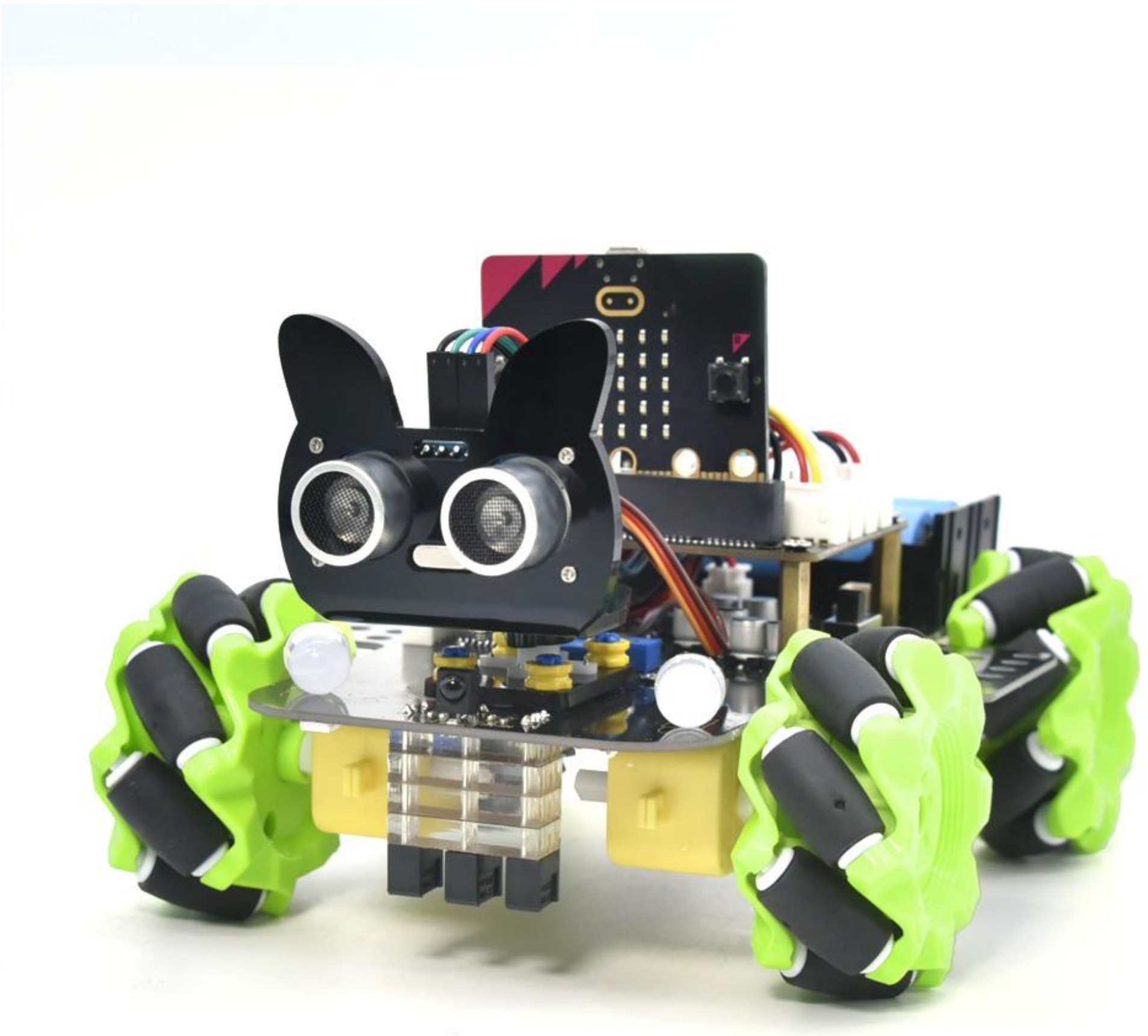
keyestudio 4WD Mecanum Robot Car V2.0



Contents

1. Introduction

Have you wondered to learn programming or have your own programming robot? Nowadays, programming has developed to a lower age group, and it will be a trend for everyone thanks to the spread of simple graphical programming platforms, from micro:bit to Arduino and Raspberry Pi. Maybe you haven't heard of them before. However, with the help of this product and tutorial, you can easily install a multi-functional programming car and experience the fun of being a maker.

Micro:bit is a highly integrated microcontroller with powerful functions and small size. It is very suitable to be applied in STEAM education for its functions to make robots, wearable devices and electronic interactive games via the combination of code programming and graphical programming.

This Keyestudio 4WD Mecanum Robot Car is a smart DIY car dedicated to micro:bit. The smart car kit consists of a car body with extended functions, a PCB base plate with integrated motor drive sensors, 4 decelerating DC motors, Mecanum wheels, various modules and sensors as well as acrylic boards. Therefore, you can easily assemble a cool Mecanum wheel 4WD smart car by yourself.

The version of Python running on the BBC Micro: bit is called MicroPython, which will guide you to use software Mu to write MicroPython language for Micro:bit main board to control the Mecanum car. In this process, not only can you enhance your ability to make stuffs but also learn the skills of programming.

Python is one of the most popular programming languages especially in machine learning for its availability and accessibility have brought huge convenience. However, MicroPython is committed to reviving the Python programming language in microcontrollers and embedded systems.

This is a Python tutorial for 4WD Mecanum Robot Car. If you haven't learned the basic tutorial (Makecode tutorial.pdf), please learn it first. The basic one is programmed using graphical blocks, which is accessible for you to understand and start.

For your convenience, code in Python has been provided in every project, as well as code programming steps and

code explanation in details. Hope you can better understand them.

2. Description

This product is a smart car based on Micro:bit. It integrates a host of functions such as servo, ultrasonic following, line tracking, infrared control as well as Bluetooth control. There is a passive buzzer to play music, 4 WS2812RGB LEDs to display different colors, 2 seven-color lights to make direction lights for the car. This product uses two 18650 lithium batteries for power supply.

When installing and disassembling the battery, please pay attention to the positive and negative poles of the battery, and be sure not to reverse them. By the way, the motor speed of this product is adjustable.
In order to provide you with better experience, corresponding documents about installation and test code are also provided.

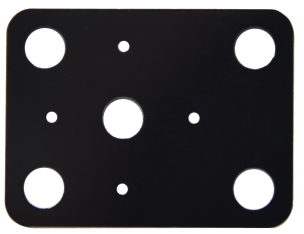
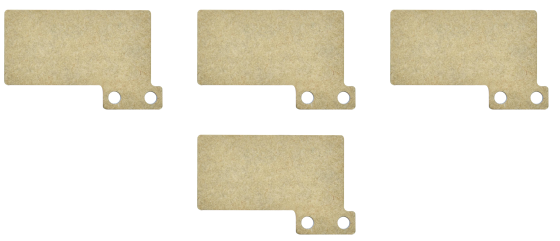
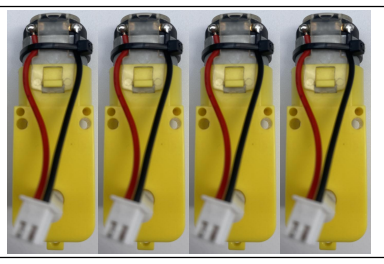
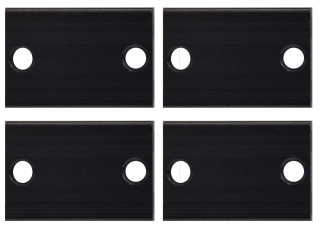
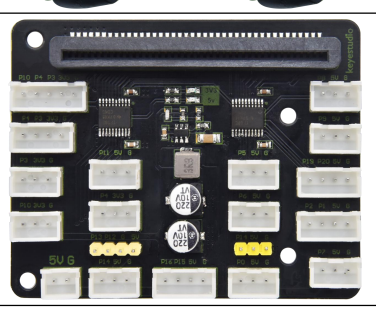
3.Parameters

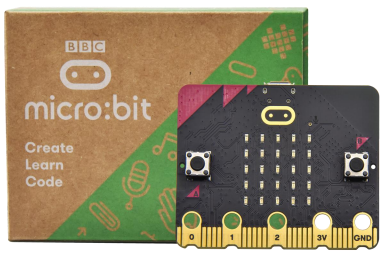
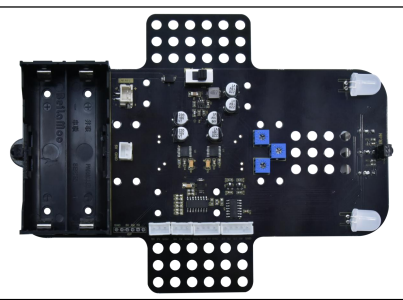
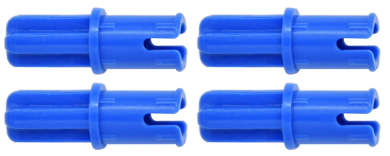
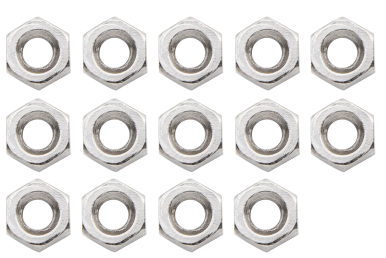
- Connector port input: DC 6V---9V
- Operating voltage of driver board system: 5V
- Standard operating power consumption: about 3W
- Maximum power: 14W
- Motor speed: 200RPM
- Working temperature range: 0-50℃
- Size: 134*181*75mm
- Environmental protection attributes: ROHS

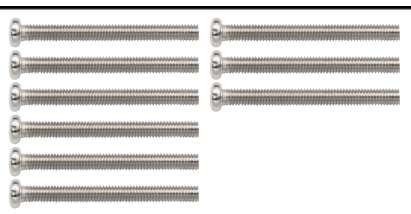
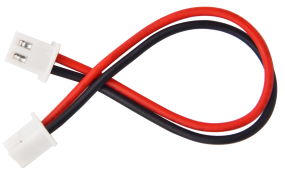
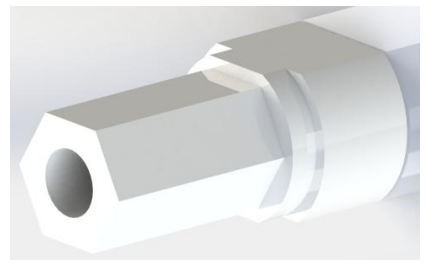
Note: The working voltage of micro:bit is 3.3V, and the driver shield integrates a 3.3V/5V communication conversion circuit.

4.Kit

#	Picture	Name	QTY

1		Acrylic Board T=3mm	1
2		Acrylic Board with Lego Holes T=3mm	1
3		Motor Plate	4
4		Motor	4
5		23*15*5MM Fixing Board	4
6		Servo	1
7		Mecanum Wheels (A direction)	2
8		Mecanum Wheels (B direction)	2
9		keyestudio Micro：bit Expansion Board	1

10		micro:bit V2.0 Mainboard (KS4034、KS4034F)	1
11		Keyestudio Mecanum Car Base Board（include motor driver)	1
12		M3*20MM Dual-pass Copper Pillar	4
13		4265c Lego Part	4
14		43093 Lego Part	4
15		Acrylic Gasket	1
16		M3*6MM Flat Head Screw	10
17		HC-SR04 Ultrasonic Sensor	1
18		M3*8MM Flat Head Screw	10
19		M3 Nickle-plated Nut	10

20		M3*30MM Round Head Screw	9
21		M2 Nickle-plated Nut	3
22		M2*8MM Round Head Screw	3
23		M1.4 Nickle-plated Nut	6
24		M1.4*10MM Round Head Screw	6
25		M2.3*16MM Round Head Screw	4
26		Remote Control（KS4034F、KS4035F） Remote Control (without batteries)（KS4034、KS4035）	1
27		Plastic String 3*100MM	5
28		USB Cable	1
29		HX-2.54 2P DuPont Wire 100mm	1
30		XH2.54 5P DuPont Wire 100mm	1
31		HX-2.54 4P DuPont Wire 50mm	1
32		HX2.54mm-4P to 2.54 F-F DuPont Wire 150mm	1
33		XH2.54 3P DuPont Wire 50mm	2
34		3*40MM Screwdriver	1
35		TT Coupling	4

36		M1.2*5MM Round Head Self-tapping Screw	6

5.Preparations

5.1BBC Micro:bit

(1)What is Micro:bit?

Micro:bit is an open source hardware platform based on the ARM architecture launched by British Broadcasting Corporation (BBC) together with ARM, Barclays, element14, Microsoft as well as other institutions. The core device is a 32-bit Arm Cortex-M4 with FPU micro-processing.

It is just the size of a credit card but it's very powerful. The Micro:bit main board is equipped with a host of components such as a 5*5 LED dot matrix, 2 programmable buttons, an accelerometer, a compass, a thermometer, a touch-sensitive logo and a MEMS microphone, a Bluetooth module of low energy as well as a buzzer and so on, making it empower to play a variety of sounds without external devices.

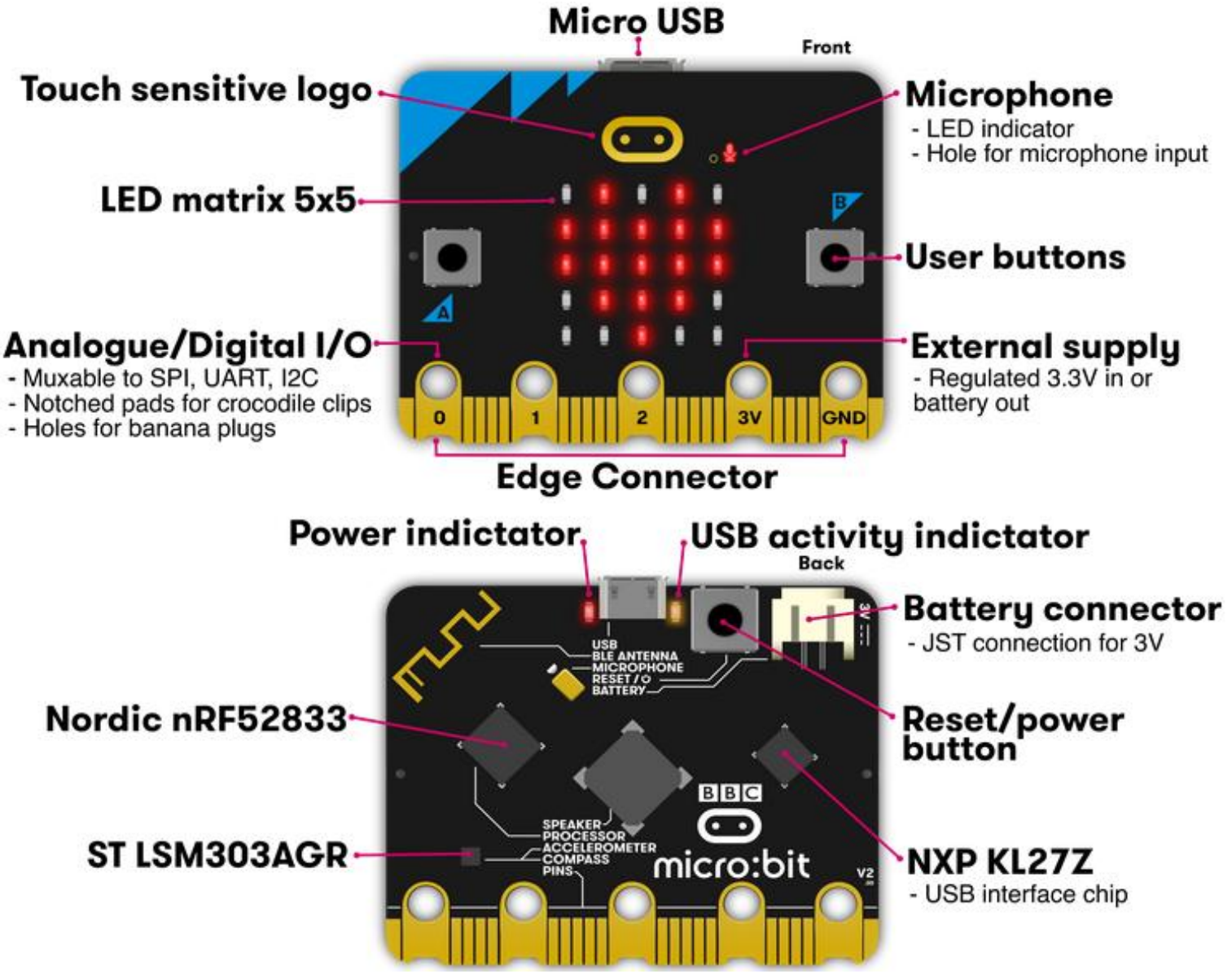
Moreover, this board supports a sleeping mode to lower power consumption of batteries and it can be entered if users long press the Reset & Power button on the back of it.

Micro: Bit development board is easy to use and expand, the bottom gear design of the gold finger can be used to interact with various electronic components via fixed alligator clip. It is capable of reading the data of sensors, controlling servos and RGB lights and inserting an expansion board so as to connect various sensors.

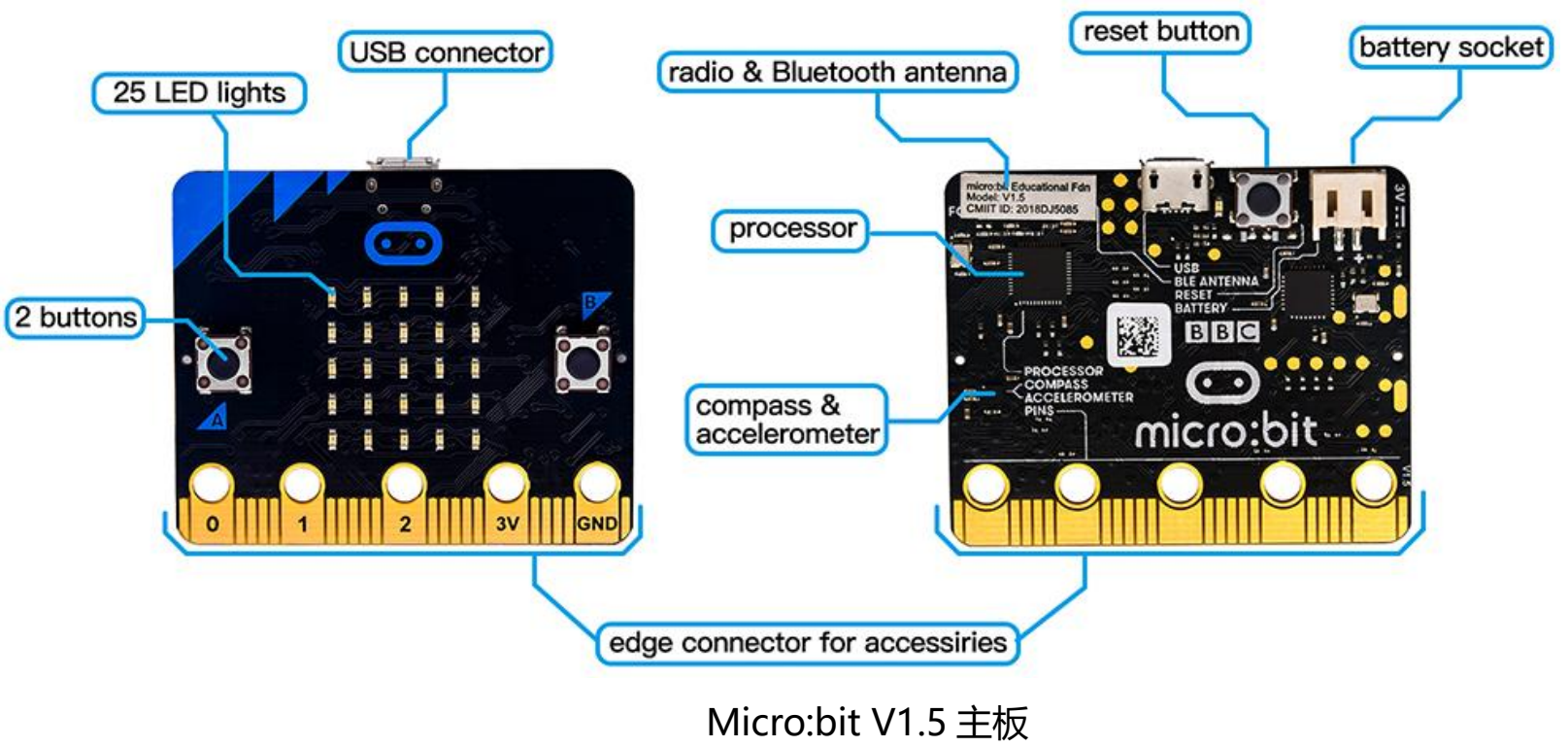
Furthermore, it also supports a variety of codes and graphical programming platforms, and is compatible with almost all PCs and mobile devices and a free-installation driver. It has high integration electronic modules and a serial port monitoring function for easy debugging.

The board is widely used in programming video games, interactions between light and sound, robots controls, scientific experiments, wearable devices as well as some cool inventions like robots and musical instruments.

(2) Layout

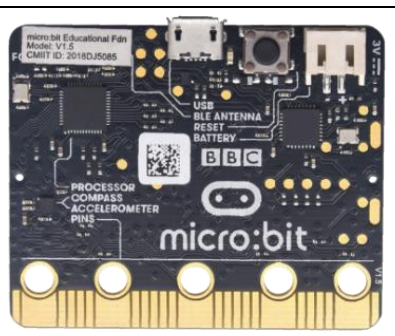


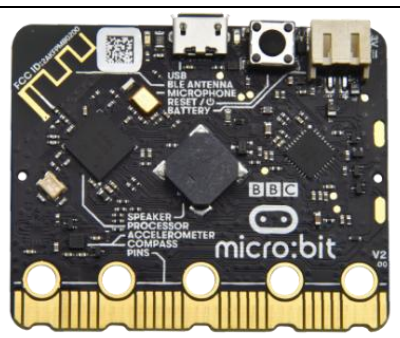
Micro:bit V2 主板



Micro:bit V1.5 主板

Comparison between V2.0 & V1.5





	V1.5	V2
PROCESSOR	Nordic Semiconductor nRF51822	Nordic Semiconductor **nRF52833**
MEMORY	256KB Flash, 16KB RAM	**512KB** Flash, **128KB** RAM
INTERFACECHIP	NXP KL26Z, 16KB RAM	NXP **KL27Z, 32KB** RAM
MICROPHONE	N/A	**MEMS microphone and LED indicator**
SPEAKER	N/A	**On board speaker**
TOUCH	N/A	**Touch sensitive logo**
EDGE CONNECTOR	25pins,PWM,I2C,SPI and Extension interface. 3 ring pins for connectin crocodile clips/banana plugs.	
	3 dedicated GPIO	**4 dedicated GPIO Notched for easier connection**
I2C	Shared (mux) I2C bus	**Dedicated I2C bus**
WIRELESS	2.4GHz Radio/BLE Blutooth 4.0	2.4GHz Radio/BLE **Blutooth 5.0**
POWER	Micro USB 5V power supply, 3V port or battery power supply	Micro USB 5V power supply, 3V port or battery power supply **LED Indicator, Power off (push and hold power button)**
CURRENT AVAILABLE	90mA	**200mA**
MOTION SENSOR	ST LSM 303	
PROGRAMMING SOFTWARE	C++, Makecode, Python, Scratch	
SIZE	5cm(W) x 4cm(H)	

For the Micro: bit main board V2, pressing the Reset & Power button , it will reset the micro: bit and run the program again. If you hold it down, the red LED will slowly get darker. When the power indicator getting dimmer, release the button, then your micro: bit board will enter into sleep mode for power saving. Only in this way can make your battery more durable. And you could press this button again to ‘wake up’ your Micro:bit.

For more information, please resort to the following links:

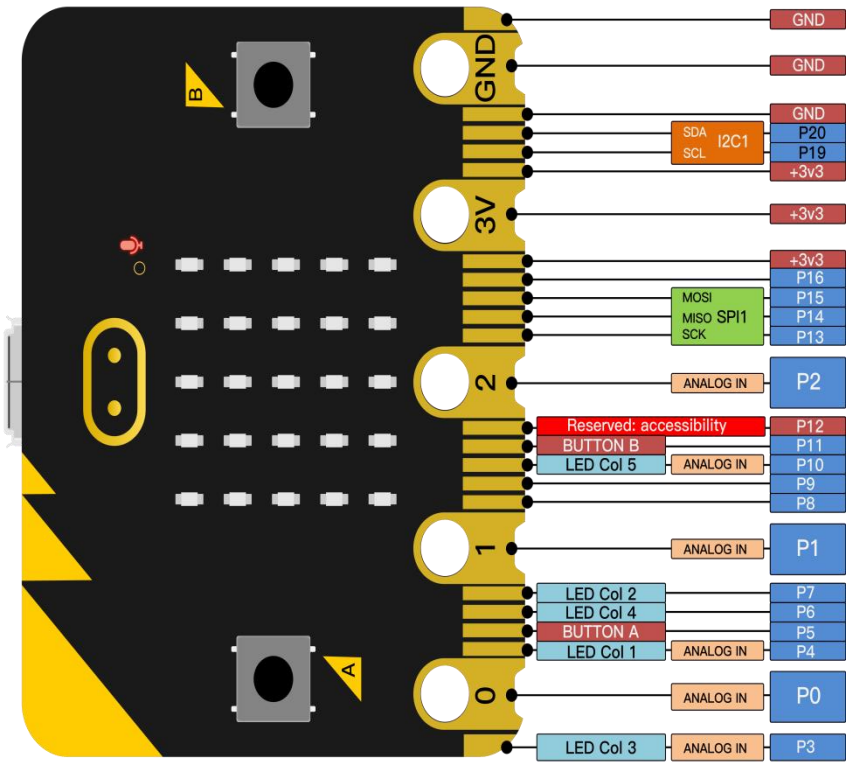
https://tech.microbit.org/hardware/

https://microbit.org/new-microbit/

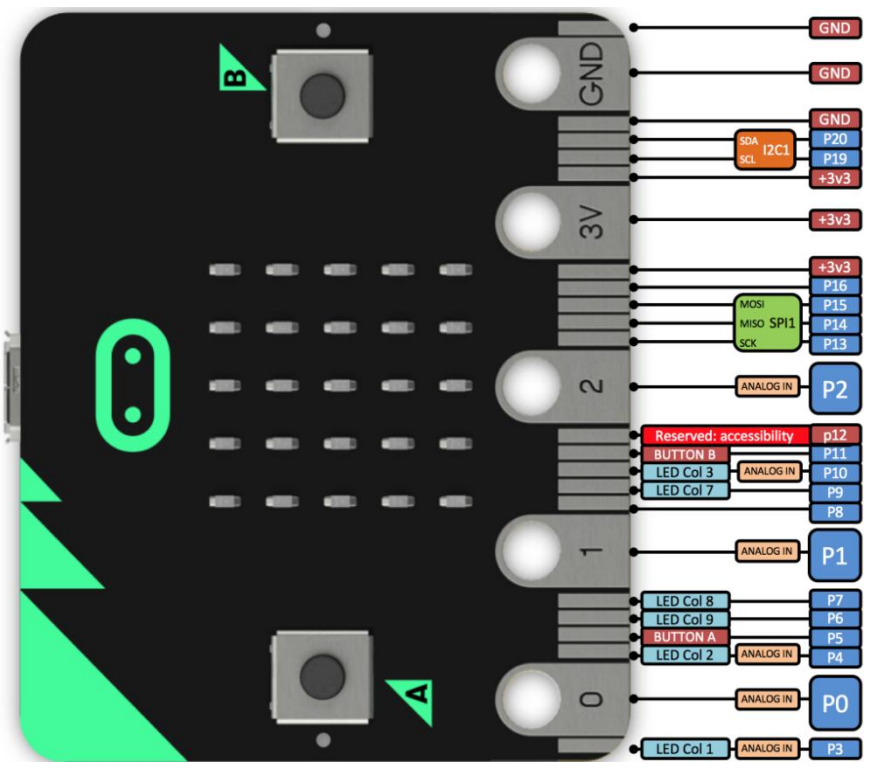
https://www.microbit.org/get-started/user-guide/overview/

https://microbit.org/get-started/user-guide/features-in-depth/

(3) Pin out



Micro:bit V2 Mainboard



Micro:bit V1.5 Mainboard

Functions:

GPIO	P0，P1，P2，P3，P4，P5，P6，P7，P8，P9，P10，P11，P12，P13，P14，P15，P16，P19，P20
ADC/DAC	P0，P1，P2，P3，P4，P10
IIC	P19（SCL），P20（SDA）
SPI	P13（SCK），P14（MISO），P15（MOSI）
PWM（used frequently）	P0，P1，P2，P3，P4，P10
PWM（not frequently used）	P5、P6、P7、P8、P9、P11、P12、P13、P14、P15、P16、P19、P20
Occupied	P3(LED Col3)，P4(LED Col1)，P5(Button A)，P6(LED Col4)，P7(LED Col2)，P10(LED Col5)，P11(Button B)

Please browse the official website for more details：https://tech.microbit.org/hardware/edgeconnector/ https://microbit.org/guide/hardware/pins/

(4)Notes for the Micro:bit V2 main board

a. It is recommended to cover with a silicone protector to prevent short circuit for its sophisticated electronic components.

b. Its IO port is very weak in driving since it can merely handle current less than 300mA. Therefore, do not connect it with devices operating in a large current, such as MG995 servo and DC motor or it will get burnt. Furthermore, you

must figure out the current requirements of the devices before you use them and it is generally recommended to use the board together with a Micro:bit expansion board.

c. It is recommended to power the main board via the USB interface or the battery of 3V. The IO port of this board is 3V, so it does not support sensors of 5V. If you need to connect sensors of 5 V, a Micro: Bit expansion board is required.

d. When using pins(P3, P4, P6, P7 and P10)shared with the LED dot matrix, blocking them from the matrix or the LEDs may display randomly and the data about sensors connected maybe wrong.

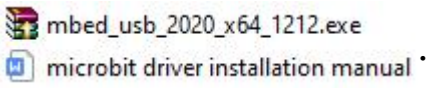
e. The battery port of 3V cannot be connected with battery more than 3.3V or the main board will be damaged.

f. Forbid to operate it on metal products to avoid short circuit.

To put it simple, Micro:bit V2 main board is like a microcomputer, which has made programming at our fingertips and enhanced digital innovation. And as for programming environment, BBC provides a website: https://microbit.org/code/, which has a graphical MakeCode program easy for use.

5.2.Install Micro:bit driver

Micro: Bit can be installed without USB driver. However, if your computer fails to recognize the main board, you can install the diver too.

Just enter the file folder 

.

6.Keyestudio 4WD Mecanum Robot Car V2.0

It is a programmable car based on BBC micro:bit. It integrates a motor driver, a line tracking sensor and an IR receiver into the base plate, which also contains an ultrasonic sensor, a servo, 2 seven-color lights as well as 4 WS2812 RGB lights. The wiring is not complicated and it has Lego jacks to facilitate connection with other peripheral devices. Abundant hardware resources will enable you to master more knowledge and skills to create more technological inventions.

6.1. Keyestudio 4WD Mecanum Robot Car V2.0

This car can help you to better learn how to use the Micro:bit and make electronic knowledge accessible to you.

Functions

Sensor	Seven-color light	Decelerating DC motor	Servo	Ultrasonic sensor	Line Tracking Sensor	IR Receiver	WS2812 RGB light	Power switch
QTY	2	4	1	1	1	1	4	1

Note: the seven-color lights, line tracking sensor, WS2812 RGB lights, IR receiver and motor driver are integrated in the base plate.

Pins:

Pin on Micro:bit	Sensor
P10 P4 P3	Line Tracking Sensor
P14	Servo
P7	4 WS2812RGB Lights
P0	IR Receiver
P15P16	Ultrasonic Sensor

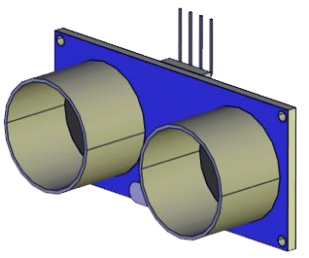
Power supply and Battery

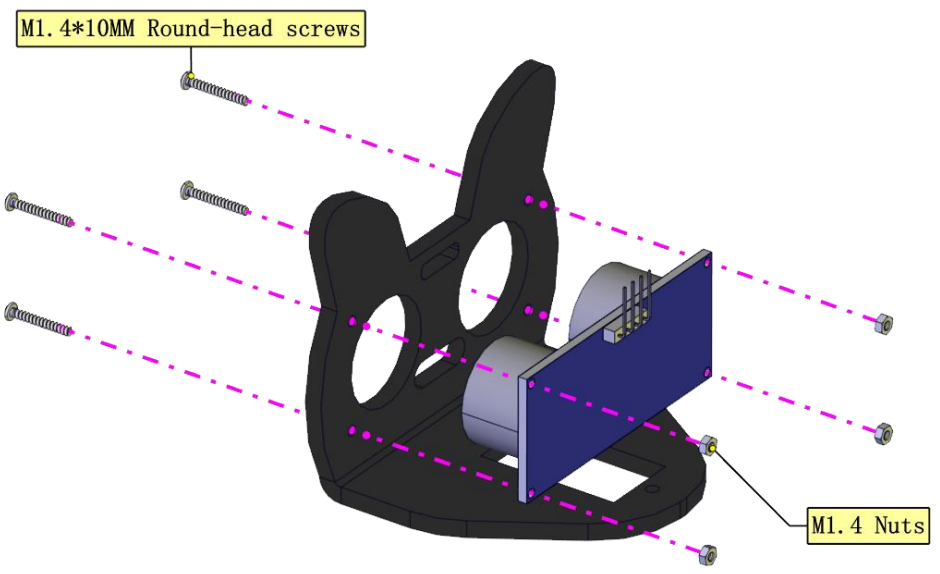
The keyestudio 4WD Mecanum Robot Car is powered by two 18650 batteries. The battery holder of the car is compatible with any type of 18650 lithium battery (rechargeable). You can use a universal battery charger to charge the 18650 lithium battery.

Note: This product does not contain batteries.

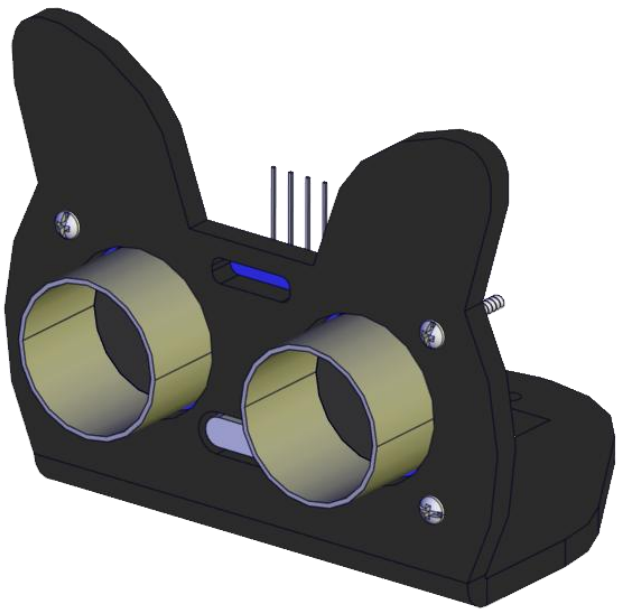
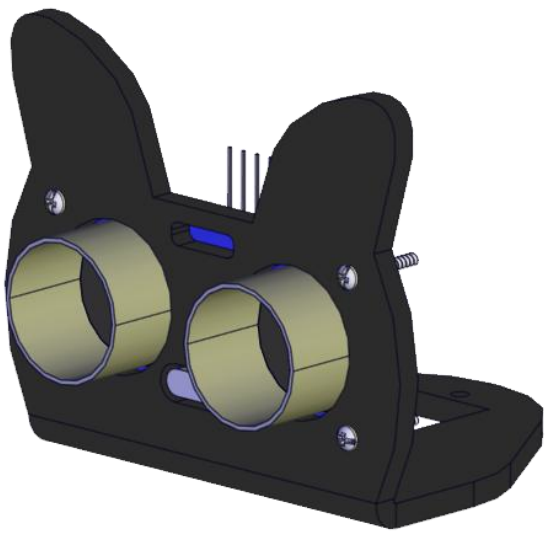
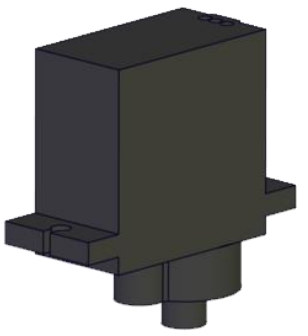
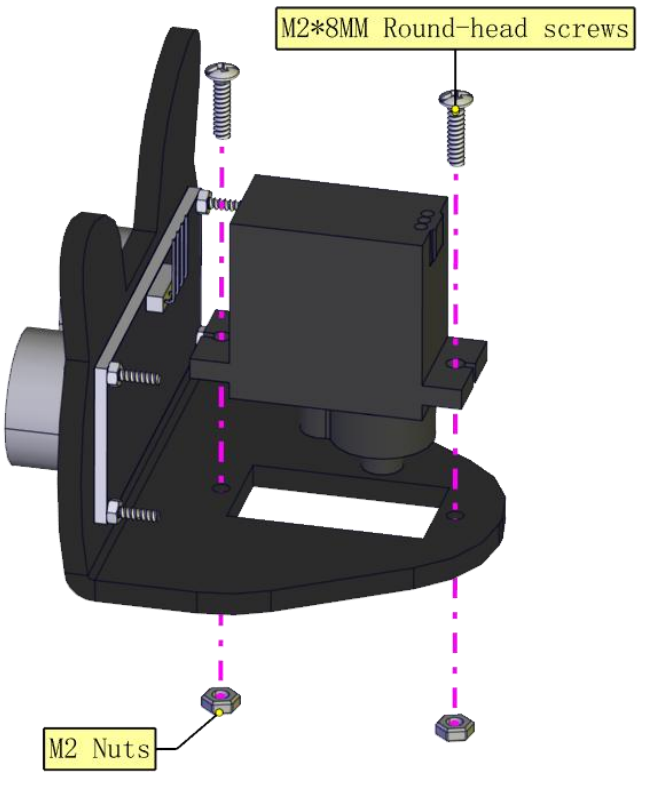
6.2. The Installation of Keyestudio 4WD Mecanum Robot Car V2.0

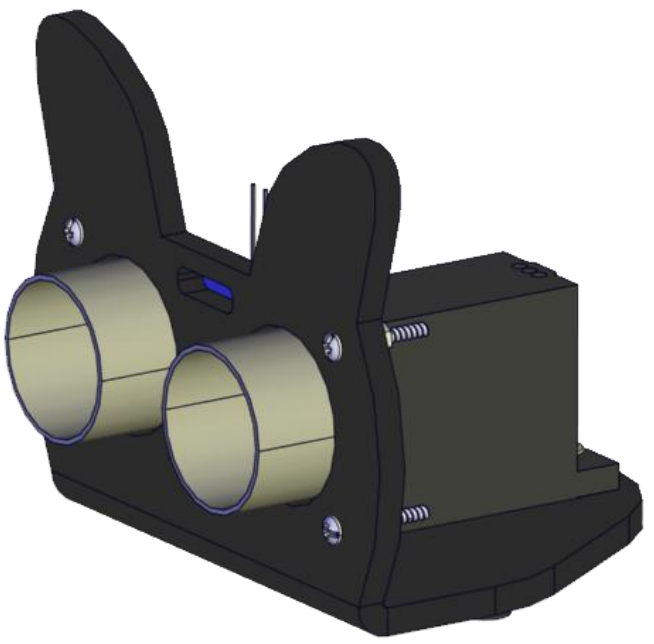
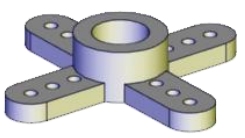
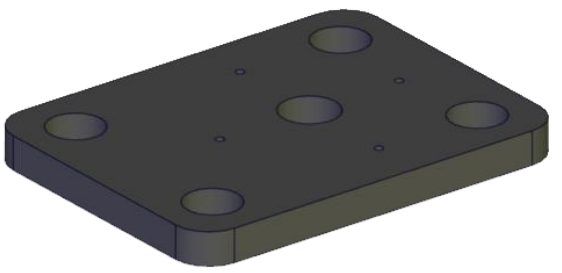
<table>
<tr><th colspan="2">Part 1</th></tr>
<tr><td>Components Needed</td><td>

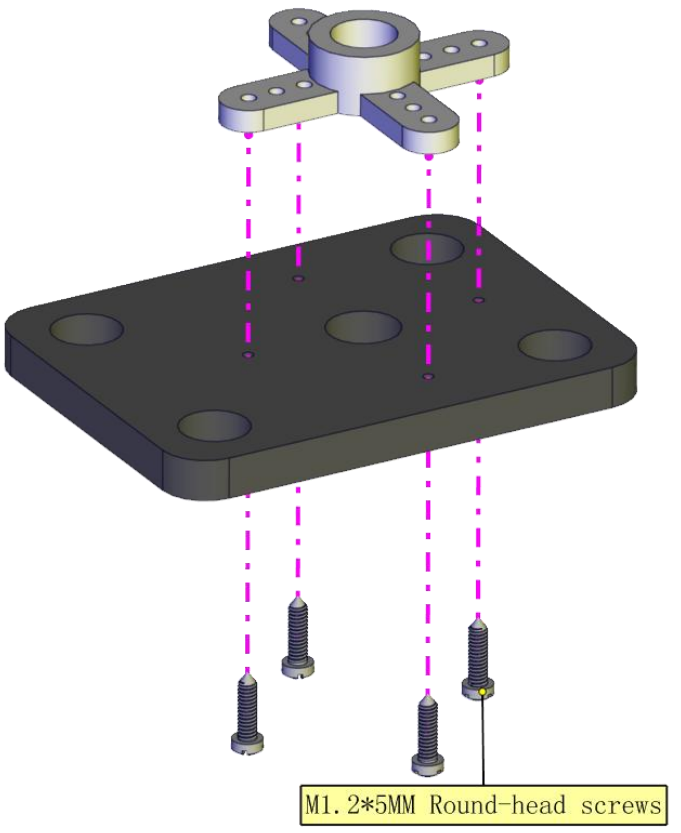
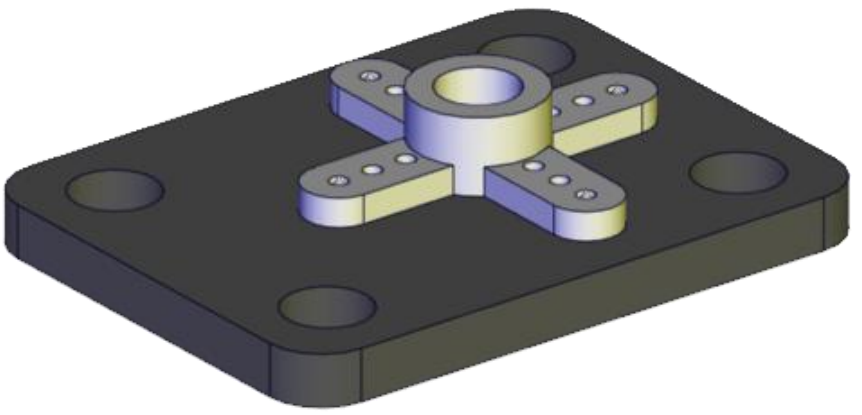



</td></tr>
<tr><td>Installation Diagram</td><td>


</td></tr>
</table>

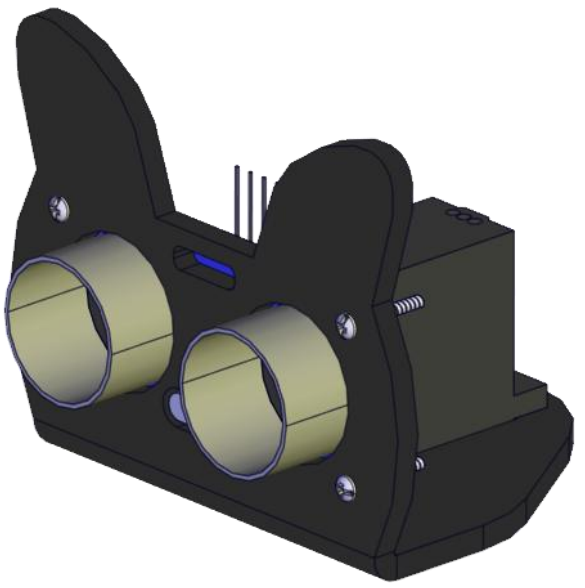
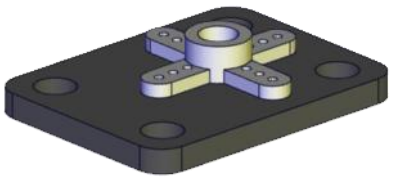
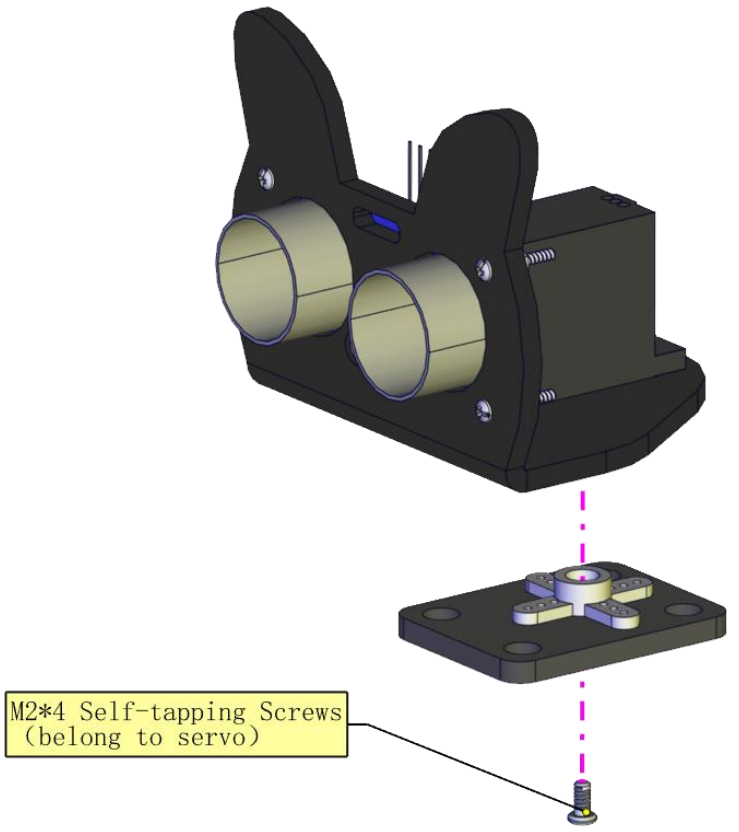
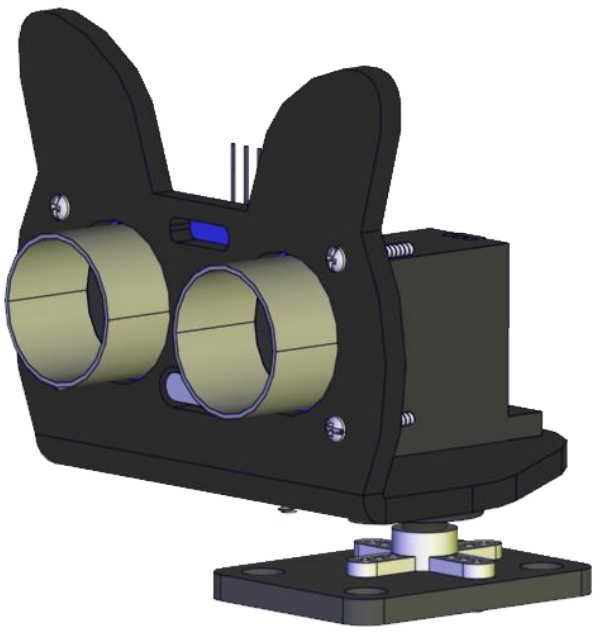
Prototype	
Part 2	
Components Needed	  Keyestudio Servo ×1  M2*8MM Round-head screws ×2  M2 Nuts ×2
Installation Diagram	

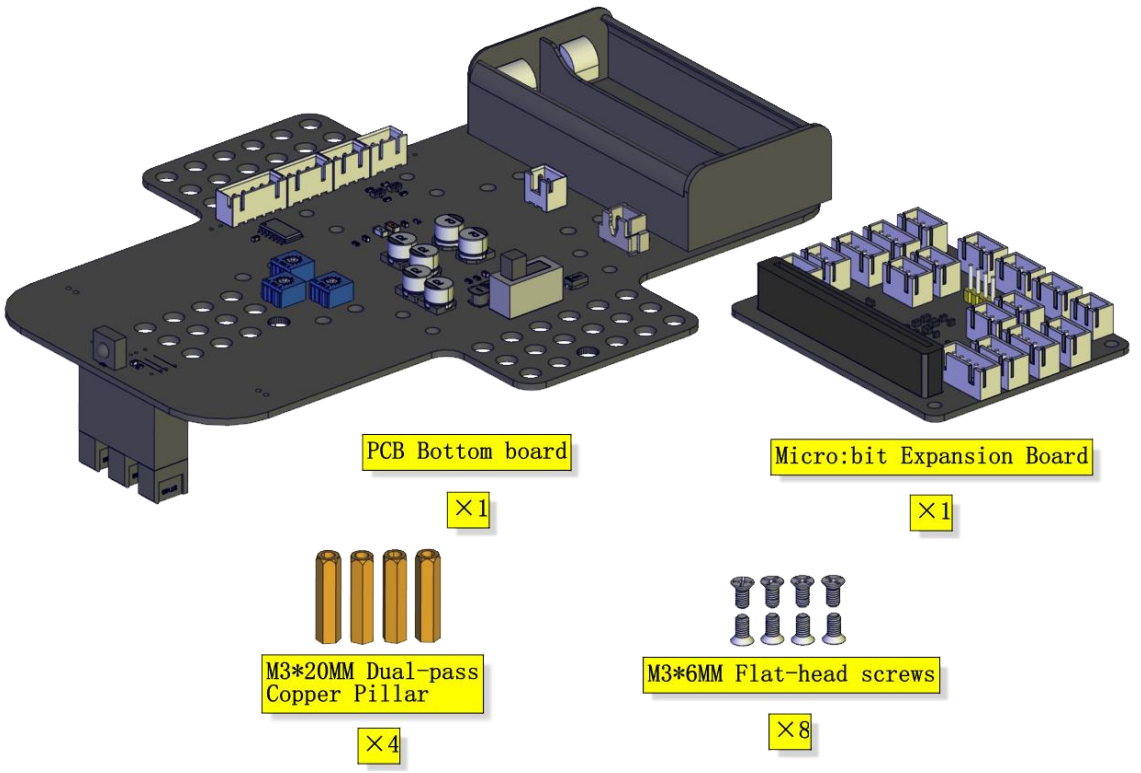
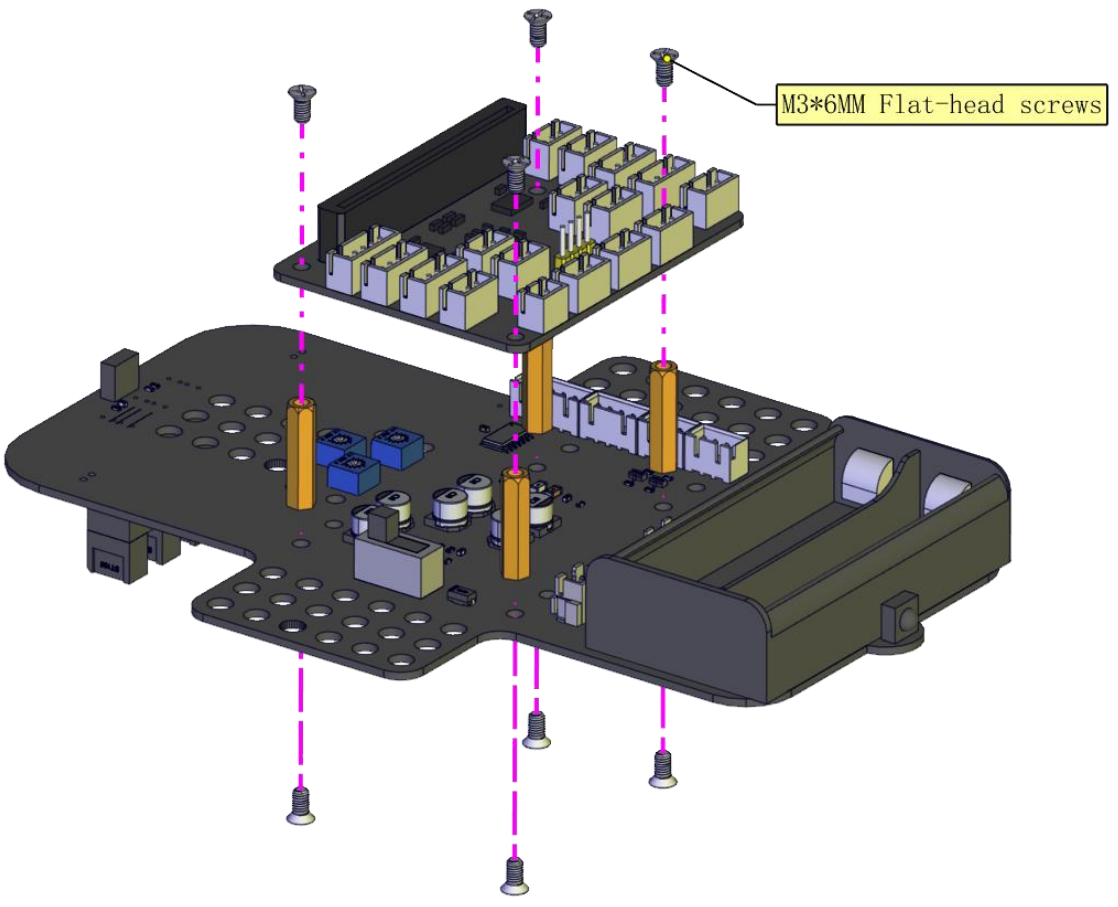
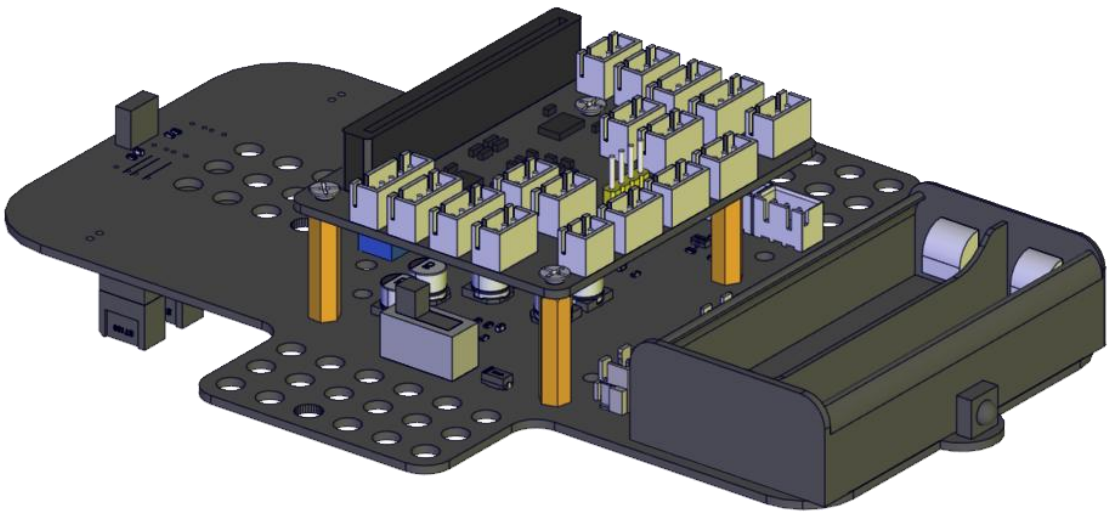
Prototype	
Part 3	
Components Needed	 Control horn(belong to servo) ×1  Acrylic Boards ×1  M1.2*5MM Round-head screws ×4

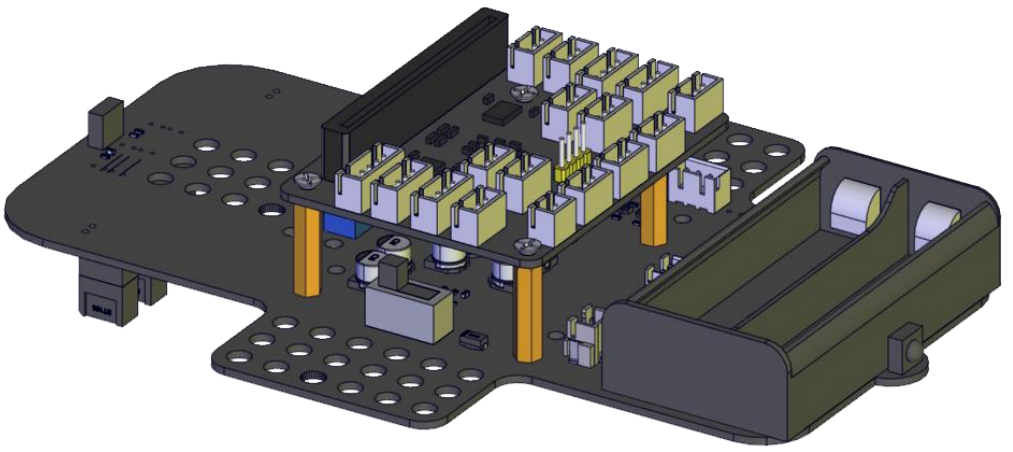
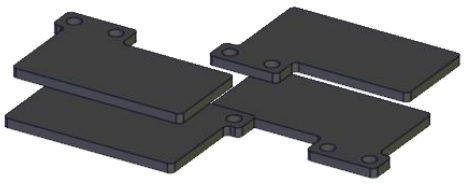
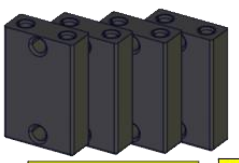
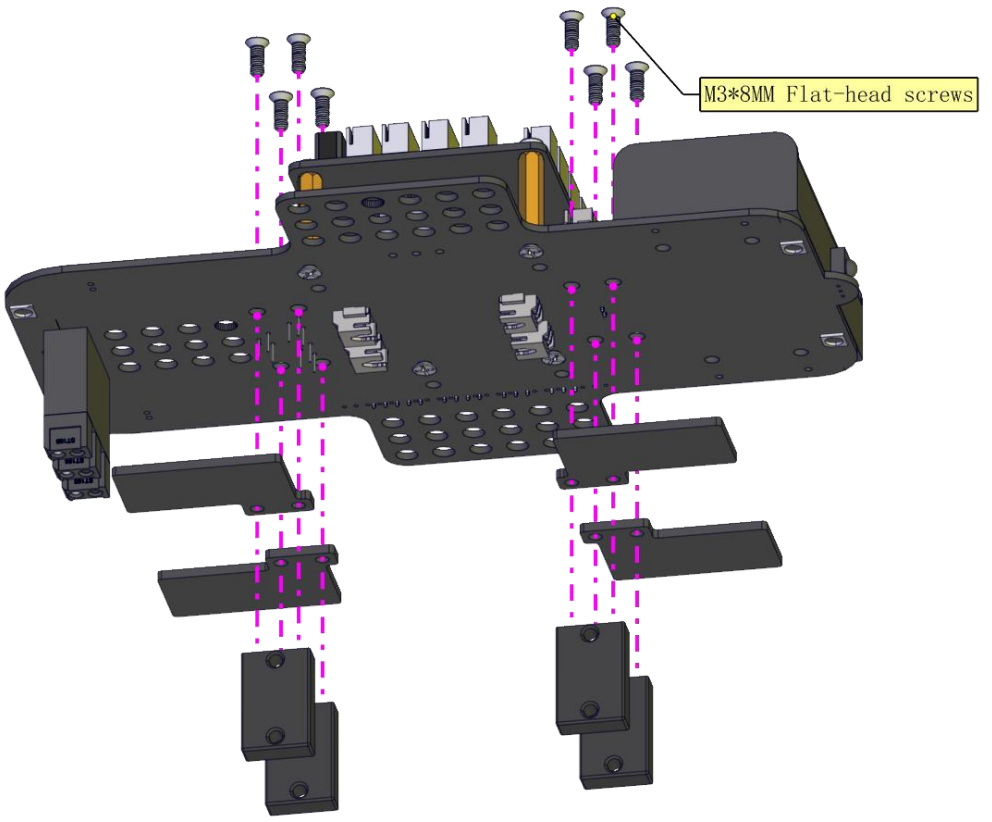
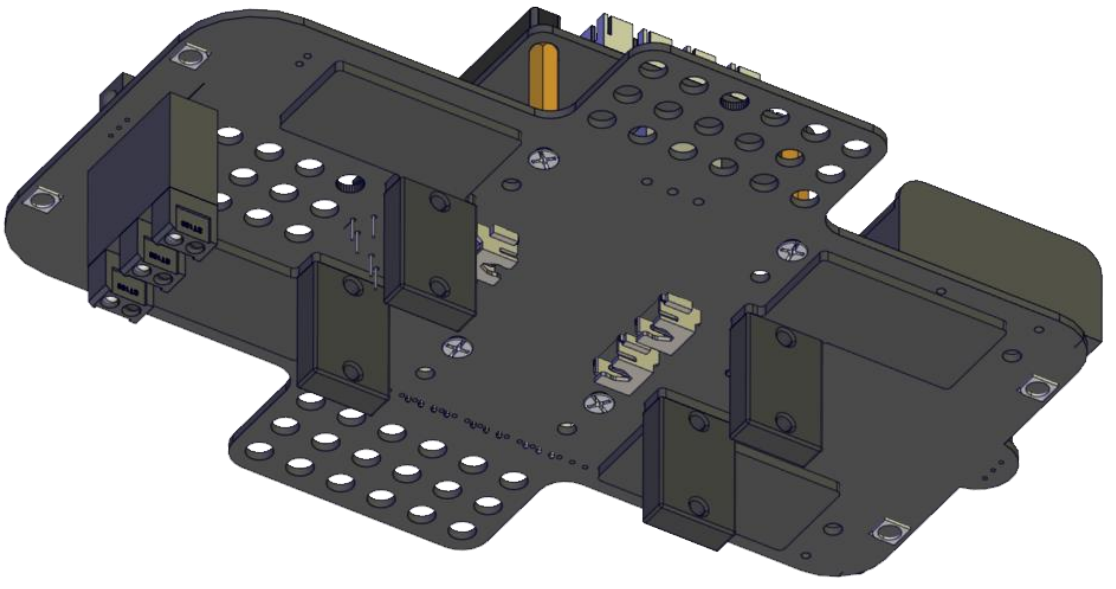
Installation Diagram	
Prototype	
Part 4 (adjust the angle of the servo first)	

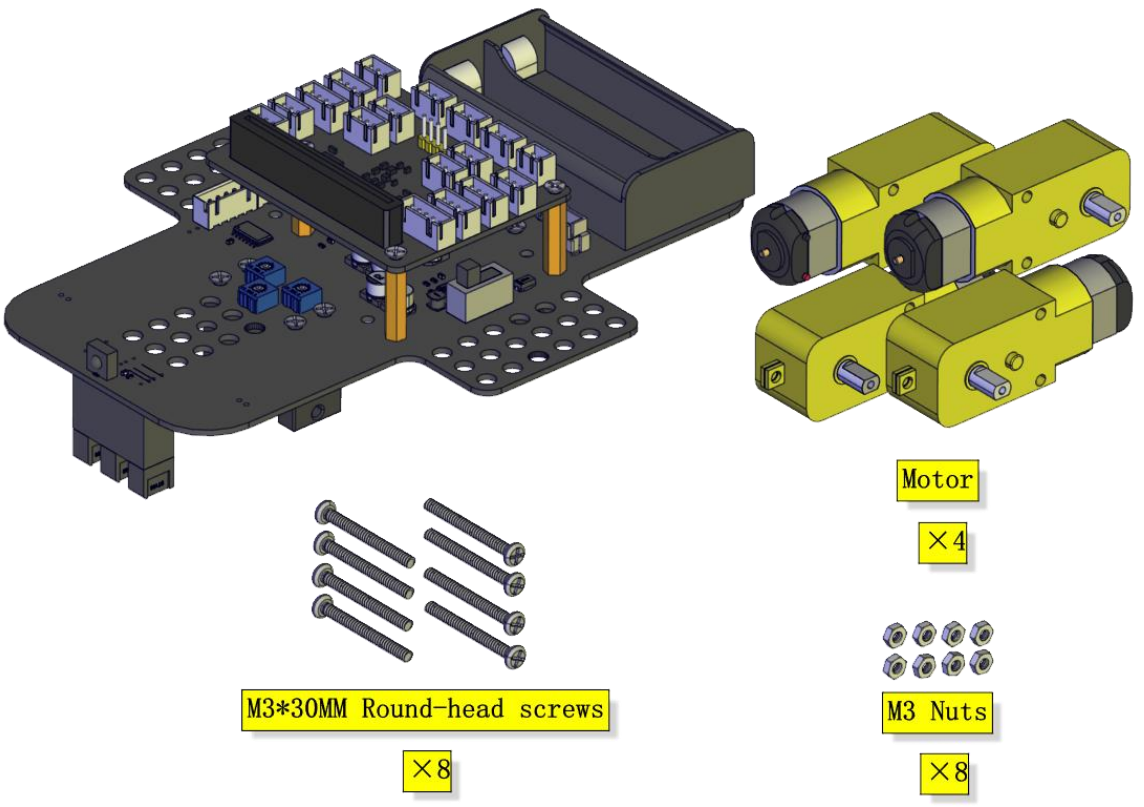
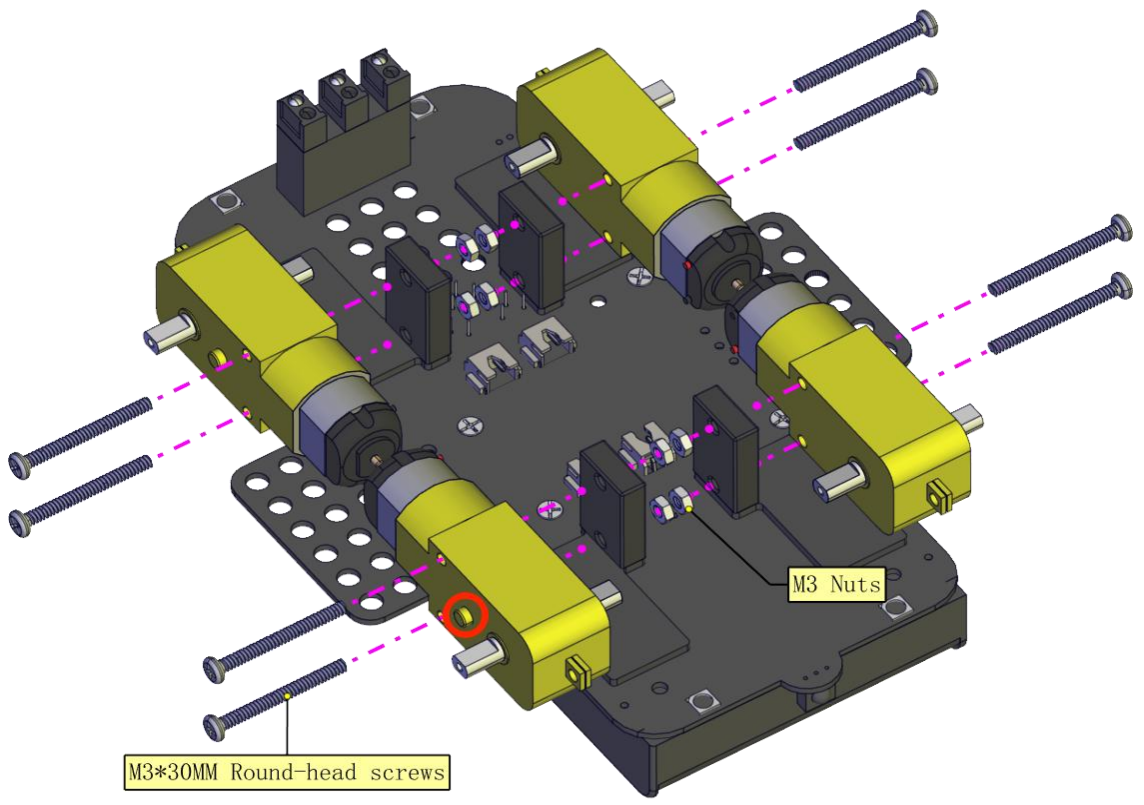
Adjust the angle of the servo to 90 degrees according to the test code	from microbit import *
	
	class Servo:
	  def __init__(self, pin, freq=50, min_us=600, max_us=2400, angle=180):
	  self.min_us = min_us
	  self.max_us = max_us
	  self.us = 0
	  self.freq = freq
	  self.angle = angle
	  self.analog_period = 0
	  self.pin = pin
	  analog_period = round((1/self.freq) * 1000) # hertz to miliseconds

