Chapter 1 LCD1602

In this chapter, we will learn about the LCD1602 Display Screen.

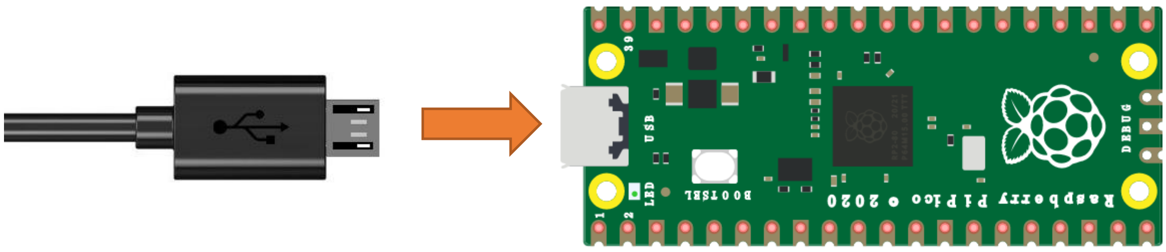
Project 1.1 LCD1602

In this section we learn how to use LCD1602 to display something.

Component knowledge

Power

In this tutorial, we connect Raspberry Pi Pico and compter with a USB cable.

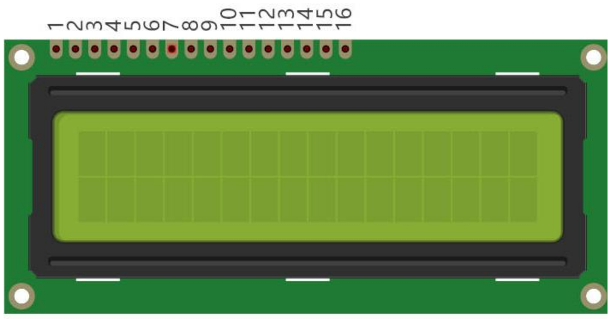


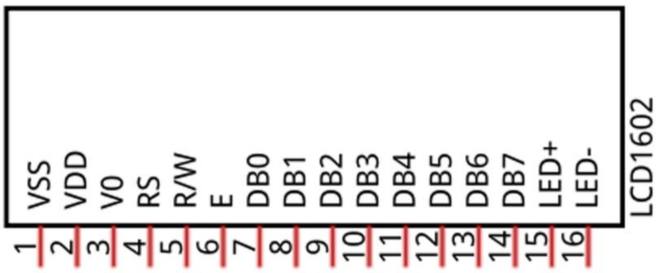
I2C communication

I2C (Inter-Integrated Circuit) is a two-wire serial communication mode, which can be used for the connection of micro controllers and their peripheral equipment. Devices using I2C communication must be connected to the serial data (SDA) line, and serial clock (SCL) line (called I2C bus). Each device has a unique address and can be used as a transmitter or receiver to communicate with devices connected to the bus.

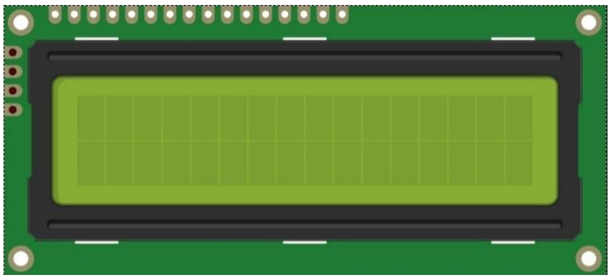
LCD1602 communication

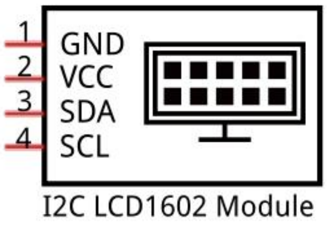
The LCD1602 Display Screen can display 2 lines of characters in 16 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD1602 Display Screen along with its circuit pin diagram.





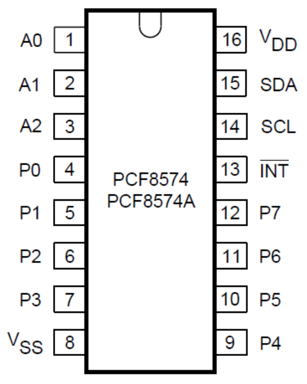
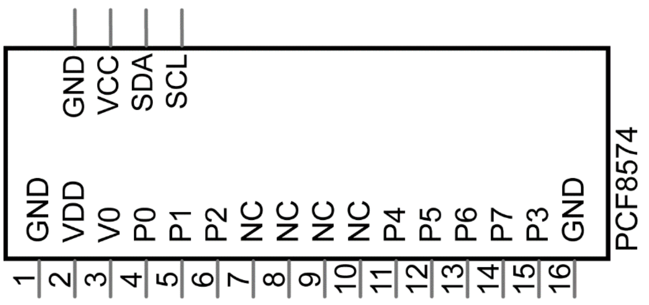
I2C LCD1602 Display Screen integrates an I2C interface, which connects the serial-input & parallel-output module to the LCD1602 Display Screen. This allows us to use only 4 lines to the operate the LCD1602.



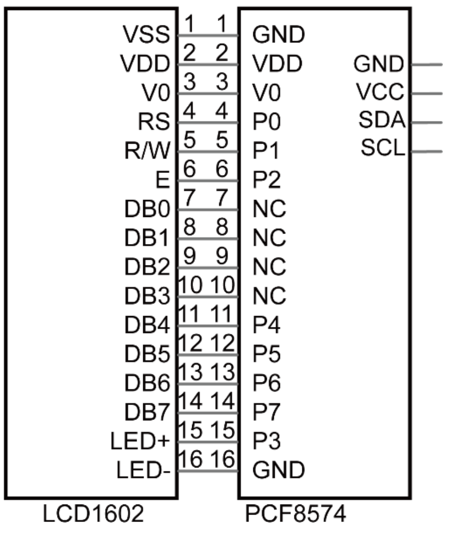


The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F).

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

PCF8574 chip pin diagram	PCF8574 module pin diagram
 A0 1 A1 2 A2 3 P0 4 P1 5 P2 6 P3 7 V_{SS} 8 PCF8574 PCF8574A 16 V_{DD} 15 SDA 14 SCL 13 $\overline{INT}$ 12 P7 11 P6 10 P5 9 P4	 GND VCC SDA SCL GND 1 VDD 2 V0 3 P0 4 P1 5 P2 6 NC 7 NC 8 NC 9 NC 10 P4 11 P5 12 P6 13 P7 14 P3 15 GND 16 PCF8574

