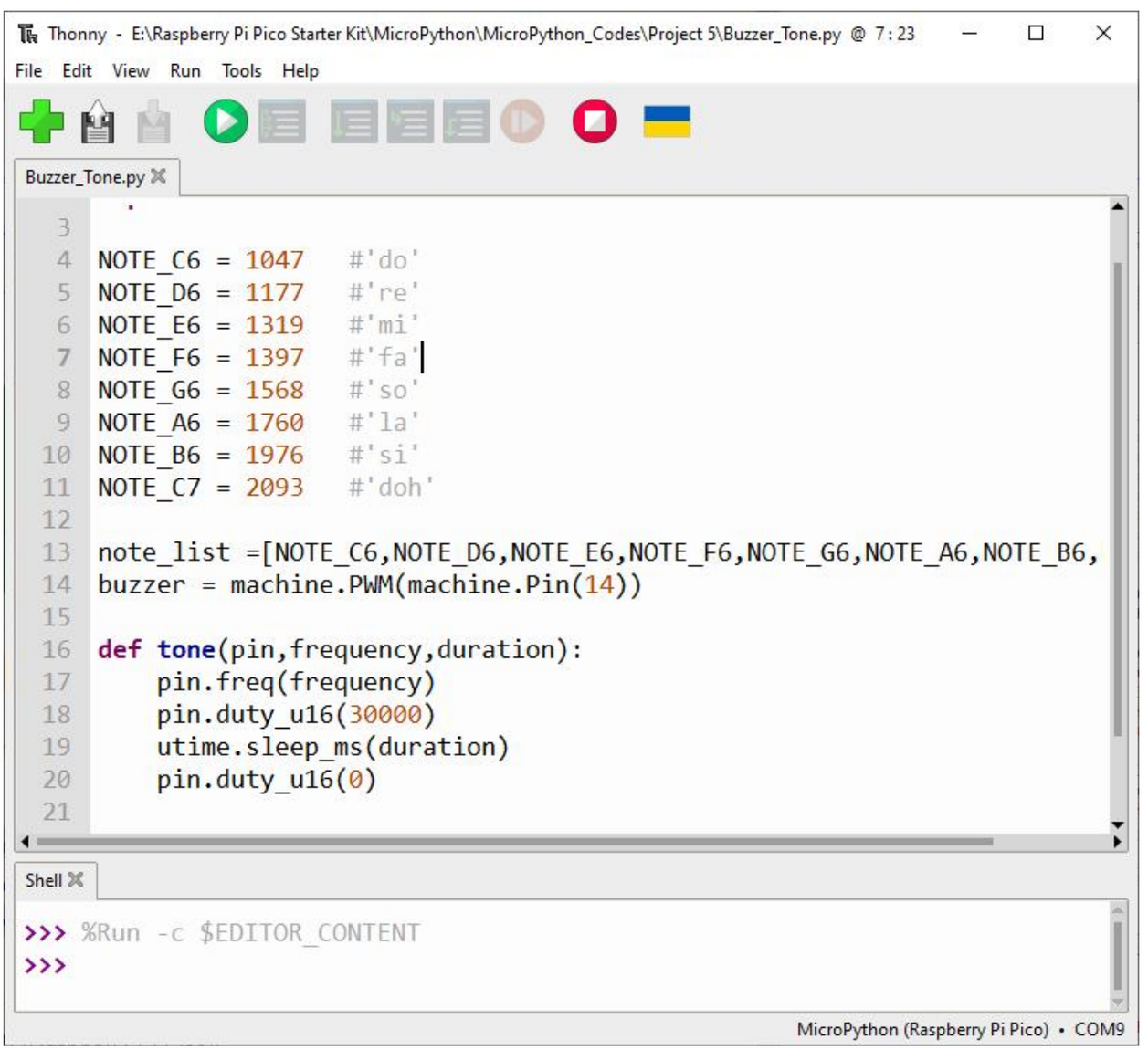
Click the File>>open icon to open Project 5 \ Buzzer_Tone.py file.

If you have downloaded the tutorial. You can find the .py file.

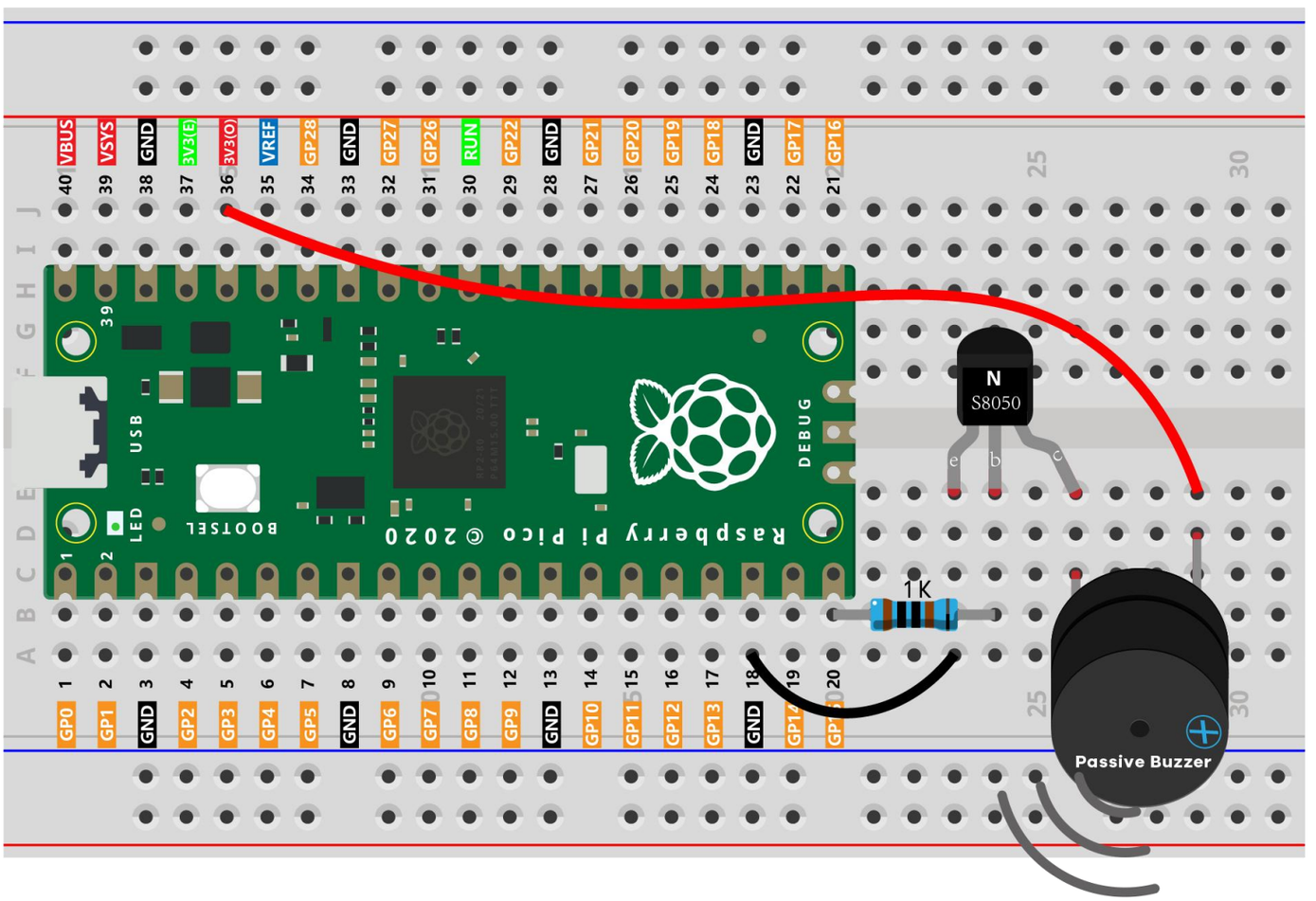
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

Don't forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click “Run current script”  , and Buzzer will play note:”do, re, mi, fa, so, la, si, doh”.



How it works?

If the passive buzzer given a digital signal, it can only keep pushing the diaphragm without producing sound.

Therefore, we use the tone() function to generate the PWM signal to make the passive buzzer sound.

This function has three parameters:

- **pin**, the GPIO pin that controls the buzzer.
- **frequency**, the pitch of the buzzer is determined by the frequency, the higher the frequency, the higher the pitch.
- **Duration**, the duration of the tone.

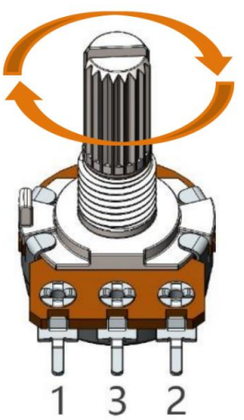
We use the duty_u16() function to set the duty cycle to 30000(about 50%). It can be other numbers, and it only needs to generate a discontinuous electrical signal to oscillate.

Project 6 Analog Input

In the previous projects, we have used the digital input on the Pico. For example, a button can change the pin from low level (off) to high level (on). This is a binary working state.

However, Pico can receive another type of input signal: analog input. It can be in any state from fully closed to fully open, and has a range of possible values. The analog input allows the microcontroller to sense the light intensity, sound intensity, temperature, humidity, etc. of the physical world.

In this project, we try to read the analog value of potentiometer.



Tip:Learn more about potentiometer

Related knowledge

ADC

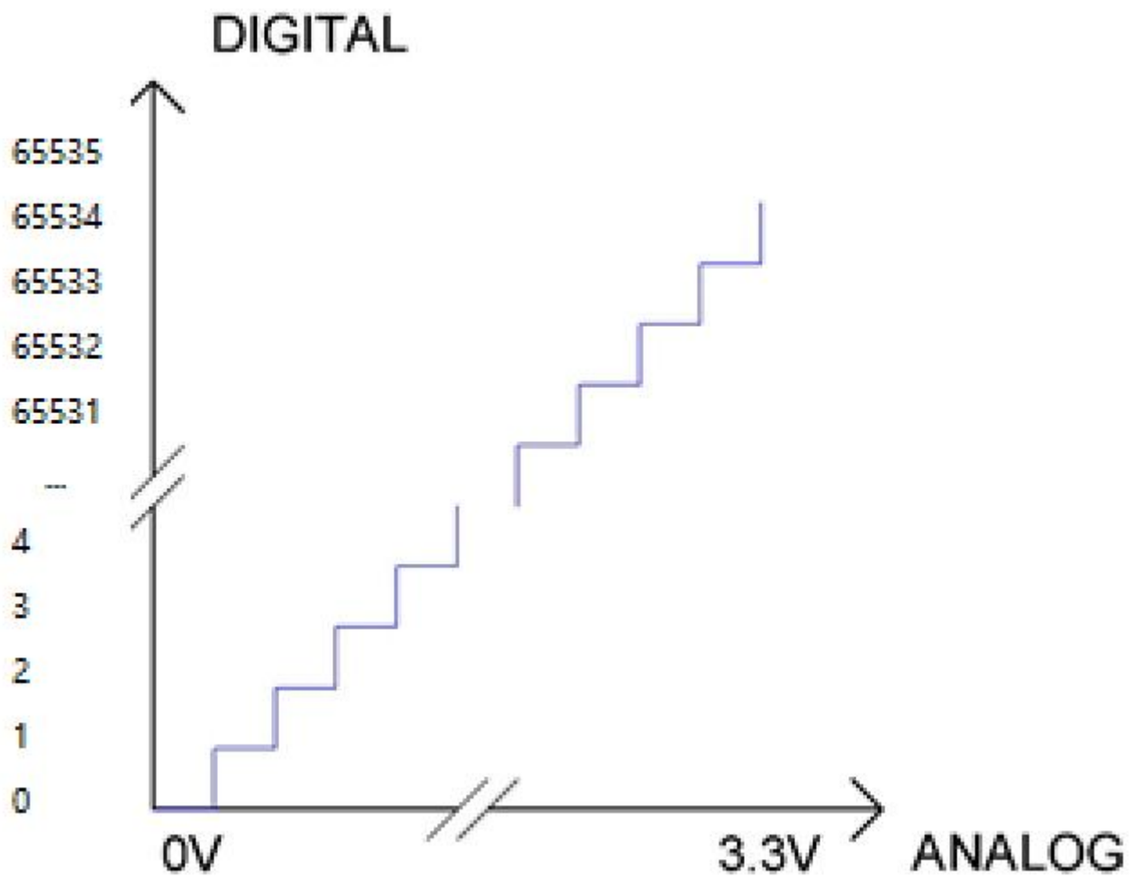
An analog-to-digital converter(ADC) converts a measured analog signal into a digital code. ADC has two key features: resolution and channels.

ADC resolution of Raspberry Pi Pico

Raspberry Pi Pico uses RP2040 chip. With a 12-bit ADC resolution, it can convert any analog signal to digital signal, ranging from 0-4095. For example, if the analog voltage range it measures is 0-3.3V, the ADC can divide it into 4096 equal parts.

However, when using Micropython firmware to call Raspberry Pi Pico ADC, the digital signal it obtains ranges from 0 to 65535. This is because Micropython internally processes Pico's ADC resolution to 16 bits, and the values is also changed to 0-65535, so as to make it the same as the ADC of other Micropython

microcontrollers. The way that ADC converts doesn't change but only the resolution changes. So if the measured analog voltage ranges from 0-3.3V, ADC can devide it into 65536 equal parts.



Subsection 1: the analog in a range of 0V---3.3/65535 V corresponds to digital 0;

Subsection 2: the analog in a range of 3.3/65535 V---2*3.3 /65535 V corresponds to digital 1;

…

Subsection 65535: the analog in range of 65534*3.3/65535 V---65535*3.3 /65535 V corresponds to digital

65534;

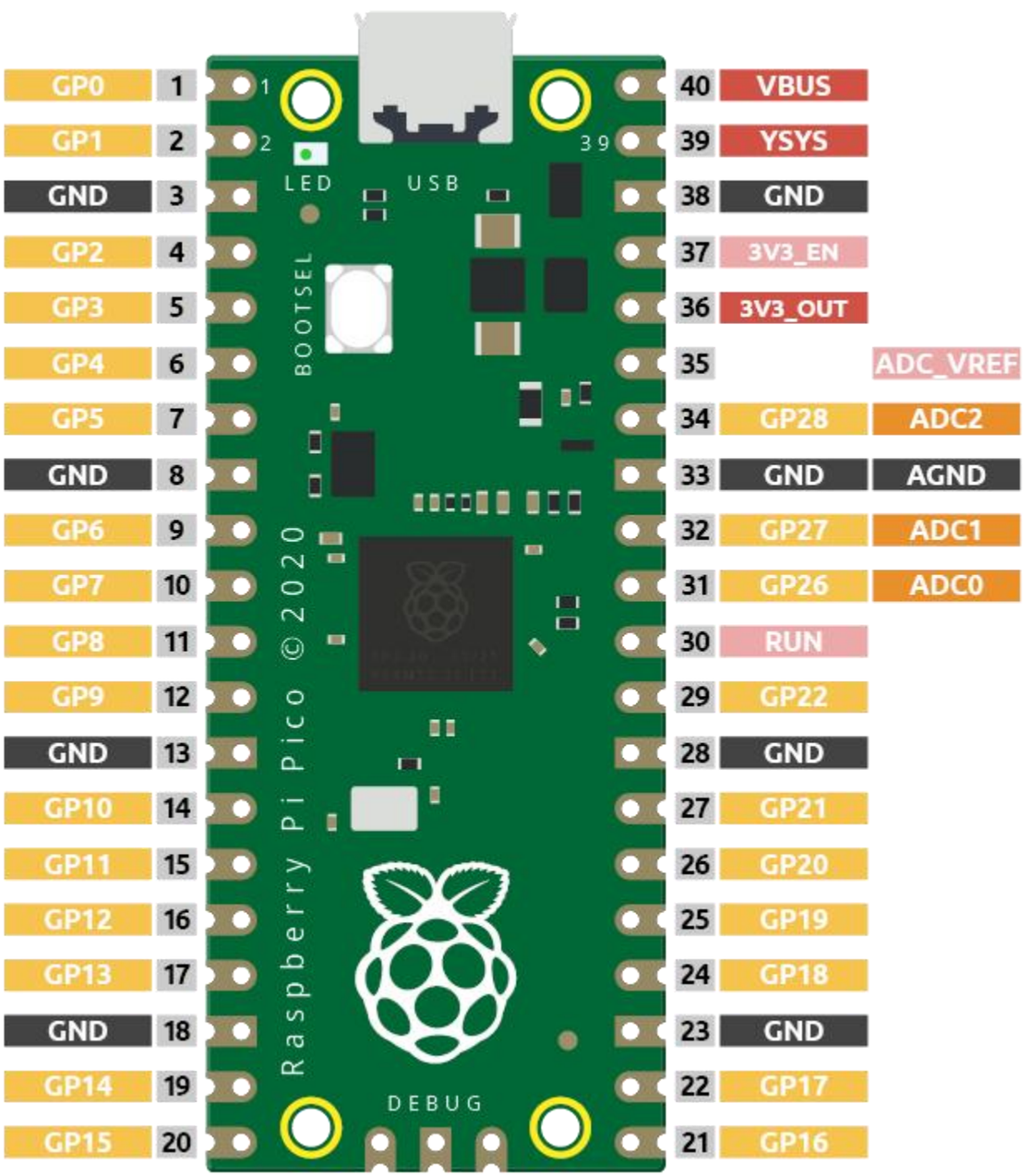
The following analog will be divided accordingly.

The conversion formula is as follows:

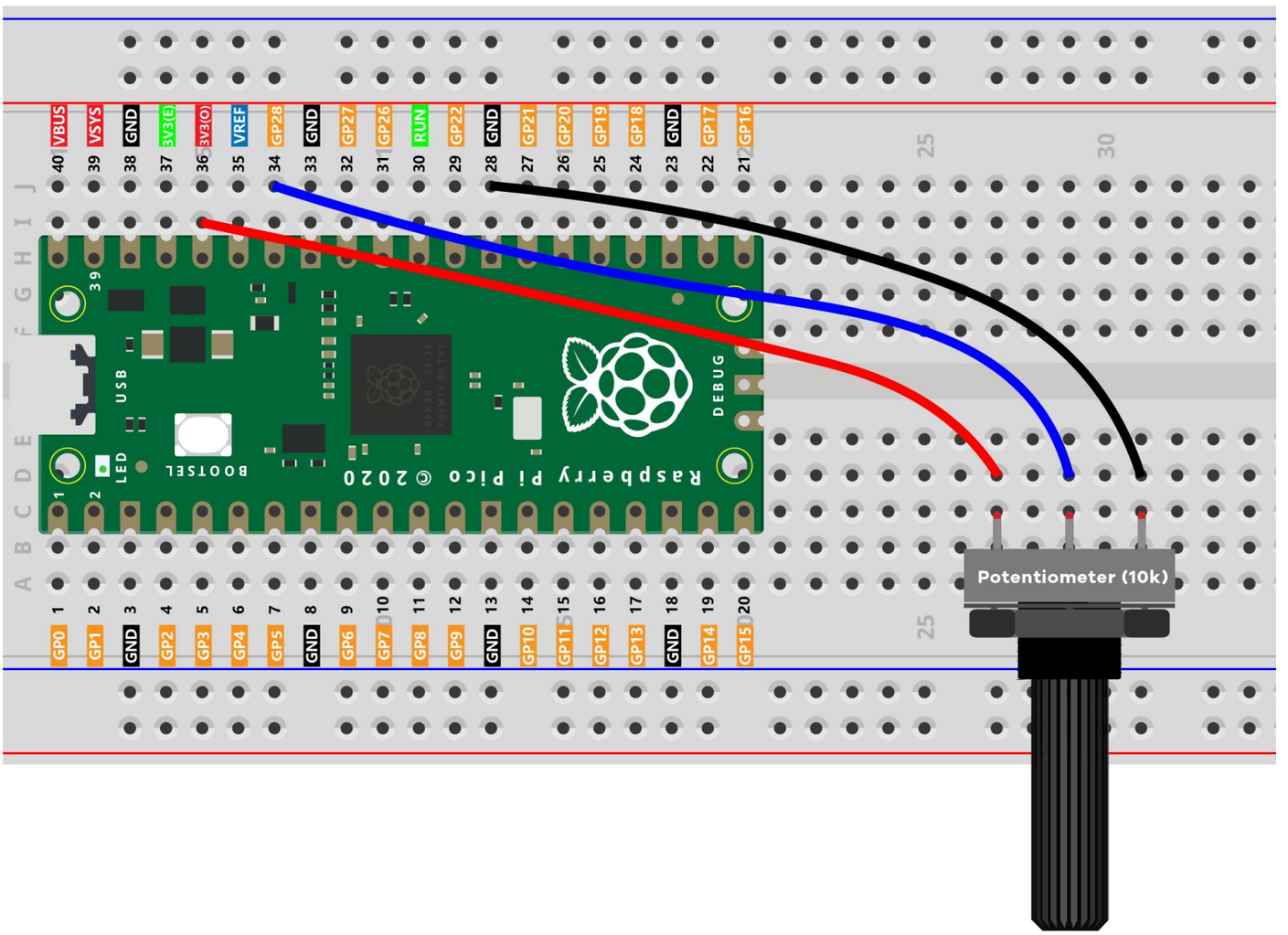
$$ADCValue = \frac{\text{Analog Voltage}}{3.3} * 65535$$

ADC ChannelsRaspberry Pi Pico

Raspberry Pi Pico has 5 ADC channels, which are ADC0(GP26), ADC1(GP27), ADC2(GP28), ADC3(GP29): used to measure VSYS on Pico board, and ADC4, which directly connects to the built-in temperature sensor of RP2040 chip. Therefore, there are only three generic ADC channels that can be directly used, namely, ADC0,ADC1 and ADC2.



Wiring



Code

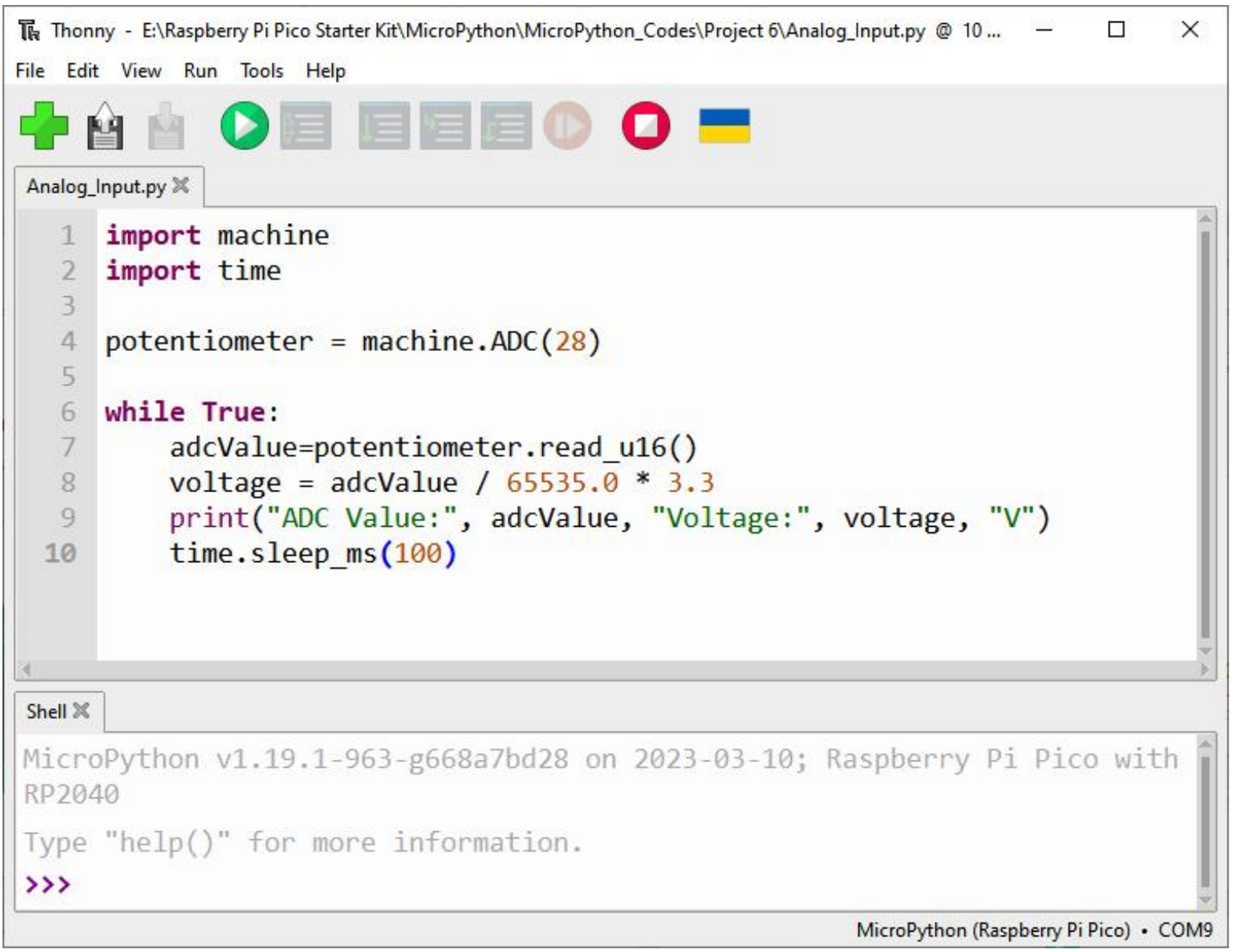
Click the File>>open icon to open Project 6 \ Analog_Input.py file.

If you have downloaded the tutorial. You can find the .py file.

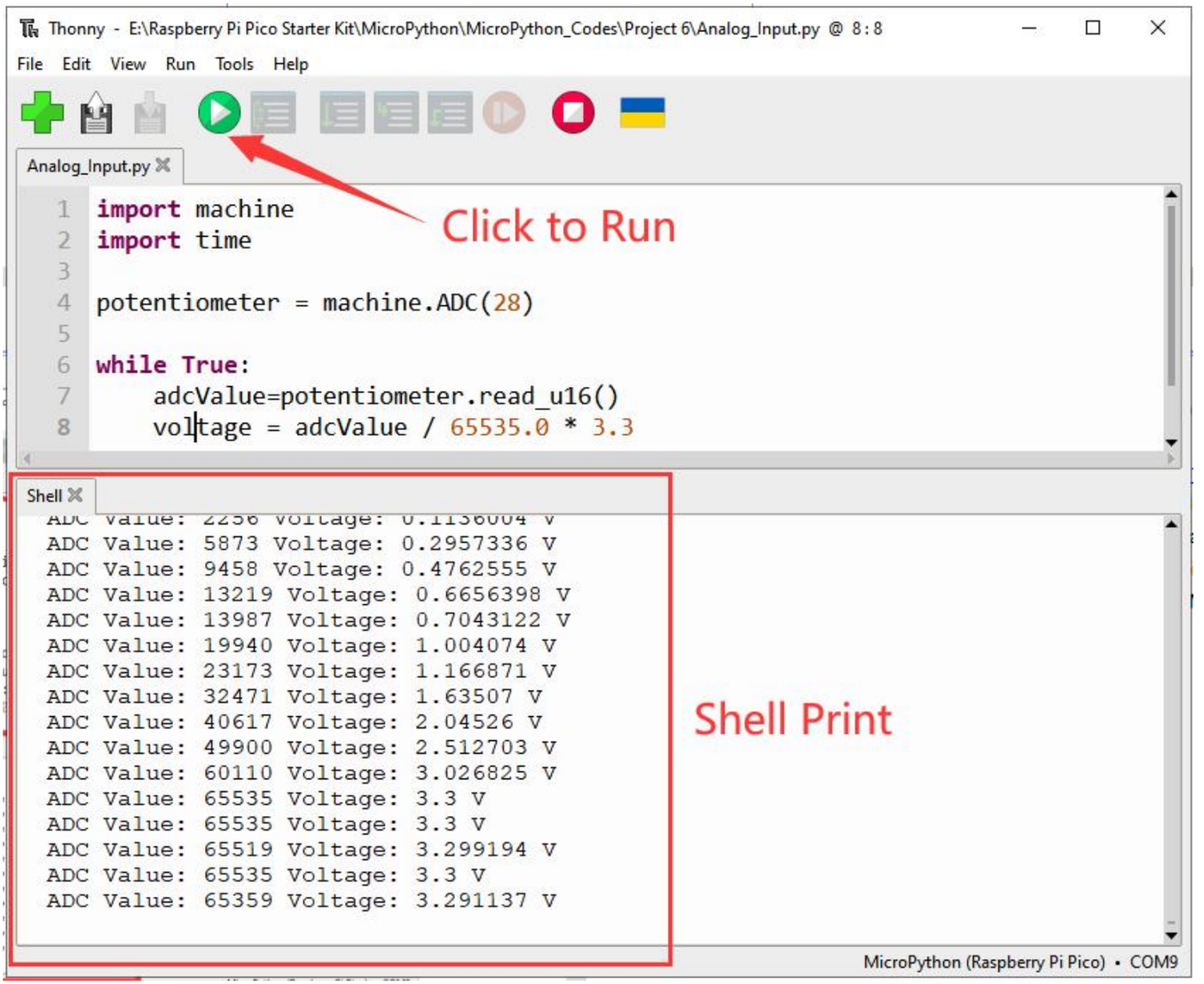
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

Don’t forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click “Run current script”  and observe the message printed in “Shell”. Press Ctrl+C or click “Stop/Restart backend” to exit the program.



How it works?

Before each use of ADC module, please add the statement “import machine” to the top of the python file.

machine.ADC(pin): Create an ADC object associated with the given pin.

pin: Available pins are:GP26,GP27,GP28.

ADC0->machine.ADC(26)

ADC1-> machine.ADC(27)

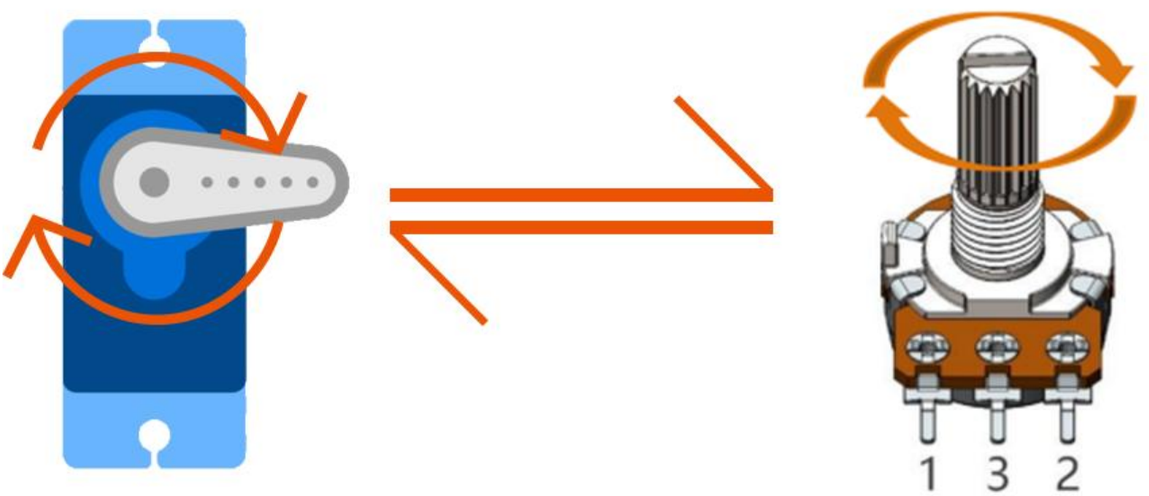
ADC2-> machine.ADC(28)

ADC.read_16(): reads the current ADC value and returns it, with a range of 0-65535.