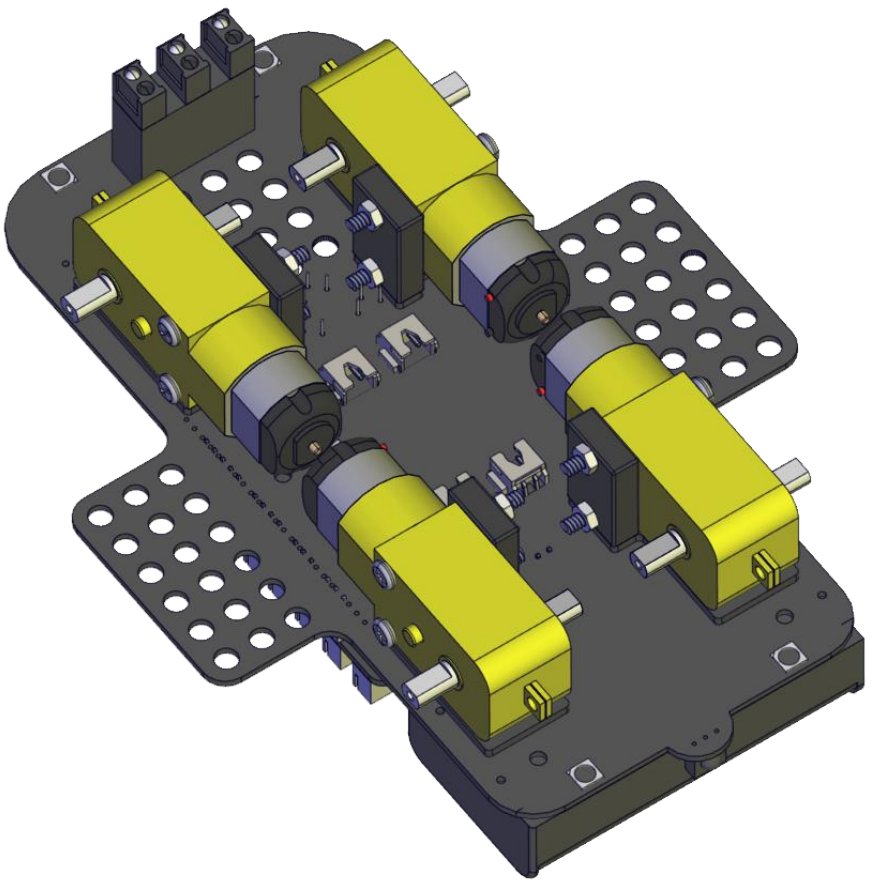
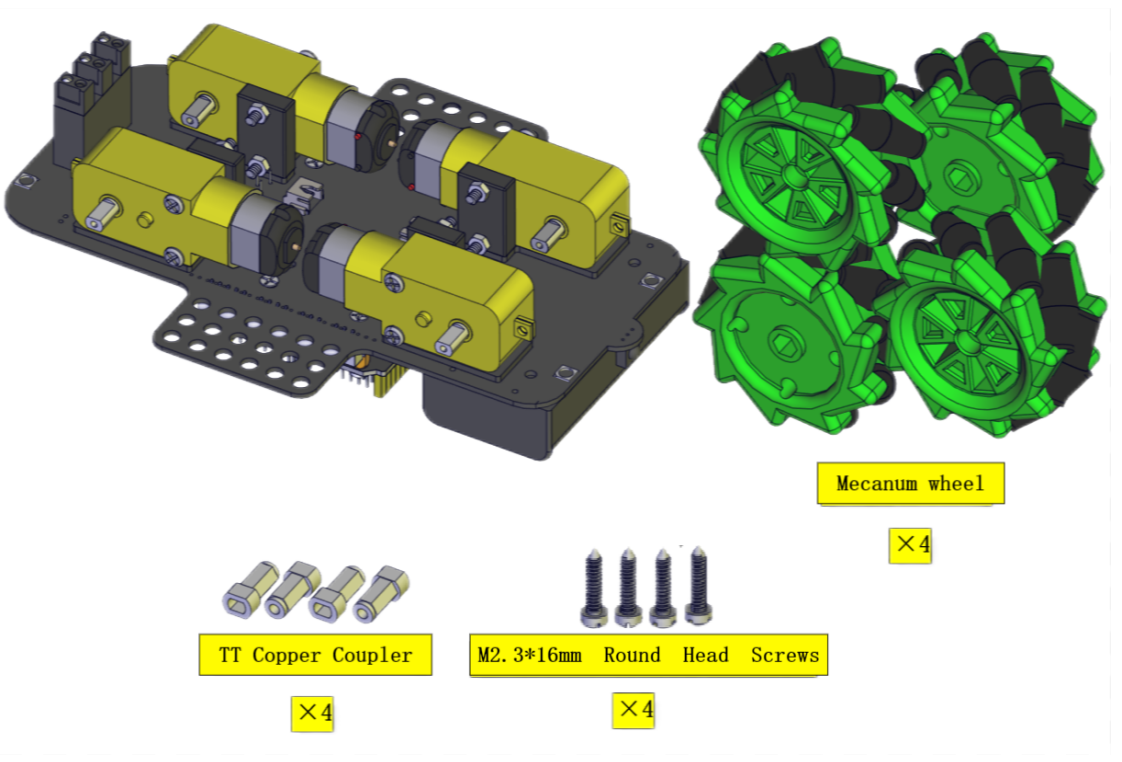
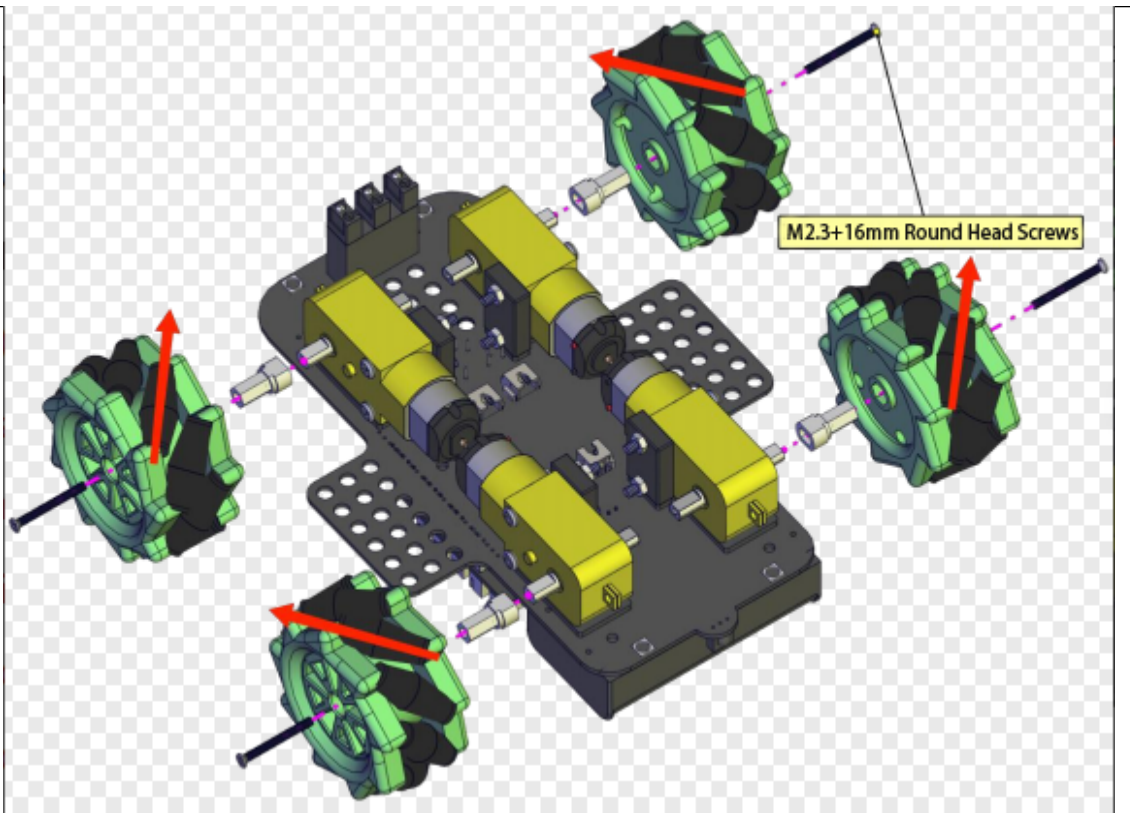
```
        self.pin.set_analog_period(analog_period)


    def write_us(self, us):
        us = min(self.max_us, max(self.min_us, us))
        duty = round(us * 1024 * self.freq // 1000000)
        self.pin.write_analog(duty)
        sleep(100)
        self.pin.write_analog(0)


    def write_angle(self, degrees=None):
        if degrees is None:
            degrees = math.degrees(radians)
        degrees = degrees % 360
        total_range = self.max_us - self.min_us
        us = self.min_us + total_range * degrees // self.angle
        self.write_us(us)



Servo(pin14).write_angle(90)
sleep(1000)
```

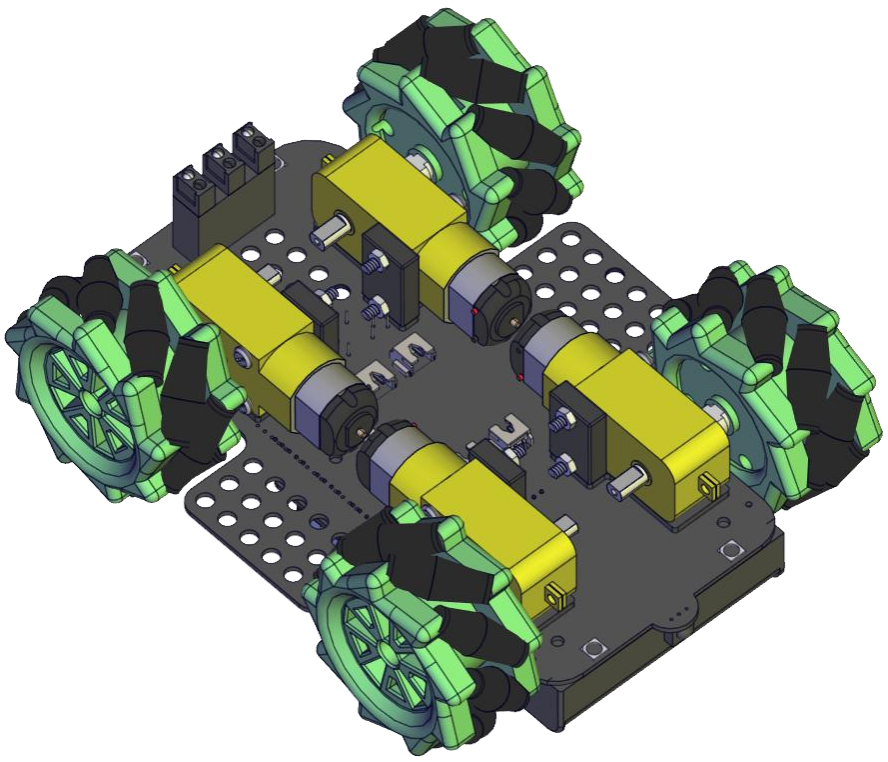
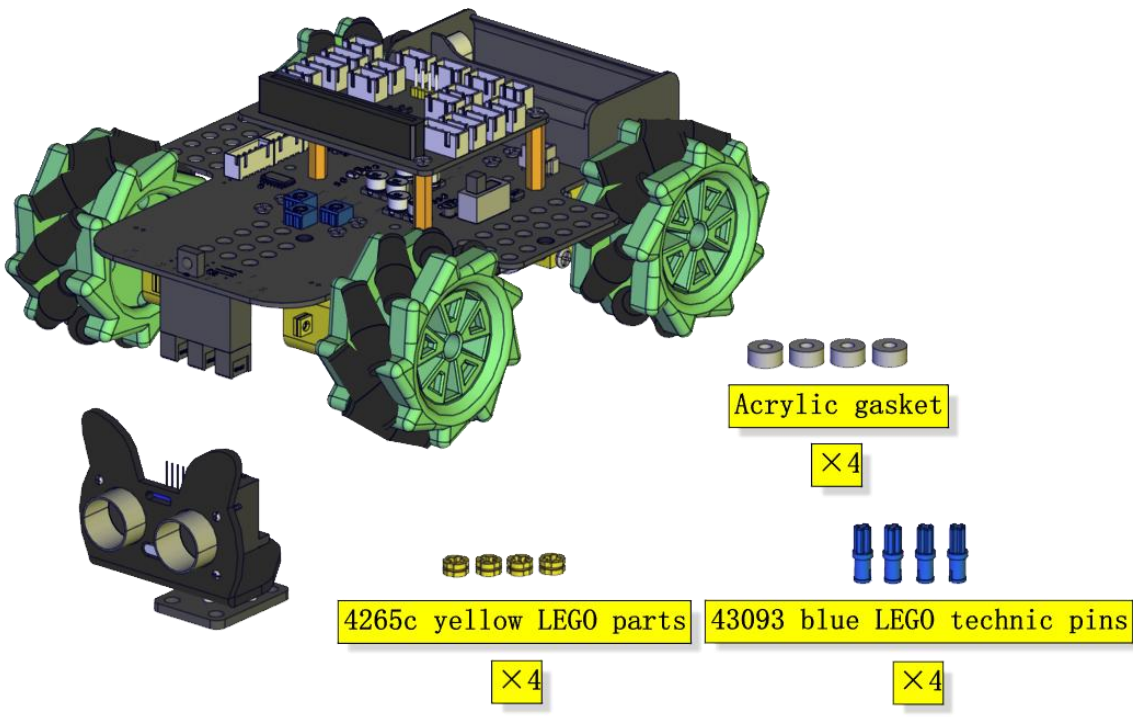
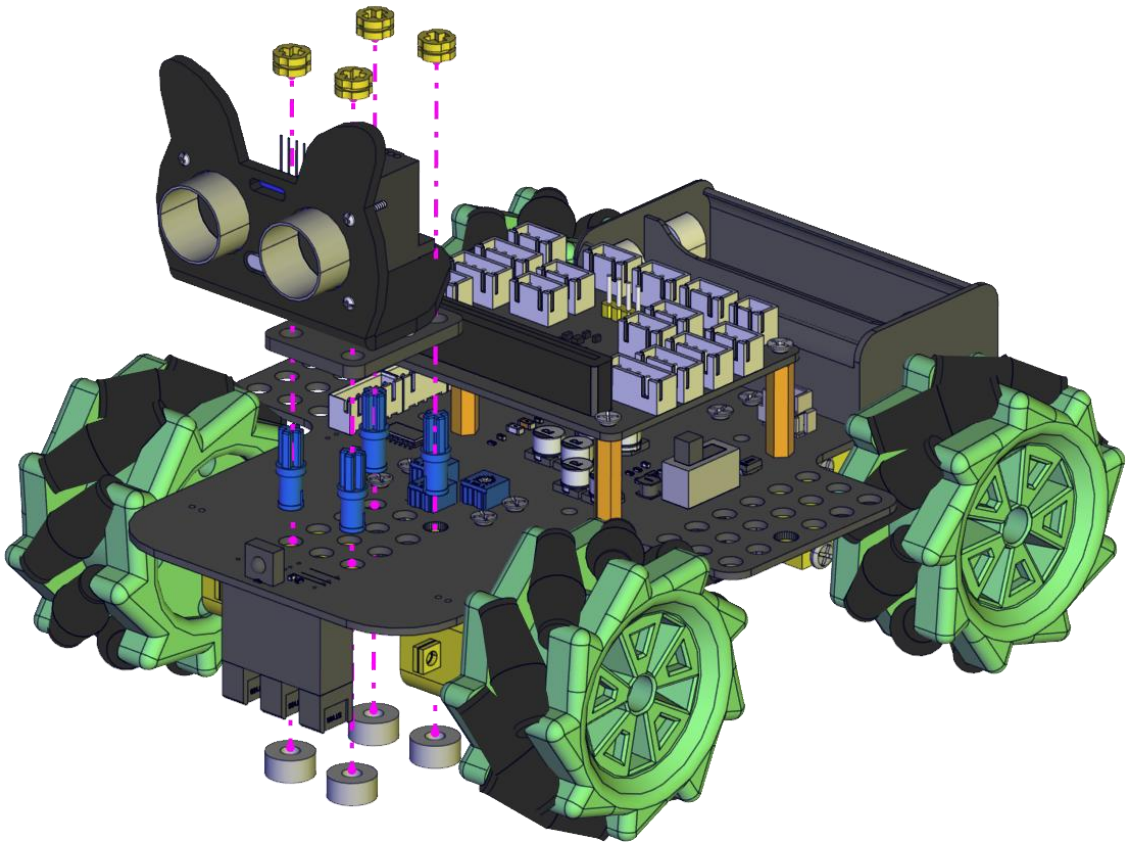
Components Needed	   ×1
Installation Diagram (mind the installation direction)	
Prototype	

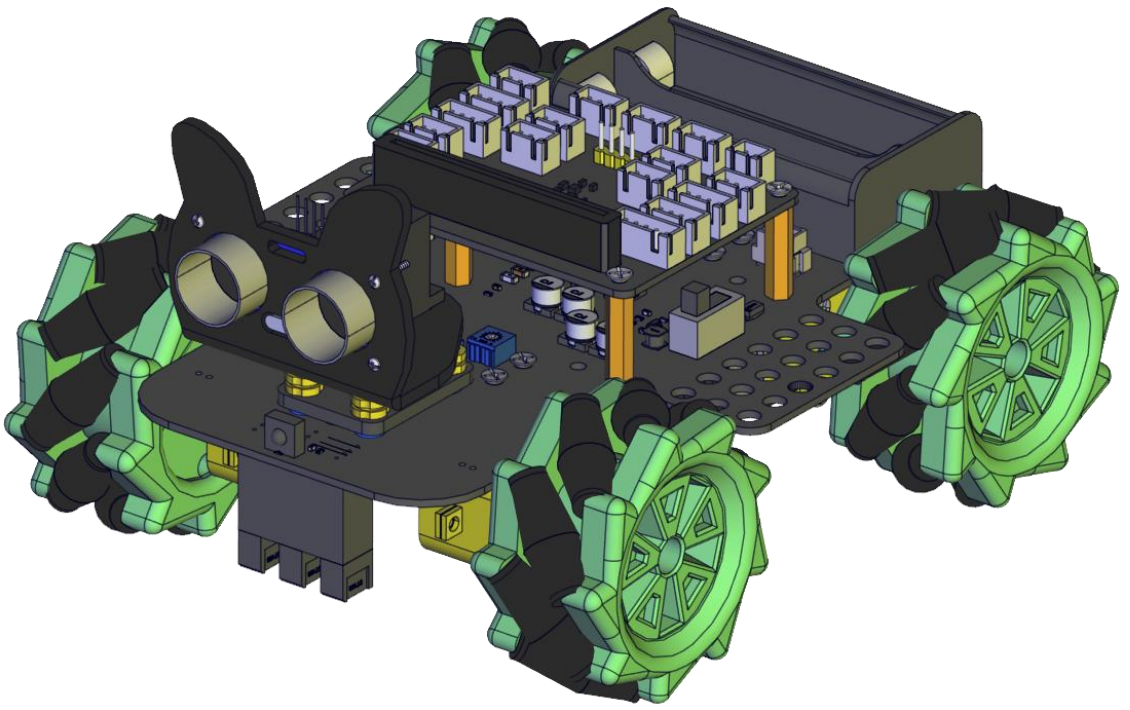
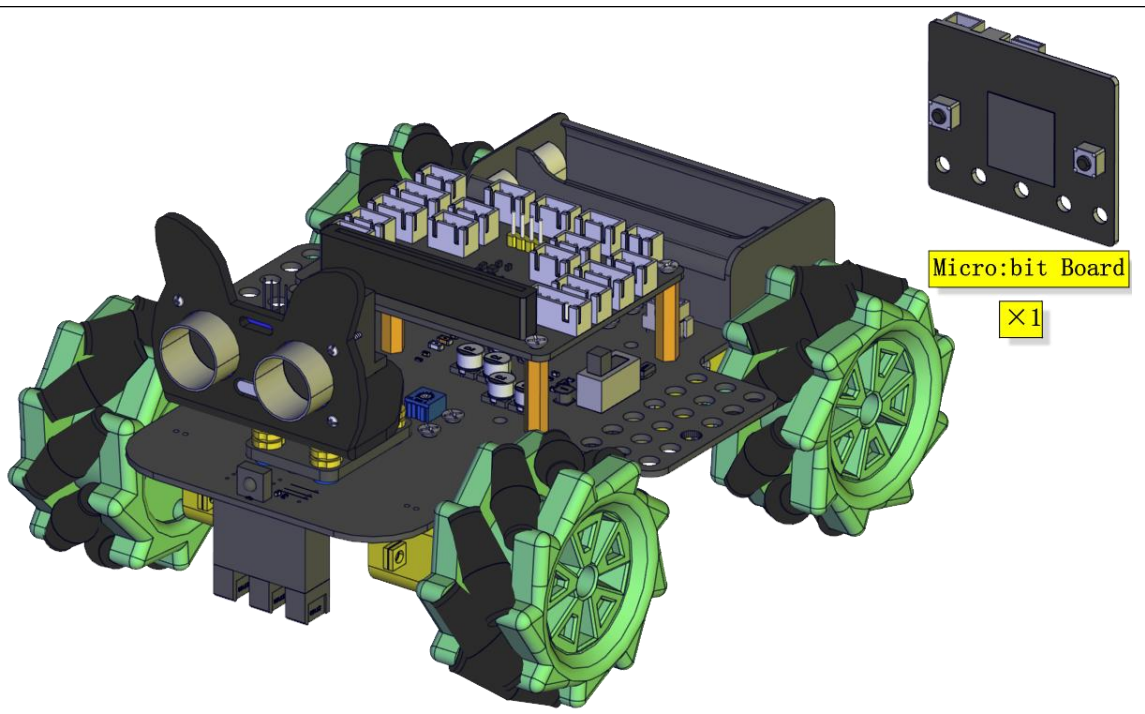
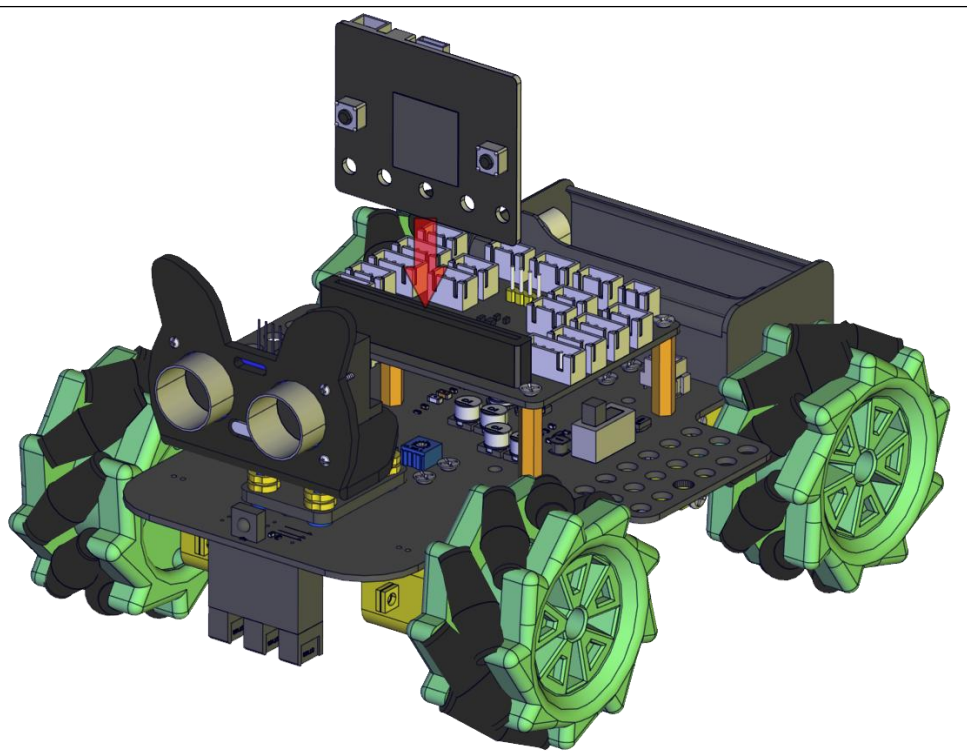
Part 5	
Components Needed	
Installation Diagram	
Prototype	

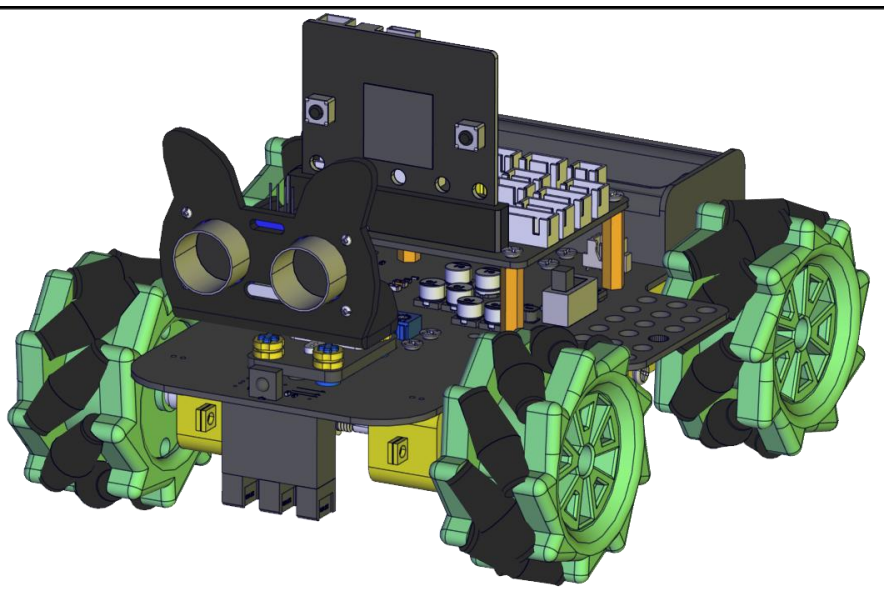
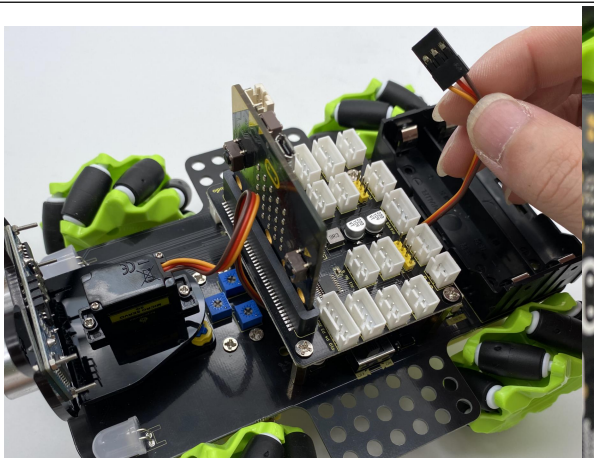
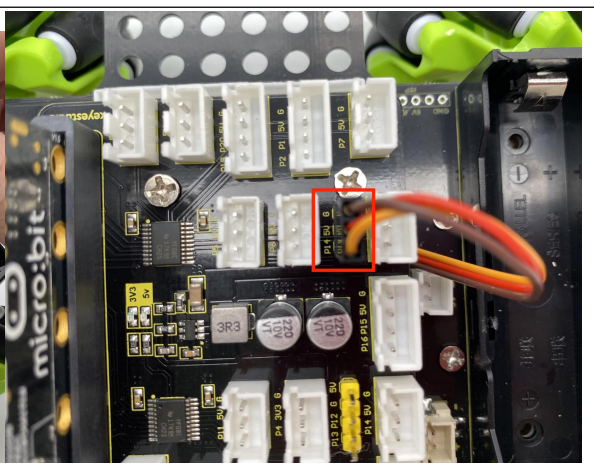
	Part 6
Components Needed	  Acrylic Boards ×4  Fixed parts  M3*8MM Flat-head screws ×8
Installation Diagram	
Prototype	

	Part 7
Components Needed	
Installation Diagram (mind the direction of the motor)	

Prototype	
Part 8	
Components Needed	
Installation Diagram (Pay attention to the installation direction of the mecanum wheel)	

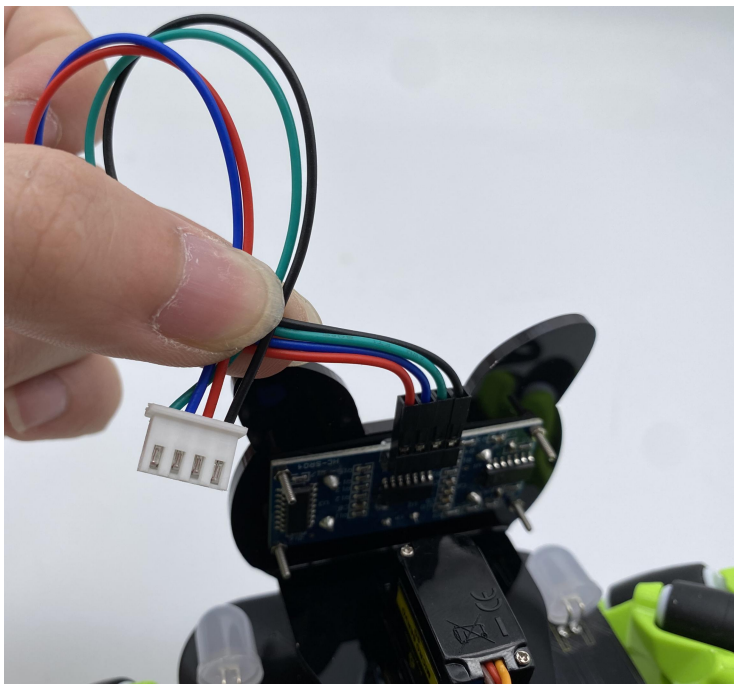
Prototype	
Part 9	
Components Needed	
Installation Diagram	

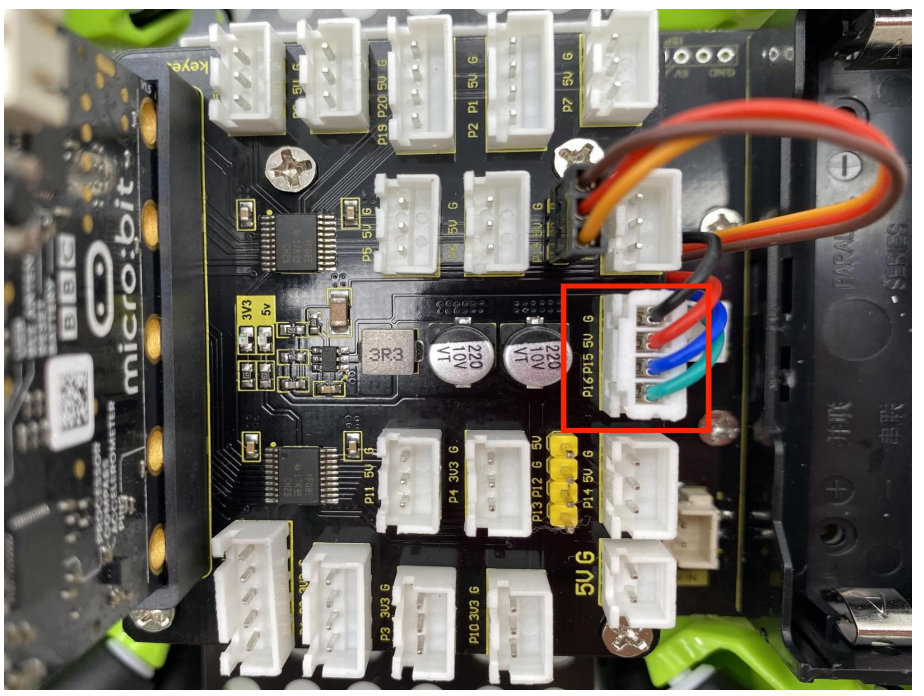
Prototype	
Part 10	
Components Needed	
Installation Diagram	

Prototype	
Wiring Diagram	
The wiring of the servo	 

Servo	Expansion Board
Brown	G
Red	V
Yellow	P14

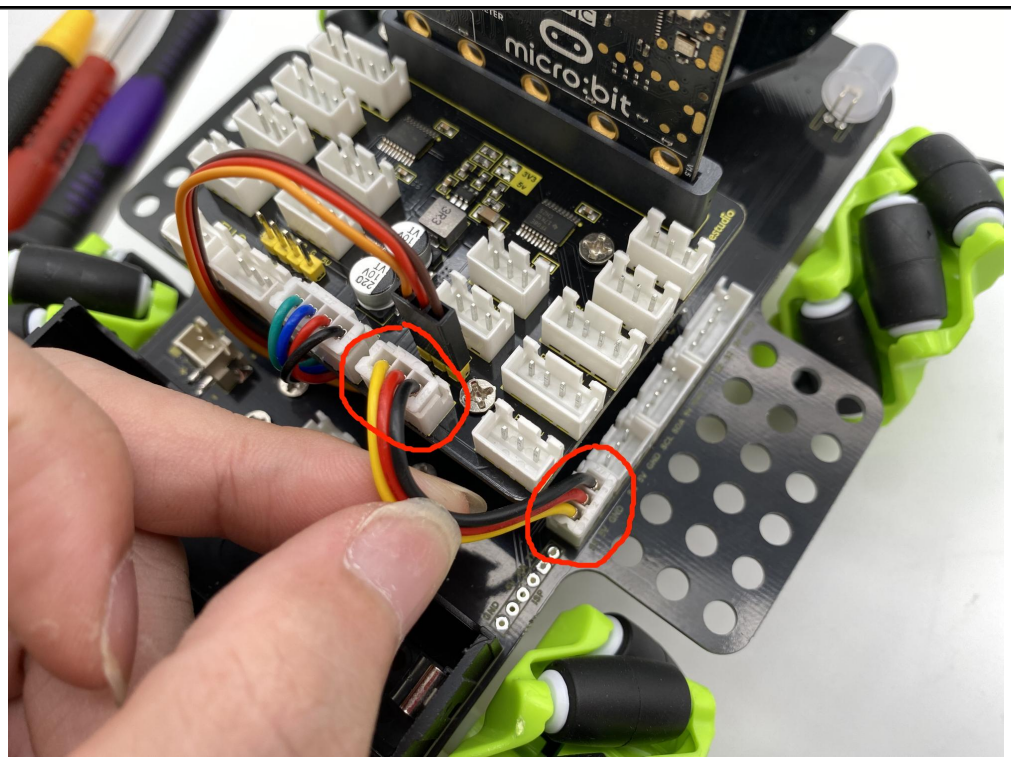
The wiring of the ultrasonic sensor





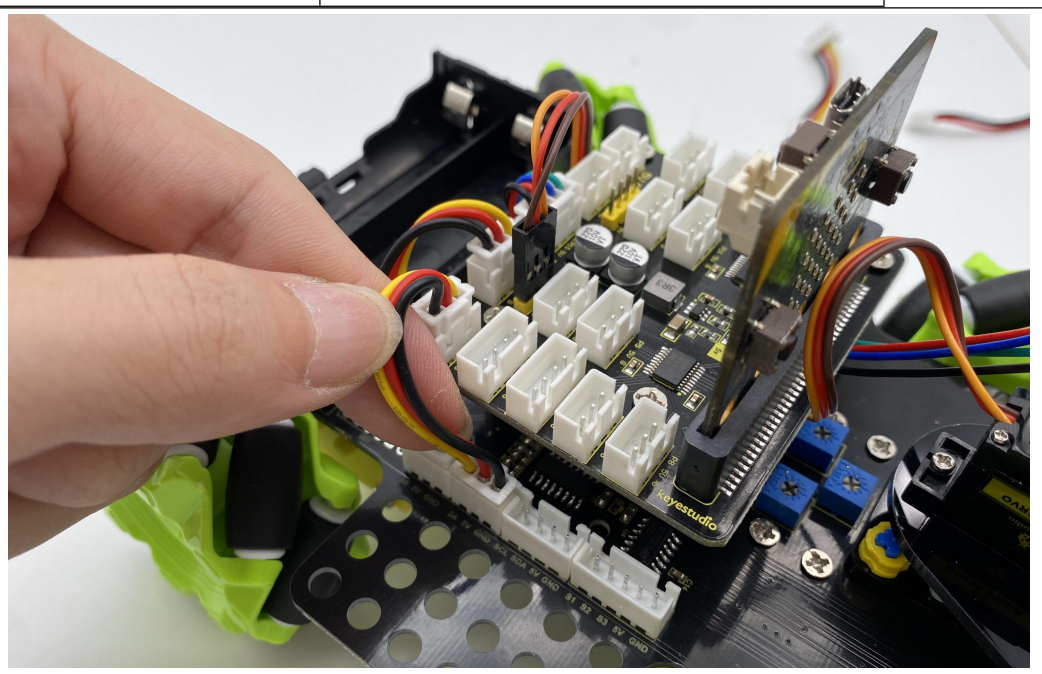
Ultrasonic Sensor	Expansion Board
VCC	5V
TRIG	P15
ECHO	P16
GND	G

The wiring of the IR receiver module



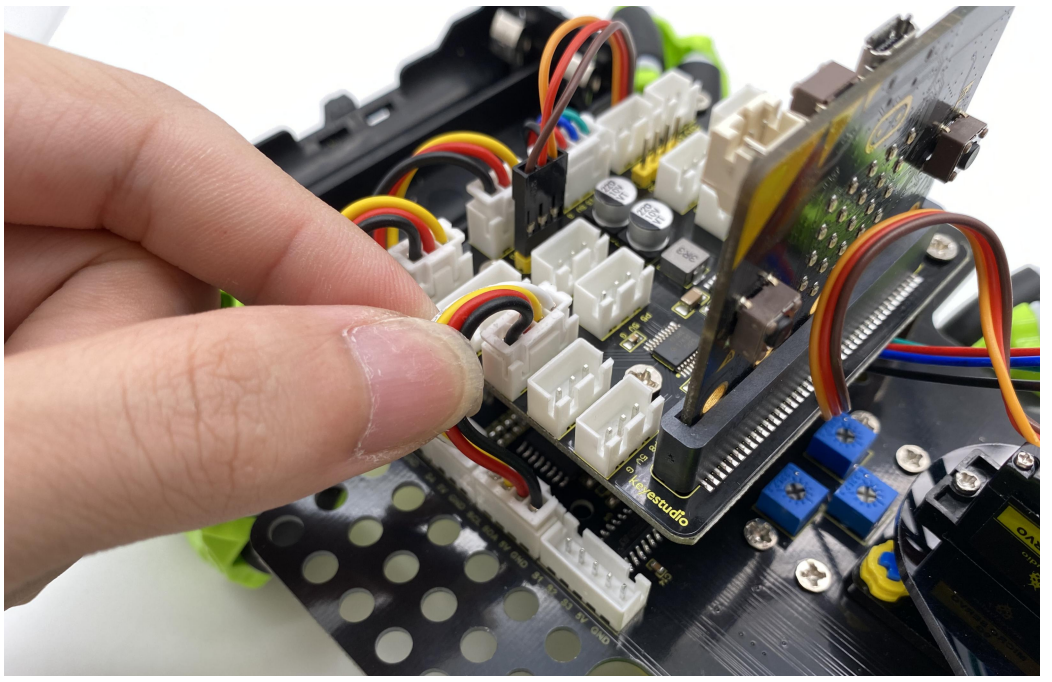
Driver Board	Expansion Board
GND	G
5V	5V
S5	P0

The wiring of the RGB



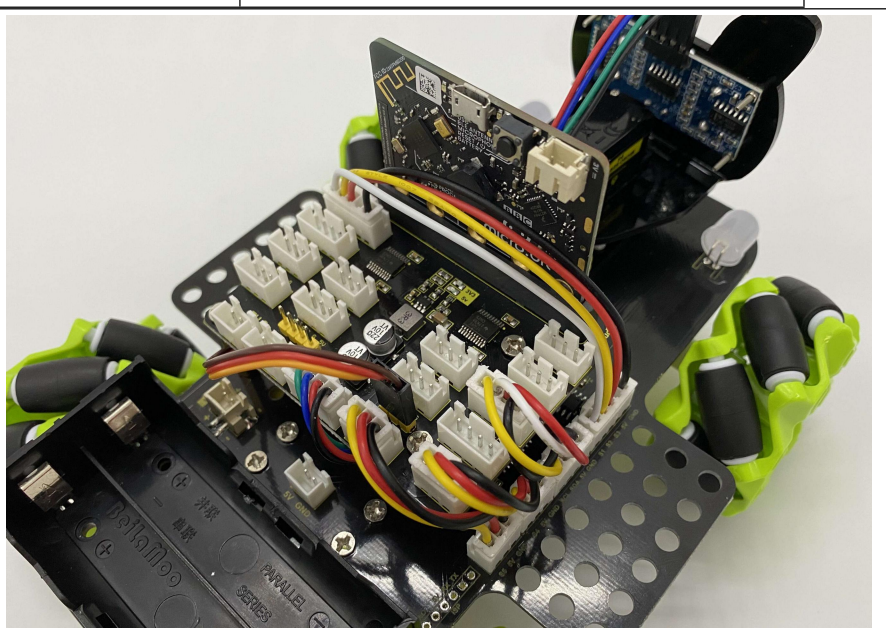
Driver Board	Expansion Board
GND	G
5V	5V
S4	P7

The wiring of controlling the motor and seven-color light

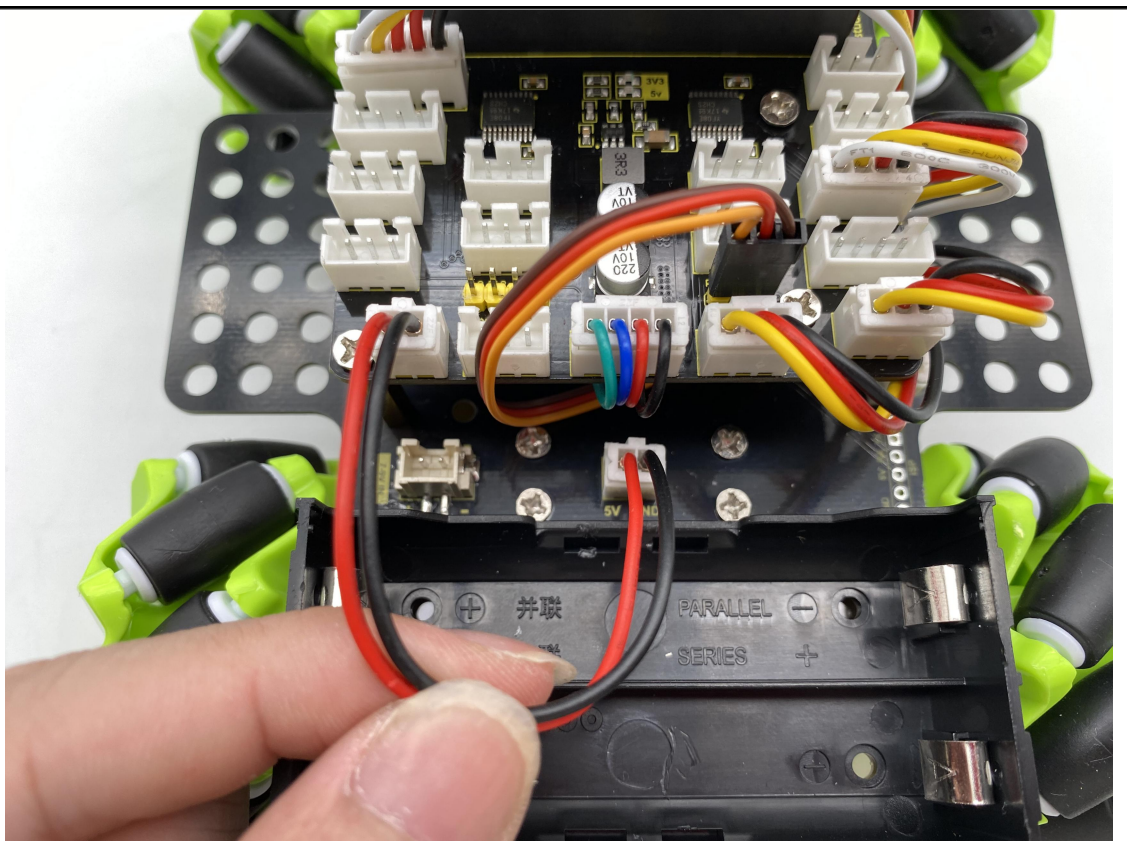
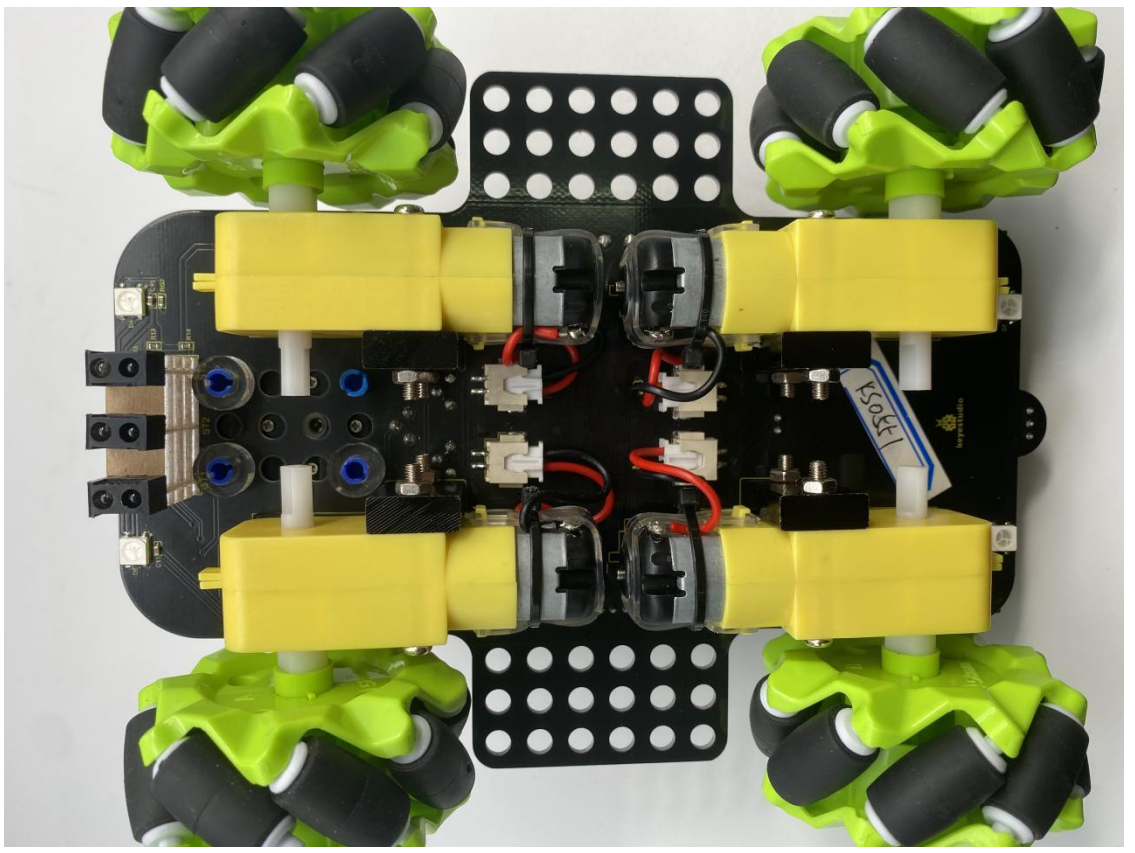
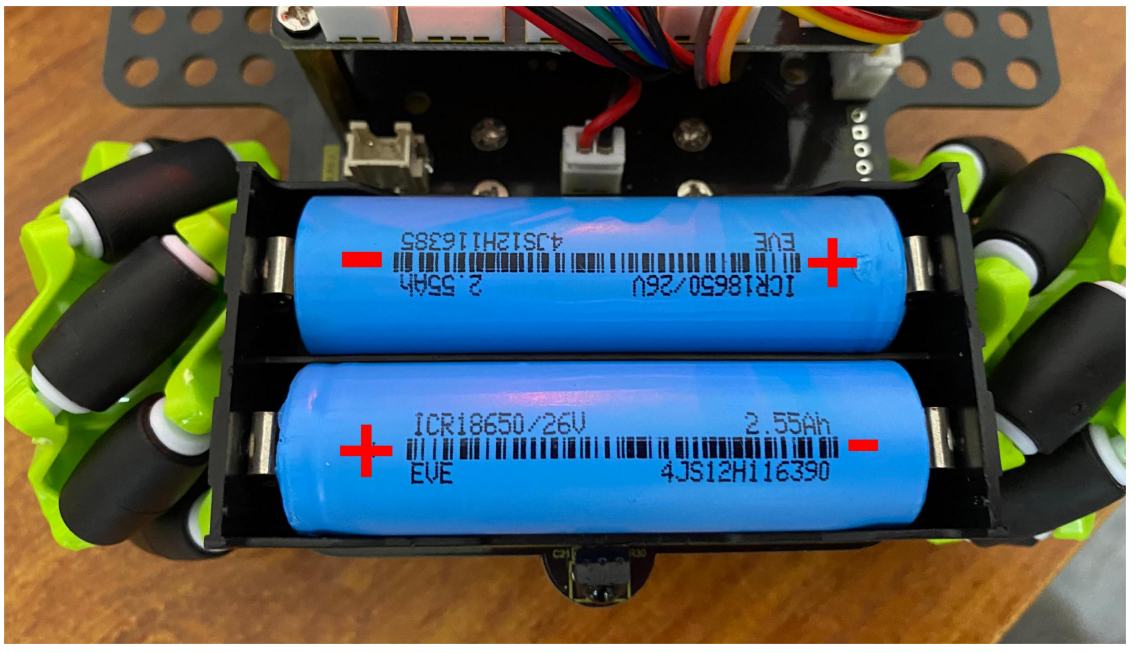


Driver Board	Expansion Board
SCL	P19
SDA	P20
5V	5V
GND	G

The wiring of controlling the 3-channel line-tracking sensor



Driver Board	Expansion Board
S1	P10
S2	P4
S3	P3
GND	G

The wiring of the power supply	
The corresponding interface of the motor	
The installation of the battery	

7.Python

This tutorial is written for Python language. If you want to use graphical code programming, please refer to the manual "Makecode Tutorial". In the root directory of the resource you downloaded, there is a folder named "Python tutorial", which stores all the Python code of the Micro:bit 4WD Mecanum Robot Car V2.0. The Python code file is a file ending with ".py".

What is MicroPython?

Python is a text-based language, which is widely used in education and is also used by professional programmers in fields such as data science and machine learning.

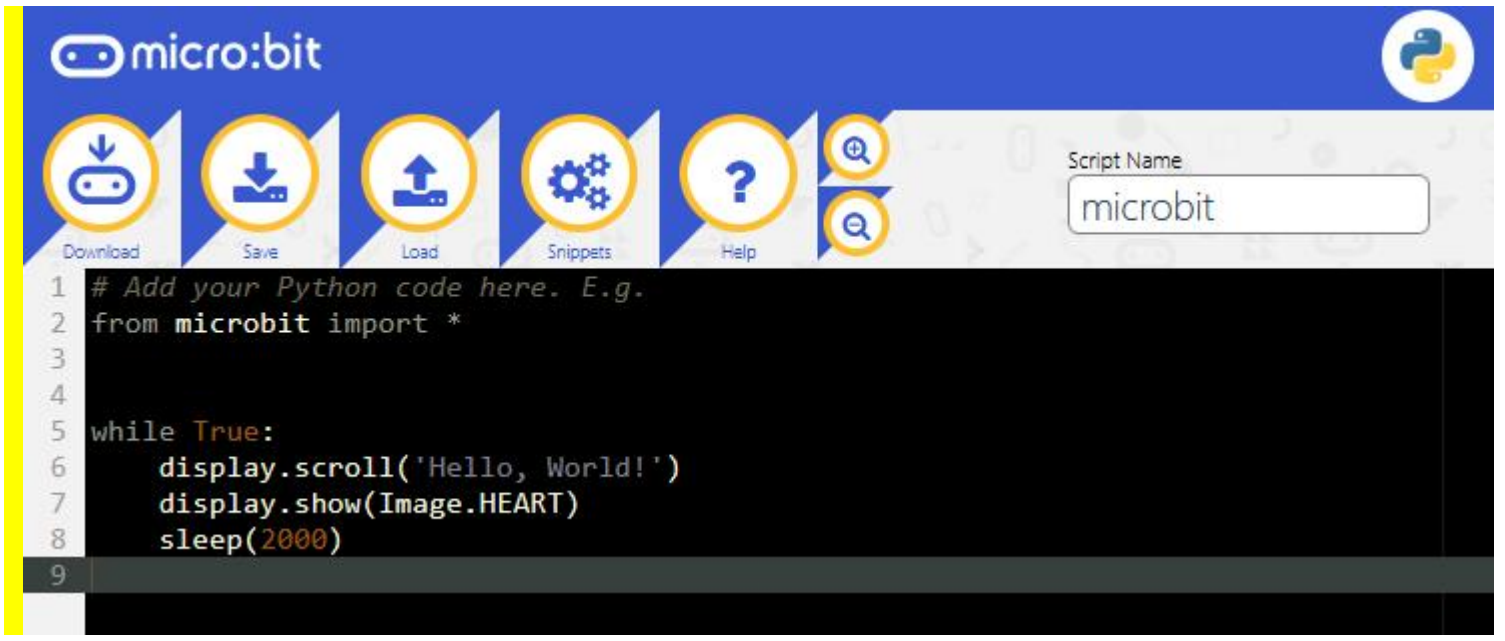
Micro: bit can be programmed in Python, which is a microcontroller, and the hardware differences prevent micro: bit from fully supporting Python. The MicroPython is dedicated to micro：bit，which is an efficient implementation of the Python3 programming language. It contains a small portion of the Python standard library and is optimized to run on micro: bit microcontrollers.

The version of Python used by BBC micro: bit is called MicroPython. The MicroPython is perfect for those who want to learn more about programming, which helps you program with a series of code snippets, a variety of pre-made graphics and music.

Link for BBC microbit MicroPyth：https://microbit-micropython.readthedocs.io/en/latest/tutorials/introduction.html

Python has two types of editors (web version and offline version)

1. Web version: https://python.microbit.org/v/1.1



2. The other one is the offline compiler tool -----Mu 

(Download Mu：https://codewith.mu/en/download)

Mu

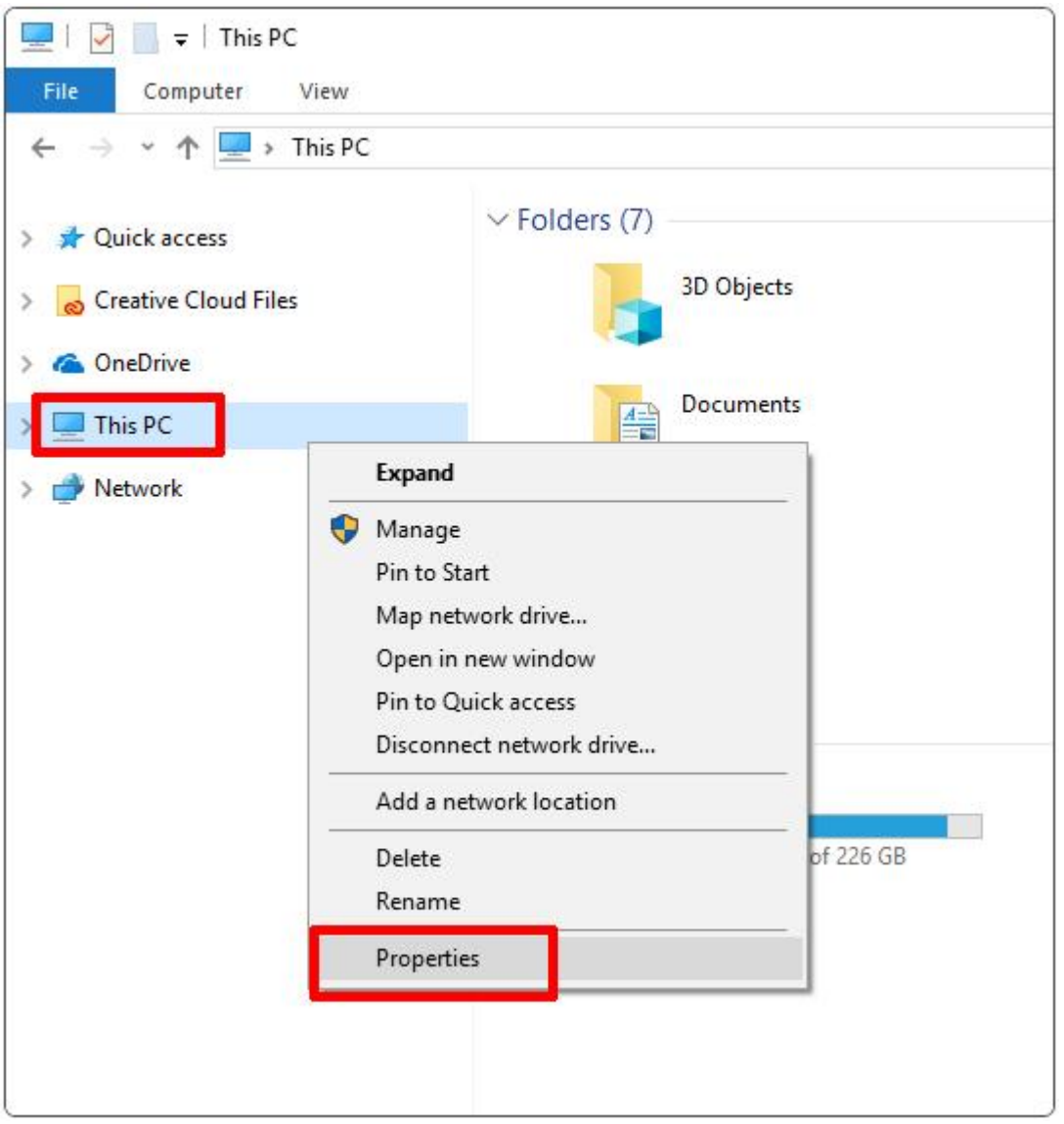
Official Website：https://codewith.mu/

Mu, a Python code editor, is suitable for starters.

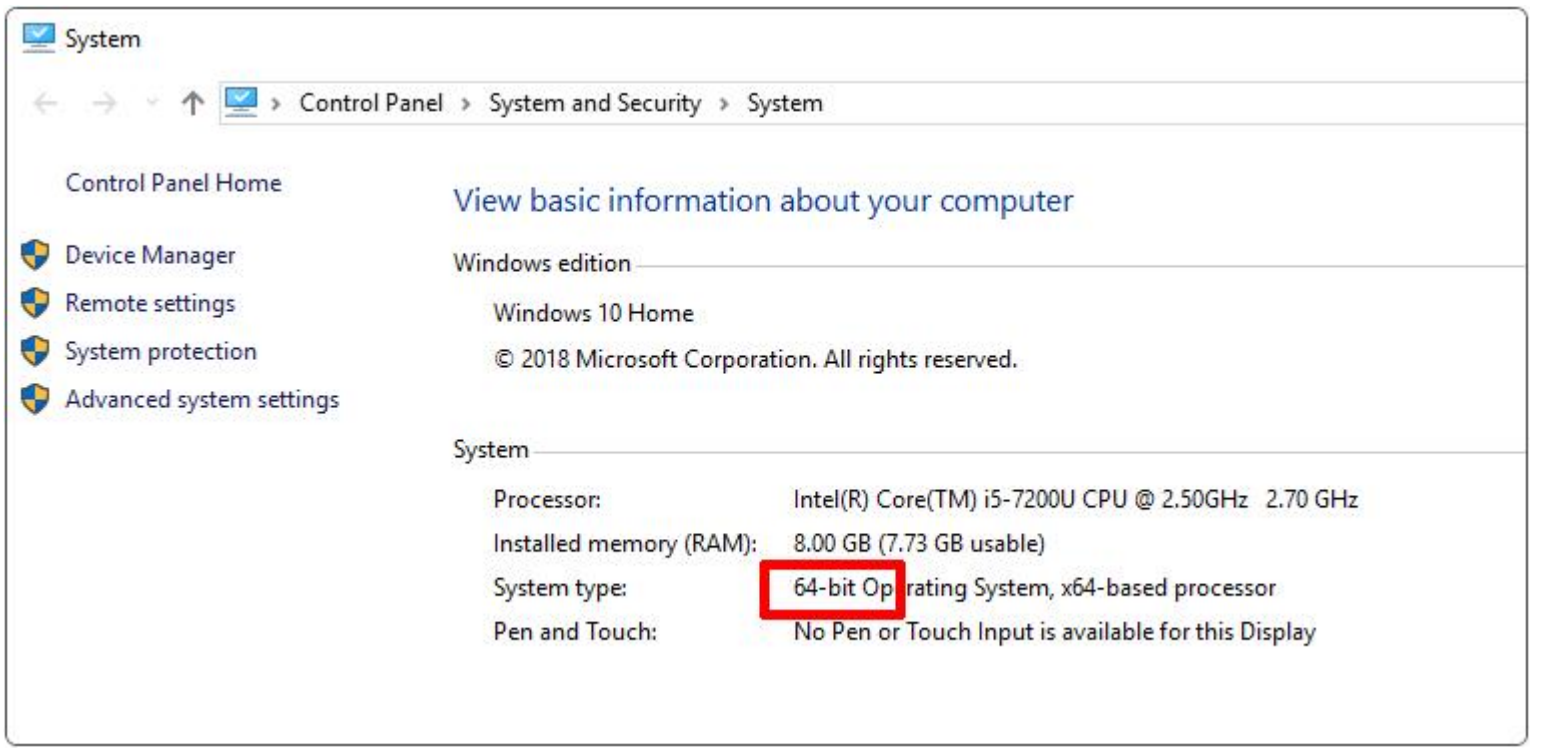
It doesn’ t support 32-bit Windows. The latest version is Mu 1.1.0-beta 2

1. Download Mu

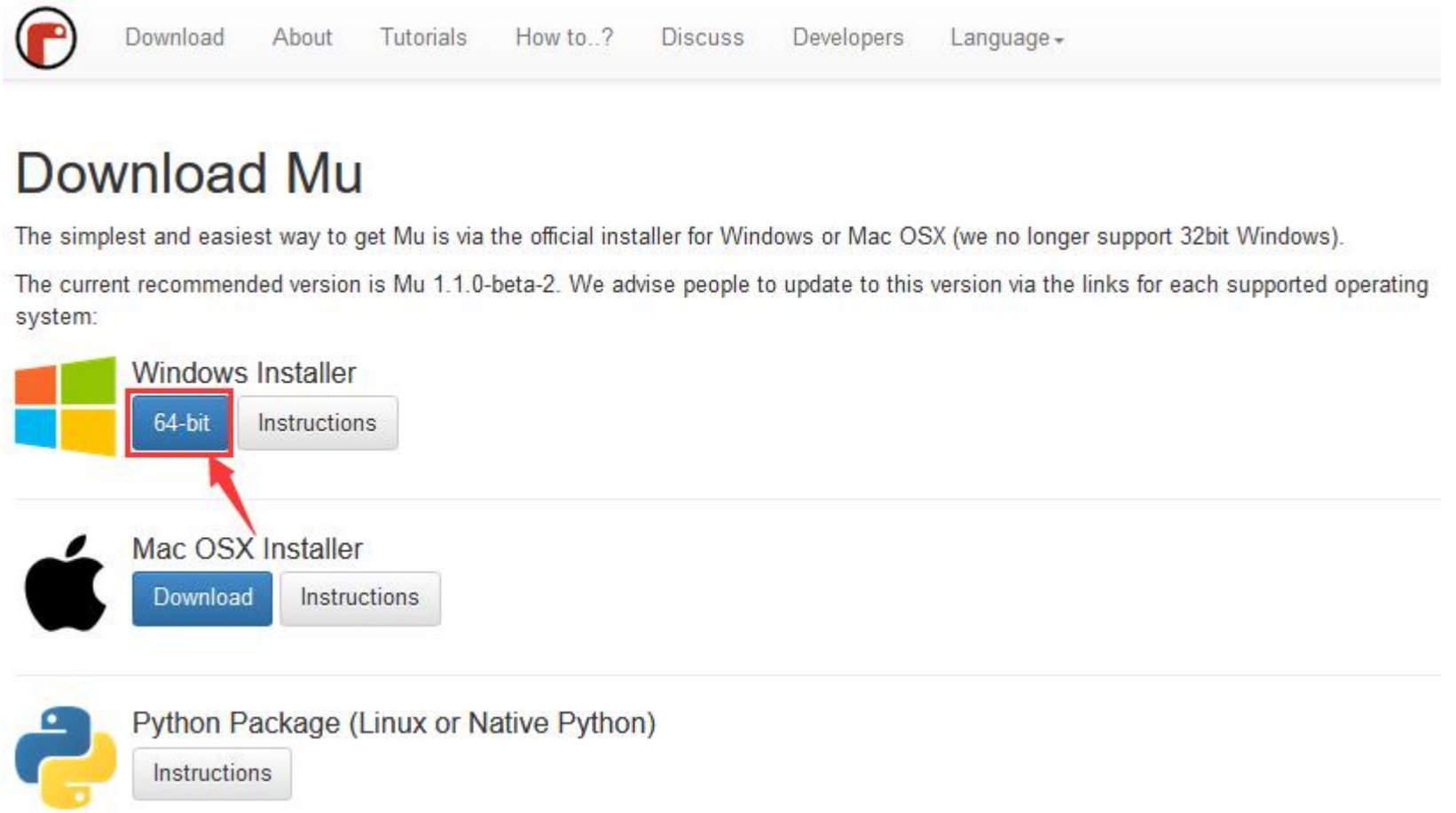
Click “This PC” and right- click to select Properties to check the version of your computer.



Check the system type of your computer.