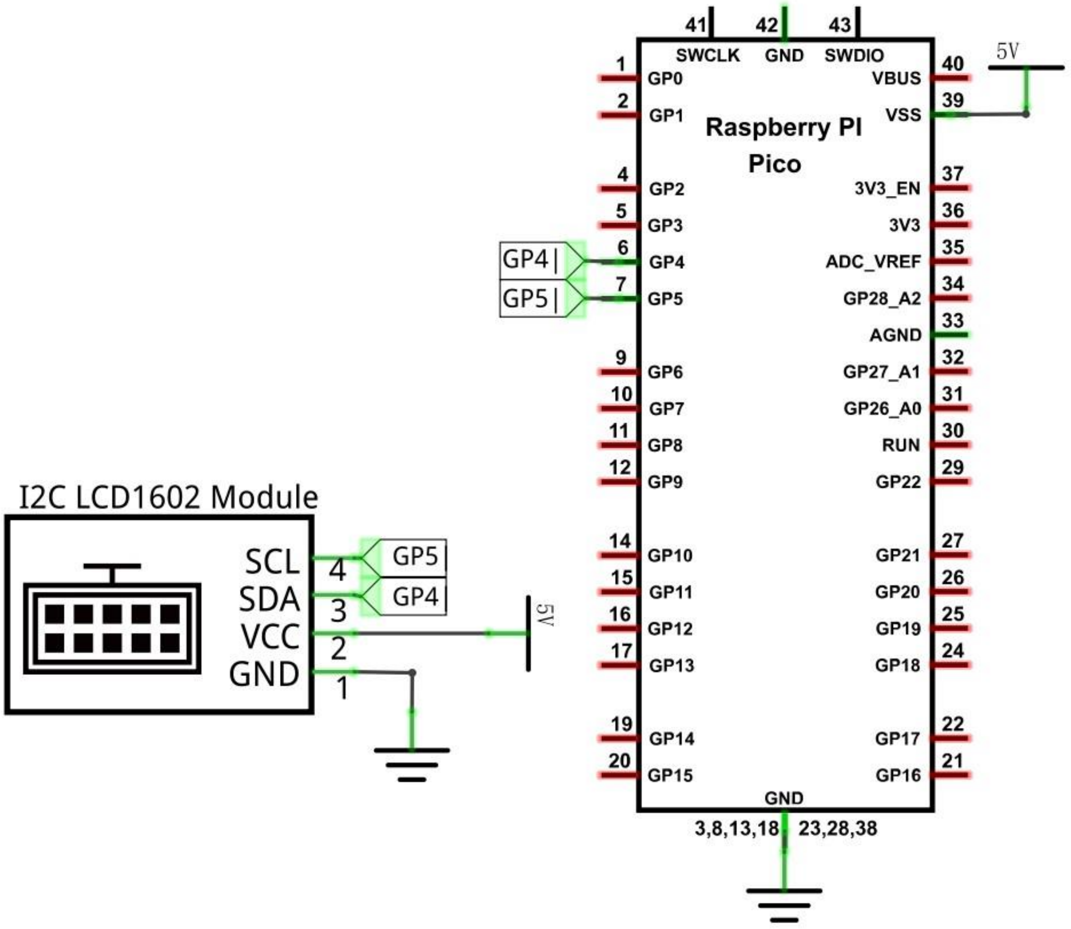
PCF8574 module pin and LCD1602 pin are corresponding to each other and connected with each other:



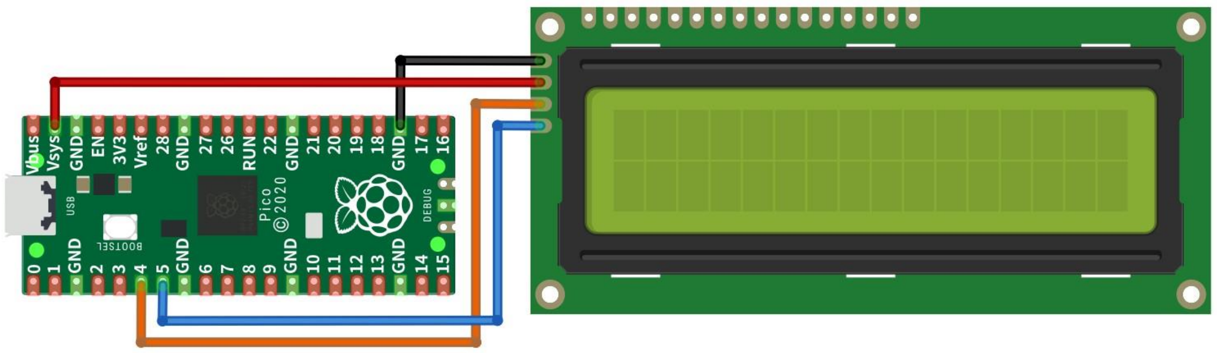
So we only need 4 pins to control the 16 pins of the LCD1602 Display Screen through the I2C interface.
In this project, we will use the I2C LCD1602 to display some static characters and dynamic variables.

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

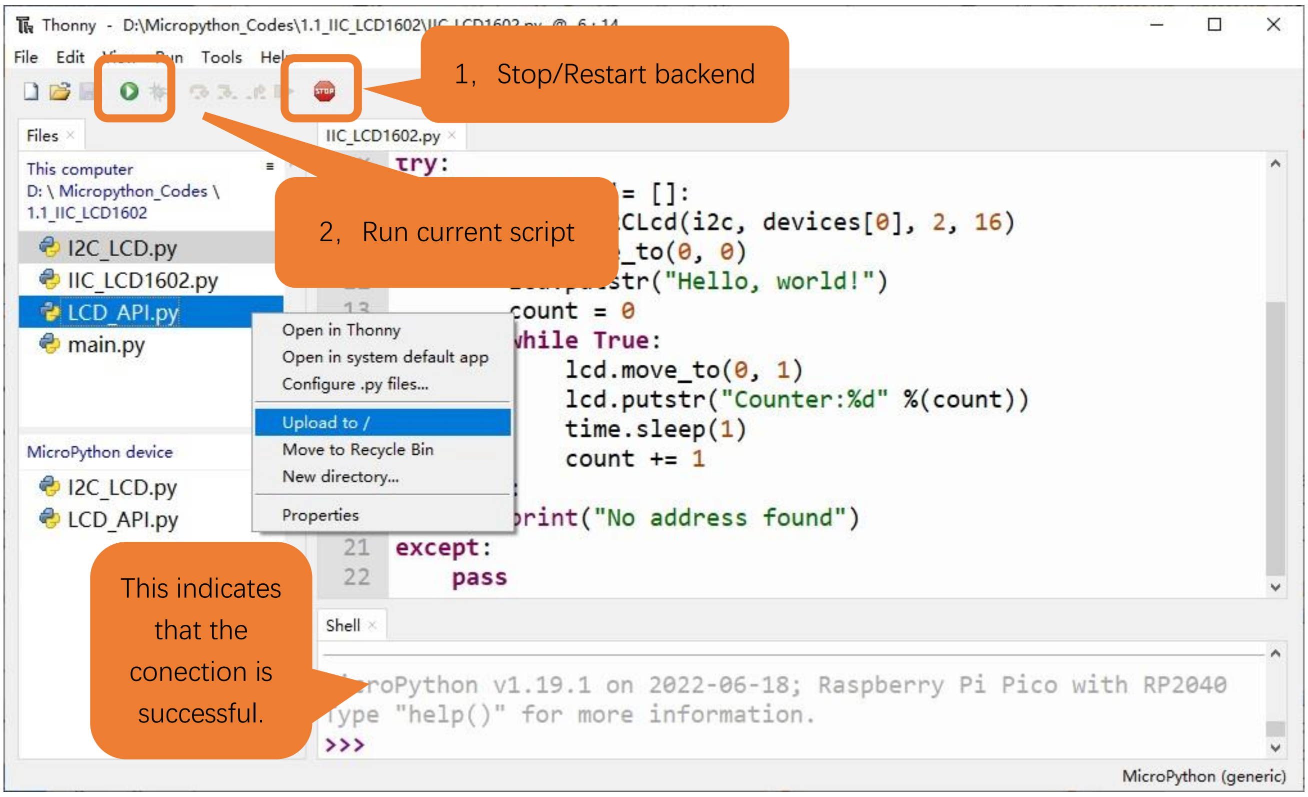


Code

Codes used in this tutorial are saved in "**Freenove_LCD_Module/Freenove_LCD_Module_for_Raspberry_Pi_Pico/Python_Codes**".
You can move the codes to any location. For example, we save the codes in Disk(D) with the path of "**D:/Micropython_Codes**".