Project 7 Potentiometer Control Servo

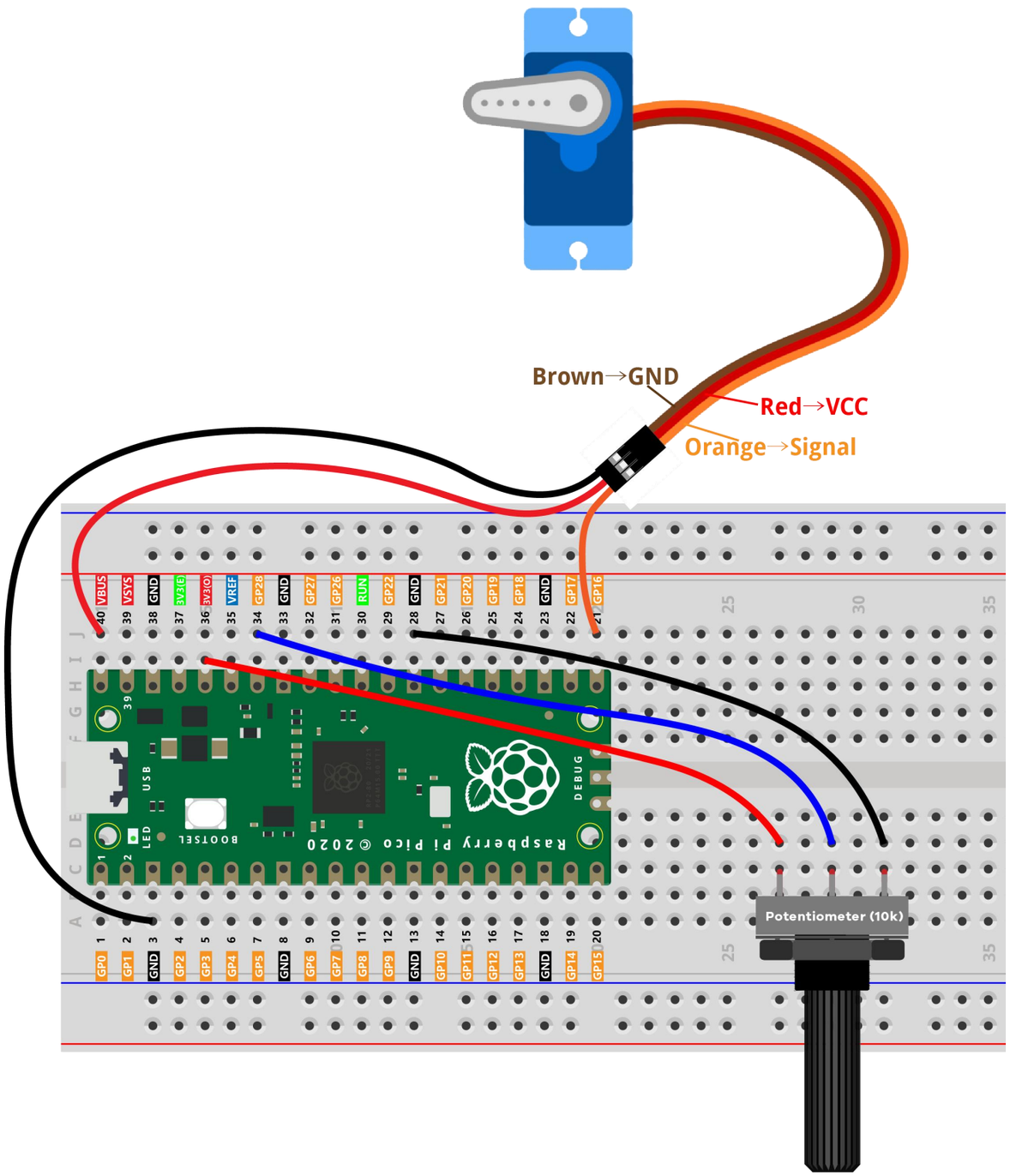
We have learned ADC analog input and PWM analog output concept. This time, we use some new analog input device and PWM output device.

In this project, we will show you how to read analog signal from a potentiometer(adjustable resistor) through a Pico ADC pin(GP28) . We will also connect a Servo motor to GP16 which can generate PWM signal. The servo will rotate its arm when you rotate potentiometer.



Tip:Learn more about potentiometer servo

Wiring



Note:

The working voltage of Servo is 5V. Use the VBUS pin to power it.

Code

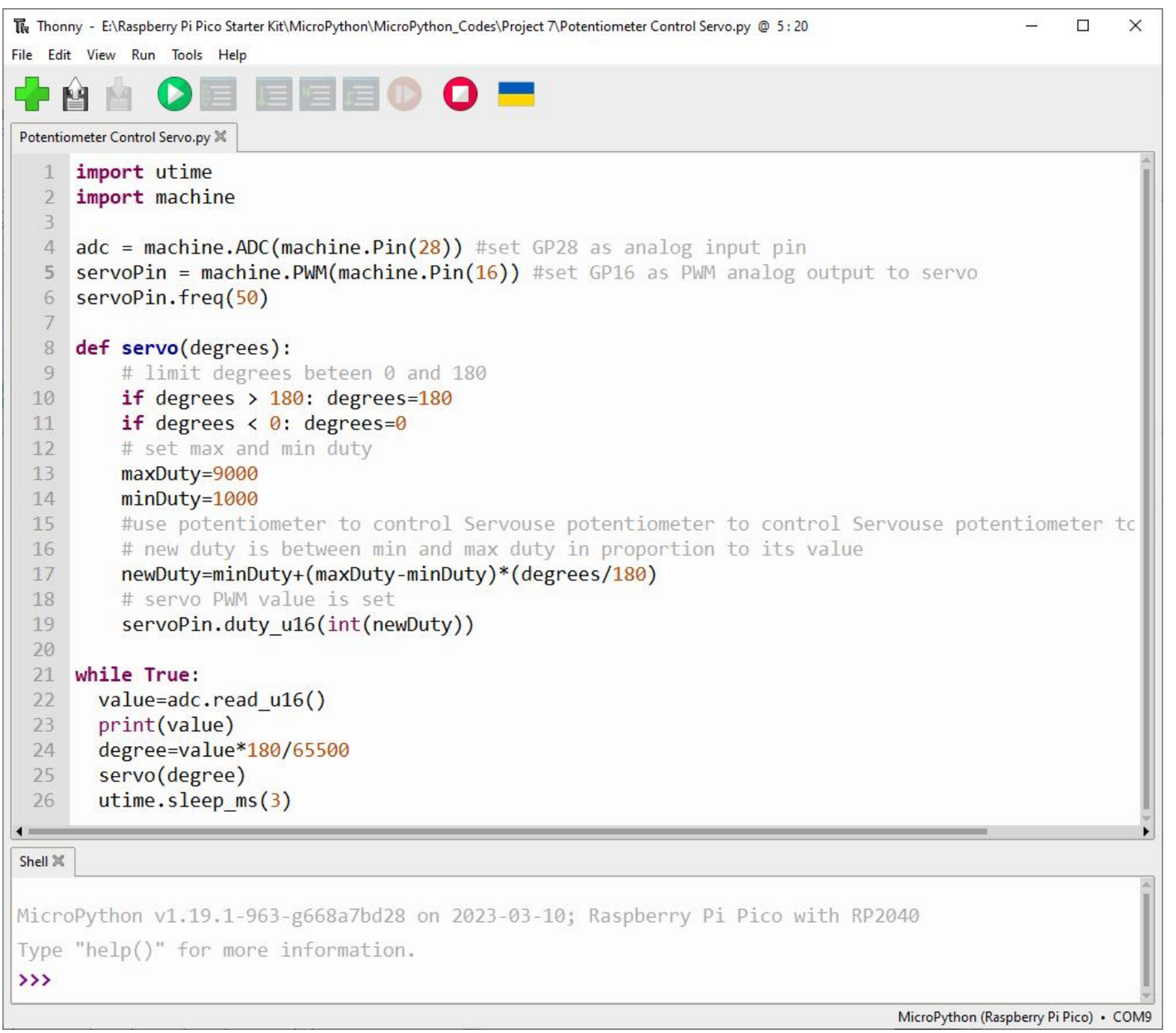
Click the File>>open icon to open Project 7 \ Potentiometer_Control_Servo.py file.

If you have downloaded the tutorial. You can find the .py file.

File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

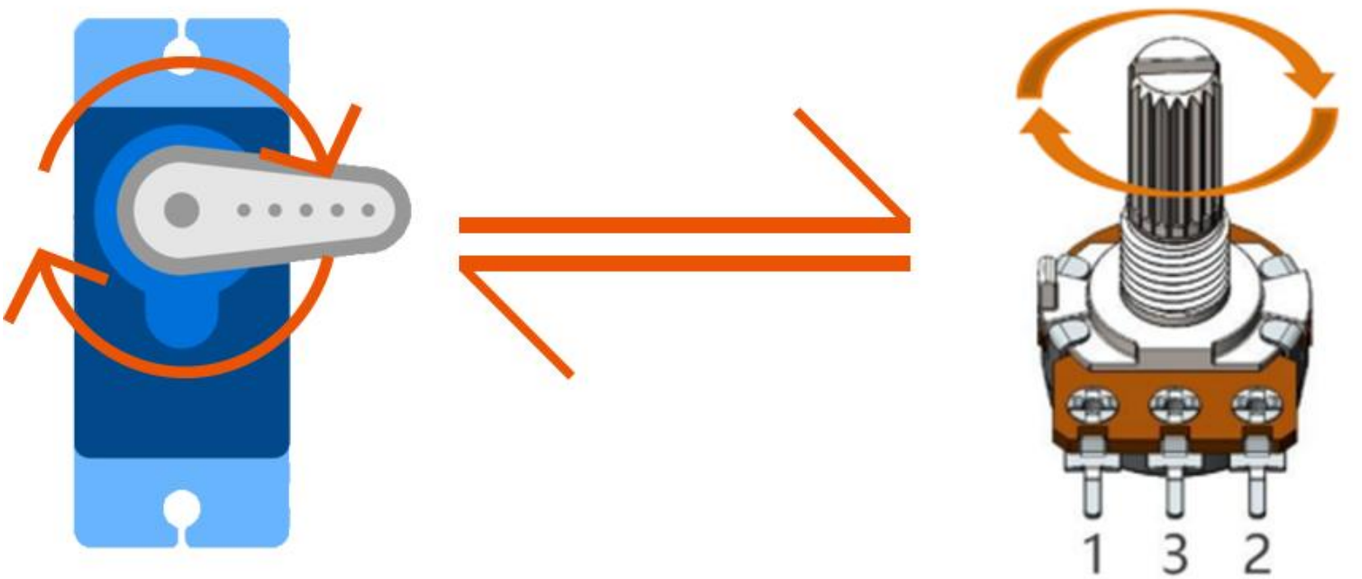
Note:

Don't forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



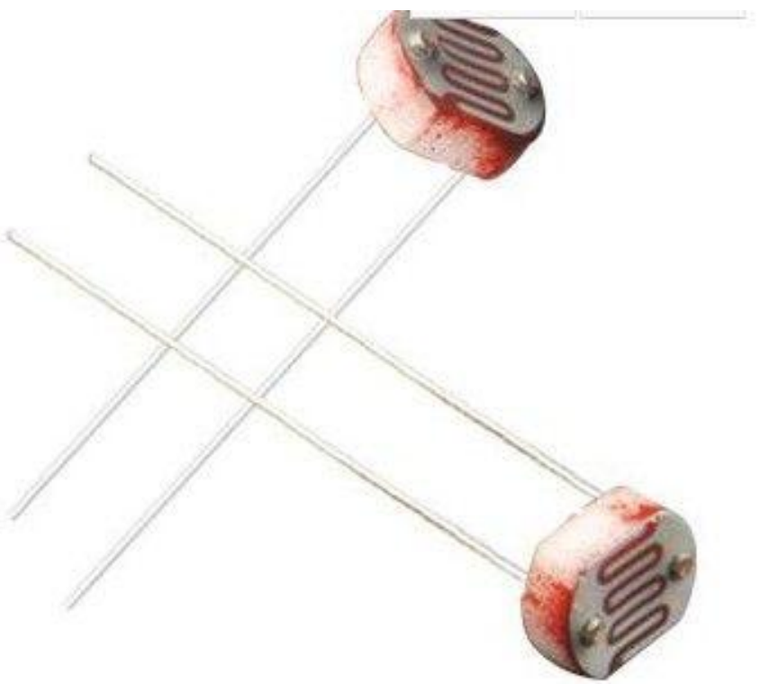
Click “Run current script”  and you can rotate the potentiometer, and you will see the servo arm rotate accordingly.

After the operation is complete,Press Ctrl+C or click “Stop/Restart backend” to exit the program.



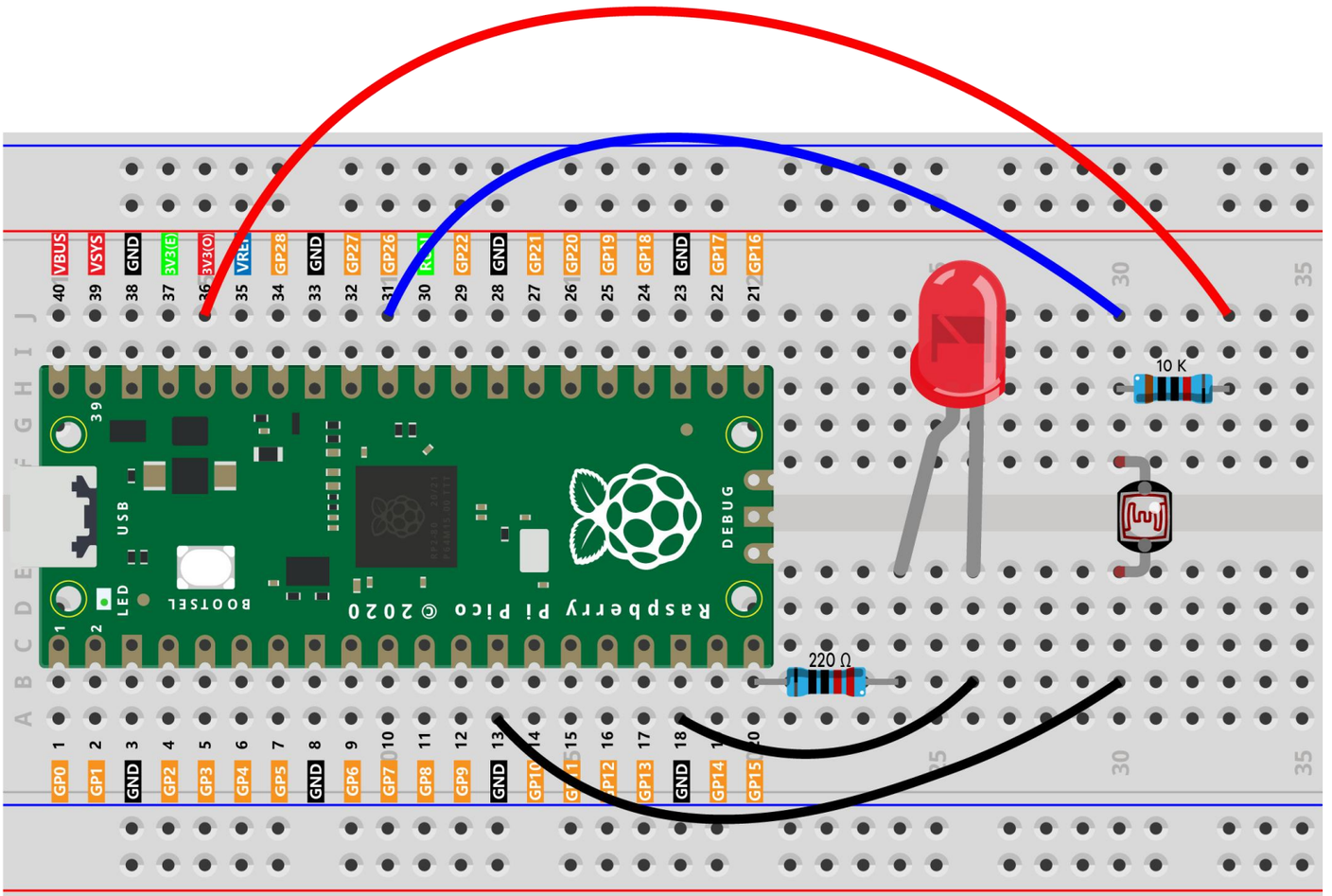
Project 8 Photoresistor Control LED

A Photoresistor is very sensitive to the amount of light present. We can take advantage of the characteristic to make a night lamp with the following function: when the ambient light is less (darker environment) the LED will automatically become brighter to compensate and when the ambient light is greater (brighter environment) the LED will automatically dim to compensate.



Tip:Learn more about Photoresistor

Wiring



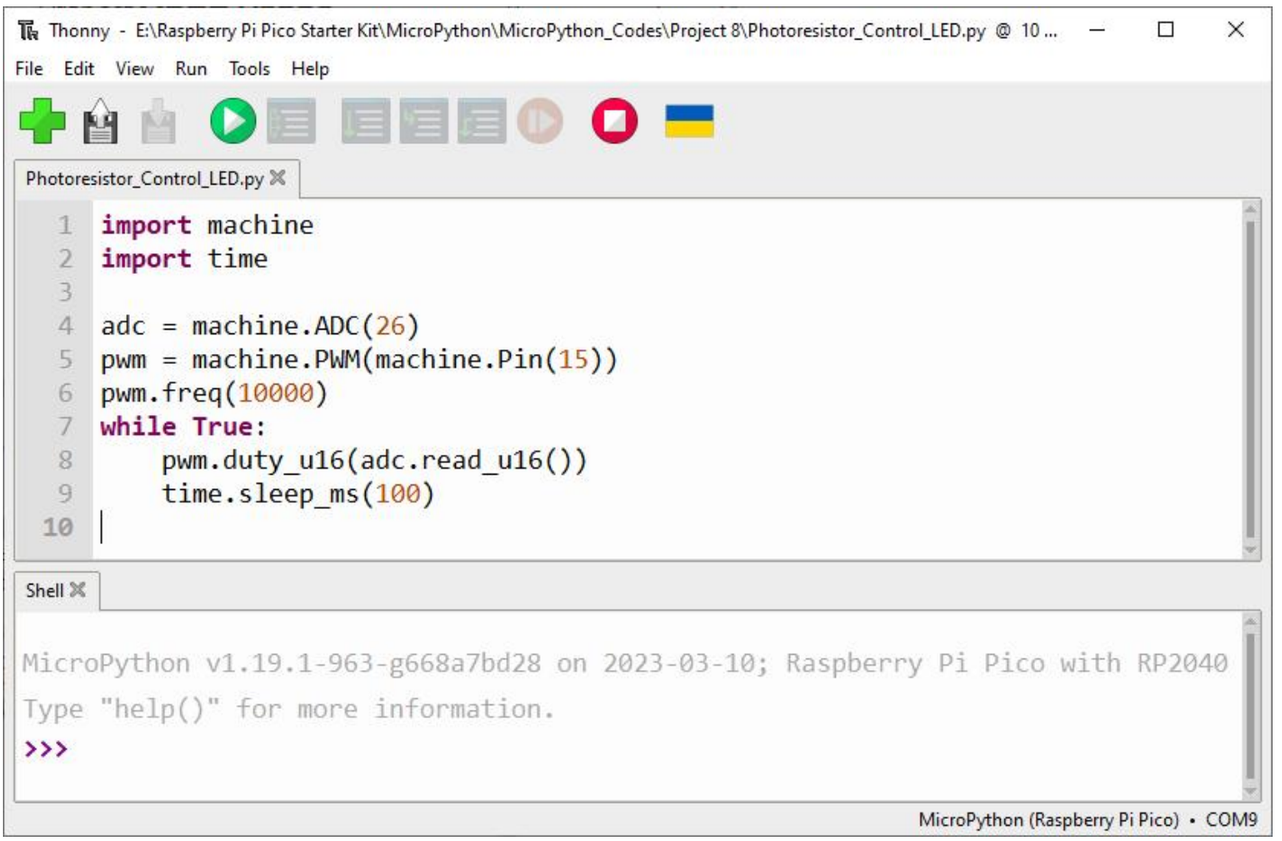
Code

Click the File>>open icon to open Project 8 \ Photoresistor_Control_LED.py file.

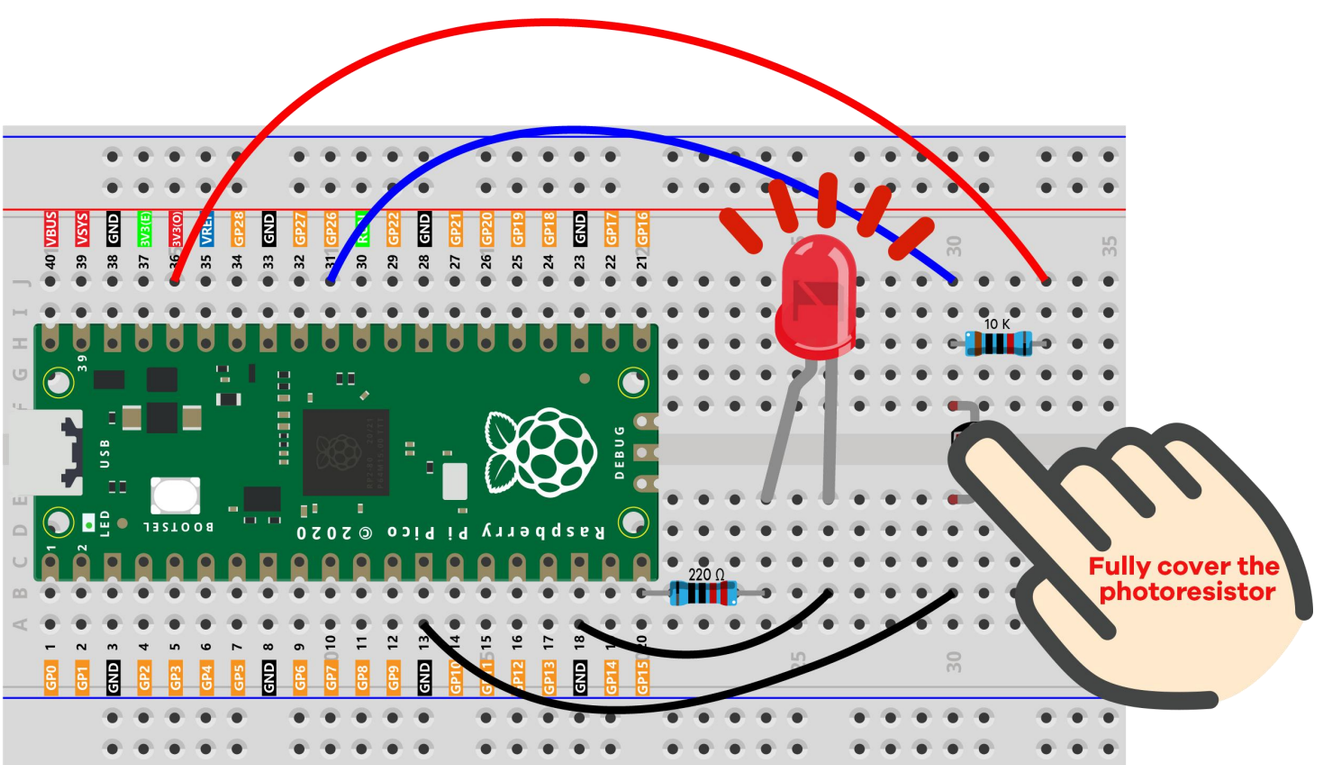
If you have downloaded the tutorial. You can find the .py file.

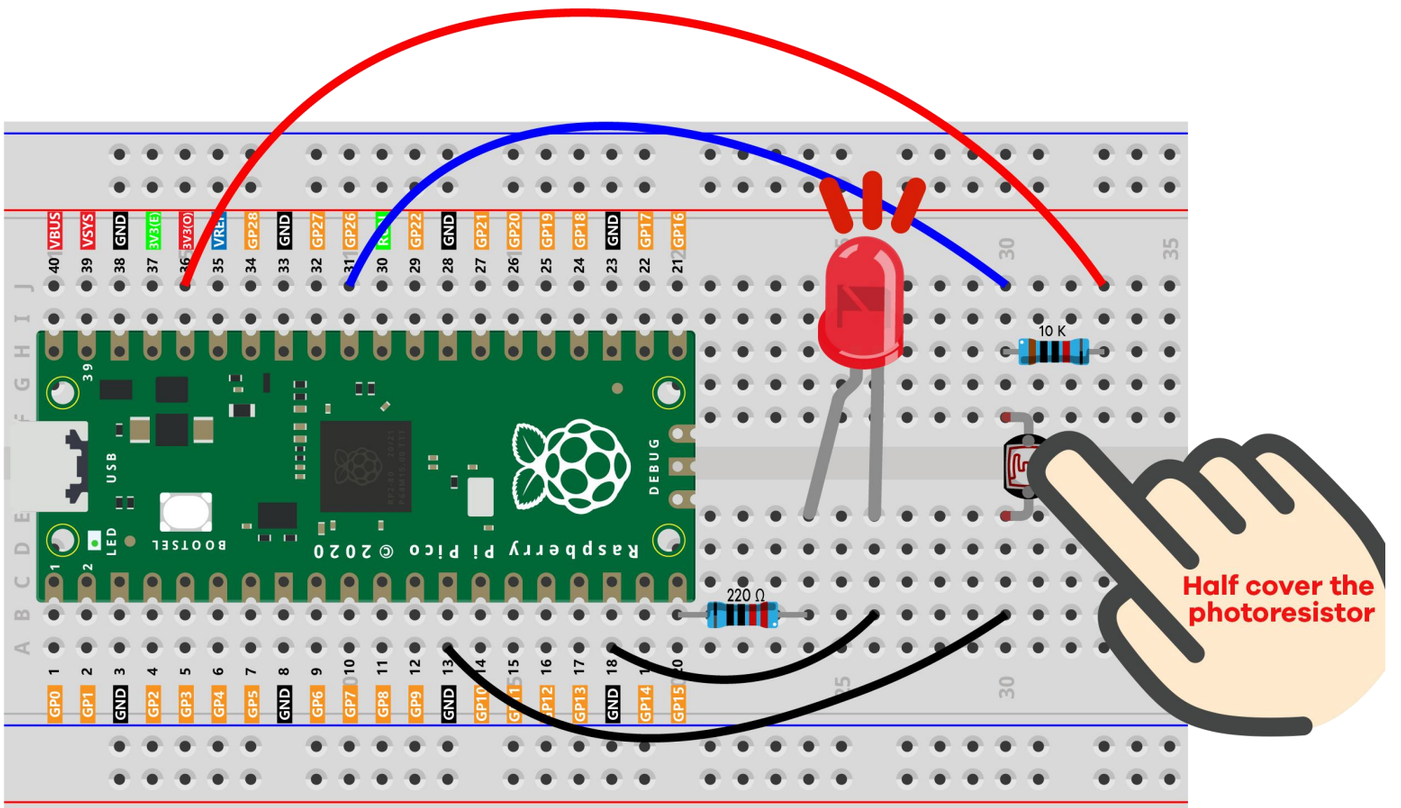
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

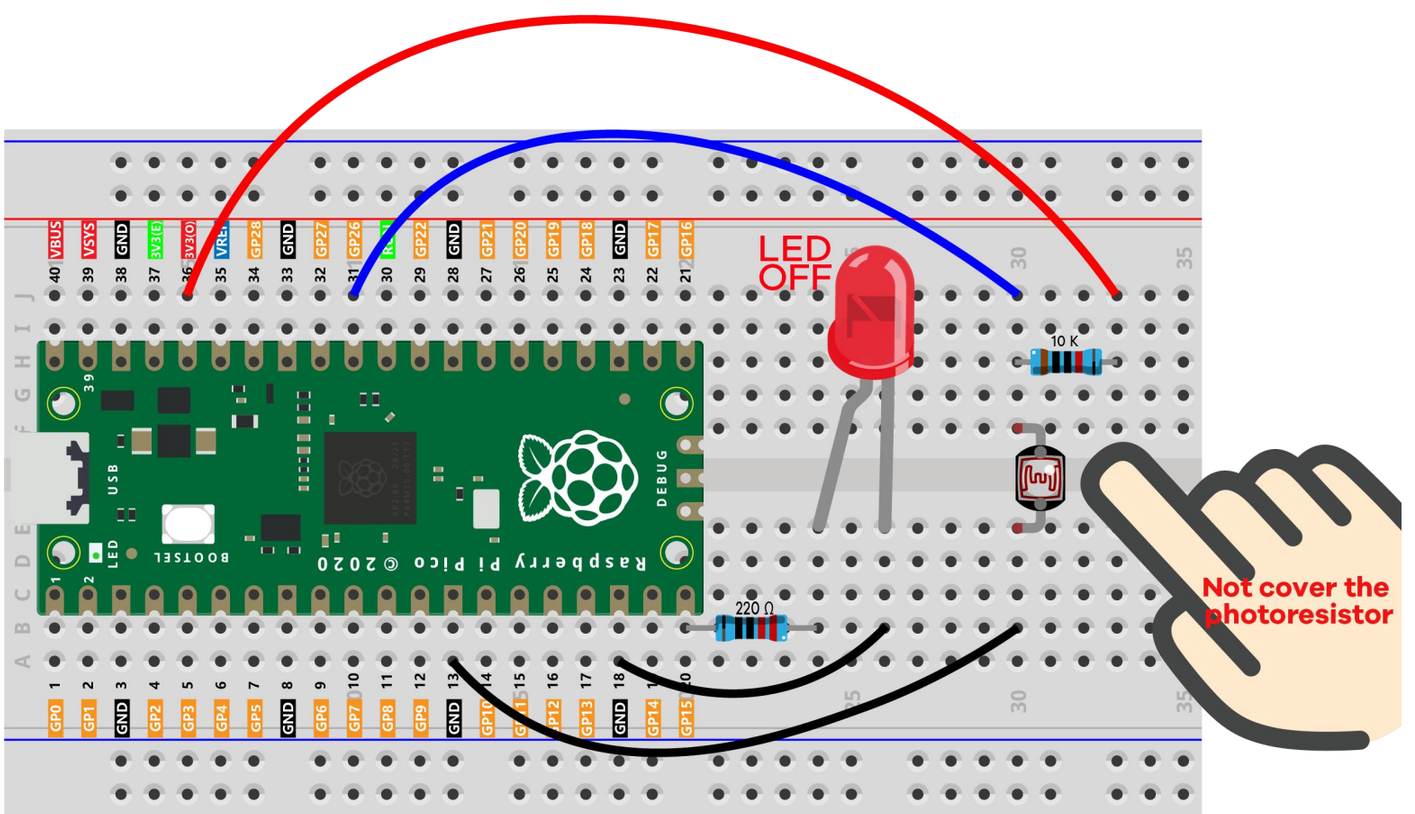
Note:
Don’t forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click “Run current script”  . Cover the photoresistor with your hands or illuminate it with lights, the brightness of LEDs will change.