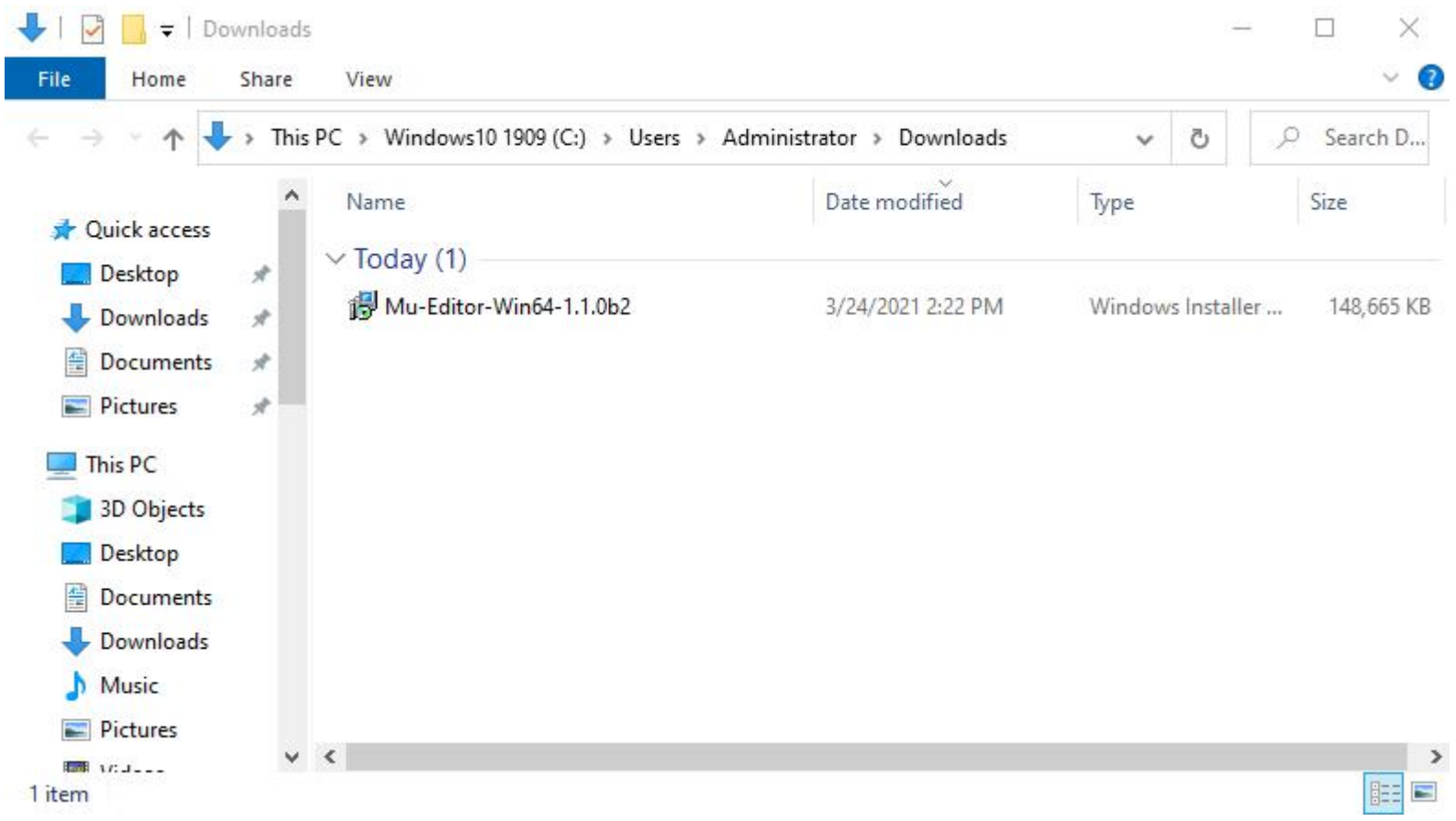


Enter the link: https://codewith.mu/en/download to download the corresponding version of Mu.



2. Run Setup

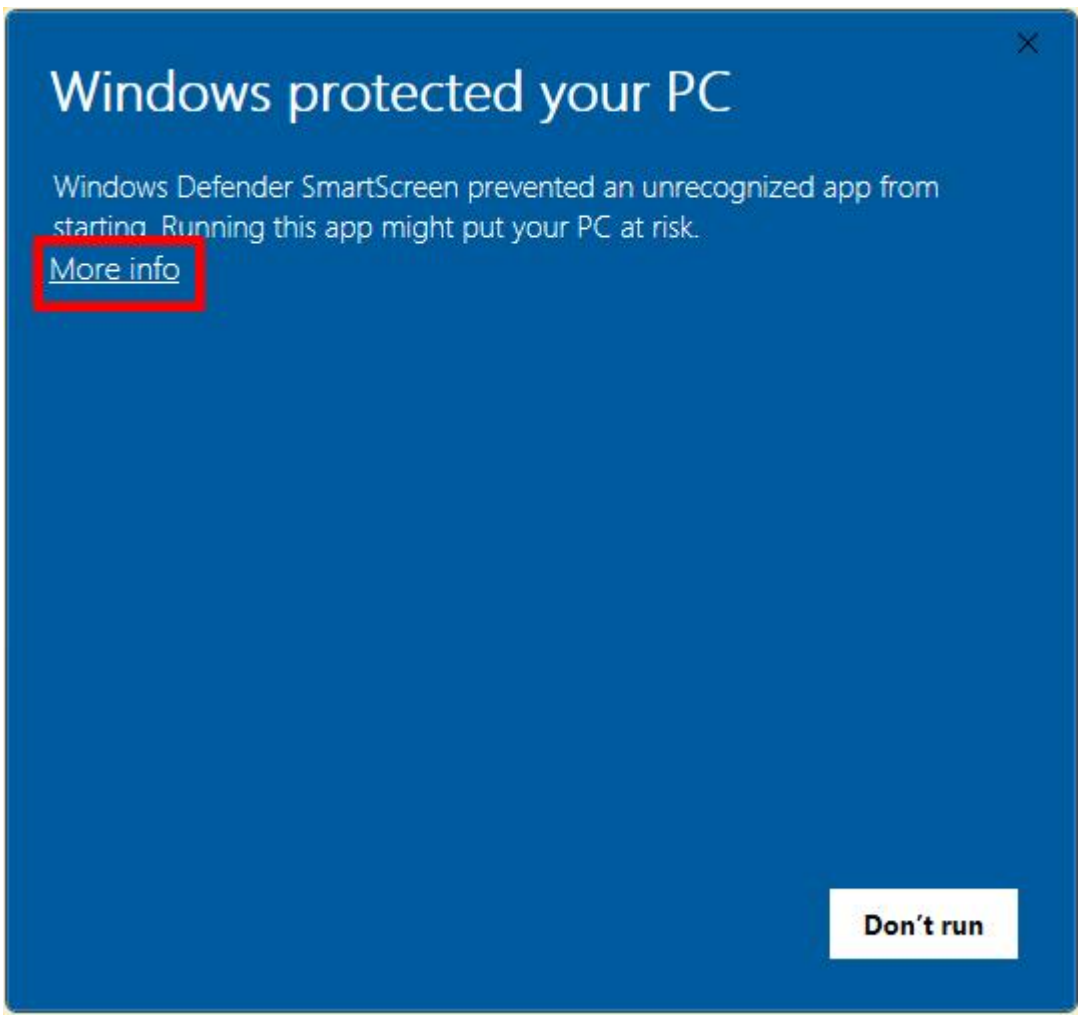
Open the file below



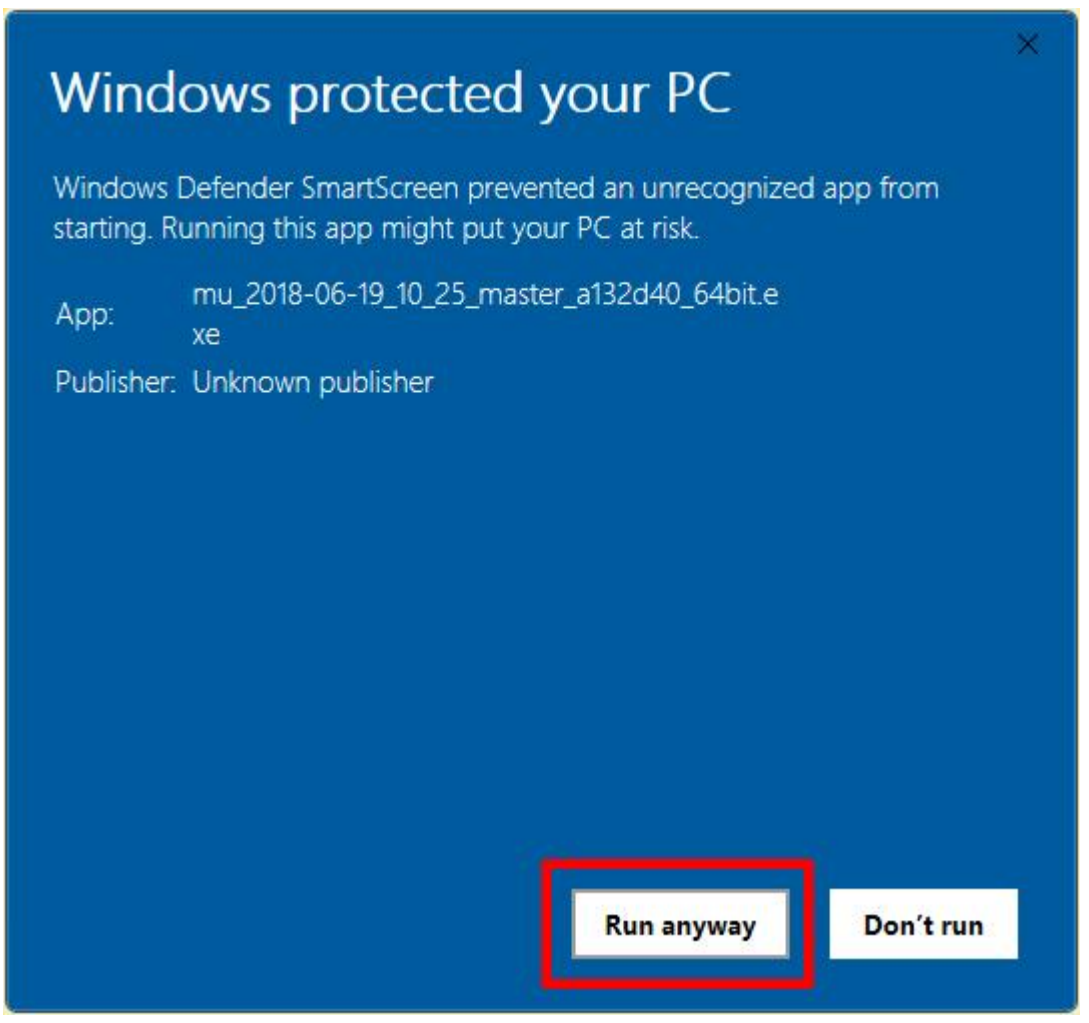
Mac OSX： https://codewith.mu/en/howto/1.1/install_macos.

Windows 10

You will view the page pop-up, then click “More info”.

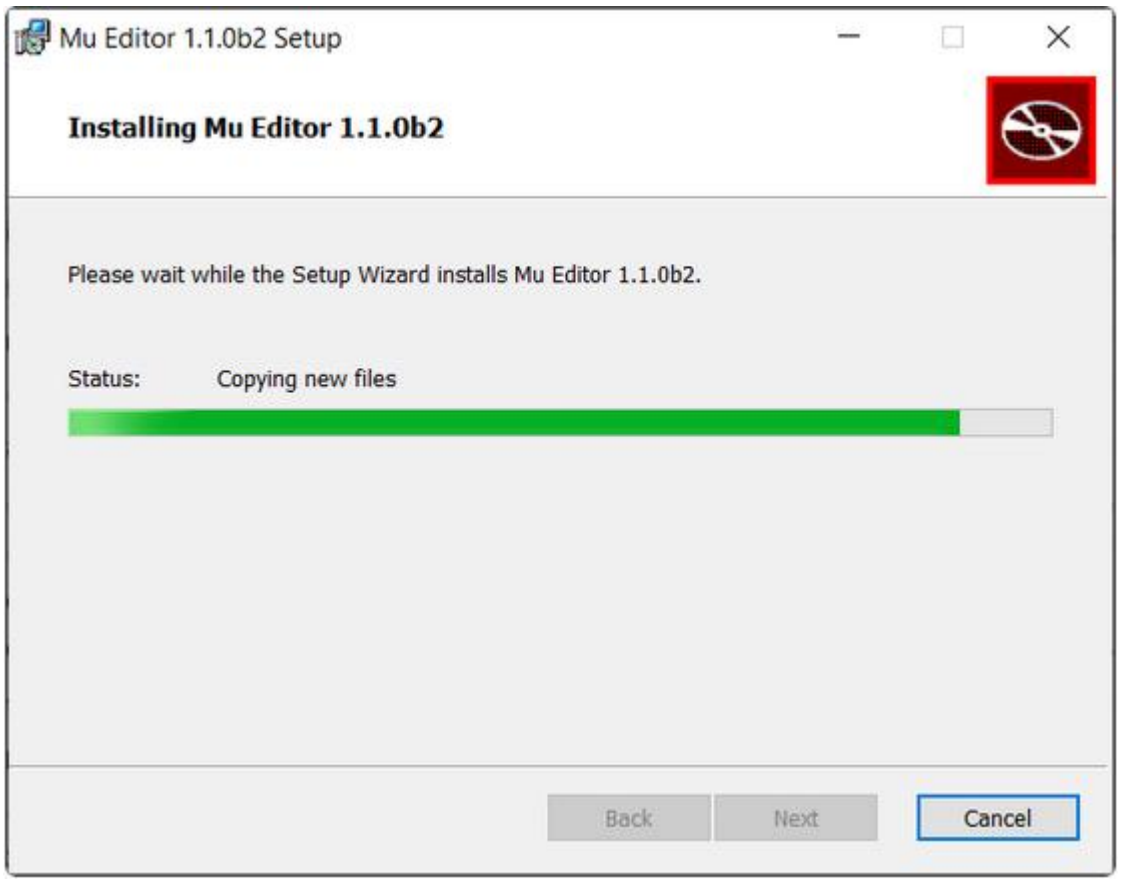


Then click "Run anyway".

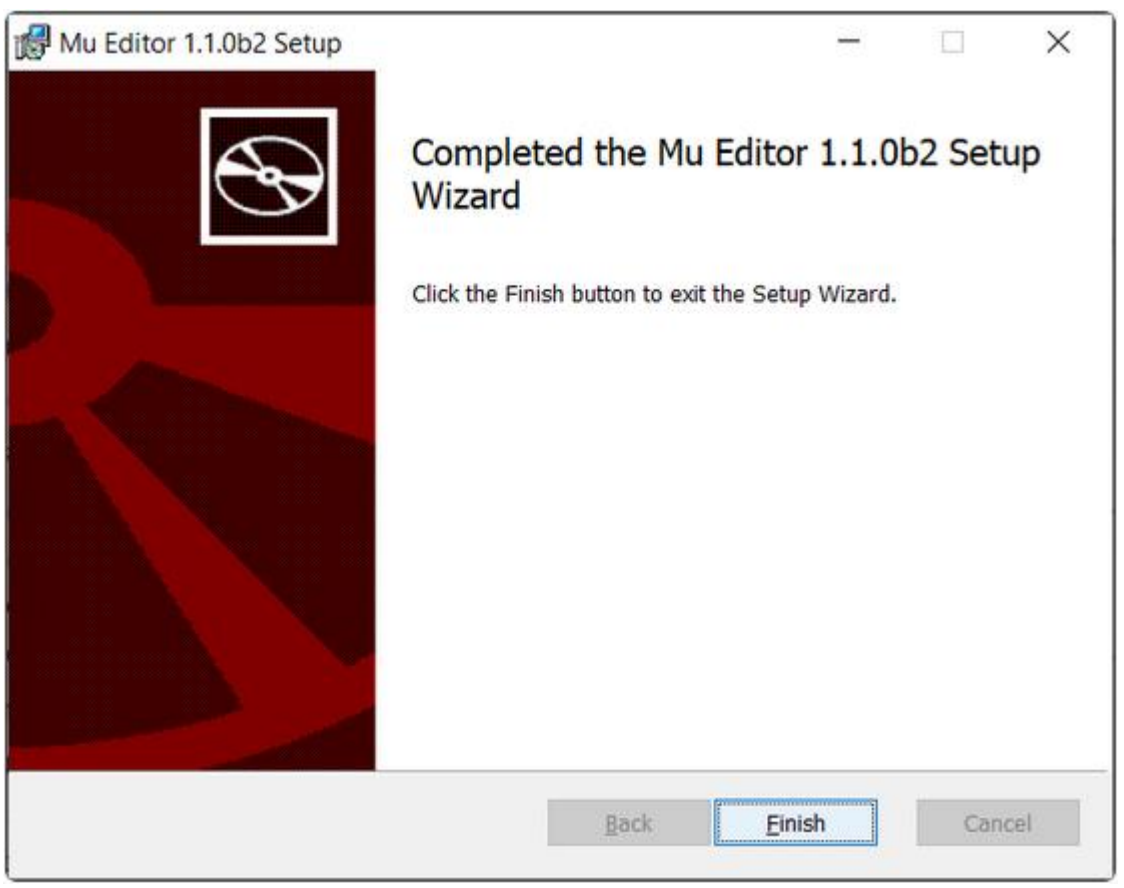


3. License Agreement

Click "Install".

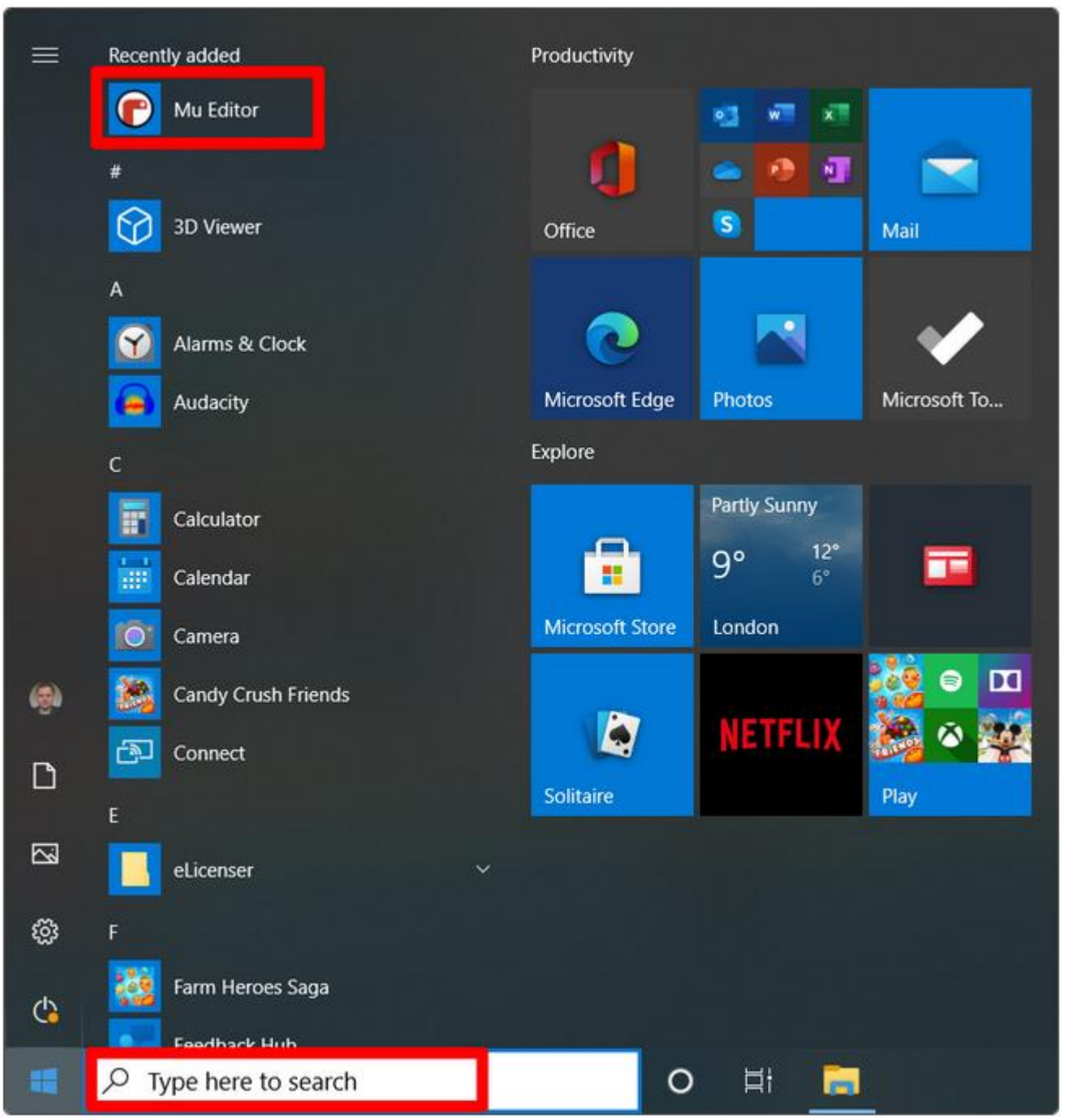


After installed , click “finish” .

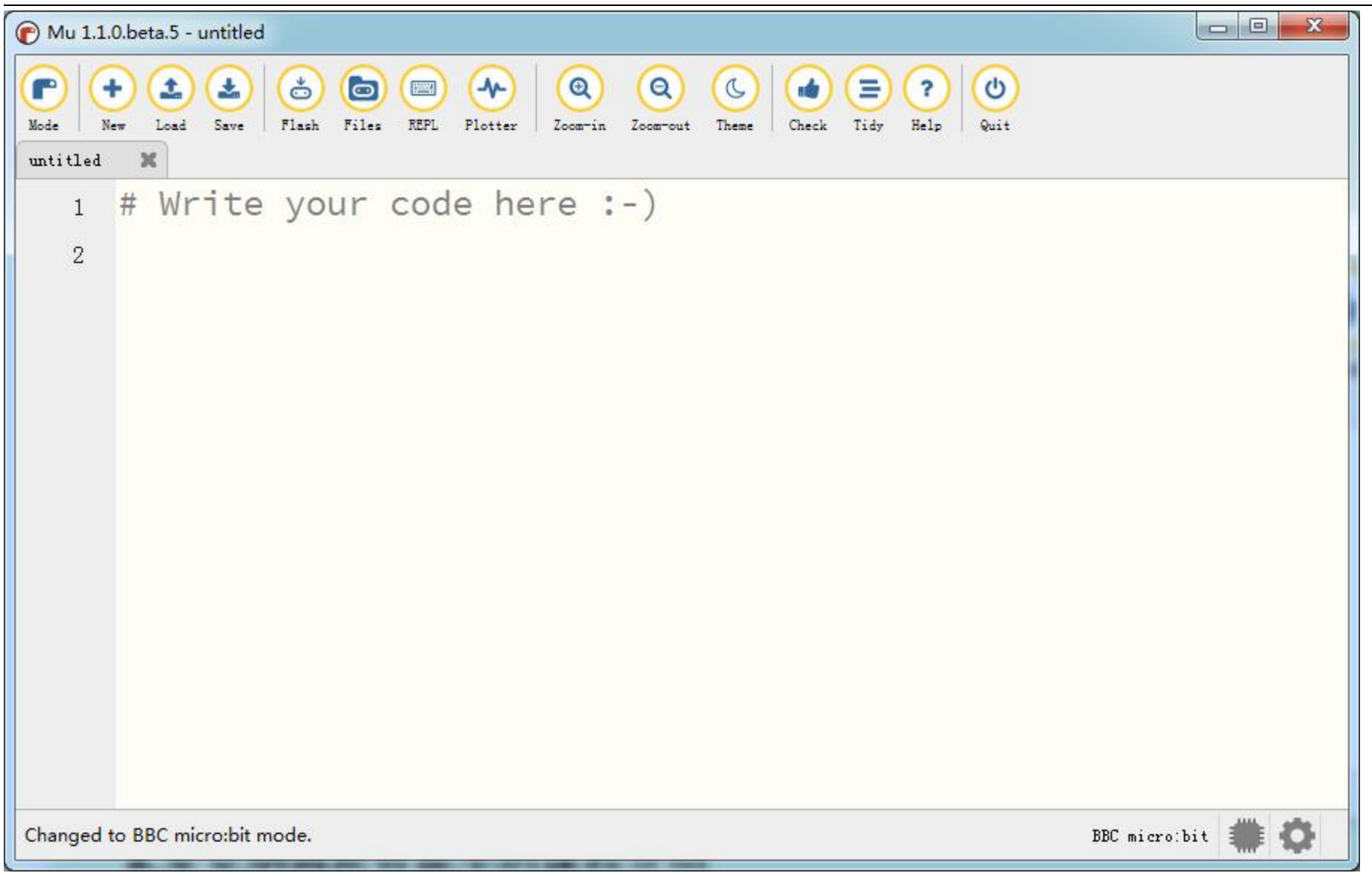


Start Mu

Next, find it according to the following picture



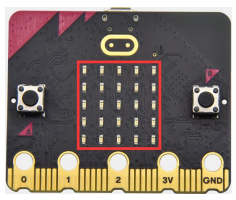
Its main interface is shown as below:



8. Projects

(Note: project 1 to 12 will be conducted with the built-in sensors and LED dot matrix of the Micro:bit main board V2)

Project 1: Heart Beat



1. Description

This project is easy to conduct solely with a micro:bit main board and a micro USB cable. This experiment serves as a starter for your to entry to the magical programming world of the micro:bit.

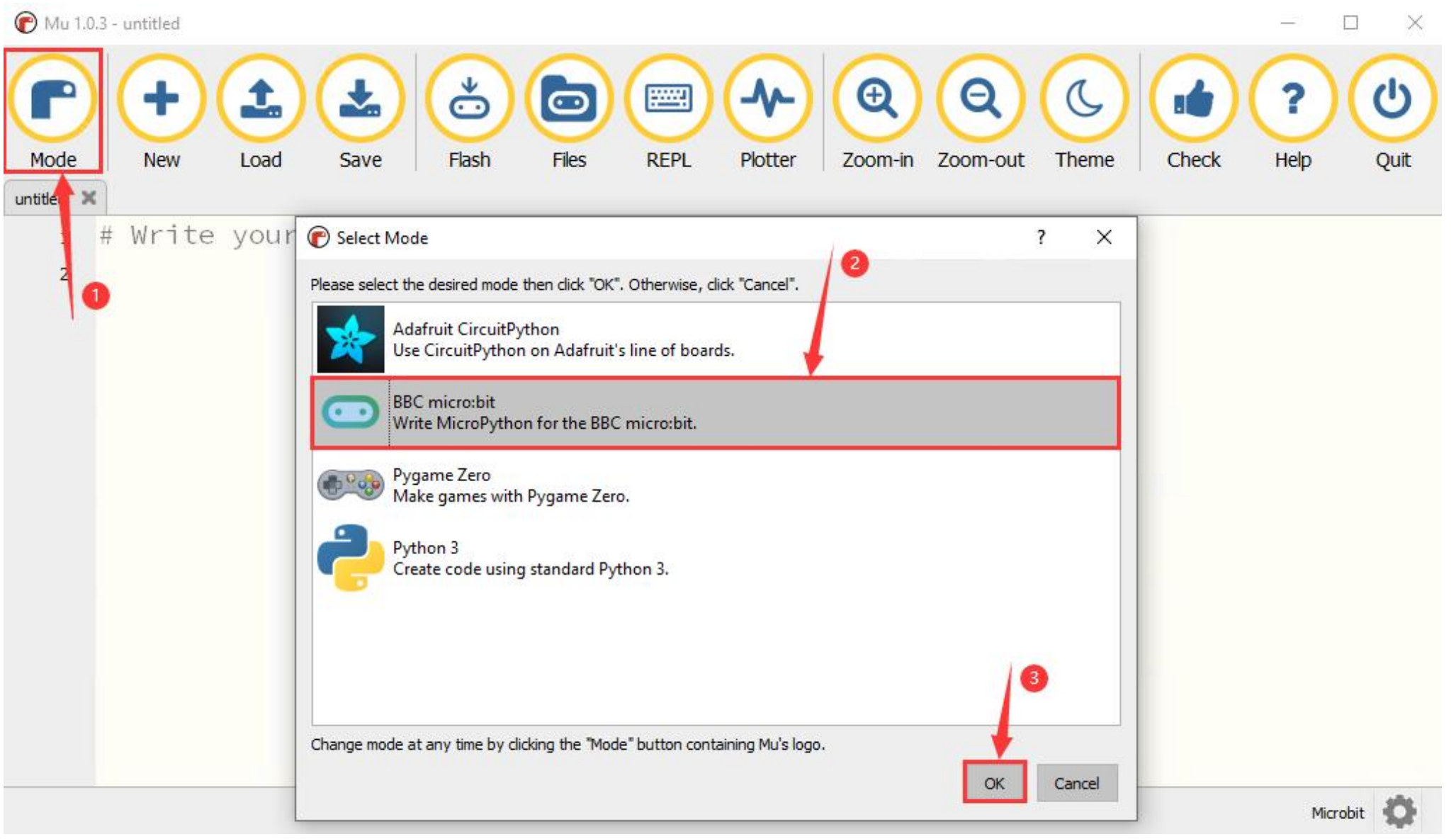
2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

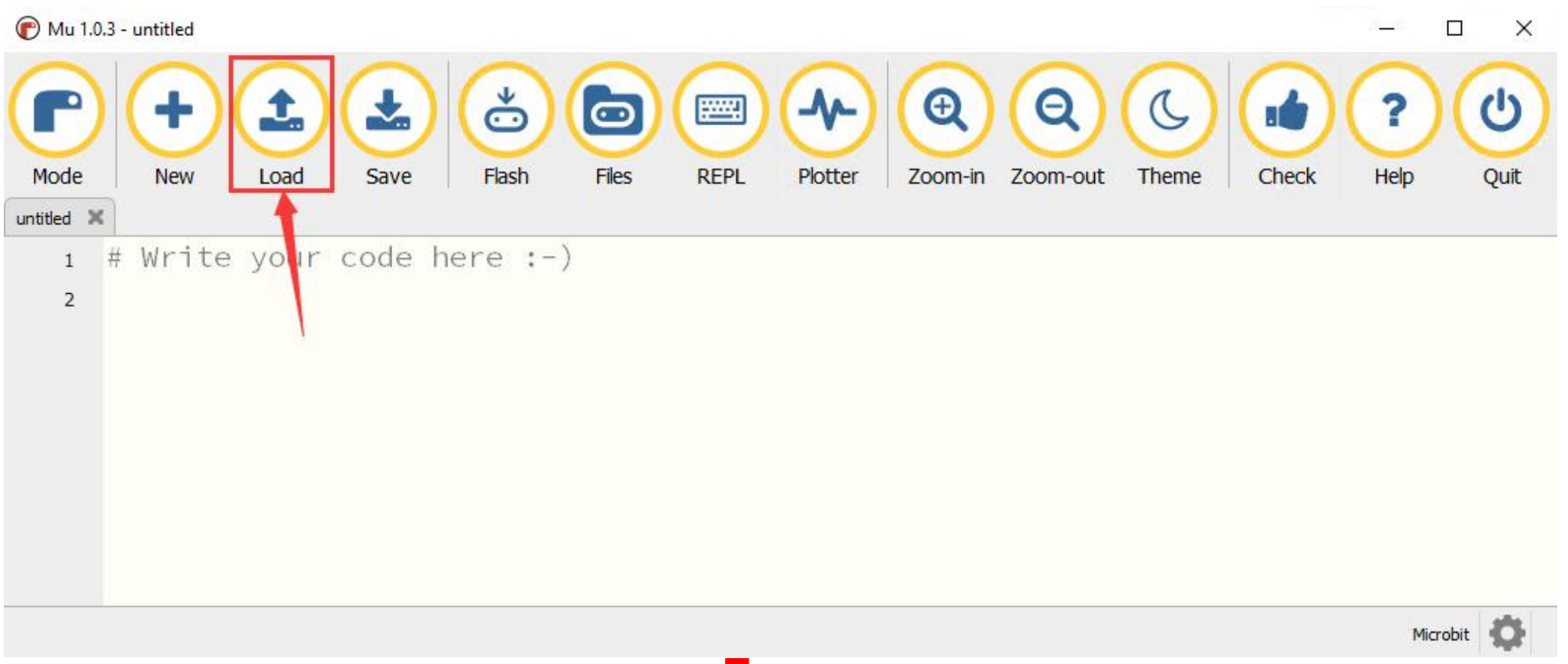
3. Test Code

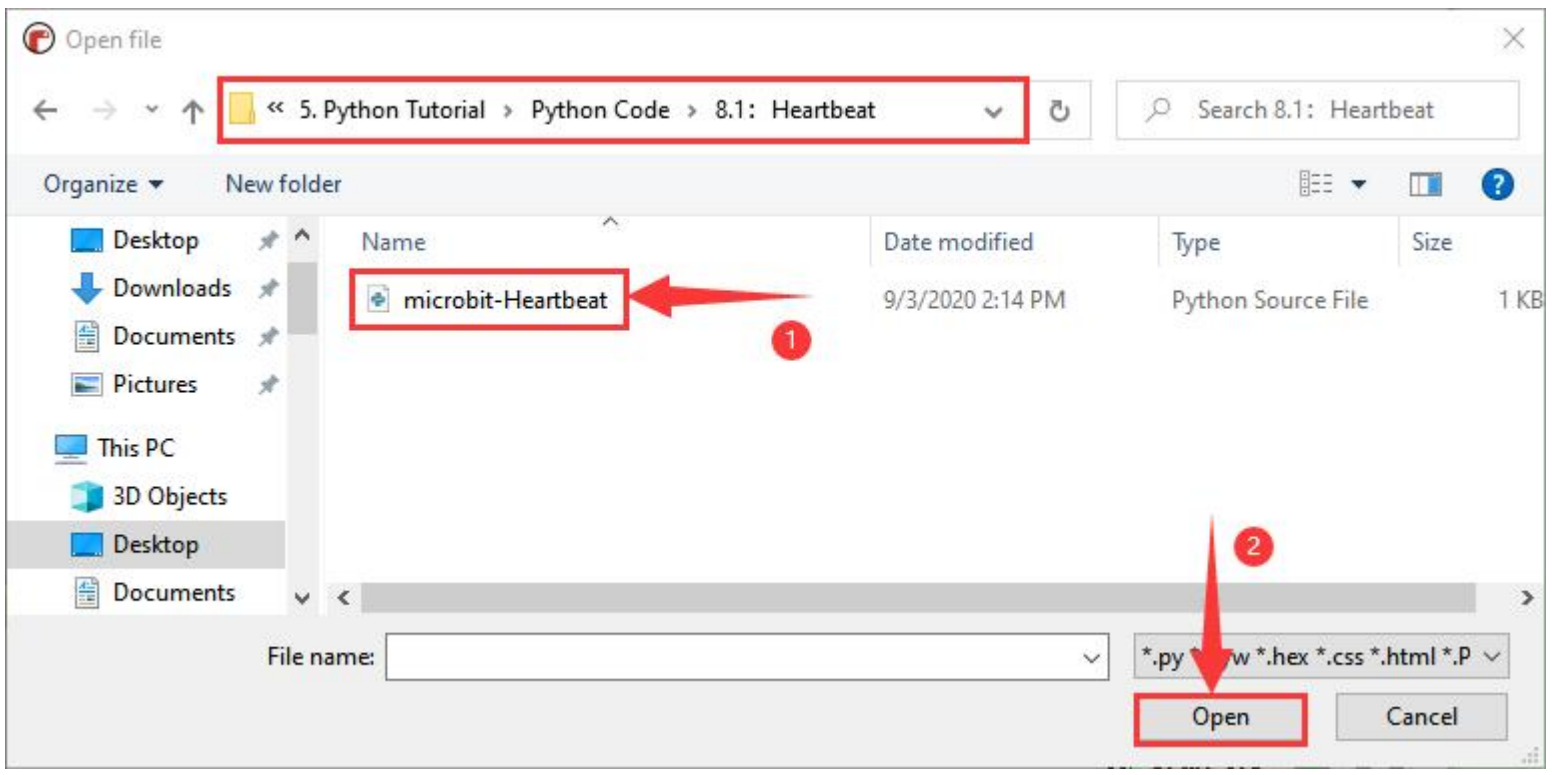
Open the Mu software, click Mode, then click "BBC micro：bit" and "OK"：



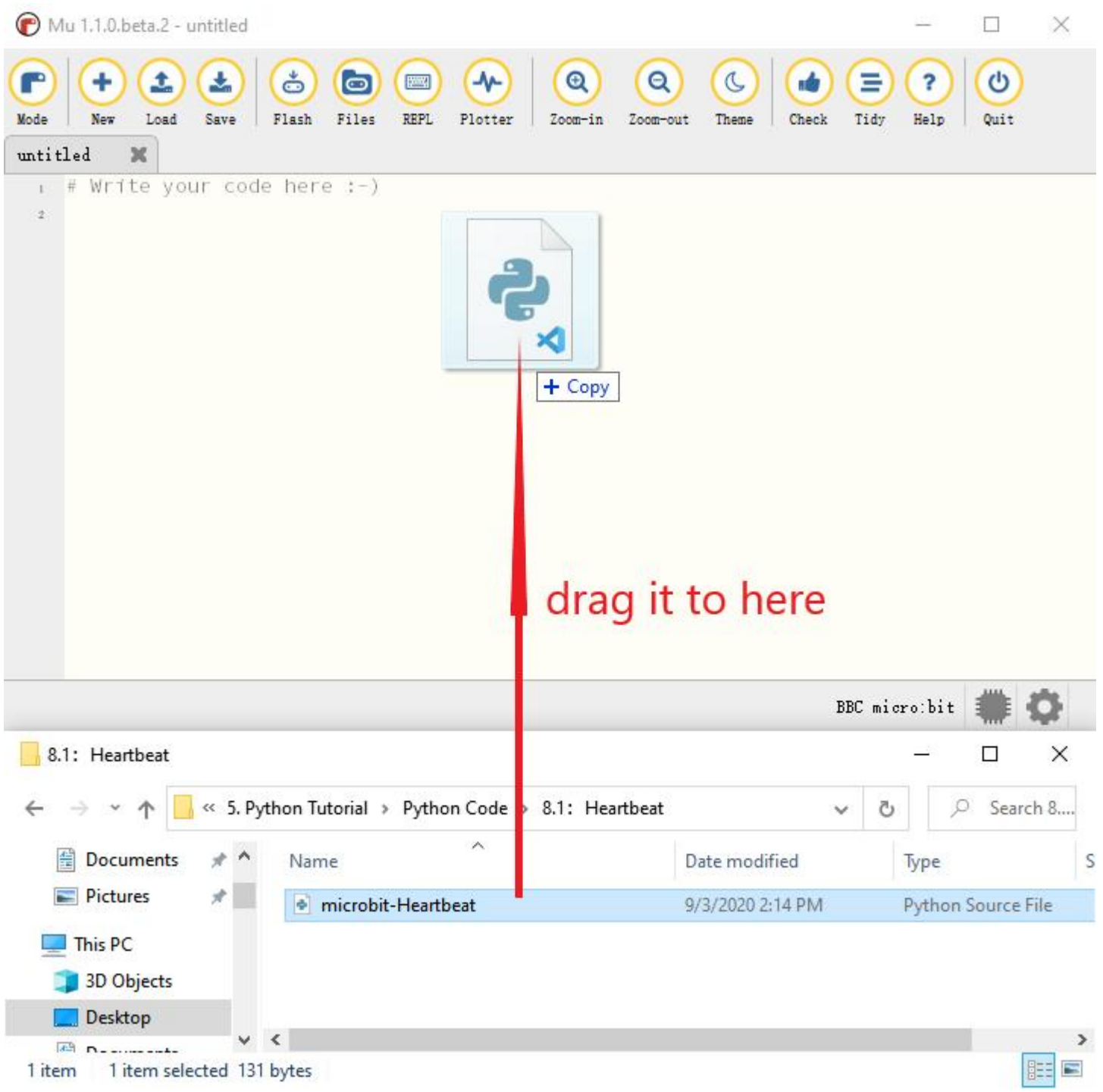
Tap “Load”, select “ “microbit-Heartbeat.py “ file and click “open”:

File Type	Route	File Name
Python file	../Python Code/8.1：Heart beat	microbit-Heartbeat.py



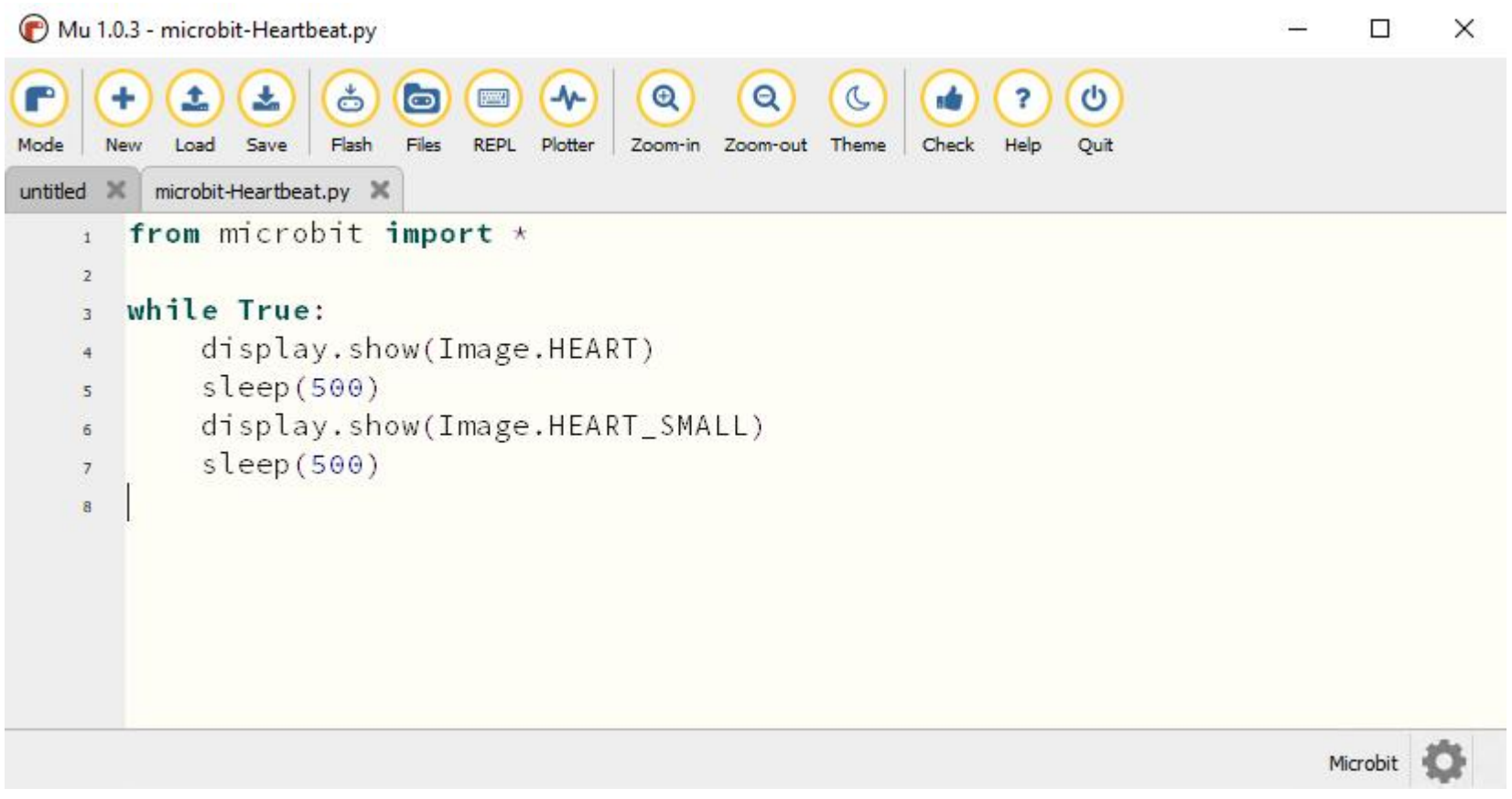


There is another way to import code. Open the Mu software and drag file" microbit-Heartbeat.py" into it.



You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

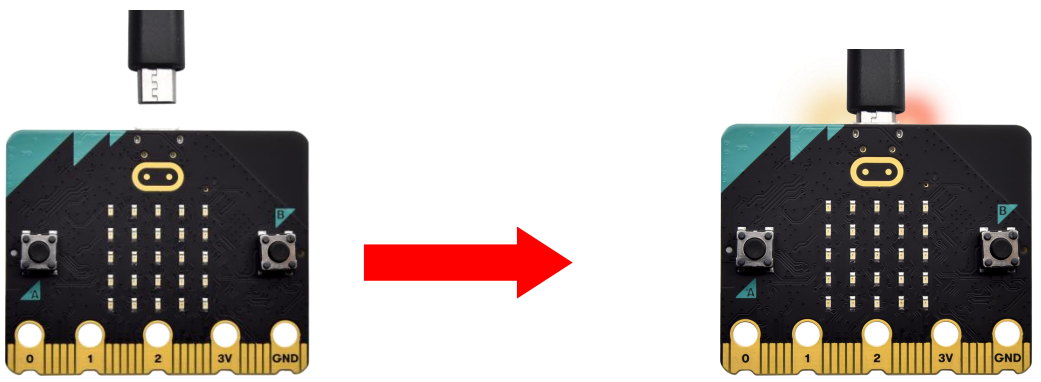


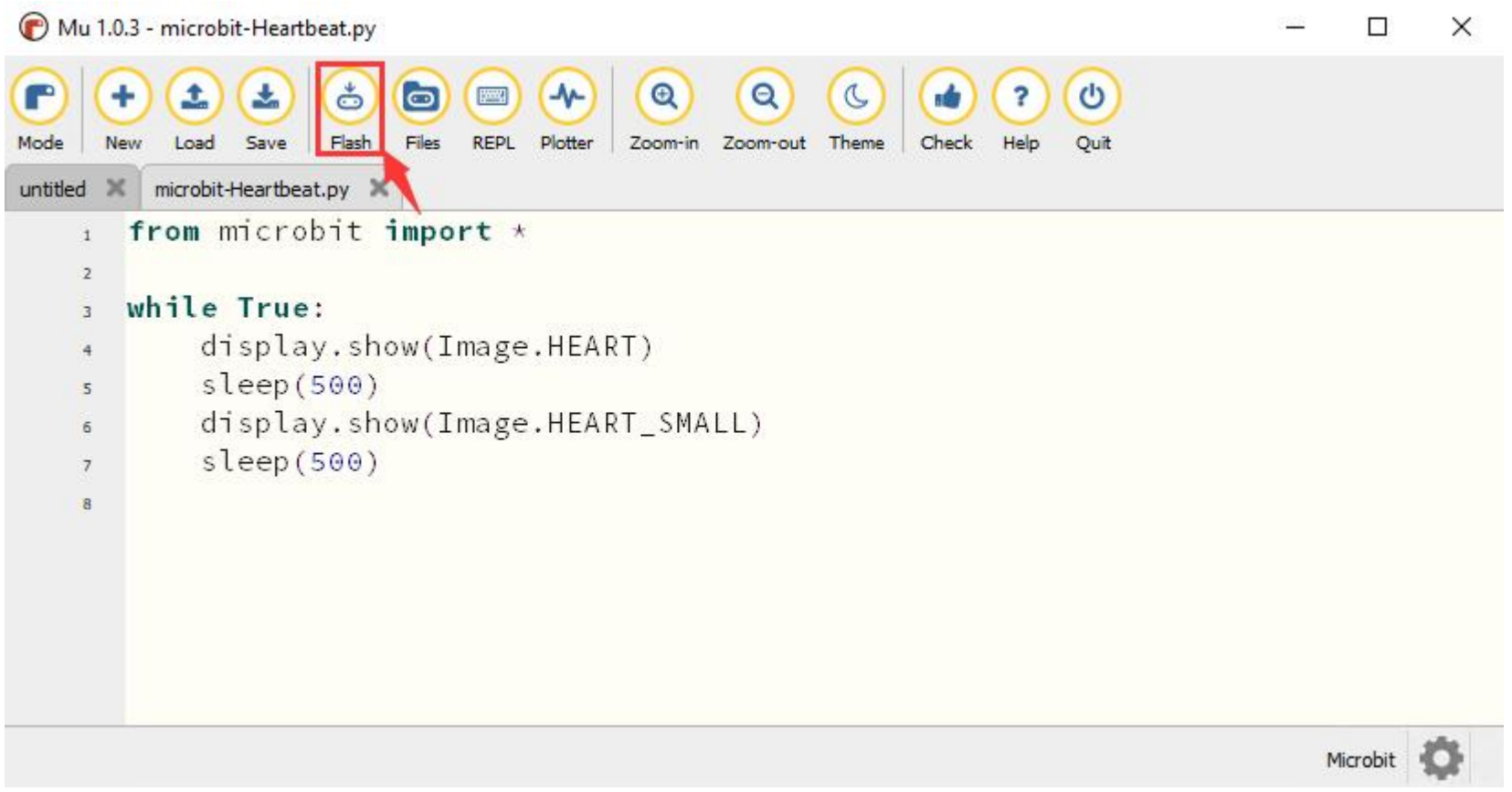
The following are a list of built-in images:

- Image.HEART
- Image.HEART_SMALL
- Image.HAPPY
- Image.SMILE
- Image.SAD
- Image.CONFUSED
- Image.ANGRY
- Image.ASLEEP
- Image.SURPRISED
- Image.SILLY
- Image.FABULOUS
- Image.MEH
- Image.YES
- Image.NO
- Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9, Image.CLOCK8, Image.CLOCK7, Image.CLOCK6, Image.CLOCK5,
  Image.CLOCK4, Image.CLOCK3, Image.CLOCK2,Image.CLOCK1
- Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW
- Image.TRIANGLE
- Image.TRIANGLE_LEFT
- Image.CHESSBOARD
- Image.DIAMOND
- Image.DIAMOND_SMALL
- Image.SQUARE
- Image.SQUARE_SMALL
- Image.RABBIT
- Image.COW
- Image.MUSIC_CROTCHET
- Image.MUSIC_QUAVER
- Image.MUSIC_QUAVERS
- Image.PITCHFORK

- Image.PACMAN
- Image.TARGET
- Image.TSHIRT
- Image.ROLLERSKATE
- Image.DUCK
- Image.HOUSE
- Image.TORTOISE
- Image.BUTTERFLY
- Image.STICKFIGURE
- Image.GHOST
- Image.SWORD
- Image.GIRAFFE
- Image.SKULL
- Image.UMBRELLA
- Image.SNAKE，Image.ALL_CLOCKS，Image.ALL_ARROWS

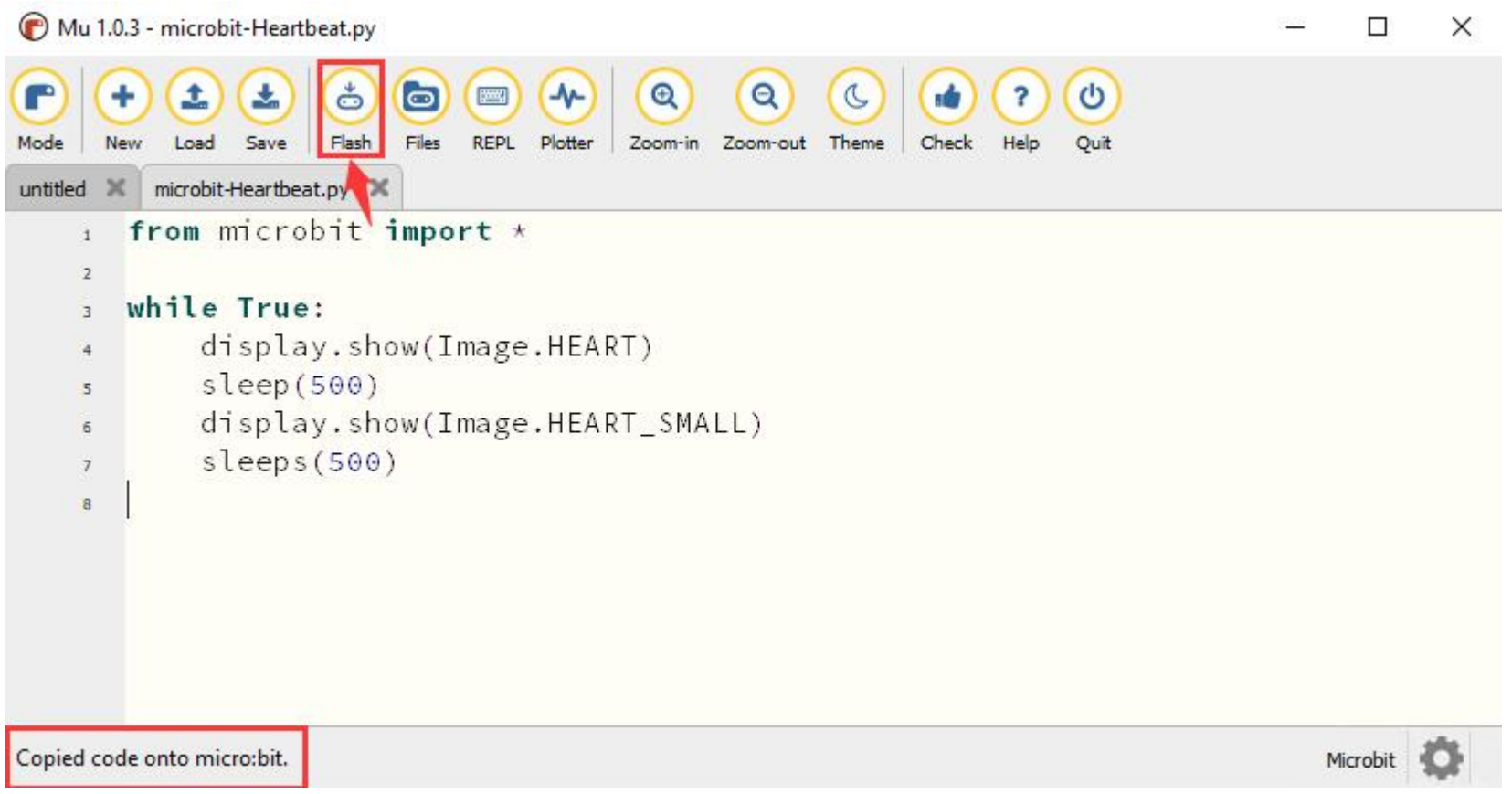
Connect the micro:bit board to computer via an USB cable, click “Flash” to download code to the board.



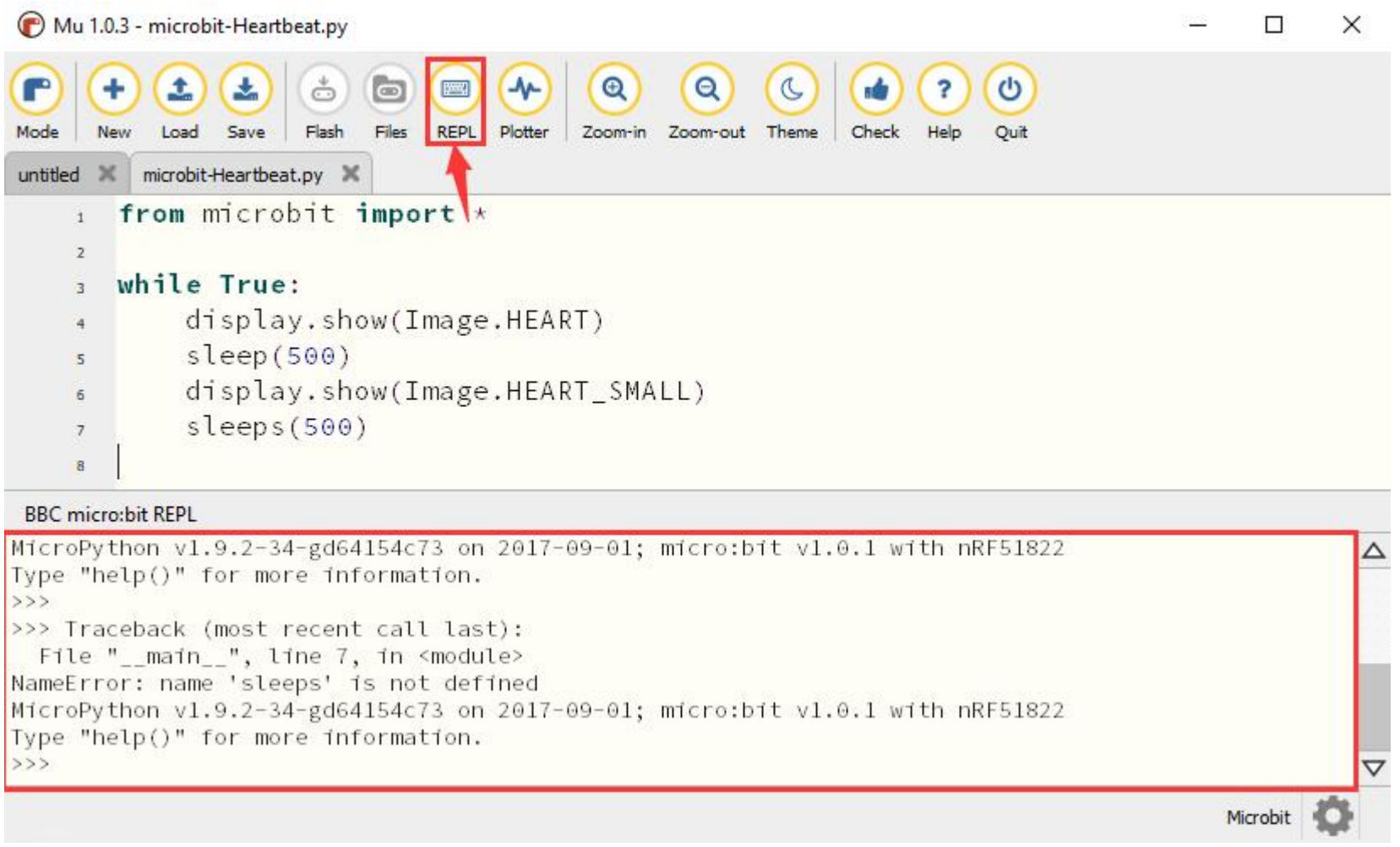


The code, even it is wrong, can be downloaded to the micro:bit board successfully, but can not work on micro:bit board.

Click "Flash" to download code to micro:bit.

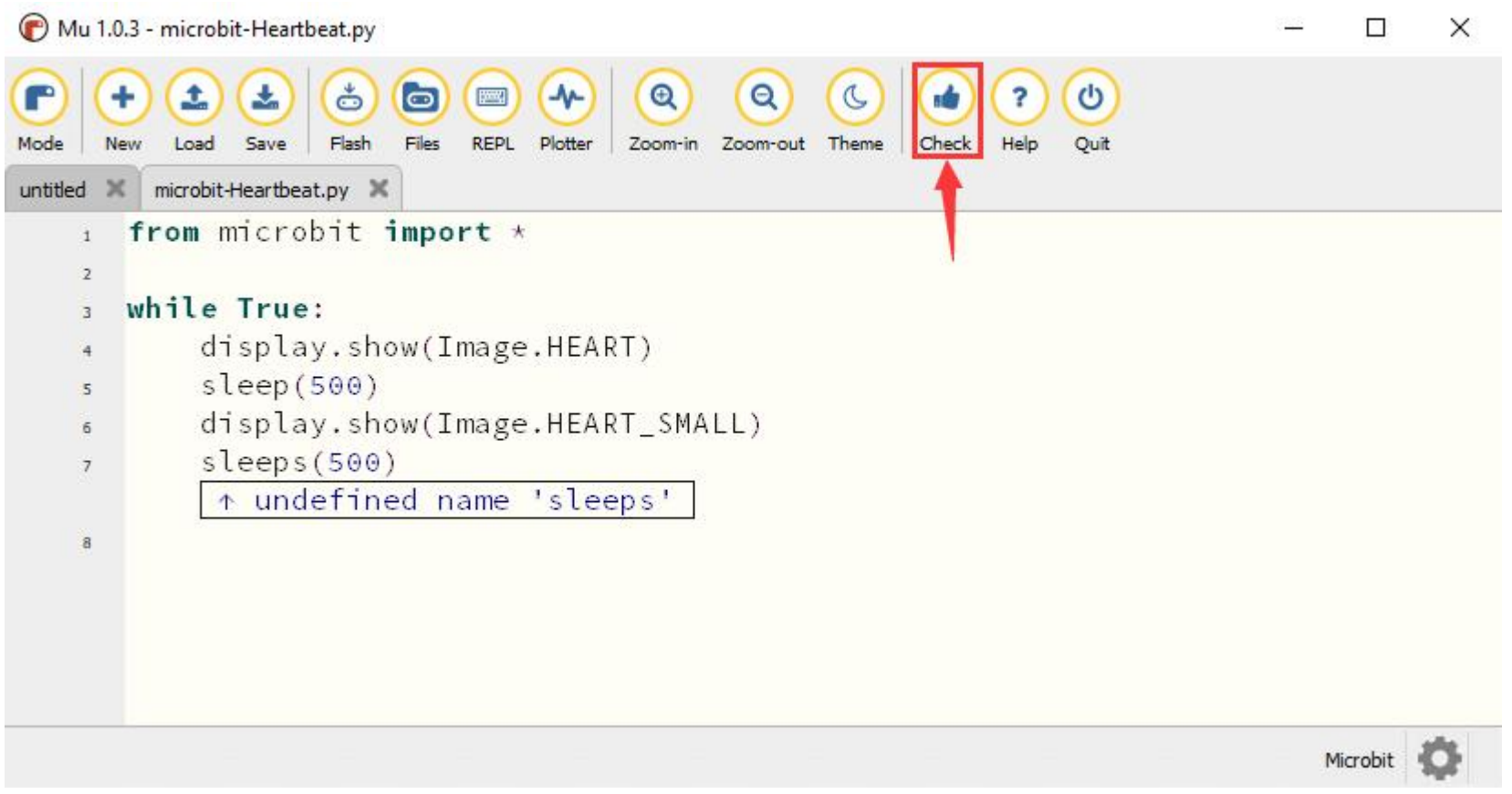


Click "REPL" and press the reset button on micro:bit, the error information will be displayed on the REPL window, as shown below:

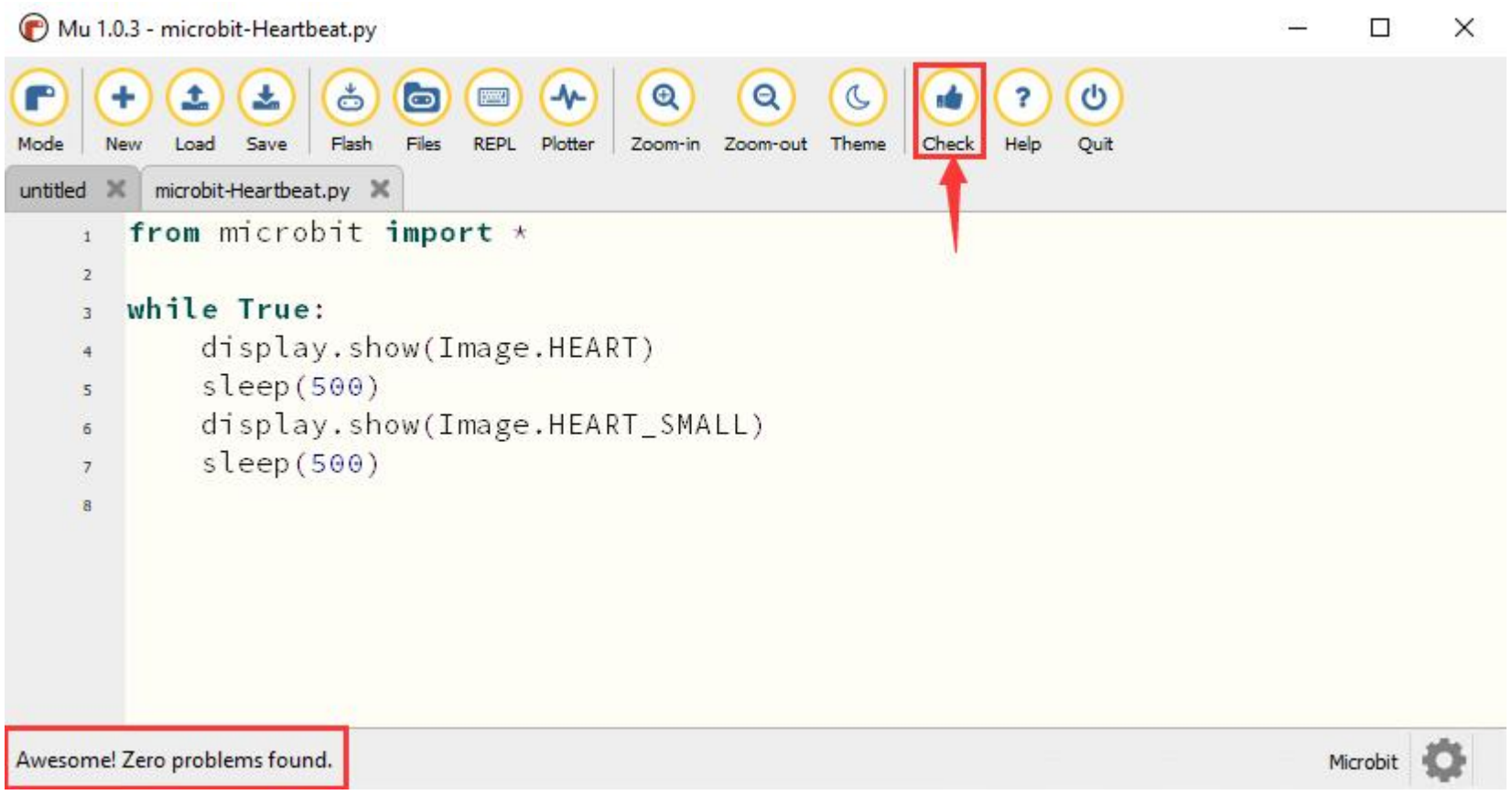


Click “REPL” again to turn off the REPL mode, then you could refresh new code.

To make sure the correct code, you only need to tap “Check”. The errors will be shown on the window.



Modify the code according to the prompt and click “Check”.



Please log in the website for more tutorials：https://codewith.mu/en/tutorials/

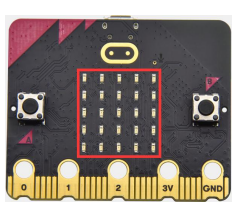
4. Test Result

After uploading the test code successfully, clicking "Flash" button and keeping the connection with the computer to power the main board, then the LED dot matrix shows the pattern "❤" and then "▪" alternatively.