Open "Thonny", click "This computer" → "D:" → "Micropython_Codes" → "1.1_I2C_LCD1602". Select "I2C_LCD.py"and "LCD_API.py", right click your mouse to select "Upload to /", wait for "I2C_LCD.py" and "LCD_API.py" to be uploaded to Raspberry Pi Pico and then double click "I2C_LCD1602.py".

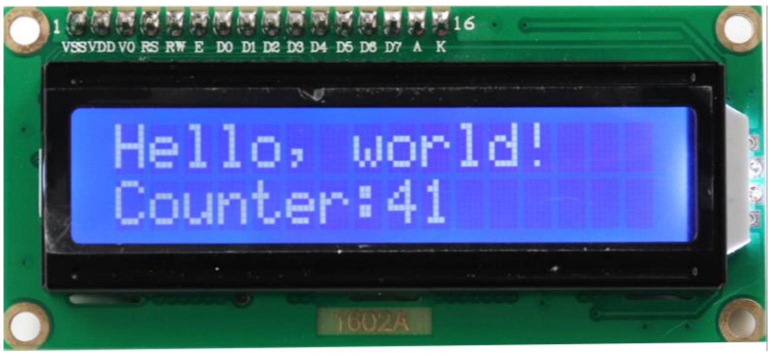
1.1_I2C_LCD1602



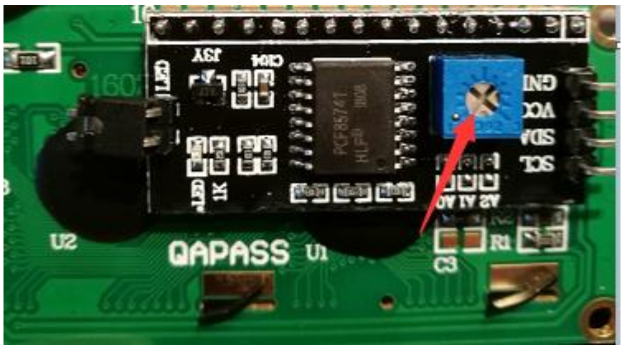
```
try:
    = []:
    CLcd(i2c, devices[0], 2, 16)
    _to(0, 0)
    str("Hello, world!")
    count = 0
    while True:
        lcd.move_to(0, 1)
        lcd.putstr("Counter:%d" %(count))
        time.sleep(1)
        count += 1
    :
    print("No address found")
except:
    pass
```

```
roPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

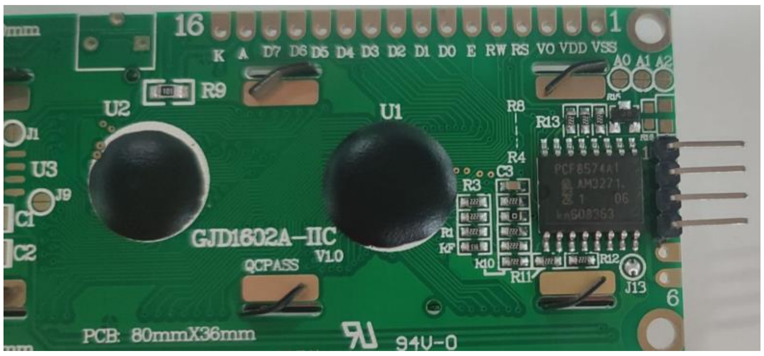
Click "Run current script" and LCD1602 displays some characters.Press Ctrl+C or click "Stop/Restart backend" to exit the program.



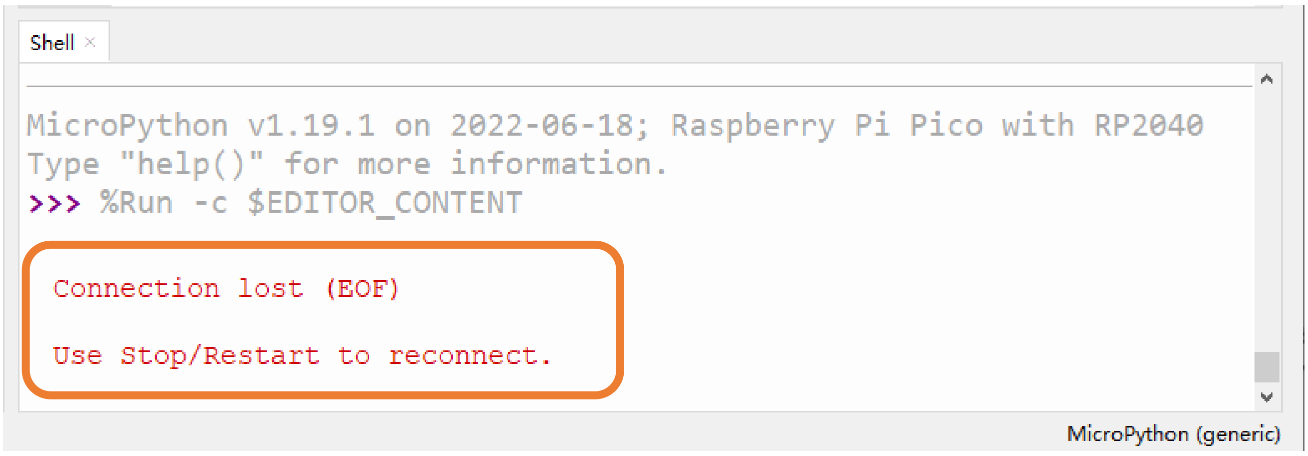
Note: If you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD1602 slowly, which adjusts the contrast, until the screen can display clearly.



In addition, if the back of your LCD module looks like the picture below, there is no need to adjust the contrast, this circuit has adjusted the contrast to a suitable value.



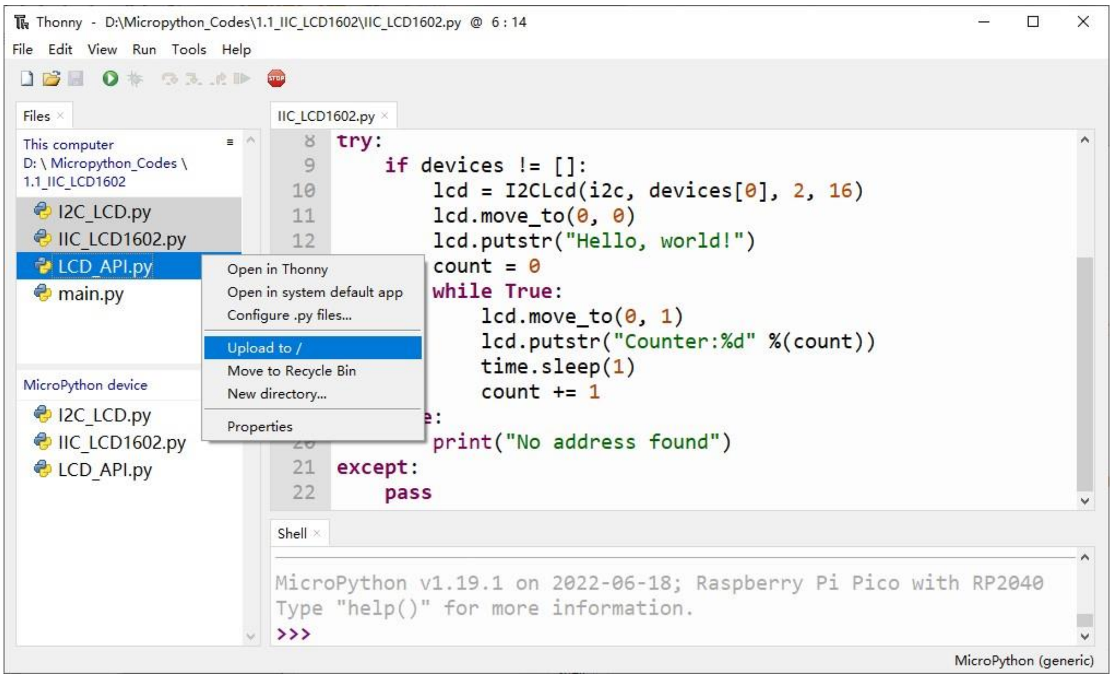
Note:
This is the code running online. If you disconnect USB cable and repower Raspberry Pi Pico, LED stops blinking and the following messages will display in Thonny.