10 K
220 Ω
Half cover the photoresistor

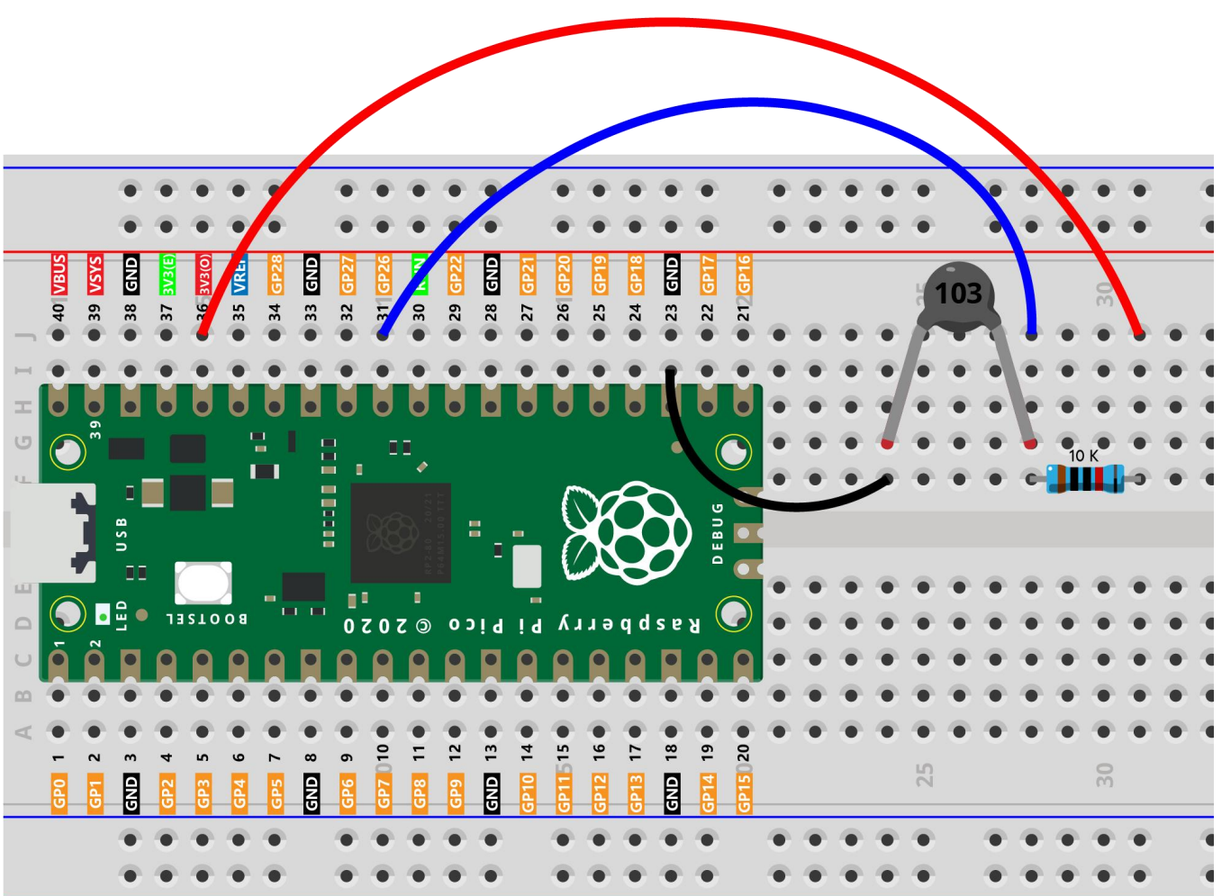

LED OFF
10 K
220 Ω
Not cover the photoresistor

Project 9 Thermeometer

A Thermistor is a type of Resistor whose resistance value is dependent on temperature and changes in temperature. Therefore, we can take advantage of this characteristic to make a Thermometer.

Tip:Learn more about Thermistor

Wiring



Code

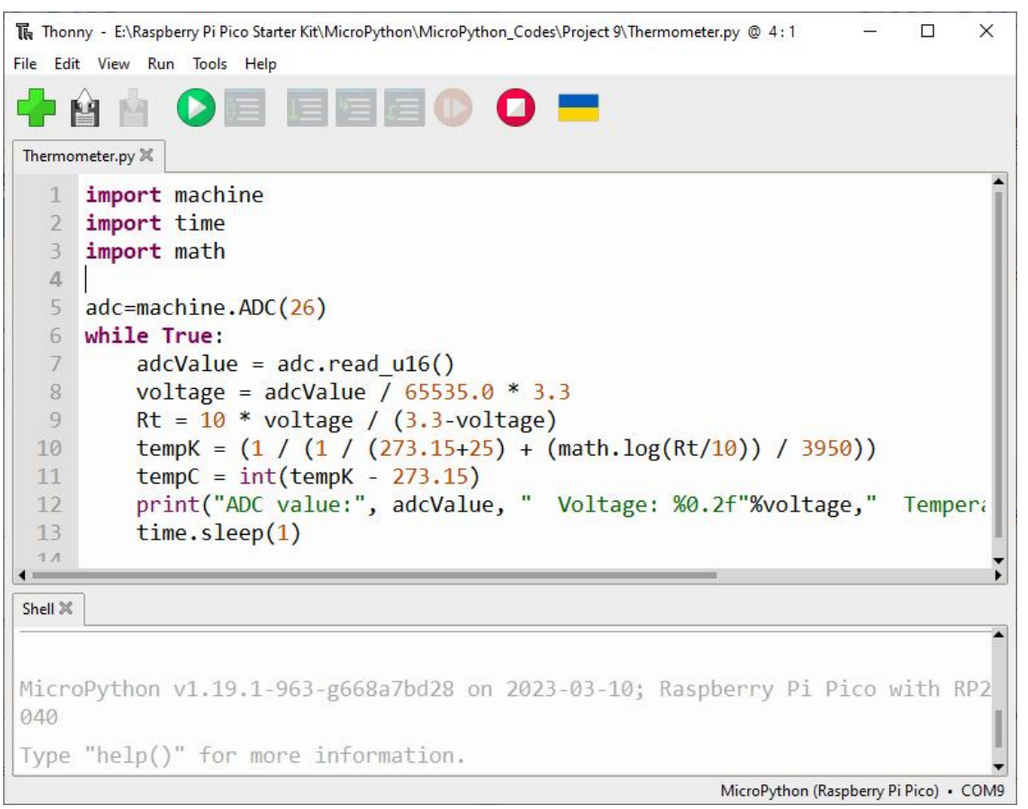
Click the File>>open icon to open Project 9 \ Thermometer.py file.

If you have downloaded the tutorial. You can find the .py file.

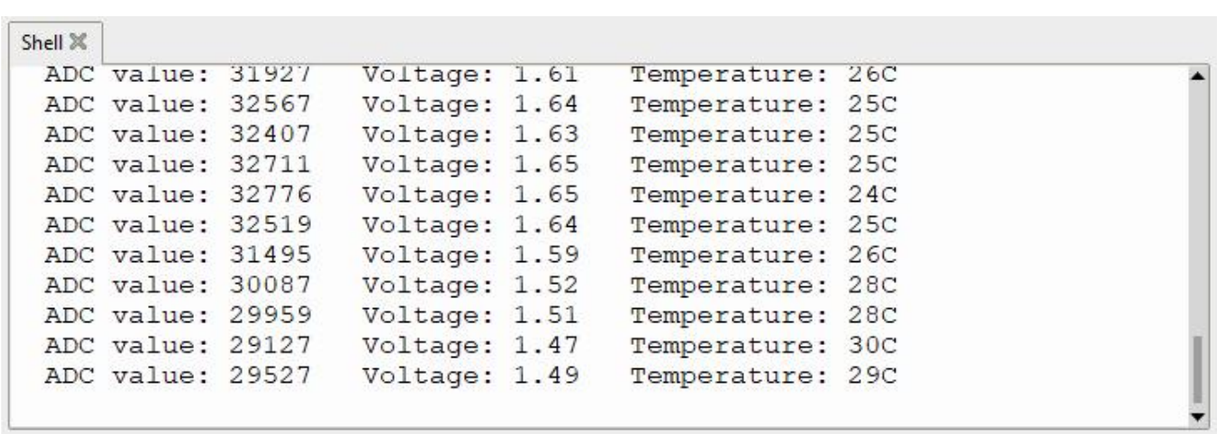
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

Don't forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click "Run current script"  .Print centigrade degree, Fahrenheit degree and their units in the shell.



How it works?

Each thermistor has a normal resistance. Here it is 10k ohm, which is measured under 25 degree Celsius. When the temperature gets higher, the resistance of the thermistor decreases. Then the voltage data is converted to digital quantities by the A/D adapter.The temperature in Celsius or Fahrenheit is output via programming.

```
import math
```

There is a numerics library which declares a set of functions to compute common mathematical operations and transformations.

- math

```
adcValue = adc.read_u16()
```

read_u16() function is called to read the value of ADC0

```
voltage = adcValue / 65535.0 * 3.3
```

Convert the read ADC0 value to get the current Thermistor voltage value.

```
Rt = 10 * voltage / (3.3-voltage)
tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
```

According to the formula:

$$T2 = 1/(\frac{1}{T1} + \ln(\frac{Rt}{R})/B)$$

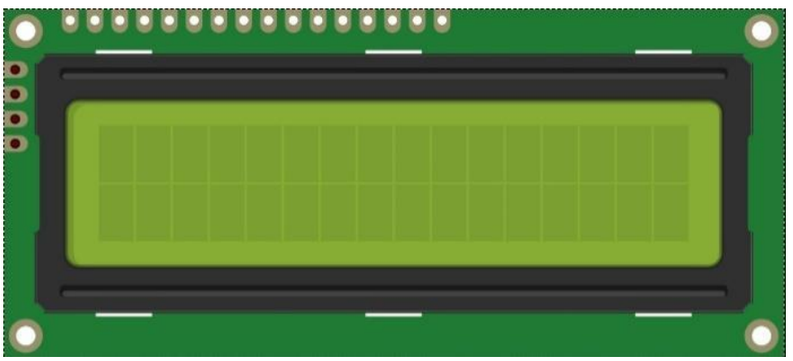
where T1 = 25℃, R = 10KΩ, B = 3950 and the Rt calculated in the previous step, substitute the formula to calculate tempK(T2) . Get the value of the temperature unit K.Learn More

```
tempC = int(tempK - 273.15)
```

Finally, tempK (unit: K) is converted to tempC (unit: °C). You can also convert to Fahrenheit based on your needs.

Project 10 LCD1602 Show Temperature

Using a thermistor and an I2C LCD1602, we can create a room temperature meter.

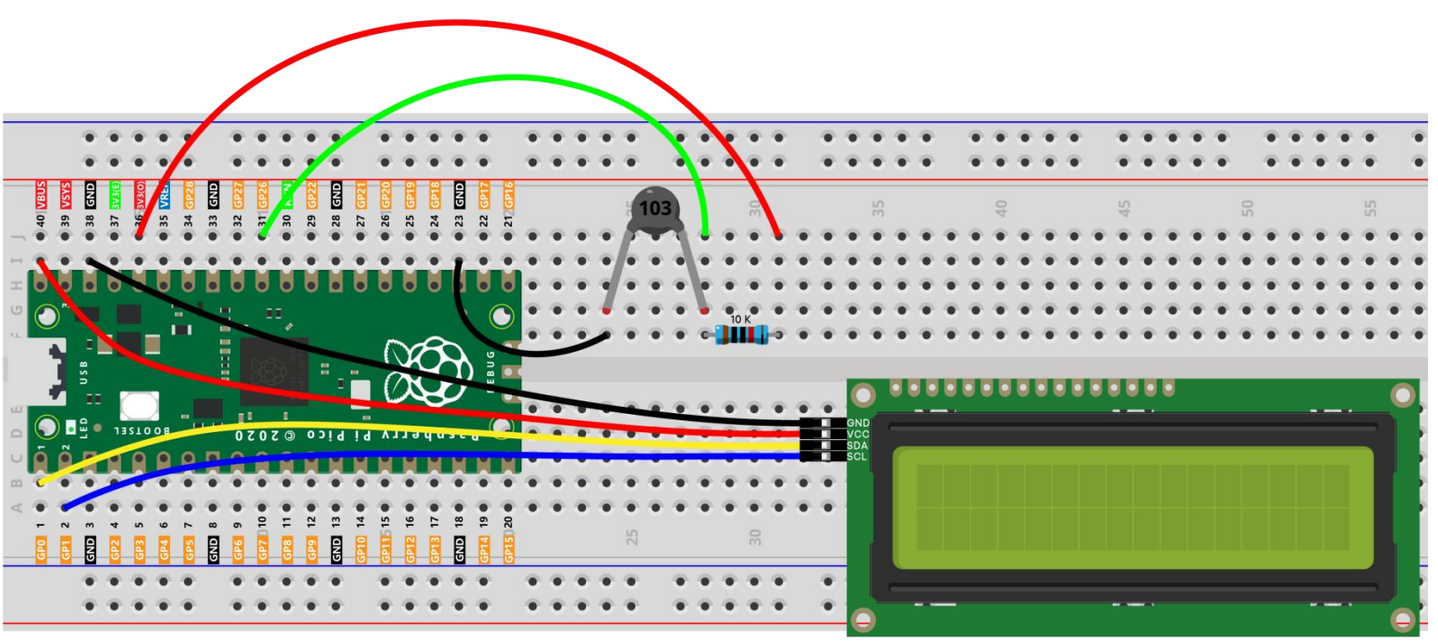


LCD1602 is a character type liquid crystal display, which can display 32 (16*2) characters at the same time.

I2C LCD1602 Display Screen integrates an I2C interface, which connects the serial-input & parallel-output module to the LCD1602 Display Screen. This allows us to use only 4 lines to the operate the LCD1602.

Tip:Learn more about LCD1602

Wiring

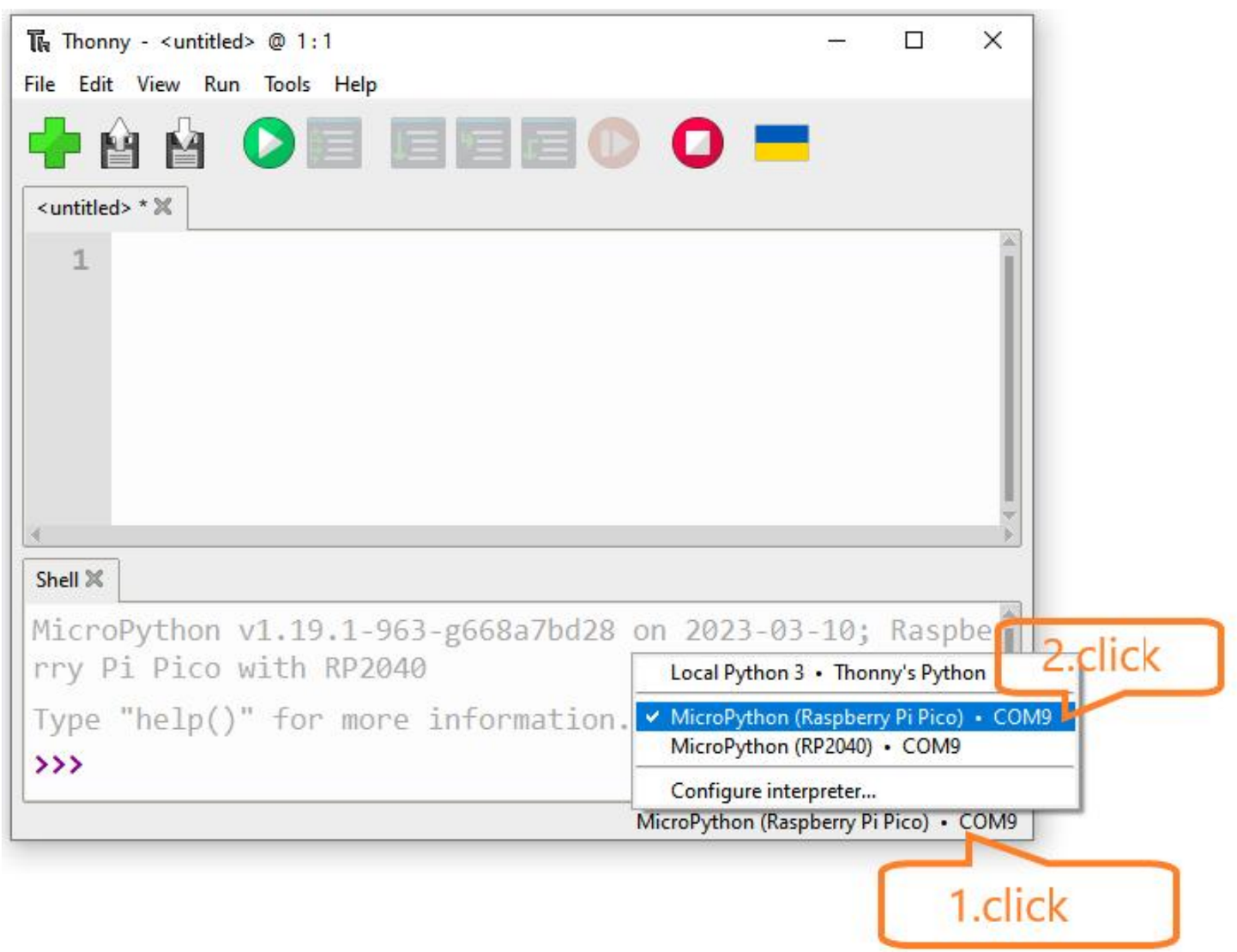


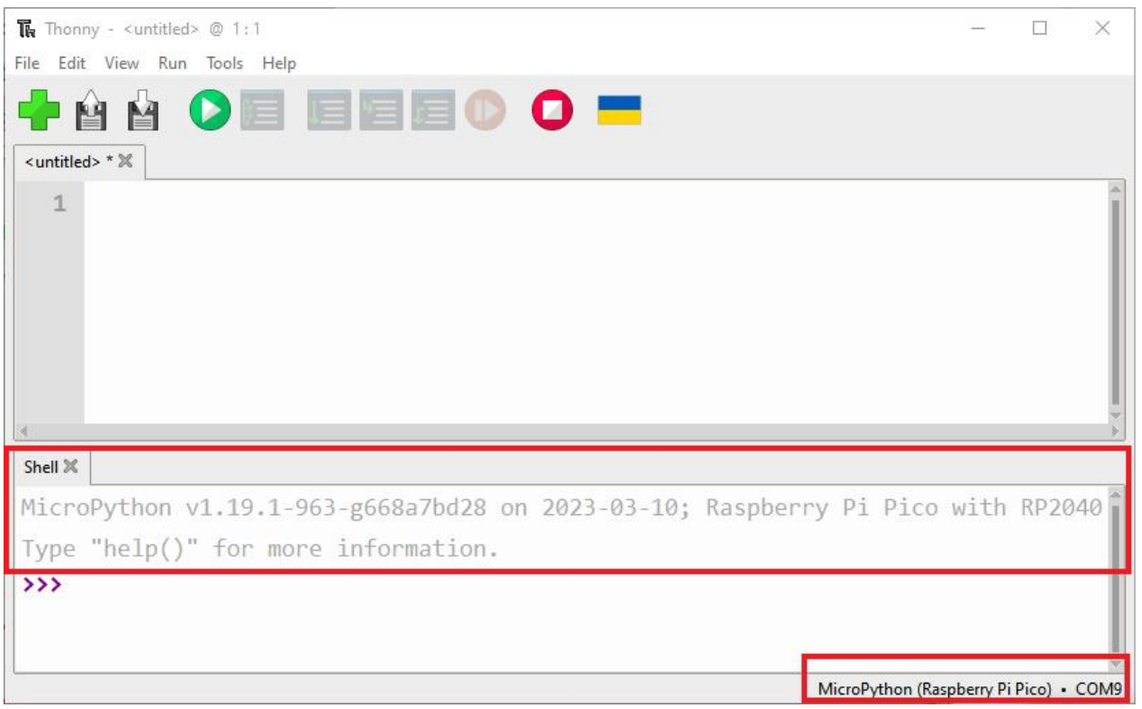
Upload LCD1602.py to Raspberry Pi Pico(Important)

In order for the display control code to be compiled successfully, the lcd1602.py library file must be upl oaded to the internal storage of the Raspberry Pi Pico.

Step①:Thonny Connected to Raspberry Pi Pico

Left click on the bottom right corner and switch the interpreter version name to MicroPython (RaspPi Pberry ico) ● COMx.Have Question?

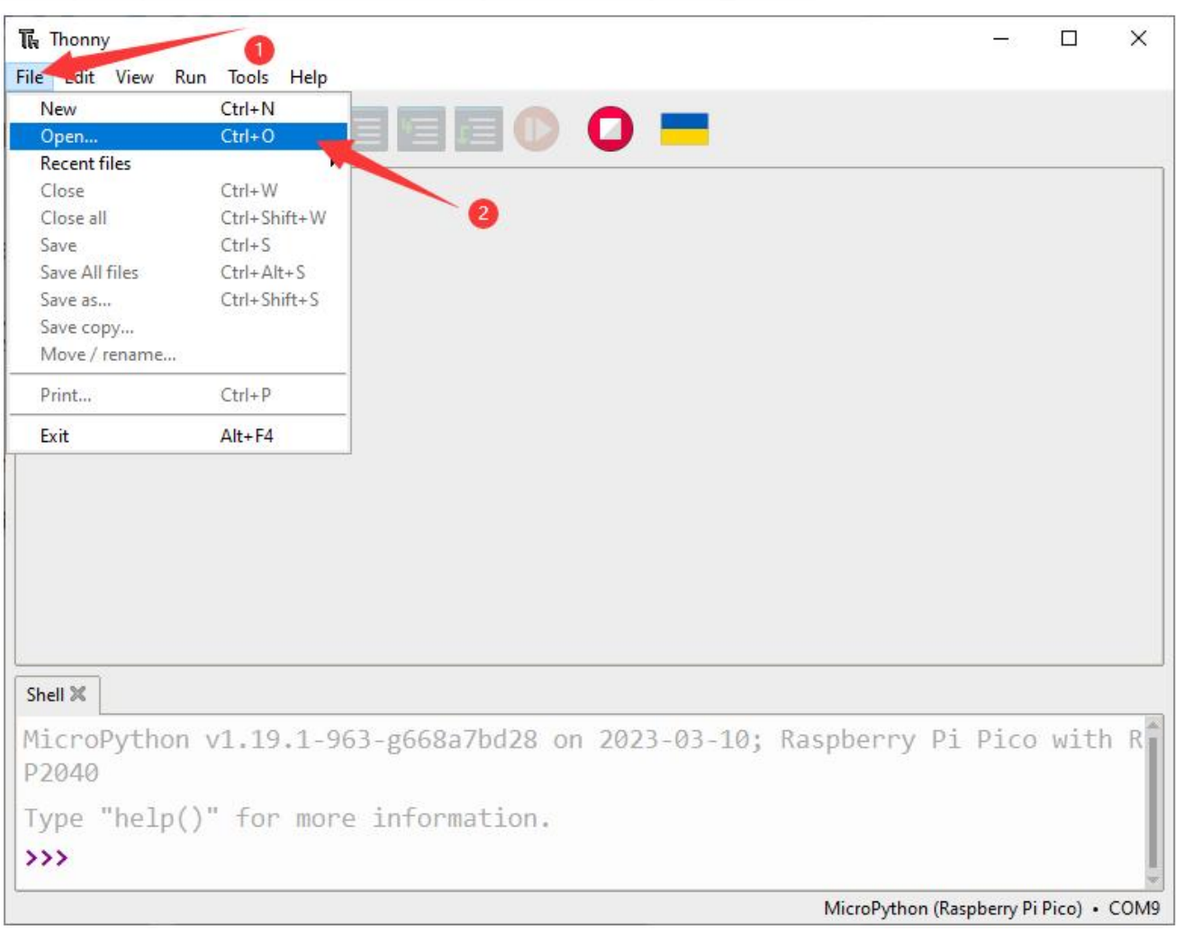


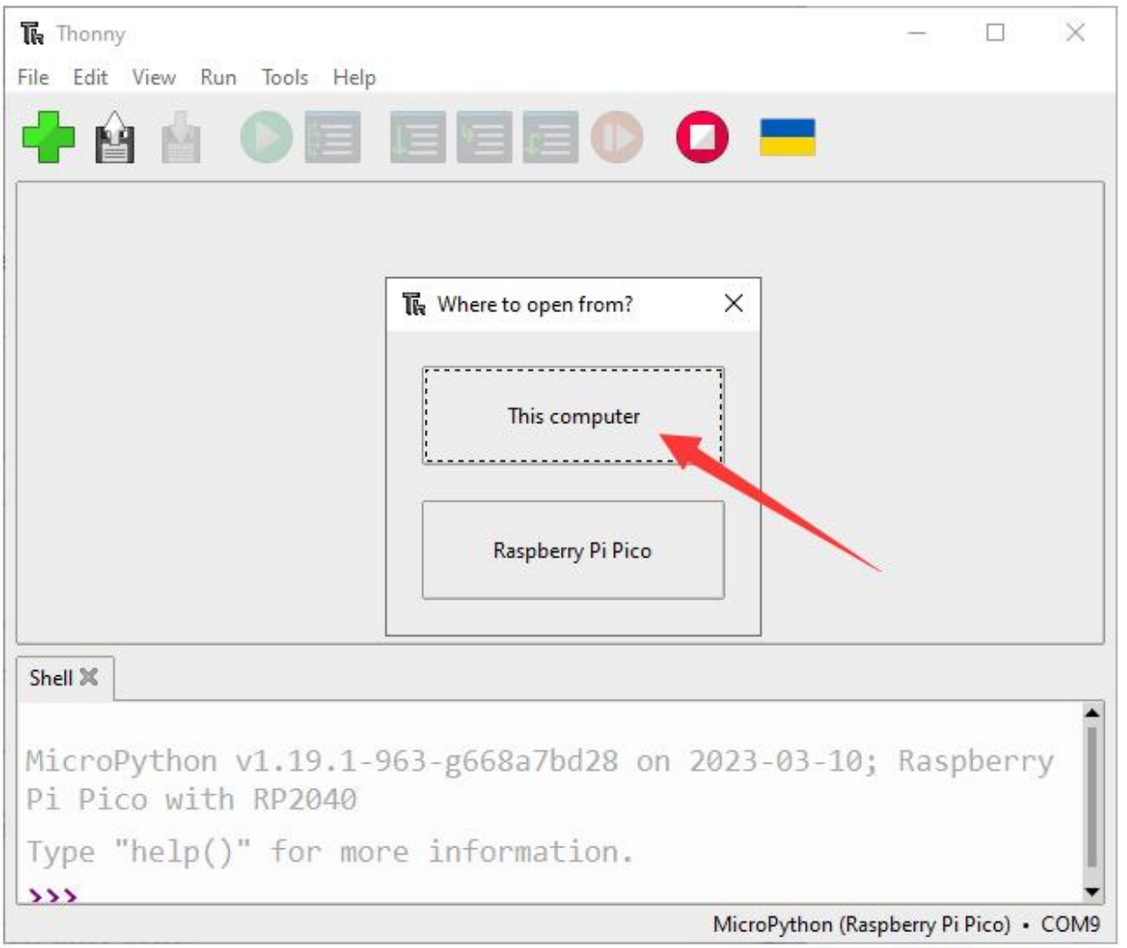


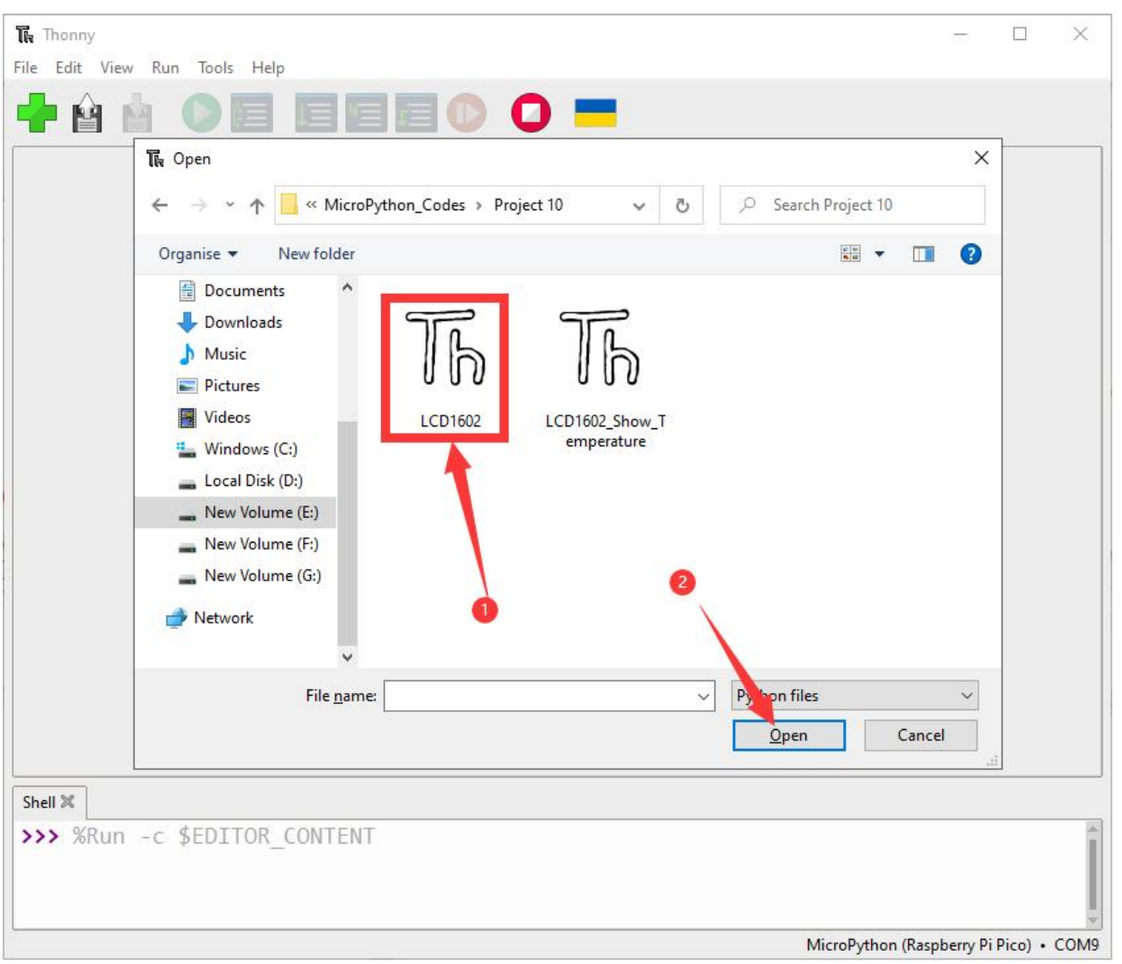
Step②:Open LCD1602.py

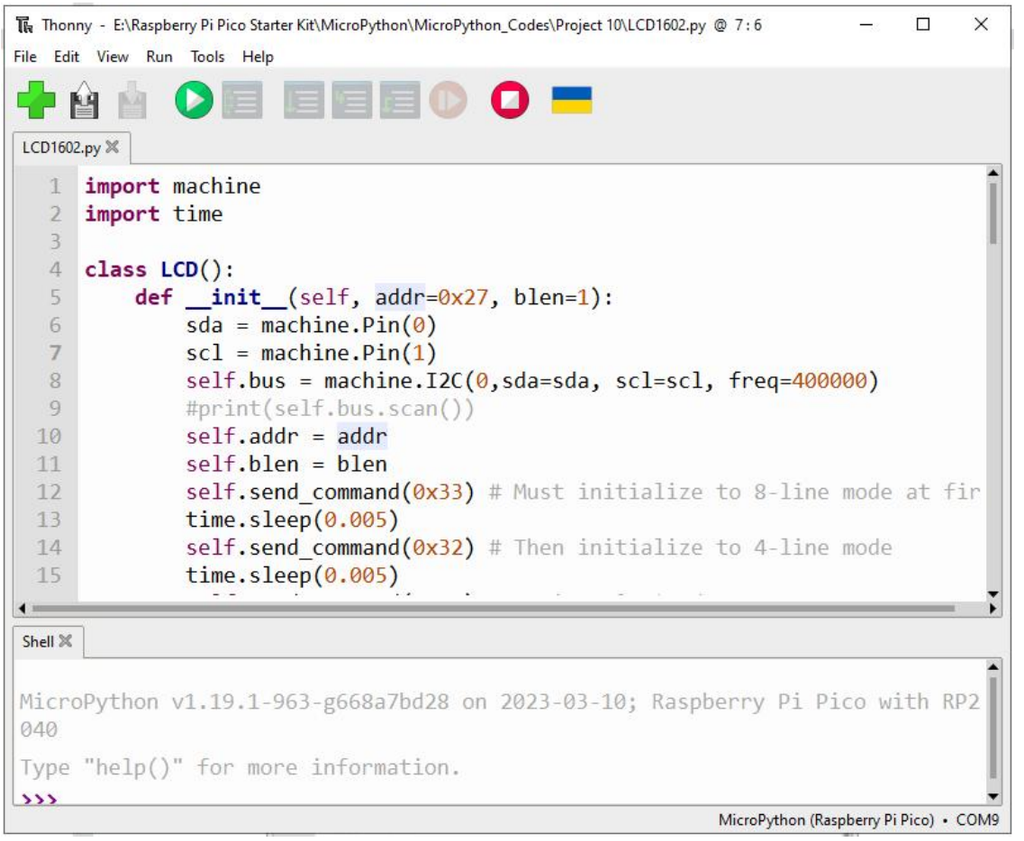
Click the File>>open icon to open Project 10 \ LCD1602_Show_Temperature.py file.If you have downloaded the tutorial. You can find the .py file in This computer.