5. Code Explanation

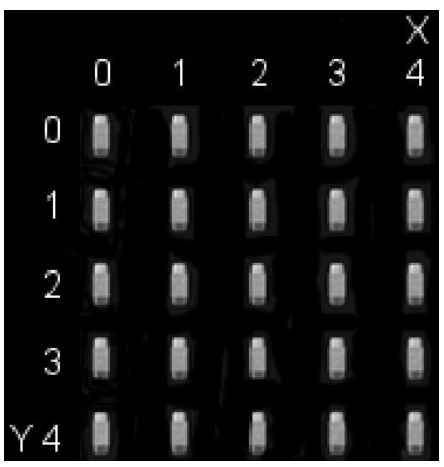
from microbit **import** *	Import the library file of micro：bit
while True:	This is a permanent loop that makes micro:bit execute the code in this loop forever. .
display.show(Image.HEART)	micro：bit shows "❤"
sleep(500)	Delay in 500ms
display.show(Image.HEART_SMALL)	The LED dot matrix displays "▪"

Project 2：Light A Single LED



1. Description

The LED dot matrix consists of 25 Diodes arranged in a 5 by 5 square and placed at the intersection of row lines (X) and column lines (Y). We can control one of the 25 LEDs by setting coordinate points. For example, the first LED sits in the first line is (0,0) and the third LED positioned in the first line is (2,0) and others likewise.



2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

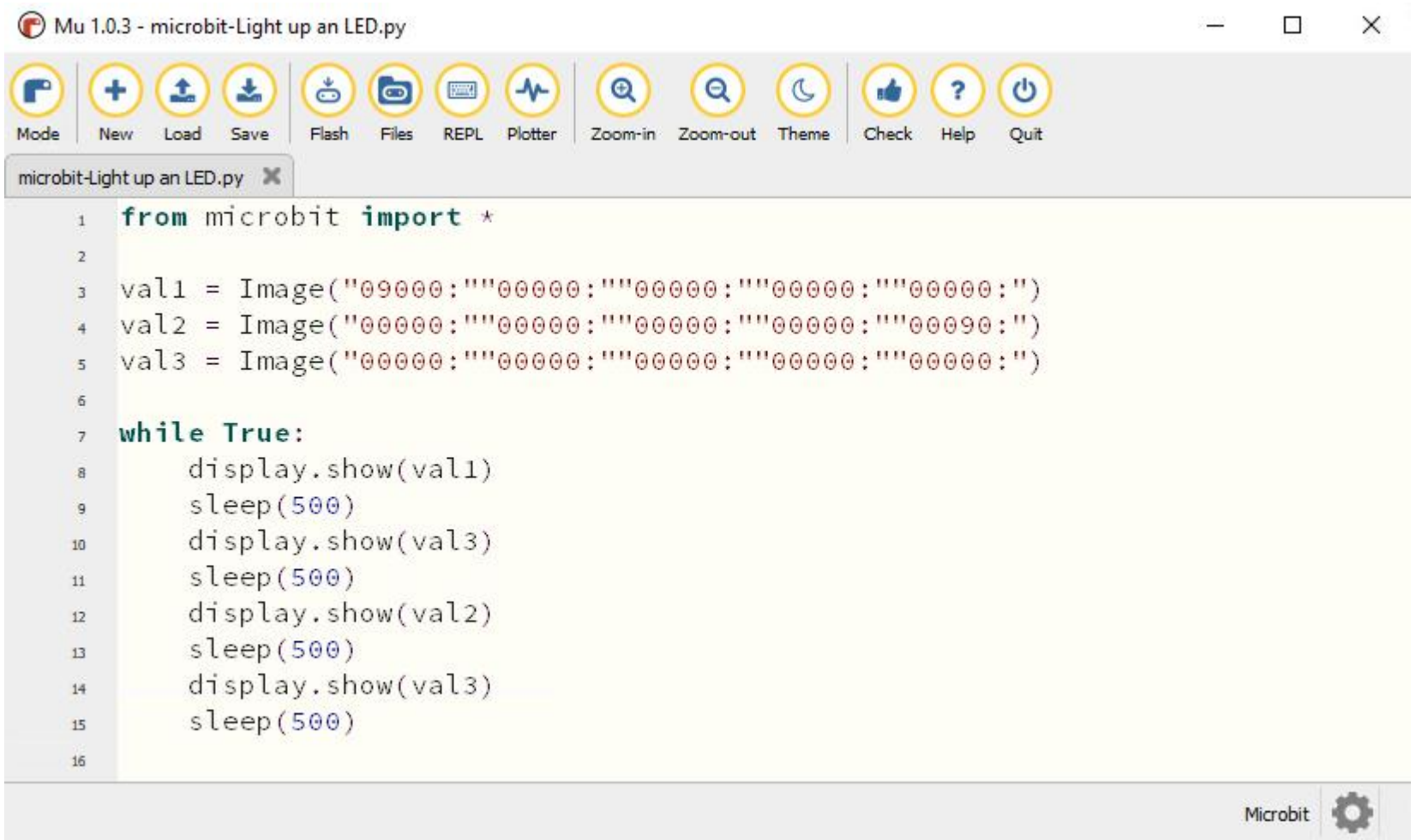
3. Test Code

Enter the Mu software and open the “Single LED display.py.” file to import code:

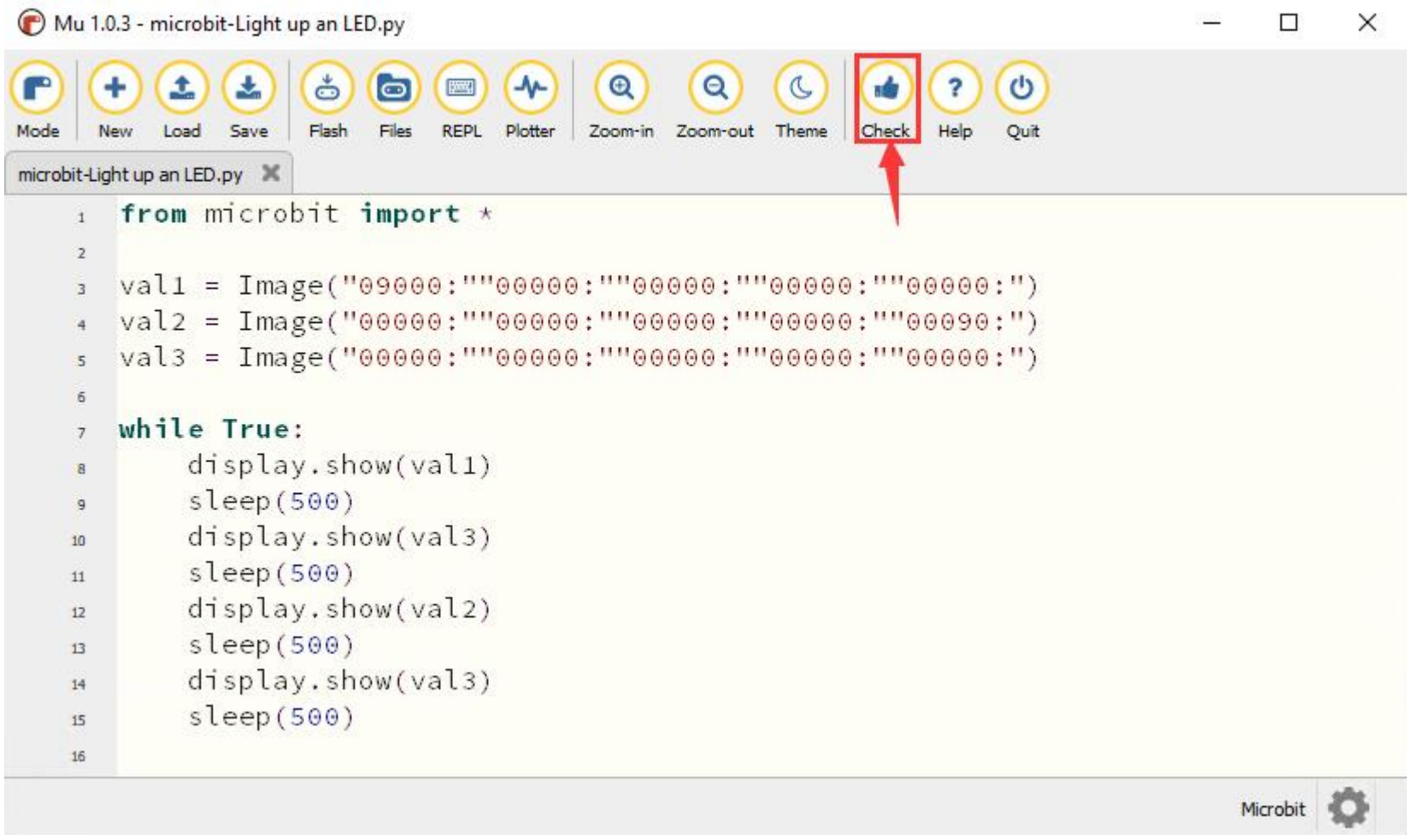
Type	Route	File Name
Python file	../Python code/8.2：Light up an LED	microbit-Light up an LED .py

You can also input code in the editing window yourself.

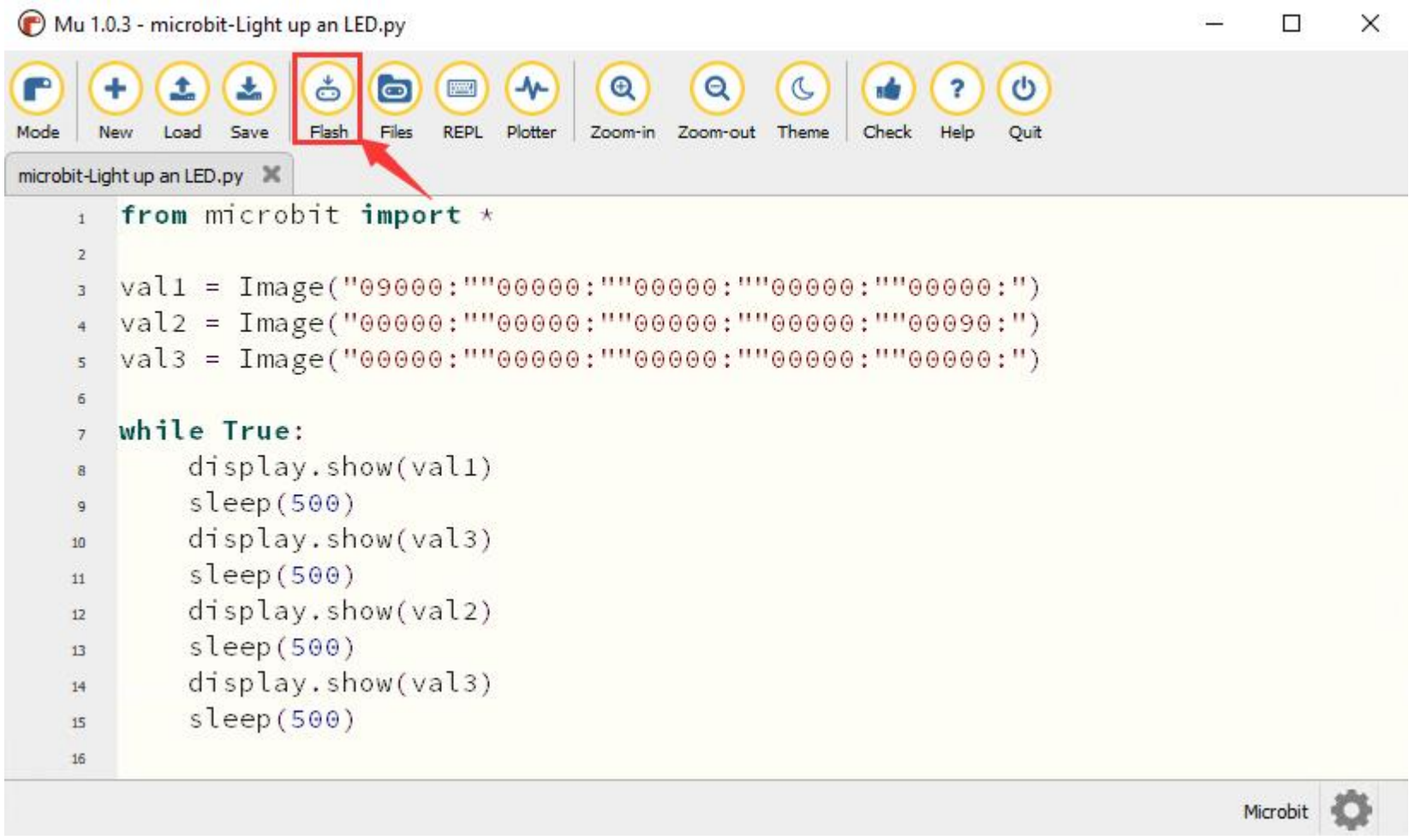
(Note: All English words and symbols must be written in English)



Click “Check” to examine error in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download code to micro:bit board.



4. Test Result

After uploading the test code to the micro:bit main board and powering the board via an USB cable, then the LED in (1,0) will be on and off for 0.5s and the one in (3,4) will be on and off for 0.5s and repeat this sequence.

5. Code Explanation

from microbit import *	Import the library file of micro：bit

val1 = Image("09000:""00000:""00000:""00000:""00000:") val2 = Image("00000:""00000:""00000:""00000:""00090:") val3 = Image("00000:""00000:""00000:""00000:""00000:")	Set Image() to val1 Pixel of each LED on micro:bit can be set in one of ten values If set pixel to 0 (zero) , which means in close state, literally, 0 is brightness, 9 is best brightness Set Image() to val2 Set Image() to val3
while True:	This is a permanent loop that makes micro:bit execute the code in this loop forever.
display.show(val1) sleep(500) display.show(val3) sleep(500)	LED at (1,0) blinks for 0.5s
display.show(val2) sleep(500) display.show(val3) sleep(500)	LED at (3,4) flashes for 0.5s

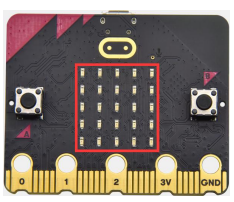
6. Reference

sleep(ms) : delay time

For more details about delay, please refer to the link:

https://microbit-micropython.readthedocs.io/en/latest/utime.html

Project 3：5* 5 LED Dot Matrix



1. Description

Dot matrix is very commonplace in daily life, which has found wide applications in LED advertisement screens, elevator floor display, bus stop announcement and so on.

The LED dot matrix of Micro: Bit main board contains 25 Diodes. Previously, we have succeeded in controlling a certain LED via its position point. Supported by the same theory, we can turn on many LEDs at the same time to showcase patterns, digits and characters.

What’ s more, we can also click” show icon “ to choose the pattern we like to display. Last but not the least, we can design patterns by ourselves as well.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

3. Test Code

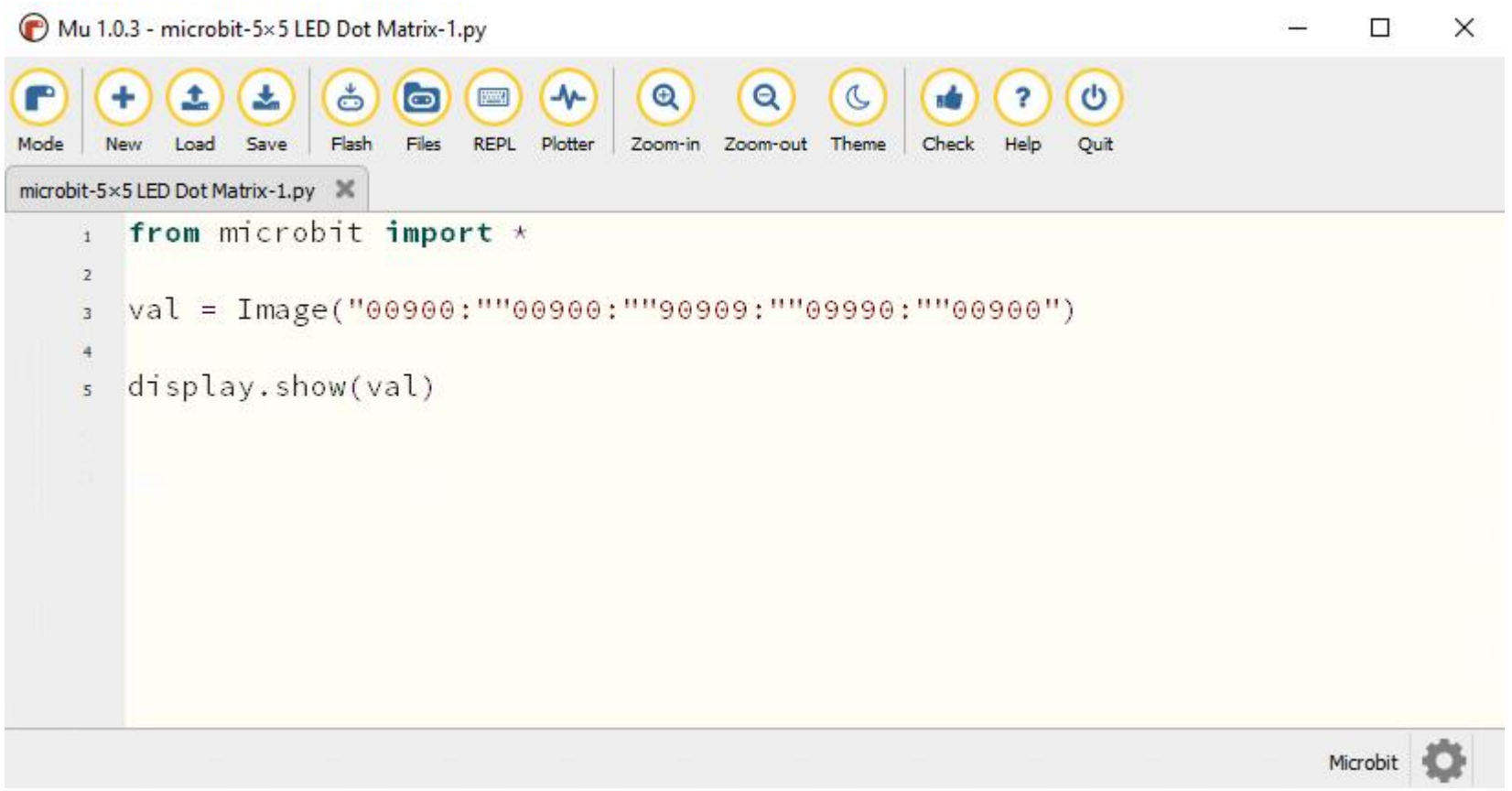
Test Code1：

You could open “5×5 LED Dot Matrix-1.py “file to import the code (How to load the project code?)

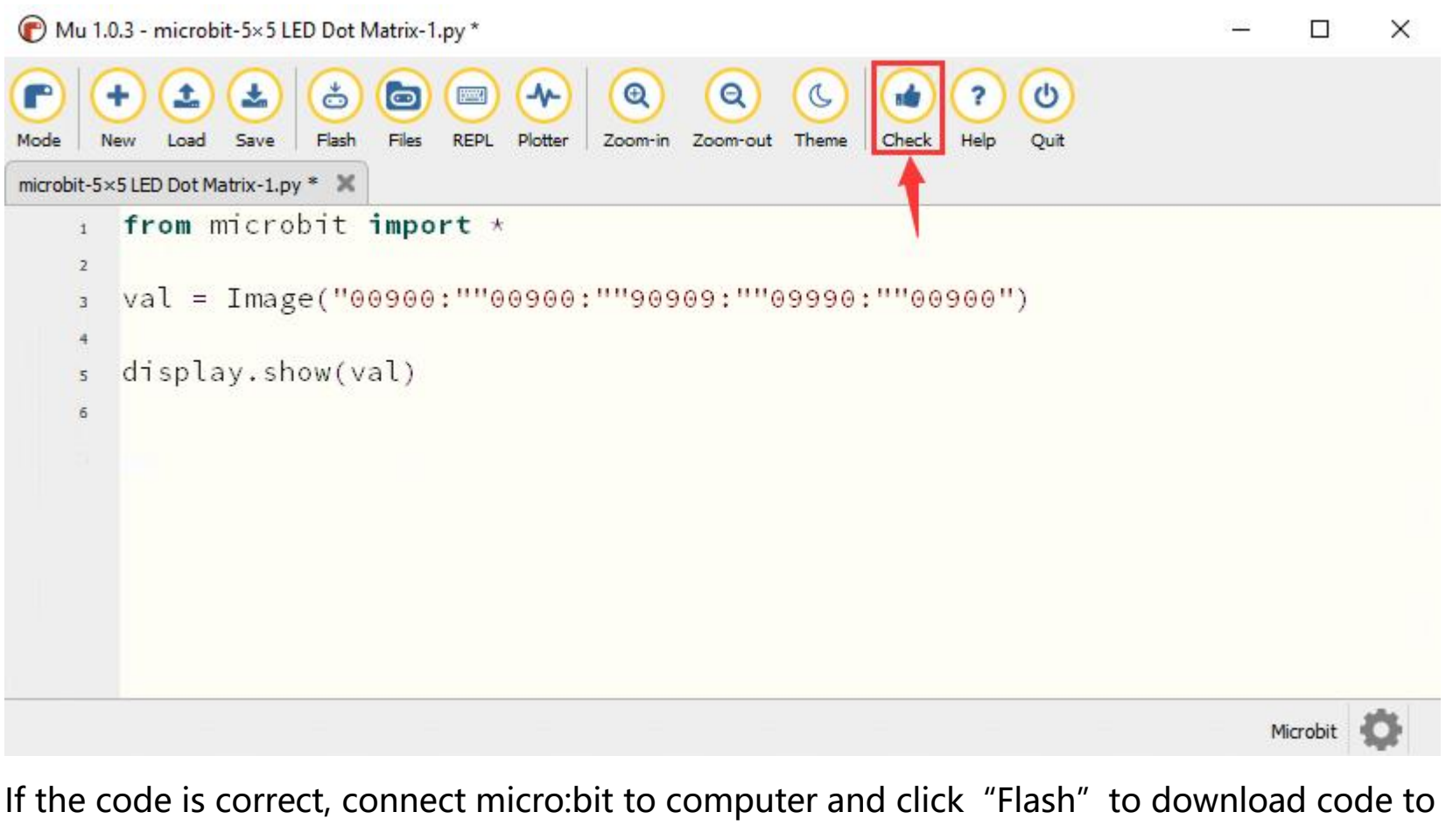
File Type	Route	File Name
Python file	../Python code/8.3：5×5 LED Dot Matrix	microbit-5× 5 LED Dot Matrix-1.py

You can also input code in the editing window yourself.

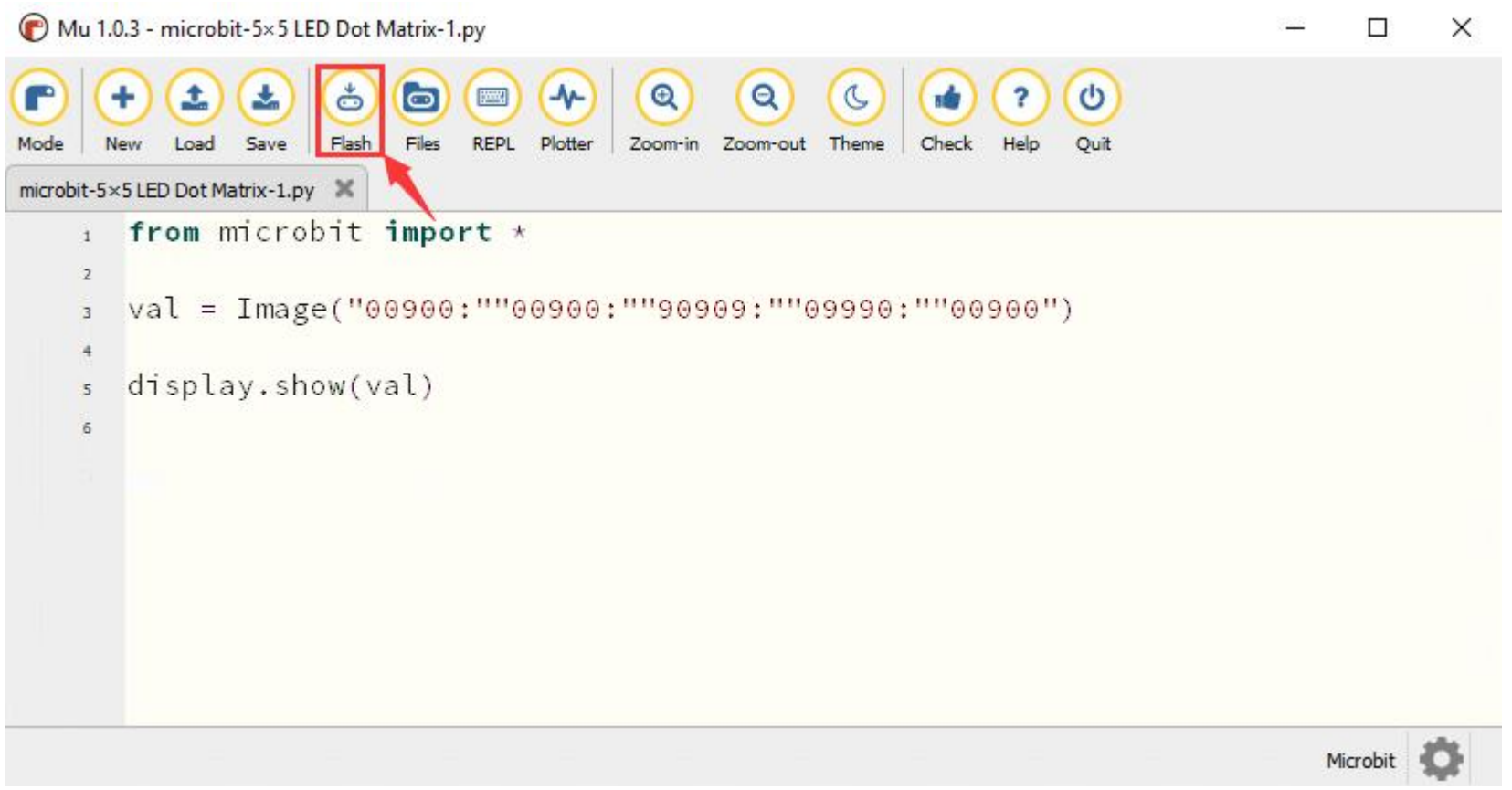
(Note: All words and symbols must be written in English.)



Click "Check" to examine the error in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect micro:bit to computer and click "Flash" to download code to micro:bit board.



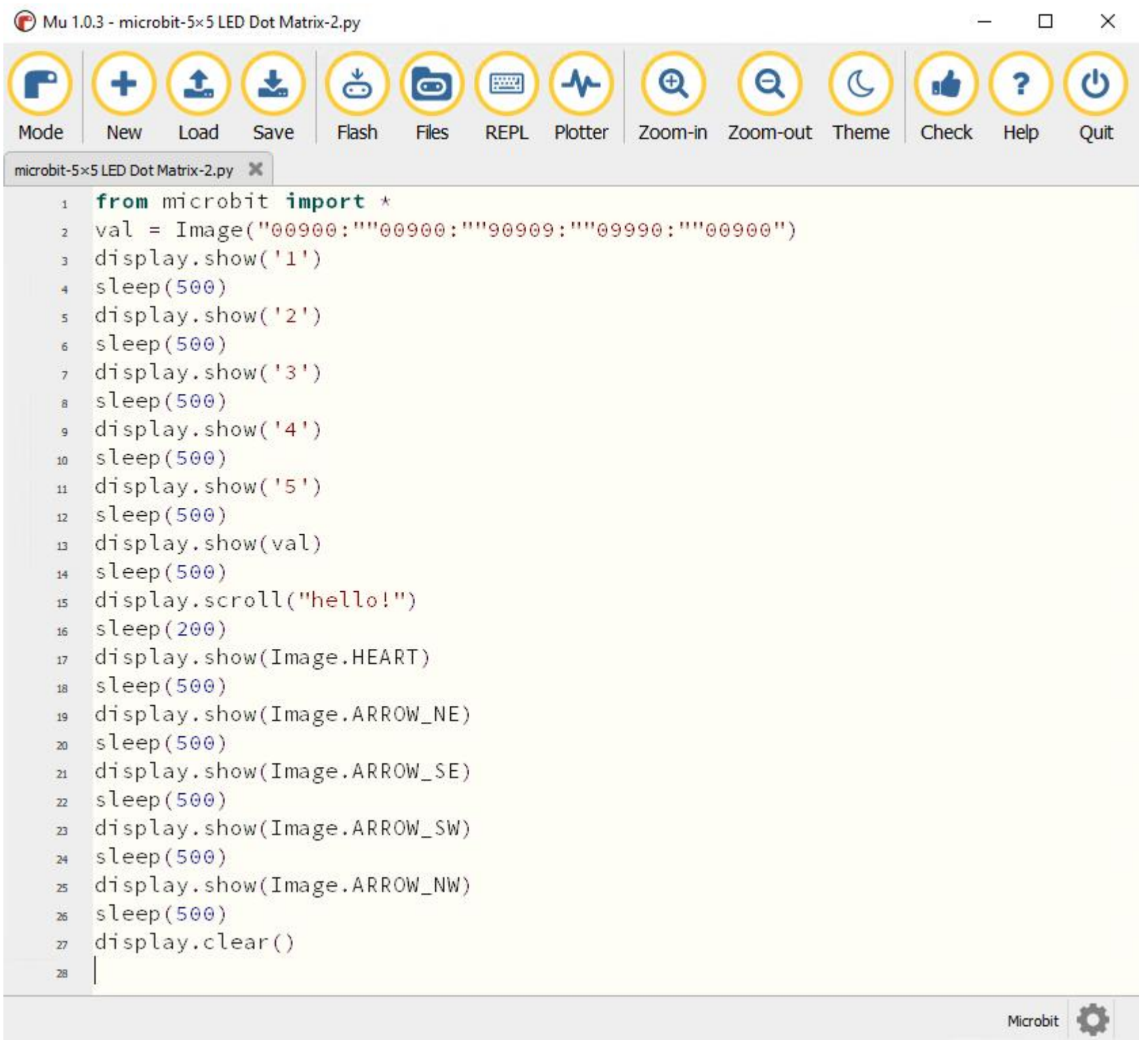
Test Code 2：

You could open "5×5 LED Dot Matrix-2.py " file to import the code (How to load the project code?)

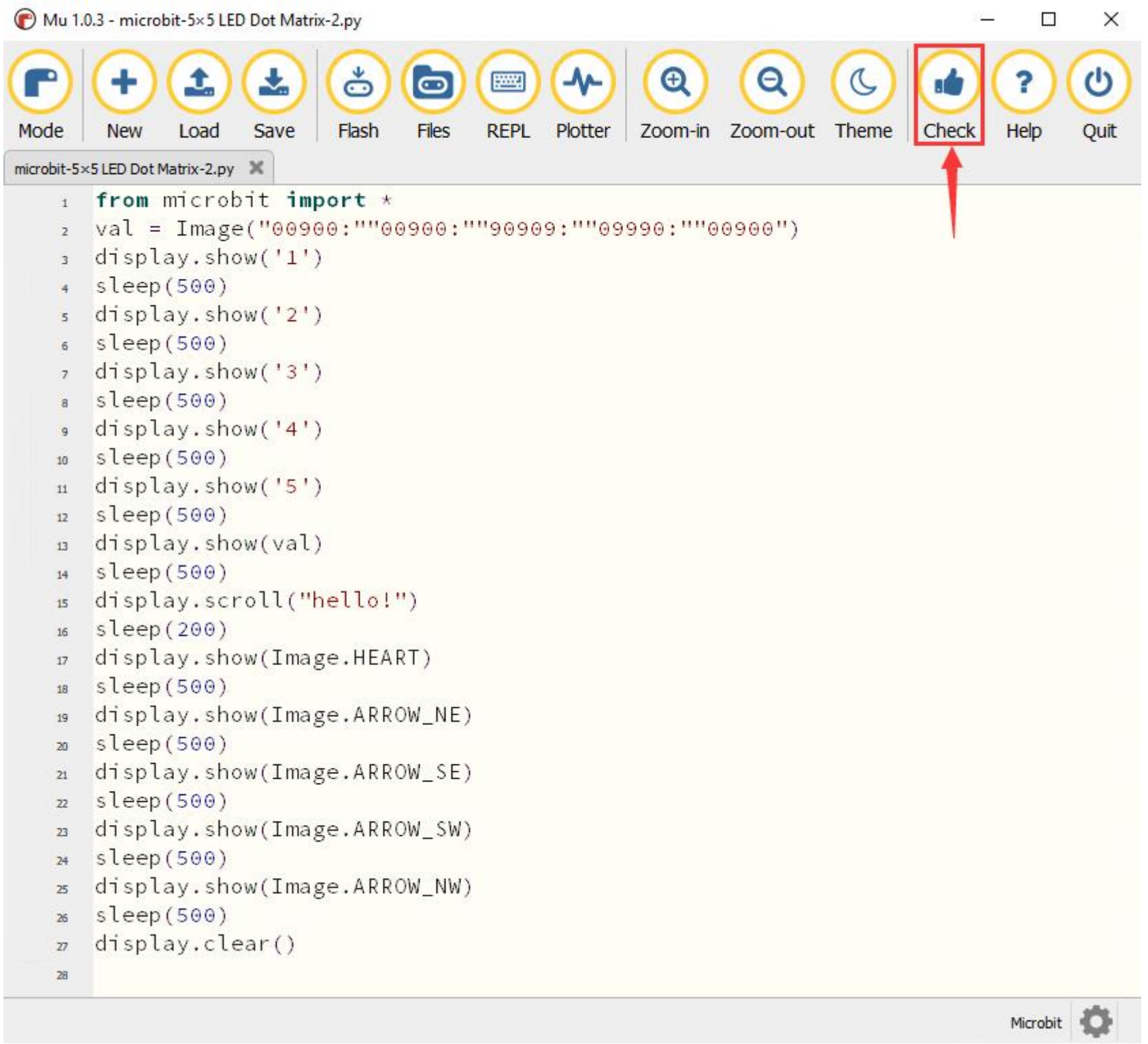
File Type	Route	File Name
Python file	../Python code/8.3：5×5 LED Dot Matrix	microbit-5×5 LED Dot Matrix-2.py

You can also input code in the editing window yourself.

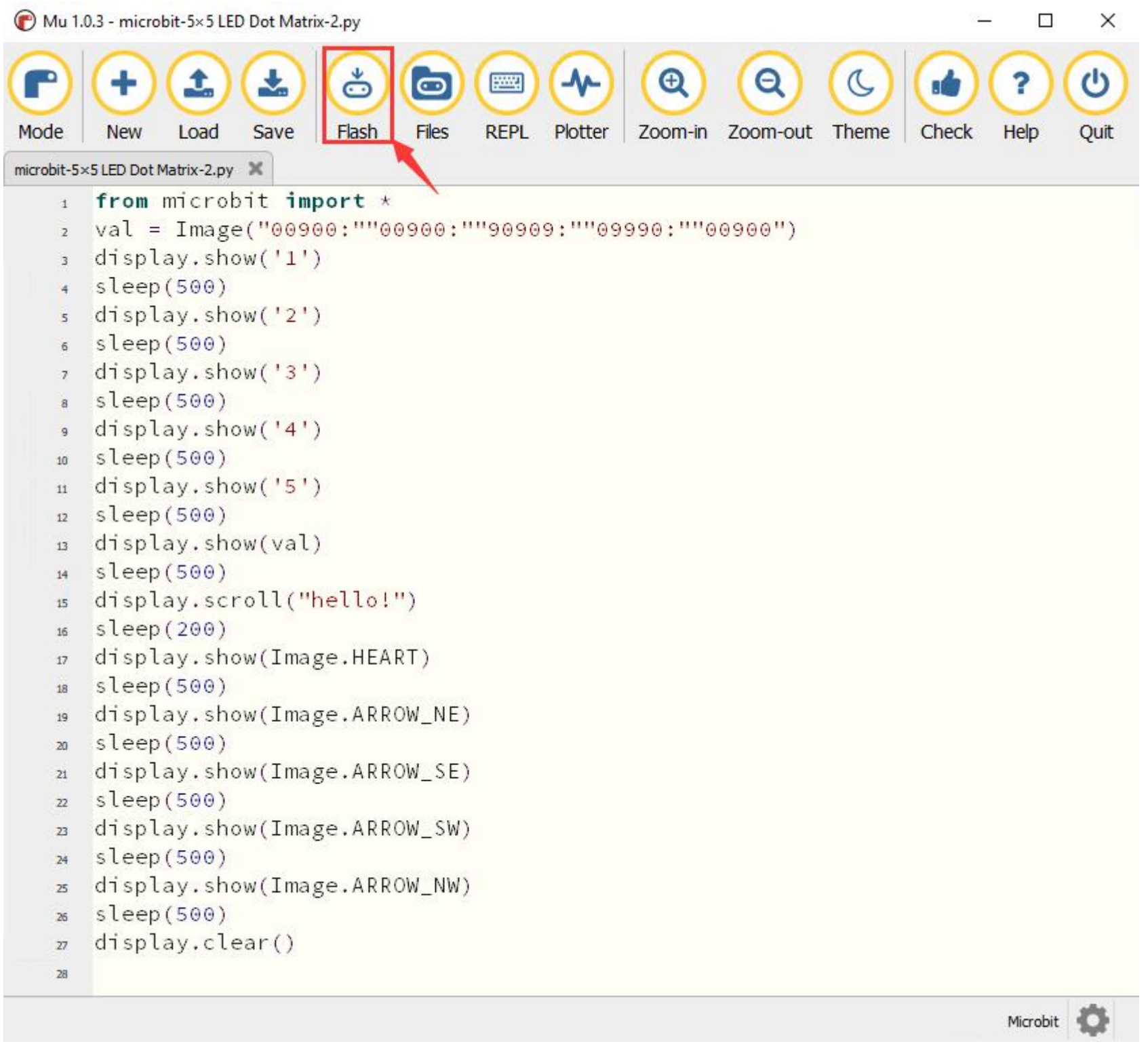
(Note: All words and symbols must be written in English.)



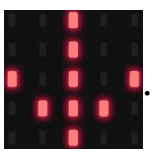
Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.

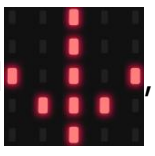
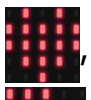
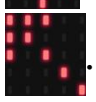


If the code is correct, connect the micro:bit to the computer and click “Flash” to download code to micro:bit board.



4. Test Result

After uploading the test code 1 to the micro:bit main board and powering the main board via the USB cable, we will find that the 5*5 dot matrix start to show a downward arrow  .

After uploading the test code 2 to the micro:bit main board, we will find that the 5*5 dot matrix start to show numbers 1,2,3,4 and 5 and then it alternatively shows a downward arrow word  , “Hello”, a heart pattern  , an arrow pointing at northeast  , then at southeast  , then at southwest  , and then at northwest  .

5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
val = Image("09000:""00000:""00000:""00000:""00000:")	Set Image() to variable val
display.show(val)	micro:bit shows “→”
display.show('1')	micro:bit shows “1”
sleep(500)	Delay in 500ms
display.scroll("hello!")	micro:bit scrolls to show “hello!”
display.show(Image.HEART)	micro:bit displays “❤” pattern
display.show(Image.ARROW_NE) display.show(Image.ARROW_SE) display.show(Image.ARROW_SW) display.show(Image.ARROW_NW)	micro:bit shows “Northeast” arrow micro:bit displays “Southeast” arrow micro:bit shows “Southwest” arrow micro:bit displays “Northwest” arrow
display.clear()	The LED dot matrix of micro:bit clears

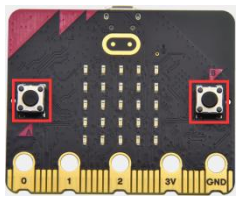
6. Reference

display.scroll()：

The display scrolls to show the values, if it is integer or float, we will use str（）to transfer it into character strings.

More details, please refer to the link：https://microbit-micropython.readthedocs.io/en/latest/utime.html

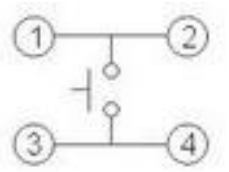
Project 4：Programmable Buttons



1. Description

Buttons can be used to control circuits. In an integrated circuit with a push button, the circuit is connected when pressing the button and but after release, it will break again.

Both ends of the  button like two mountains. There is a river in between. The internal metal piece connect the two sides to let the current pass, just like building a bridge to connect two mountains.

The internal structure of the button is shown as follows: before pressing the button, 1 ,2 , 3 and 4 are turned on. However, 1, 3 or 1, 4 or 2, 3 or 2 and 4 are disconnected, which is only enabled when the button is pressed. 

Micro: Bit main board boasts three push buttons, two are programmable buttons(marked with A and B), and the one on the other side is a reset button. By pressing the two programmable buttons can input three different signals. We can press button A or B alone or press them together and the LED dot matrix shows A,B and AB respectively. Let’ s get started.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

3. Test Code

Test Code 1：

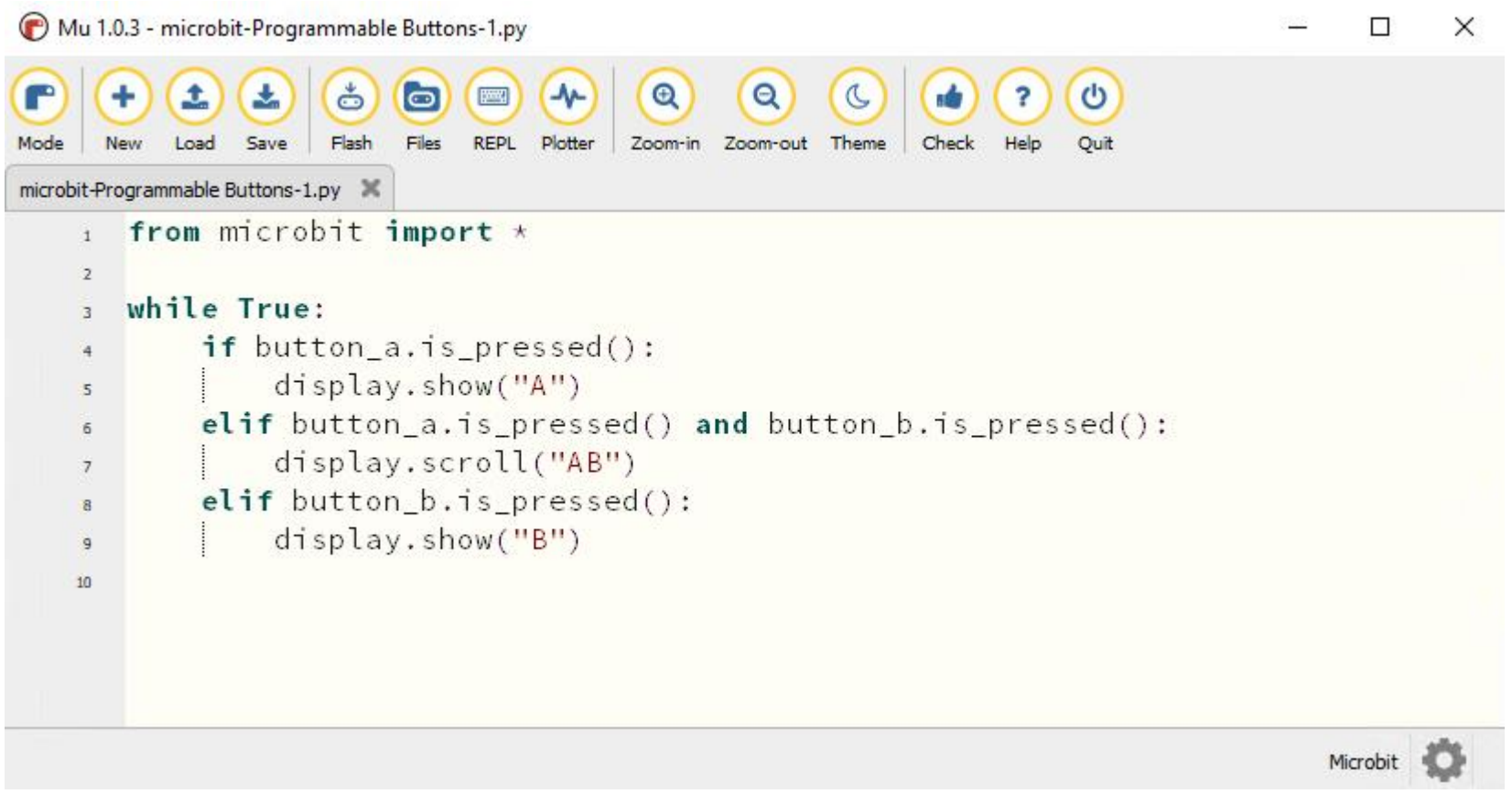
Enter Mu software and open the file “Project 4：Programmable Buttons-1.py” to import the code:

(How to load the project code?)

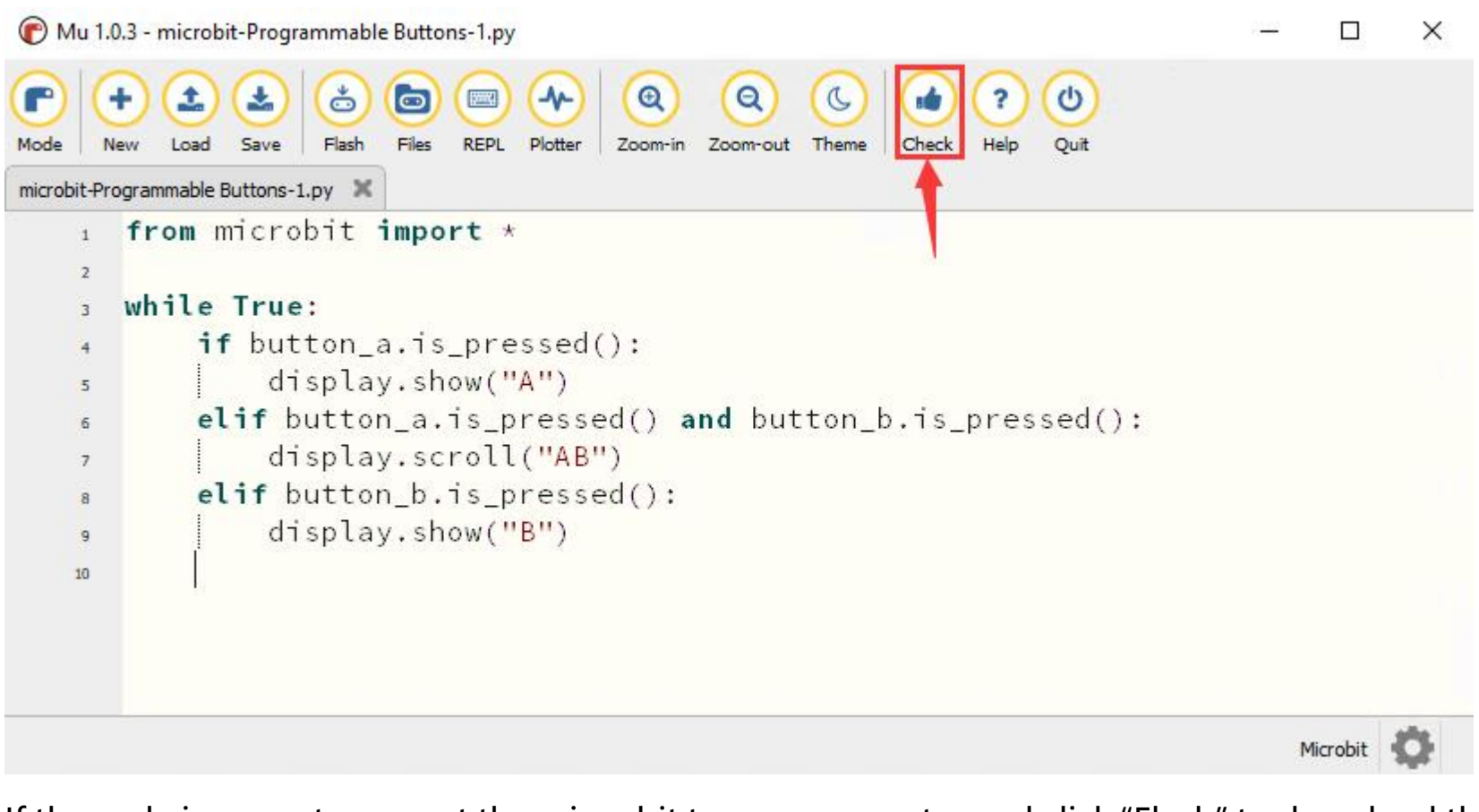
File Type	Route	File Name
Python file	../Python code/8.4：Programmable Buttons	microbit-Programmable Buttons-1.py

You can also input code in the editing window yourself.

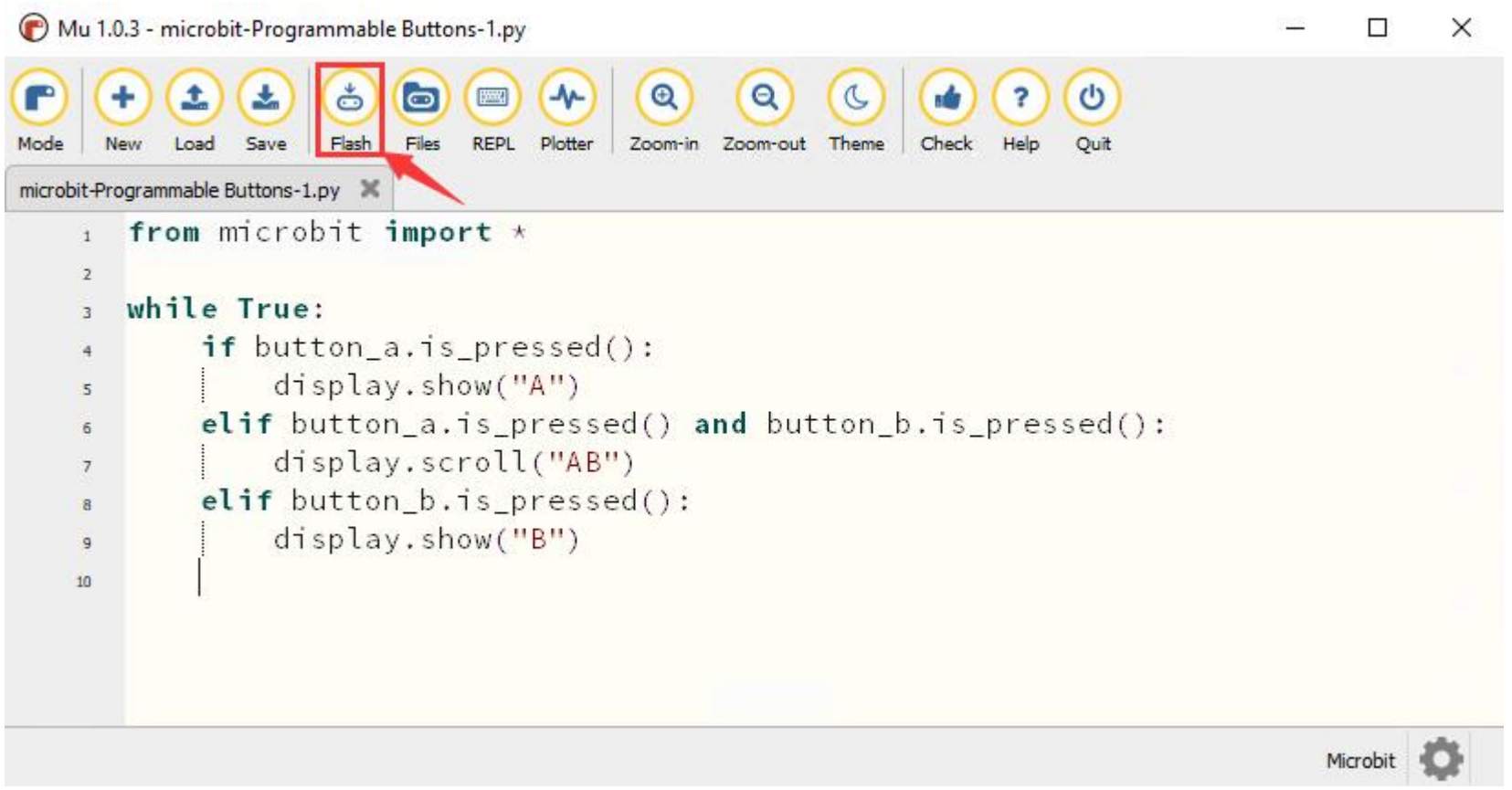
(Note: All words and symbols must be written in English.)



Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



Test Code 2:

Enter Mu software and open the file "Project 4: Programmable Buttons-2.py" to import the code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Projects/8.4: Programmable Buttons	microbit- Programmable Buttons-2.py

You can also input code in the editing window yourself.

(Note: All words and symbols must be written in English.)

Mu 1.0.3 - microbit- Programmable Buttons-2.py

Mode | New | Load | Save | Flash | Files | REPL | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

microbit- Programmable Buttons-2.py

```
from microbit import *
a = 0
b = 0
val1 = Image("00000:""00000:""00000:""00000:""00900")
val2 = Image("00000:""00000:""00000:""00900:""99999")
val3 = Image("00000:""00000:""00900:""99999:""99999")
val4 = Image("00000:""00900:""99999:""99999:""99999")
val5 = Image("00900:""99999:""99999:""99999:""99999")
val6 = Image("99999:""99999:""99999:""99999:""99999")
display.show(val1)

while True:
    while button_a.is_pressed() == True:
        sleep(10)
        if button_a.is_pressed() == False:
            a = a + 1
            if(a >= 5):
                a = 5
            break
    while button_b.is_pressed() == True:
        sleep(10)
        if button_b.is_pressed() == False:
            a = a - 1
            if(a <= 0):
                a = 0
            break
    if a == 0:
        display.show(val1)
    if a == 1:
        display.show(val2)
    if a == 2:
        display.show(val3)
    if a == 3:
        display.show(val4)
    if a == 4:
        display.show(val5)
    if a == 5:
        display.show(val6)
```

Microbit

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.

Mu 1.0.3 - microbit- Programmable Buttons-2.py

Mode | New | Load | Save | Flash | Files | REPL | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

microbit- Programmable Buttons-2.py

```
from microbit import *
a = 0
b = 0
val1 = Image("00000:""00000:""00000:""00000:""00900")
val2 = Image("00000:""00000:""00000:""00900:""99999")
val3 = Image("00000:""00000:""00900:""99999:""99999")
val4 = Image("00000:""00900:""99999:""99999:""99999")
val5 = Image("00900:""99999:""99999:""99999:""99999")
val6 = Image("99999:""99999:""99999:""99999:""99999")
display.show(val1)

while True:
    while button_a.is_pressed() == True:
        sleep(10)
        if button_a.is_pressed() == False:
            a = a + 1
            if(a >= 5):
                a = 5
            break
    while button_b.is_pressed() == True:
        sleep(10)
        if button_b.is_pressed() == False:
            a = a - 1
            if(a <= 0):
                a = 0
            break
    if a == 0:
        display.show(val1)
    if a == 1:
        display.show(val2)
    if a == 2:
        display.show(val3)
    if a == 3:
        display.show(val4)
    if a == 4:
        display.show(val5)
    if a == 5:
        display.show(val6)
```

Microbit

If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.

Mu 1.0.3 - microbit- Programmable Buttons-2.py

Mode | New | Load | Save | Flash | Files | REPL | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

microbit- Programmable Buttons-2.py

```
from microbit import *
a = 0
b = 0
val1 = Image("00000:""00000:""00000:""00000:""00900")
val2 = Image("00000:""00000:""00000:""00900:""99999")
val3 = Image("00000:""00000:""00900:""99999:""99999")
val4 = Image("00000:""00900:""99999:""99999:""99999")
val5 = Image("00900:""99999:""99999:""99999:""99999")
val6 = Image("99999:""99999:""99999:""99999:""99999")
display.show(val1)

while True:
    while button_a.is_pressed() == True:
        sleep(10)
        if button_a.is_pressed() == False:
            a = a + 1
            if(a >= 5):
                a = 5
            break
    while button_b.is_pressed() == True:
        sleep(10)
        if button_b.is_pressed() == False:
            a = a - 1
            if(a <= 0):
                a = 0
            break
    if a == 0:
        display.show(val1)
    if a == 1:
        display.show(val2)
    if a == 2:
        display.show(val3)
    if a == 3:
        display.show(val4)
    if a == 4:
        display.show(val5)
    if a == 5:
        display.show(val6)
```

Microbit

4. Test Result

After uploading the test code1 to the micro:bit main board and powering the board via the USB cable, the 5*5 LED dot matrix shows "A" if button A is pressed, then "B" if button B is pressed, and "AB" if button A and B are pressed together.

After uploading the test code 2 to the micro:bit main board and powering the board via the USB cable, if the button A is pressed, the LEDs turning red increase while if the button B pressed, the LEDs turning red reduce.

5. Code Explanation

from microbit import *	Import the library file of micro：bit
while True:	This is a permanent loop that makes micro:bit execute the code of it.

if button_a.is_pressed():
display.show("A")
elif button_a.is_pressed() and button_b.is_pressed():
display.scroll("AB")
elif button_b.is_pressed():
display.show("B")
	If button A is pressed
	micro:bit shows “A”
	If button A and B are pressed at same time
	micro:bit displays “AB”
	If button B is pressed
	micro:bit shows “B”

while button_a.is_pressed() == True:
sleep(10)
if button_a.is_pressed() == False:
a = a + 1
if(a >= 5):
a = 5
break
while button_b.is_pressed() == True:
sleep(10)
if button_b.is_pressed() == False:
a = a - 1
if(a <= 0):
a = 0
break
if a == 0:
display.show(val1)
if a == 1:
display.show(val2)
if a == 2:
display.show(val3)
	When the button A is pressed
	Delay in 10ms to eliminate the shaking of button A
	when button A is released,
	Variable a adds 1
	If variable a≥5
	Variable a=5
	exit the loop
	when button B is pressed
	Delay in 10ms to eliminate the shaking of button B
	When the button B is released
	Variable a reduces 1 gradually
	When a≤0
	Variable a=0
	exit the loop
	When a=0
	micro:bit shows pattern val1
	When a=1
	micro:bit displays pattern val2

if a == 3:
display.show(val4)
if a == 4:
display.show(val5)
if a == 5:
display.show(val6)
	When a=2
	micro:bit shows pattern val3
	If a=3
	micro:bit displays pattern val4
	If a=4
	micro:bit shows pattern val5
	If a=5
	micro:bit displays pattern val6

Project 5：Temperature Detection

1. Description

The Micro:bit main board is not equipped with a temperature sensor, but uses the built-in temperature sensor in NFR52833 chip for temperature detection. Therefore, the detected temperature is more closer to the temperature of the chip, and there maybe deviation from the ambient temperature.

In this project, we will seek to use the sensor to test the temperature in the current environment, and display the test results in the display data (device). And then control the LED dot matrix to display different patterns by setting the temperature range detected by the sensor.

Note: the temperature sensor of Micro:bit main board is shown below:

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

3. Test Code

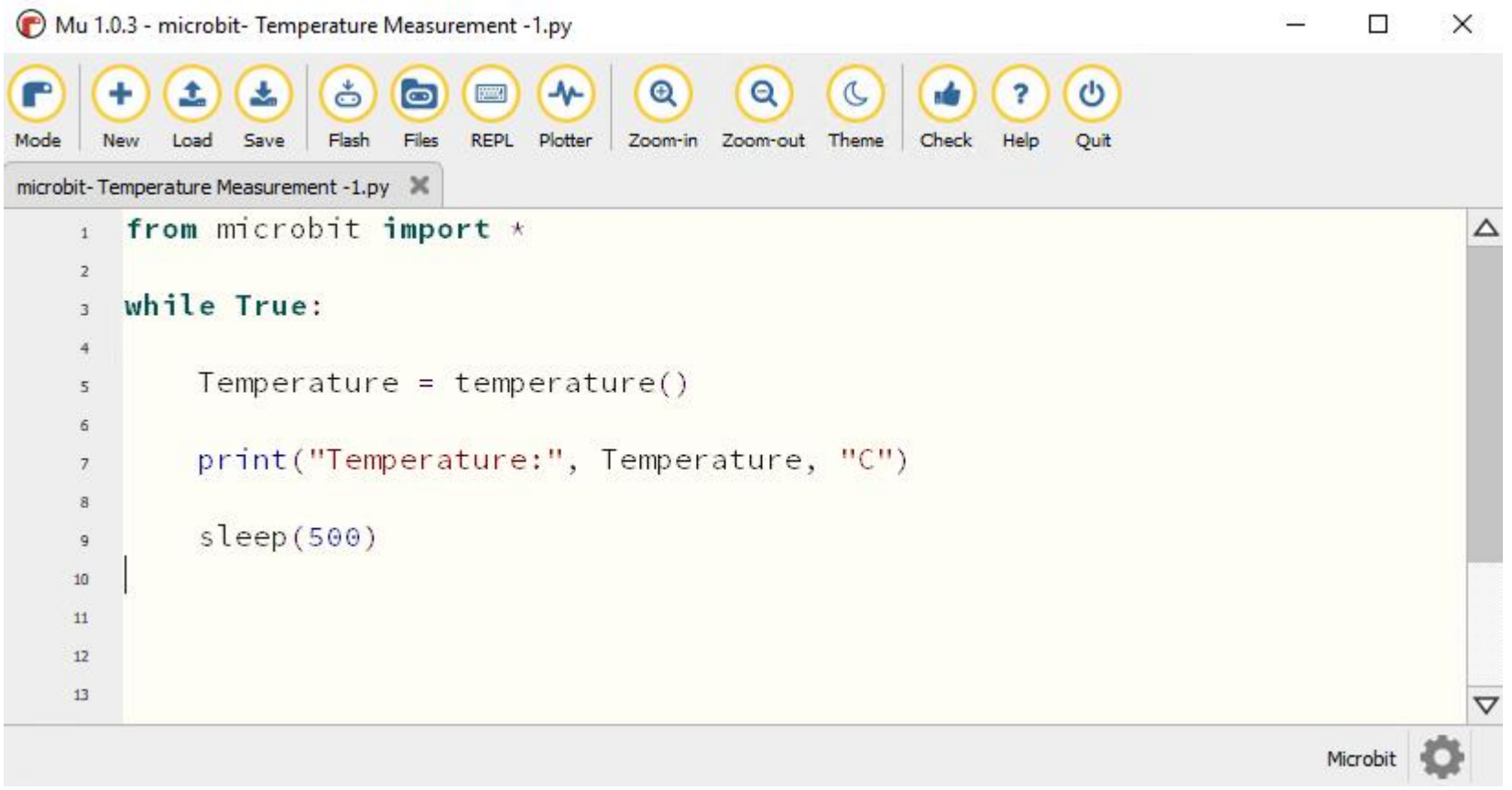
Test Code 1：

Enter Mu software and open the file “Project 5：Temperature Measurement -1.py “ to import code:

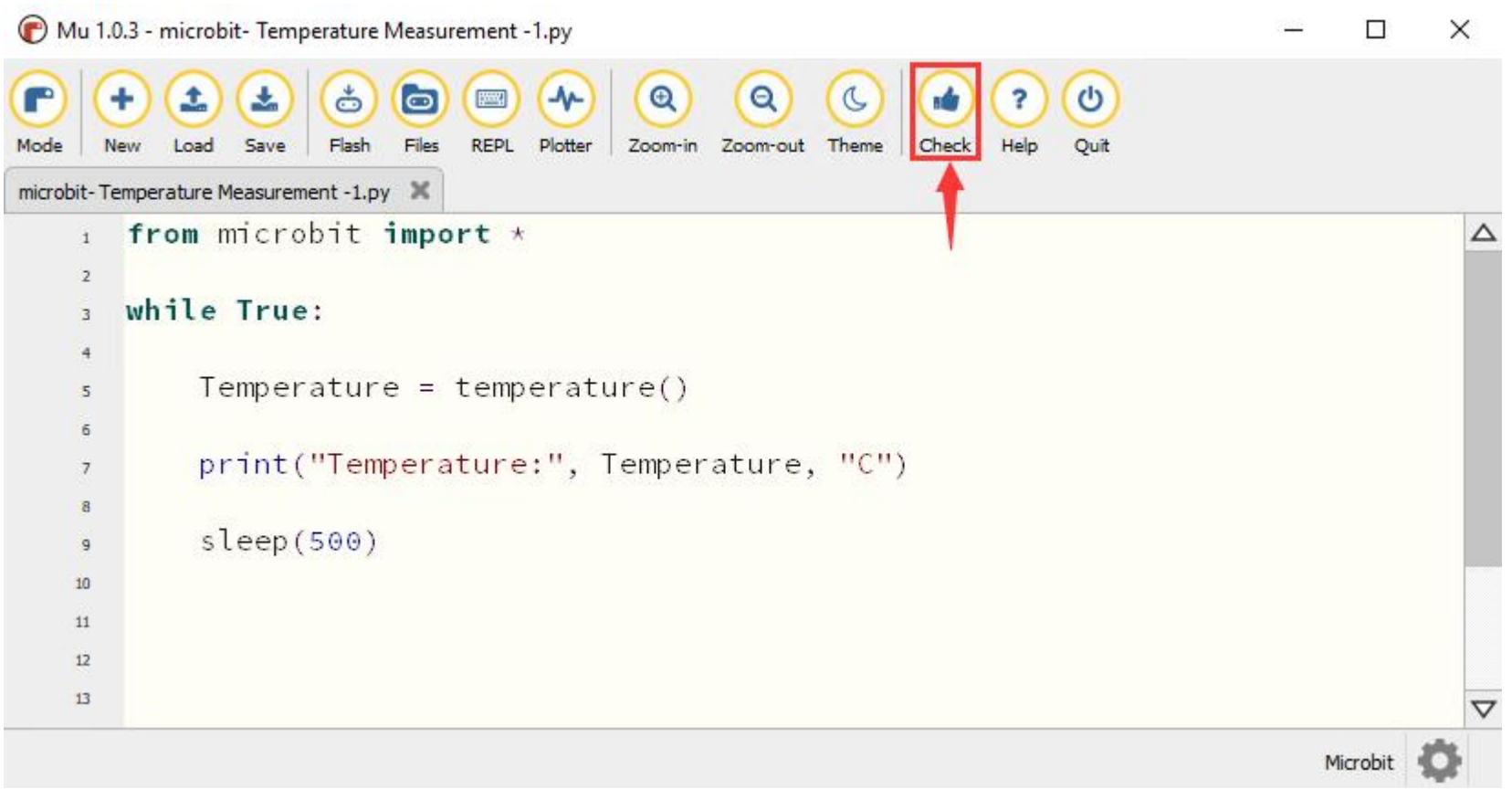
File Type	Route	File Name
Python file	../Python code/8.5： Temperature Measurement	microbit- Temperature Measurement -1.py

You can also input code in the editing window yourself.