```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Connection lost (EOF)

Use Stop/Restart to reconnect.
```

Uploading code to Raspberry Pi Pico

As shown in the following illustration, Select “I2C_LCD.py”and“LCD_API.py”and“I2C_LCD1602.py”, right click your mouse to select “Upload to /” to upload code to Raspberry Pi Pico.

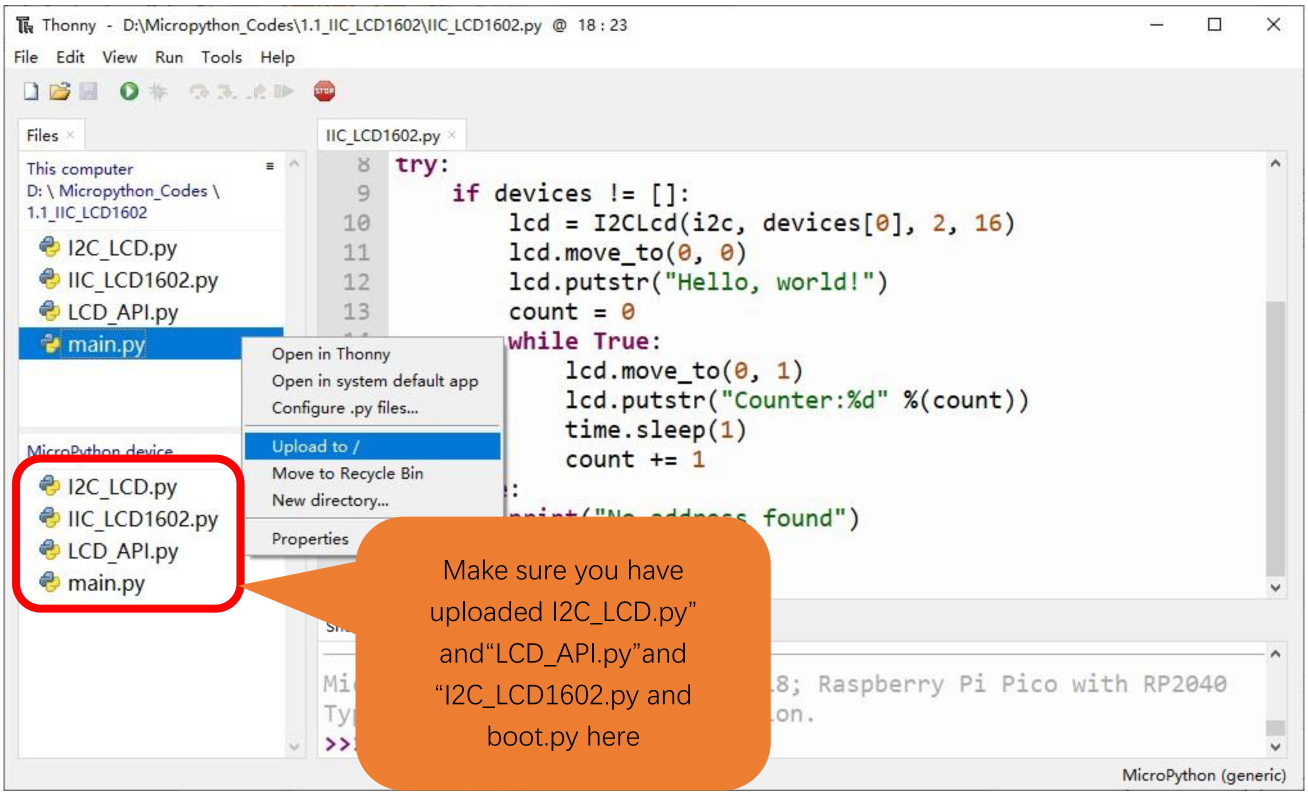


Thonny - D:\Micropython_Codes\1.1_IIC_LCD1602\IIC_LCD1602.py @ 6 : 14

```
try:
    if devices != []:
        lcd = I2CLcd(i2c, devices[0], 2, 16)
        lcd.move_to(0, 0)
        lcd.putstr("Hello, world!")
        count = 0
        while True:
            lcd.move_to(0, 1)
            lcd.putstr("Counter:%d" %(count))
            time.sleep(1)
            count += 1
    e:
        print("No address found")
except:
    pass
```

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

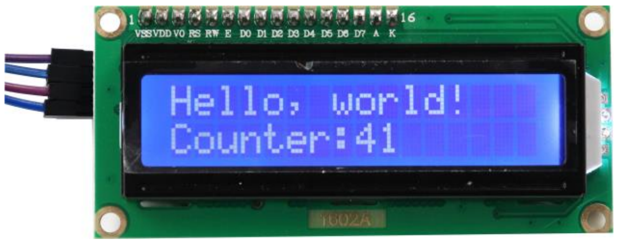
Upload main.py in the same way.



Thonny - D:\Micropython_Codes\1.1_IIC_LCD1602\IIC_LCD1602.py @ 18 : 23

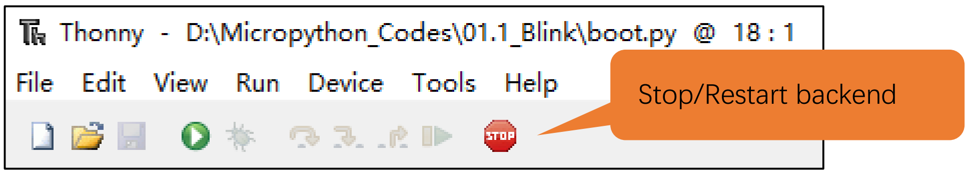
```
try:
    if devices != []:
        lcd = I2CLcd(i2c, devices[0], 2, 16)
        lcd.move_to(0, 0)
        lcd.putstr("Hello, world!")
        count = 0
        while True:
            lcd.move_to(0, 1)
            lcd.putstr("Counter:%d" %(count))
            time.sleep(1)
            count += 1
```

Disconnect Raspberry Pi Pico USB cable and reconnect it, you can see LCD1602 displays some characters.



Note:

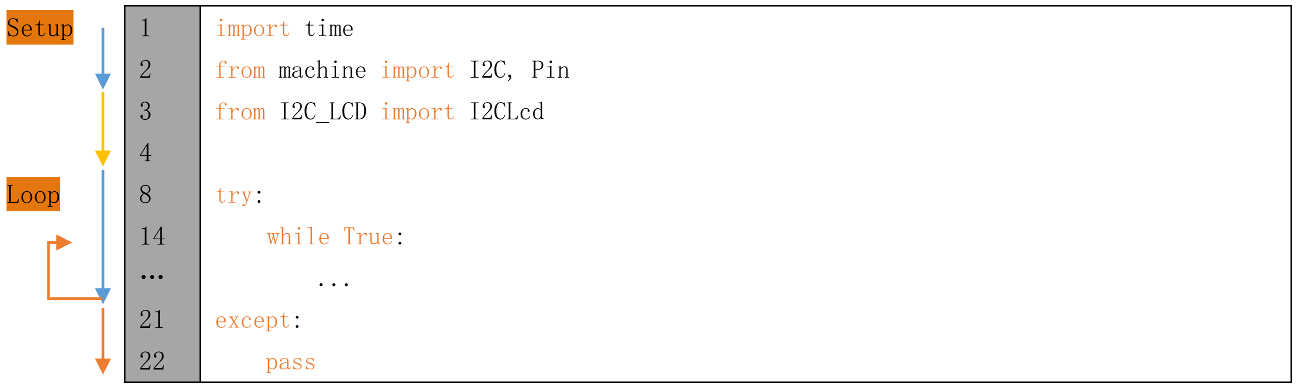
Codes here are run offline. If you want to stop running offline and enter Shell, just click “Stop” in Thonny.


Thonny - D:\Micropython_Codes\01.1_Blink\boot.py @ 18 : 1
File Edit View Run Device Tools Help
Stop/Restart backend

The following is the program code:

```python
import time
from machine import I2C, Pin
from I2C_LCD import I2CLcd

i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=400000)
devices = i2c.scan()

try:
    if devices != []:
        lcd = I2CLcd(i2c, devices[0], 2, 16)
        lcd.move_to(0, 0)
        lcd.putstr("Hello, world!")
        count = 0
        while True:
            lcd.move_to(0, 1)
            lcd.putstr("Counter:%d" %(count))
            time.sleep(1)
            count += 1
    else:
        print("No address found")
except:
    pass
```

Each time a new file is opened, the program will be executed from top to bottom. When encountering a loop construction, it will execute the loop statement according to the loop condition.



```
1    import time
2    from machine import I2C, Pin
3    from I2C_LCD import I2CLcd
4
8    try:
14       while True:
…            ...
21   except:
22       pass
```

Import time, I2C and I2C_LCD modules.

```python
import time
```

```
from machine import I2C, Pin
from I2C_LCD import I2CLcd
```

Create an I2C object, initialize the I2C parameter configuration, and associate it with the LCD1602 pin. Call the scan() function to query the LCD1602 device address.

```
i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=400000)
devices = i2c.scan()
```

Use the if statement to determine whether the queried I2C device address is empty. If it is not empty, create an I2CLcd object and set the created I2C object, I2C device address, and the number of rows and columns of LCD1602; if it is empty, print out "No address" found" and the program exits.

```
    if devices != []:
        lcd = I2CLcd(i2c, devices[0], 2, 16)
        …
    else:
        print("No address found")
```

Move the cursor of LCD1602 to the first row, first column, and print out "Hello, world!"

```
    lcd.move_to(0, 0)
    lcd.putstr("Hello, world!")
```

The second line of LCD1602 continuously prints the number of seconds after the Raspberry Pi Pico program runs.

```
    while True:
        lcd.move_to(0, 1)
        lcd.putstr("Counter:%d" %(count))
        time.sleep(1)
        count += 1
```

Put statements that may cause an error in “try” block and the executing statements when an error occurs in “except” block. In general, when the program executes statements, it will execute those in “try” block. However, when an error occurs to Raspberry Pi Pico due to some interference or other reasons, it will execute statements in “except” block.
“Pass” is an empty statement. When it is executed, nothing happens. It is useful as a placeholder to make the structure of a program look better.

```
try:
    ...
except:
    pass
```

MicroPython uses indentations to distinguish different blocks of code instead of braces. The number of indentations is changeable, but it must be consistent throughout one block. If the indentation of the same code block is inconsistent, it will cause errors when the program runs.

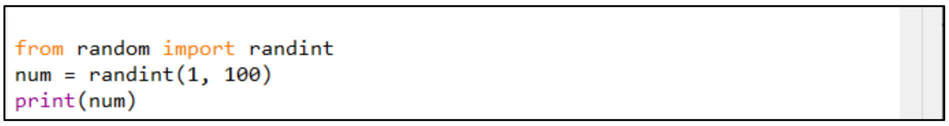
```
        while True:
            lcd.move_to(0, 1)
            lcd.putstr("Counter:%d" %(count))
            time.sleep(1)
            count += 1
```

How to import python files

Whether to import the built-in python module or to import that written by users, the command “import” is needed.
If you import the module directly you should indicate the module to which the function or attribute belongs when using the function or attribute (constant, variable) in the module. The format should be: <module name>.<function or attribute>, otherwise an error will occur.

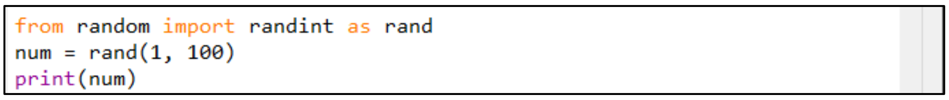
```
import random

num = random.randint(1, 100)
print(num)
```

If you only want to import a certain function or attribute in the module, use the from...import statement. The format is as follows



```
from random import randint
num = randint(1, 100)
print(num)
```

When using “from...import” statement to import function, to avoid conflicts and for easy understanding, you can use “as” statement to rename the imported function, as follows