File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

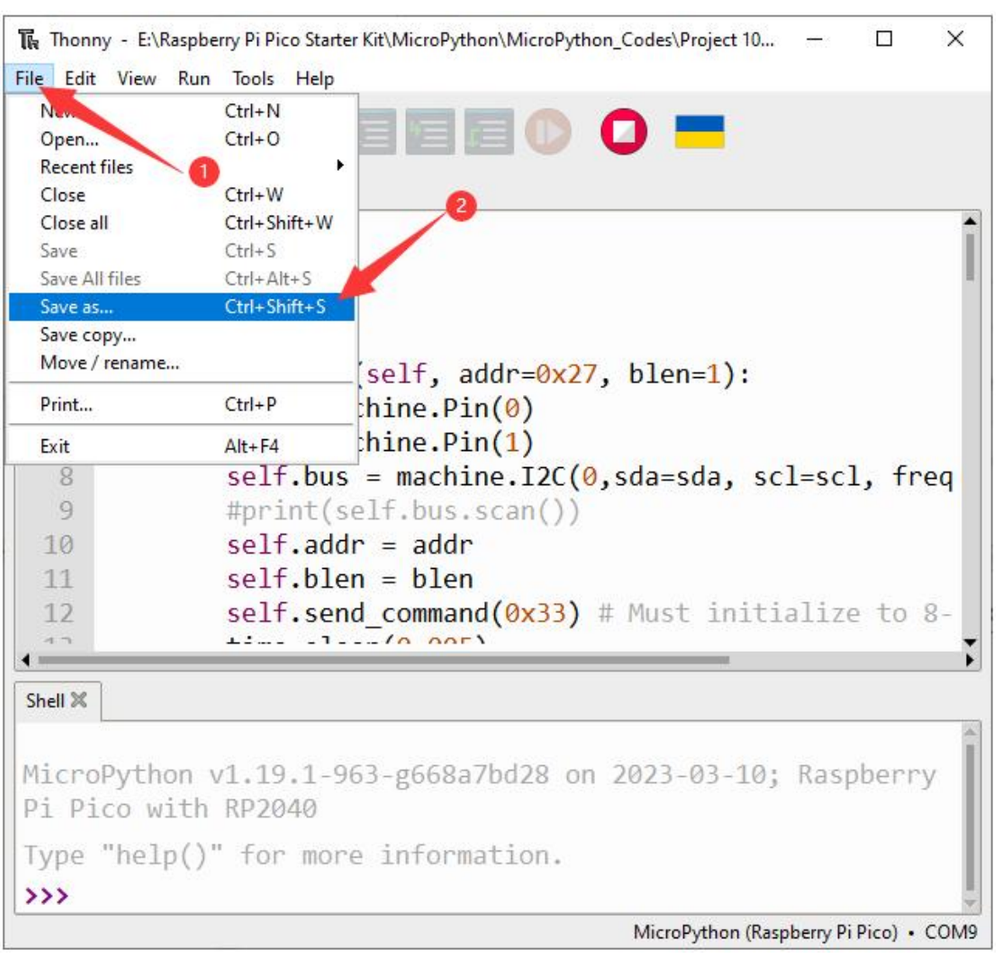


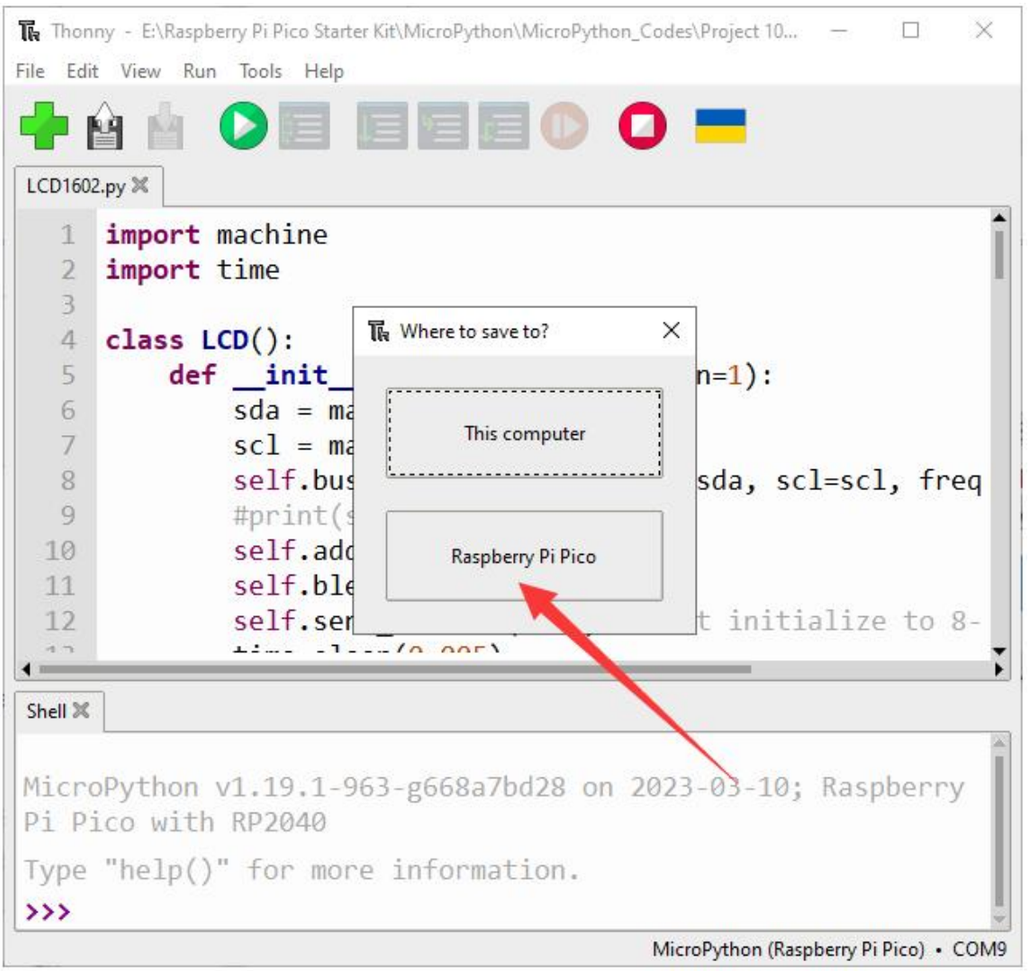

Thonny
File Edit View Run Tools Help
Where to open from?
This computer
Raspberry Pi Pico
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico) • COM9

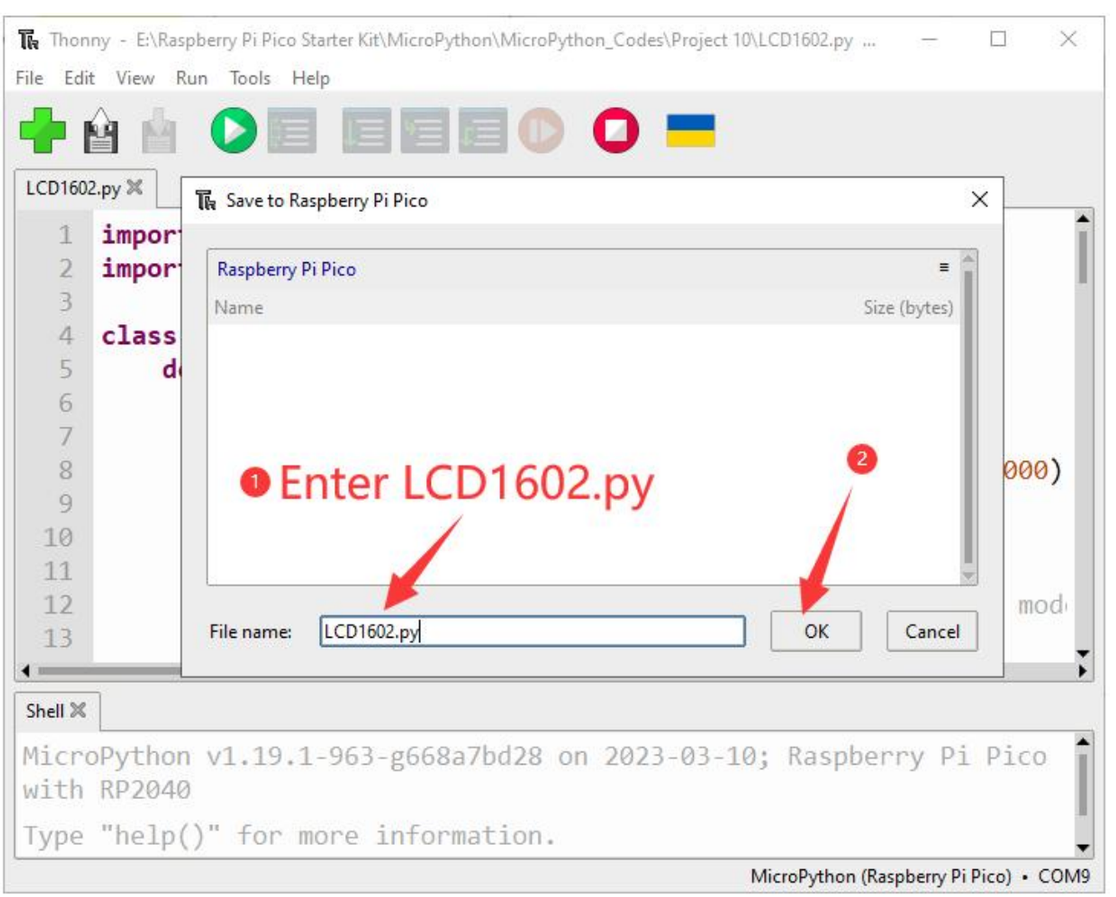

Thonny
File Edit View Run Tools Help
Open
« MicroPython_Codes › Project 10
Search Project 10
Organise
New folder
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Local Disk (D:)
New Volume (E:)
New Volume (F:)
New Volume (G:)
Network
LCD1602
LCD1602_Show_Temperature
1
2
File name:
Python files
Open
Cancel
Shell
>>> %Run -c $EDITOR_CONTENT
MicroPython (Raspberry Pi Pico) • COM9



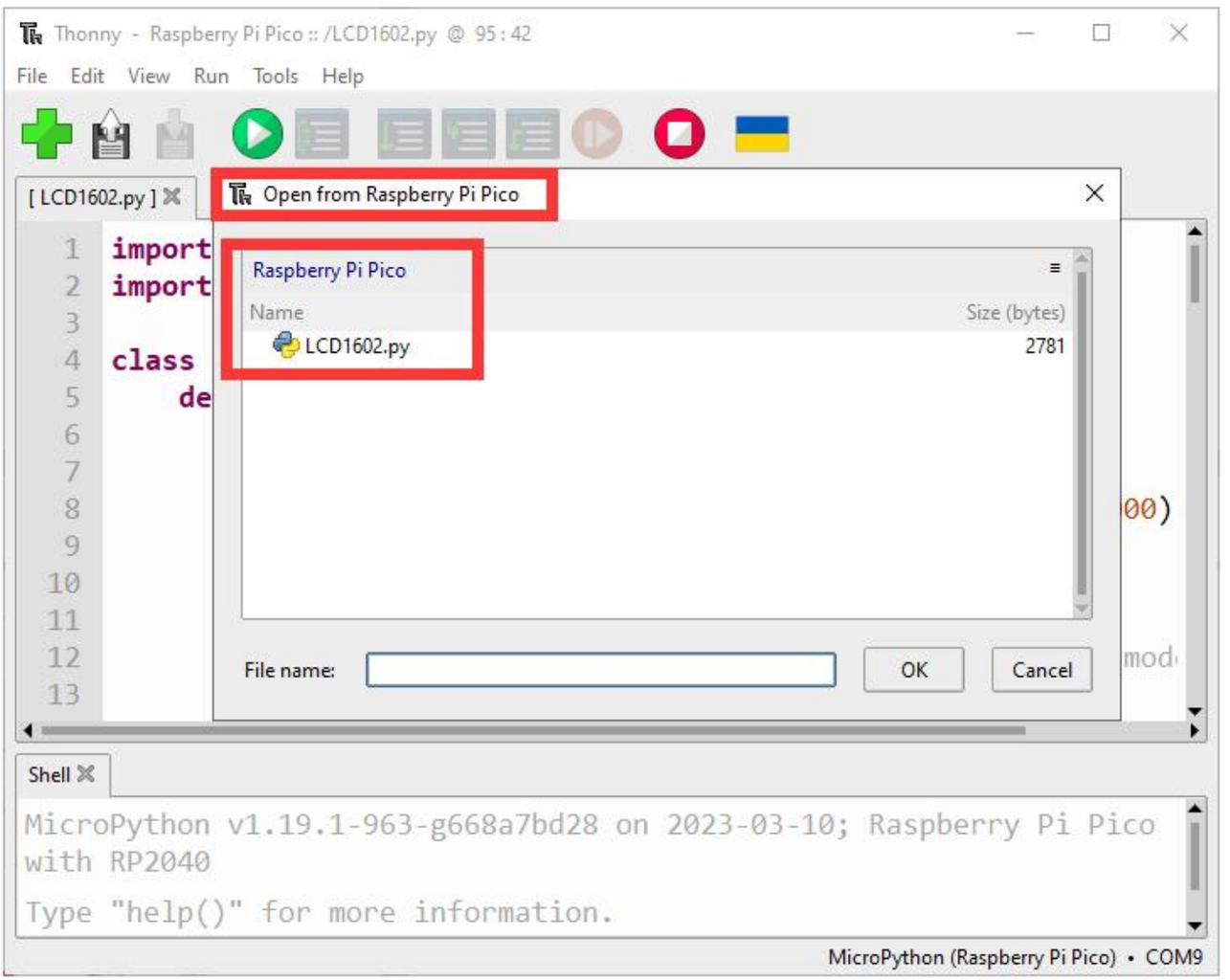
Step③:Save as LCD1602.py to Raspberry Pi Pico




Thonny - E:\Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes\Project 10...
File Edit View Run Tools Help
LCD1602.py
import machine
import time
class LCD():
Where to save to?
This computer
Raspberry Pi Pico
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico) • COM9


Thonny - E:\Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes\Project 10\LCD1602.py ...
File Edit View Run Tools Help
LCD1602.py
Save to Raspberry Pi Pico
Raspberry Pi Pico
Name
Size (bytes)
1 Enter LCD1602.py
2
File name:
LCD1602.py
OK
Cancel
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
MicroPython (Raspberry Pi Pico) • COM9

If the library file is successfully saved to the Pi Pico memory, click **File>Open>open from Raspberry Pi Pico**, and you will see LCD1602.py in the Pi Pico memory.



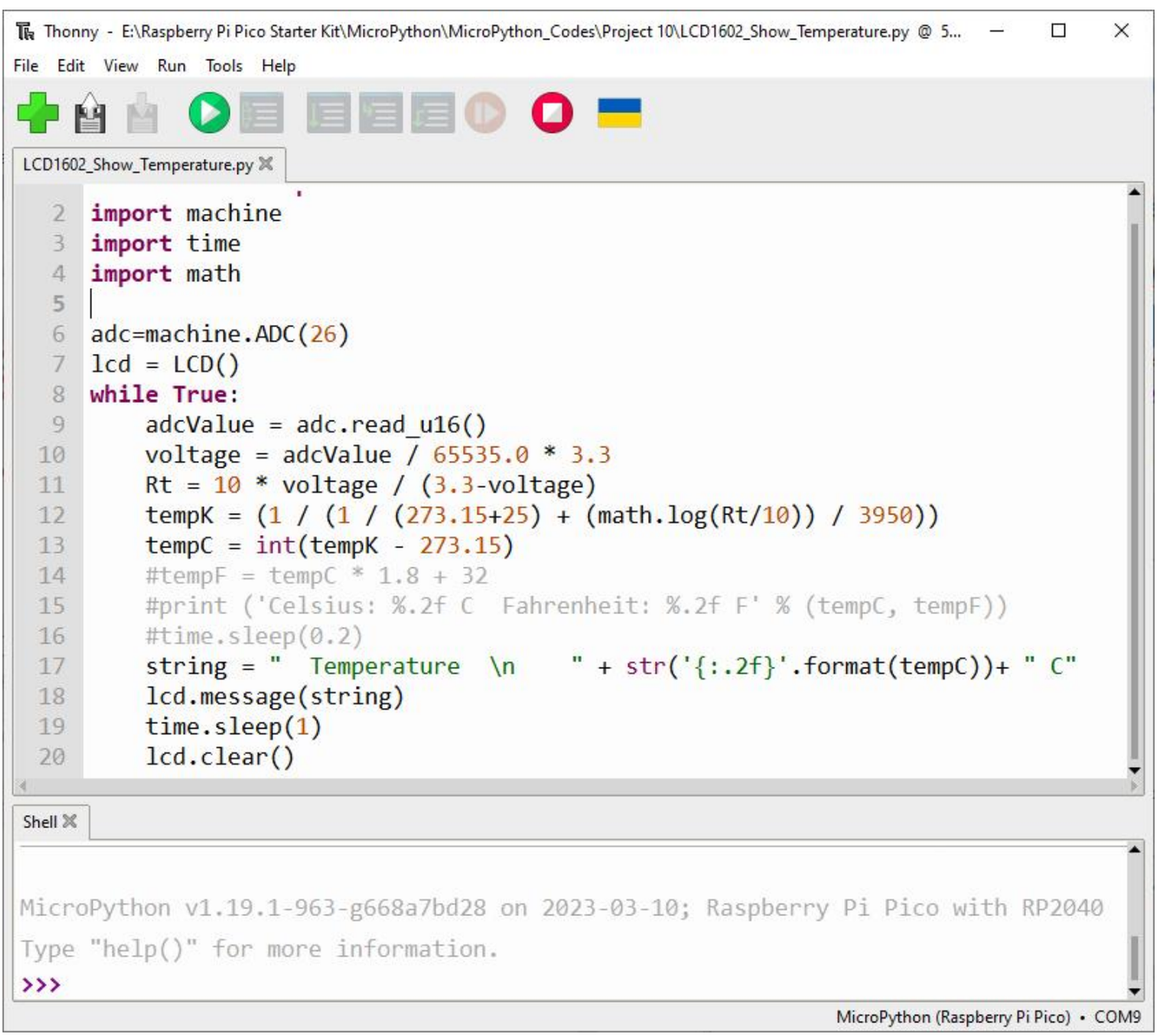
Code

Click the File>>open icon to open Project 10 \ LCD1602_Show_Temperature.py file.If you have downloaded the tutorial. You can find the .py file.

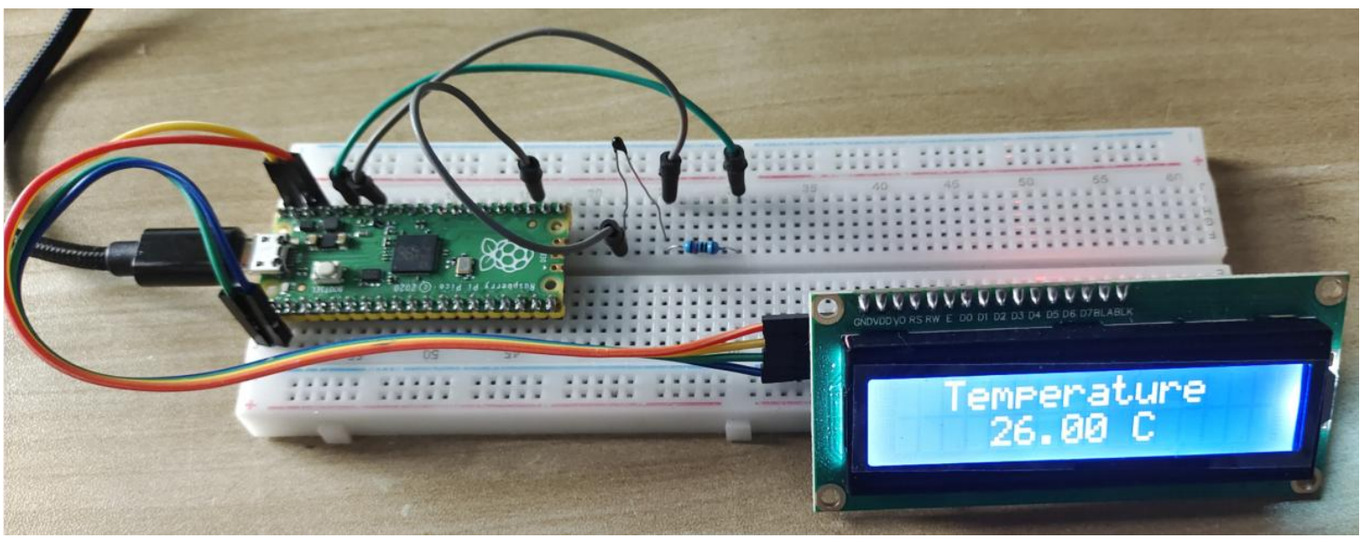
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

Don't forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



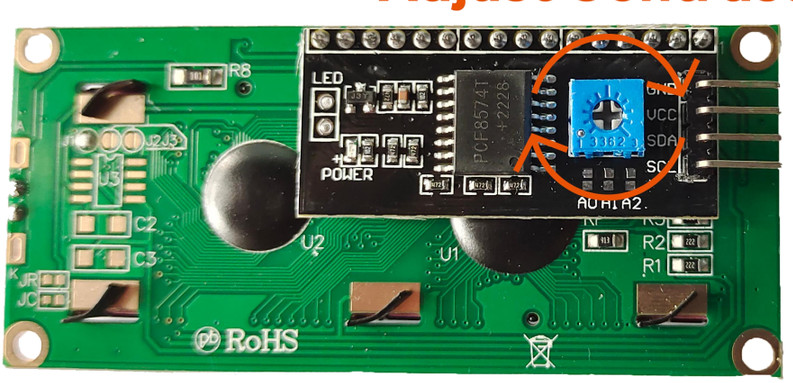
Click “Run current script”  .The LCD will display the temperature value in the current environment.



Note:

If the code and wiring are fine, but the LCD still does not display content or the display is not clear, you can turn the potentiometer on the back to increase the contrast.

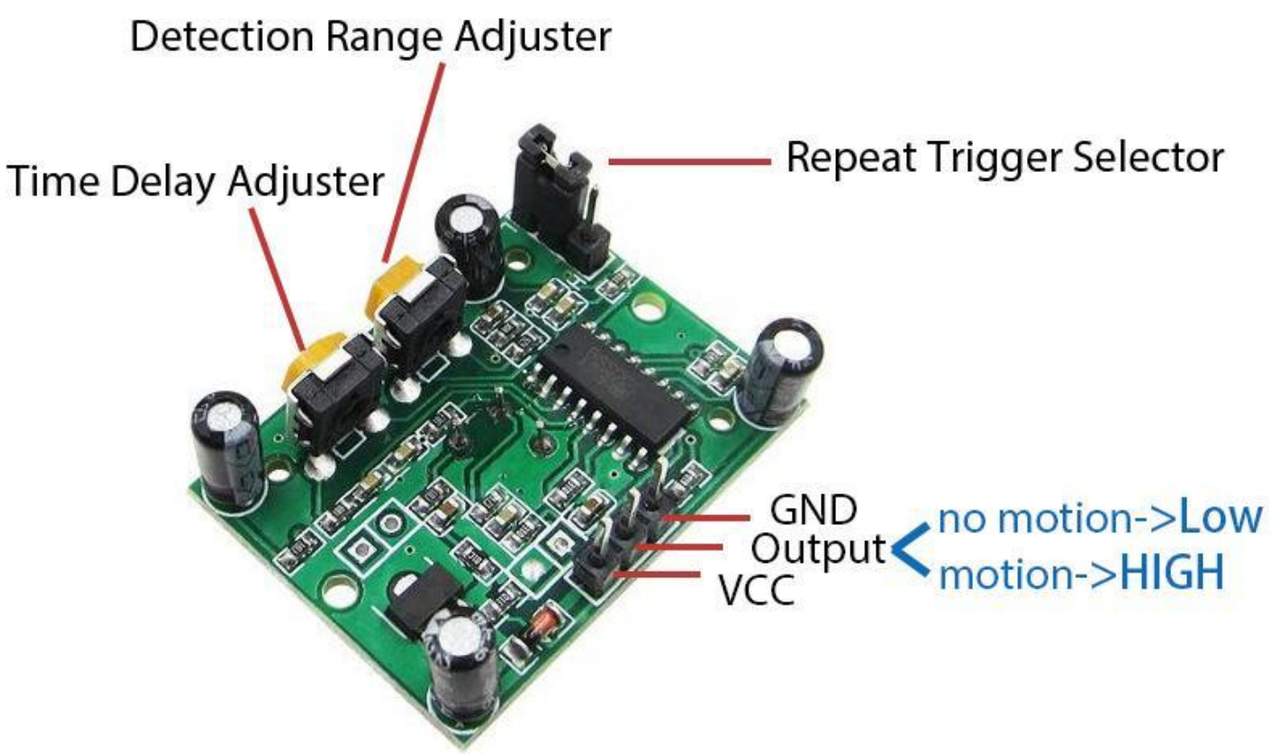
Adjust Contrast



Project 11 Intruder Alarm

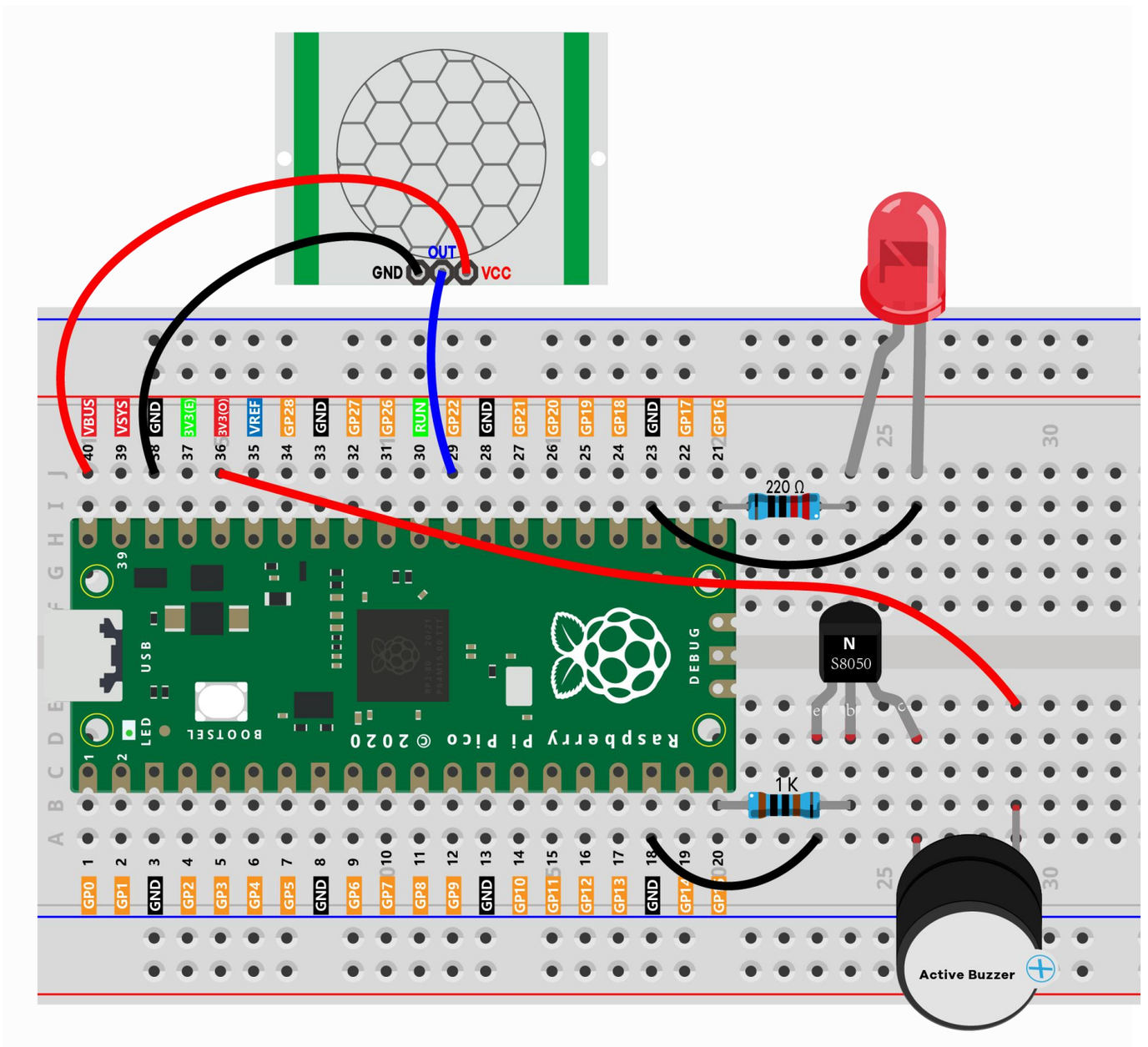
In this project,use PIR,LED and active buzzer to build an Intruder Alarm.