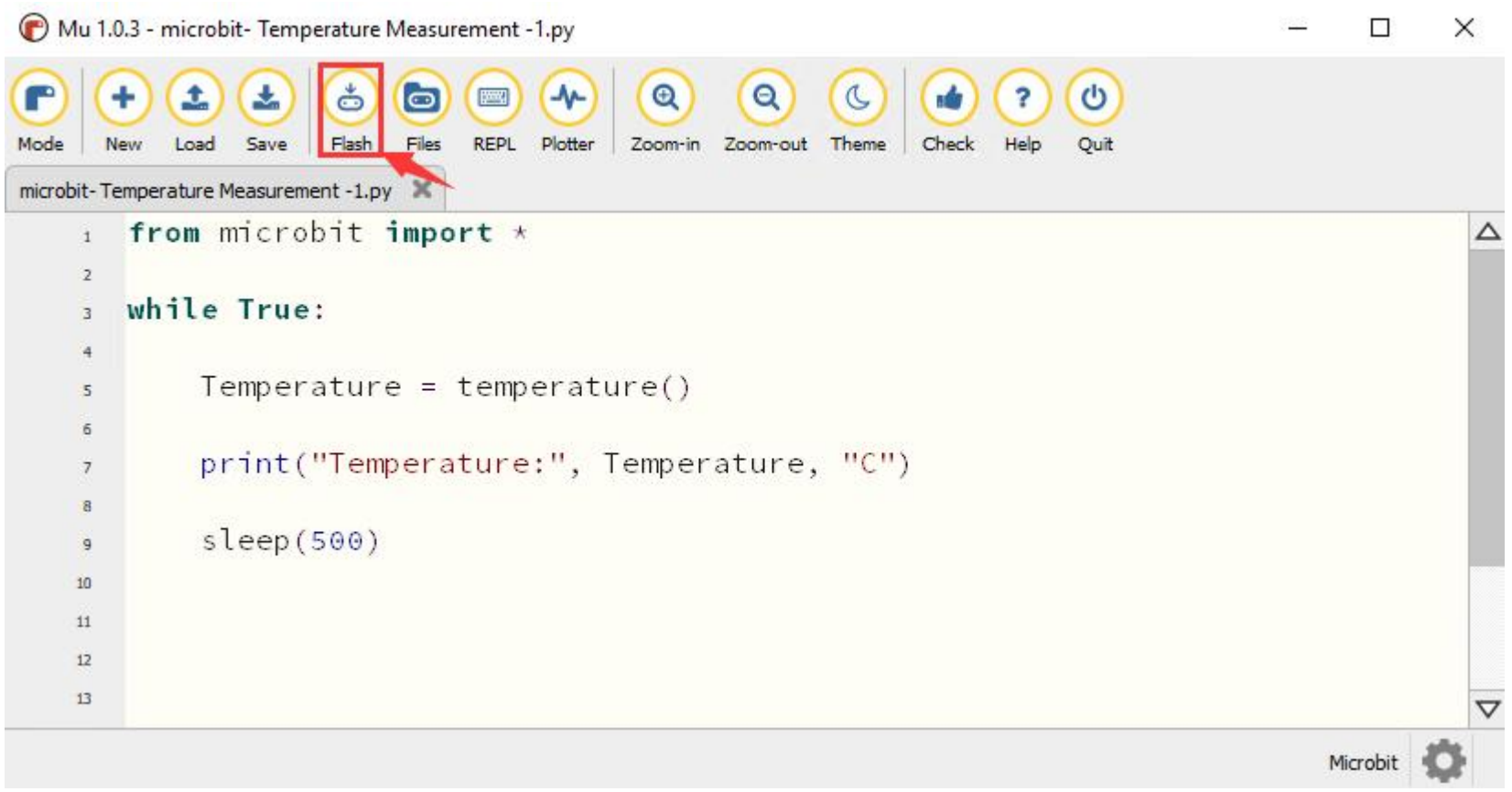
(Note: All words and symbols must be written in English.)



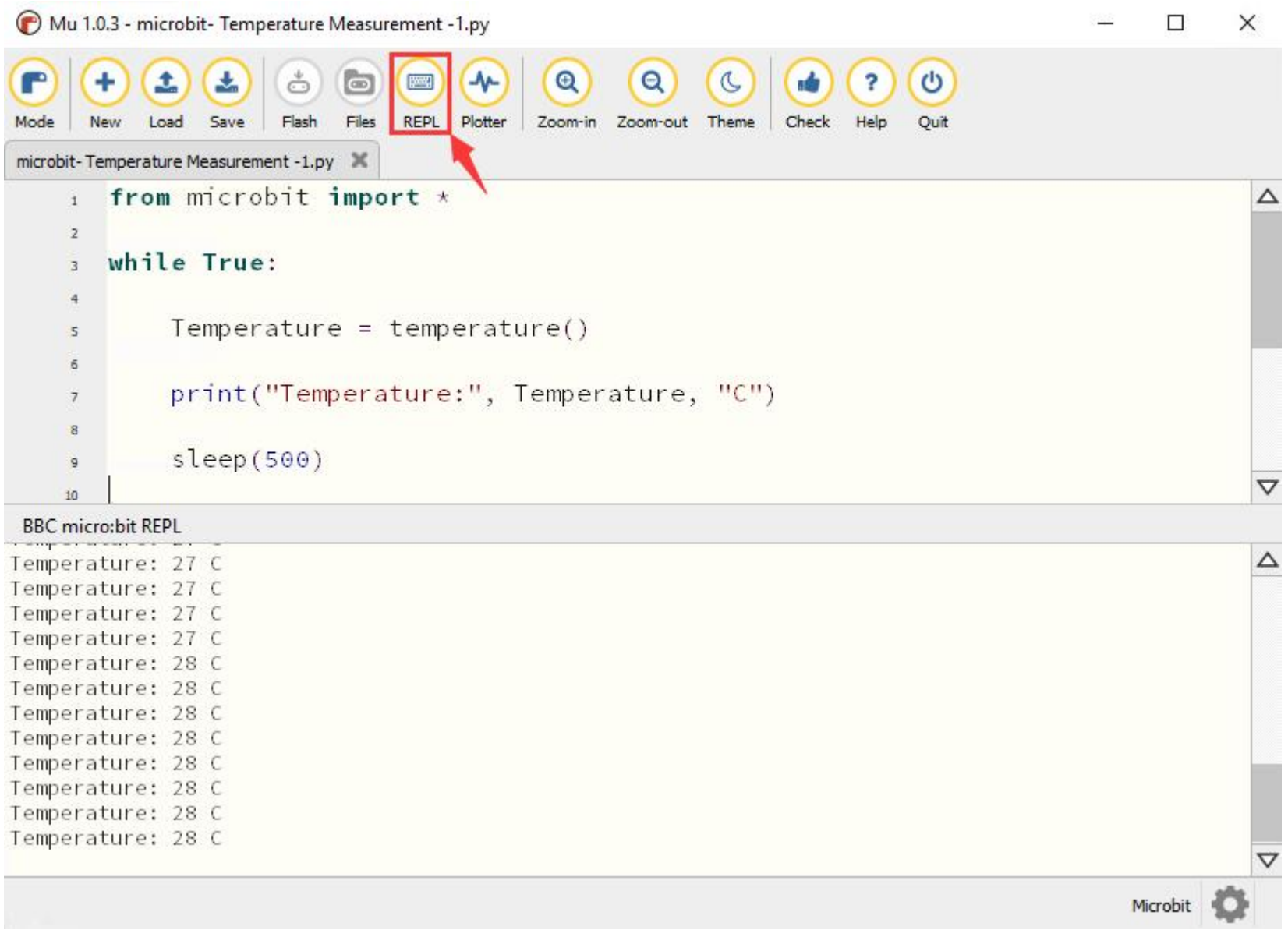
Click “Check” to examine error in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect micro:bit to computer and click “Flash” to download code to micro:bit board.



After downloading the test code 1 to the micro:bit board, keep USB connected and click "REPL" and press the reset button on micro:bit. Then REPL window will show the ambient temperature value, as shown below: (C stands for temperature unit)



Test Code 2:

Enter Mu software and open the file "Project 5: Temperature Measurement -2.py" to import code:

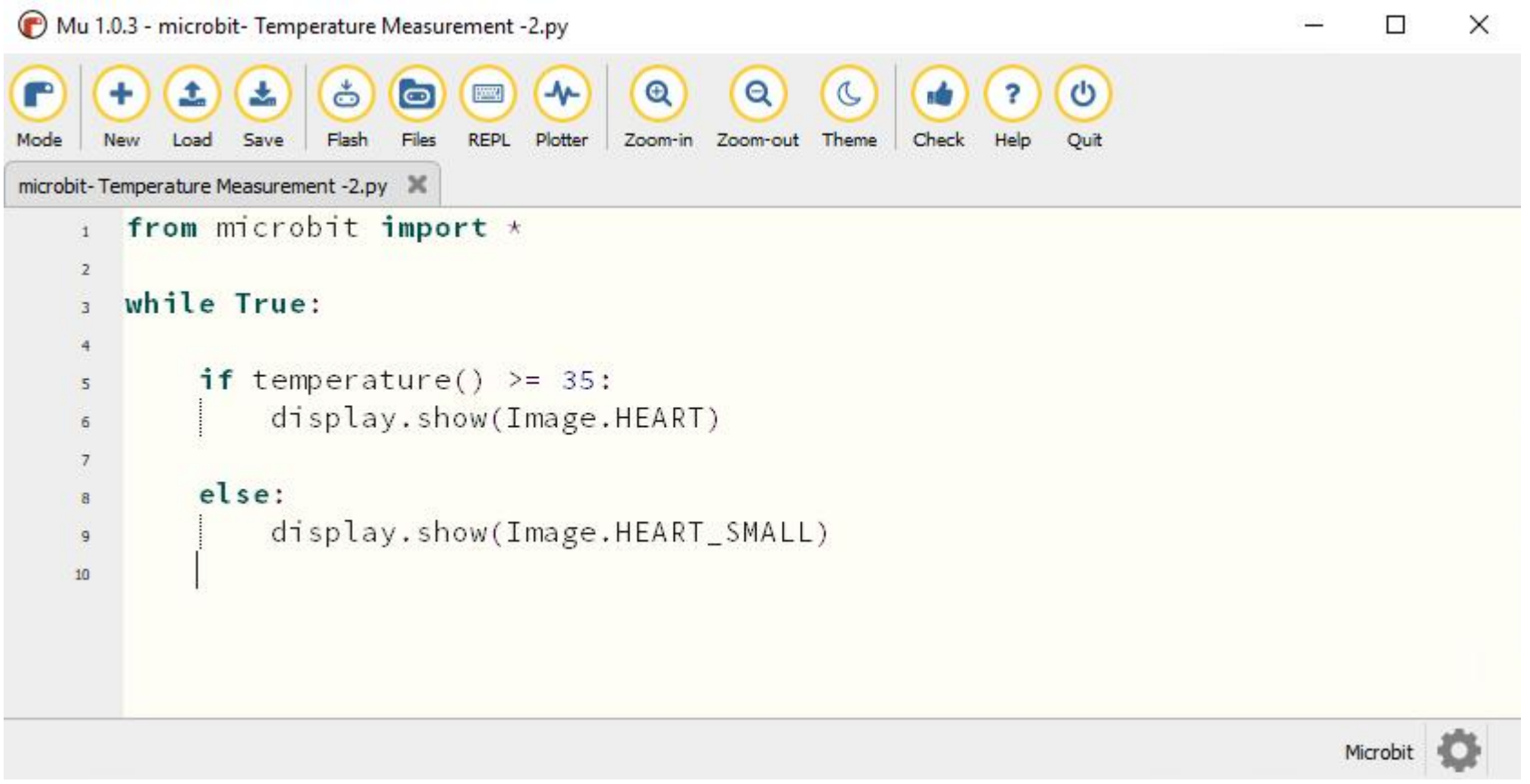
File	Route	File Name

Type		
Python file	../Python code/8.5： Temperature Measurement	microbit- Temperature Measurement -2.py

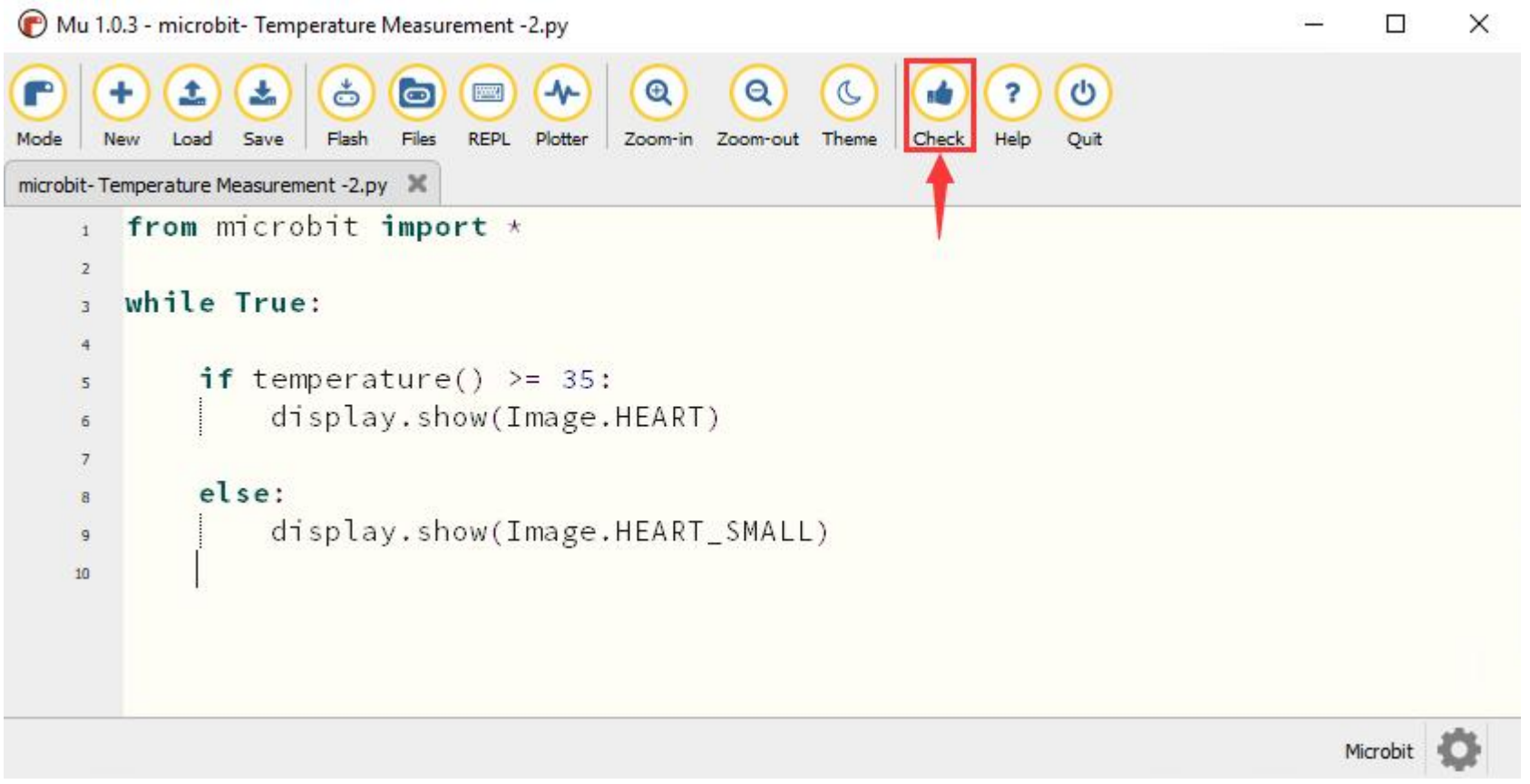
You can also input code in the editing window yourself.

(Note: All words and symbols must be written in English.)

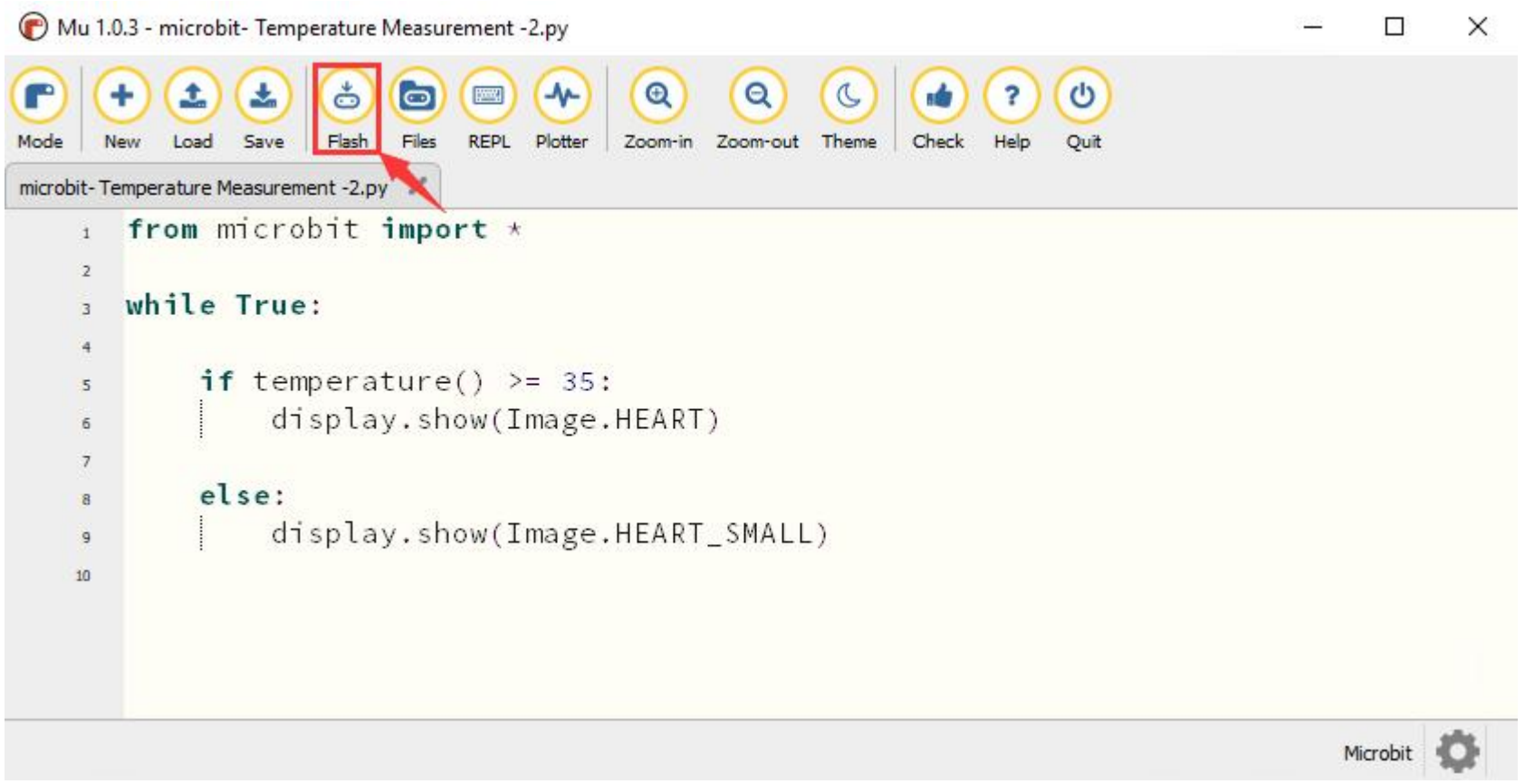
The temperature value can be set in compliance with the real temperature.



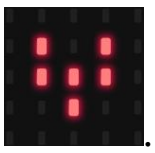
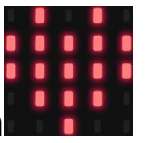
Click “Check” to examine error in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to the computer and click “Flash” to download the code to the micro:bit board.



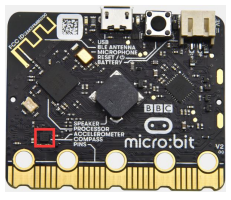
4. Test Result

After uploading the code 2 to the board, when the ambient temperature is less than 35℃, the 5*5 LED dot matrix shows  . When the temperature is equivalent to or greater than 35℃, the pattern  appears.

5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
while True:	This is a permanent loop that makes micro:bit execute the code of it.
Temperature = temperature()	Set temperature() to Temperature
print("Temperature:", Temperature, "C")	BBC micro:bit REPL prints temperature value
sleep(500)	Delay in 500ms
if temperature() >= 35: display.show(Image.HEART) **else**: display.show(Image.HEART_SMALL)	If temperature value ≥35℃ micro:bit shows "♥ " If temperature value<35℃ micro:bit displays " "

Project 6：Geomagnetic Sensor



1. Description

This project mainly introduces the use of the micro:bit’ s geomagnetic sensor. In addition to detecting the strength of the magnetic field, it can also be used to determine the direction, which is an important part of the heading and attitude reference system (AHRS) as well.

It uses FreescaleMAG3110 three-axis magnetometer. Its I2C interface communicates with the outside, and the range is ±1000μT, the maximum data update rate is 80Hz. Combined with an accelerometer, it can calculate the position. Additionally, it is applied to magnetic detection and compass blocks.

Then we could read the value detected by it to determine the location. We need to calibrate the micro:bit board when the magnetic sensor works. The correct calibration method is to rotate the micro:bit board.

In addition, the objects nearby may affect the accuracy of readings and calibration.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

3. Test Code

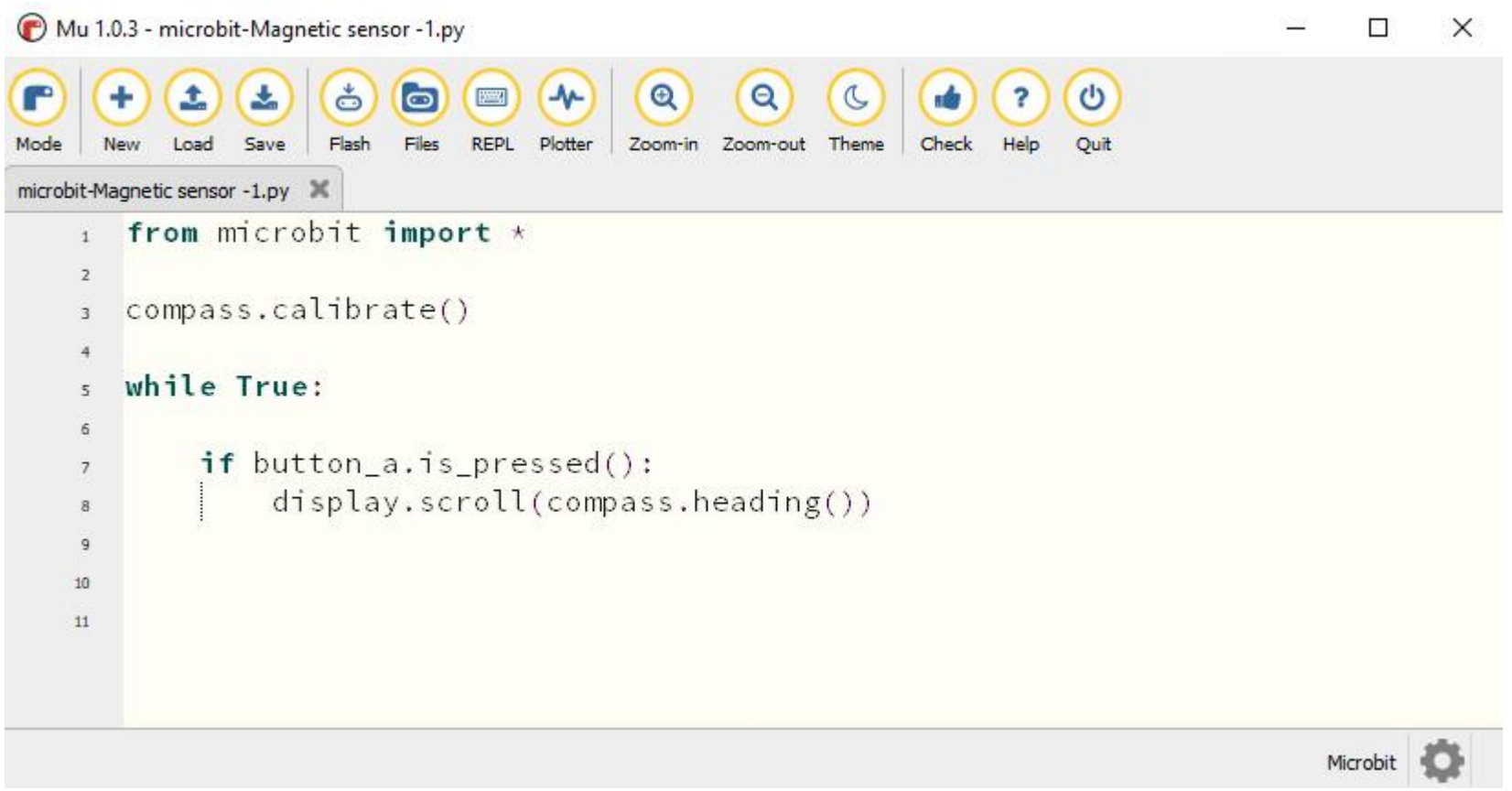
Test Code 1：

Enter Mu software and open the file “Project 6：Magnetic sensor -1.py “ to import code:

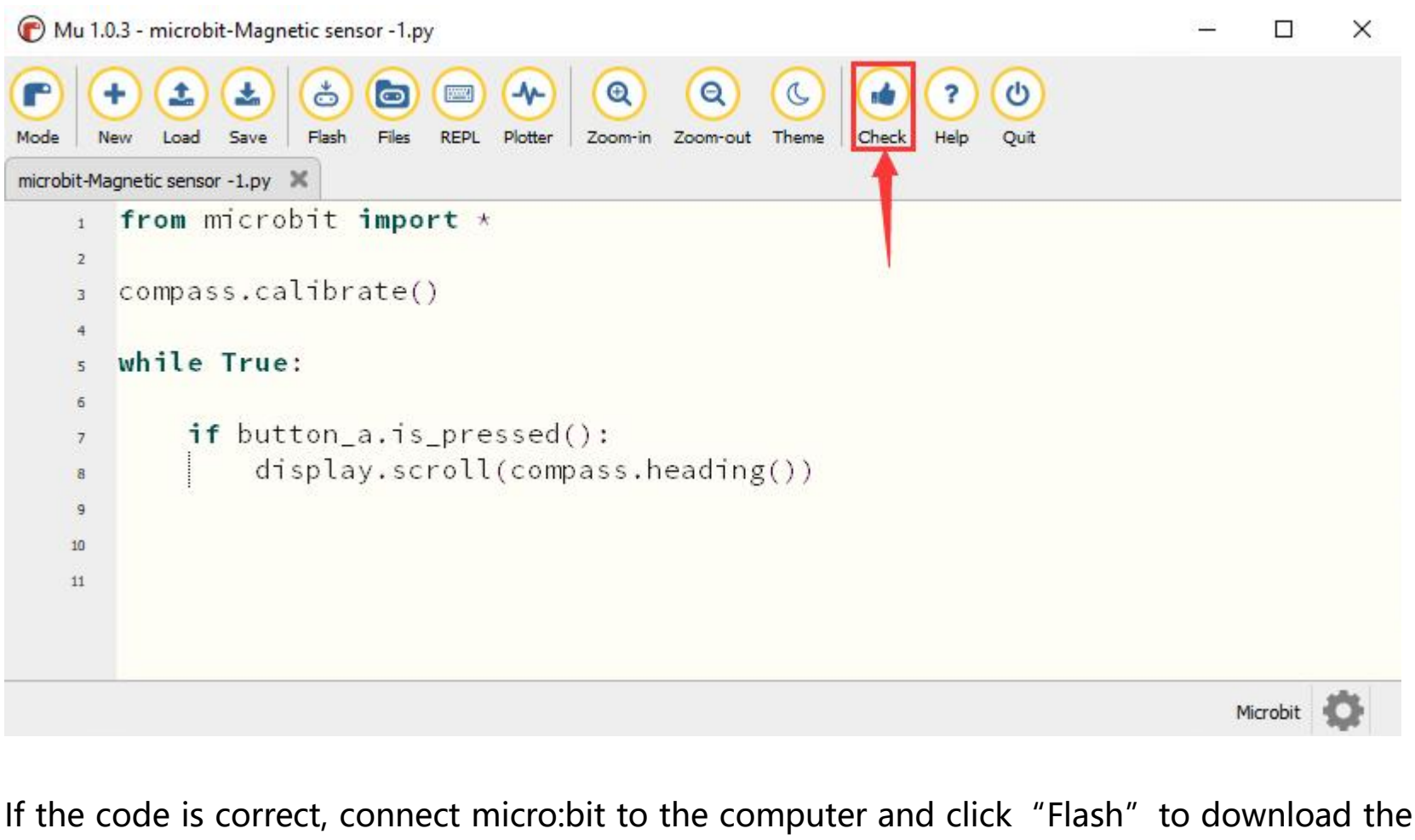
File Type	Route	File Name
Python file	../Python code/8.6：Magnetic sensor	microbit-Magnetic sensor -1.py

You can also input the code in the editing window yourself.

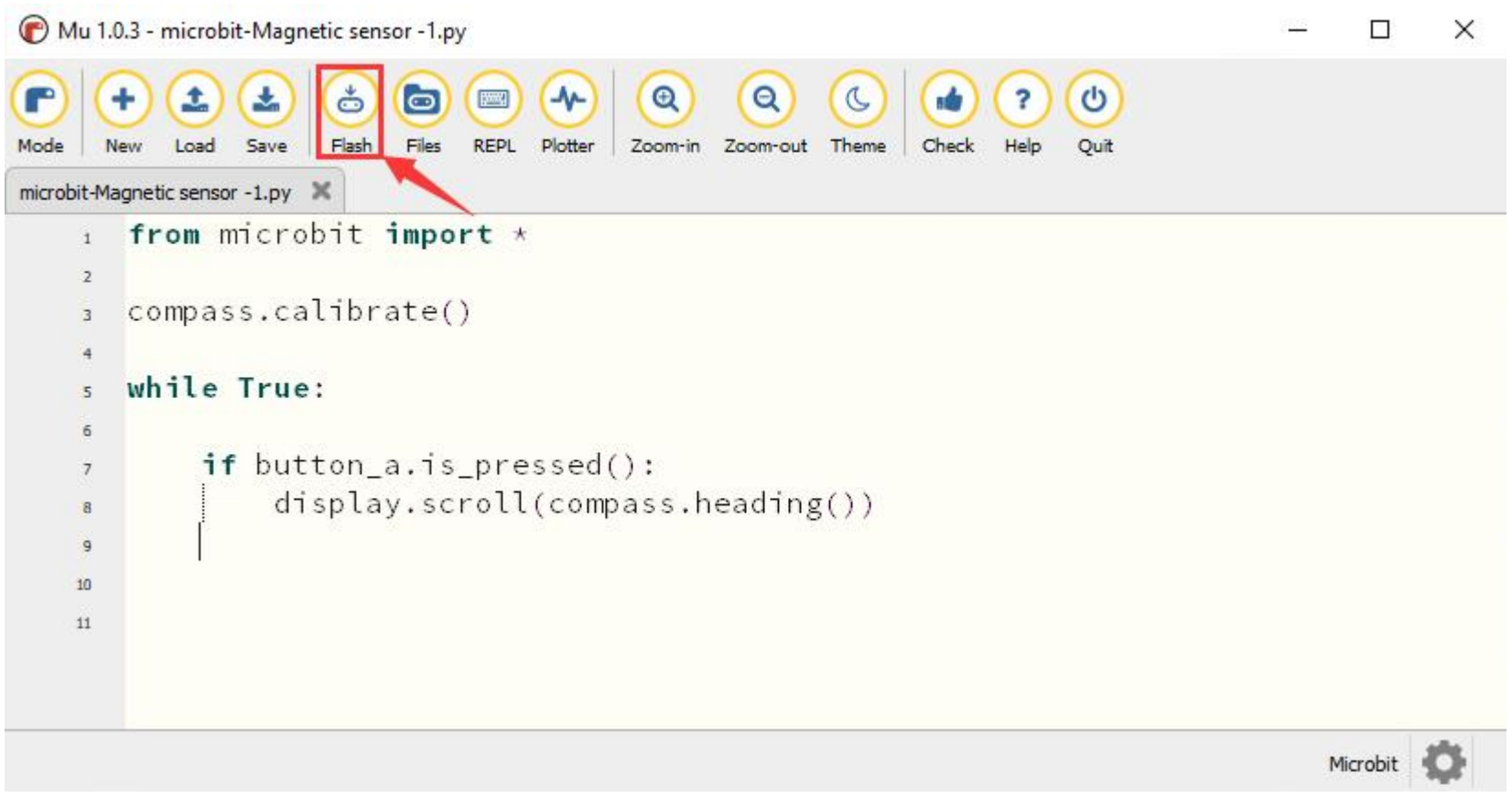
(Note: All words and symbols must be written in English.)



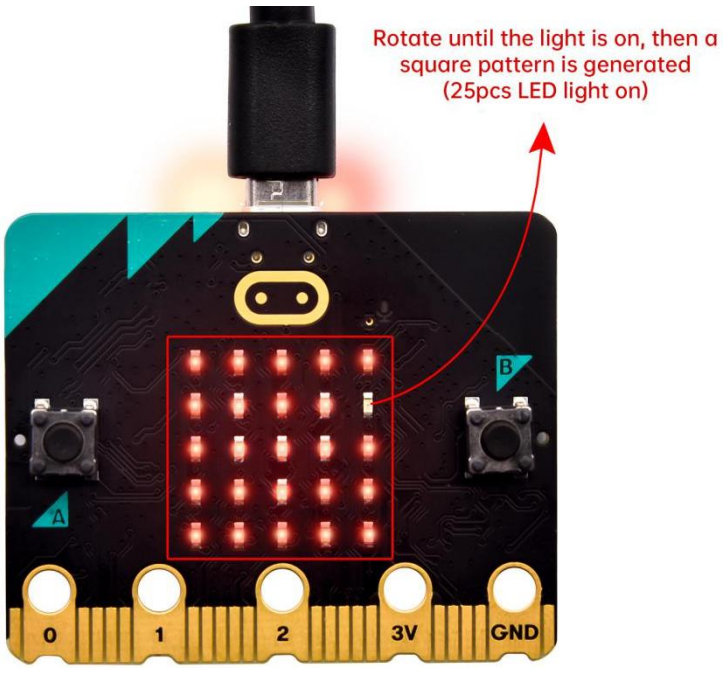
Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect micro:bit to the computer and click “Flash” to download the code to the micro:bit board.



After uploading the test code1 to the micro:bit main board and powering the board via an USB cable, and pressing the button A, the board asks us to calibrate the compass and the LED dot matrix shows "TILT TO FILL SCREEN". Then enter the calibration page. Rotate the board until all 25 red LEDs are on, as shown below.

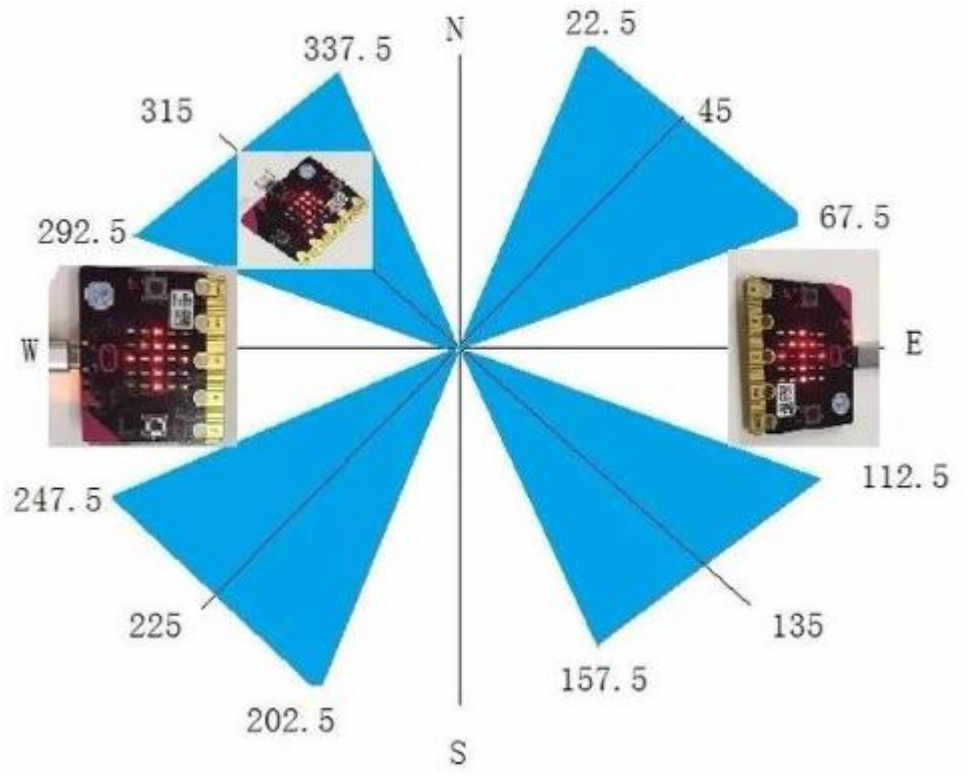


After that, a smile pattern  appears, which implies the calibration is done. When the calibration process is completed, pressing the button A will make the magnetometer reading display directly on the screen. And the direction north, east, south and west correspond to 0°, 90°, 180° and 270° respectively.

Test Code 2:

For the below picture, the arrow will work to point to the upper right when the value ranges from 292.5 to 337.5. Since 0.5 can't be input in the code, the values we get are 293 and 338.

Then add other statements to make a set of complete code.



Enter Mu software and open the file “Project 6：Magnetic sensor -2.py “ to import the code:

File Type	Route	File Name
Python file	../Python code/8.6：Magnetic sensor	microbit-Magnetic sensor -2.py

You can also input code in the editing window yourself.

(Note: All words and symbols must be written in English.)

Mu 1.0.3 - microbit-Magnetic sensor -2.py

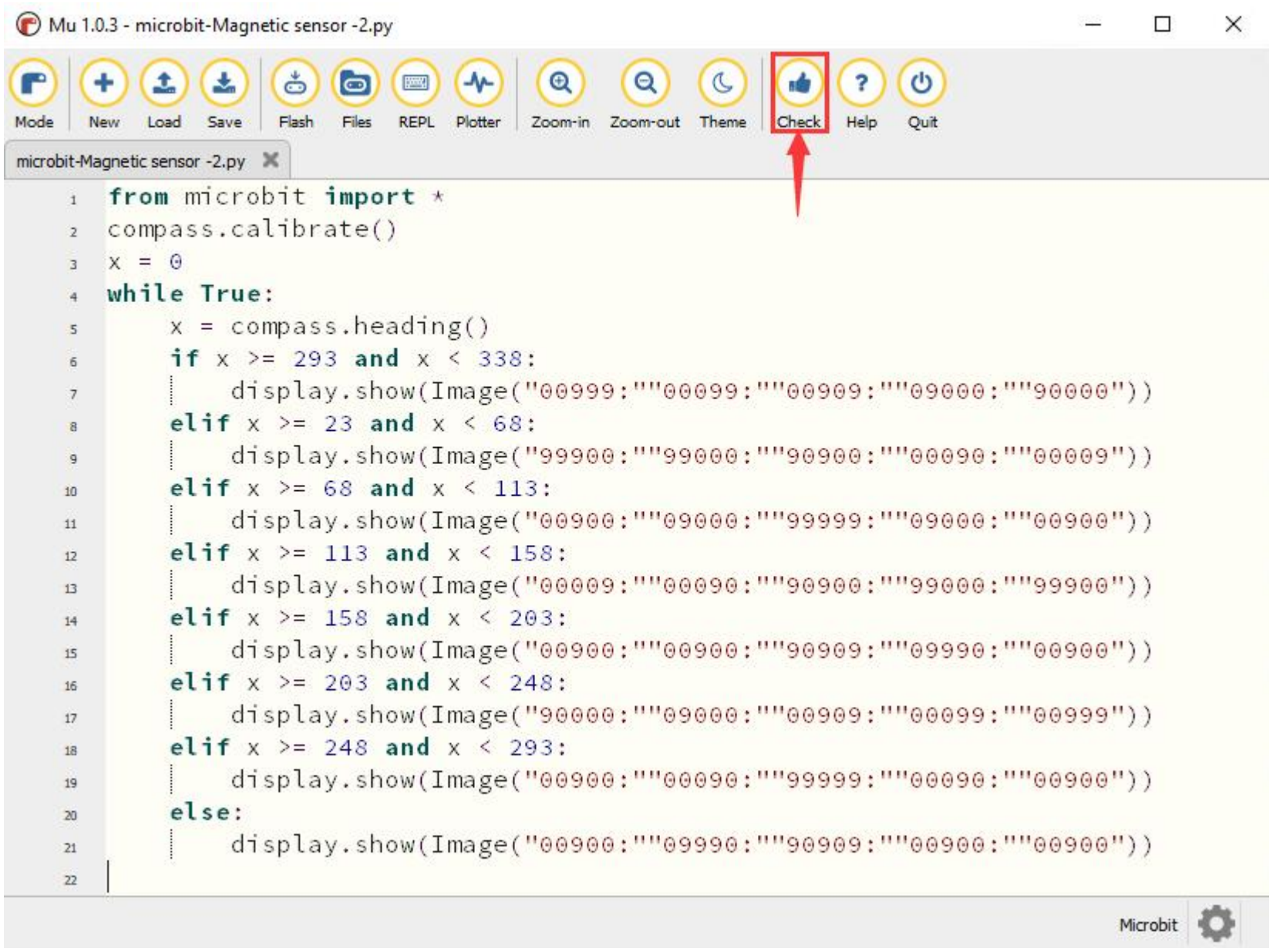
Mode | New | Load | Save | Flash | Files | REPL | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

microbit-Magnetic sensor -2.py

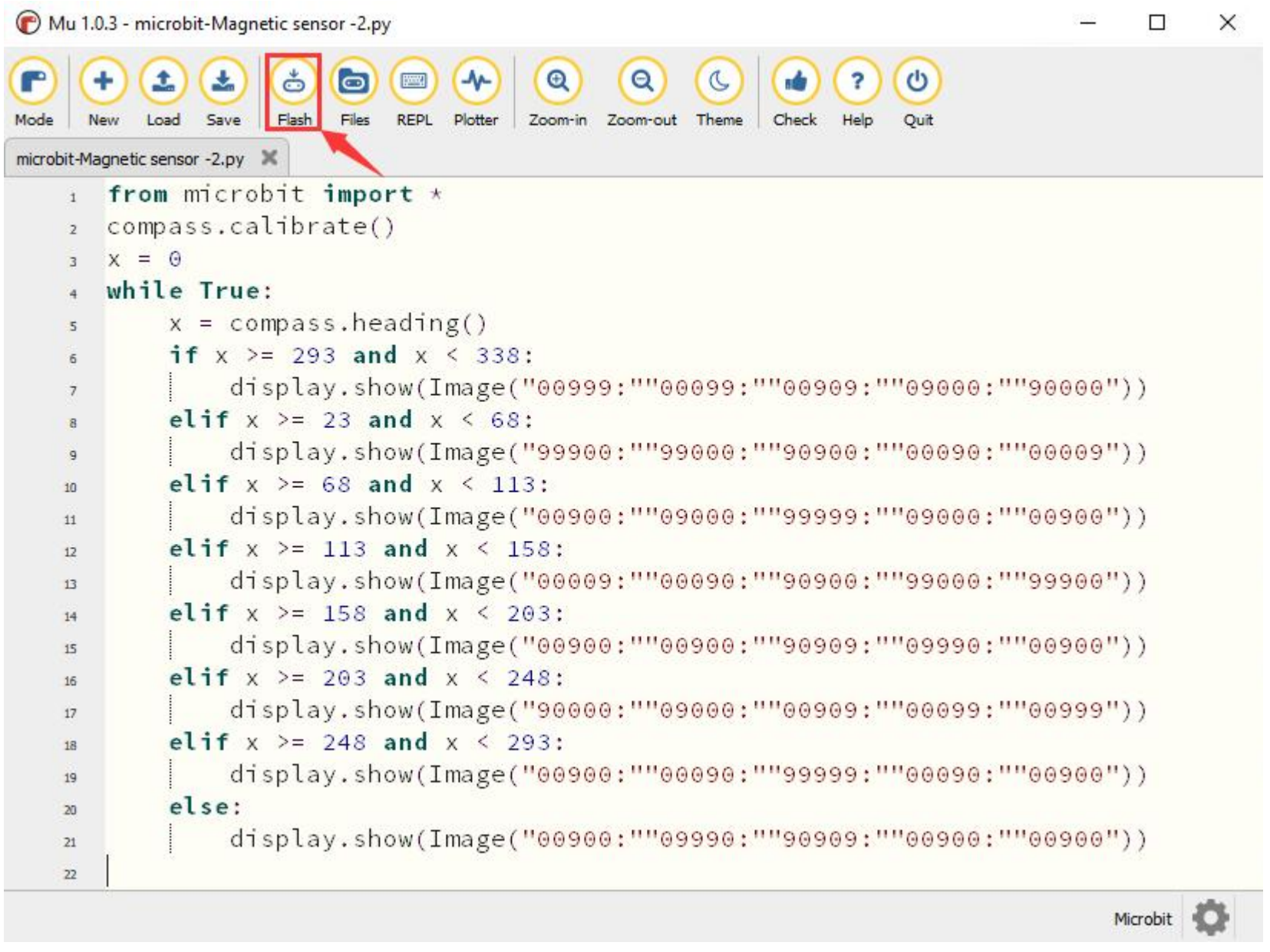
```
from microbit import *
compass.calibrate()
x = 0
while True:
    x = compass.heading()
    if x >= 293 and x < 338:
        display.show(Image("00999:""00099:""00909:""09000:""90000"))
    elif x >= 23 and x < 68:
        display.show(Image("99900:""99000:""90900:""00090:""00009"))
    elif x >= 68 and x < 113:
        display.show(Image("00900:""09000:""99999:""09000:""00900"))
    elif x >= 113 and x < 158:
        display.show(Image("00009:""00090:""90900:""99000:""99900"))
    elif x >= 158 and x < 203:
        display.show(Image("00900:""00900:""90909:""09990:""00900"))
    elif x >= 203 and x < 248:
        display.show(Image("90000:""09000:""00909:""00099:""00999"))
    elif x >= 248 and x < 293:
        display.show(Image("00900:""00090:""99999:""00090:""00900"))
    else:
        display.show(Image("00900:""09990:""90909:""00900:""00900"))

```

Microbit

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



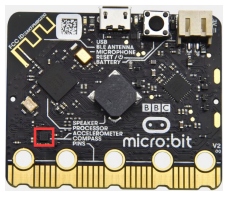
4. Test Result

Upload the code 2 and plug the micro:bit into power. After calibration, rotate the micro:bit board, then the LED dot matrix displays the direction signs.

5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
compass.calibrate()	Compass calibration
while True:	This is a permanent loop that makes micro:bit execute the code of it.
if button_a.is_pressed(): display.scroll(compass.heading())	When the button A is pressed, the micro:bit scrolls to show the value of compass
x = 0	Set variable x=0
x = compass.heading()	Set the value of compass to variable x
if...**elif**...**else**	Condition judgement statement:if...else if...else
display.show(Image("00999:""00099:""00909:""09000:""90000"))	Micro:bit shows the upper right arrow sign
display.show(Image("99900:""99000:""90900:""00090:""00009"))	Micro:bit shows the upper left arrow sign
display.show(Image("00900:""09000:""99999:""09000:""00900"))	Micro:bit shows the left arrow sign
display.show(Image("00009:""00090:""90900:""99000:""99900"))	Micro:bit shows the lower left arrow sign
display.show(Image("00900:""00900:""90909:""09990:""00900"))	Micro:bit shows the downward arrow sign
display.show(Image("90000:""09000:""00909:""00099:""00999"))	Micro:bit shows the lower right arrow sign
display.show(Image("00900:""00090:""99999:""00090:""00900"))	Micro:bit shows the right arrow sign
display.show(Image("00900:""09990:""90909:""00900:""00900"))	Micro:bit shows the upward arrow sign

Project 7：Accelerometer



1. Description

The micro: bit main board V2 has a built-in LSM303AGR gravity acceleration sensor, also known as accelerometer, with a resolution of 8/10/12 bits. The code section sets the range to 1g, 2g, 4g, and 8g.

We often use an accelerometer to detect the status of machines.

In this project, we will work to introduce how to measure the position of the board with the accelerometer. And then have a look at the original three-axis data output by the accelerometer.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

3. Test Code

Test Code 1：

Enter Mu software and open the file“Project 7：Three-axis acceleration sensor -1.py“ to import the code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.7：Three-axis acceleration sensor	microbit-Three-axis acceleration sensor -1.py

You can also input the code in the editing window yourself.

(Note: All words and symbols must be written in English.)

Mu 1.0.3 - microbit-Three-axis acceleration sensor -1.py

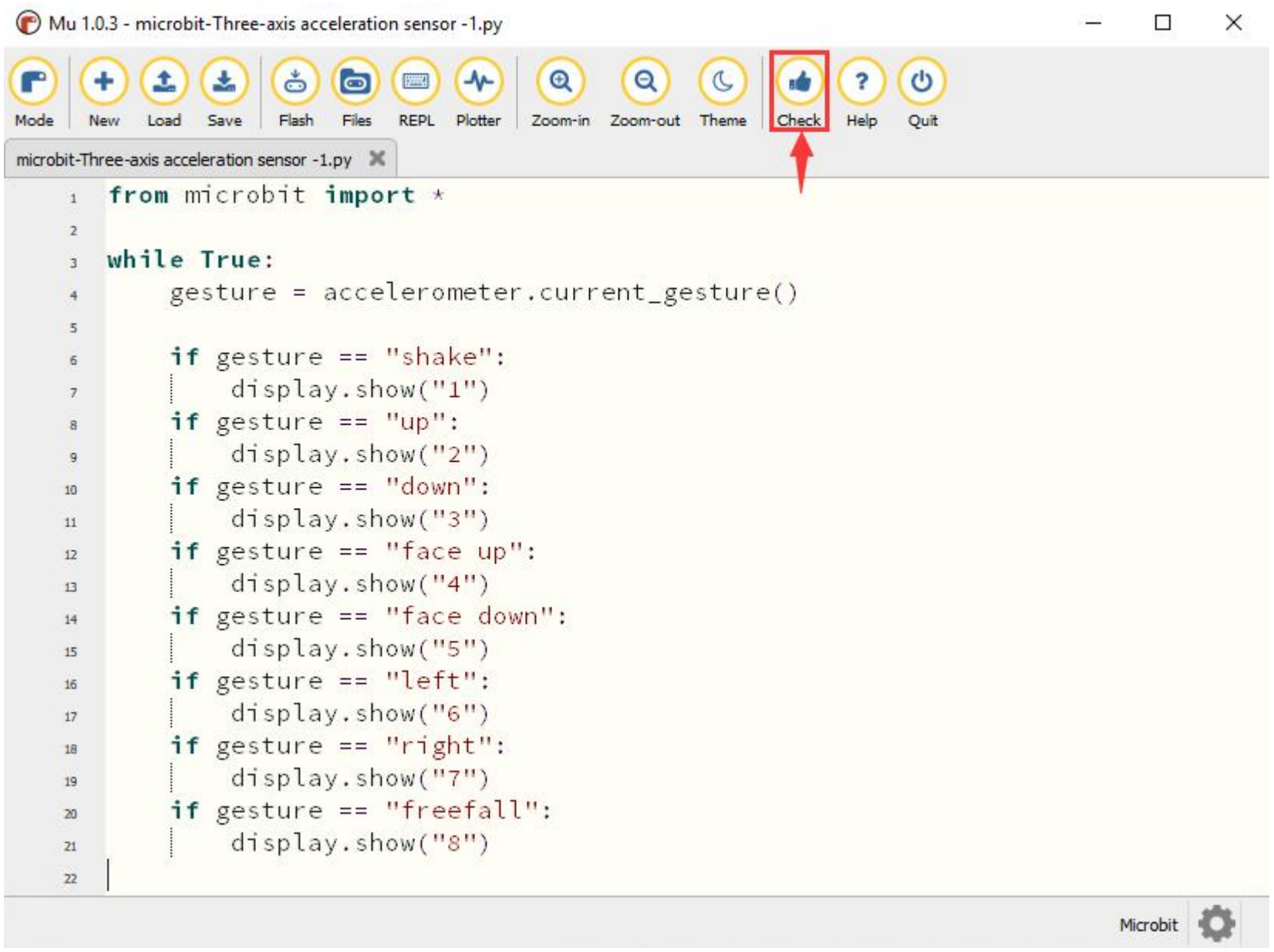
Mode | New | Load | Save | Flash | Files | REPL | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

microbit-Three-axis acceleration sensor -1.py

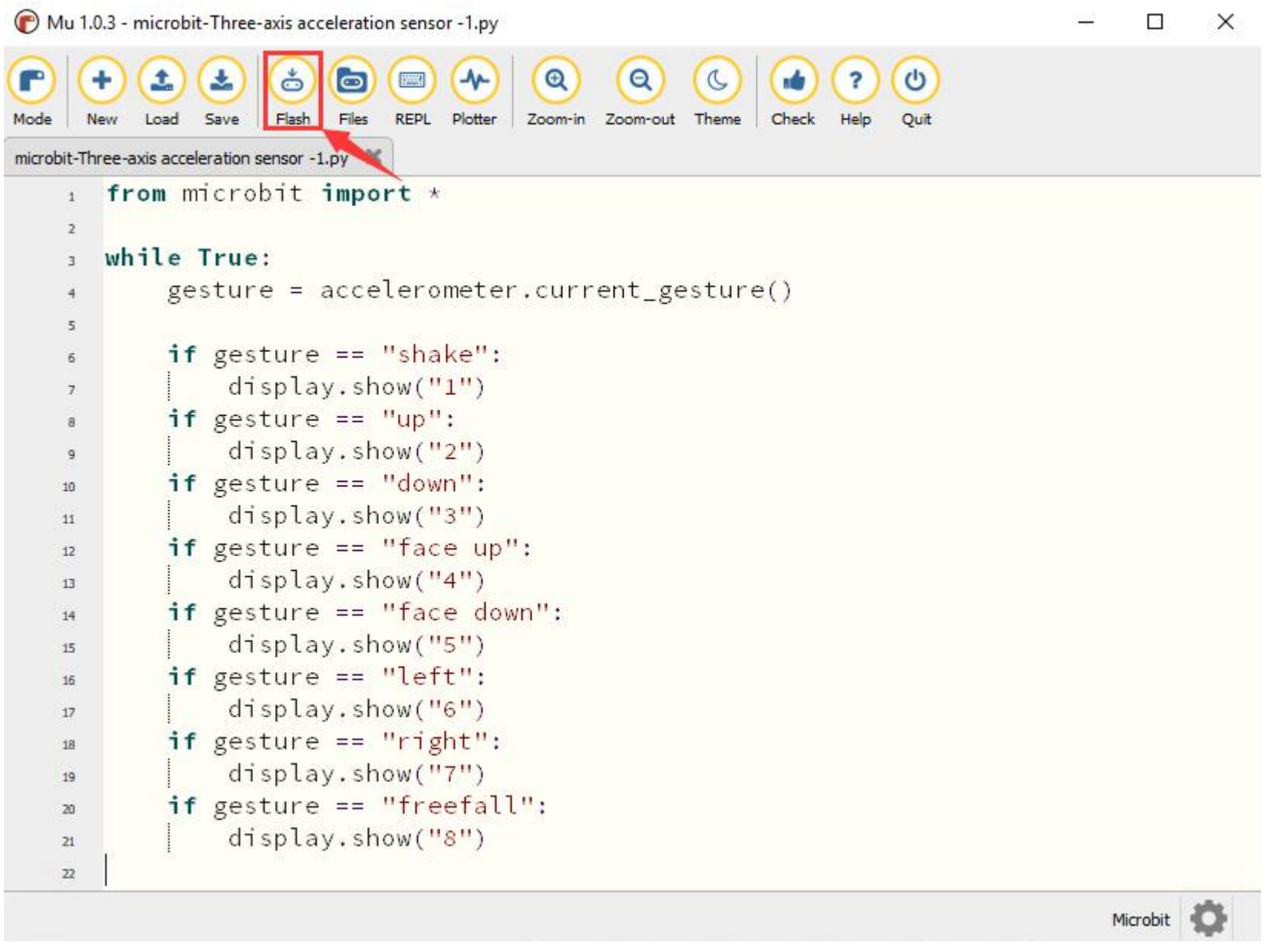
```
from microbit import *

while True:
    gesture = accelerometer.current_gesture()

    if gesture == "shake":
        display.show("1")
    if gesture == "up":
        display.show("2")
    if gesture == "down":
        display.show("3")
    if gesture == "face up":
        display.show("4")
    if gesture == "face down":
        display.show("5")
    if gesture == "left":
        display.show("6")
    if gesture == "right":
        display.show("7")
    if gesture == "freefall":
        display.show("8")

```

Microbit

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



Test Code 2：

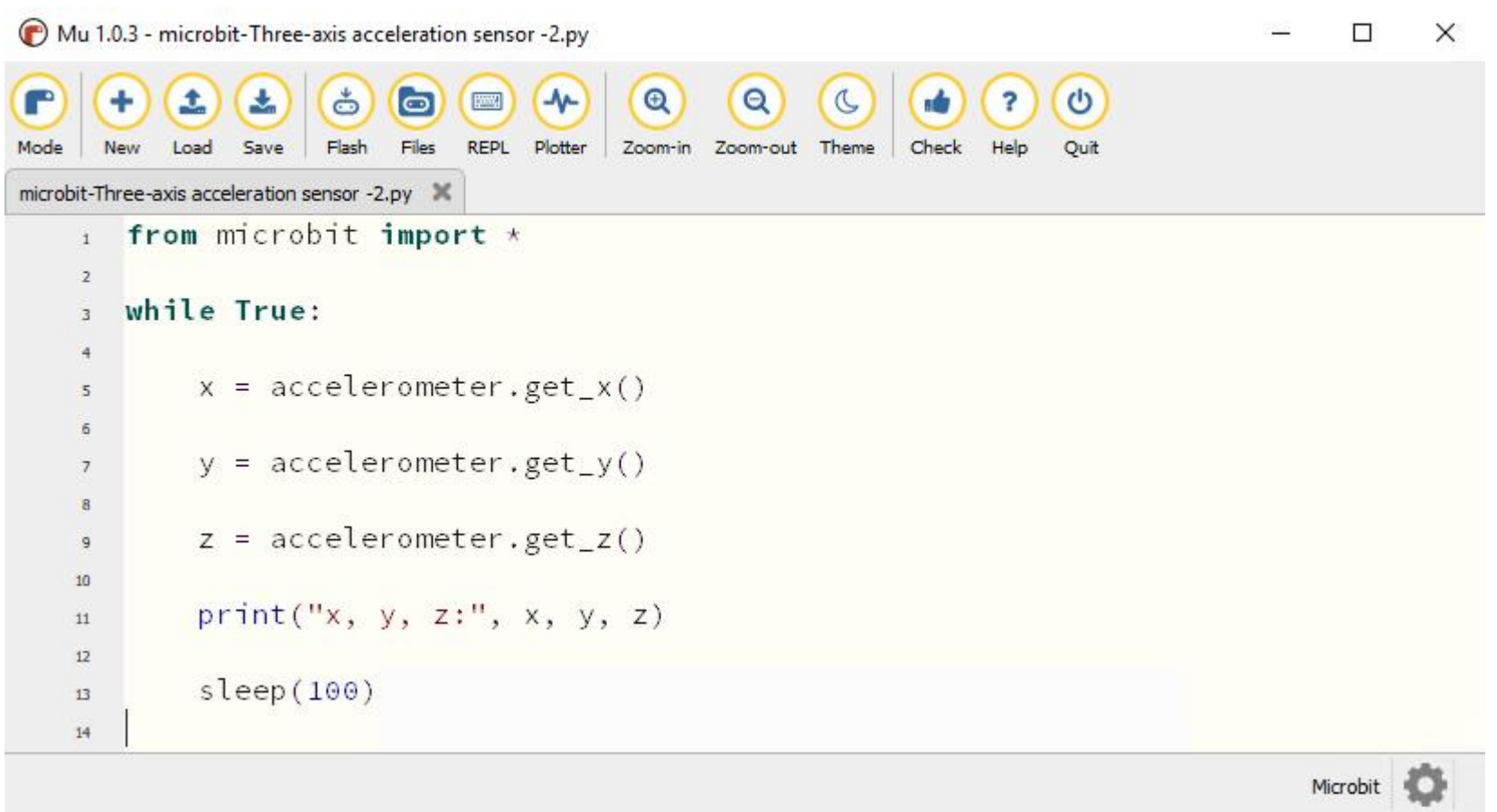
Enter Mu software and open the file “Project 7：Three-axis acceleration sensor -2.py “ to import the code:

(How to load the project code?)

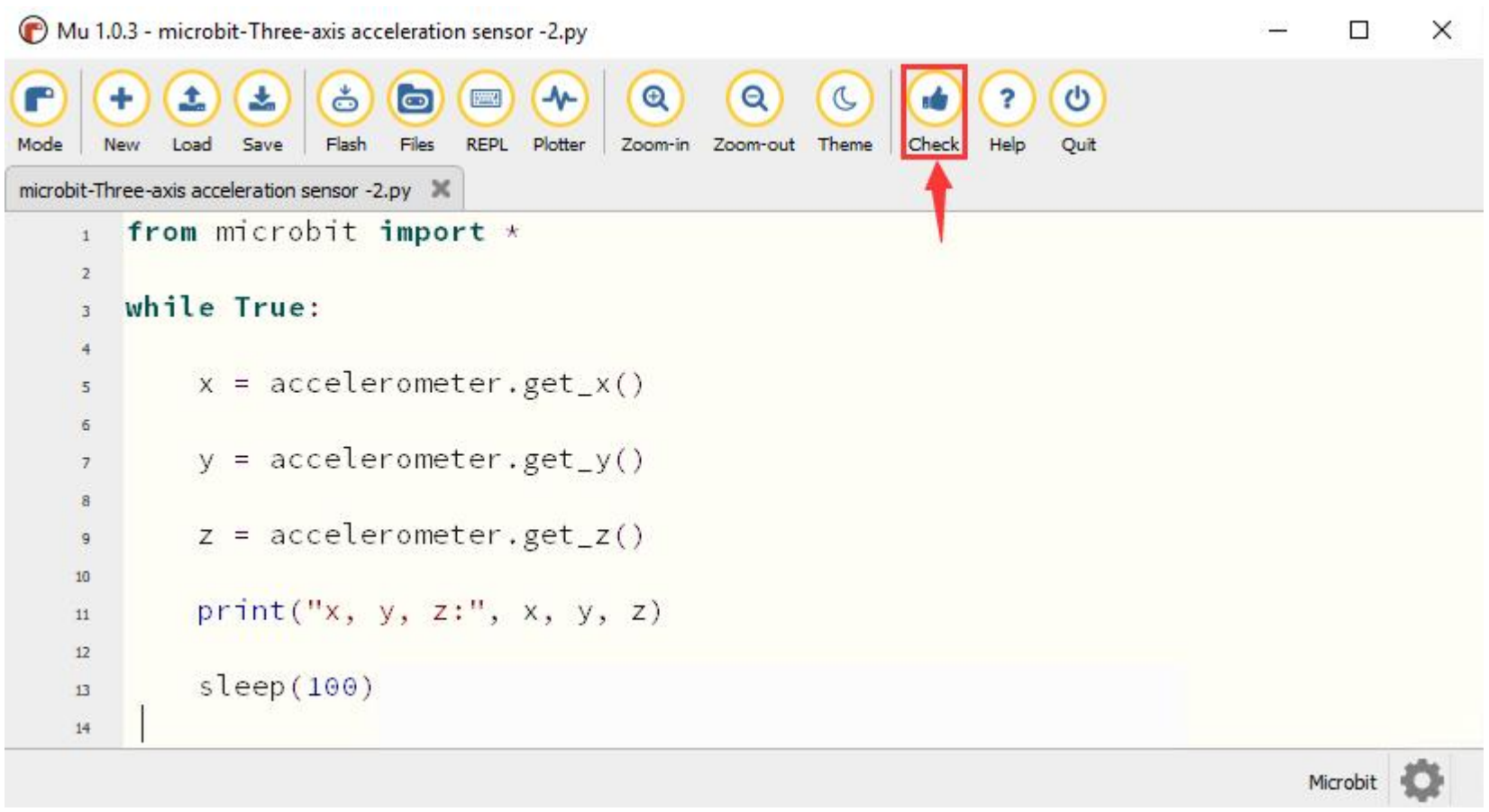
File Type	Route	File Name
Python file	../Python code/8.7：Three-axis acceleration sensor	microbit-Three-axis acceleration sensor -2.py

You can also input the code in the editing window yourself.

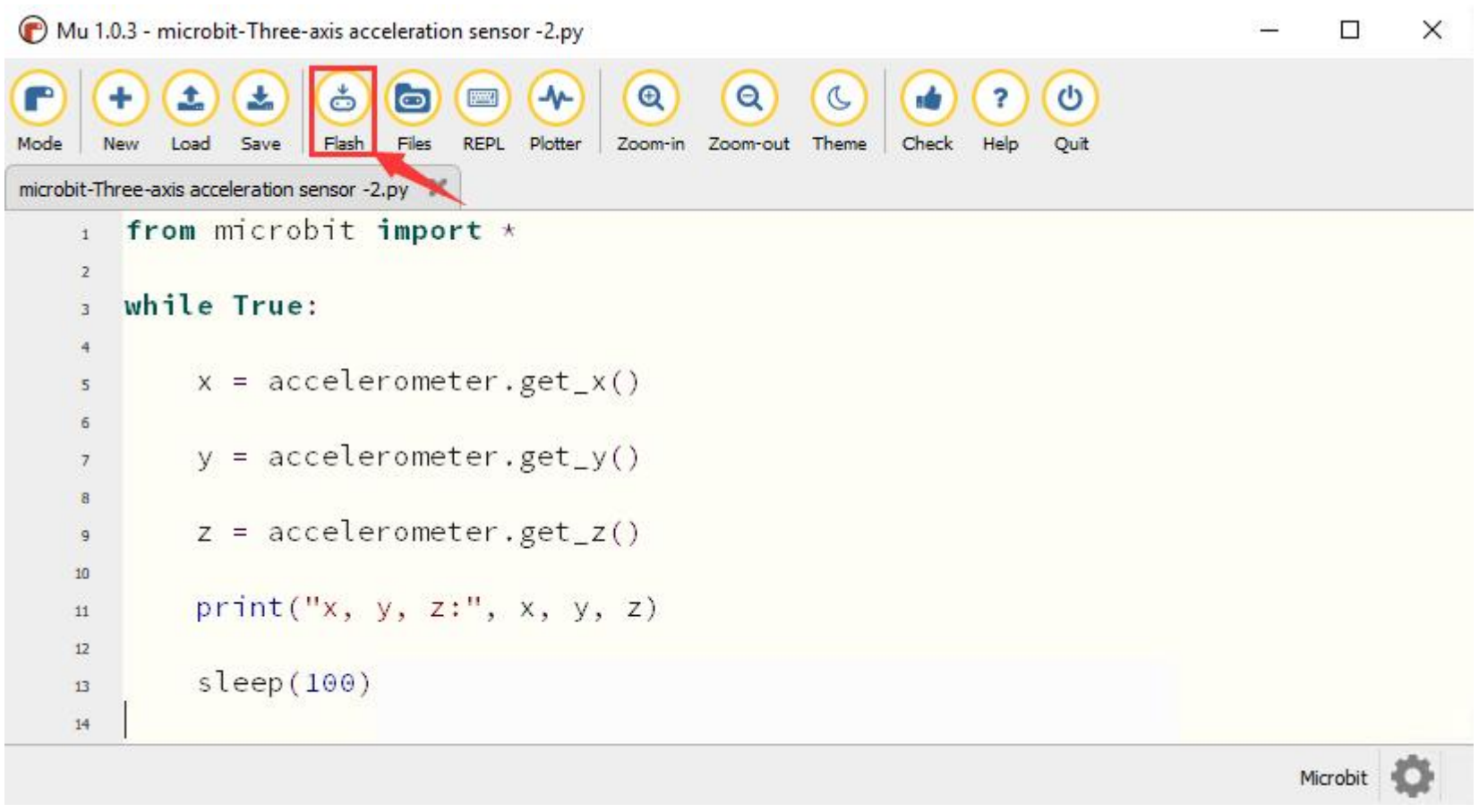
(Note: All words and symbols must be written in English.)