```
from random import randint as rand
num = rand(1, 100)
print(num)
```

Reference

Class machine

Before each use of the **machine** module, please add the statement "**import machine**" to the top of python file.

machine.freq(freq_val): When "freq_val" is not specified, it is to return to the current CPU frequency; Otherwise, it is to set the current CPU frequency.

freq_val: 125000000Hz(125MHz).

machine.reset(): A reset function. When it is called, the program will be reset.

machine.unique_id(): Obtains MAC address of the device.

machine.idle(): Turns off any temporarily unused functions on the chip and its clock, which is useful to reduce power consumption at any time during short or long periods.

machine.disable_irq(): Disables interrupt requests and return the previous IRQ state. The disable_irq () function and enable_irq () function need to be used together; Otherwise the machine will crash and restart.

machine.enable_irq(state): To re-enable interrupt requests. The parameter **state** should be the value that was returned from the most recent call to the disable_irq() function.

machine.time_pulse_us(pin, pulse_level, timeout_us=1000000):

Tests the duration of the external pulse level on the given pin and returns the duration of the external pulse level in microseconds. When pulse level = 1, it tests the high level duration; When pulse level = 0, it tests the low level duration.

If the setting level is not consistent with the current pulse level, it will wait until they are consistent, and then start timing. If the set level is consistent with the current pulse level, it will start timing immediately.

When the pin level is opposite to the set level, it will wait for timeout and return "-2". When the pin level and the set level is the same, it will also wait timeout but return "-1". **timeout_us** is the duration of timeout.

For more information about class and function, please refer to:
https://docs.micropython.org/en/latest/rp2/quickref.html

Class time

Before each use of the **time** module, please add the statement "**import time**" to the top of python file

time.sleep(sec): Sleeps for the given number of seconds.

sec: This argument should be either an int or a float.

time.sleep_ms(ms): Sleeps for the given number of milliseconds, ms should be an int.

time.sleep_us(us): Sleeps for the given number of microseconds, us should be an int.

time.time(): Obtains the timestamp of CPU, with second as its unit.

time.ticks_ms(): Returns the incrementing millisecond counter value, which recounts after some values.

time.ticks_us(): Returns microsecond.

time.ticks_cpu(): Similar to ticks_ms() and ticks_us(), but it is more accurate(return clock of CPU).

time.ticks_add(ticks, delta): Gets the timestamp after the offset.

ticks: ticks_ms()、ticks_us()、ticks_cpu().

delta: Delta can be an arbitrary integer number or numeric expression.

time.ticks_diff(old_t, new_t): Calculates the interval between two timestamps, such as ticks_ms(), ticks_us() or ticks_cpu().

old_t: Starting time.

new_t: Ending time.

Class Pin(id, mode, pull, value)

Before each use of the **Pin** module, please add the statement "**from machine import Pin**" to the top of python file.

id: Arbitrary pin number.

mode: Mode of pins.

Pin.IN: Input Mode.

Pin.OUT: Output Mode.

Pin.OPEN_DRAIN: Open-drain Mode.

Pull: Whether to enable the internal pull up and down mode.

None: No pull up or pull down resistors.

Pin.PULL_UP: Pull-up Mode, outputting high level by default.

Pin.PULL_DOWN: Pull-down Mode, outputting low level by default.

Value: State of the pin level, 0/1.

Pin.init(mode, pull): Initialize pins.

Pin.value([value]): Obtain or set state of the pin level, return 0 or 1 according to the logic level of pins. Without parameter, it reads input level. With parameter given, it is to set output level.

value: It can be either True/False or 1/0.

Pin.irq(trigger, handler): Configures an interrupt handler to be called when the pin level meets a condition.

trigger:

Pin.IRQ_FALLING: interrupt on falling edge.

Pin.IRQ_RISING: interrupt on rising edge.

Handler: callback function.

Class I2CLcd