Passive infrared sensor (PIR sensor) is a common sensor that can measure infrared (IR) light emitted by objects in its field of view. Simply put, it will receive infrared radiation emitted from the body, thereby detecting the movement of people and other animals. More specifically, it tells the main control board that someone has entered your room.



Tip:Learn more about PIR Motion Sensor Transistor Buzzer

Wiring



Note:

①The working voltage of PIR senso is 5V. Use the VBUS pin to power it.

②Two types of buzzers are included in the kit. We need to use active buzzer. Turn them around, the sealed back (not the exposed PCB) is the one we want.

③The 1K resistor between the NPN transistor and GP15 is used for current limiting when the transistor is energized.

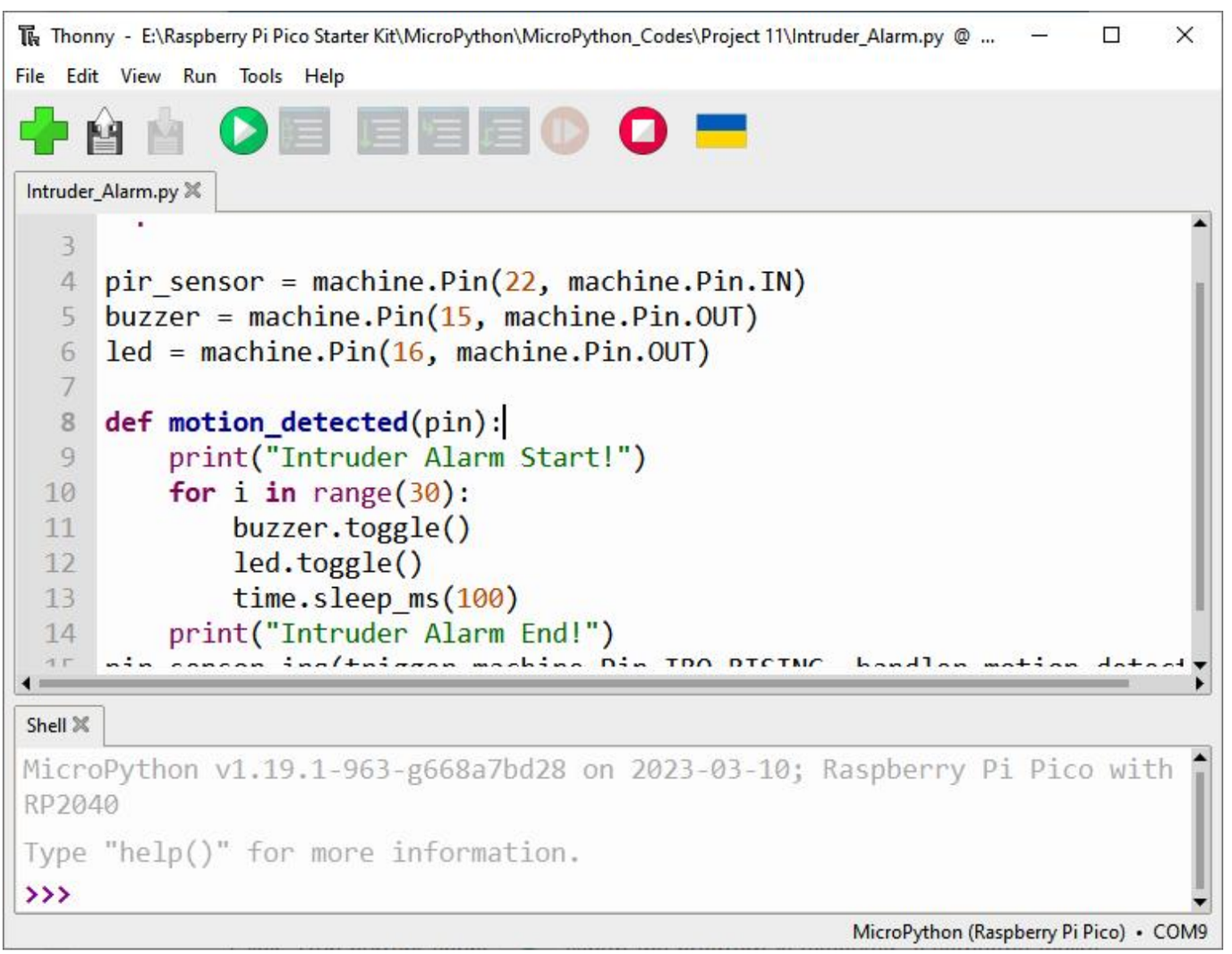
Code

Click the File>>open icon to open Project 11 \ Intruder Alarm.py file.If you have downloaded the tutorial. You can find the .py file.

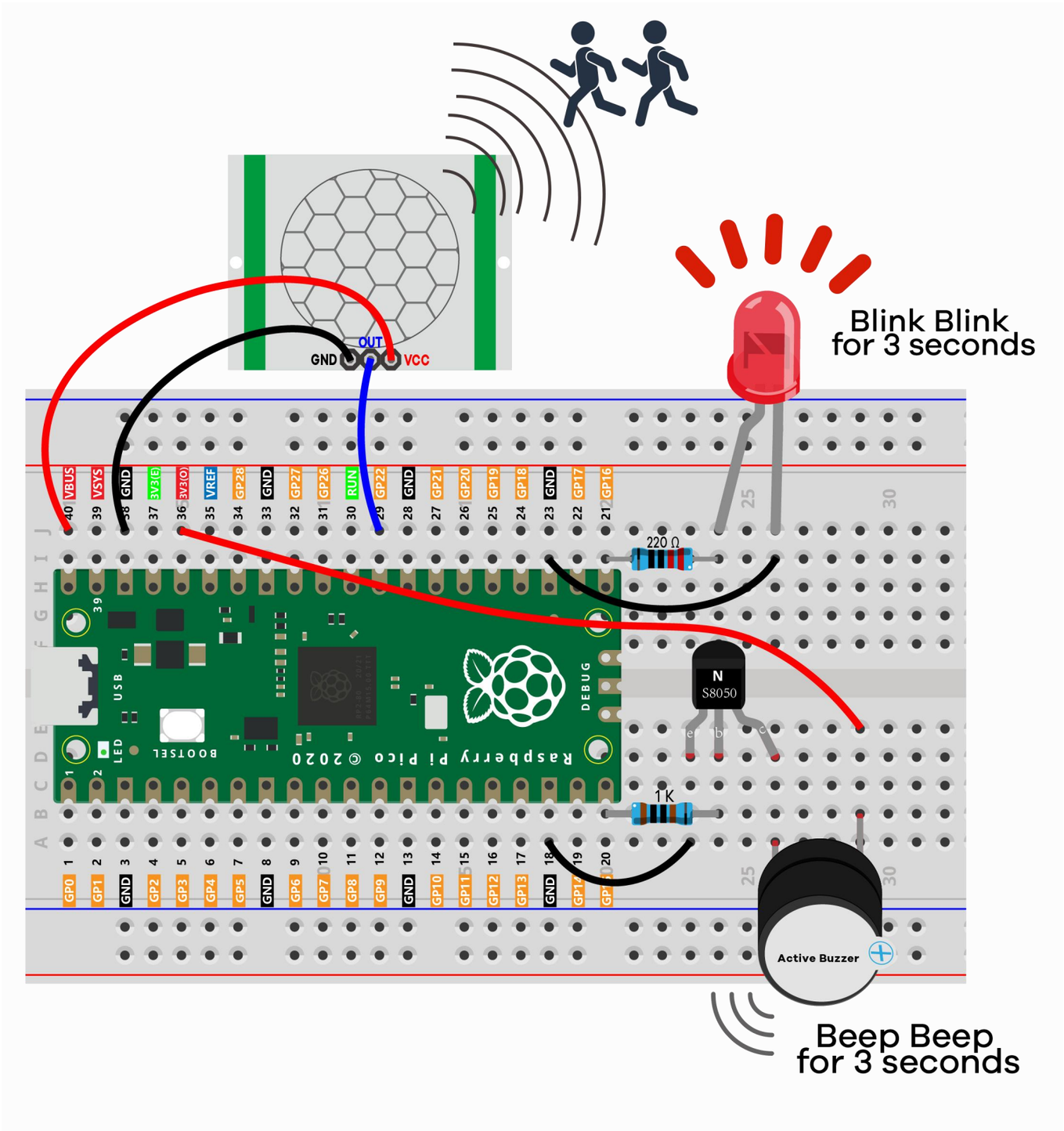
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

Note:

Don’t forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



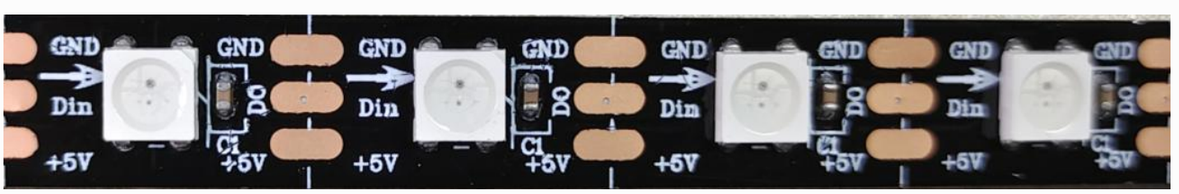
Click “Run current script” .When the program is executed, if someone walks into the PIR detection range, the buzzer will be ’BEEP BEEP’ for 3 seconds.the led will be ’blink blink’ for 3 seconds. Have a question about using PIR Motion Sensor?



Project 12 RGB LED Strip

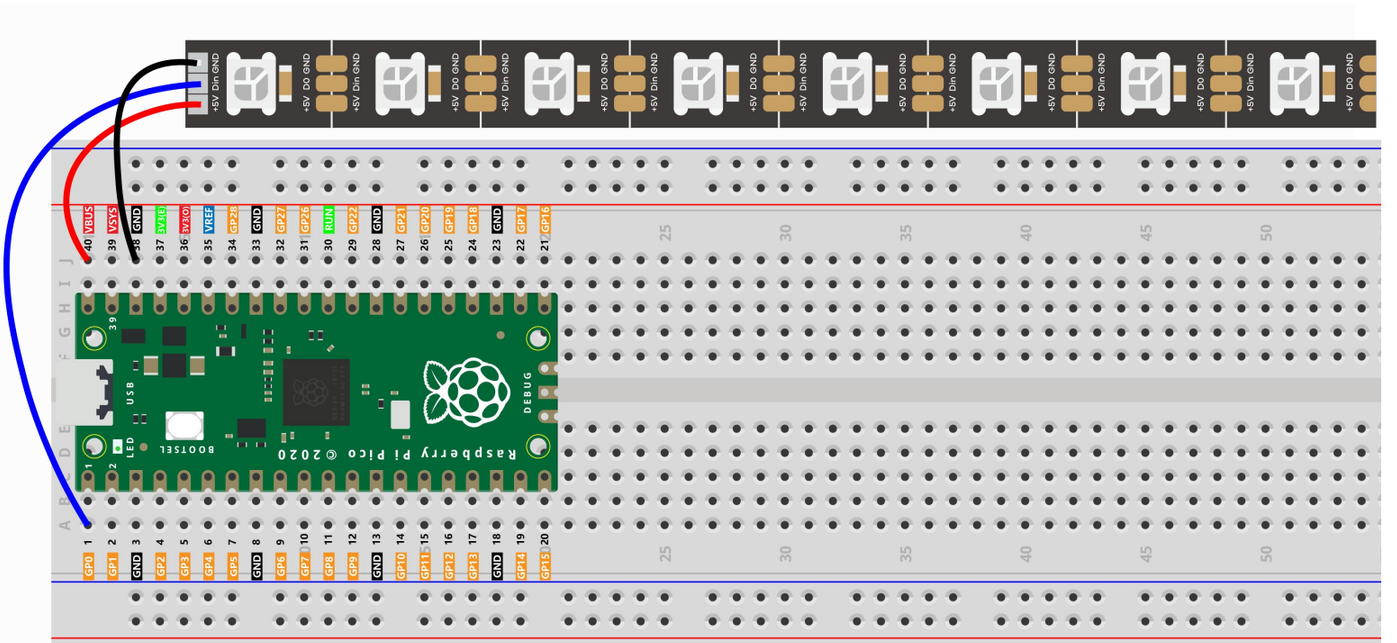
WS2812 is a intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. It internal include intelligent digital port data latch and signal reshaping amplification drive circuit. Also include a precision internal oscillator and a 12V voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.

The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel，the signal to reduce 24bit. pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depend on the speed of signal transmission.



Tip:Learn more about RGB LED Strip

Wiring



Note:

The working voltage of servo is 5V. Use the VBUS pin to power it.

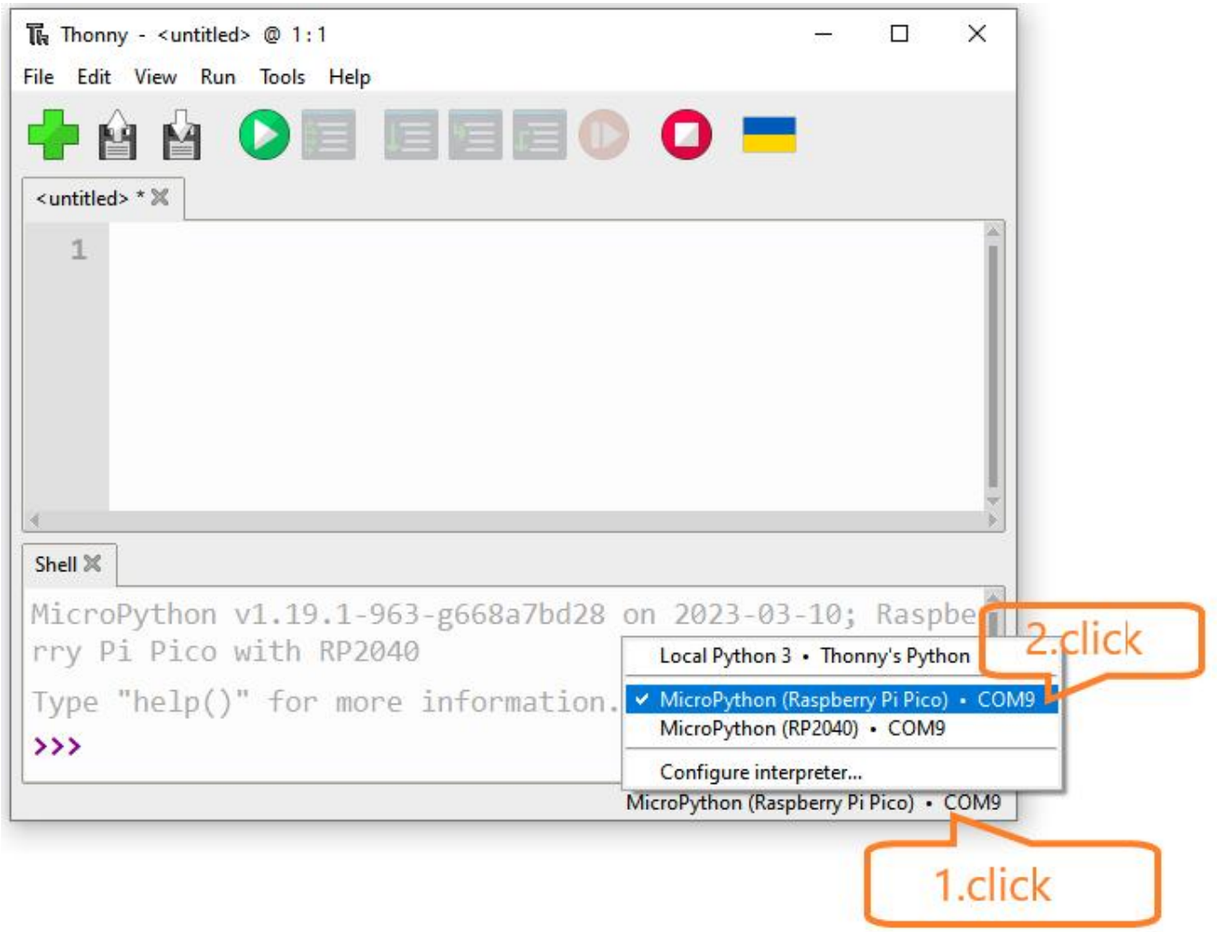
Although the LED Strip with any number of LEDs can be used in Pico, the power of its VBUS pin is limited. Here, we will use eight LEDs, which are safe. But if you want to use more LEDs, you need to add a separate power supply.

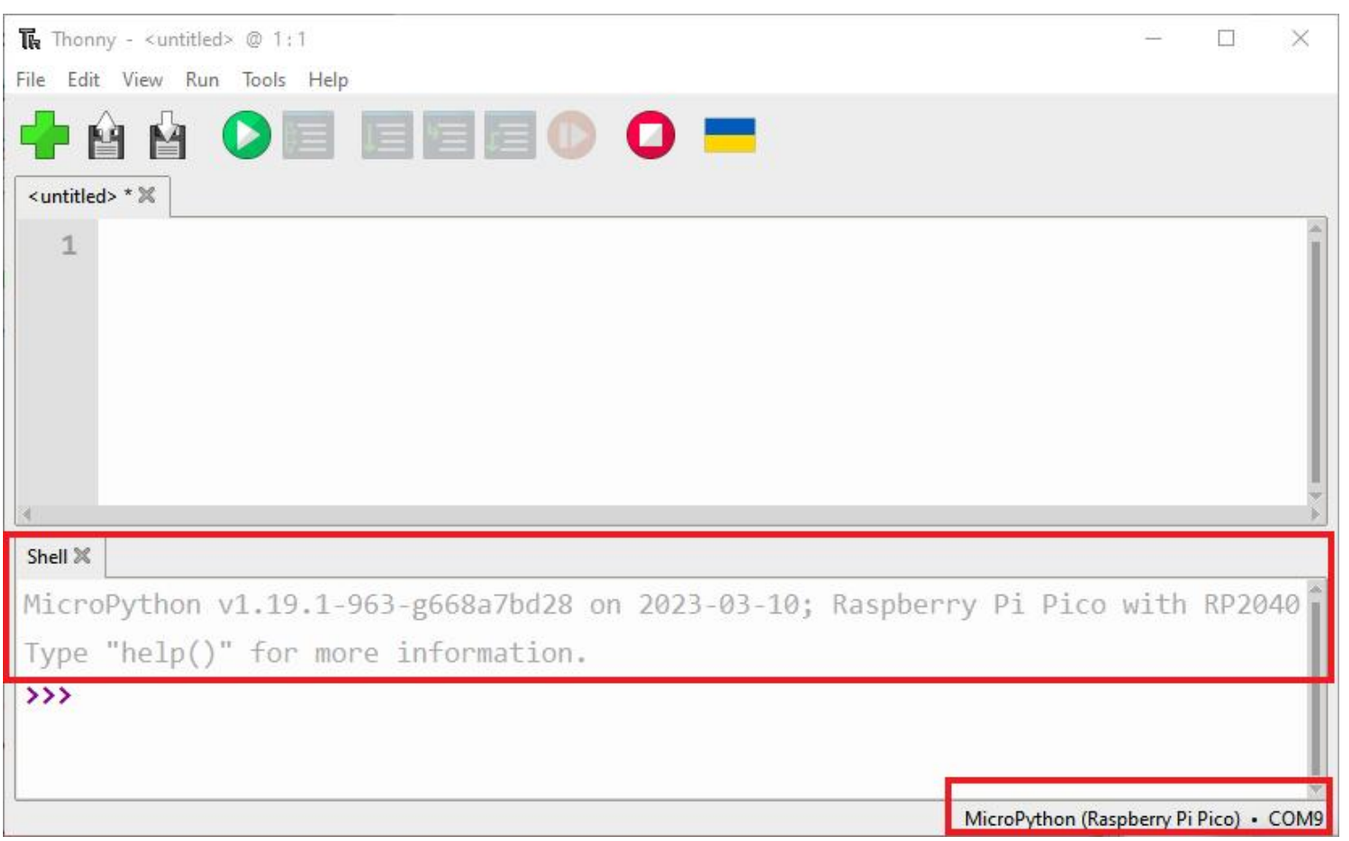
Upload ws2812.py to Raspberry Pi Pico(important)

In order for the LED Strip control code to be compiled successfully, the ws2812.py library file must be uploaded to the internal storage of the Raspberry Pi Pico.

Step①:Thonny Connected to Raspberry Pi Pico

Left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?

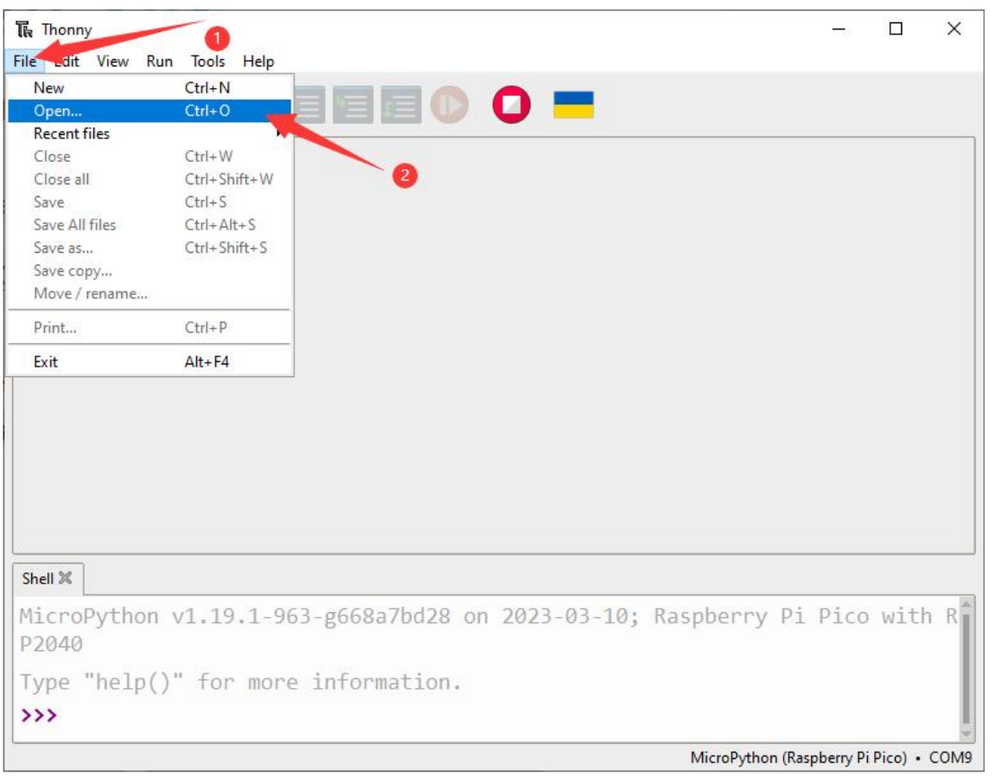

Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled> *
1
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspbe
rry Pi Pico with RP2040
Type "help()" for more information.
>>>
Local Python 3 • Thonny's Python
MicroPython (Raspberry Pi Pico) • COM9
MicroPython (RP2040) • COM9
Configure interpreter...
MicroPython (Raspberry Pi Pico) • COM9
2.click
1.click

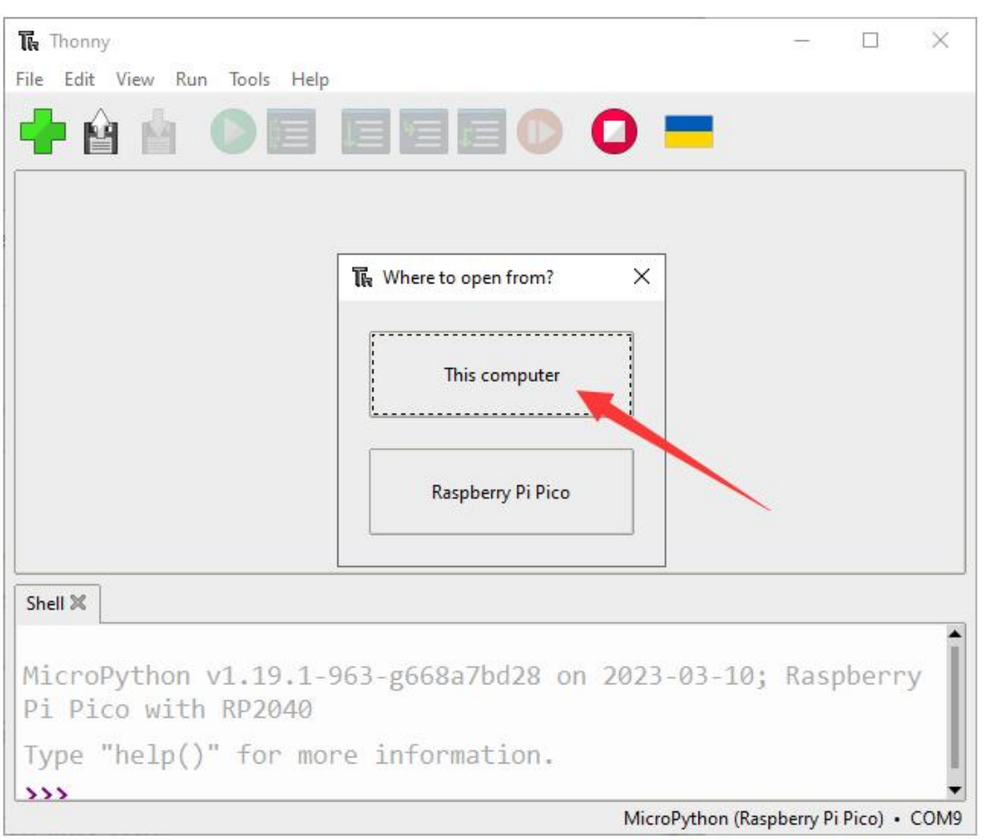

Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled> *
1
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico) • COM9

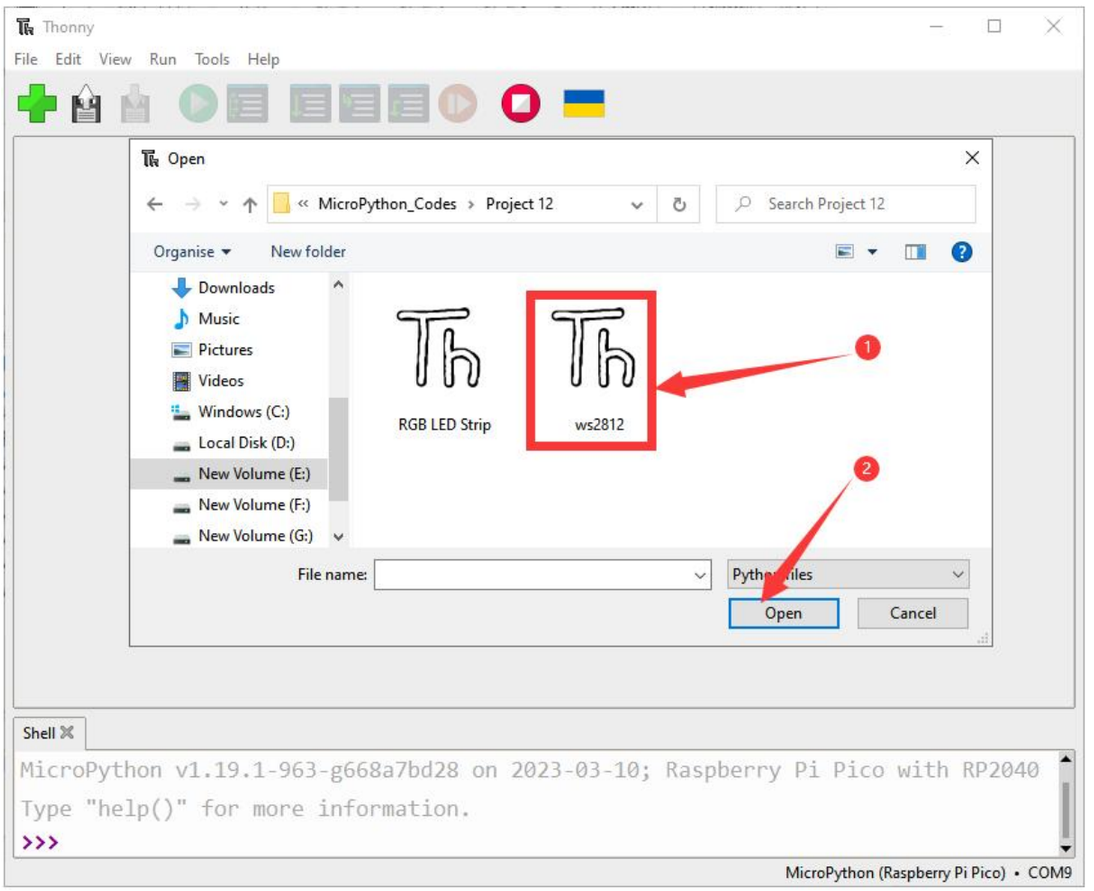
Step②:Open ws2812.py

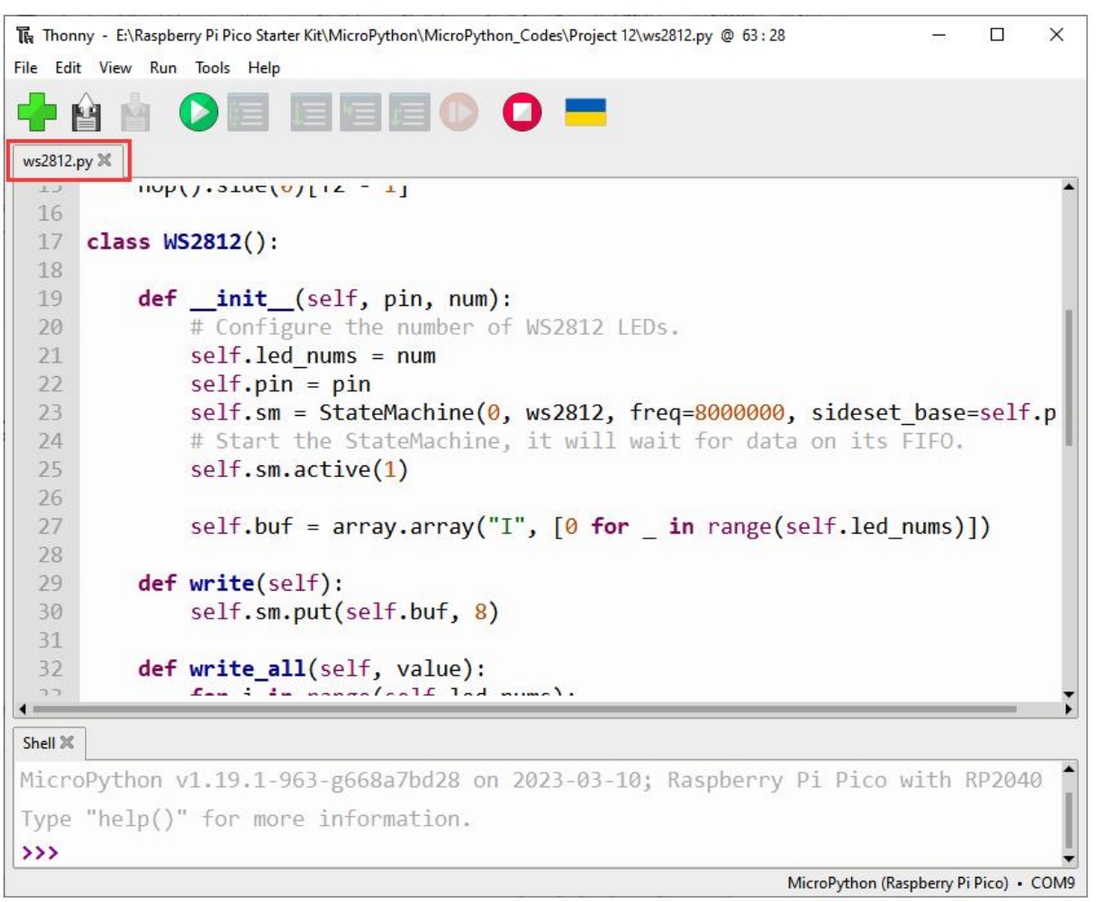
Click the File>>open icon to open Project 12 \ ws2812.py file.If you have downloaded the tutorial. You can find the .py file in This computer.