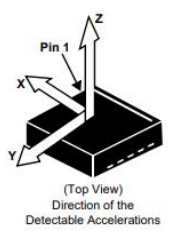
Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.

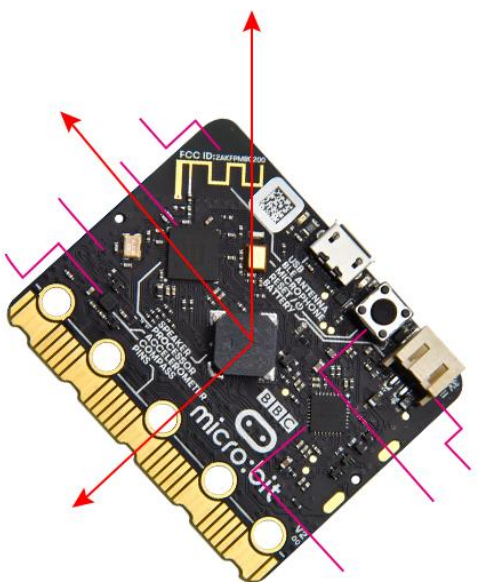


If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.

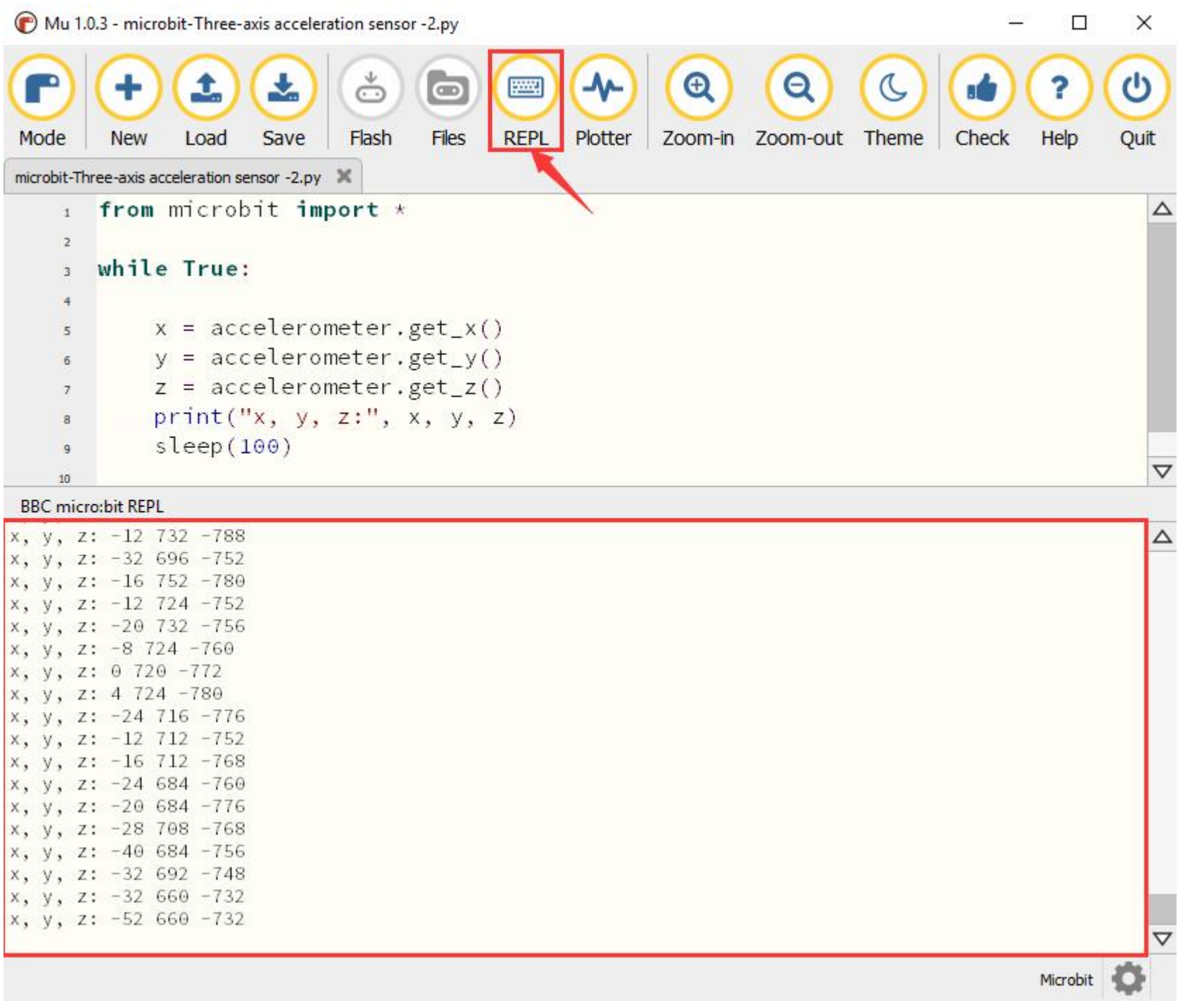


After referring to the MMA8653FC data manual and the hardware schematic diagram of the micro: bit main board, the accelerometer coordinate of the micro: bit is shown in the figure below:





Upload the test code 2 to the micro:bit main board and power the board via the USB cable. Click “REPL” and press the reset button. The value of the acceleration on X axis, Y axis and Z axis are shown below:



4. Test Result

After uploading the test code 1 to the micro:bit main board and powering the board via the USB cable, when we shake the micro: bit main board，no matter at any direction, the LED dot matrix displays the digit “1”.

When it is kept upright（make its logo above the LED dot matrix），the number 2 appears.

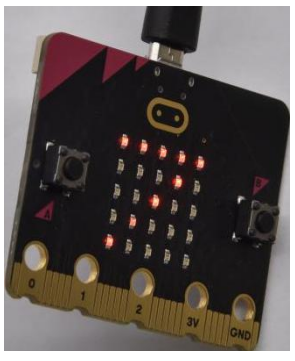
When it is kept upside down(make its logo below the LED dot matrix) , it shows as below.

When it is placed still on the desk, showing its front side, the number 4 appears.

When it is placed still on the desk, showing its back side, the number 5 exhibits.

When the board is tilted to the left , the LED dot matrix shows the number 6, as shown below:

When the board is tilted to the right , the LED dot matrix displays the number 7, as shown below:



When the board is knocked to the floor, this process can be considered as a free fall and the LED dot matrix shows the number 8. (Please note that this test is not recommended for it may damage the main board.)

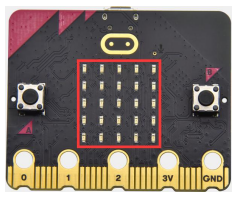
Attention: If you’ d like to try this function, you can also set the acceleration to 3g, 6g or 8g.

5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
gesture = accelerometer.current_gesture()	Set accelerometer.current_gesture() to gesture

while True:	This is a permanent loop that makes micro:bit execute the code of it.
if gesture == "shake": display.show("1") **if** gesture == "up": display.show("2") **if** gesture == "down": display.show("3") **if** gesture == "face up": display.show("4") **if** gesture == "face down": display.show("5") **if** gesture == "left": display.show("6") **if** gesture == "right": display.show("7") **if** gesture == "freefall": display.show("8")	Shaking micro:bit board, number 1 will appear When log points to the North, number 2 will show up. When log points to the South, number 3 will be shown When the LED dot matrix is upward, the number 4 is shown. the number 5 is displayed when the LED dot matrix is downward. When Micro:bit board is tilt to the left, number 6 is shown. When micro:bit is tilt to the right When Micro:bit board is inclined to the right, number 7 is displayed. When it is free fall(accidentally making it fall), number 8 appears on dot matrix.
x = accelerometer.get_x() y = accelerometer.get_y() z = accelerometer.get_z()	Read the acceleration value on x axis，the return value is integer, and set x= the read value on x axis Read the acceleration value on y axis，the return value is integer, and set y= the read value on y axis Read the acceleration value on z axis，the return value is integer, and set z= the read value on z axis
print("x, y, z:", x, y, z)	The value of acceleration will be shown
sleep(100)	Delay in 100ms

Project 8：Light Detection



1. Description

In this project, we will focus on the light detection function of the Micro: Bit main board. It is achieved by the LED dot matrix since the main board is not equipped with a photoresistor.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

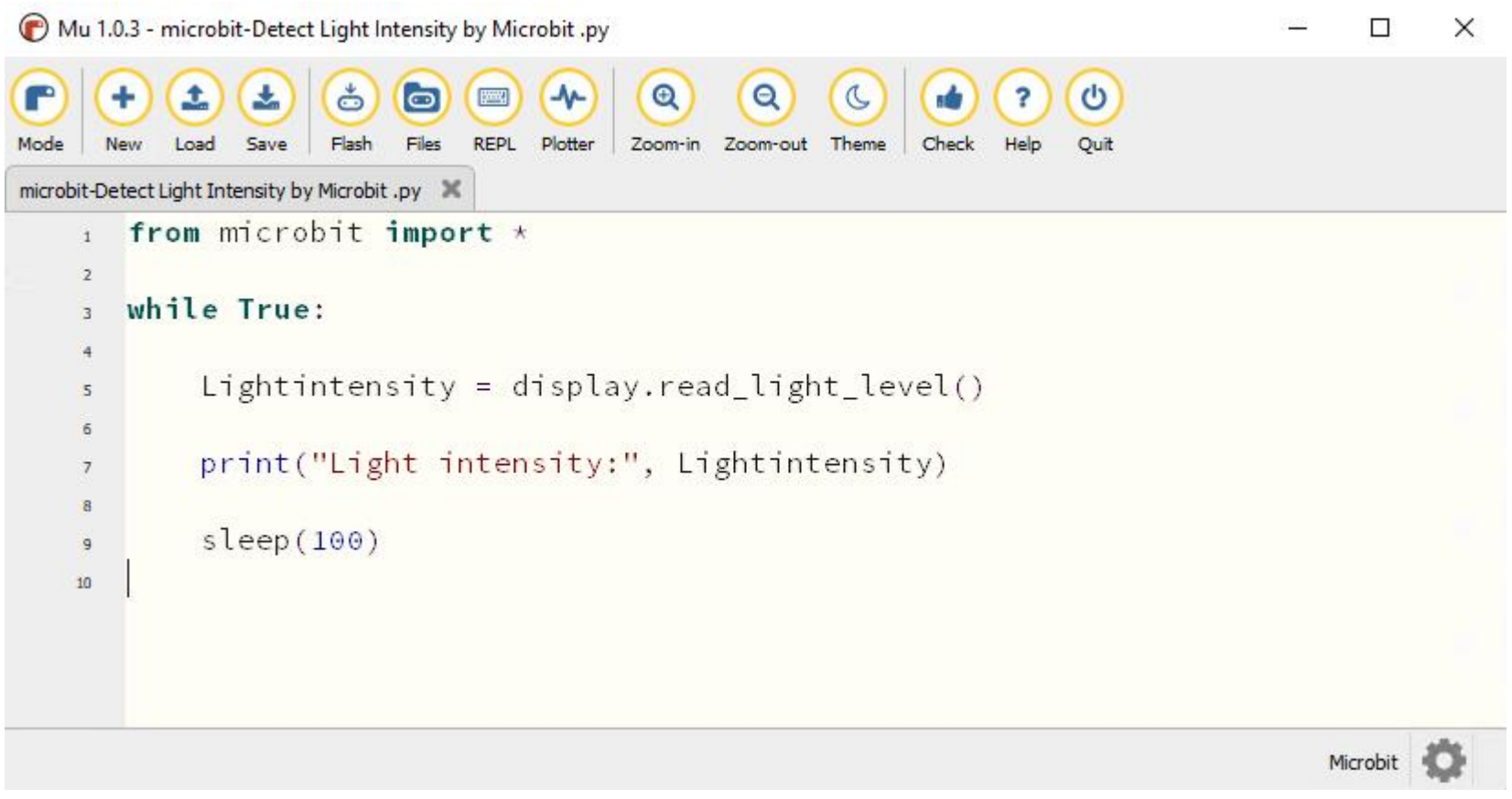
3. Test Code

Enter Mu software and open the file "project 8：Light Detection.py" to import the code

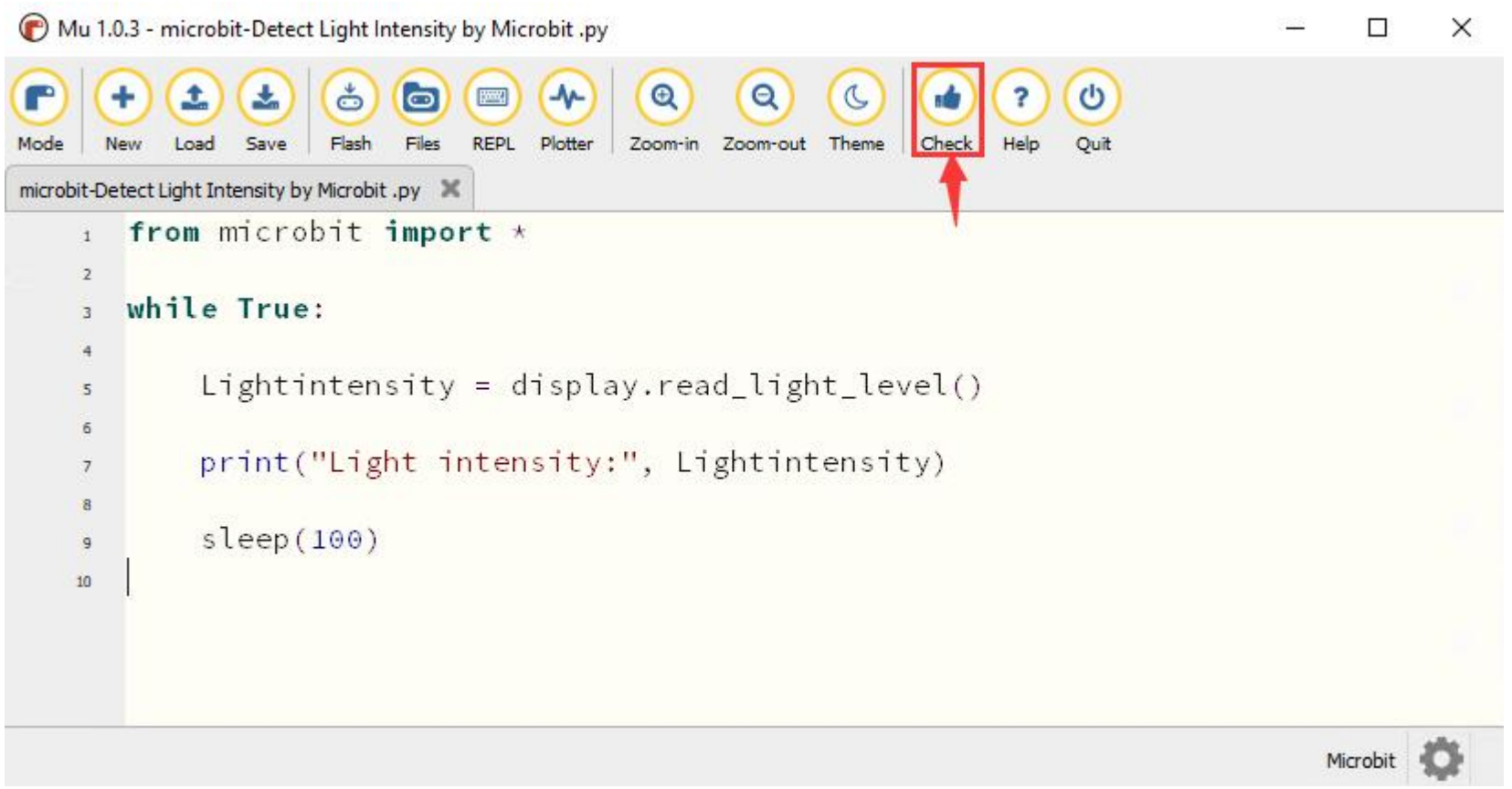
File Type	Route	File Name
Python file	../Python code/8.8：Detect Light Intensity by Micro:bit	microbit-Detect Light Intensity by Micro:bit .py

You can also input code in the edit window yourself.

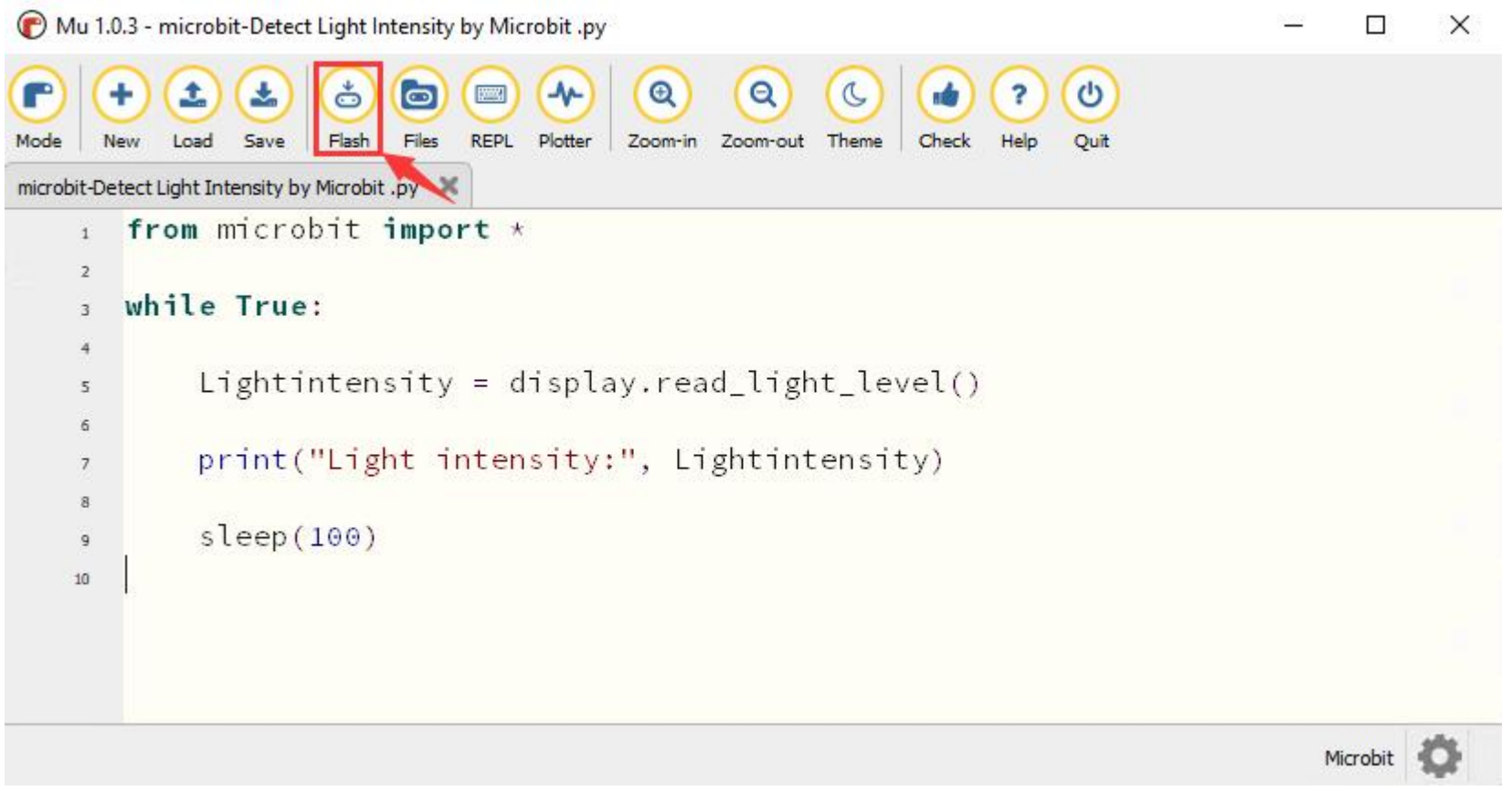
(Note: All English words and symbols must be written in English.)



Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.



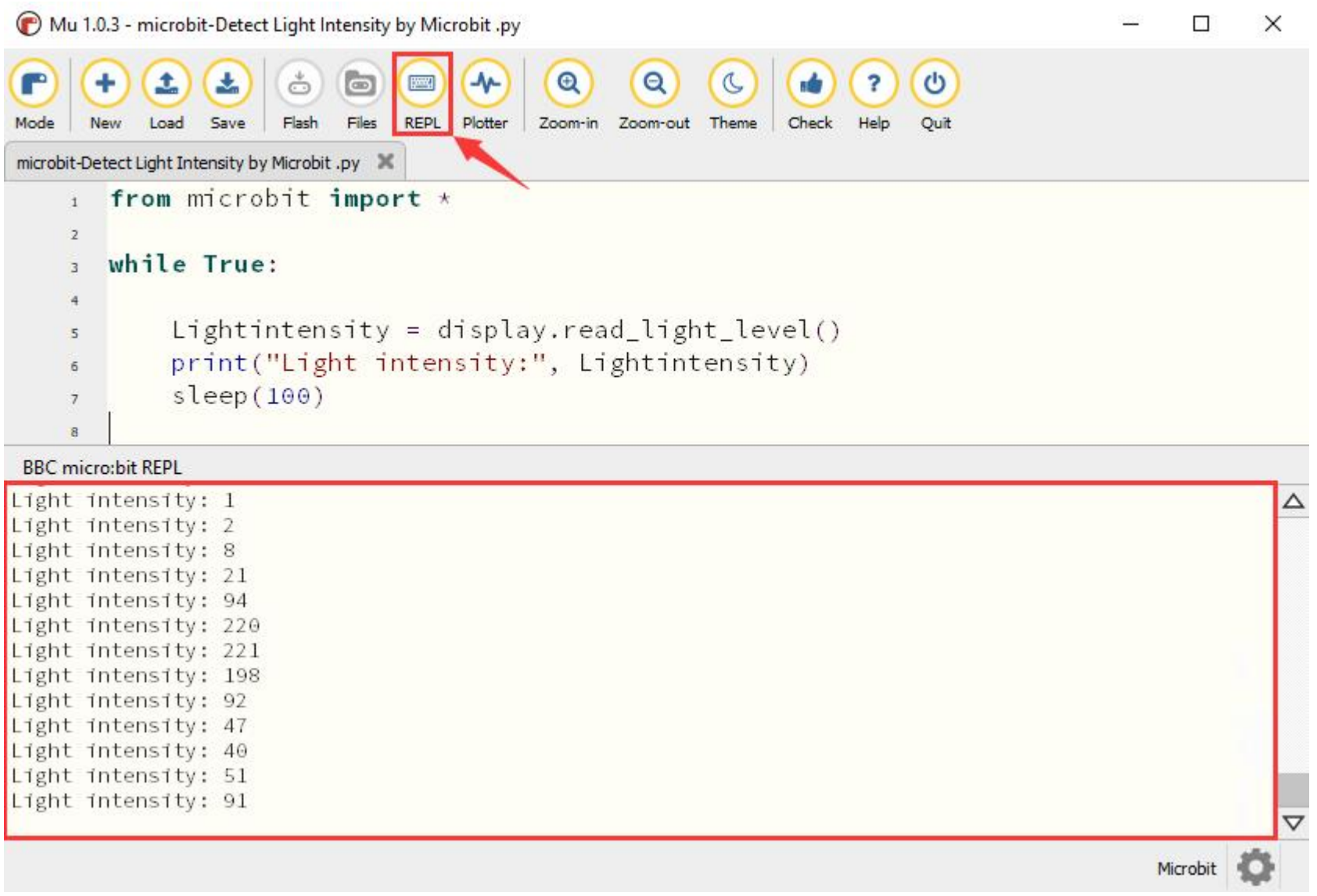
If the code is correct, connect the micro:bit to your computer and click "Flash" to download code to the micro:bit board.



4. Test Result

Upload the test code to the micro:bit main board and power it via the USB cable don't plug off it. Click "REPL" and press the reset buttons, the light intensity value will be displayed, as shown below.

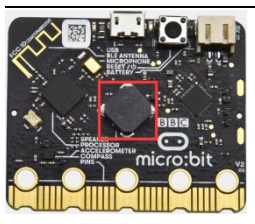
When the LED dot matrix is covered by hand, the light intensity showed is approximately 0; when the LED dot matrix is exposed to light, the light intensity displayed gets stronger with the light.



5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
gesture = accelerometer.current_gesture()	Set accelerometer.current_gesture() to gesture
while True:	This is a permanent loop that makes micro:bit execute the code of it.
Lightintensity = display.read_light_level()	Set display.read_light_level() to Lightintensity
print("Light intensity:", Lightintensity)	BBC microbit REPL prints the detected light intensity value
sleep(100)	Delay in 100ms

Project 9：Speaker



1. Description

Micro: Bit main board has an built-in speaker, which makes adding sound to the programs easier. It can also be programmed to produce all kinds of tones, like playing the song *Ode to Joy.*

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

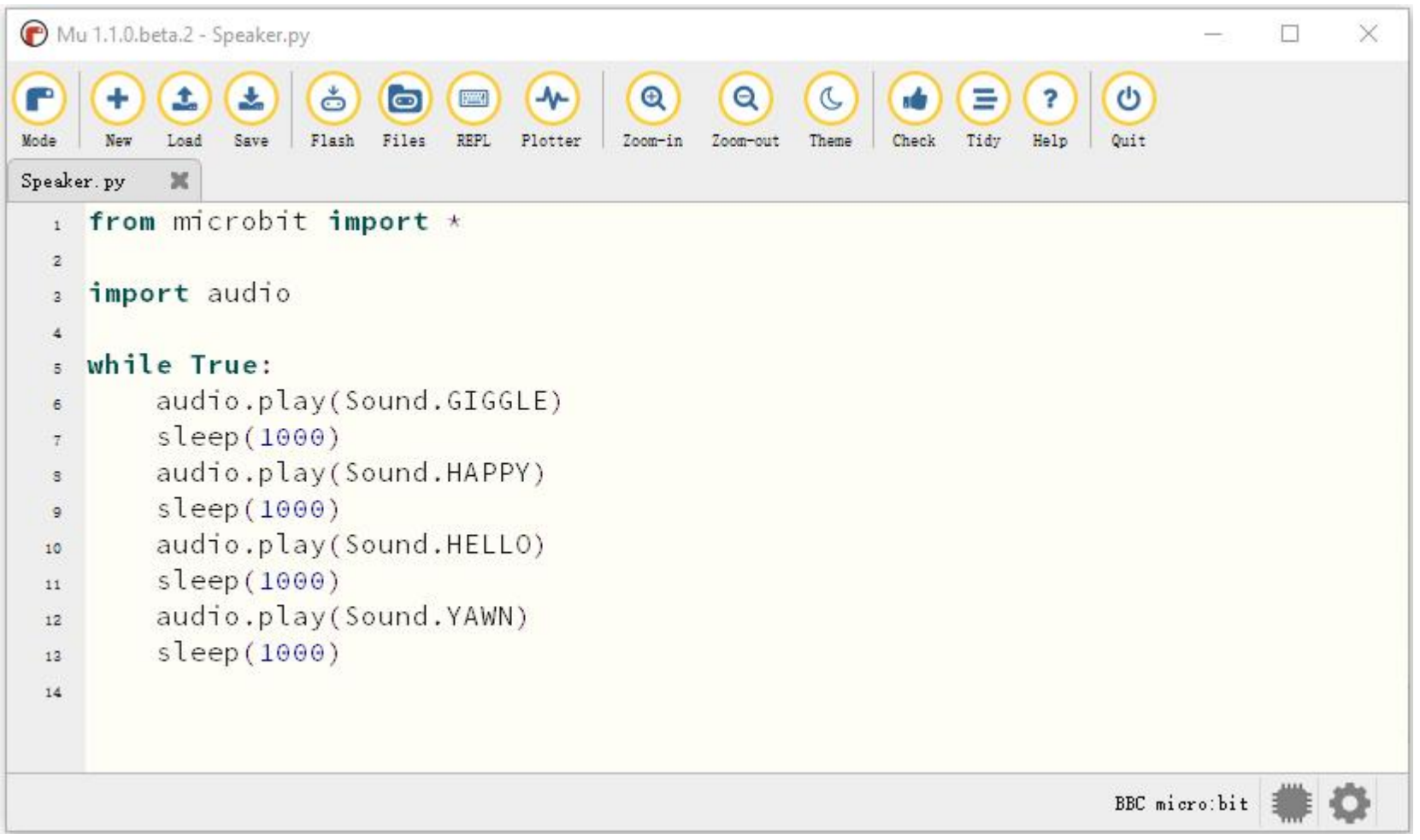
3. Test Code

Enter Mu software and open the file "Project 9：Speaker.py" to import code: (How to load the project code?)

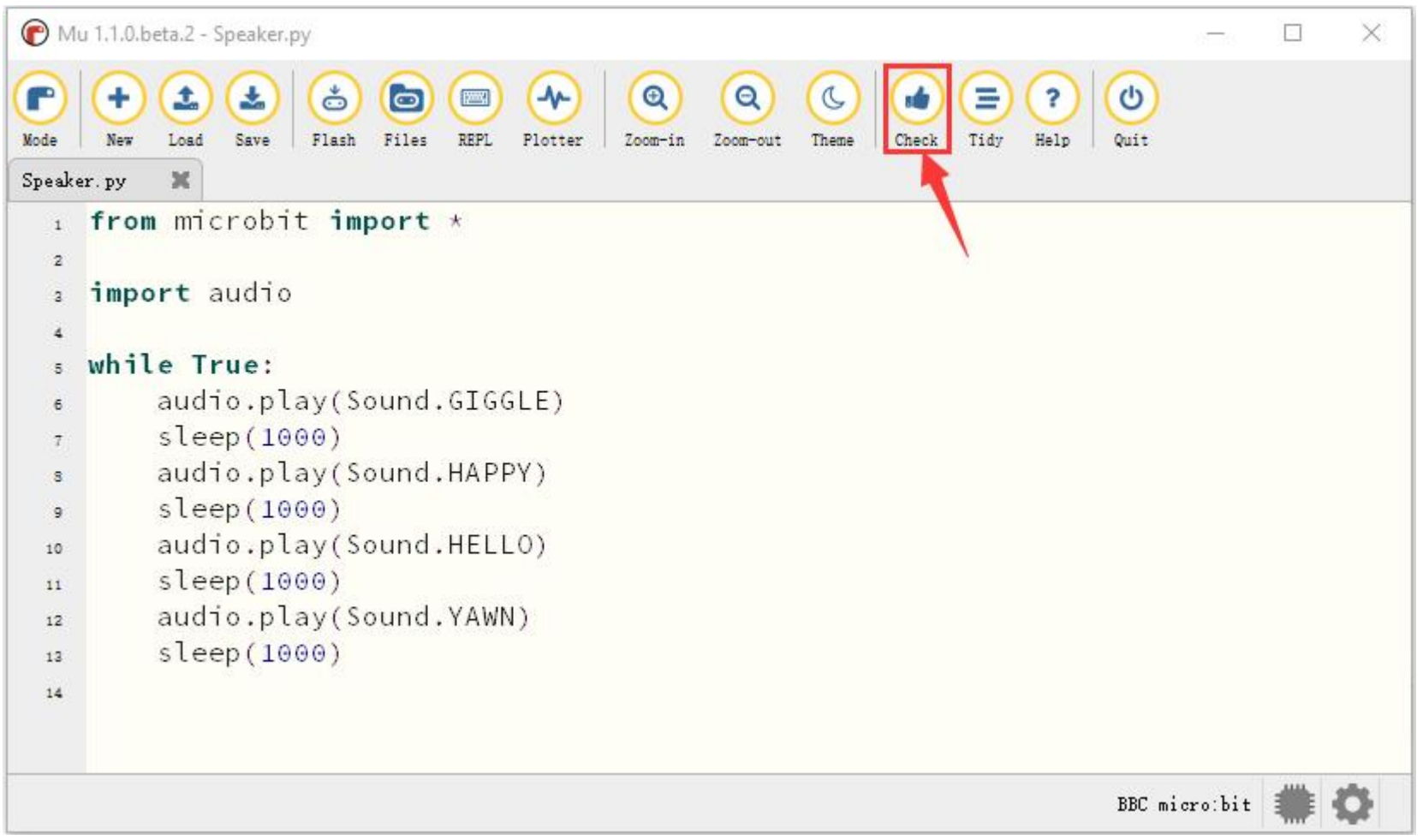
File Type	Route	File Name
Python file	../Python code/8.9：Speaker	microbit-Speaker.py

You can also input code in the editing window yourself.

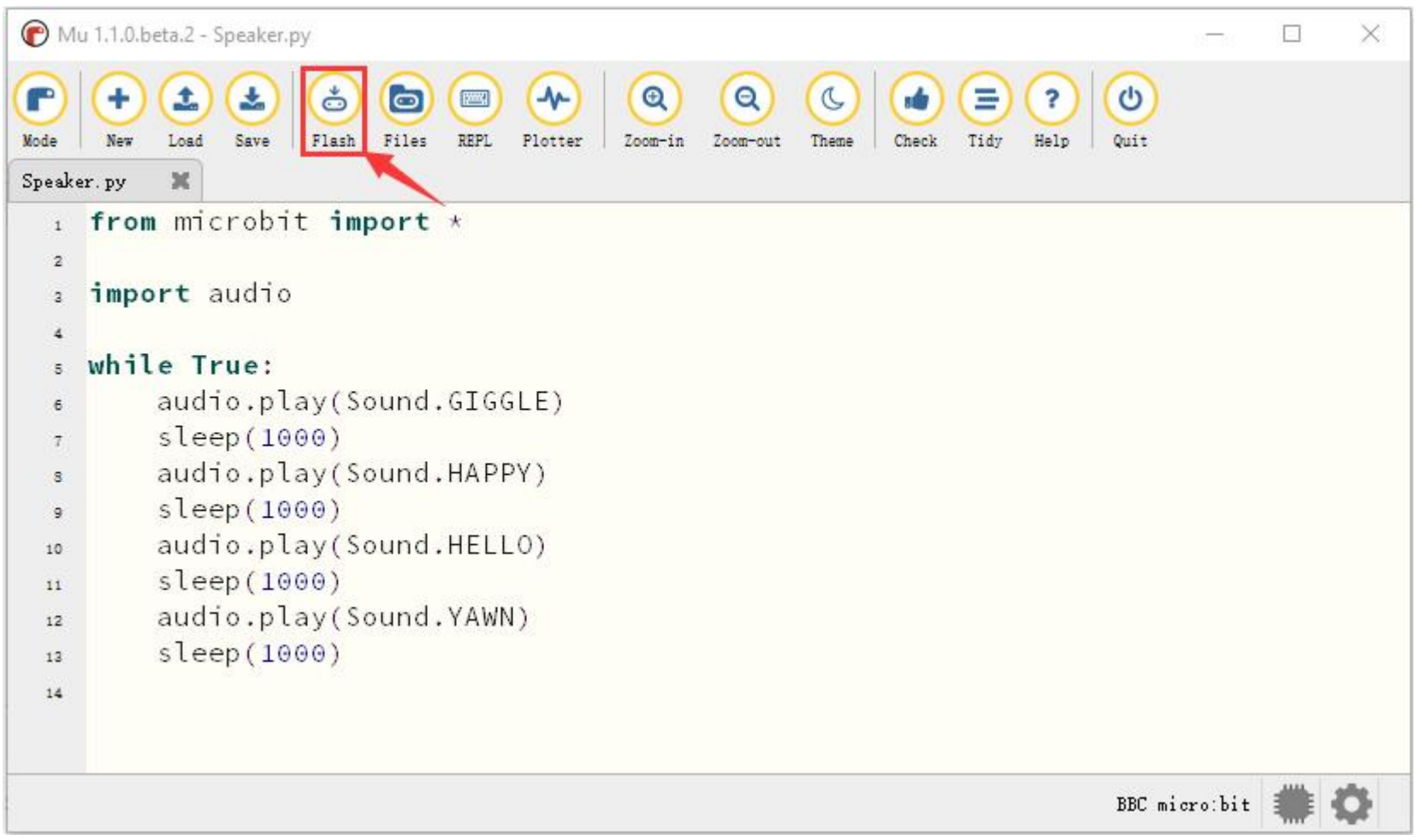
(Note: All words and symbols must be written in English.)



Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



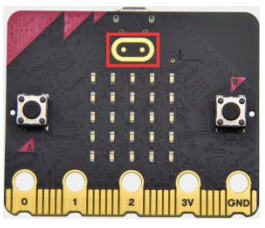
4. Test Result

After uploading the test code to the micro:bit main board and powering it via the USB cable, the speaker utters sound and the LED dot matrix shows the logo of music.

5. Code Explanation

from microbit **import** *	Import the library of micro：bit
import audio	Audio library
while True:	This is a permanent loop that makes micro:bit execute the code of it.
audio.play(Sound.GIGGLE)	Emit the “giggle” sound
sleep(1000)	delay in 1000ms

Project 10: Touch-sensitive Logo



1. Description

The Micro: Bit main board V2 is equipped with a golden touch-sensitive logo, which can act as an input component like an button.

It contains a capacitive touch sensor that senses small changes in the electric field when pressed (or touched), just like your phone or tablet screen. When you press it , the program can be activated.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

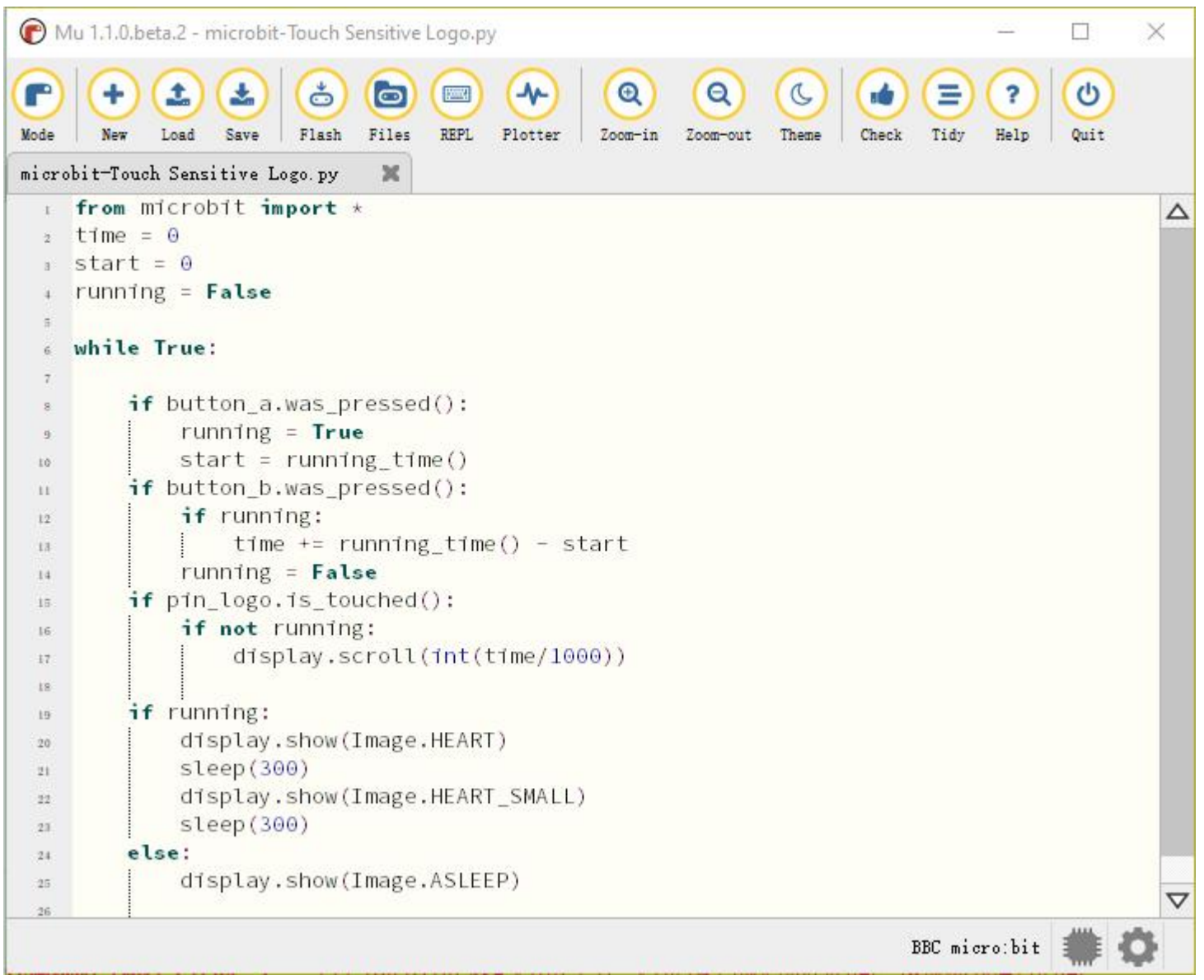
3. Test Code

Enter Mu software and open the file “Project 10：Touch-sensitive Logo.py” to import code: (How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.10：Touch Sensitive Logo	microbit-Touch Sensitive Logo.py

You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)



How Micro:bit works?

A. The runtime is recorded in milliseconds(ms) .

B. When you press button A, a variable named start will be set to the current running time.

C. When you press button B, the start time will be subtracted from the new running time to calculate the passed time since you started the stopwatch. This difference is added to the total time, which is stored in a variable named time.

D. If you press the golden logo, the program will display the total elapsed time on the LED display. It converts time from milliseconds (thousandths of a second) to seconds by dividing by 1000. It uses the integer division operator to give an integer (integer) result.

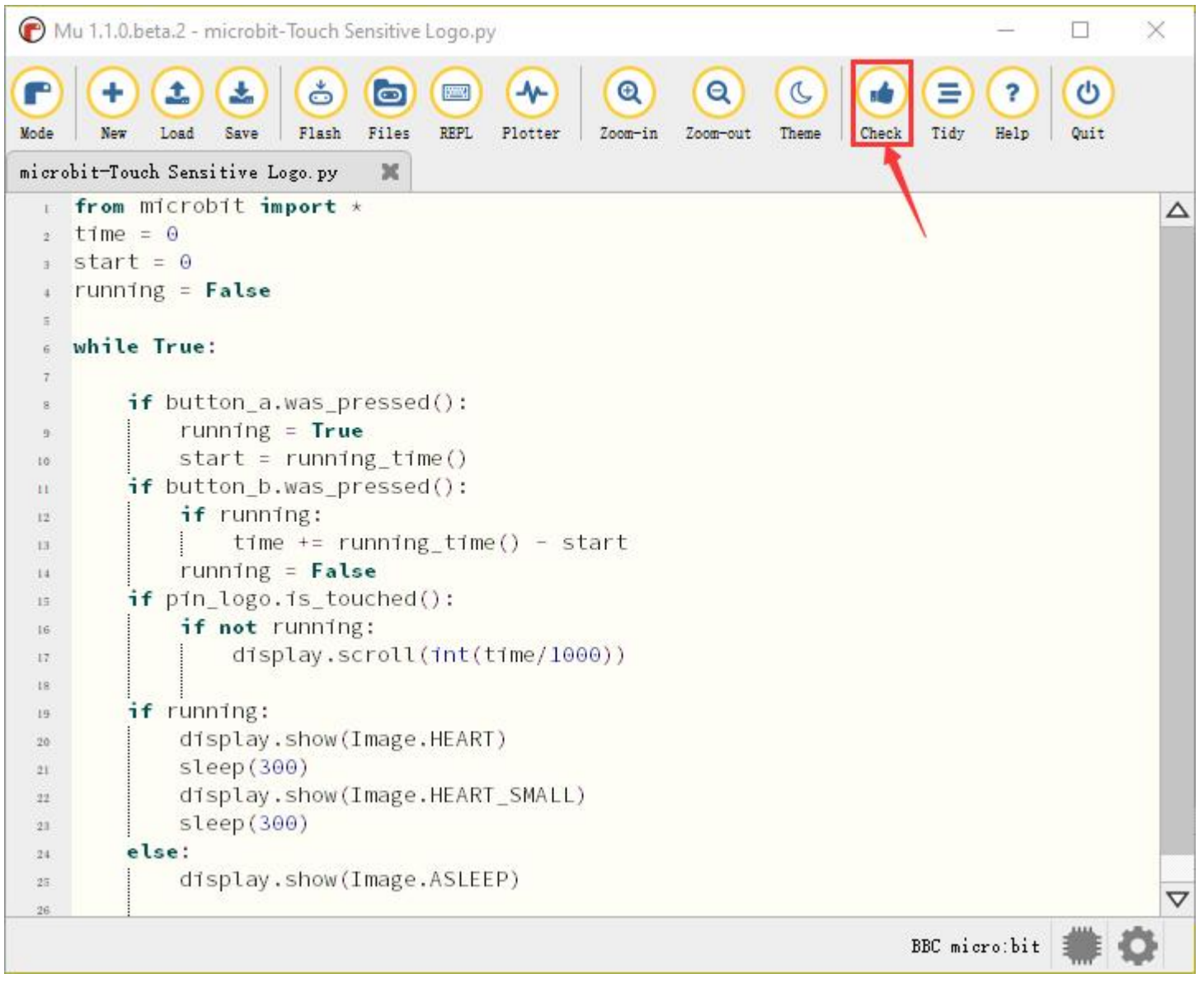
E. The program is also controlled by a Boolean variable named running. Boolean variable only has two values: true or false. If "running" is "true", it means that the stopwatch has started. If "running" is false, it means that the stopwatch has not started or has stopped.

F. If "running" is true, the beating heart pattern will be displayed on the LED dot matrix screen.

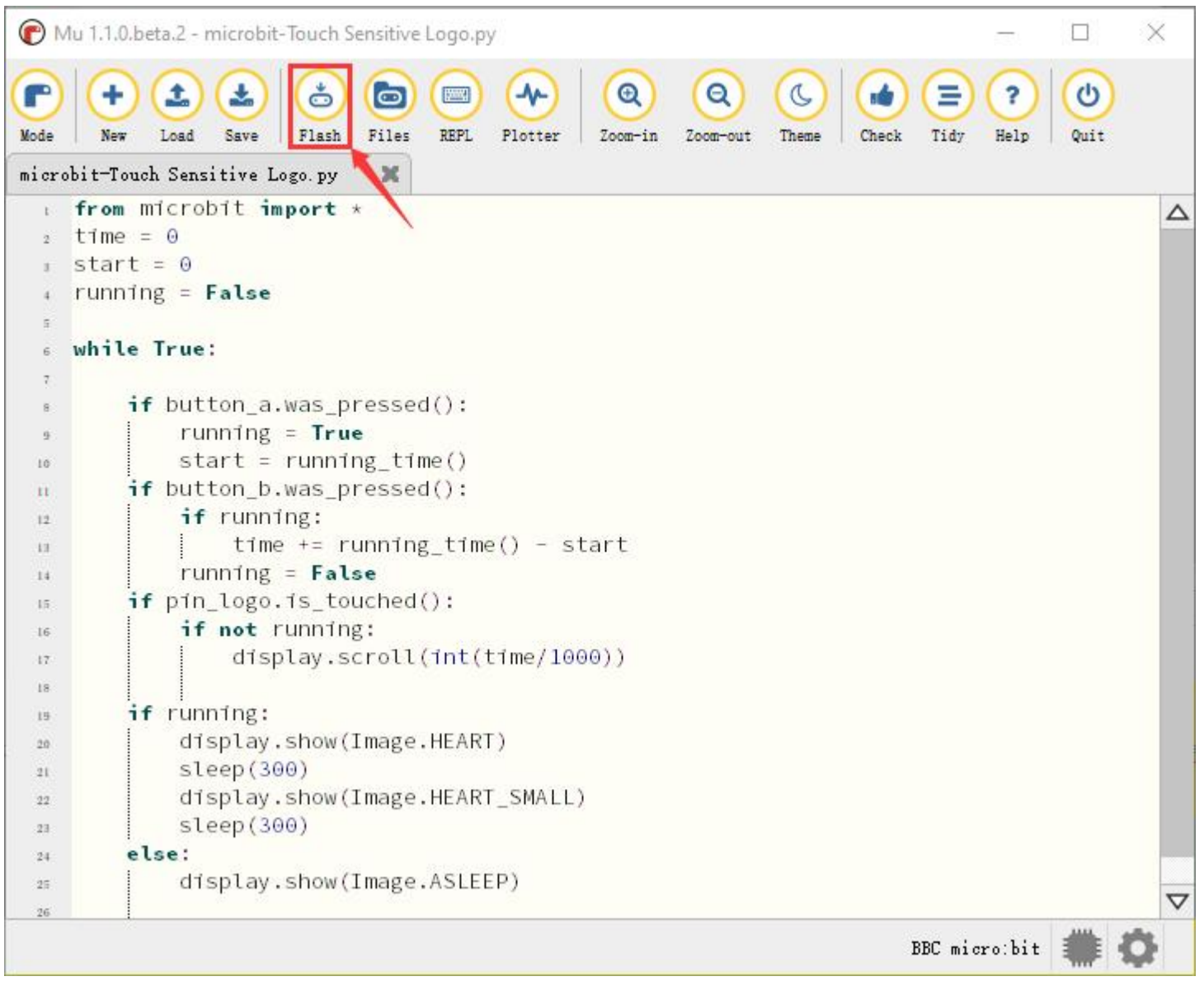
G. (7) If the stopwatch has stopped and the "running" is false, when you press the golden logo, it will only display the time.

H. If the stopwatch has been started and"running" is true, it only need to ensure that the time variable will change when button B is pressed, and the code can also prevent false readings.

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download code to the micro:bit board.

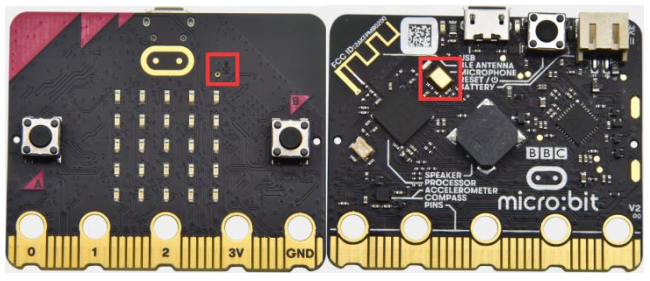


4. Test Result

Upload the test code to the micro:bit main board and power it via the USB cable, then press button A to start the stopwatch. When timing, the beating heart pattern will be displayed on the LED dot matrix screen. Press button B to stop it and you can start and stop it at any time.

It will keep recording time, just like a real stopwatch. Press the golden logo in the front of the micro:bit to display the measured time in seconds. And the time can be reset to zero by pressing the reset button on the back of it.

Project 11: Microphone



1. Description

The Micro: Bit main board has a built-in microphone, which can test the volume of ambient environment. When you clap, the microphone LED indicator turns on. Furthermore, it can measure the intensity of sound, thereby you can make a noise scale or disco lighting changing with music.

The microphone is placed on the opposite side of the microphone LED indicator and in proximity with holes that lets sound pass. When the board detects the sound, the LED indicator lights up.

2. Preparation

A. Attach the micro:bit main board to your computer via the USB cable

B. Open the offline version of Mu.

3. Test Code

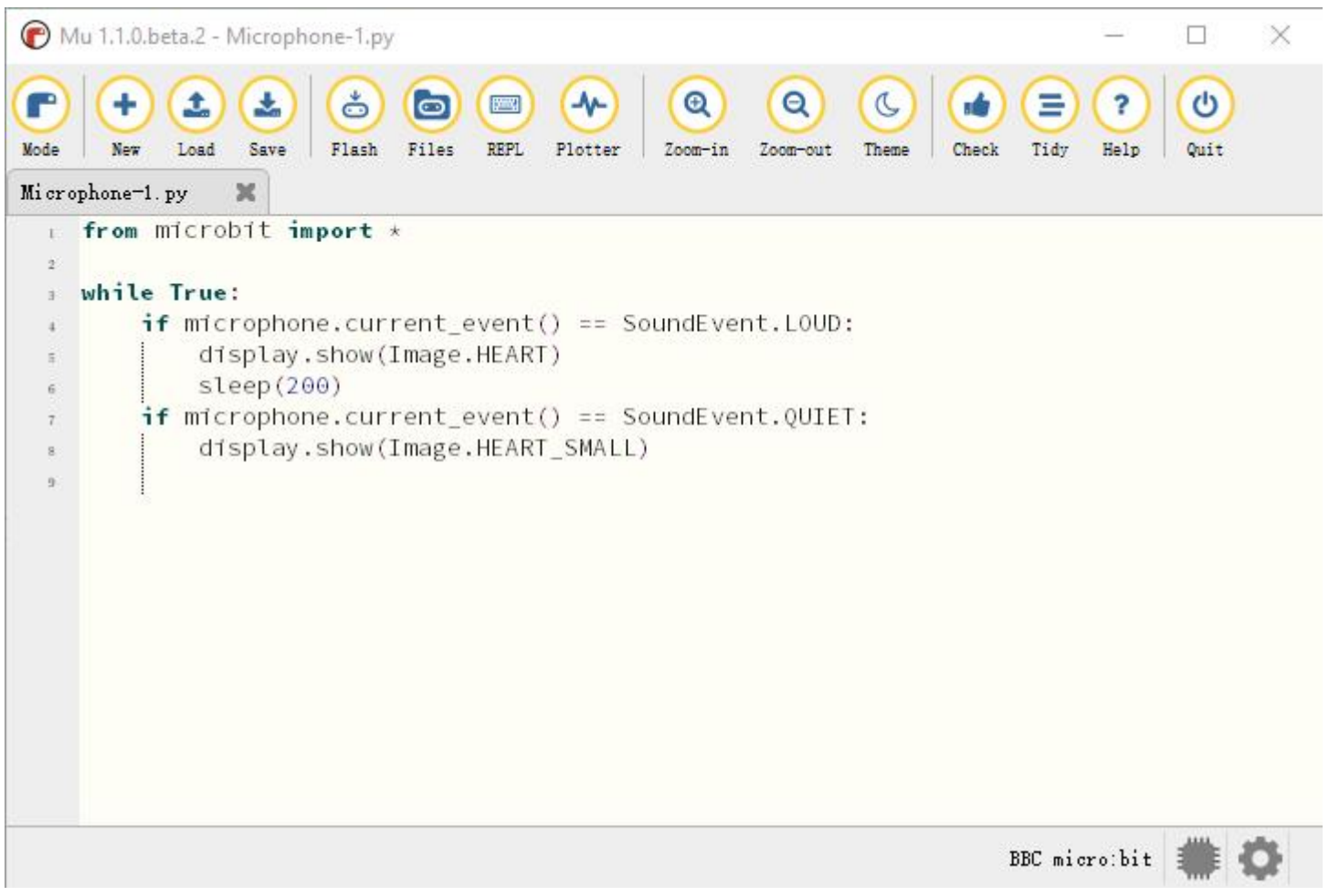
Test Code 1：

Enter Mu software and open the file “ Project 11： Microphone-1.py” to import the code: (How to load the project code?)

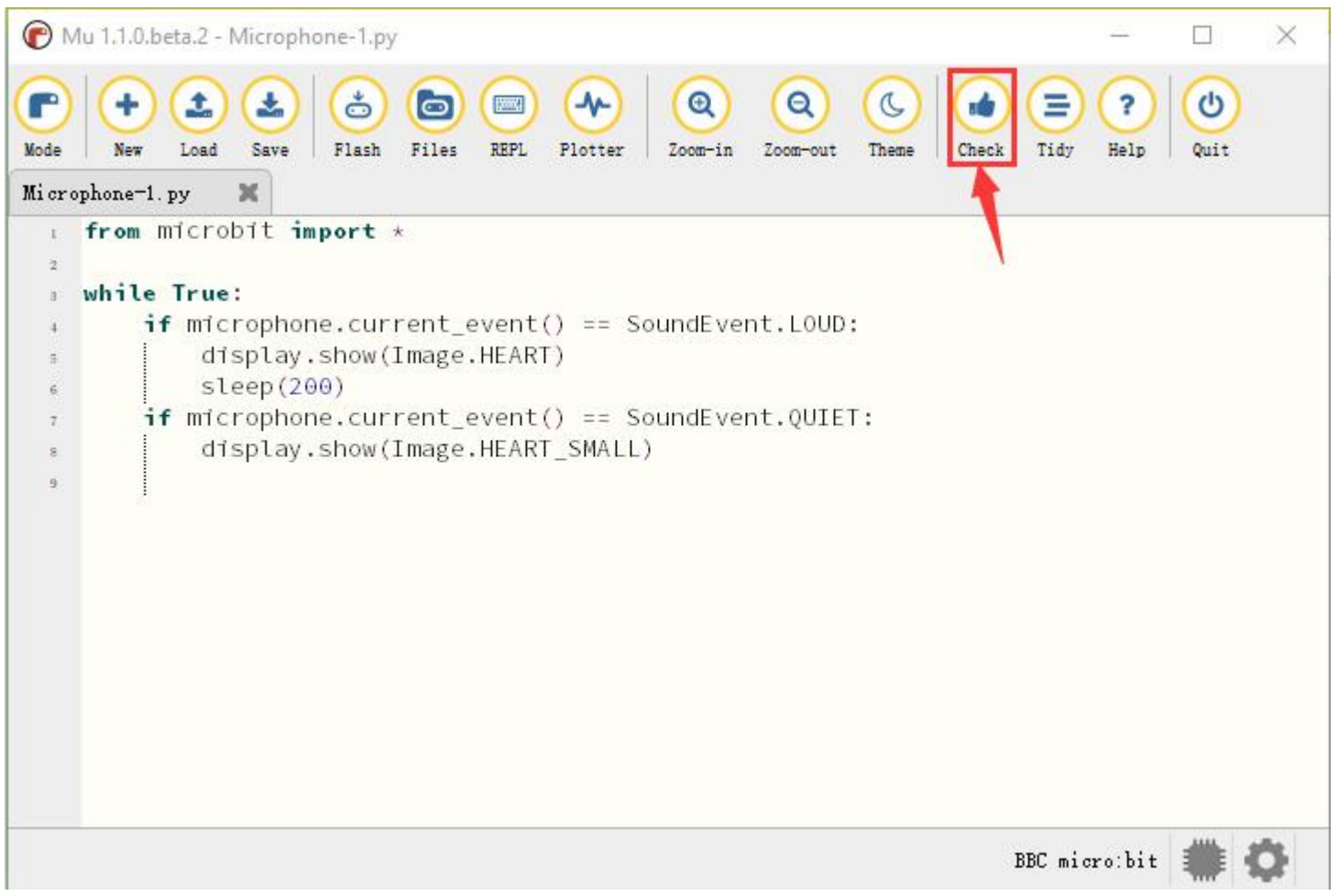
File Type	Route	File Name
Python file	../Python code/8.11： Microphone	microbit-Microphone-1.py

You can also input code in the editing window yourself.

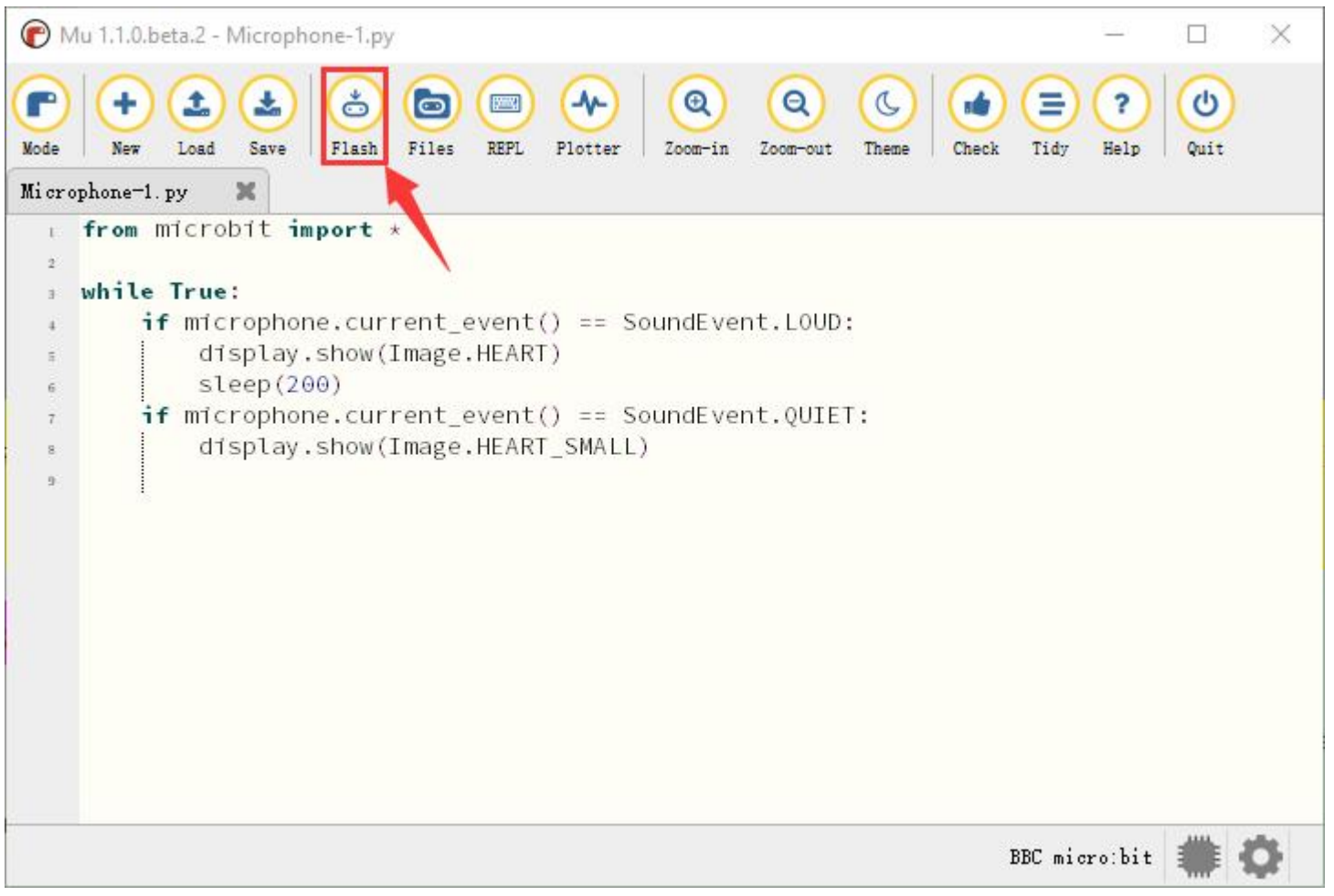
(Note: All words and symbols must be written in English.)



Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click "Flash" to download code to the micro:bit board.



After uploading the test code1 to the micro:bit main board and powering it via the USB cable, the LED dot matrix displays the pattern "❤" when you clap and the pattern when it is quiet around.

Test Code2:

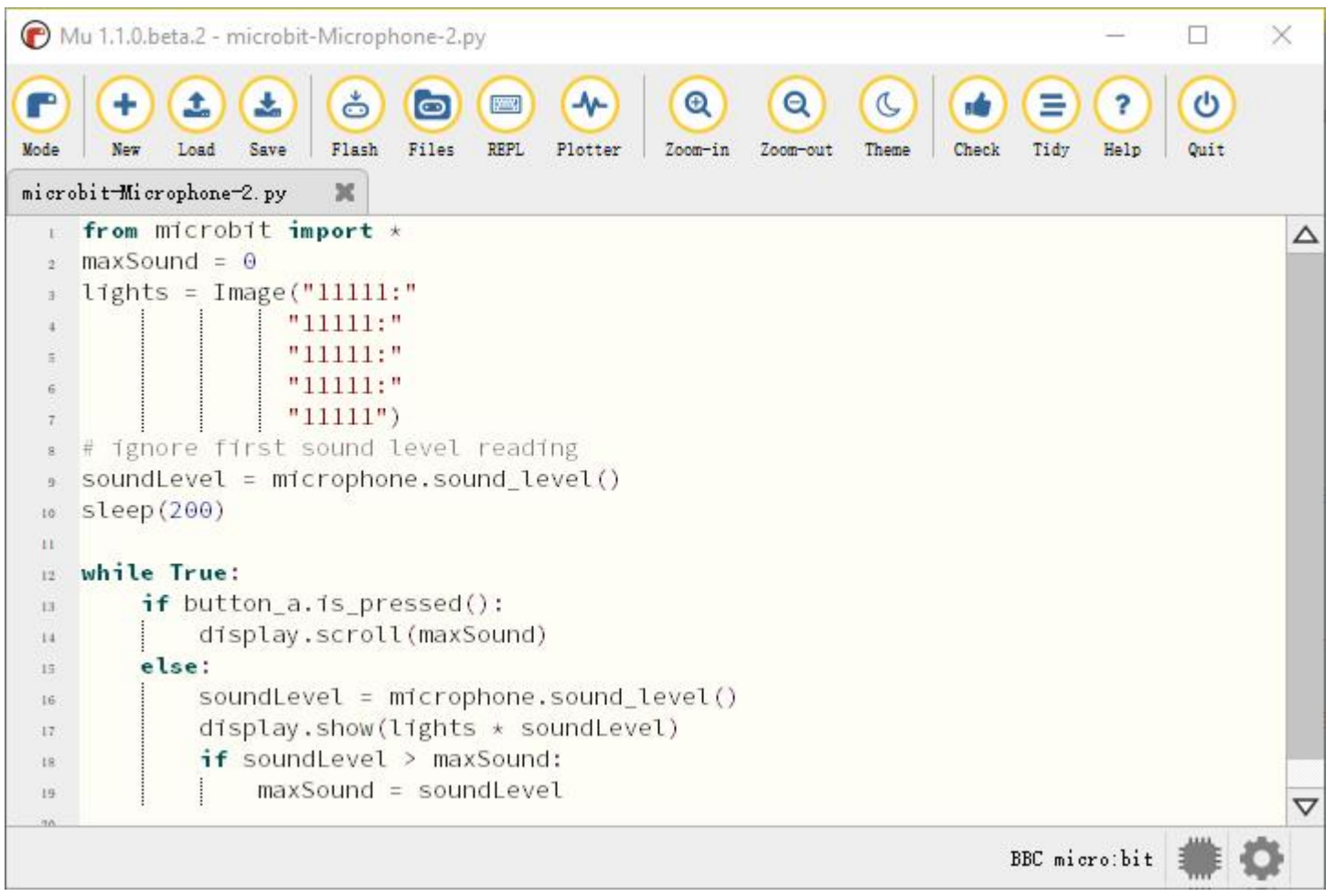
Enter Mu software and open the file "Project 11: Microphone-2.py" to import the code: (How to load the project

code?)