Before each use of the object **I2CLcd**, please make sure that **I2C_LCD.py** and **LCD_API.py** have been uploaded to "/" of Raspberry Pi Pico, and then add the statement "**from I2C_LCD import I2CLcd**" to the top of the python file.

clear(): Clear the LCD1602 screen display.

show_cursor(): Show the cursor of LCD1602.

hide_cursor(): Hide the cursor of LCD1602.

blink_cursor_on(): Turn on cursor blinking.

blink_cursor_off(): Turn off cursor blinking.

display_on(): Turn on the display function of LCD1602.

display_off(): Turn on the display function of LCD1602.

backlight_on(): Turn on the backlight of LCD1602.

backlight_off(): Turn on the backlight of LCD1602.

move_to(cursor_x, cursor_y): Move the cursor to a specified position.

cursor_x: Column cursor_x.

cursor_y: Row cursor_y.

putchar(char): Print the character in the bracket on LCD1602.

putstr(string): Print the string in the bracket on LCD1602.

Chapter 2 LCD2004

In the previous chapter, we studied the LCD1602 display. In order to display more content,In this chapter, we will learn about the LCD2004 Display Screen.

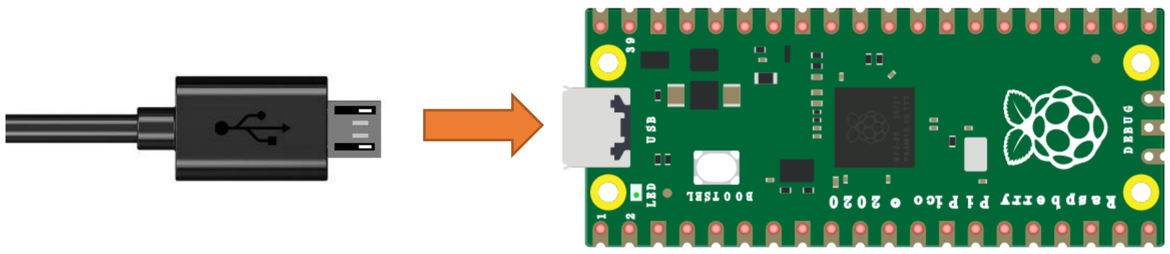
Project 1.1 LCD2004

In this section we learn how to use lcd2004 to display something.

Component knowledge

Power

In this tutorial, we connect Raspberry Pi Pico and compter with a USB cable.

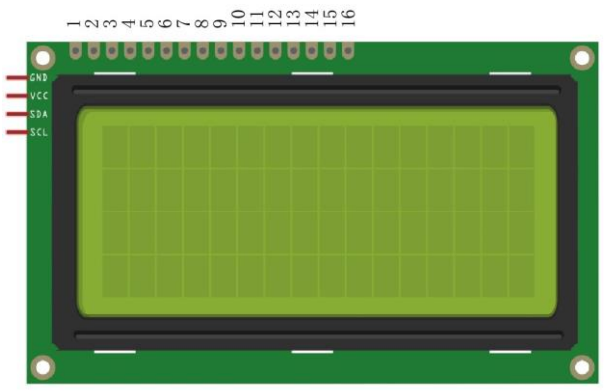


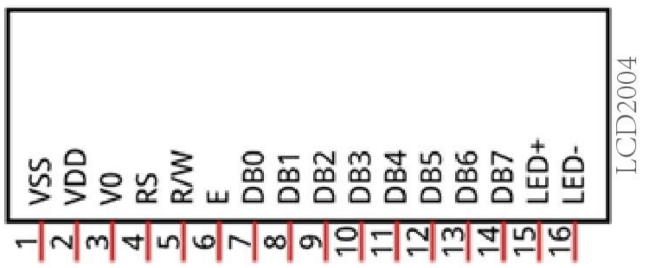
I2C communication

I2C (Inter-Integrated Circuit) is a two-wire serial communication mode, which can be used for the connection of micro controllers and their peripheral equipment. Devices using I2C communication must be connected to the serial data (SDA) line, and serial clock (SCL) line (called I2C bus). Each device has a unique address and can be used as a transmitter or receiver to communicate with devices connected to the bus.

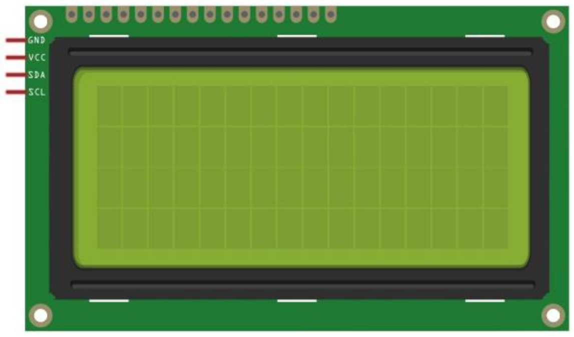
LCD2004 communication

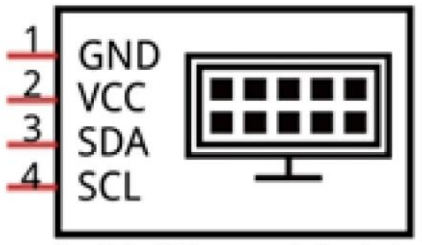
The LCD2004 Display Screen can display 4 lines of characters in 20 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD2004 Display Screen along with its circuit pin diagram





I2C LCD2004 Display Screen integrates an I2C interface, which connects the serial-input & parallel-output module to the LCD2004 Display Screen. This allows us to use only 4 lines to the operate the LCD2004.

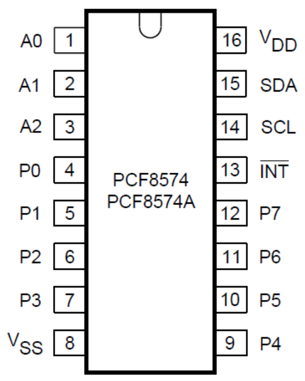
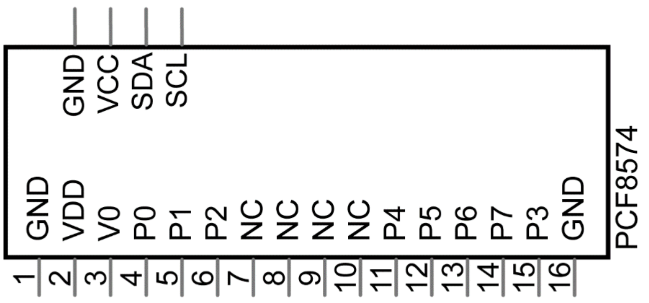




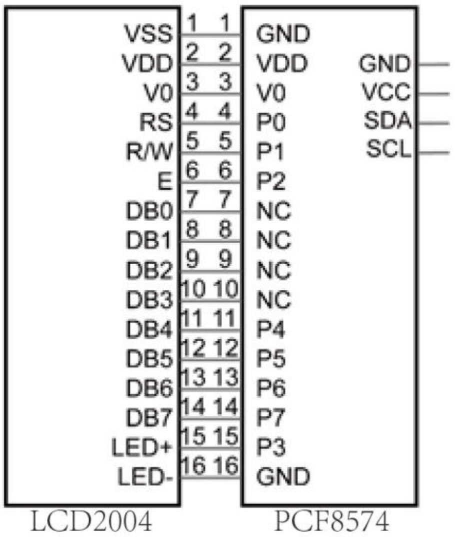
I2C LCD2004 Module

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

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

PCF8574 chip pin diagram	PCF8574 module pin diagram
	

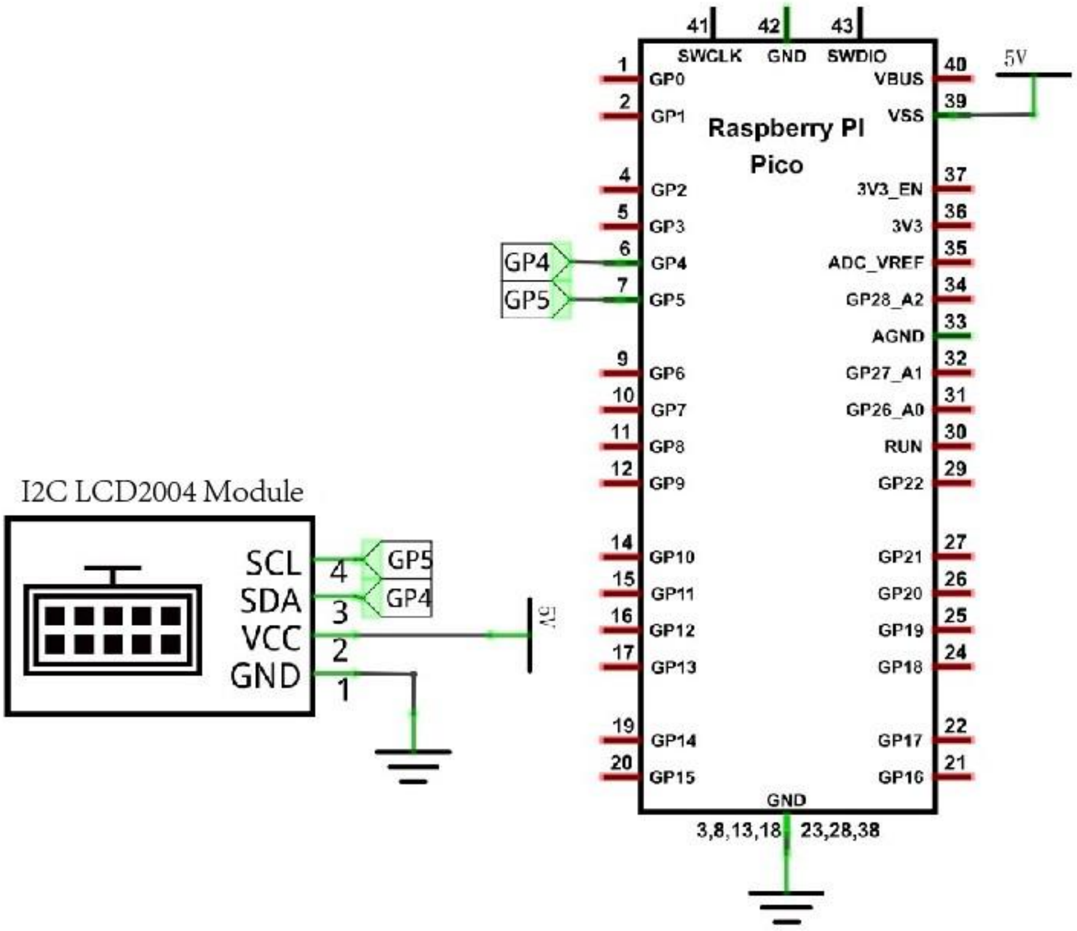
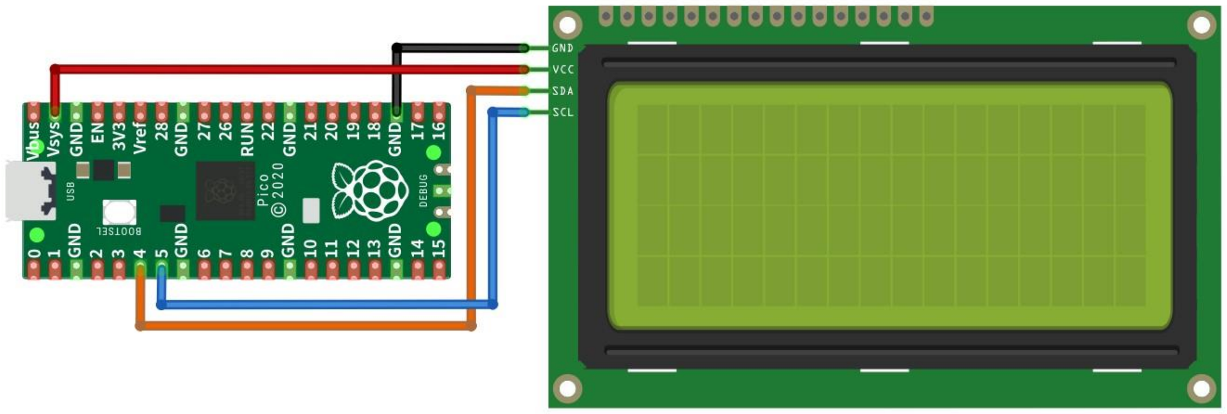
PCF8574 module pin and LCD2004 pin are corresponding to each other and connected with each other:



LCD2004 PCF8574

So we only need 4 pins to control the 16 pins of the LCD2004 display screen through the I2C interface.
In this project, we will use the I2C LCD2004 to display some static characters and dynamic variables.

Circuit

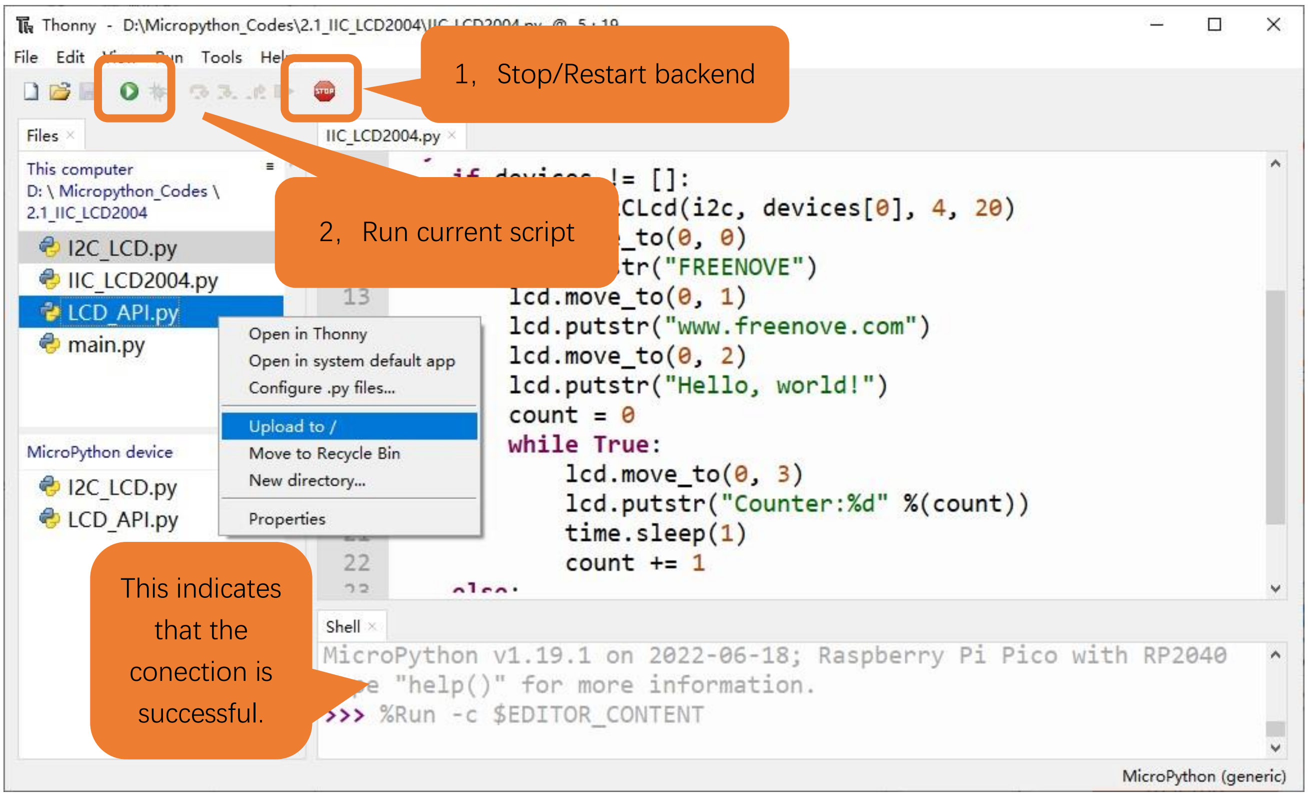
Schematic diagram

Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com


Code

Codes used in this tutorial are saved in
"**Freenove_LCD_Module/Freenove_LCD_Module_for_Raspberry_Pi_Pico/Python_Codes**".
You can move the codes to any location. For example, we save the codes in Disk(D) with the path of "**D:/Micropython_Codes**".

Open "Thonny", click "This computer" → "D:" → "Micropython_Codes" → "2.1_I2C_LCD2004". Select "I2C_LCD.py"and "LCD_API.py", right click your mouse to select "Upload to /", wait for "I2C_LCD.py" and "LCD_API.py" to be uploaded to Raspberry Pi Pico and then double click "I2C_LCD2004.py".

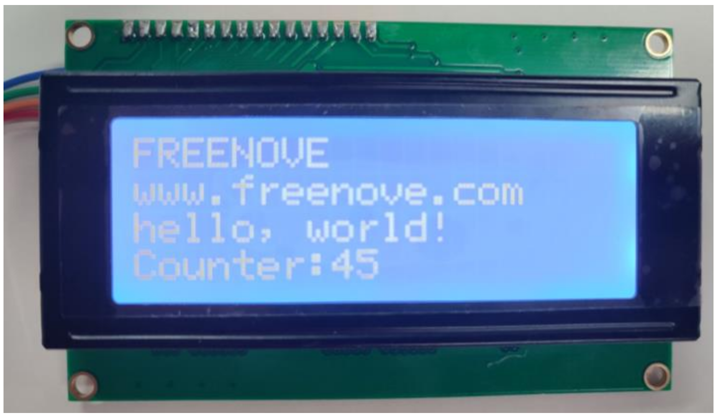
2.1_I2C_LCD2004



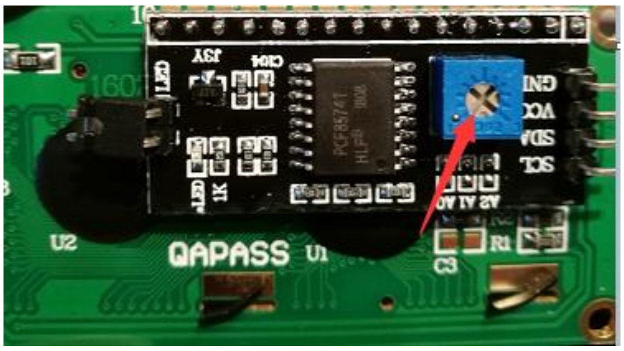
```
if devices != []:
    CLcd(i2c, devices[0], 4, 20)
    _to(0, 0)
    tr("FREENOVE")
    lcd.move_to(0, 1)
    lcd.putstr("www.freenove.com")
    lcd.move_to(0, 2)
    lcd.putstr("Hello, world!")
    count = 0
    while True:
        lcd.move_to(0, 3)
        lcd.putstr("Counter:%d" %(count))
        time.sleep(1)
        count += 1
else:
```

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
e "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
```