File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

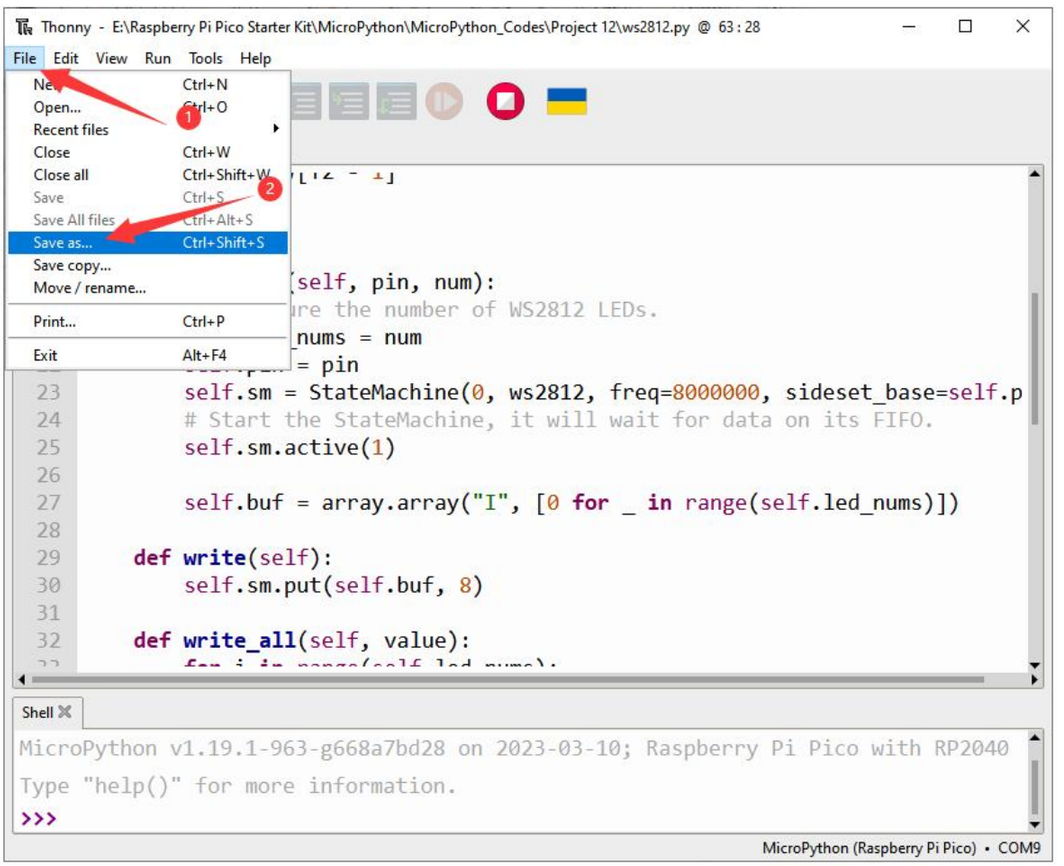


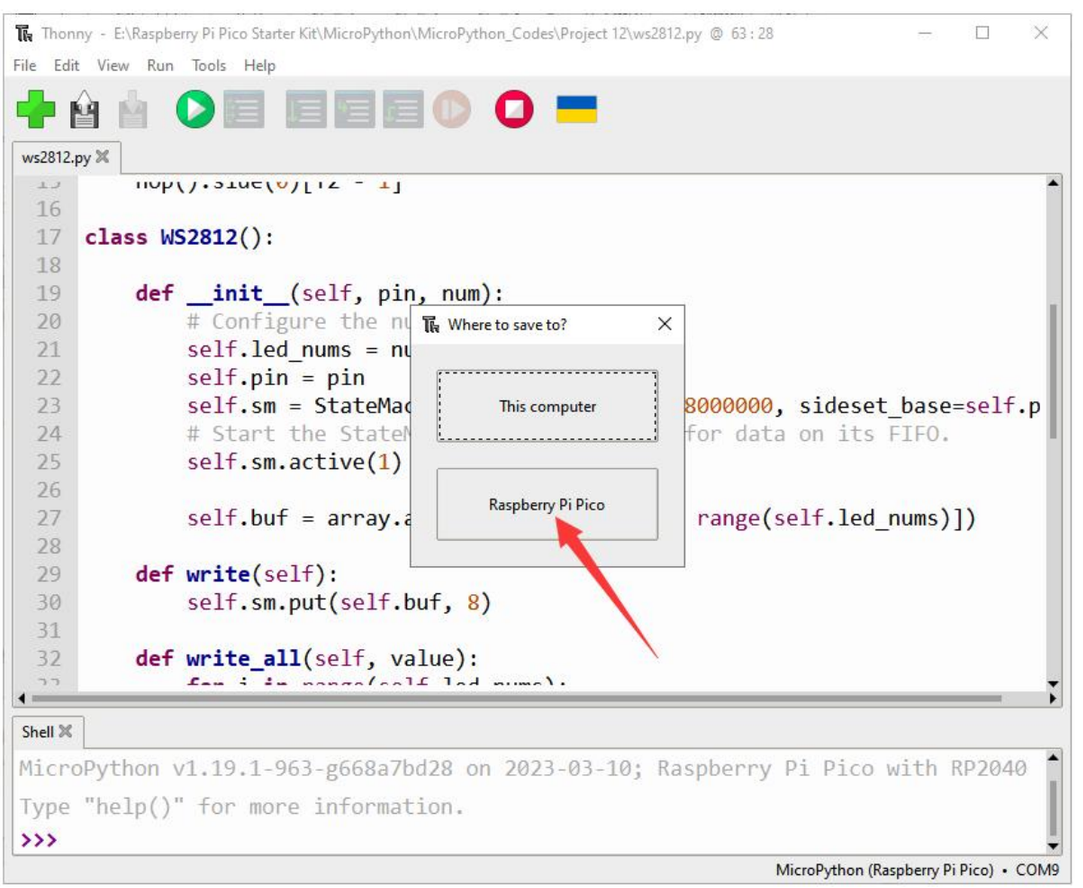


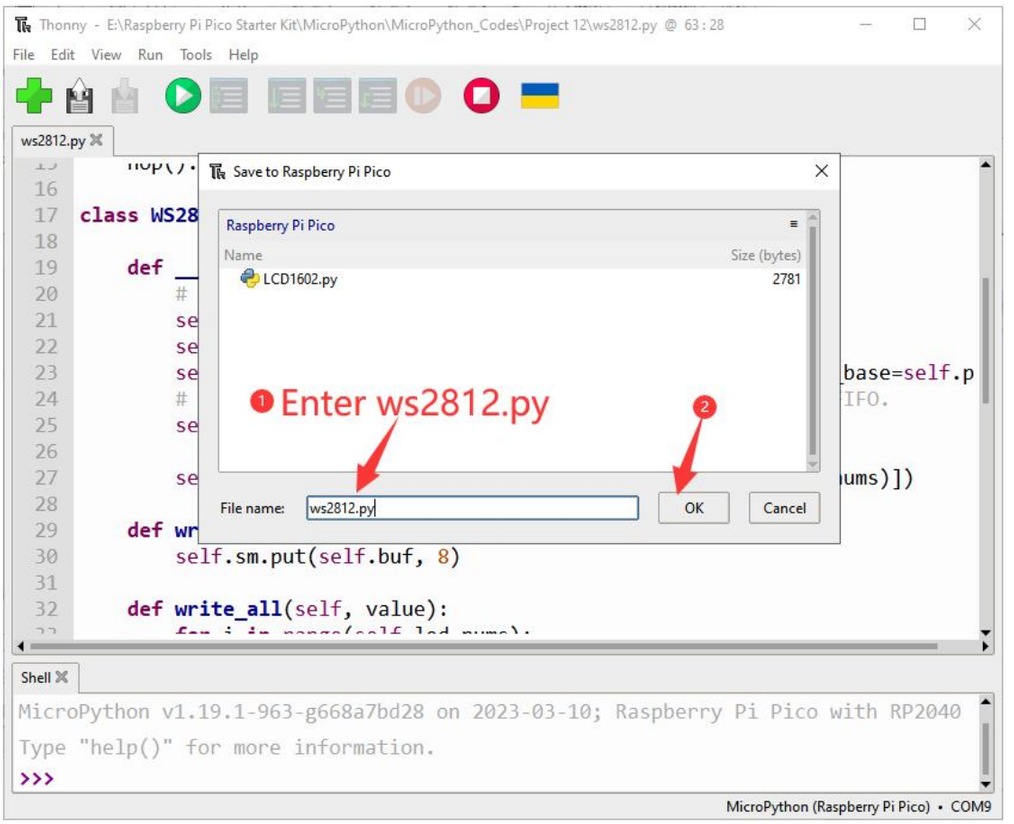

Thonny
File Edit View Run Tools Help
Open
MicroPython_Codes › Project 12
Search Project 12
Organise
New folder
Downloads
Music
Pictures
Videos
Windows (C:)
Local Disk (D:)
New Volume (E:)
New Volume (F:)
New Volume (G:)
RGB LED Strip
ws2812
1
2
File name:
Open
Cancel
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico) • COM9


Thonny - E:\Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes\Project 12\ws2812.py @ 63 : 28
File Edit View Run Tools Help
ws2812.py
class WS2812():
def __init__(self, pin, num):
# Configure the number of WS2812 LEDs.
self.led_nums = num
self.pin = pin
self.sm = StateMachine(0, ws2812, freq=8000000, sideset_base=self.p
# Start the StateMachine, it will wait for data on its FIFO.
self.sm.active(1)
self.buf = array.array("I", [0 for _ in range(self.led_nums)])
def write(self):
self.sm.put(self.buf, 8)
def write_all(self, value):
Shell
MicroPython v1.19.1-963-g668a7bd28 on 2023-03-10; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico) • COM9

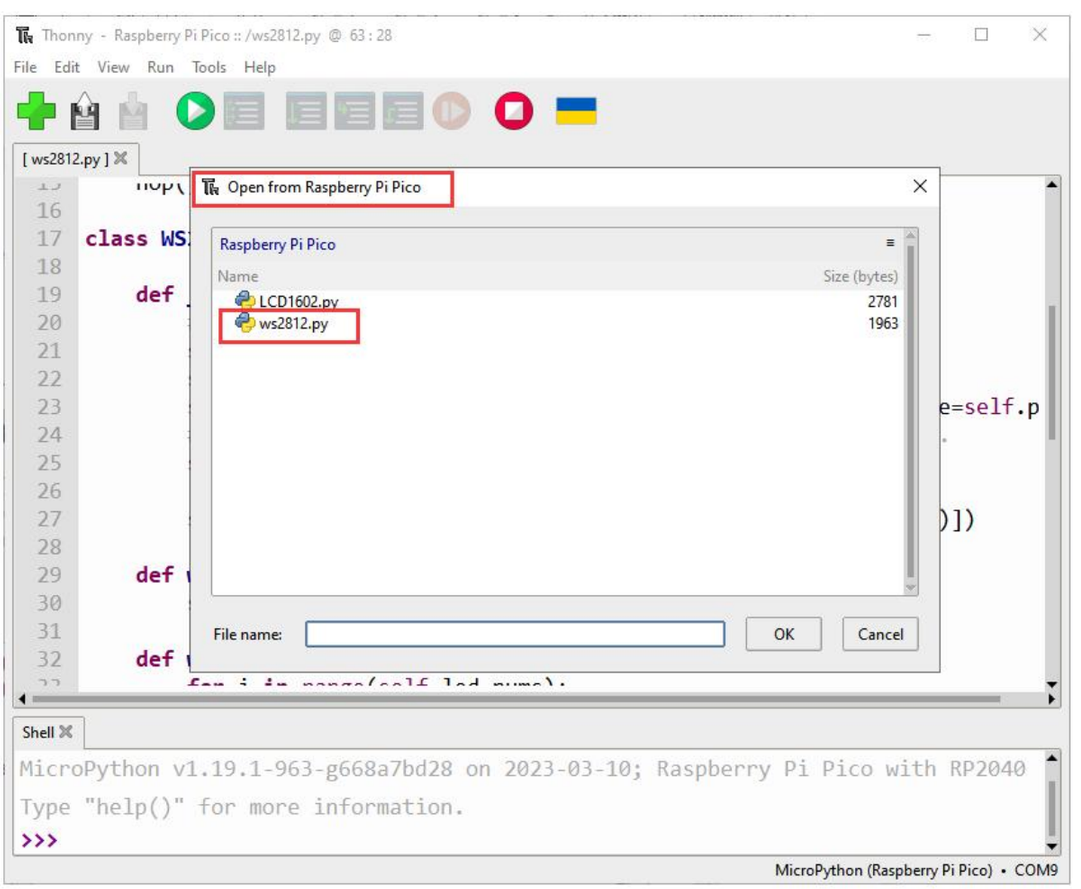
Step③:Save as LCD1602.py to Raspberry Pi Pico







If the library file is successfully saved to the Pi Pico memory, click File>Open>open from Raspberry Pi Pico, and you will see ws2812.py in the Pi Pico memory.



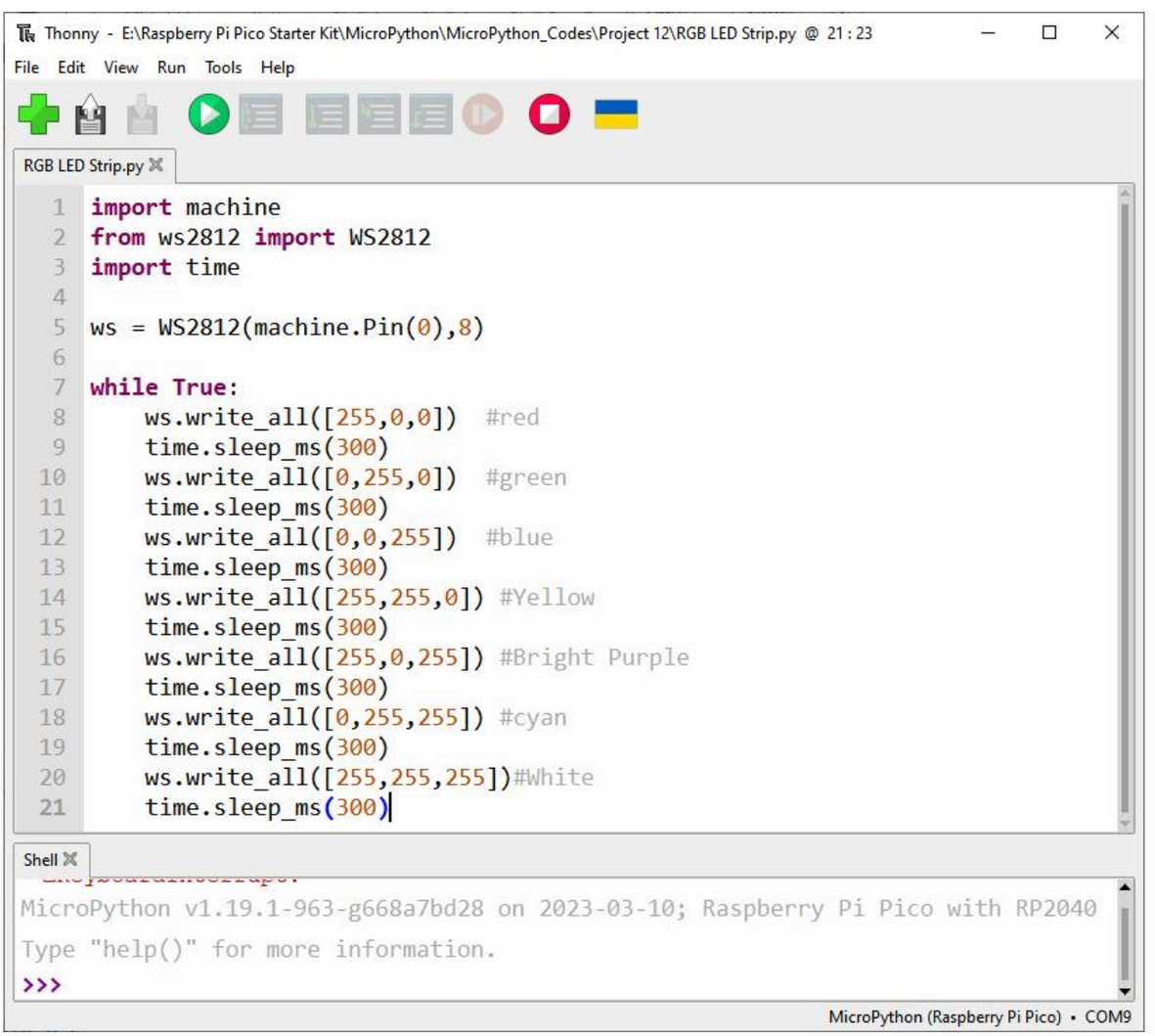
Code

Click the File>>open icon to open Project 12 \ RGB_LED_Strip.py file.If you have downloaded the tutorial. You can find the .py file.

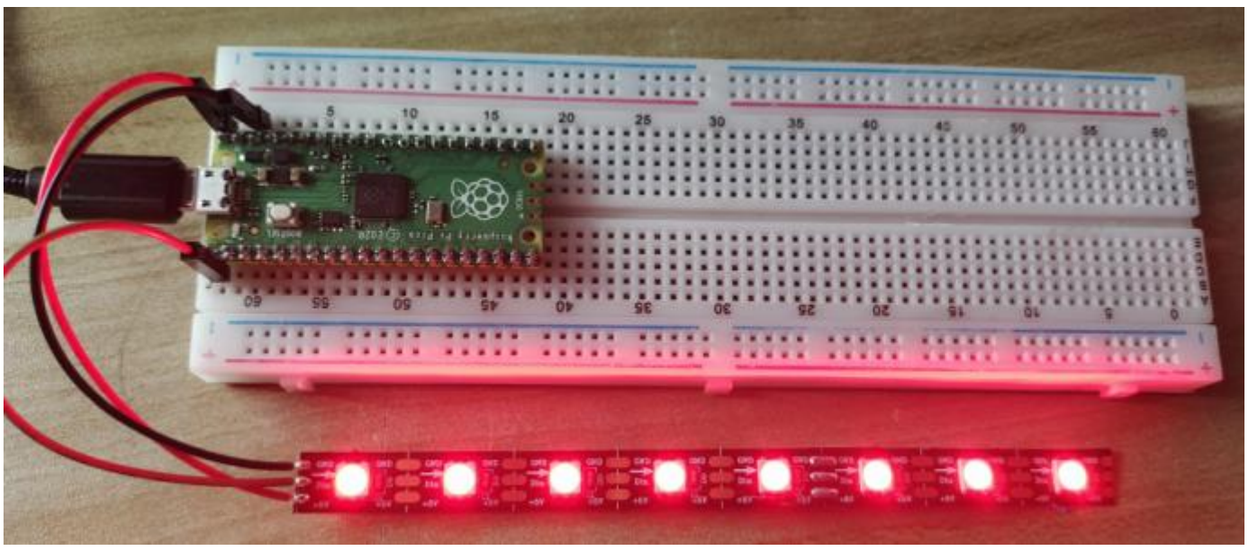
File path :Raspberry Pi Pico Starter Kit\MicroPython\MicroPython_Codes

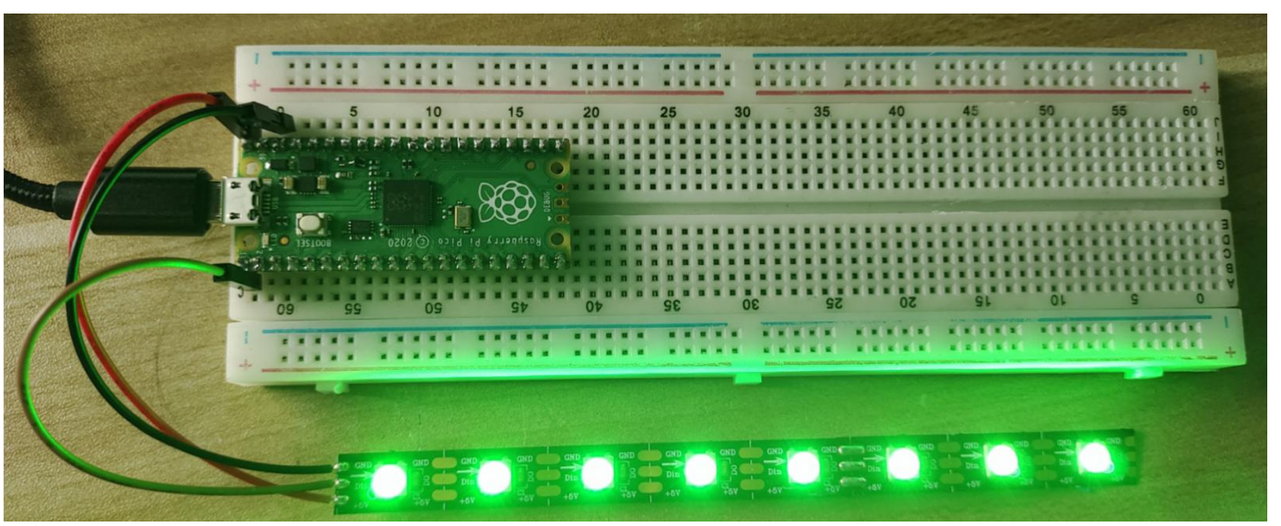
Note:

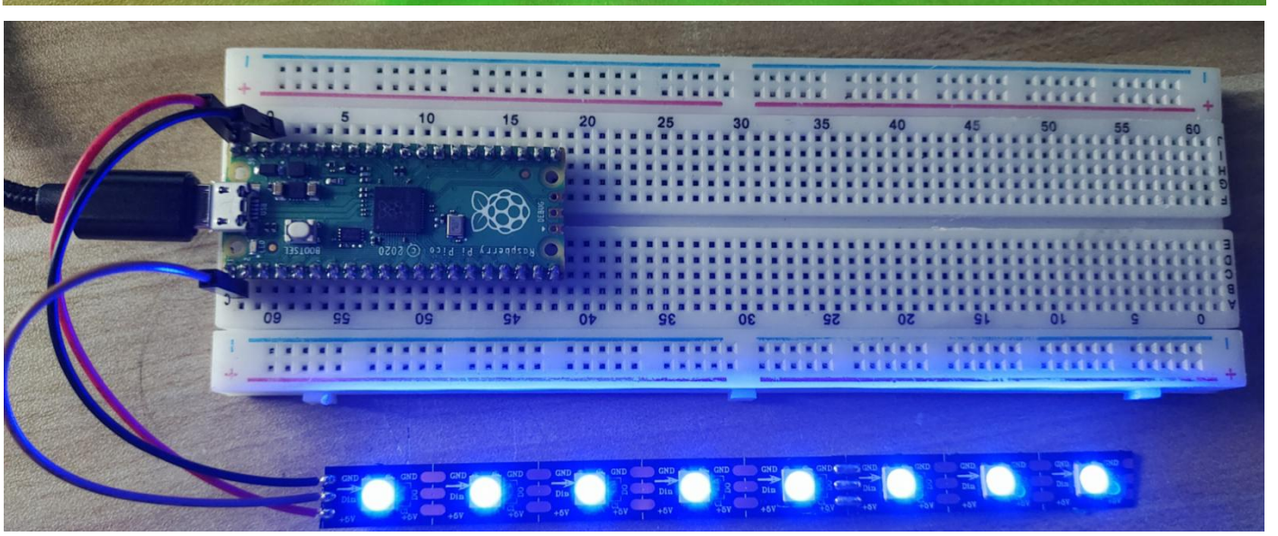
Don’t forget to left click on the bottom right corner and switch the interpreter version name to MicroPython (Raspberry Pi Pico) ● COMx.Have Question?



Click “Run current script”  .LED light bar display color: red->green->blue->Yellow->Purple->cyan->White, changing another color every 300ms.







How it works?

In the ws2812 library, we have integrated related functions into the WS2812 class.

You can use the RGB LED Strip with the following statement.

```
from ws2812 import WS2812
```

Declare a WS2812 type object, named “ws”, it is connected to “pin”, there are “number” RGB LEDs on the WS2812 strip.

```
ws = WS2812(pin,number)
```

ws is an array object, each element corresponds to one RGB LED on the WS2812 strip, for example, ws[0] is the first one, ws[7] is the eighth.

We can assign color values to each RGB LED, these values must be 24-bit color (represented with six hexadecimal digits) or list of 3 8-bit RGB.

For example, the red value is “0xFF0000” or “[255,0,0]”.

```
ws[i] = color value
```

Then use this statement to write the color for the LED Strip and light it up.

```
ws.write()
```

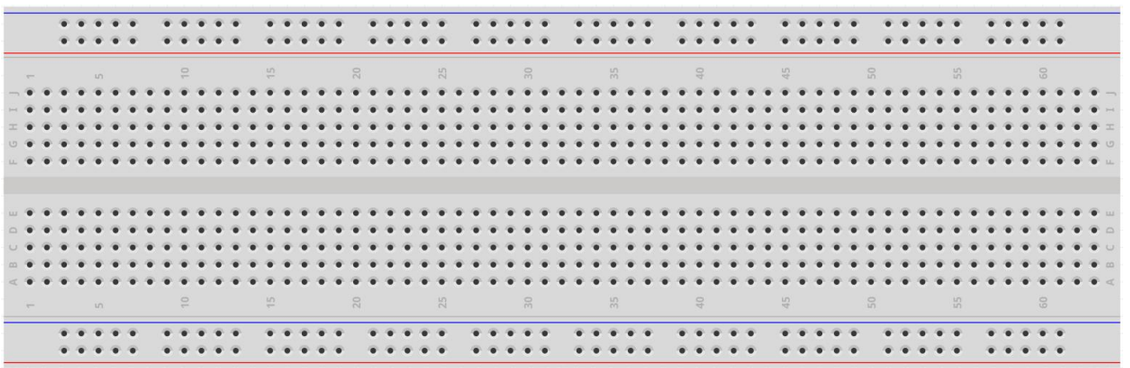
```
ws.write_all(color value)
```

You can also directly use the following statement to make all LEDs light up the same color.