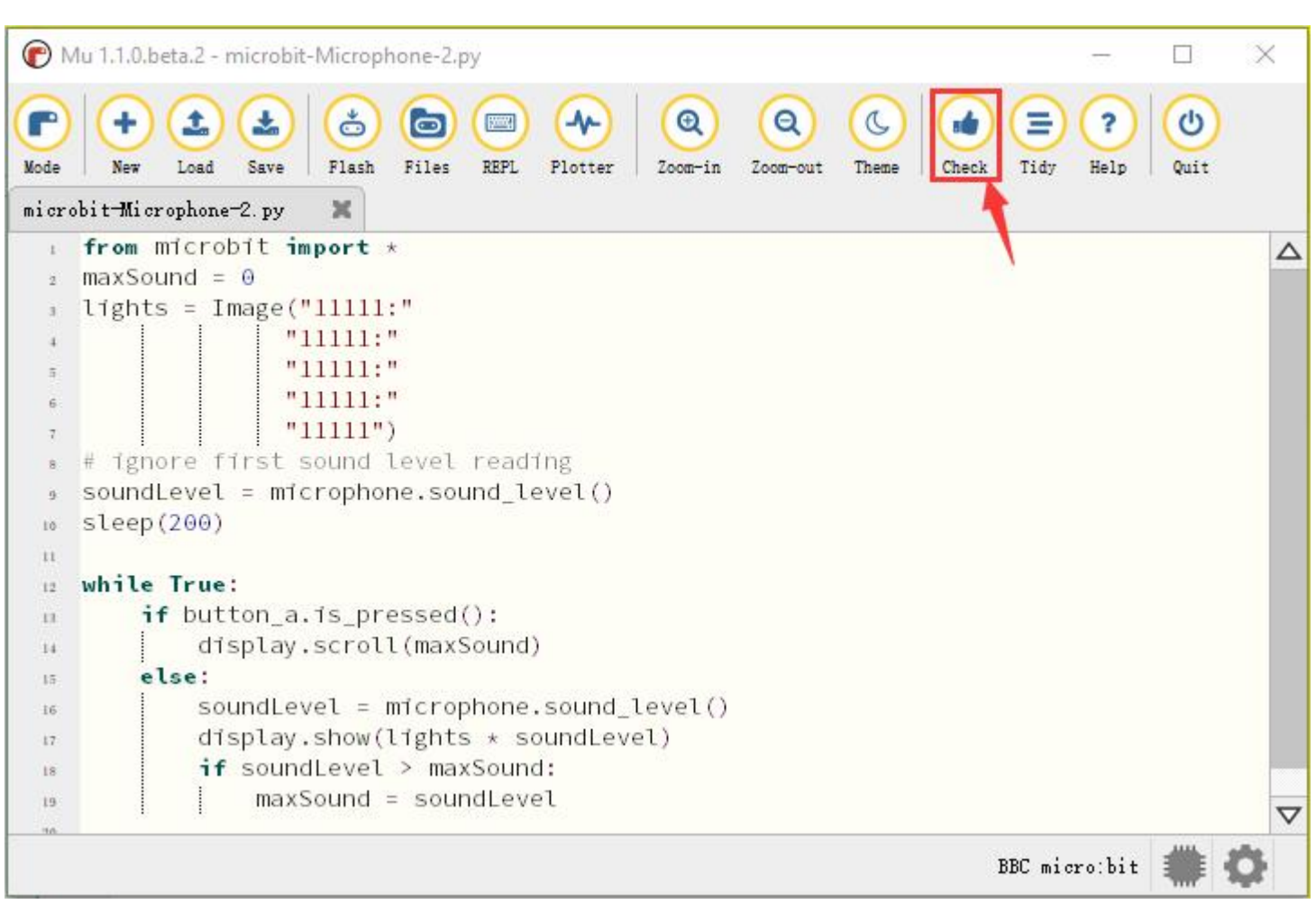
File Type	Route	File Name
Python file	../Python code/8.11： Microphone	microbit- Microphone -2.py

You can also input code in the editing window yourself.

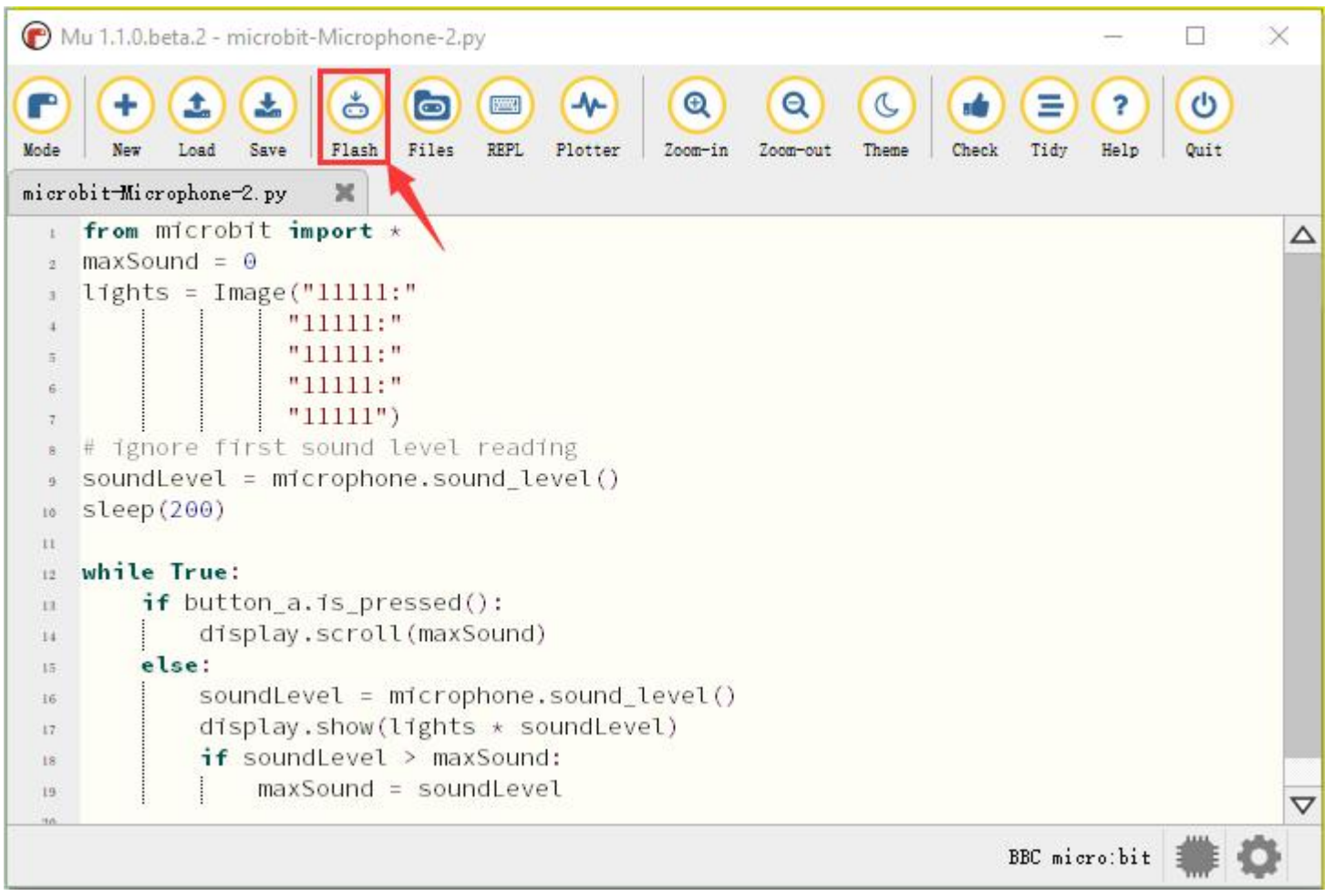
(Note: All words and symbols must be written in English.)



Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download code to the micro:bit board.



4. Test Result

Upload the test code to the micro:bit main board and power it via the USB cable. When the button A is pressed, the LED dot matrix displays the value of the biggest volume(please note that the biggest volume can be reset via the Reset button on the other side of the board). When clapping, the louder the tested sound, the brighter the 25 LEDs on the LED dot matrix screen.

5. Code Explanation

from microbit **import** *	Import the library of micro：bit
while True:	This is a permanent loop that makes micro:bit execute the code of it.
if microphone.current_event() == SoundEvent.LOUD: display.show(Image.HEART) sleep(200) **if** microphone.current_event() == SoundEvent.QUIET: display.show(Image.HEART_SMALL)	If there is a sound LED shows ❤ Delay in 200ms if no sound is detected LED lights show

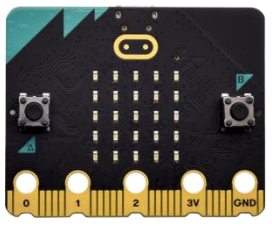
print("Light intensity:", Lightintensity)	BBC microbit REPL prints the detected light intensity value
maxSound = 0	The initial value of maxSound is 0
lights = Image("11111:""11111:""11111:""11111:""11111")	Assign Image() to variable lights
soundLevel = microphone.sound_level()	Assign microphone.sound_level() to the variable soundLevel
if button_a.is_pressed(): display.scroll(maxSound) **else:** soundLevel = microphone.sound_level() display.show(lights * soundLevel) **if** soundLevel > maxSound: maxSound = soundLevel	if the button A is pressed LED lights show the sound value If not Assign microphone.sound_level() to the variable soundLevel As the sound changes，the micro:bit will display the breathing light effect If the sound value is higher than its maximum value the maximum sound value is equal to sound level value

Project 12: Control Speaker

1. Description

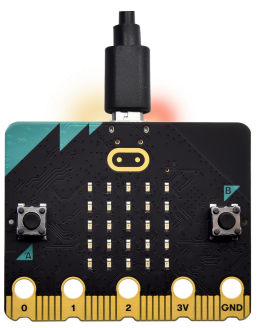
In the previous projects, we have learned about the touch-sensitive logo and the speaker respectively. In the project, we will combine these two components to play music.

2. Components Needed

	
Micro:bit main board *1	USB cable*1

3. Wiring Diagram

Attach the Micro:bit main board to your computer via the USB cable.



4. Test Code

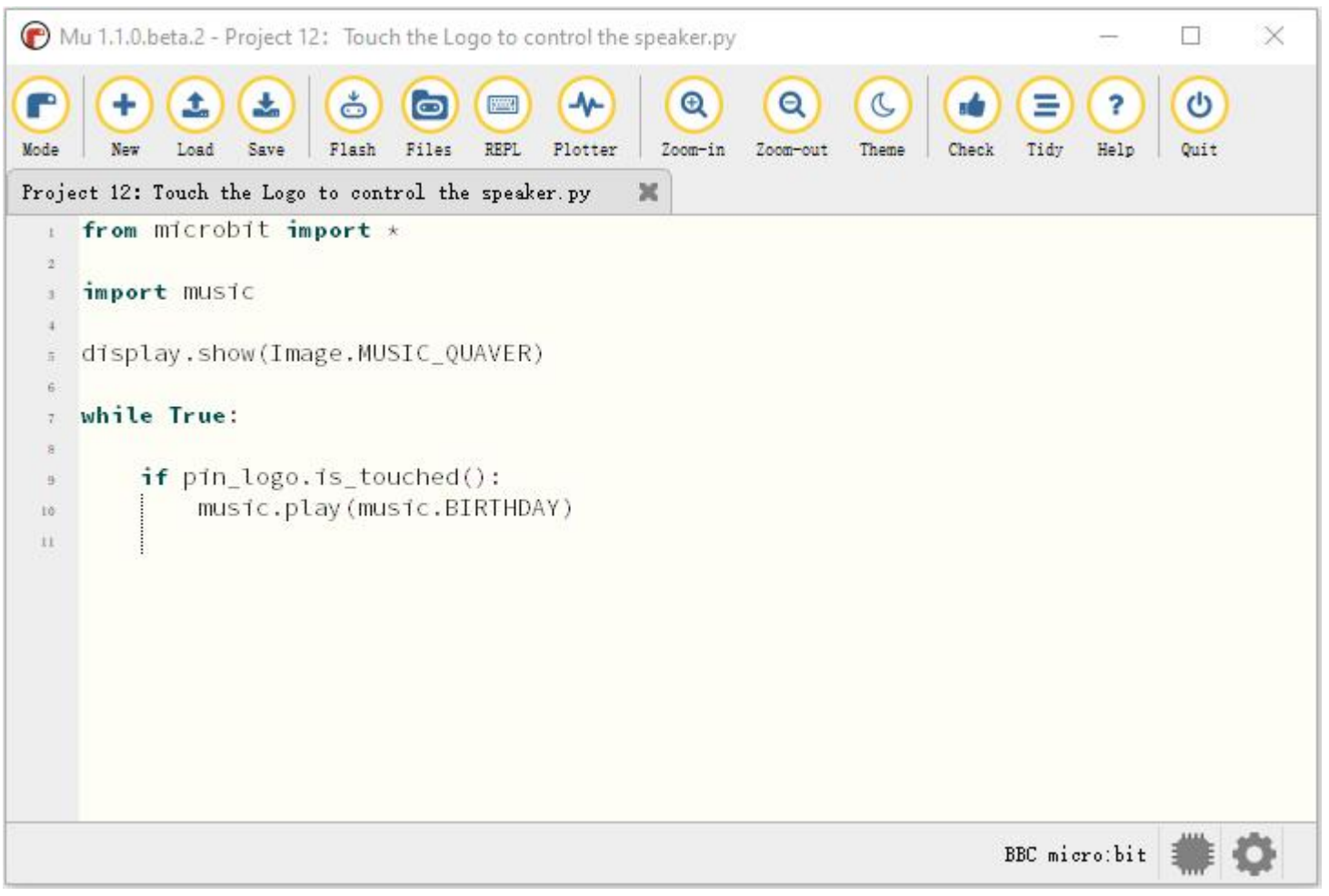
Enter Mu software and open the file “Project 12：Control Speaker.py” to import code:

(How to load the project code?)

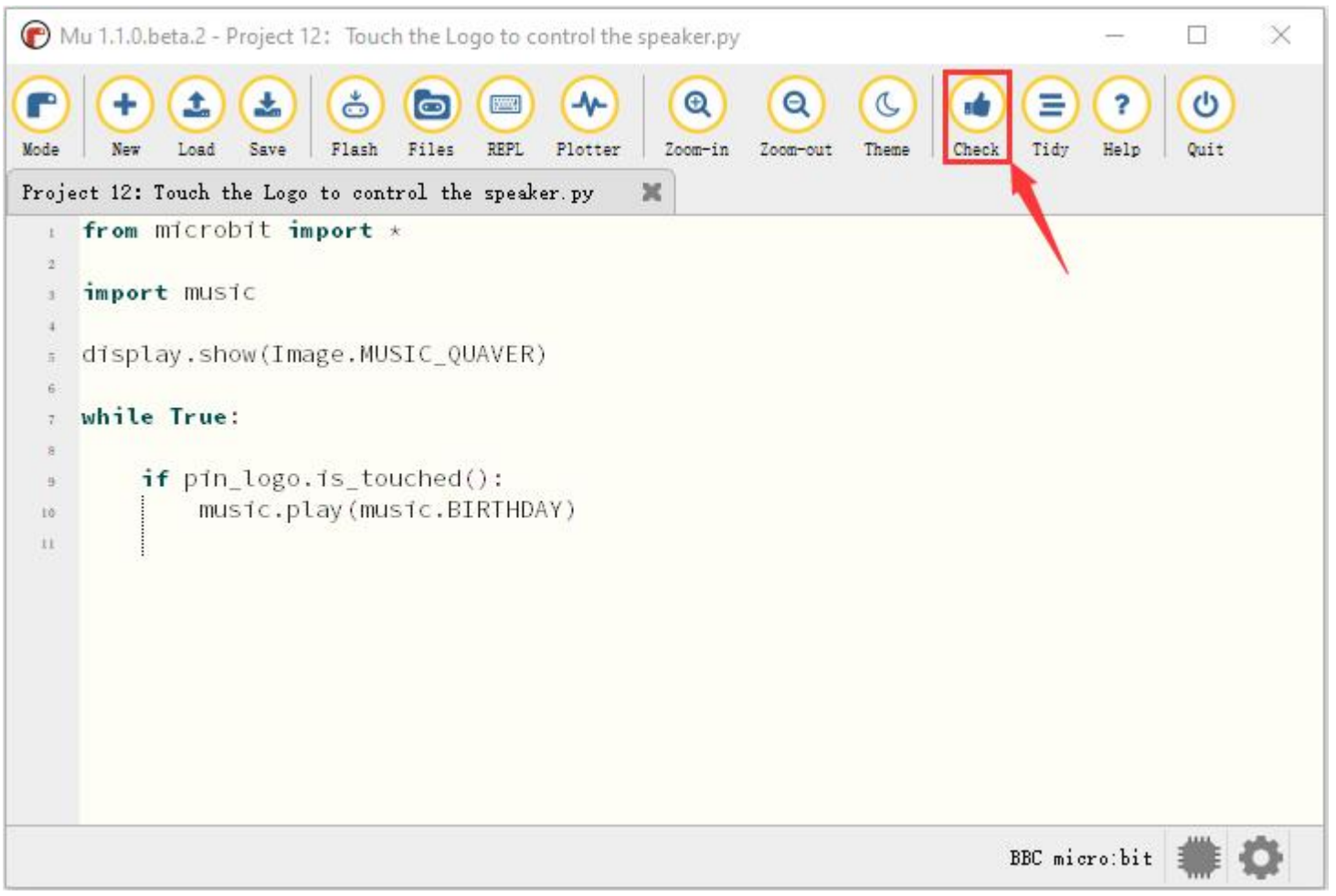
File Type	Route	File Name
Python file	../Python code/8.12：Touch-sensitive Logo Controlled Speaker	Touch the Logo to control the speaker.py

You can also input code in the editing window yourself.

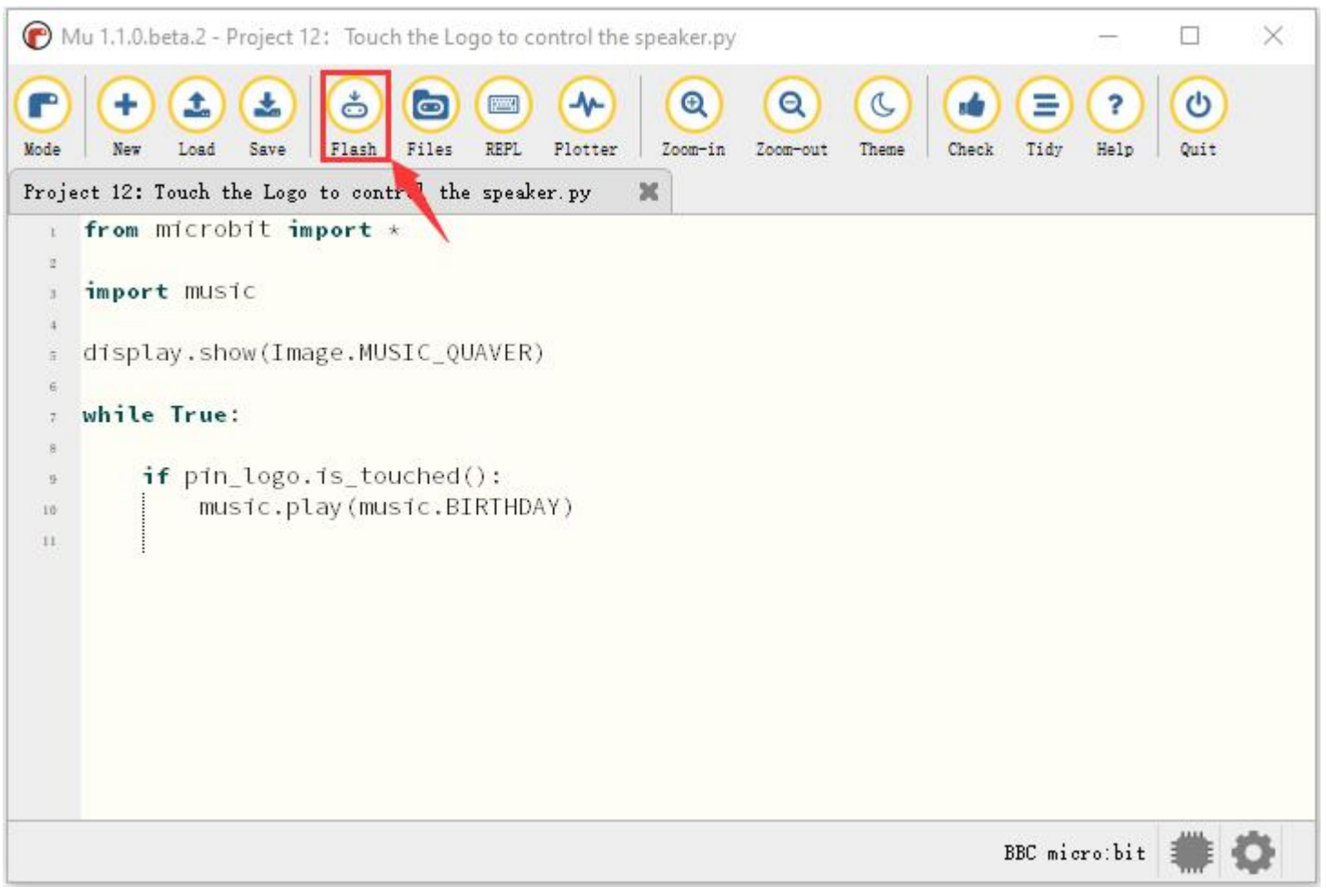
(Note: All words and symbols must be written in English.)



Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



5. Test Result

After uploading the test code to the micro:bit main board and powering it via the USB cable, the speaker plays the song *"Happy Birthday to You"* when the logo is touched.

6. Code Explanation

from microbit **import** *	Import the library of micro：bit
while True:	This is a permanent loop that makes micro:bit execute the code of it.
display.show（Image.MUSIC_QUAVER)	The LED dot matrix will show the music logo
if pin_logo. is_touched( ):	When the logo is touched, it executes the following command
music.play（music.BIRTHDAY)	The speaker plays the song" *Happy Birthday to You*'

Bluetooth Wireless Communication

The micro:bit owns a low-consumption Bluetooth module to communicate but with 16k RAM. However, BLE heap stack occupies 12K RAM, thereby there is no enough space to run microPython.

At present, microPython doesn’ t support the Bluetooth service.

https://microbit-micropython.readthedocs.io/en/latest/ble.html

The former projects are the introduction of sensors and modules. The further lessons are challenging for new starters.

(Note: In order to refrain the micro:bit board from being burned, disconnect the micro USB cable from it and turn off the power on the micro:bit motor driver base plate before installing it on the car expansion board and dial the POWER switch to the OFF end. Likewise, before removing the the main board from the car expansion board, disconnect the micro USB cable from it and turn off the power on the micro:bit motor driver base plate.

Project 13: Seven-Color LED

\

1. Description

This module consists of a commonly used LED with 7colors but in white appearance. It can automatically flash different colors to create fantastic light effects when high level is input like a normal LED.

2. Preparation

- Insert the micro:bit board into the slot of keyestudio 4WD Mecanum Robot Car V2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect the micro:bit to your computer via an USB cable
- Open the offline version of Mu.

3. Test Code

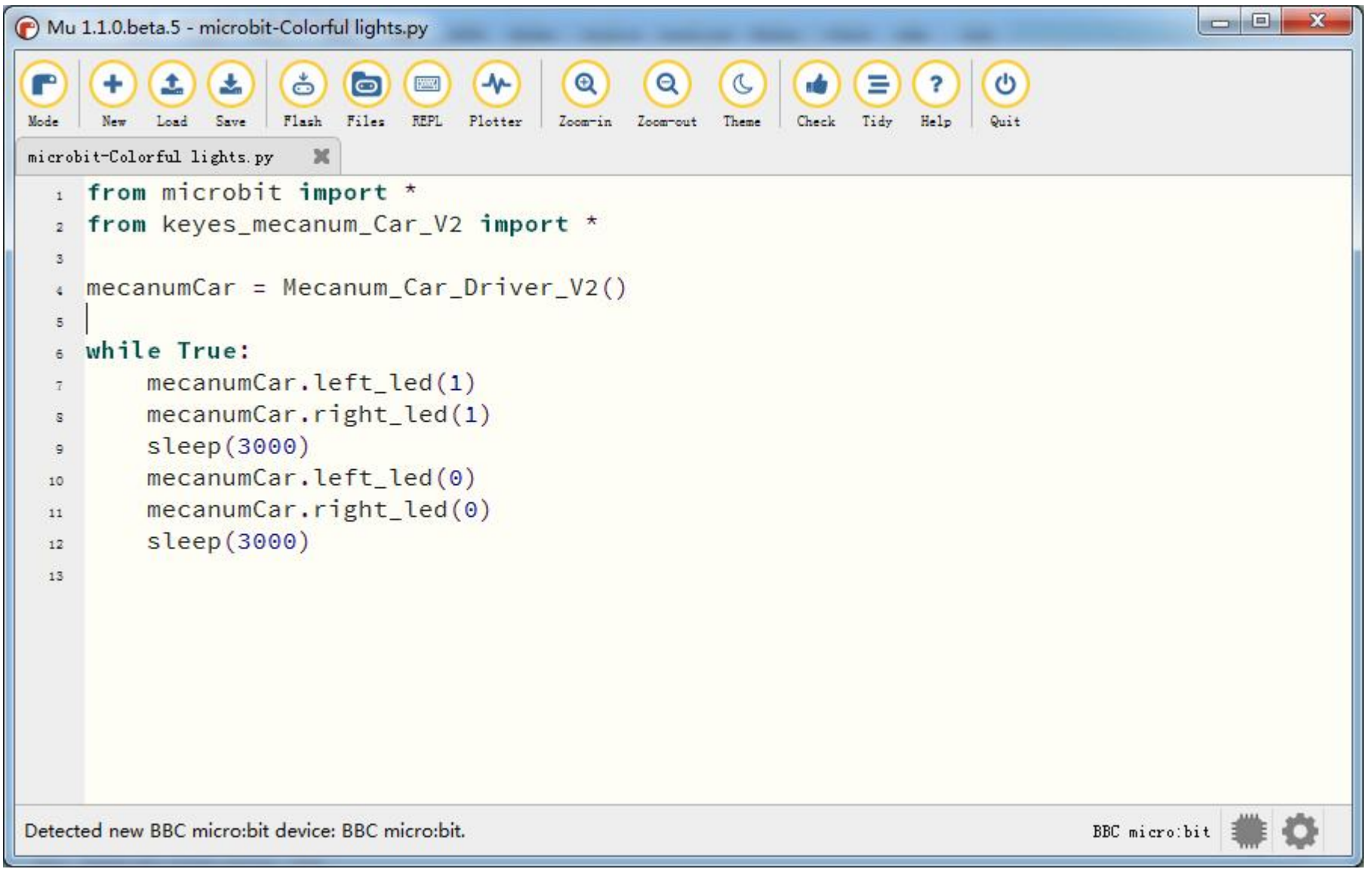
Enter Mu software and open the file "Project 13: Seven-Color LED to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.13: Colorful lights	microbit-Colorful lights.py

You can also input code in the editing window yourself.

(Note: All words and symbols must be written in English.)



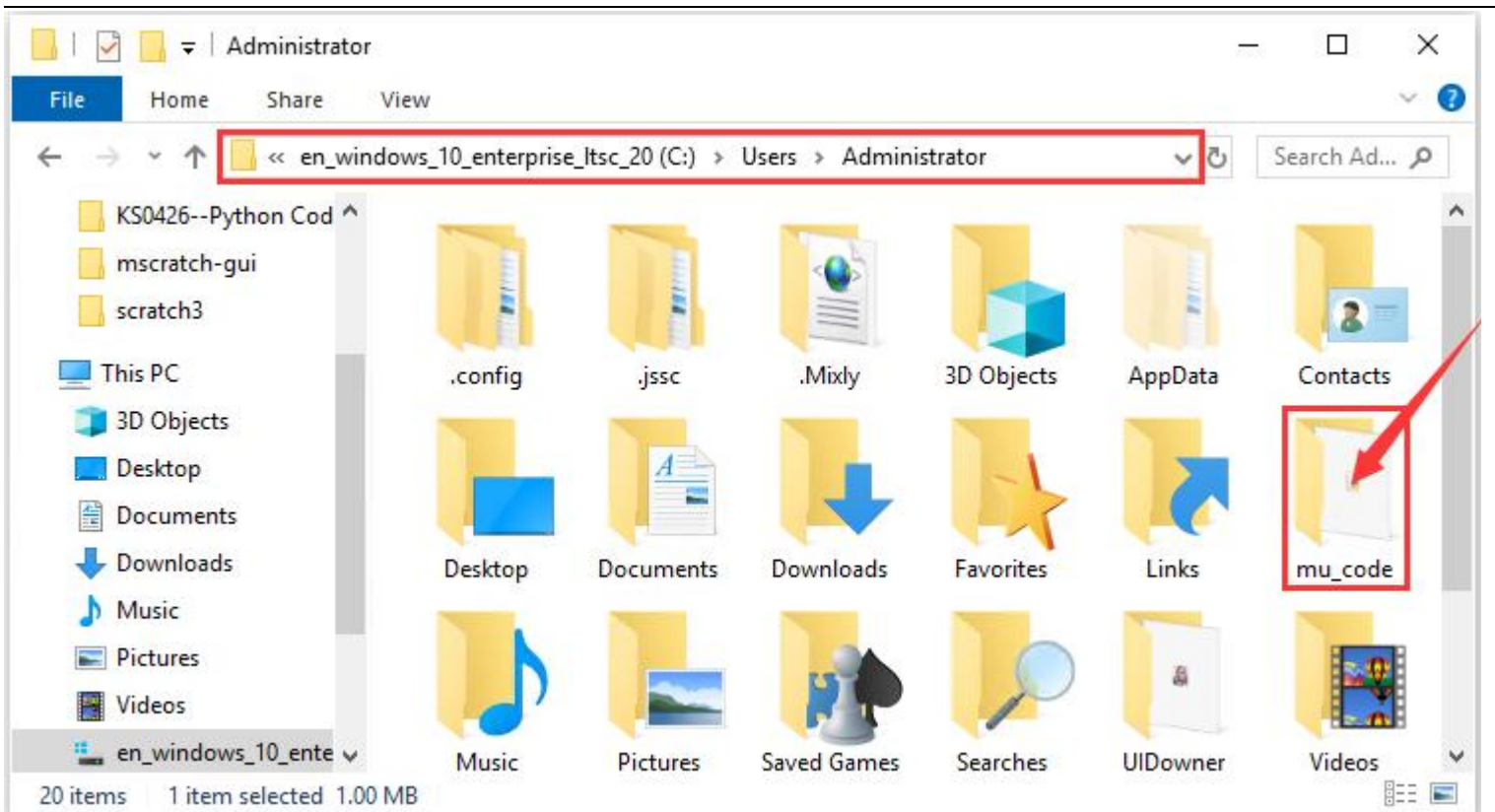
Don't click "Flash", but import the "keyes_mecanum_Car_V2.py" library file into the micro:bit. This file contains the control method of the Micro:bit Mini Smart Mecanum Wheel Smart Car.

Import the "keyes_mecanum_Car_V2.py" file.

The default directory for Mu files is "Mu_code", which is located in the root directory of the user directory. Reference link: https://codewith.mu/en/tutorials/1.0/files

For example, on the windows system, suppose your system is installed on the C driver of the computer, and the user name is "Administrator", then the path of the "mu_code" directory is "C:\Users\Administrator\mu_ code". On Linux systems, the path of the "mu_code" directory is "~/home/mu_code".

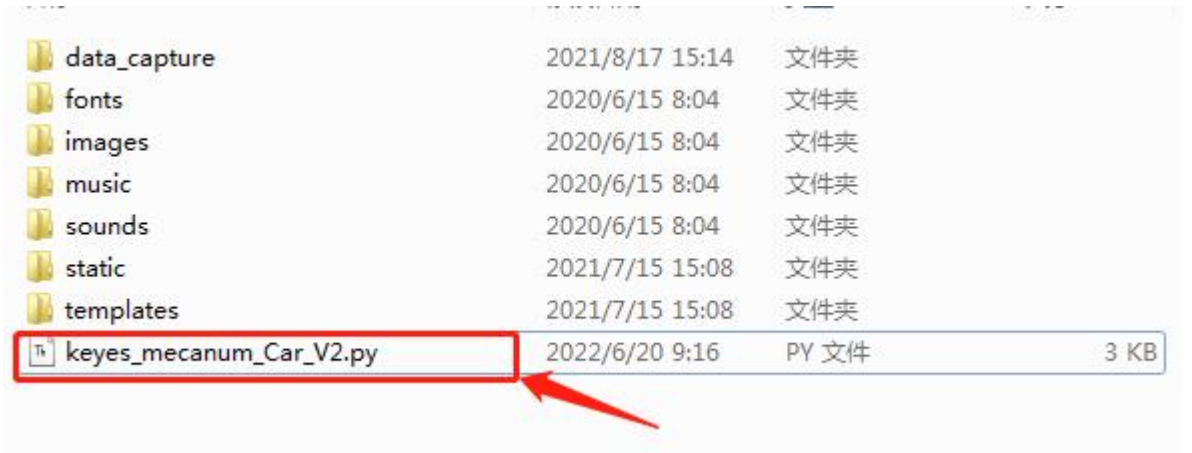
Enter the "mu_code" folder.



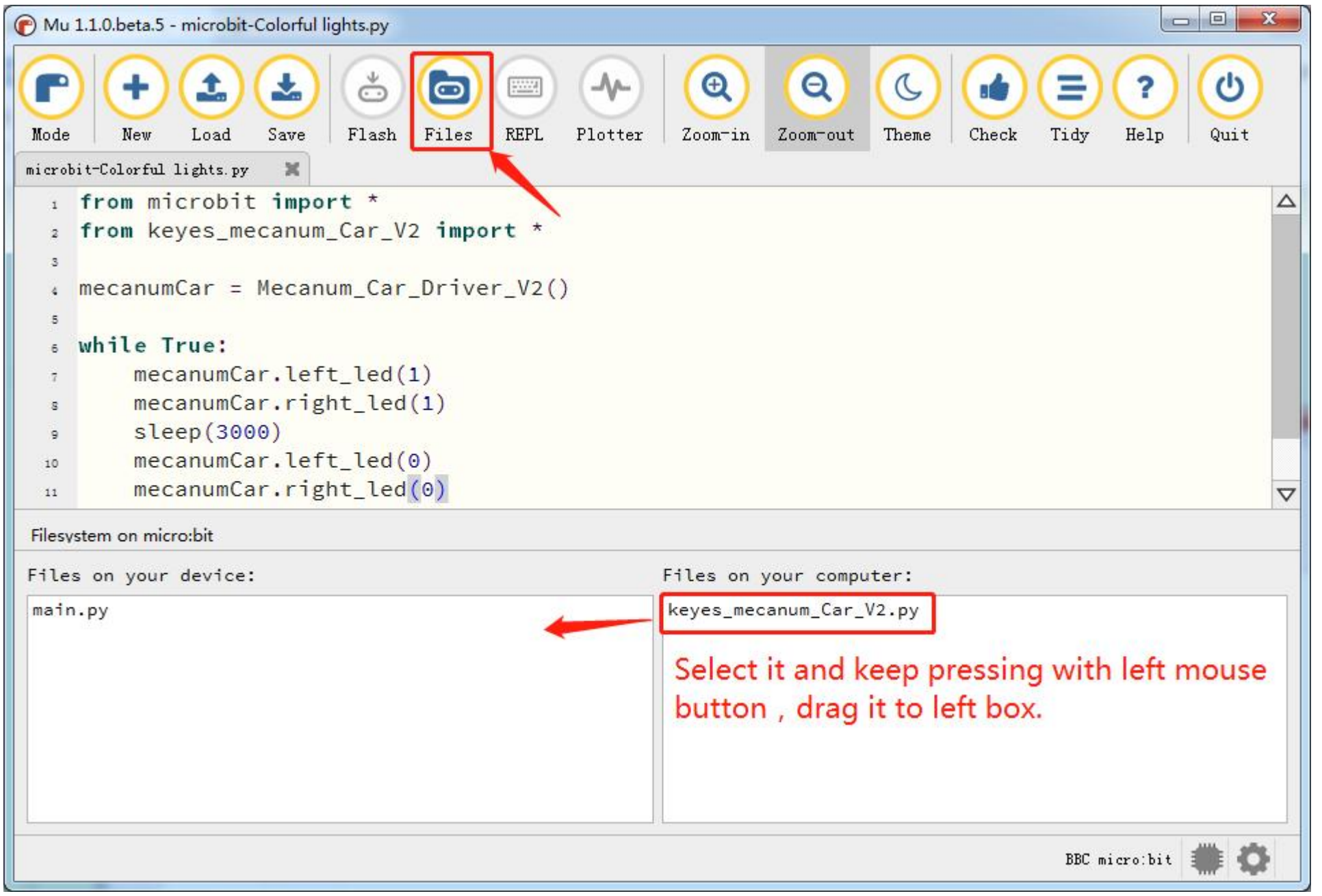
Copy “keyes_mecanum_Car.py “library file to the folder” mu_code “ and the path is below :

File type	Path	File name
Python file	../PythonCode/LibrariesmecanumCar_python_Libraries	keyes_mecanum_Car_V2.py

When the copy is done, as shown below:

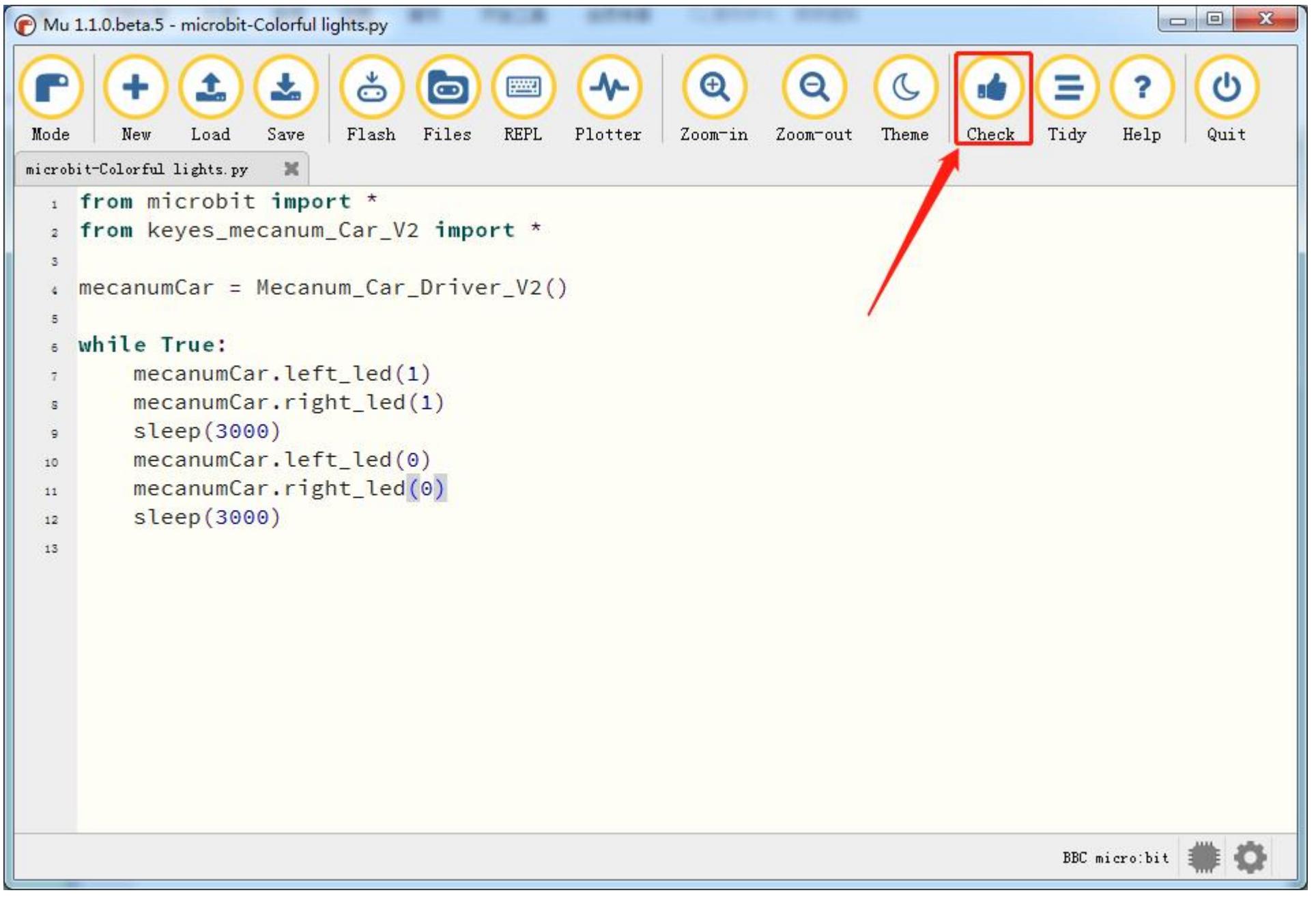


First open the Mu software and connect the micro:bit to your computer, then click the "Files" button, and drag the "keyes_mecanum_Car.py" library file to the micro:bit.



After a few seconds, the import is complete and you can see it in the box on the left.

After the library file is imported successfully, you also need to click the "Check" button to check the code. If a cursor or an underline appears on a certain line, then errors appear in the program.

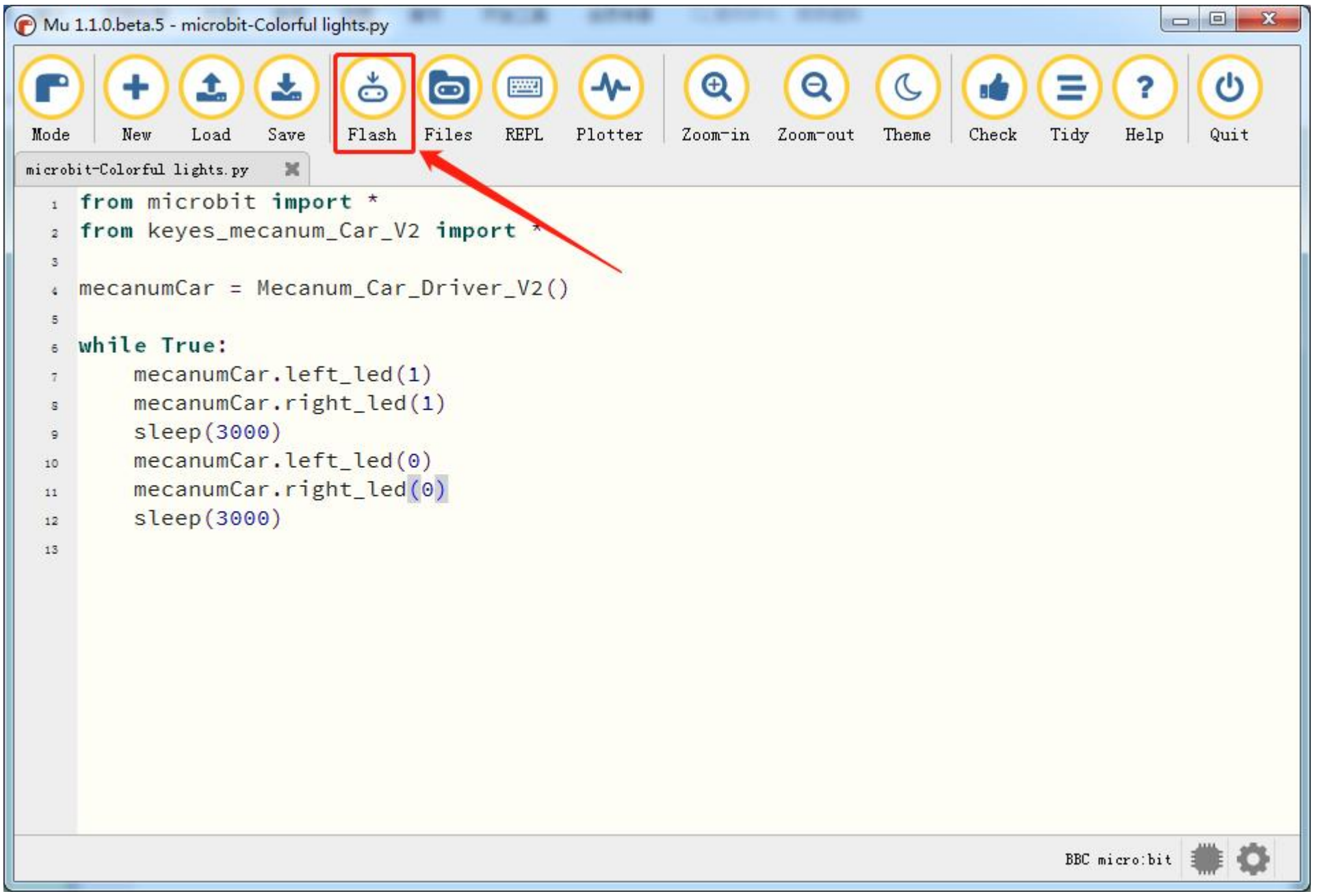


However, during this process, the following prompt will appear even if there is no error in the code. These prompts are just warnings not the code error prompts.

```
↑ 'from keyes_mecanum_Car import *' used; unable to detect undefined names
```

```
↑ 'Mecanum_Car_Driver' may be undefined, or defined from star imports: keyes_mecanum_Car
```

If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



If errors appear after clicking the "Flash" button, please confirm whether you have imported the provided "keyes_mecanum_Car_V2.py" library file.

Note:

Before programming with Micropython, you need to import the "keyes_mecanum_Car_V2.py" library file to the micro:bit. If you program with different micro:bit, the library file"keyes_mecanum_Car_V2.py" needs to be imported again to a new micro:bit.

4. Test Result

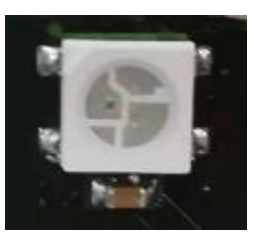
Download the code to the micro:bit board and dial POWER switch to ON end, then the seven-color LED will flash in 3s and then stop in 3s and repeat this pattern.

5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
from keyes_mecanum_Car **import ***	Import the library file of keyes_mecanum_Car_V2

mecanumCar = Mecanum_Car_Driver()	Instantiate an object Mecanum_Car_V2Driver() as mecanumCar
while True:	This is a permanent loop that makes micro:bit execute the code of it.
mecanumCar.left_led(1)	Light up the seven-color LED on the left. (1 is on, 0 is off)
mecanumCar.right_led(1)	Light up the seven-color LED on the right. (1 is on, 0 is off)
sleep(3000)	Delay in 3000ms
mecanumCar.left_led(0)	Turn off the seven-color LED on the left. (1 is on, 0 is off)
mecanumCar.right_led(0)	Turn off the seven-color LED on the right. (1 is on, 0 is off)

Project 14: 4 WS2812 RGB LEDs



1. Description

The driver shield cooperates 4 pcs WS2812 RGB LEDs, compatible with micro:bit board and controlled by P7. In this lesson, we will make the RGB LEDs display different colors by P7. In this lesson, 3 sets of test code are provided to make the 4 WS2812 RGB LEDs display different effects.

Sample	Color	RGB Value（R,G,B）	Color Code (16 colors)	Sample	Color	RGB Value（R,G,B）	Color Code (16 colors))
	Red	255, 0, 0	#FF0000		Orange	255, 165, 0	#FFA500
	Yellow	255, 255, 0	#FFFF00		Green	0, 255, 0	#00FF00
	Blue	0, 255, 0	#0000FF		Indigo	75, 0, 130	#4B0082
	Violet	238, 130, 238	#EE82EE		Purple	160, 32, 240	#A020F0
	Black	0, 0, 0	#000000		White	255, 255, 255	#FFFFFF
.....	.....	.....	.....	.....	.....	.....	.....
Change the value of the R,G and B to get different colors							

2. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect the micro:bit to your computer via an USB cable
- Open the offline version of Mu.

3. Test Code

Code1:

Enter Mu software and open the file “Project 14：4 WS2812 RGB LEDs-1.py” to import code:

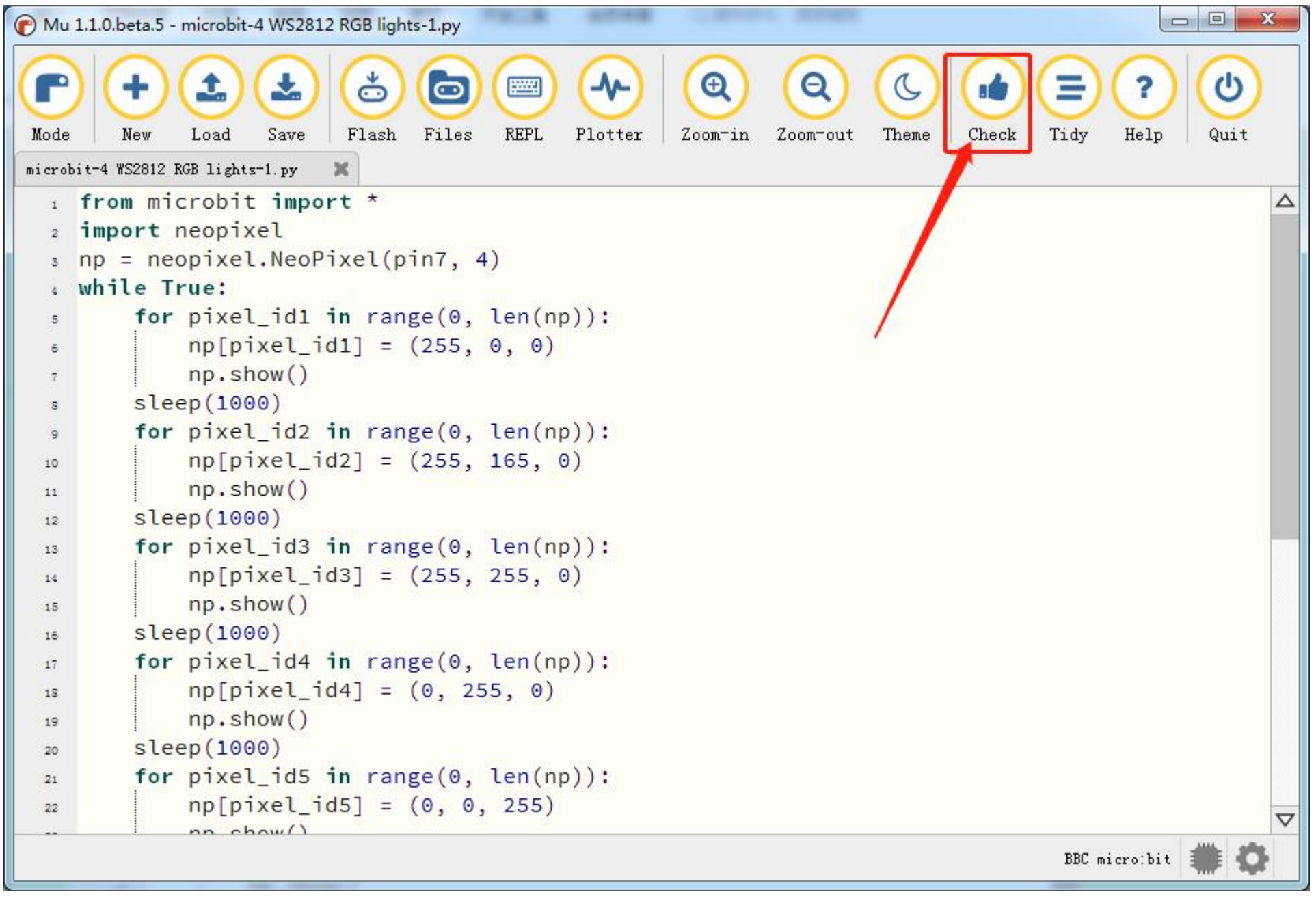
(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.14：WS2812 RGB lights	microbit-4 WS2812 RGB lights-1.py

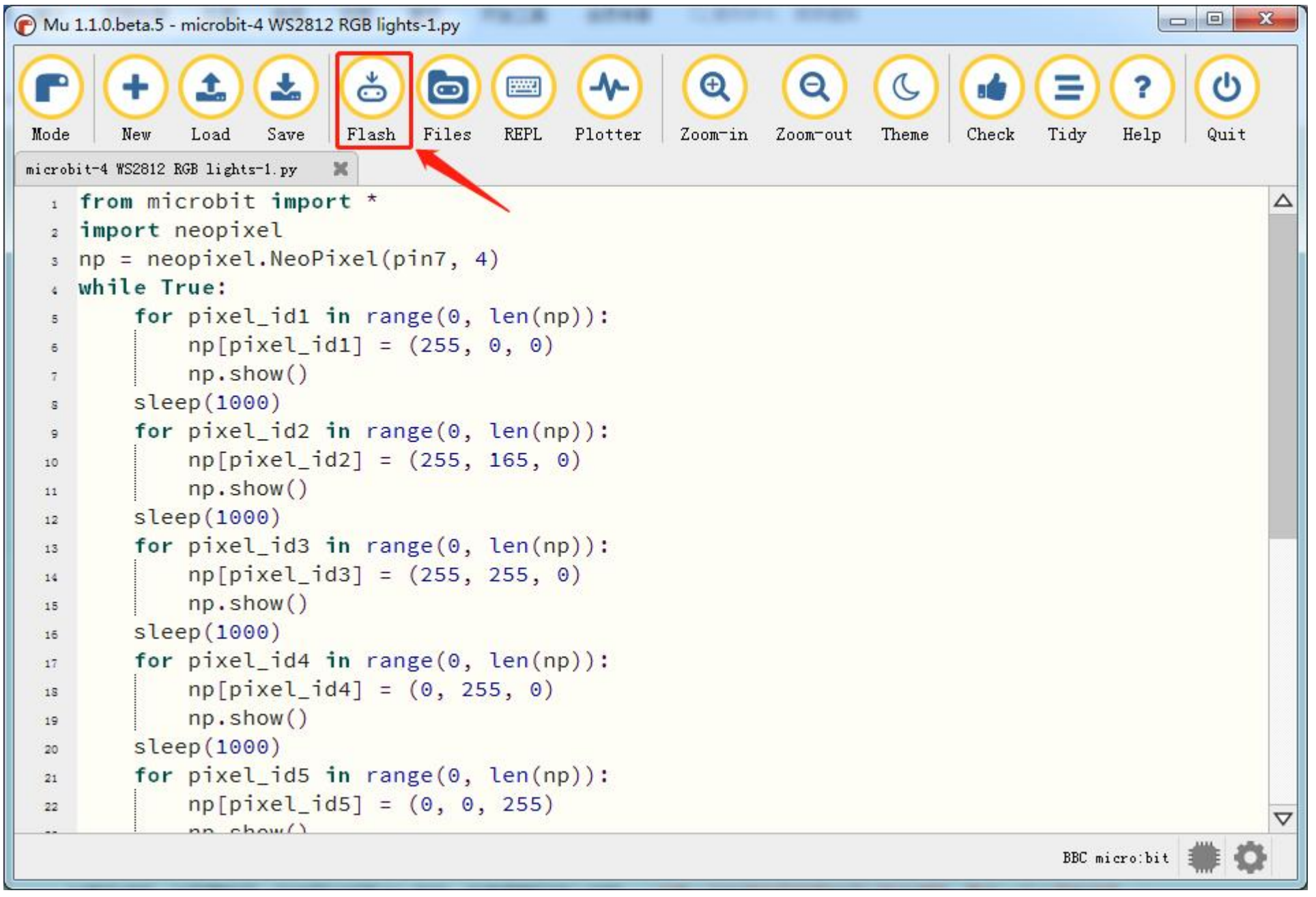
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.



If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



Code 2:

Enter Mu software and open the file "Project 14: 4 WS2812 RGB LEDs-2.py" to import code:

(How to load the project code?)

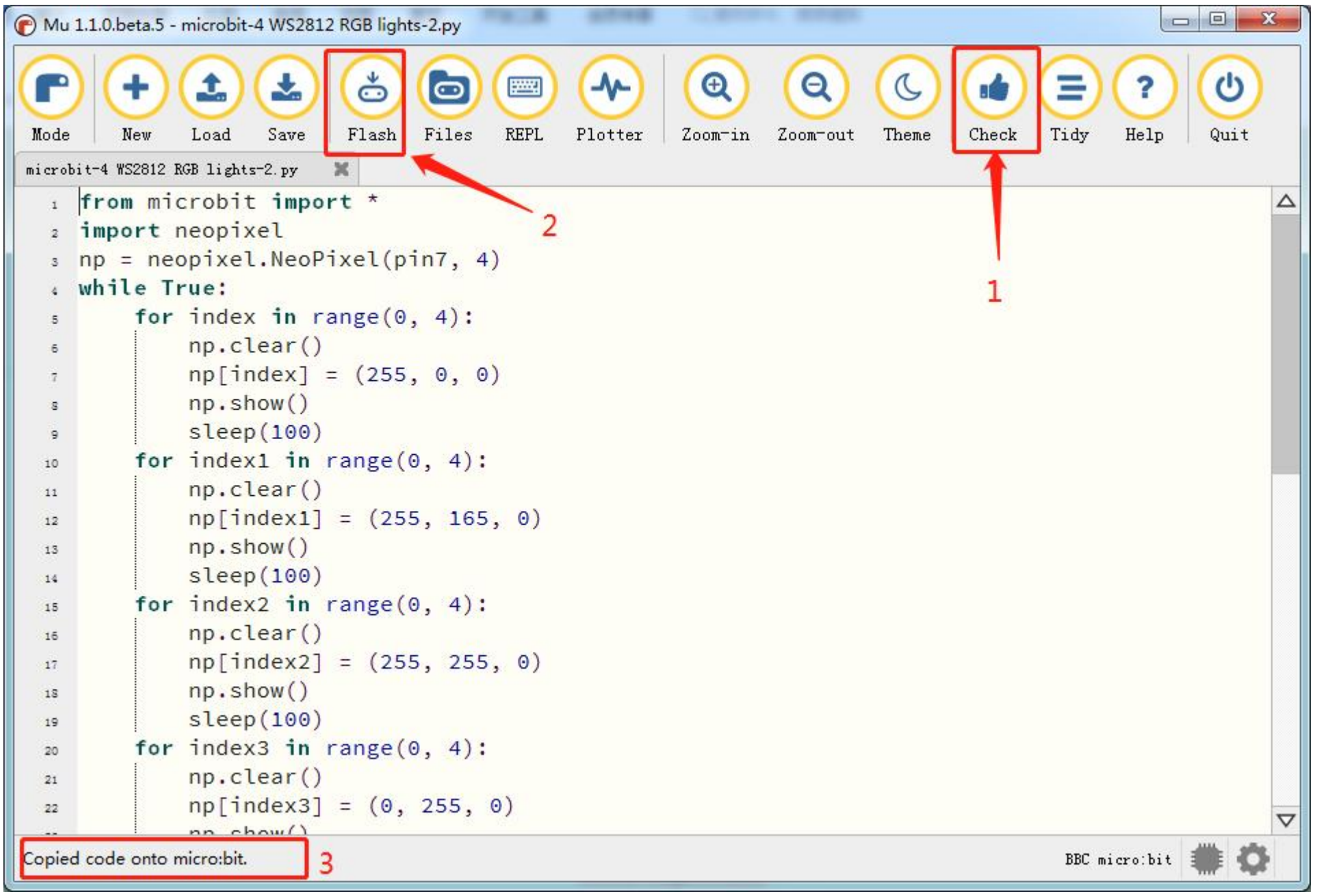
File Type	Route	File Name
Python file	../Python code/8.14: WS2812 RGB lights	microbit-4 WS2812 RGB lights-1.py

You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



Code 3:

：Enter Mu software and open the file “Project 14：4 WS2812 RGB LEDs-3.py” to import code:

(How to load the project code?)

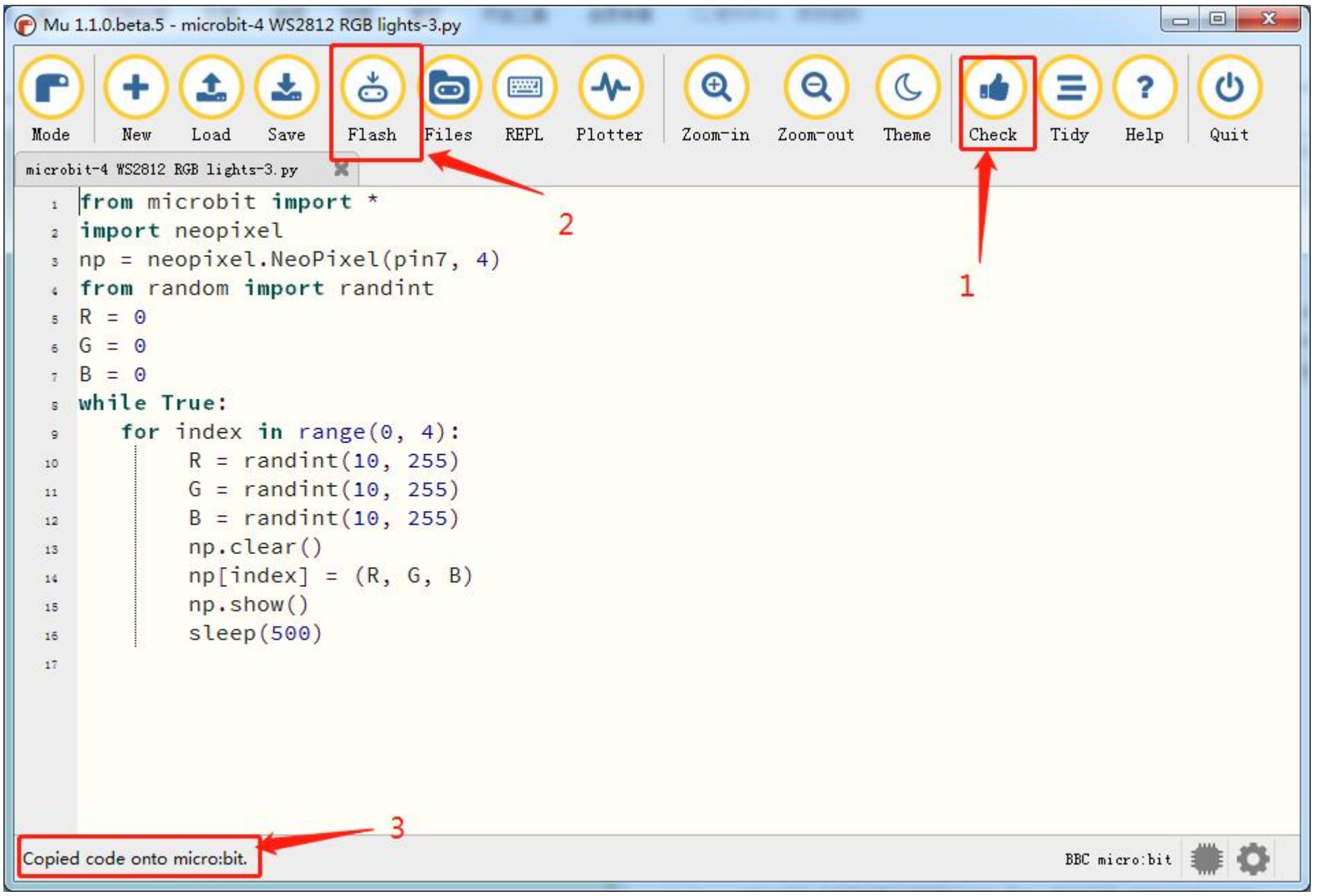
File Type	Route	File Name
Python file	../Python code/8.14：WS2812 RGB lights	microbit-4 WS2812 RGB lights-3.py

You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



4. Test Result

Download code 1 to micro：bit, and dial POWER to ON end. Then the 4 WS2812RGB LEDs light up a different color a time cyclically.

Download code 2 to micro：bit, the WS2812RGB LEDs display like a flow light.

Download code 3 to micro：bit, every WS2812RGB light shows random color one by one.

5. Code Explanation

from microbit **import** *	Import the library file of micro：bit
import neopixel	Import the library file of neopixel
np = neopixel.NeoPixel(pin7, 4)	LED Set Neopixel to pin P7, and initialize 4 LEDs
np.clear()	The RGB lights on the Neopixel strip are all off
while True:	This is a permanent loop that makes micro:bit execute the code of it.
for pixel_id1 **in** range(0, len(np)):	For the RGB pixels in the range of (0, len(np)), pixel_id1
for index **in** range(0, 4):	The RGB pixels in the range (0, 4) are index

np.show()	Display the current pixel on the Neopixel strip
np[pixel_id1] = (255, 0, 0) np[pixel_id2] = (255, 165, 0) np[pixel_id3] = (255, 255, 0) np[pixel_id4] = (0, 255, 0) np[pixel_id5] = (0, 0, 255) np[pixel_id6] = (75, 0, 130) np[pixel_id7] = (238, 130, 238) np[pixel_id8] = (160, 32, 240) np[pixel_id9] = (255, 255, 255)	Set the RGB light on the Neopixel strip to pixel_id1 to turn on the red light; Set the RGB light on the Neopixel strip to pixel_id2 to turn on the orange light; Set the RGB light on the Neopixel strip to pixel_id3 to turn on the yellow light; Set the RGB light on the Neopixel strip to pixel_id4 to turn on the green light; Set the RGB light on the Neopixel strip to pixel_id5 to turn on the blue light; Set the RGB light on the Neopixel strip to pixel_id6 to turn on the indigo light; Set the RGB light on the Neopixel strip to pixel_id7 to turn on the violet light; Set the RGB light on the Neopixel strip to pixel_id8 to turn on the purple light; Set the RGB light on the Neopixel strip to pixel_id 9 to turn on the white light;
from random **import** randint	Import randint from random variables
np[pixel_id] = (R, G, B)	Set the RGB light on the Neopixel strip to pixel_id to turn on colorful light;
R = 0 G = 0 B = 0	Set the initial value of variable R to 0 Set the initial value of variable G to 0 Set the initial value of variable B to 0
R = randint(10, 255) G = randint(10, 255) B = randint(10, 255)	Set R=randint(10, 255) Set G=randint(10, 255) Set B=randint(10, 255)

Project 15：Servo

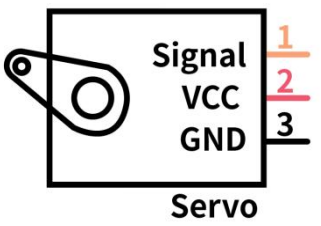
1. Description

The DIY smart cars usually contain the function of automatic obstacle avoidance. In the DIY process, we need a servo to control the ultrasonic module to rotate left and right, and then detect the distance between the car and the obstacle, so as to control the car to avoid the obstacle.

If other microcontrollers are used to control the rotation of the servo, we need to set a certain frequency and width of pulse to control the servo angle. But if the micro:bit main board is used to control the servo angle, we only need to set the control angle in the development environment where the corresponding pulse will be automatically set to control the servo rotation. In this project, you will learn how to control the servo to rotate back and forth between 0° and 90°.

Servo motor is a position control rotary actuator, which mainly consists of housing, circuit board, core-less motor, gear and position sensor. Its working principle is that the servo receives the signal sent by MCU or receiver, and produces a reference signal with a period of 20ms and width of 1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and obtains the voltage difference output.