Click "Run current script" and LCD2004 displays some characters.Press Ctrl+C or click "Stop/Restart backend" to exit the program.



Note: If you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD2004 slowly, which adjusts the contrast, until the screen can display clearly.



The following is the program code:

```python
import time
from machine import I2C, Pin
from I2C_LCD import I2CLcd

i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=400000)
devices = i2c.scan()

try:
    if devices != []:
        lcd = I2CLcd(i2c, devices[0], 4, 20)
        lcd.move_to(0, 0)
        lcd.putstr("FREENOVE")
        lcd.move_to(0, 1)
        lcd.putstr("www.freenove.com")
        lcd.move_to(0, 2)
        lcd.putstr("Hello, world!")
        count = 0
        while True:
            lcd.move_to(0, 3)
            lcd.putstr("Counter:%d" %(count))
            time.sleep(1)
            count += 1
    else:
        print("No address found")
except:
    pass
```

Import time, I2C and I2C_LCD modules.

```python
import time
from machine import I2C, Pin
from I2C_LCD import I2CLcd
```

Create an I2C object, initialize the I2C parameter configuration, and associate it with the LCD2004 pin. Call the scan() function to query the LCD2004 device address.

```
i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=400000)
devices = i2c.scan()
```

Use the if statement to determine whether the queried I2C device address is empty. If it is not empty, create an I2CLcd object and set the created I2C object, I2C device address, and the number of rows and columns of LCD2004; if it is empty, print out "No address" found" and the program exits.

```
    if devices != []:
        lcd = I2CLcd(i2c, devices[0], 4, 20)
        …
    else:
        print("No address found")
```

Move the cursor of LCD2004 to the third row, first column, and print out "Hello, world!"

```
    lcd.move_to(0, 2)
    lcd.putstr("Hello, world!")
```

The fourth line of LCD2004 continuously prints the number of seconds after the Raspberry Pi Pico program runs.

```
    while True:
        lcd.move_to(0, 3)
        lcd.putstr("Counter:%d" %(count))
        time.sleep(1)
        count += 1
```

Reference

Class I2CLcd

Before each use of the object **I2CLcd**, please make sure that **I2C_LCD.py** and **LCD_API.py** have been uploaded to "/" of Raspberry Pi Pico, and then add the statement "**from I2C_LCD import I2CLcd**" to the top of the python file.

clear(): Clear the LCD2004 screen display.
show_cursor(): Show the cursor of LCD2004.
hide_cursor(): Hide the cursor of LCD2004.
blink_cursor_on(): Turn on cursor blinking.
blink_cursor_off(): Turn off cursor blinking.
display_on(): Turn on the display function of LCD2004.
display_off(): Turn on the display function of LCD2004.
backlight_on(): Turn on the backlight of LCD2004.
backlight_off(): Turn on the backlight of LCD2004.
move_to(cursor_x, cursor_y): Move the cursor to a specified position.
    cursor_x: Column cursor_x.
    cursor_y: Row cursor_y.
putchar(char): Print the character in the bracket on LCD2004.
putstr(string): Print the string in the bracket on LCD2004.