Components Introduction

- Breadboard
- LED
- Button
- RGB LED
- Resistor
- Transistor
- Buzzer
- Potentiometer
- IR Proximity Sensor Module
- Photoresistor
- Thermistor
- Tilt Switch
- Servo
- I2C LCD1602
- PIR Motion Sensor
- WS2812 RGB 8 LEDs Strip

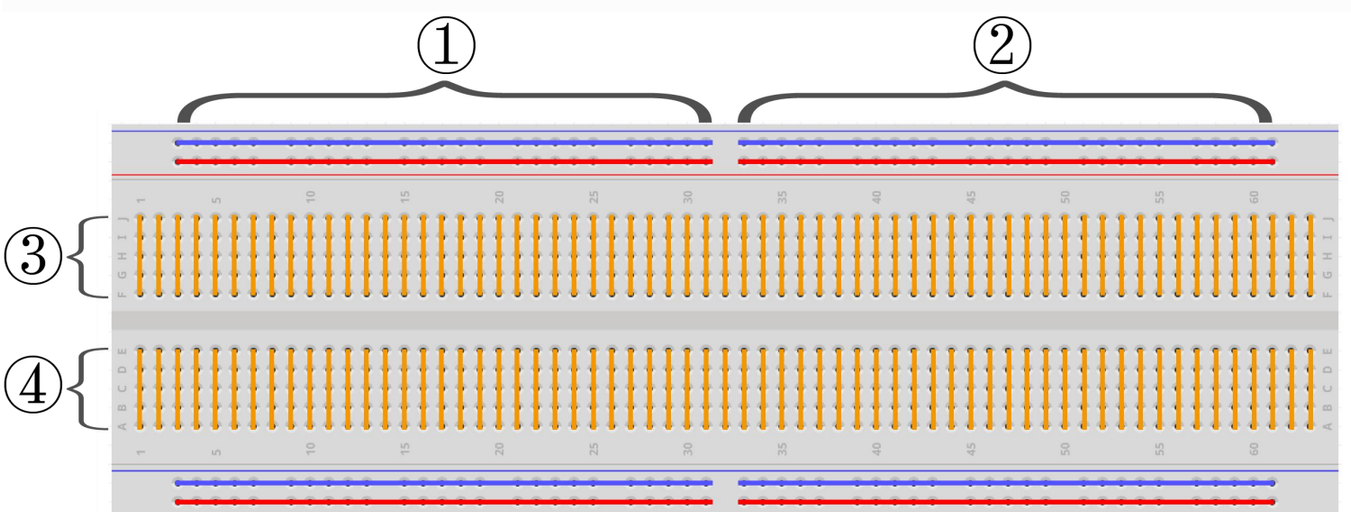
➢ Breadboard



A breadboard is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used for slicing bread.[1] In the 1970s the solderless breadboard (a.k.a. plugboard, a terminal array board) became available and nowadays the term “breadboard” is commonly used to refer to these.

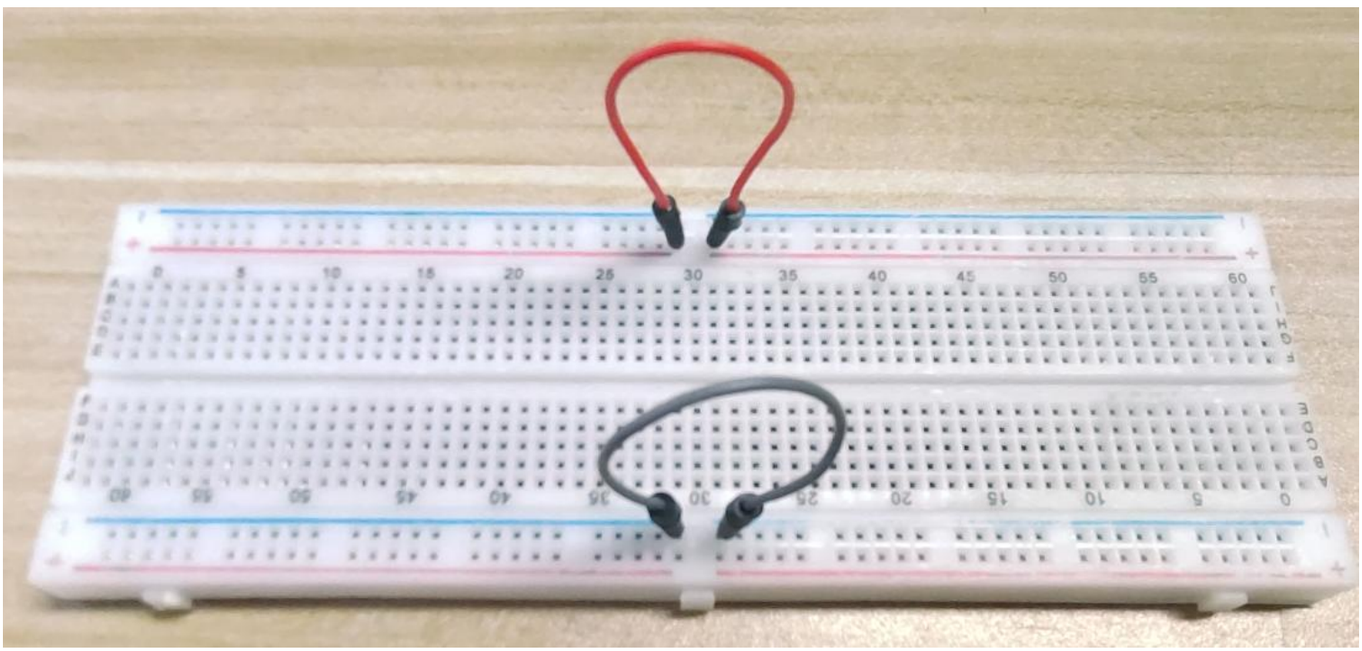
It is used to build and test circuits quickly before finishing any circuit design. And it has many holes into which components mentioned above can be inserted like ICs and resistors as well as jumper wires. The breadboard allows you to plug in and remove components easily.

The picture shows the internal structure of a breadboard. Although these holes on the breadboard appear to be independent of each other, they are actually connected to each other through metal strips internally.



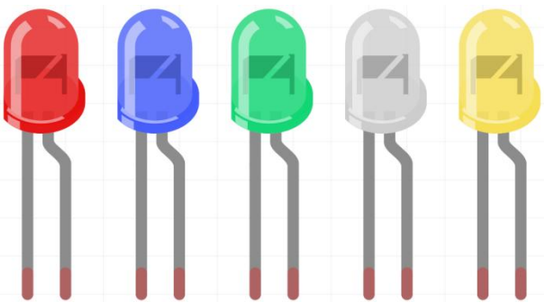
Note:

Area ① and area ② are not connected, and area ① and area ② can be connected through a jumper.

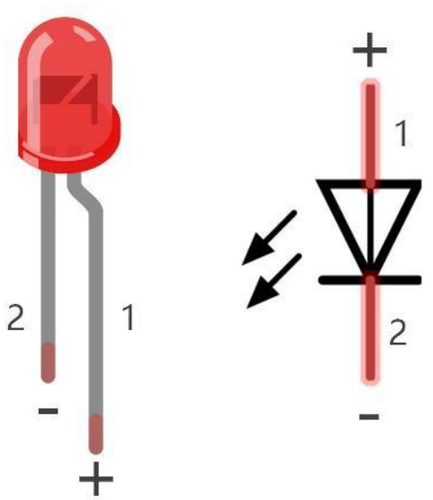


If you want to know more about breadboard, refer to: How to Use a Breadboard - Science Buddies

➢ LED



An LED is a type of diode. All diodes only work if current is flowing in the correct direction and have two Poles.An LED will only work (light up) if the longer pin (+) of LED is connected to the positive output from a power source and the shorter pin is connected to the negative (-). Negative output is also referred to as Ground (GND). This type of component is known as “Polar” (think One-Way Street).All common 2 lead diodes are the same in this respect. Diodes work only if the voltage of its positive electrode is higher than its negative electrode and there is a narrow range of operating voltage for most all common diodes of 1.9 and 3.4V. If you use much more than 3.3V the LED will be damaged and burn out.



Note:

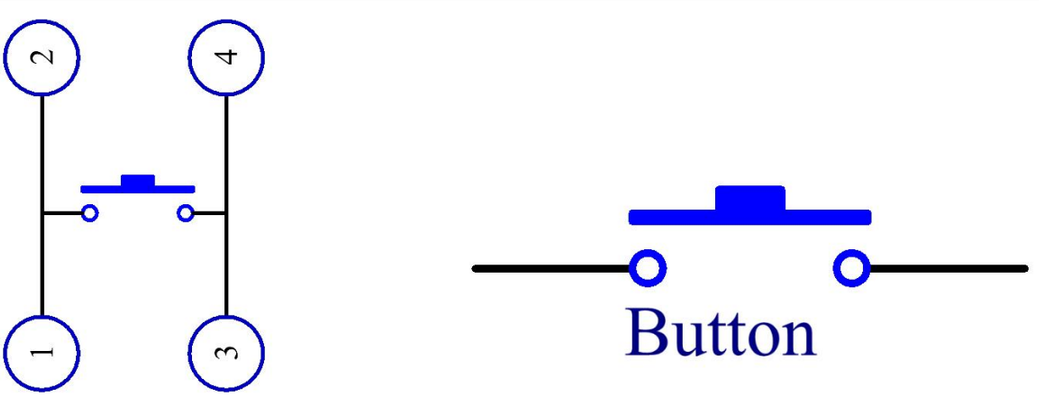
LEDs cannot be directly connected to a power supply, which usually ends in a damaged component. A

resistor with a specified resistance value must be connected in series to the LED you plan to use.

➢ Button

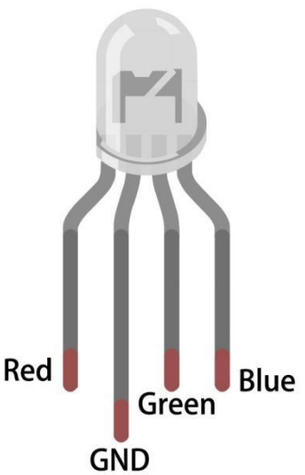
Buttons are a common component used to control electronic devices. They are usually used as switches to connect or break circuits. Although buttons come in a variety of sizes and shapes, the one used here is a 6mm mini-button as shown in the following pictures. Pin 1 is connected to pin 2 and pin 3 to pin 4. So you just need to connect either of pin 1 and pin 2 to pin 3 or pin 4.

The following is the internal structure of a button. The symbol on the right below is usually used to represent a button in circuits.

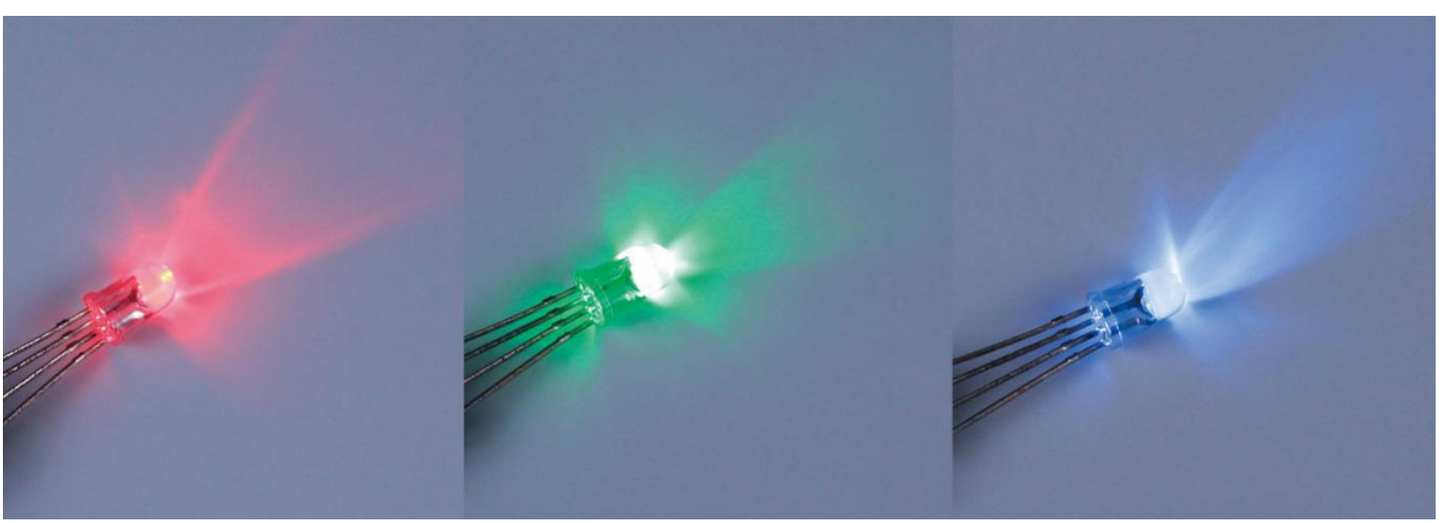


Since the pin 1 is connected to pin 2, and pin 3 to pin 4, when the button is pressed, the 4 pins are connected, thus closing the circuit.

➢ RGB LED

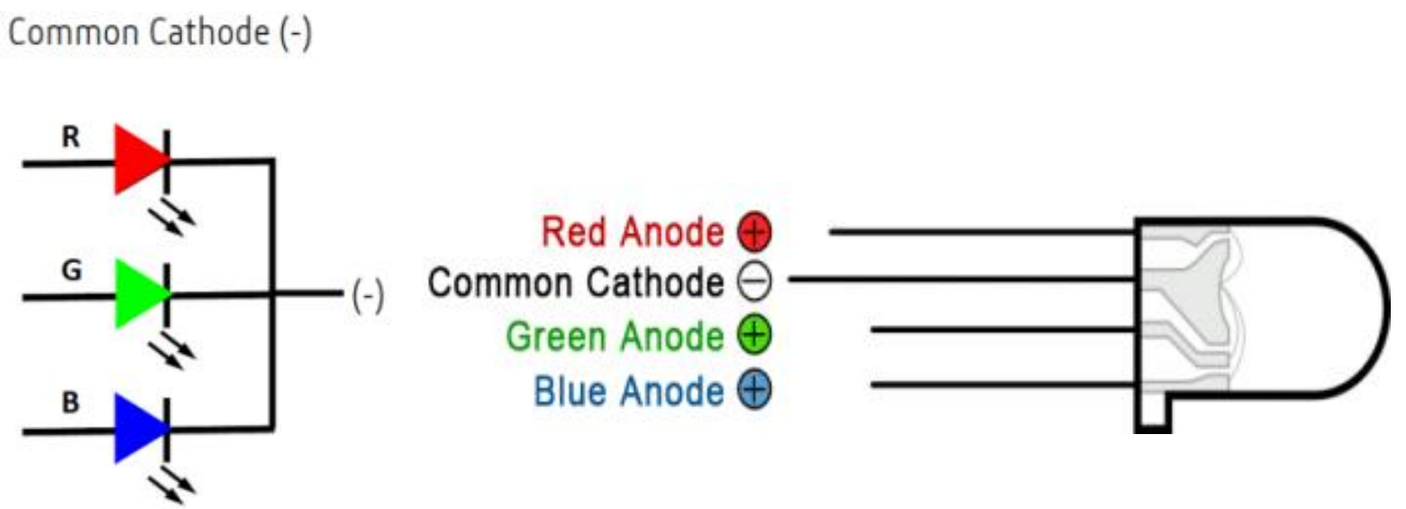


RGB LEDs emit light in various colors. An RGB LED packages three LEDs of red, green, and blue into a transparent or semitransparent plastic shell. It can display various colors by changing the input voltage of the three pins and superimpose them, which, according to statistics, can create 16,777,216 different colors.



RGB LEDs can be categorized into common anode and common cathode ones. In this kit, the latter is used. The common cathode, or CC, means to connect the cathodes of the three LEDs. After you connect it with GND and plug in the three pins, the LED will flash the corresponding color.

Its circuit symbol is shown as figure.

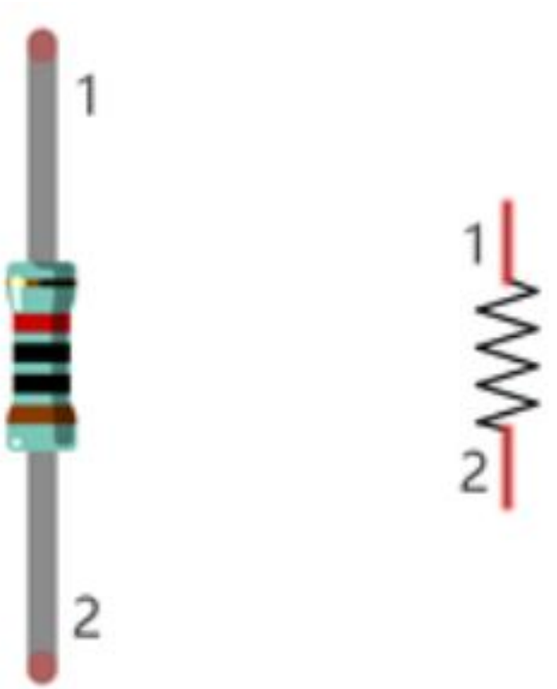


An RGB LED has 4 pins: the longest one is GND; the others are Red, Green and Blue. Touch its plastic shell and you will find a cut. The pin closest to the cut is the first pin, marked as Red, then GND, Green and Blue in turn.

➢ Resistor

Resistors use Ohms (Ω) as the unit of measurement of their resistance (R). 1MΩ=1000kΩ, 1kΩ=1000Ω.

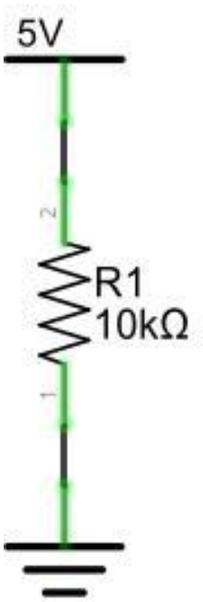
A resistor is a passive electrical component that limits or regulates the flow of current in an electronic circuit.On the left, we see a physical representation of a resistor, and the right is the symbol used to represent the presence of a resistor in a circuit diagram or schematic.



The bands of color on a resistor is a shorthand code used to identify its resistance value. For more details of resistor color codes, please refer to the appendix of this tutorial.

With a fixed voltage, there will be less current output with greater resistance added to the circuit. The relationship between Current, Voltage and Resistance can be expressed by this formula: I=V/R known as Ohm’ s Law where I = Current, V = Voltage and R = Resistance. Knowing the values of any two of these allows you to solve the value of the third.

In the following diagram, the current through R1 is: I=U/R=5V/10kΩ=0.0005A=0.5mA.



WARNING:

Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

Note:

Unlike LEDs and Diodes, Resistors have no poles and re non-polar (it does not matter which direction you insert them into a circuit, it will work the same)

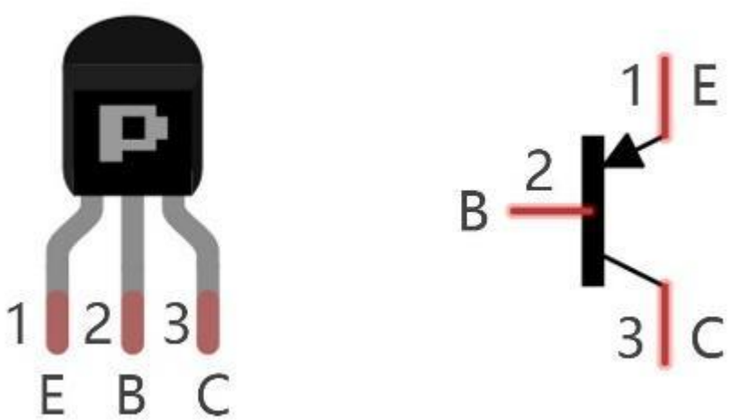
➢ Transistor

Because the buzzer requires such large current that GP of Raspberry Pi Pico output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

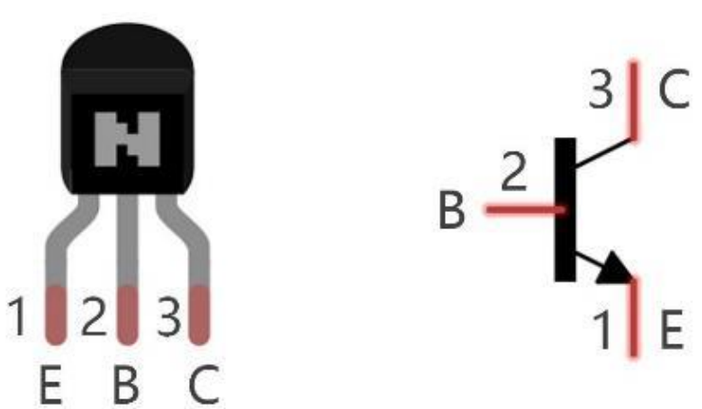
Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistor

can be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector (c) and emitter (e). When there is current passing between "be", "ce" will allow several-fold current (• t between "be" exceeds a certain value, "ce" will not allow current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types as shown below: PNP and NPN.

PNP transistor



NPN transistor

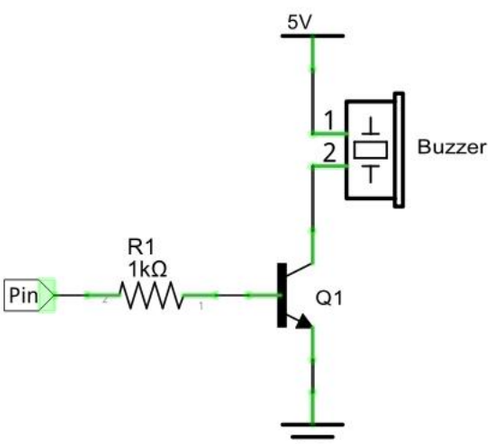


In our kit, There are two NPN transistor is marked with 8050.

Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

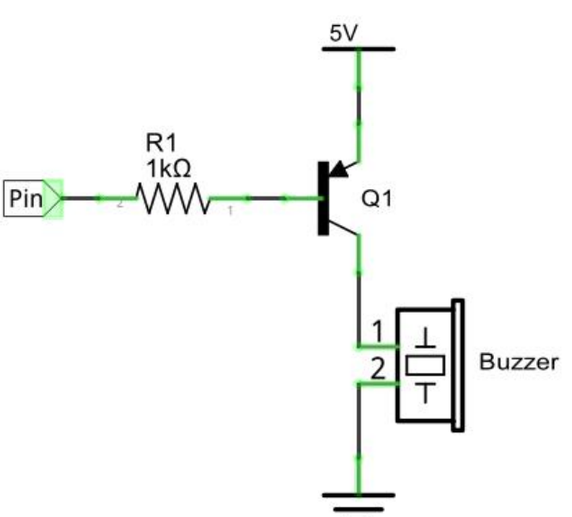
- When using NPN transistor to drive buzzer, we often adopt the following method. If GP outputs high level,current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GP outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.

NPN transistor to drive buzzer



- When using PNP transistor to drive buzzer, we often adopt the following method. If GP outputs low level,current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GP outputs high level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.

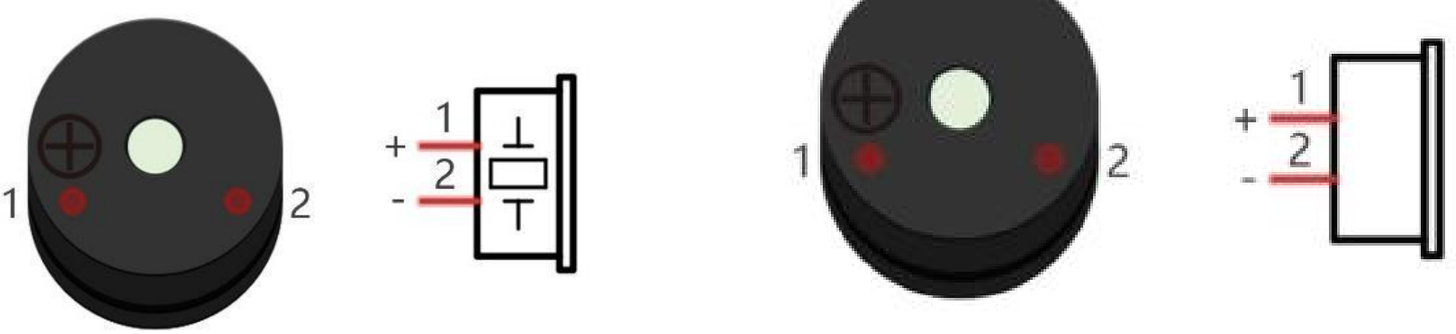
PNP transistor to drive buzzer



➢ Buzzer

Buzzer is a sounding component, which is widely used in electronic devices such as calculator, electronic warning clock and alarm. Buzzer has two types: active and passive. Active buzzer has oscillator inside, which will sound as long as it is supplied with power. Passive buzzer requires external oscillator signal (generally use PWM with different frequency) to make a sound.

Active buzzer **Passive buzzer**



Active buzzer is easy to use. Generally, it can only make a specific frequency of sound. Passive buzzer requires an external circuit to make a sound, but it can be controlled to make a sound with different frequency. The resonant frequency of the passive buzzer is 2kHz, which means the passive buzzer is loudest when its resonant frequency is 2kHz.

How to identify active and passive buzzer?

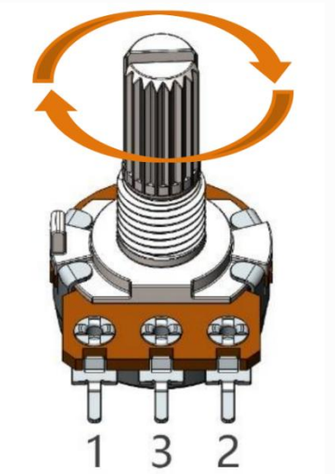
1. Usually, there is a label on the surface of active buzzer covering the vocal hole, but this is not an absolute judgment method.

2. Active buzzers are more complex than passive buzzers in their manufacture. There are many circuits and crystal oscillator elements inside active buzzers; all of this is usually protected with a waterproof coating(and a housing) exposing only its pins from the underside. On the other hand, passive buzzers do not have protective coatings on their underside. From the pin holes viewing of a passive buzzer, you can see the circuit board, coils, and a permanent magnet (all or any combination of these components depending on the model.

Active buzzer

Passive buzzer

➢ Potentiometer

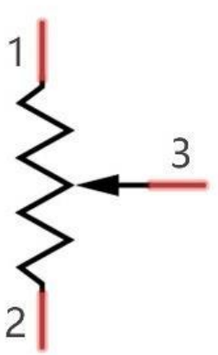


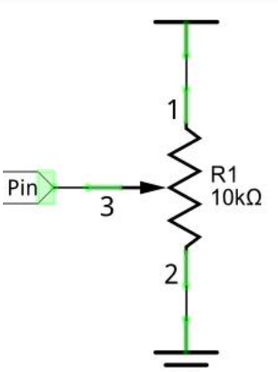
Potentiometer is also a resistance component with 3 terminals and its resistance value can be adjusted according to some regular variation.

Potentiometers come in various shapes, sizes, and values, but they all have the following things in common:

- They have three terminals (or connection points).
- They have a knob, screw, or slider that can be moved to vary the resistance between the middle terminal and either one of the outer terminals.
- The resistance between the middle terminal and either one of the outer terminals varies from 0 Ω to the maximum resistance of the pot as the knob, screw, or slider is moved.

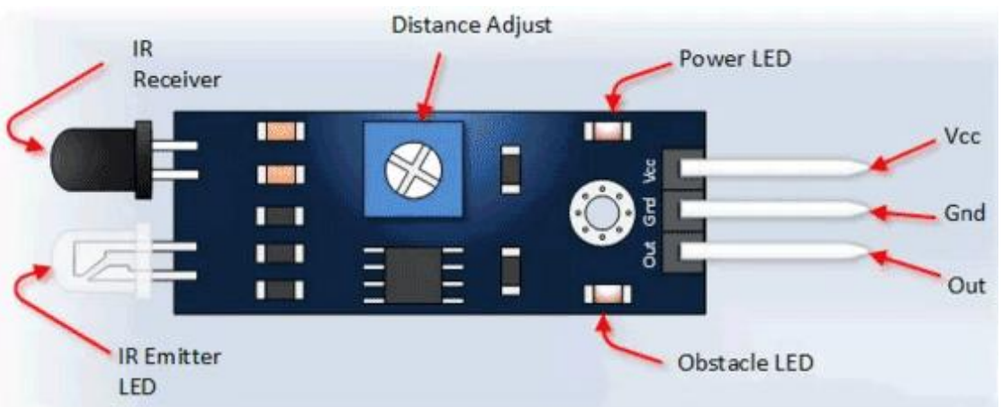
Here is the circuit symbol of potentiometer.10K ohm potentiometer is included in the kit





➢ IR Proximity Sensor Module

Proximity Sensor are used to detect objects and obstacles in front of sensor. Sensor keeps transmitting infrared light and when any object comes near, it is detected by the sensor by monitoring the reflected light from the object. It can be used in robots for obstacle avoidance, for automatic doors, for parking aid devices or for security alarm systems, or contact less tachometer by measuring RPM of rotation objects like fan blades.

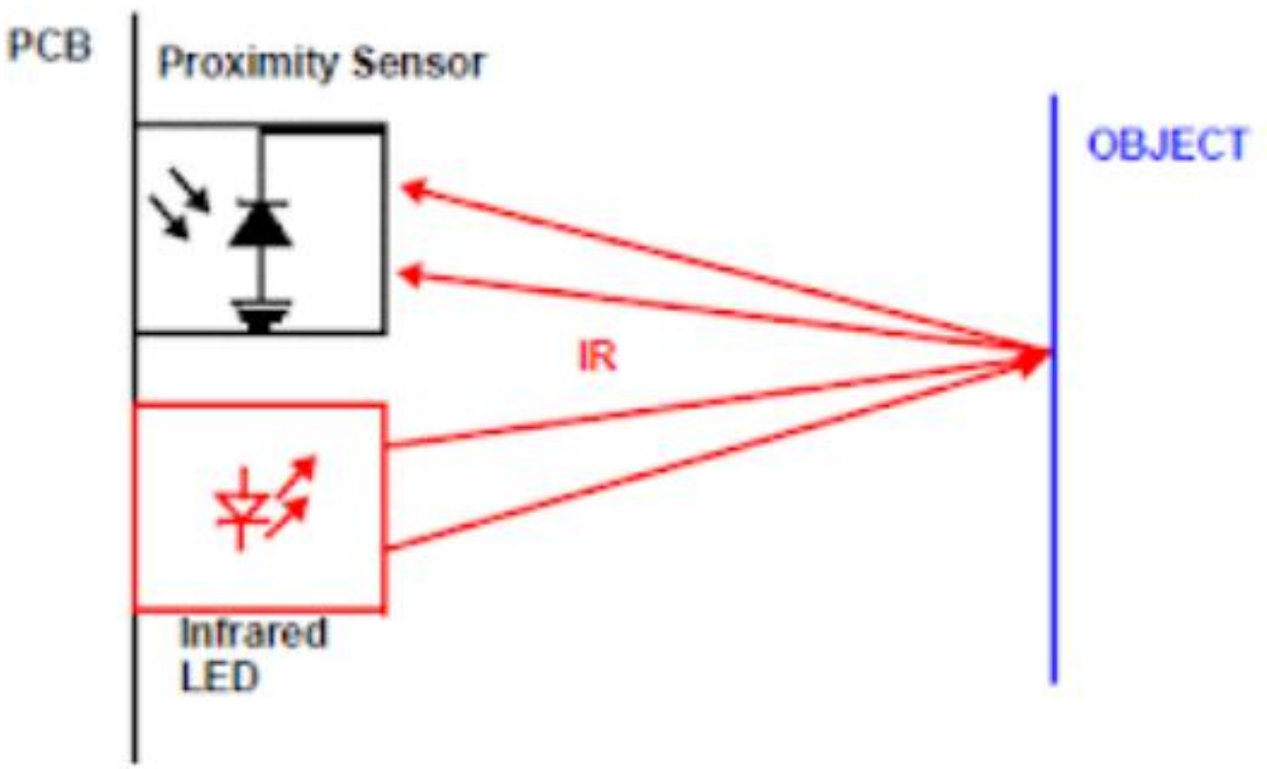


Pin, Control Indicator	Description
Vcc	3.3 to 5 Vdc Supply Input
Gnd	Ground Input
Out	Output that goes low when obstacle is in range
Power LED	Illuminates when power is applied
Obstacle LED	Illuminates when obstacle is detected
Distance Adjust	Adjust detection distance. CCW decreases distance. CW increases distance.
IR Emitter	Infrared emitter LED
IR Receiver	Infrared receiver that receives signal transmitted by Infrared emitter.

Digital OUTPUT

Digital **LOW** output on detecting objects in front.

Digital **HIGH** output on detecting no objects in front.

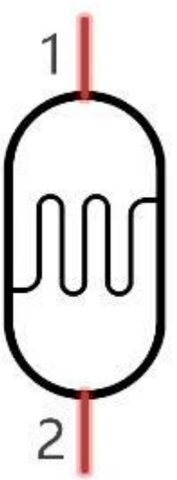


Distance Adjust

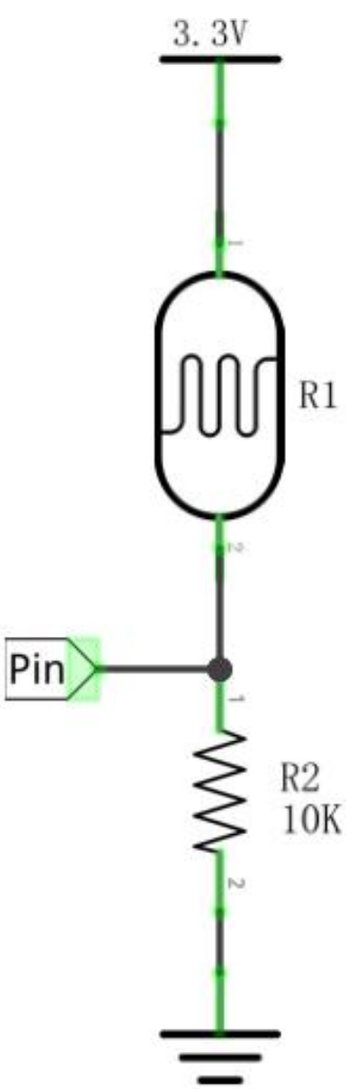
IR sensors are highly susceptible to ambient light and the IR sensor on this sensor is suitably covered to reduce effect of ambient light on the sensor. To For maximum, range the on board potentiometer should be used to calibrate the sensor. To set the potentiometer, use a screw driver and turn the potentiometer till the output LED just turns off.

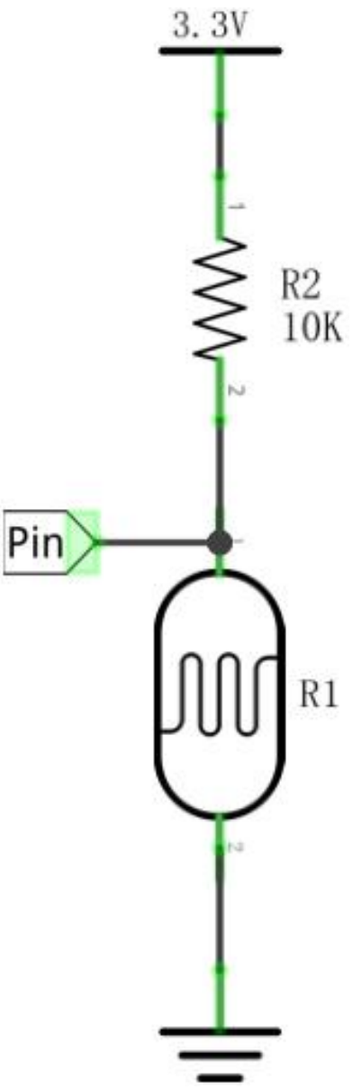
➢ Photoresistor

Photoresistor is simply a light sensitive resistor. It is an active component that decreases resistance with respect to receiving luminosity (light) on the component's light sensitive surface. Photoresistor’s resistance value will change in proportion to the ambient light detected. With this characteristic, we can use a Photoresistor to detect light intensity. The Photoresistor and its electronic symbol are as follows.



The circuit below is used to detect the change of a Photoresistor’s resistance value:





In the above circuit, when a Photoresistor's resistance value changes due to a change in light intensity, the voltage between the Photoresistor and Resistor R1 will also change. Therefore, the intensity of the light can be obtained by measuring this voltage.

➢ Thermistor

A Thermistor is a temperature sensitive resistor. When it senses a change in temperature, the resistance of the Thermistor will change. We can take advantage of this characteristic by using a Thermistor to detect temperature intensity. A Thermistor and its electronic symbol are shown below.

➢ Thermistor - Wikipedia

The relationship between resistance value and temperature of a thermistor is:

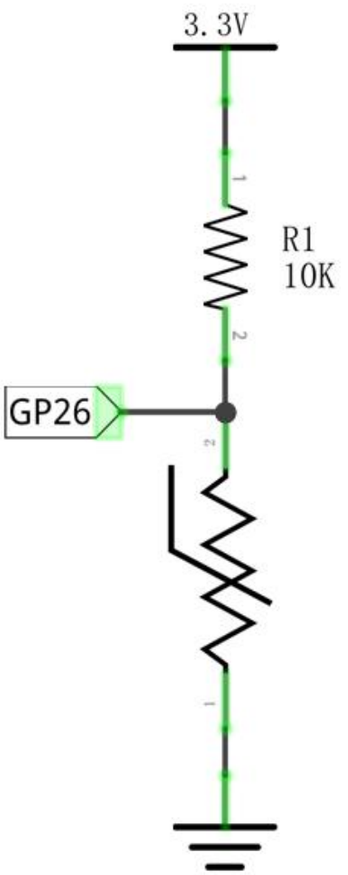
$$Rt = R * EXP[\ B * (\frac{1}{T2} - \frac{1}{T1})\ ]$$

Where:

- **Rt** is the thermistor resistance under T2 temperature;
- **R** is the nominal resistance of thermistor under T1 temperature;
- **EXP[n]** is nth power of e;
- **B** is for thermal index;
- **T1**, **T2** is Kelvin temperature (absolute temperature). Kelvin temperature=273.15 + Celsius temperature.

For the parameters of the Thermistor, we use: B=3950, R=10kΩ, T1=25℃.

The circuit connection method of the Thermistor is similar to photoresistor, as the following:



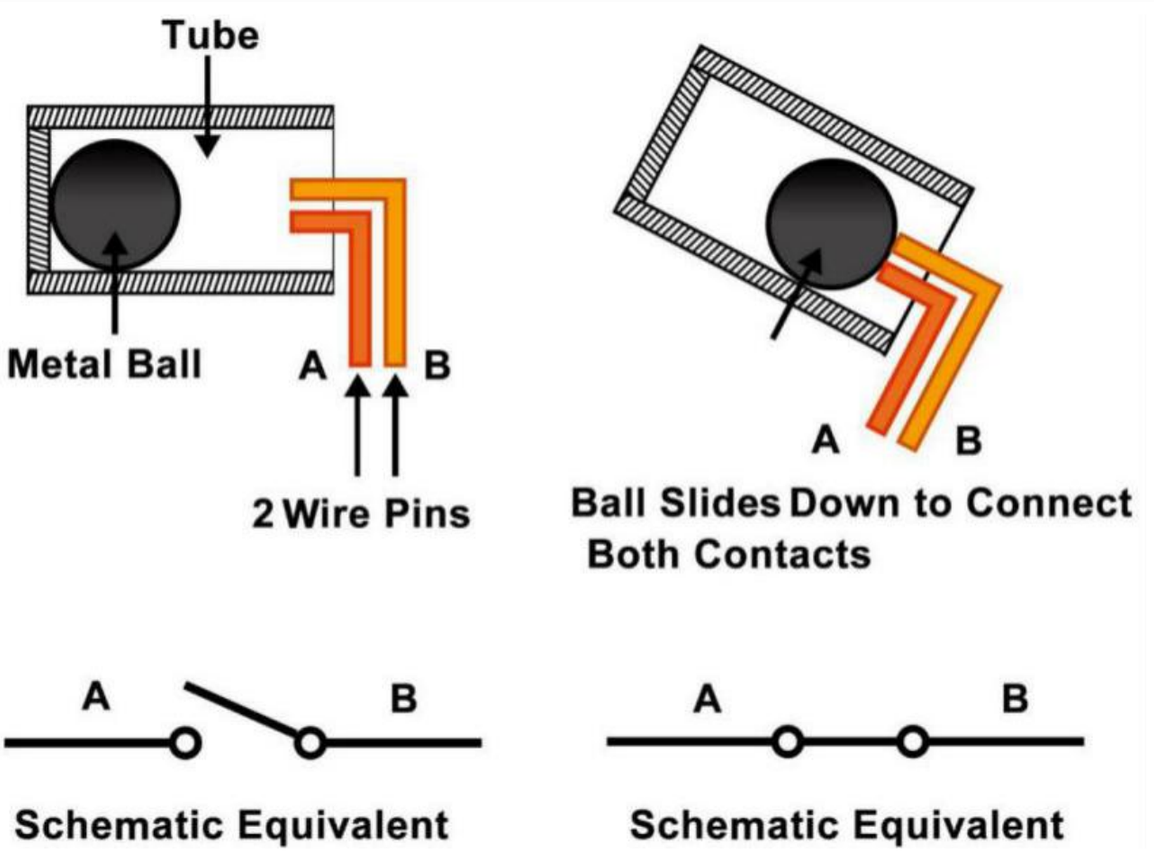
We can use the value measured by the ADC converter to obtain the resistance value of Thermistor, and then we can use the formula to obtain the temperature value.
Therefore, the temperature formula can be derived as:

$$T2 = 1/(\frac{1}{T1} + \ln(\frac{Rt}{R})/B)$$

➢ Tilt Switch

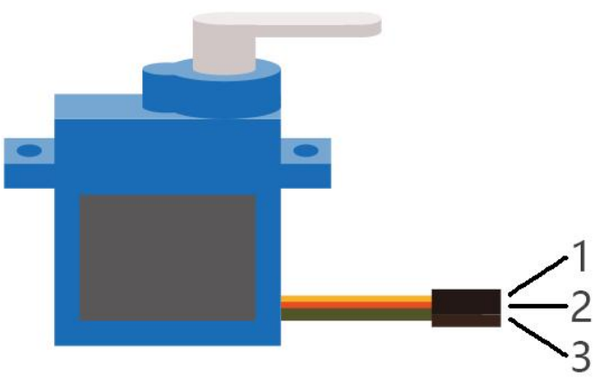
The tilt switch used here is a ball one with a metal ball inside. It is used to detect inclinations of a small angle.

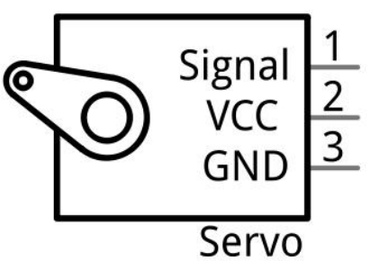
The principle is very simple. When the switch is tilted in a certain angle, the ball inside rolls down and touches the two contacts connected to the pins outside, thus triggering circuits. Otherwise the ball will stay away from the contacts, thus breaking the circuits.



➢ Servo

Servo is a compact package which consists of a DC Motor, a set of reduction gears to provide torque, a sensor and control circuit board. Most Servos only have a 180-degree range of motion via their “horn”. Servos can output higher torque than a simple DC Motor alone and they are widely used to control motion in model cars,model airplanes, robots, etc. Servos have three wire leads which usually terminate to a male or female 3-pin plug. Two leads are for electric power: Positive (2-VCC, Red wire), Negative (3-GND, Brown wire), and the signal line (1-Signal, Orange wire) as represented in the Servo provided in your Kit.





We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The time interval of 0.5ms-2.5ms of PWM single cycle high level corresponds to the Servo angle 0 degrees - 180 degrees linearly.

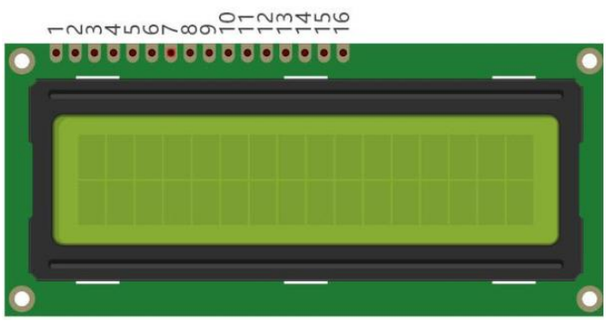
Part of the corresponding values are as follows:

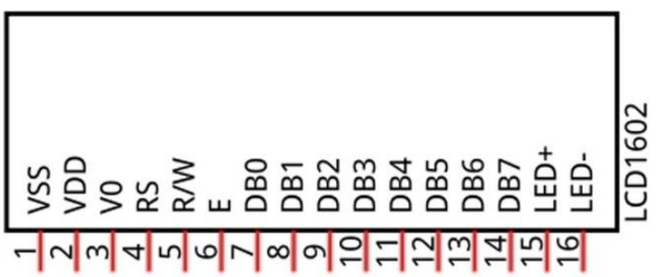
High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	0 degree
2ms	45 degree
2.5ms	180 degree

When you change the Servo signal value, the Servo will rotate to the designated angle.

➢ I2C LCD1602

The LCD1602 Display Screen can display 2 lines of characters in 16 columns. It is capable of displaying numbers,letters, symbols, ASCII code and so on. As shown below is a monochrome LCD1602 Display Screen along with its circuit pin diagram.

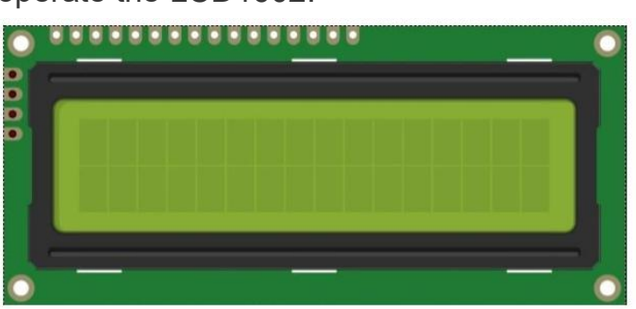


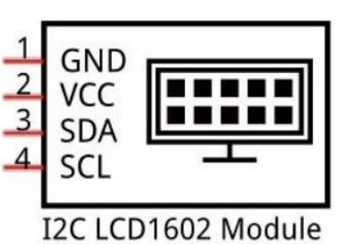


As we all know, though LCD and some other displays greatly enrich the man-machine interaction, they share a common weakness. When they are connected to a controller, multiple IOs will be occupied of the controller which has no so many outer ports. Also it restricts other functions of the controller. Therefore, LCD1602 with an I2C bus is developed to solve the problem.

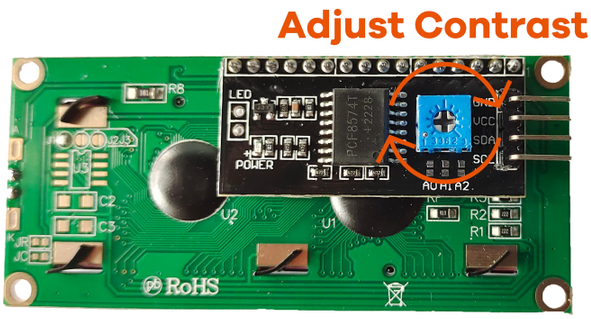
➢ Inter-Integrated Circuit - Wikipedia

I2C LCD1602 Display Screen integrates an I2C interface, which connects the serial-input & parallel-output module to the LCD1602 Display Screen. This allows us to use only 4 lines to the operate the LCD1602.





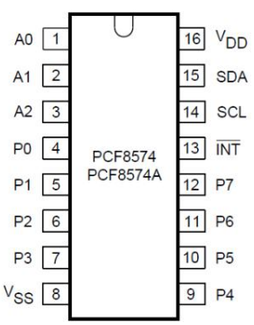
The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F).



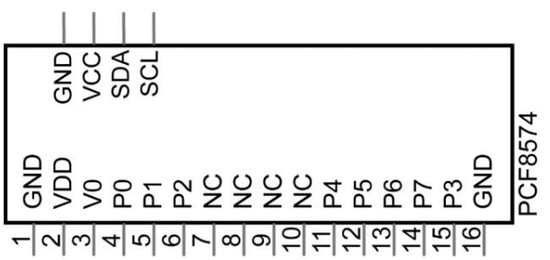
Potentiometer: It is used to adjust the contrast (the clarity of the displayed text), which is increased in the clockwise direction and decreased in the counterclockwise direction.

Below is the PCF8574 pin schematic diagram and the block pin diagram:

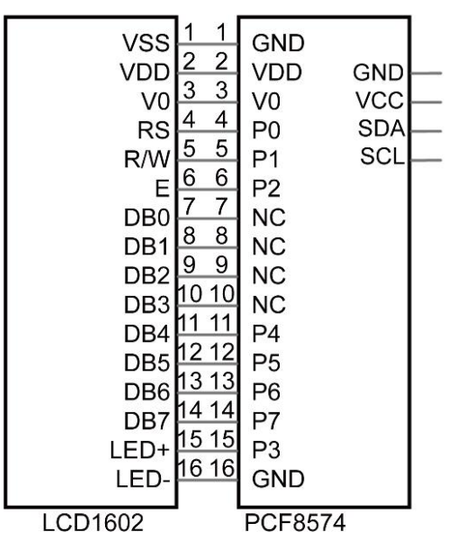
PCF8574 chip pin diagram:



PCF8574 module pin diagram



PCF8574 module pin and LCD1602 pin are corresponding to each other and connected with each other:

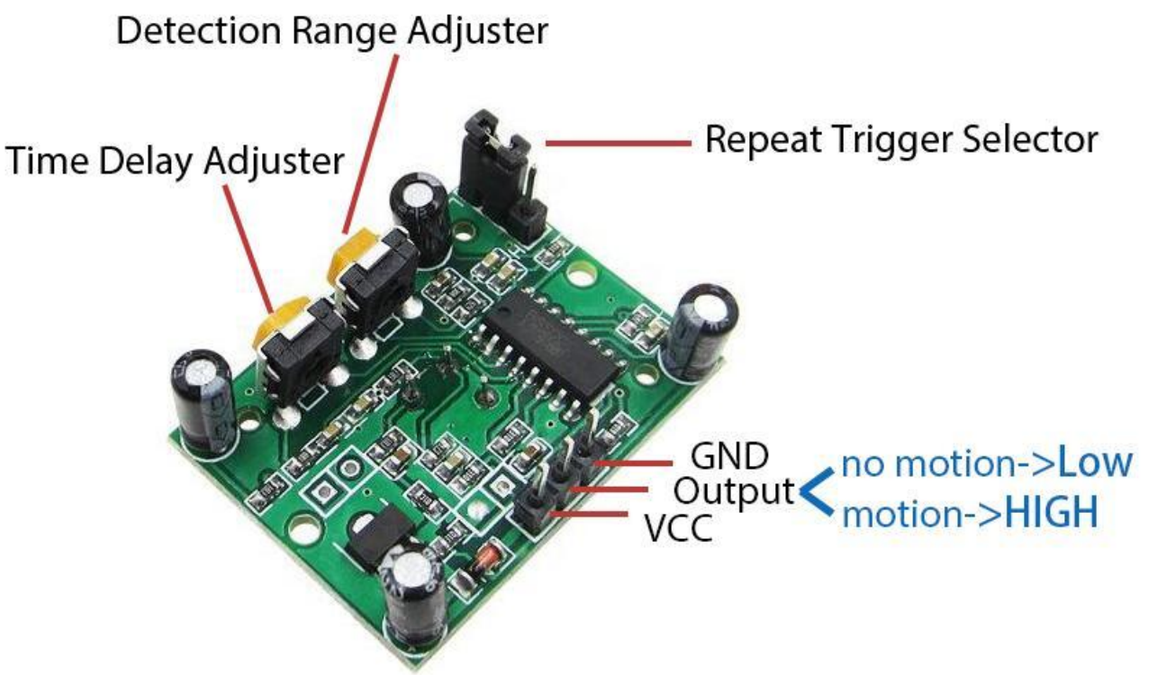


So we only need 4 pins to control the 16 pins of the LCD1602 Display Screen through the I2C interface.

we will use the I2C LCD1602 to display some static characters and dynamic variables.

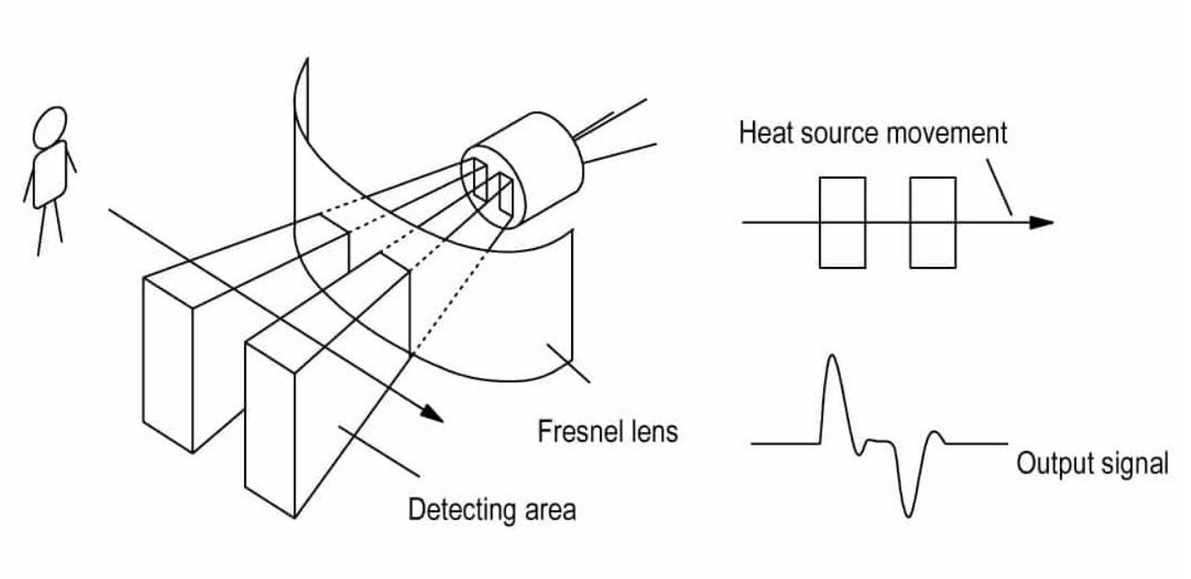
➢ PIR Motion Sensor

Passive infrared sensor (PIR sensor) is a common sensor that can measure infrared (IR) light emitted by objects in its field of view. Simply put, it will receive infrared radiation emitted from the body, thereby detecting the movement of people and other animals. More specifically, it tells the main control board that someone has entered your room.



The PIR sensor detects infrared heat radiation that can be used to detect the presence of organisms that emit infrared heat radiation.

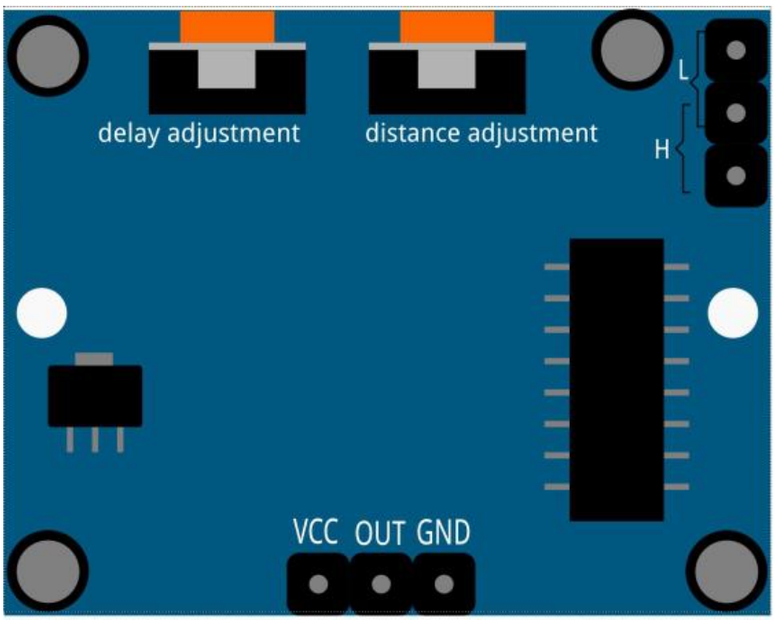
The PIR sensor is split into two slots that are connected to a differential amplifier. Whenever a stationary object is in front of the sensor, the two slots receive the same amount of radiation and the output is zero. Whenever a moving object is in front of the sensor, one of the slots receives more radiation than the other , which makes the output fluctuate high or low. This change in output voltage is a result of detection of motion.



Note:（Importance）

After the sensing module is wired, there is a one-minute initialization.After initialization,then the module will be in the standby mode.During the initialization,do not let any triggered infrared signal appear in the PIR monitoring range, including your hand. Otherwise in standby mode, it may cause false trigger detection.During the initialization, module will output for 0~3 times at intervals.This is not a real trigger result and you can ignore it until standby mode .

Please keep the interference of light source and other sources away from the surface of the module so as to avoid the misoperation caused by the interfering signal. Even you'd better use the module without too much wind, because the wind can also interfere with the sensor.



Distance Adjustment

Turning the knob of the distance adjustment potentiometer clockwise, the range of sensing distance increases, and the maximum sensing distance range is about 0-7 meters. If turn it anticlockwise, the range of sensing distance is reduced, and the minimum sensing distance range is about 0-3 meters.

Delay adjustment

Rotate the knob of the delay adjustment potentiometer clockwise, you can also see the sensing delay increasing. The maximum of the sensing delay can reach up to 300s. On the contrary, if rotate it anticlockwise, you can shorten the delay with a minimum of 1s.Usually you need to set the delay time to a minimum of 1s.

Two Trigger Modes

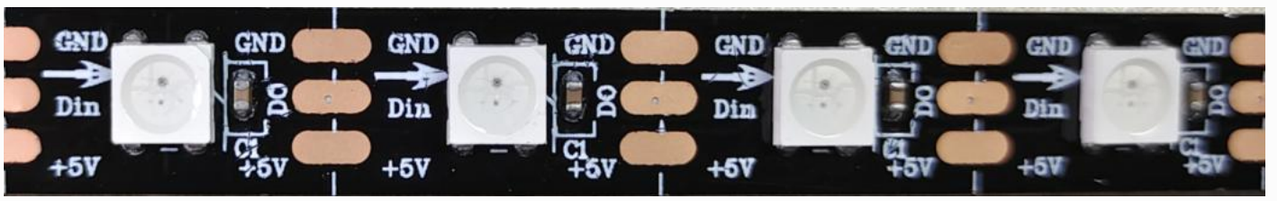
Choosing different modes by using the jumper cap.

H: Repeatable trigger mode, after sensing the human body, the module outputs high level.

During the subsequent delay period, if somebody enters the sensing range,the output will keep being the high level.

L：Non-repeatable trigger mode, outputs high level when it senses the human body. After the delay, the output will change from high level into low level automatically.

➢ WS2812 RGB 8 LEDs Strip



The WS2812 RGB 8 LEDs Strip is composed of 8 RGB LEDs. Only one pin is required to control all the LEDs. Each RGB LED has a WS2812 chip, which can be controlled independently. It can realize 256-level brightness display and complete true color display of 16,777,216 colors. At the same time, the pixel contains an intelligent digital interface data latch signal shaping amplifier drive circuit, and a signal shaping circuit is built in to effectively ensure the color height of the pixel point light Consistent.

It is flexible, can be docked, bent, and cut at will, and the back is equipped with adhesive tape, which can be fixed on the uneven surface at will, and can be installed in a narrow space.

Features

- Work Voltage: DC5V
- IC: One IC drives one RGB LED
- Consumption: 0.3w each LED
- Working Temperature: -15-50
- Color: Full color RGB
- RGB Type：5050RGB（Built-in IC WS2812B）
- Light Strip Thickness: 2mm
- Each LED can be controlled individually

WS2812B Introduction

- WS2812B Datasheet

WS2812B is a intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. It internal include intelligent digital port data latch and signal reshaping amplification drive circuit. Also include a precision internal oscillator and a 12V voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.

The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel，the signal to reduce 24bit. pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depend on the speed of signal transmission.

LED with low driving voltage, environmental protection and energy saving, high brightness, scattering angle is large, good consistency, low power, long life and other advantages. The control chip integrated in LED above becoming more simple circuit, small volume, convenient installation.

FAQ

Question:

Can Raspberry Pi Pico be used on Thonny, Arduino IDE and Piper Make at the same time?

Answer:

No, Raspberry Pi Pico only supports one kind of firmware installation at a time, you need to do some different operations according to the following situations.

①If you use it on Arduino IDE or Piper Make first, and now you want to use it on Thonny IDE, you need to **Burn Micropython Firmware** to your pico. Check out the "**Tutorial For MicroPython User**" folder.

②If you use it on Thonny or Piper Make first, and now you want to use it on Arduino IDE, you need to **Upload Arduino-compatible Firmware** for Pico. Check out the "**Tutorial For Arduino User**" folder.

③If you use it on Arduino IDE or Thonny first, and now you want to use it on Piper Make, you need to **Burn piper circuitpython Firmware** to your pico. Check out the "**Tutorial For Piper Make User**" folder.