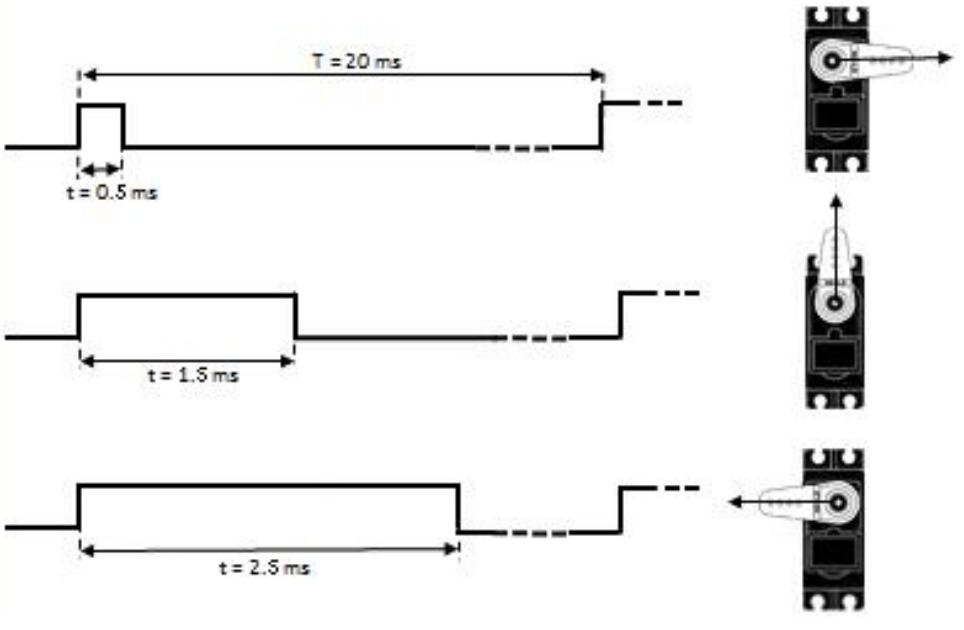
For the servo used in this project, the brown wire is the ground, the red one is the positive wire, and the orange one is the signal wire.



2. Information of the Servo

The rotation angle of servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms (50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds to the rotation angle from 0° to 180°. But note that for different brand motor, the same signal may have different rotation angle.

t	Duty Cycle	Direction
0.5 ms	0.5/20 = 2.5%	0 degs
1.5 ms	1.5/20 = 7.5%	90 degs
2.5 ms	2.5/20 = 12.5%	180 degs



After measurement, the pulse range of the servo is 0.65ms~2.5ms. For a 180 degree servo, the corresponding control relationship is as follow:

Time on High Level	Angle of the Servo	Reference Signal Cycle Time (20ms)
0.65ms	0 degree	0.65ms high level+19.35ms low level
1.5ms	90 degrees	1.5ms high level+18.5ms low level
2.5ms	180degrees	2.5ms high level+17.5ms low level

3. Parameters

- ◆ Working voltage: DC 4.8V ~ 6V
- ◆ Operating angle range: about 180 ° (at 500 → 2500 μsec)
- ◆ Dimension: 22.9*12.2*30mm
- ◆ Pulse width range: 500 → 2500 μsec
- ◆ No-load speed: 0.12 ± 0.01 sec / 60 (DC 4.8V) 0.1 ± 0.01 sec / 60 (DC 6V)
- ◆ No-load current: 200 ± 20mA (DC 4.8V) 220 ± 20mA (DC 6V)
- ◆ Stopping torque: 1.3 ± 0.01kg · cm (DC 4.8V) 1.5 ± 0.1kg · cm (DC 6V)
- ◆ Stop current: ≦ 850mA (DC 4.8V) ≦ 1000mA (DC 6V)
- ◆ Standby current: 3 ± 1mA (DC 4.8V) 4 ± 1mA (DC 6V)
- ◆ Weight: 9±1g (without servo horn)
- ◆ Working temperature: -30℃~60℃

It should be noted that do not use a computer for power supply, because if the current demand is greater than 500mA, the servo may be burned out. It is recommended to use an external battery for power supply.

4. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to computer via an USB cable
- Open the offline version of Mu.

5. Test Code

Enter Mu software and open the file "Project 15：Servo.py" to import code:

(How to load the project code?)

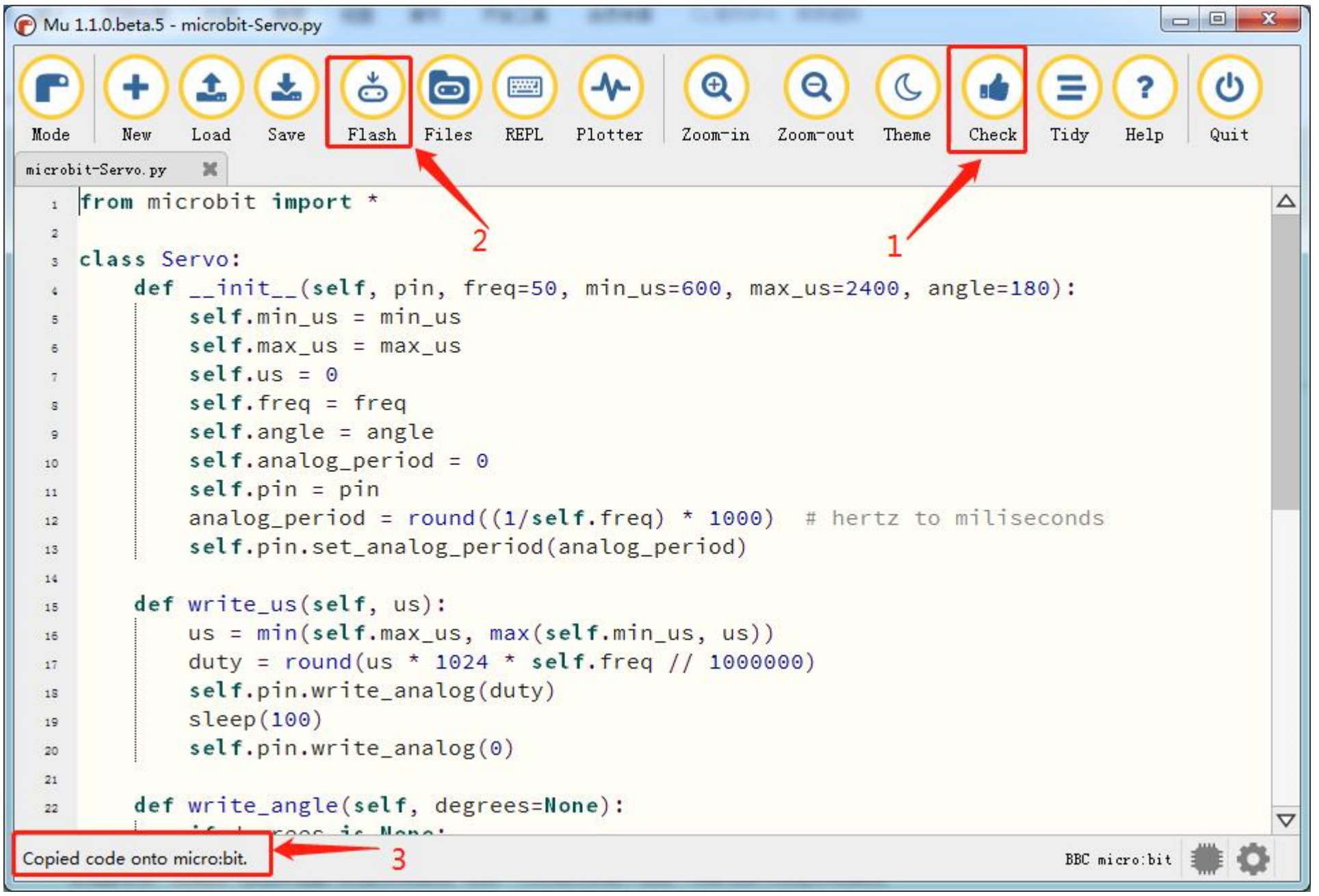
File Type	Route	File Name
Python file	../Python code/8.15：Servo	microbit-Servo.py

You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.

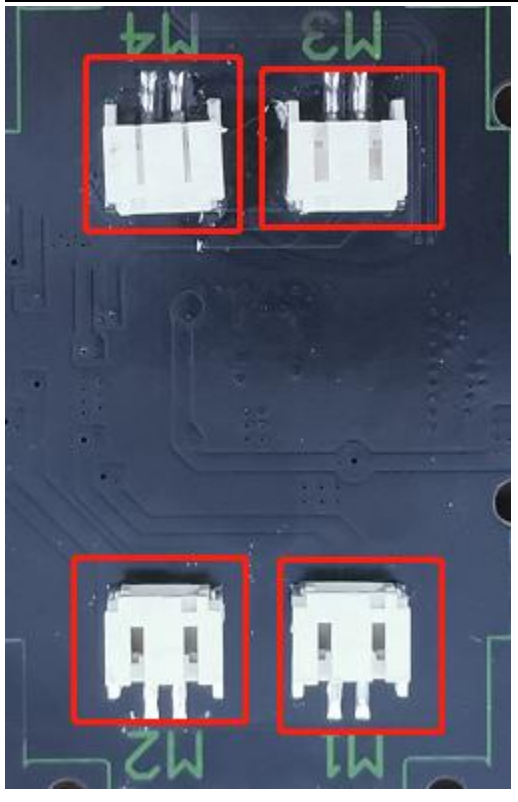



```
from microbit import *

class Servo:
    def __init__(self, pin, freq=50, min_us=600, max_us=2400, angle=180):
        self.min_us = min_us
        self.max_us = max_us
        self.us = 0
        self.freq = freq
        self.angle = angle
        self.analog_period = 0
        self.pin = pin
        analog_period = round((1/self.freq) * 1000)  # hertz to miliseconds
        self.pin.set_analog_period(analog_period)

    def write_us(self, us):
        us = min(self.max_us, max(self.min_us, us))
        duty = round(us * 1024 * self.freq // 1000000)
        self.pin.write_analog(duty)
        sleep(100)
        self.pin.write_analog(0)

    def write_angle(self, degrees=None):
```

4. Test Result

After uploading the test code, dialing POWER switch to ON end and powering it by external power , the LED dot matrix shows a smiley pattern and the servo rotates in the pattern 0°~45°~90°~135°~180°~0°.

Project 16：Motor



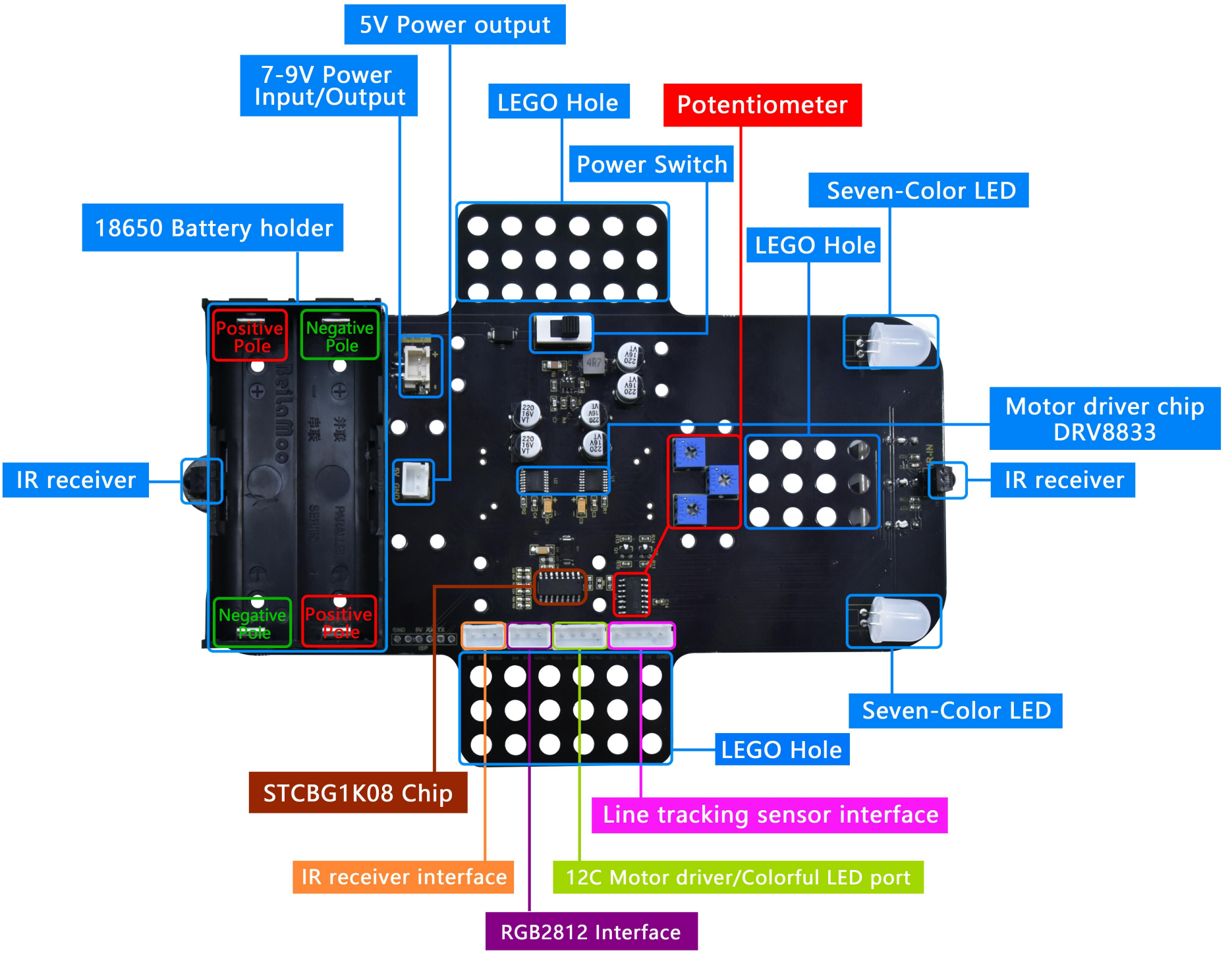
1. Description

The Keyestudio 4WD Mecanum Robot Car is equipped with 4 DC reduction motors, also called gear reduction motor, which is developed on the ordinary DC motor. It has a matching gear reduction box which provides a lower speed but a larger torque. Furthermore, different reduction ratios of the box can provide different speeds and torques.

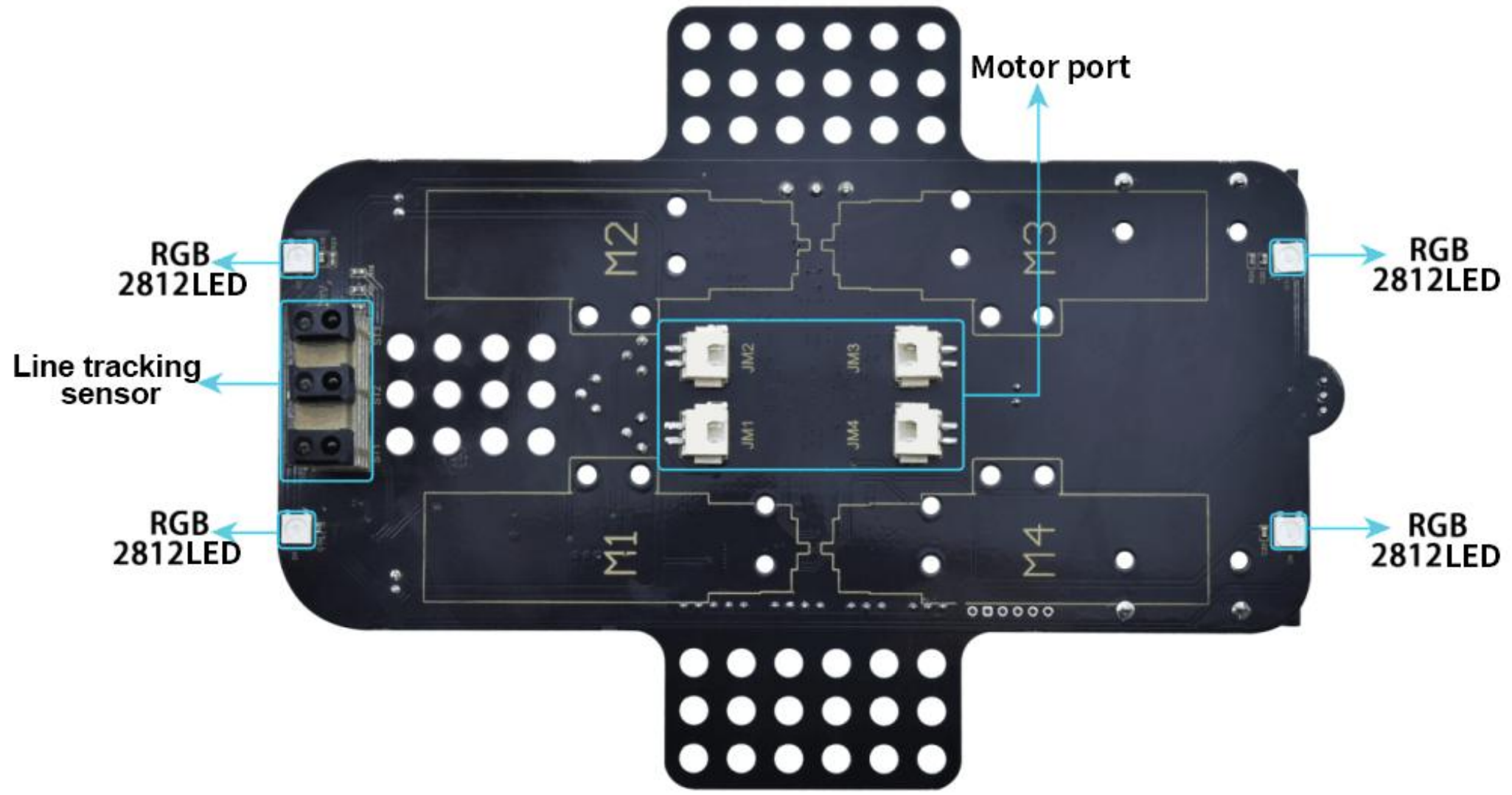
Gear motor is the integration of gearmotor and motor, which is applied widely in steel and machine industry

Micro:bit motor driver shield comes with a STC8G and HR8833 chip. In order to save the IO port resource, we control the rotation direction and speed of 4 DC gear motors with the HR8833 chip.

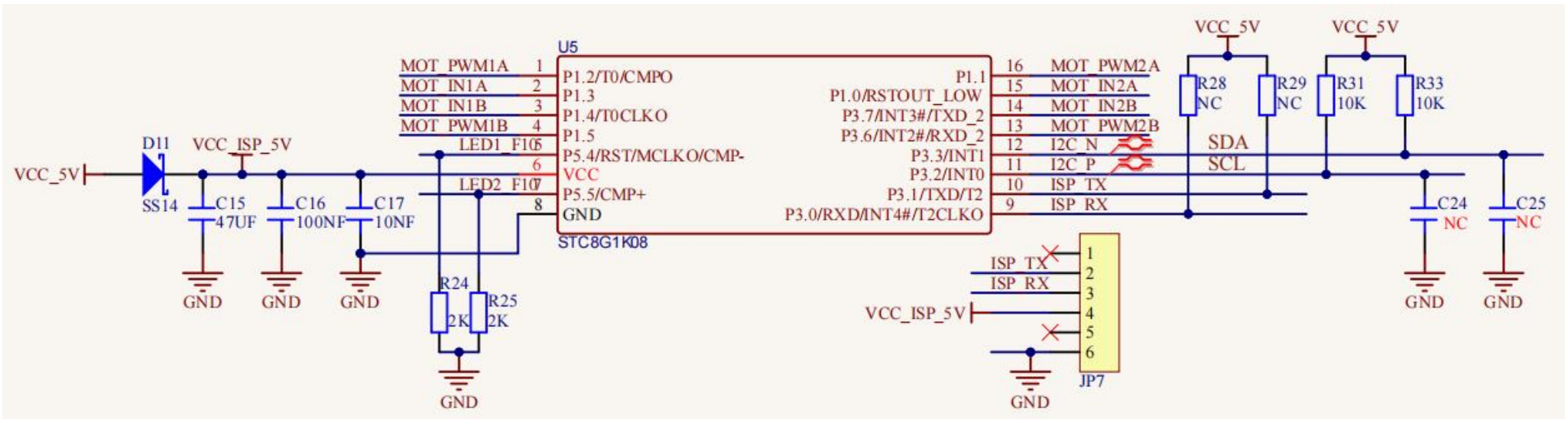
Details about chips:


5V Power output
7-9V Power Input/Output
LEGO Hole
Potentiometer
Power Switch
Seven-Color LED
18650 Battery holder
LEGO Hole
Positive Pole
Negative Pole
Motor driver chip DRV8833
IR receiver
IR receiver
Negative Pole
Positive Pole
Seven-Color LED
LEGO Hole
STCBG1K08 Chip
Line tracking sensor interface
IR receiver interface
12C Motor driver/Colorful LED port
RGB2812 Interface

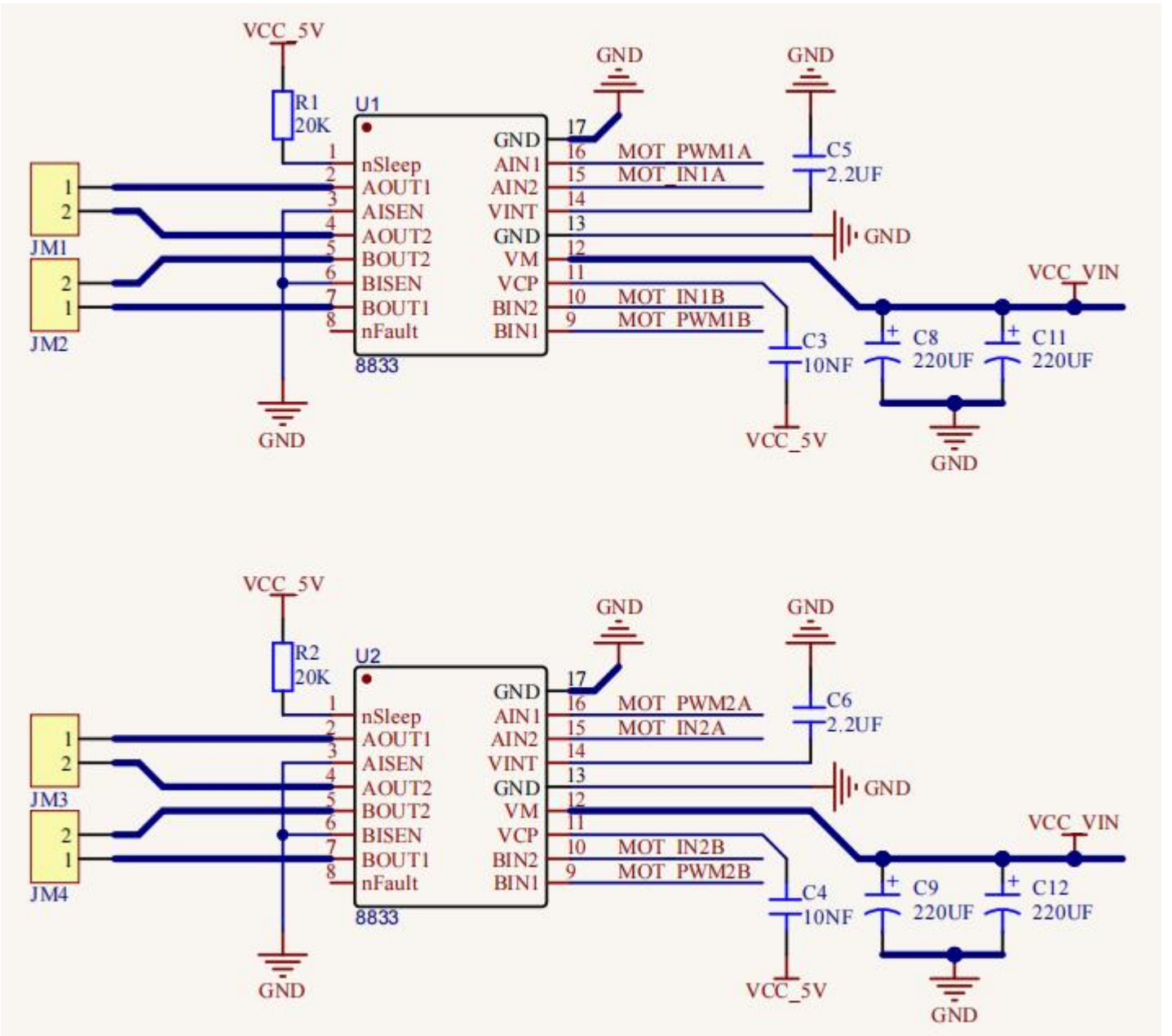
Front



Back



STC8G1K08 Chip circuit



HR8833 Motor driver circuit

2. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to the computer via an USB cable
- Open the offline version of Mu.

3. Test Code

Code 1：

Enter Mu software and open the file “Project 16：Motor.py” to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.16：Motor Driving	microbit-Motor Driving-1.py

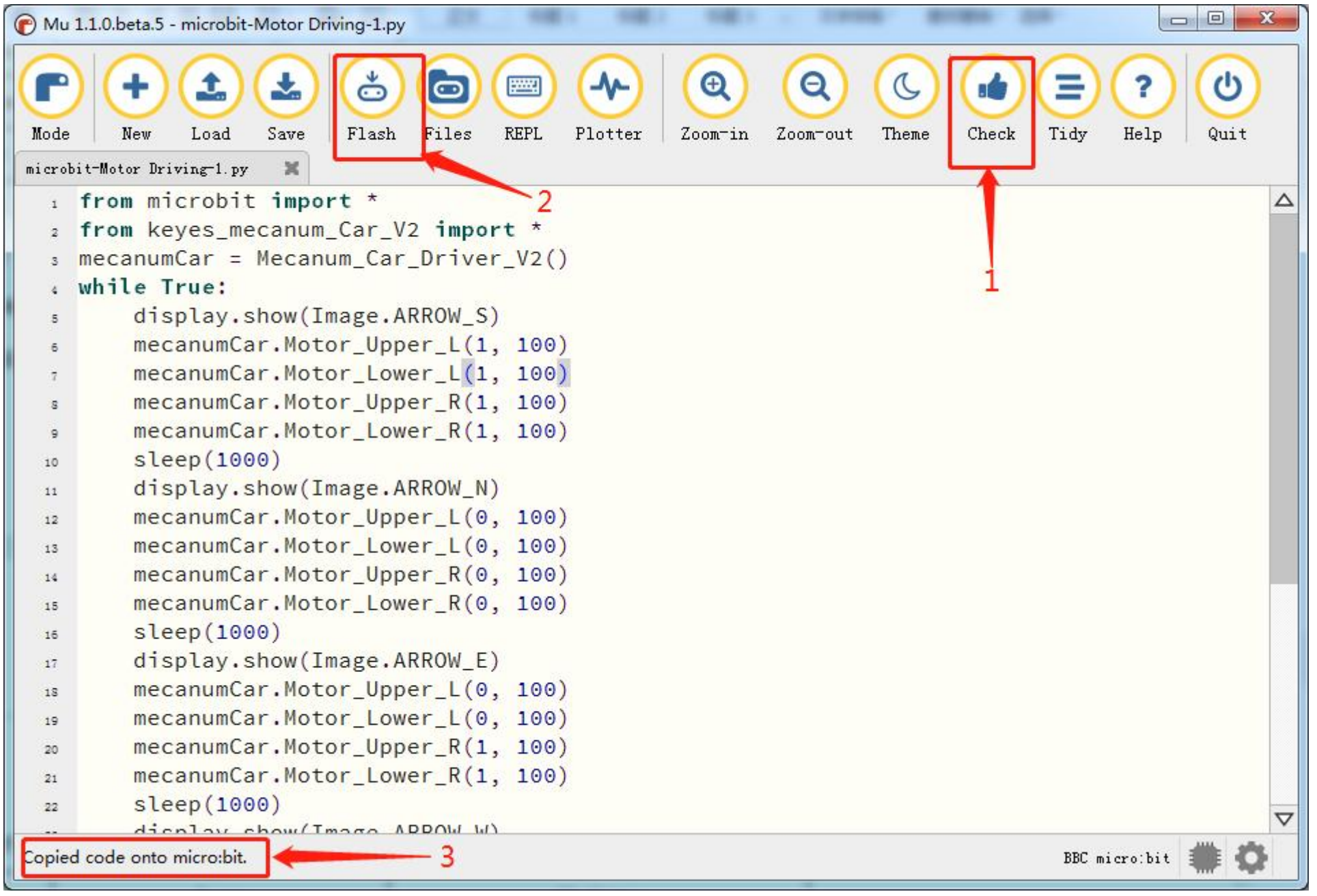
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Files" to import "keyes_mecanum_Car.py "library file to micro:bit (How to import files?).

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



Code 2:

Enter Mu software and open the file "Project 16: Motor.py" to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.16: Motor Driving	microbit-Motor Driving-2.py

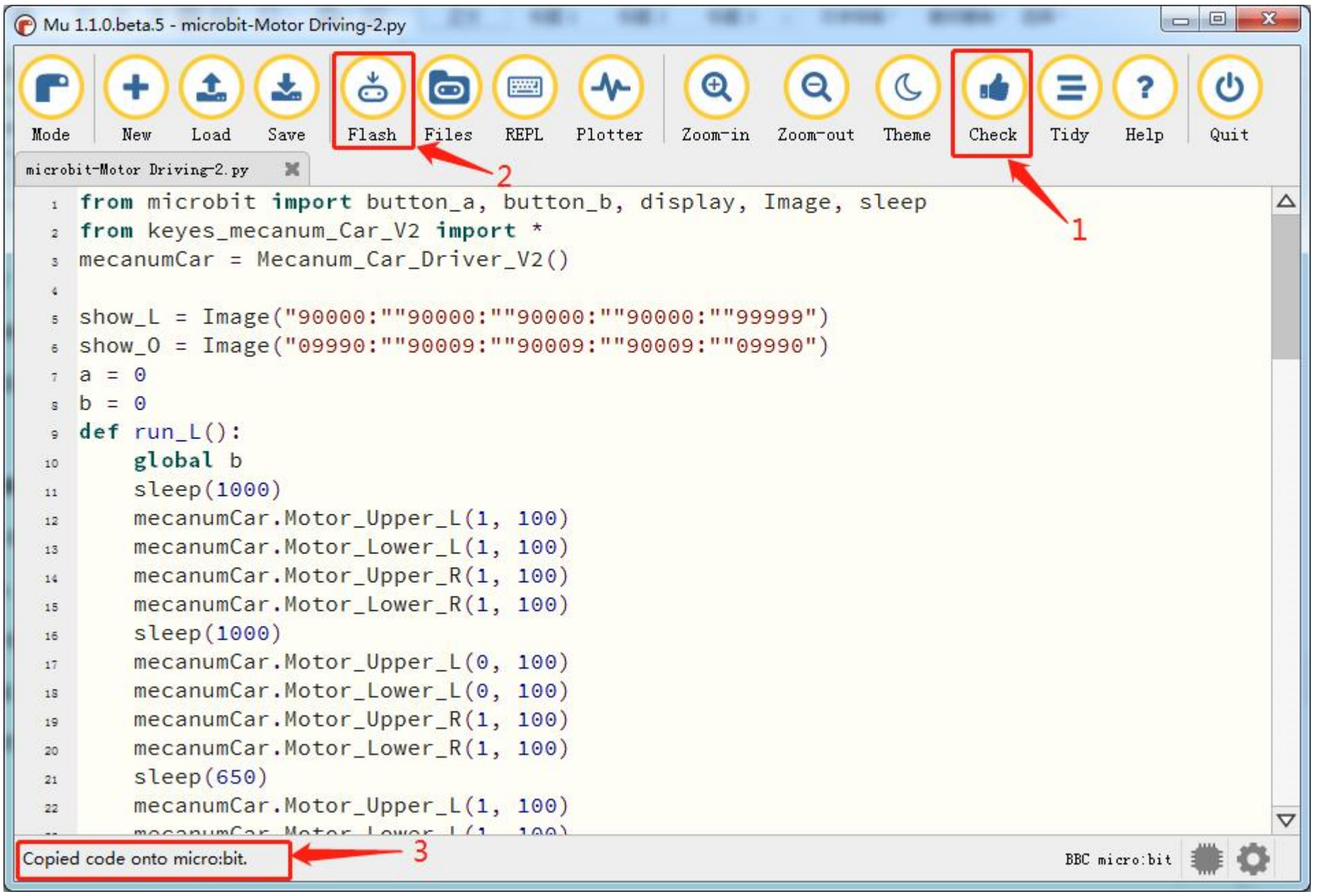
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click “Files” to import “keyes_mecanum_Car.py “library file to micro:bit (How to import files?).

Click “Check” to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click “Flash” to download the code to the micro:bit board.



4. Test Result

Download code 1 to micro:bit, and turn on the switch of the smart car. Then the car will go forward for 1s, back for 1s, turn left for 1s, right for 1s, turn anticlockwise for 1s, clockwise for 1 and stop 1s. Matrix also displays the patterns.

Download code 2 to micro:bit board, dial POWER switch to ON end. When the button A and B are firstly pressed, micro:bit will show “L”, the route of the car is “L”. When they are pressed again, “□” is shown on micro:bit, and route of the car is “□”. The car will repeat this pattern.

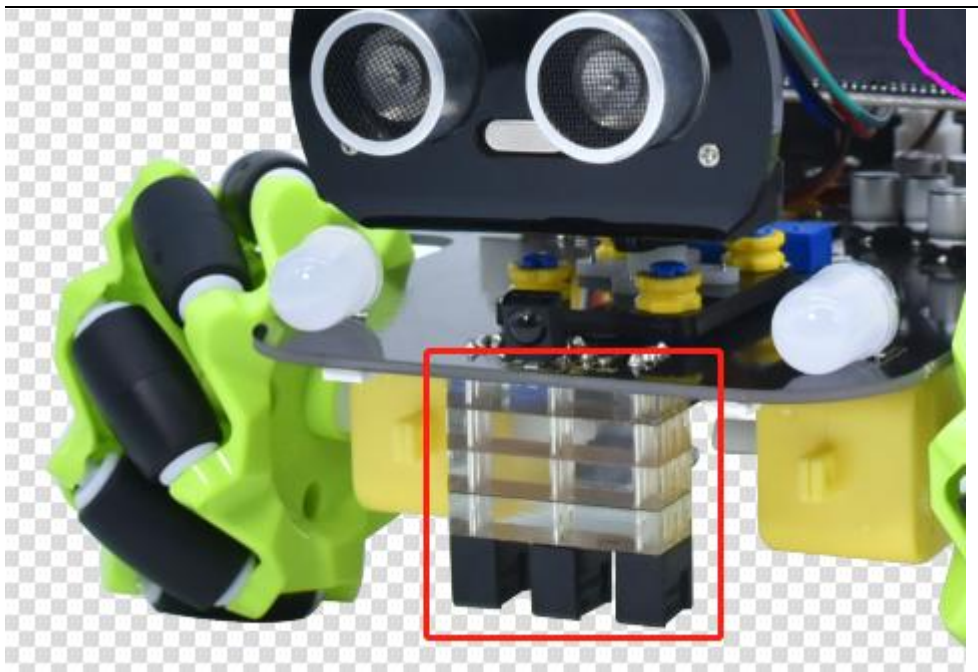
5. Code Explanation

from microbit **import** button_a, button_b, display, Image, sleep	Due to insufficient memory, only the necessary parts such as button_a, button_b, display, Image, sleep and so on in the micro:bit library file are imported here
from keyes_mecanum_Car **import ***	Import library file keyes_mecanum_Car
mecanumCar =Mecanum_Car_Driver()()	Instantiate an object Mecanum_Car_Driver()() as mecanumCar
while True:	This is a permanent loop that makes micro:bit execute the code of it.
display.show(Image.ARROW_S) display.show(Image.ARROW_N) display.show(Image.ARROW_E) display.show(Image.ARROW_W) display.show(Image("00900:""09990:""99999:""99999:""09090"))	micro:bit shows arrow pointing to South micro:bit shows arrow pointing to North micro:bit shows arrow pointing to East micro:bit shows arrow pointing to West micro:bit displays “❤”
mecanumCar.Motor_Upper_L(1, 100) mecanumCar.Motor_Lower_L(1, 100) mecanumCar.Motor_Upper_R(1, 100) mecanumCar.Motor_Lower_R(1, 100)	The left motor of car rotates clockwise at the speed of PWM100 （1: clockwise，0: anticlockwise；PWM100 means speed（0~255）） The rear left motor of car rotates clockwise at the speed of PWM100. The front right motor of car rotates clockwise at the speed of PWM100. The rear right motor of car rotates clockwise at the speed of PWM100.
mecanumCar.Motor_Upper_L(0, 100) mecanumCar.Motor_Lower_L(0, 100) mecanumCar.Motor_Upper_R(0, 100)	The front left motor of car rotates anticlockwise at the speed of PWM100.

mecanumCar.Motor_Lower_R(0, 100)	The rear left motor of car rotates anticlockwise at the speed of PWM100. The front right motor of car rotates anticlockwise at the speed of PWM100. The rear right motor of car rotates anticlockwise at the speed of PWM100.
sleep(1000)	Delay in 1000ms
a = 0 b = 0	Set the initial value of variable a to 0 Set the initial value of variable b to 0
def run_L(): **def** run_O():	Define sub-function run_L() Define sub-function run_O()
show_L = Image("90000:""90000:""90000:""90000:""99999")	Assign Image() to the variable show_L
if button_a.was_pressed(): a = a + 1 **if** a >= 3: a = 0 **if** button_b.was_pressed(): b = 1 **if** (a == 1): display.show(show_L) **if** b == 1: run_L() **elif** a == 2: display.show(show_O) **if** b == 1: run_O()	if button A is pressed, a = a + 1 If a≥3 a=0 If button B is pressed, b=1 If a=1 micro:bit shows “L” pattern If b=1 The track of car is route L If a=2 micro:bit shows “O” image If b=1 The track of car is route O

Project 17：Line Tracking Sensor

Project 17.1：Detect Line Tracking Sensor



1. Description

The motor driver board of the Keyestudio 4WD Mecanum Robot Car comes with a 3-channel line tracking sensor, which adopts TCRT5000 IR tubes and 3 potentiometers.

The TCRT5000 IR tube contains an IR emitting tube and an IR receiving tube. When the infrared signals of the emitting tube is received by the receiving tube through reflection, the resistance of the receiving tube will change, which is generally reflected in the voltage change on the circuit.

The resistance varies depending on the intensity of the infrared signals received by the receiving tube, which is often in the color of the reflecting surface and the distance of the reflecting surface receiving tube. At the time of detection, black is high level active and white is low level active.

2. Working Principle

When the car runs above a white road, the IR emitting tube installed under the car emits infrared signals to detect the road and the receiving tube will receive signals sending back. Then the output end outputs low level(0); when it detects black lines, it outputs high level(1).

The 3-channel tracking sensor integrated port on the 4WD Mecanum Robot Car is connected to the collection port of G ,5V ,P10, P4 and P3 on the micro:bit expansion board, which is controlled by the P10, P4 and P3 of the micro:bit. The left TCRT5000 infrared pair tube on the sensor is controlled by P3, the middle one is by P4 and the right one is by P10.

After putting a white paper on the bottom of the 4WD Mecanum Robot Car, we will rotate the potentiometers on the 3-way tracking sensor. When the indicator light on the sensor module is on, pick up the car to make the two wheels on the 4WD Mecanum Robot Car separate. The height of the white paper is about 1.5cm, when the indicator light on the sensor module is off, and then the sensitivity is adjusted.

Note that since the 5*5 dot matrix uses the P3P4P6P7P10, we must turn off the dot matrix function when using the line tracking sensor.

3. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to computer via an USB cable
- Open the offline version of Mu.

4. Test Code

Code 1:

Enter Mu software and open the file "Project 17: Line Tracking Sensor.py" to import code:

(How to load the project code?)

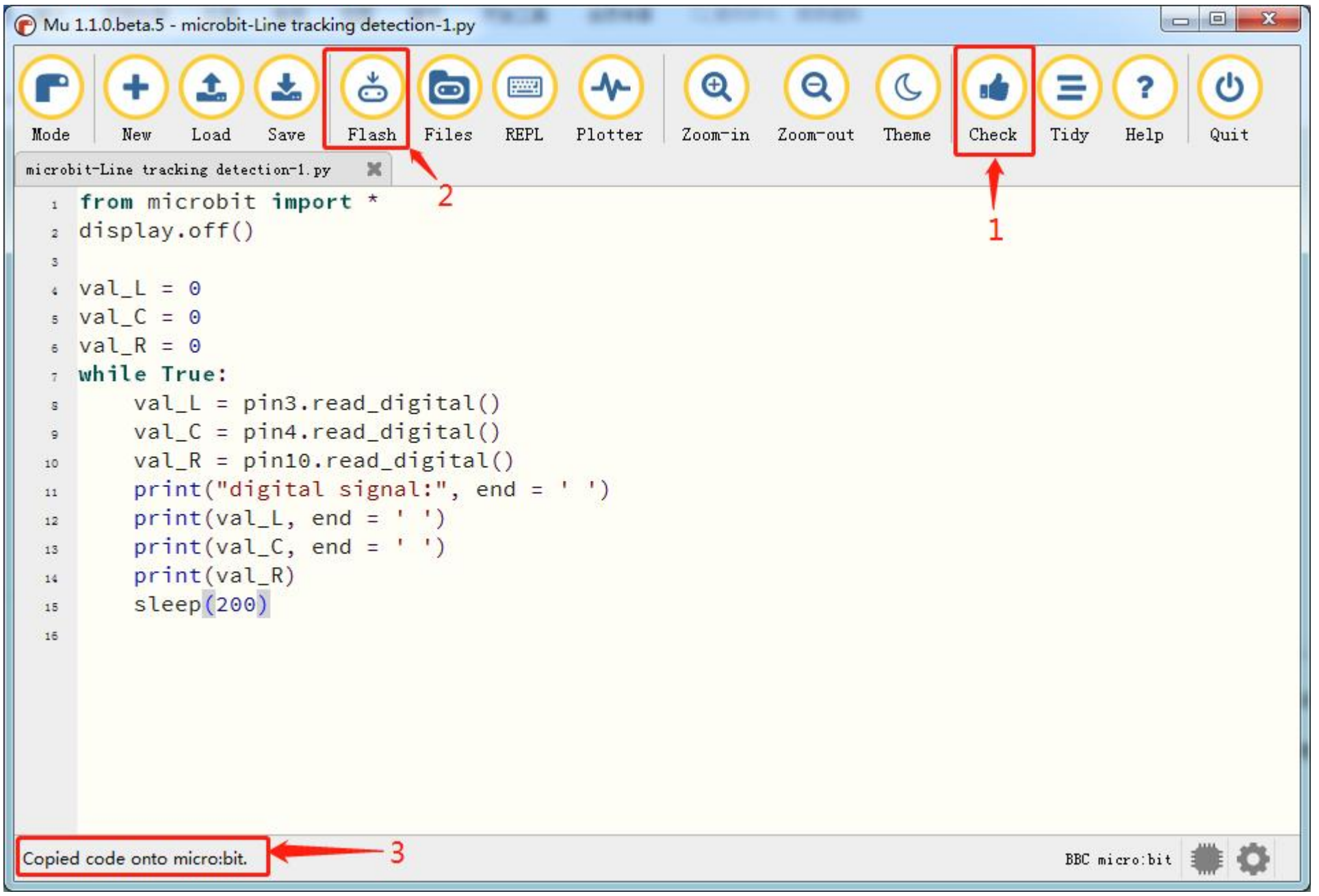
File Type	Route	File Name
Python file	../Python code/8.17: Line Tracking Car/8.17.1	microbit-Line tracking detection-1.py

You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

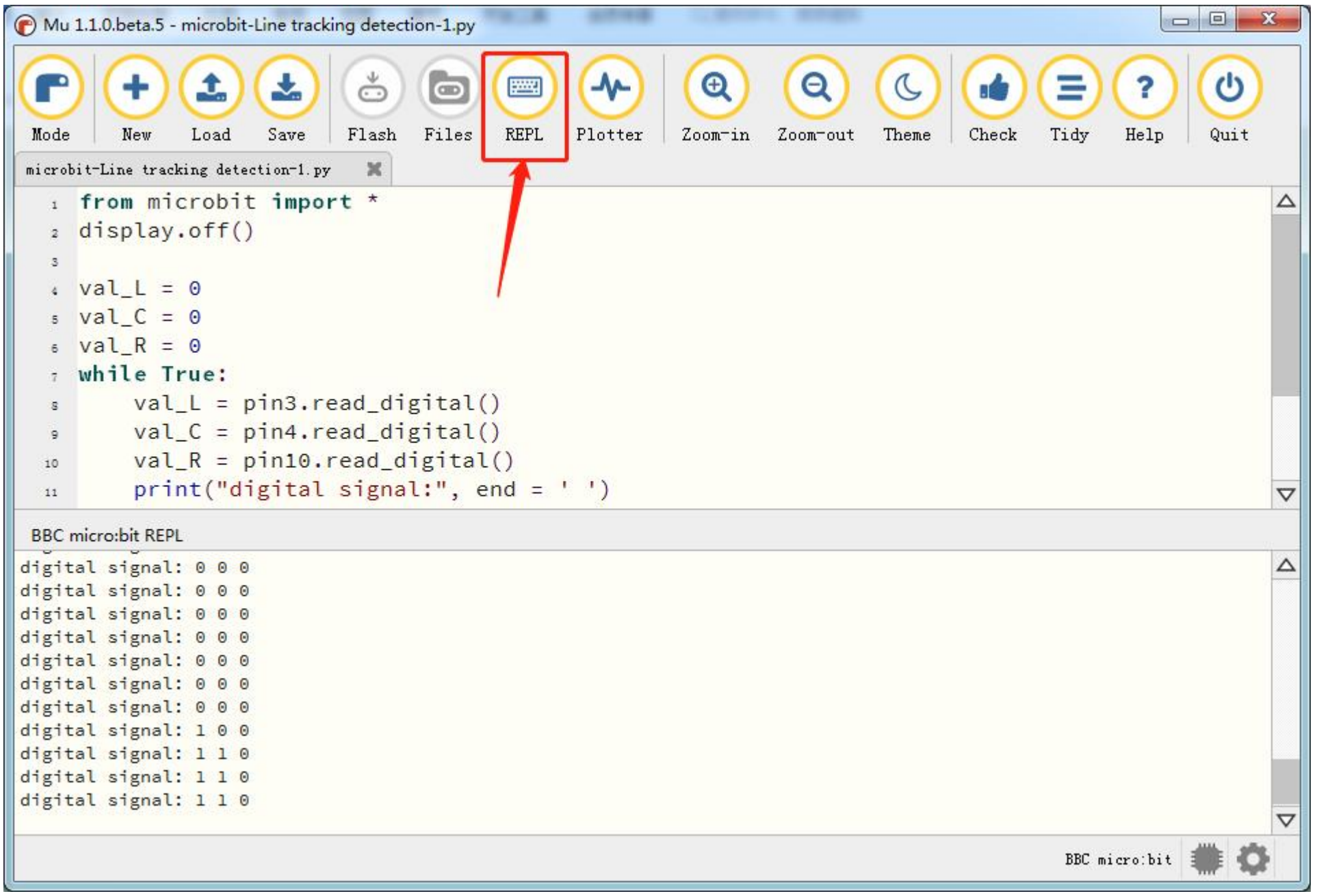
If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



5. Test Result

Download code 1 onto micro:bit board and don’ t plug off the USB cable. Click “REPL” and then press the reset button, the readings detected by the left TCRT5000 IR tube will be displayed on monitor.

When the left TCRT5000 IR tube detects the white object, 0 will be shown and the left indicator will be on; when there is only black object detected, 1 will be displayed and the indicator will be off, as shown below:

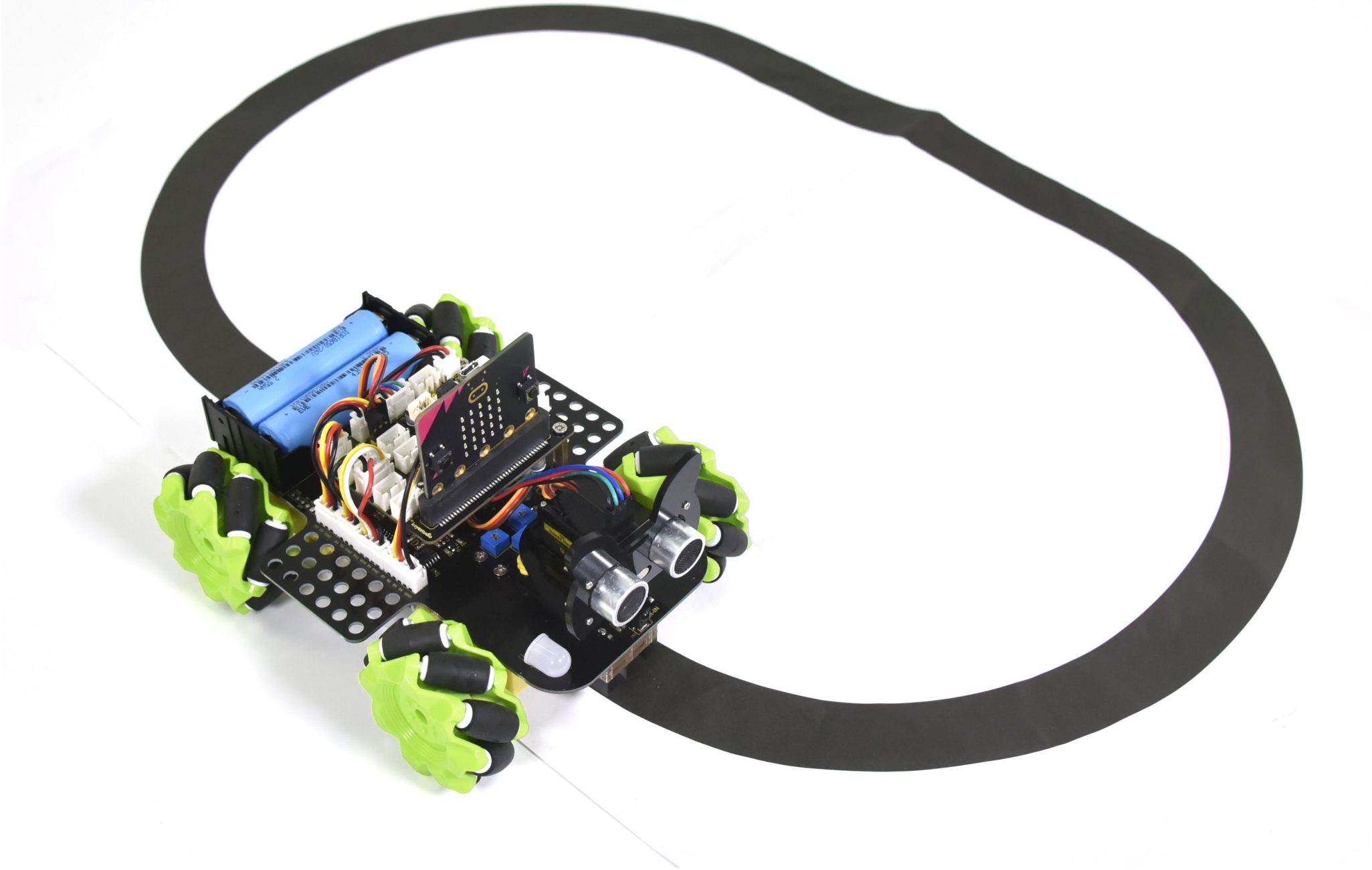


6. Code Explanation

from microbit **import** *	Import the library file of micro:bit
val_L = 0 val_C = 0 val_R = 0	Set the initial value to 0
while True:	This is a permanent loop that makes micro:bit execute the code of it.
val_L = pin3.read_digital() val_C = pin4.read_digital() val_R = pin10.read_digital()	Set the digital signal read by TCRT5000 IR tube connected to P1 to val_L Set the digital signal read by TCRT5000 IR tube connected to P1 to val_C Set the digital signal read by TCRT5000 IR tube connected to P1 to val_R
print("digital signal:", end = ' ')	Print " **digital signal**" without newline

print(val_L, end = ' ')	Print signal value **val_L** without newline
print(val_C, end = ' ')	Print signal value **val_C** without newline
print(val_R)	Print signal value **val_R** with newline
sleep(200)	Delay in 200ms

Project 17.2：Tracking Smart Car



1. Description

In this lesson we will combine a line tracking sensor with a motor to make a line tracking smart car.

The micro:bit board will analyze the signals and control the smart car to show the line tracking function.

2. Working Principle

The smart car will make different moves according to the value received by the 3-channel line tracking sensor.

Left/Middle/Right TCRT5000 IR Tunes（Level）	4WD Mecanum Robot Car

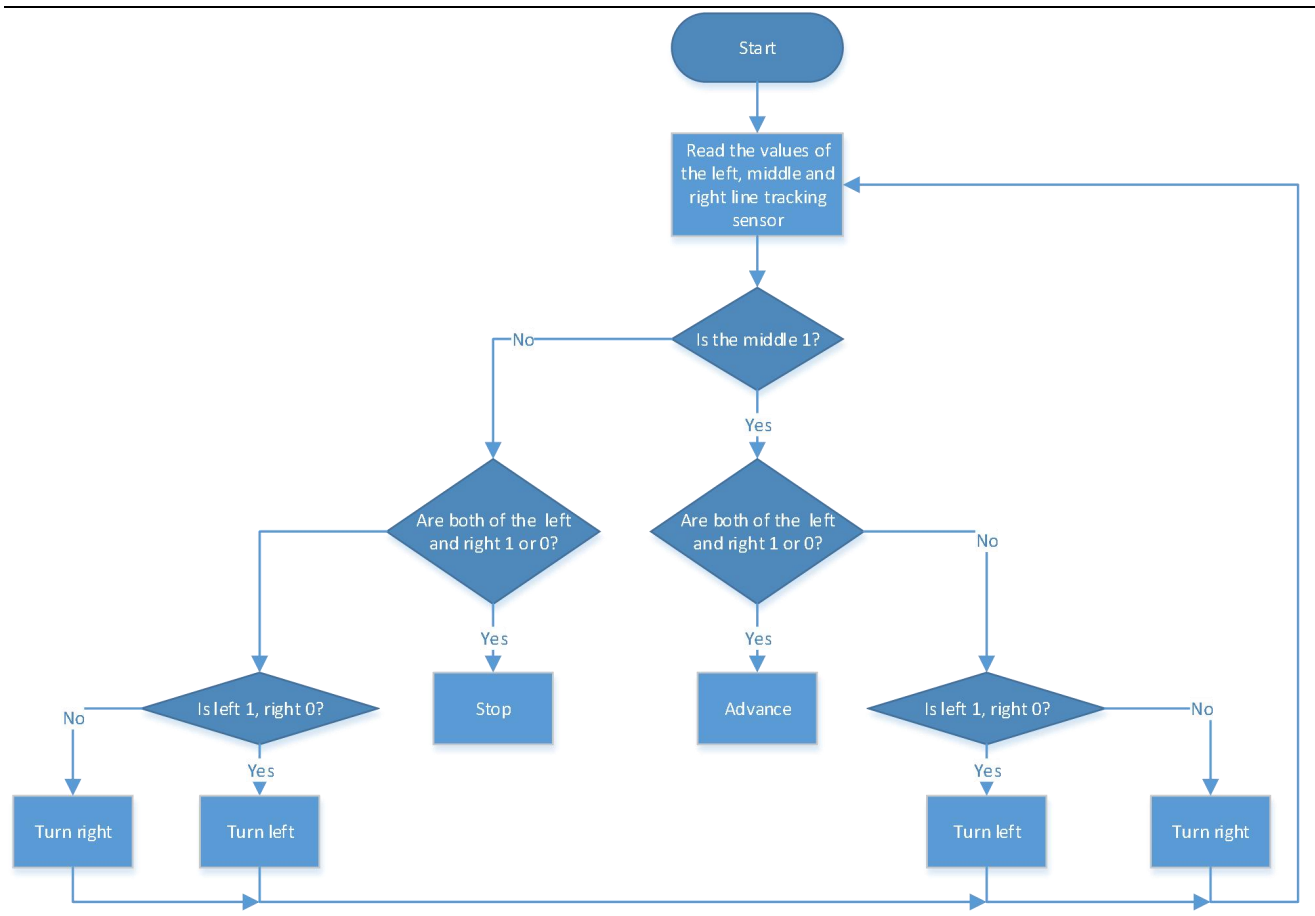
LOW（0）	LOW（0）	HIGH（1）	Turn Right
HIGH（1）	LOW（0）	LOW（0）	Turn Left
HIGH（1）	LOW（0）	HIGH（1）	Stop
LOW（0）	LOW（0）	LOW（0）	Stop
LOW（0）	HIGH（1）	HIGH（1）	Turn Right
HIGH（1）	HIGH（1）	LOW（0）	Turn Left
HIGH（1）	HIGH（1）	HIGH（1）	Go forward
LOW（0）	HIGH（1）	LOW（0）	Go forward

3. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to computer via an USB cable
- Open the offline version of Mu.

Warning: The 3-way tracking sensor should be used in environments without infrared interference such as sunlight. Sunlight contains a lot of invisible light, such as infrared and ultraviolet. In an environment with strong sunlight, the 3-way tracking sensor cannot work properly.

4. Flow Chart



5. Test Code

Enter Mu software and open the file "Project 17: Line Tracking Sensor.py" to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.17: Line Tracking Car/8.17.2	microbit- Line tracking car.py

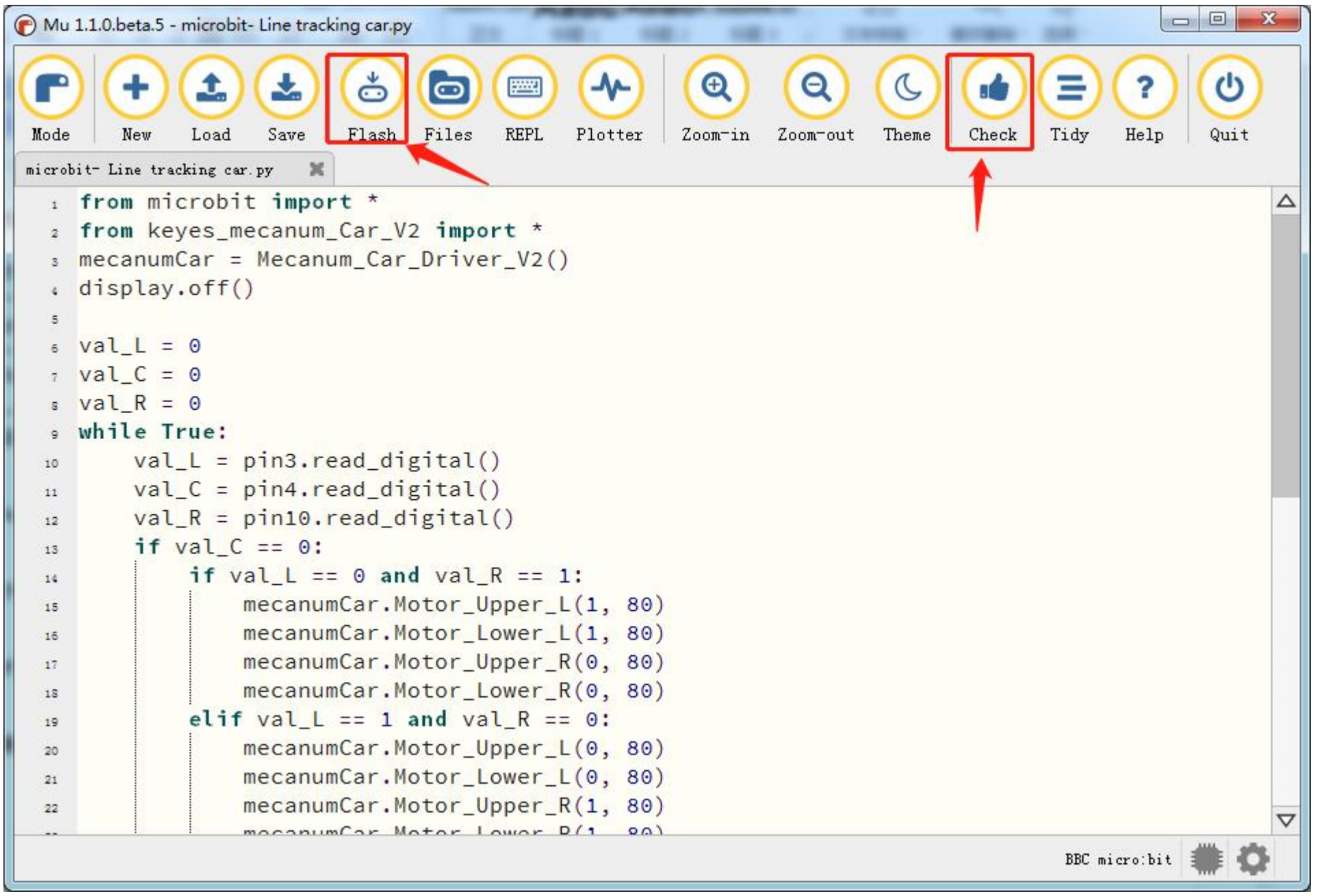
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Files" to import "keyes_mecanum_Car.py "library file to micro:bit (How to import files?).

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



6. Test Result

Download the code to the micro:bit and dial POWER to ON end, then the line tacking car goes forward along the black line .

Note: (1) The width of black line should be equal to or larger than the width of the line tracking sensor when tracking.

(2) Avoid to test the smart car under the strong light.

7. Code Explanation

from microbit **import ***	Import the library of micro:bit
from keyes_mecanum_Car_V2 **import ***	Import the library of keyes_mecanum_Car V2
mecanumCar = Mecanum_Car_Driver_V2()	Instantiate an object Mecanum_Car_V2Driver() as mecanumCar
display.off()	The LED dot matrix on the micro:bit will turn off the display
while True:	This is a permanent loop that makes micro:bit execute the code of it.
val_L = pin3.read_digital()	Assign the digital signal read by the TCRT5000 infrared pair

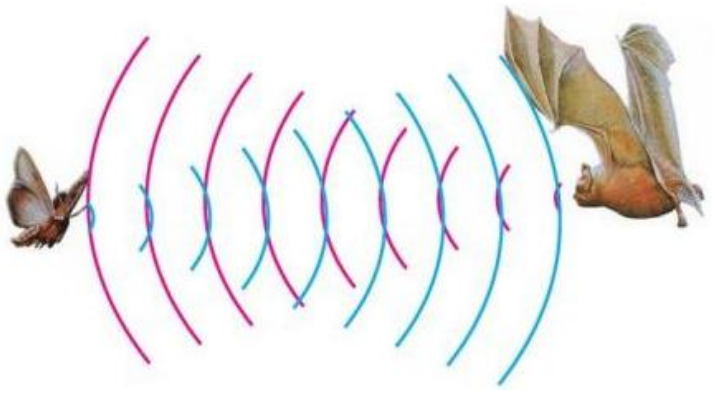
	tube connected to the P1 control port to the variable val_L
val_C = pin4.read_digital()	Assign the digital signal read by the TCRT5000 infrared pair tube connected to the P1 control port to the variable val_C
val_R = pin10.read_digital()	Assign the digital signal read by the TCRT5000 infrared pair tube connected to the P1 control port to the variable val_R
if val_C == 0: **if val_L == 0 and val_R == 1:** mecanumCar.Motor_Upper_L(1, 80) mecanumCar.Motor_Lower_L(1, 80) mecanumCar.Motor_Upper_R(0, 80) mecanumCar.Motor_Lower_R(0, 80) **elif val_L == 1 and val_R == 0:** mecanumCar.Motor_Upper_L(0, 80) mecanumCar.Motor_Lower_L(0, 80) mecanumCar.Motor_Upper_R(1, 80) mecanumCar.Motor_Lower_R(1, 80) **else:** mecanumCar.Motor_Upper_L(0, 0) mecanumCar.Motor_Lower_L(0, 0) mecanumCar.Motor_Upper_R(0, 0) mecanumCar.Motor_Lower_R(0, 0) **else:** **if val_L == 0 and val_R == 1:** mecanumCar.Motor_Upper_L(1, 80) mecanumCar.Motor_Lower_L(1, 80) mecanumCar.Motor_Upper_R(0, 80) mecanumCar.Motor_Lower_R(0, 80) **elif val_L == 1 and val_R == 0:** mecanumCar.Motor_Upper_L(0, 80)	**if middle is 0,** **if val_L = 0 and val_R = 1** the left front motor of car rotates clockwise at the speed of PWM80; the left rear motor of car rotates clockwise at the speed of PWM80; the right front motor of car rotates anticlockwise at the speed of PWM80; the right rear motor of car rotates anticlockwise at the speed of PWM80; **if val_L = 1 and val_R = 0** the left front motor of car rotates anticlockwise at the speed of PWM80; the left rear motor of car rotates anticlockwise at the speed of PWM80; the right front motor of car rotates clockwise at the speed of PWM80; the right rear motor of car rotates clockwise at the speed of PWM80; **If val_L = 0 and val_R = 0 or val_L = 1 and val_R = 1** the left front motor of car rotates at the speed of 0 and stops; the left rear motor of car rotates at the speed of 0 and stops; the right front motor of car rotates at the speed of 0 and stops; the right rear motor of car rotates at the speed of 0 and stops; **if middle is 1,** **If val_L = 0 and val_R =1**

mecanumCar.Motor_Lower_L(0, 80) mecanumCar.Motor_Upper_R(1, 80) mecanumCar.Motor_Lower_R(1, 80) **else:** mecanumCar.Motor_Upper_L(1, 80) mecanumCar.Motor_Lower_L(1, 80) mecanumCar.Motor_Upper_R(1, 80) mecanumCar.Motor_Lower_R(1, 80)	the left front motor of car rotates clockwise at the speed of PWM80; the left rear motor of car rotates clockwise at the speed of PWM80; the right front motor of car rotates anticlockwise at the speed of PWM80; the right rear motor of car rotates anticlockwise at the speed of PWM80; **If val_L = 1 or val_R =0** the left front motor of car rotates anticlockwise at the speed of PWM80; the left rear motor of car rotates anticlockwise at the speed of PWM80; the right front motor of car rotates clockwise at the speed of PWM80; the right rear motor of car rotates clockwise at the speed of PWM80; **If val_L = 0 and val_R = 0 or val_L = 1 and val_R = 1** the left front motor of car rotates clockwise at the speed of PWM80; the left rear motor of car rotates clockwise at the speed of PWM80; the right front motor of car rotates clockwise at the speed of PWM80; the right rear motor of car rotates clockwise at the speed of PWM80;

Project 18：Ultrasonic Sensor

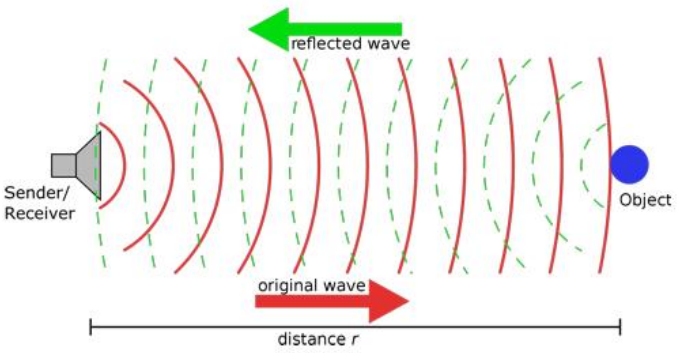
Project 18.1：Ultrasonic Ranging

1. Description



The ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.

The ultrasonic sensor is being used in a wide range of electronics projects for creating obstacle detection and distance measuring application as well as various other applications.

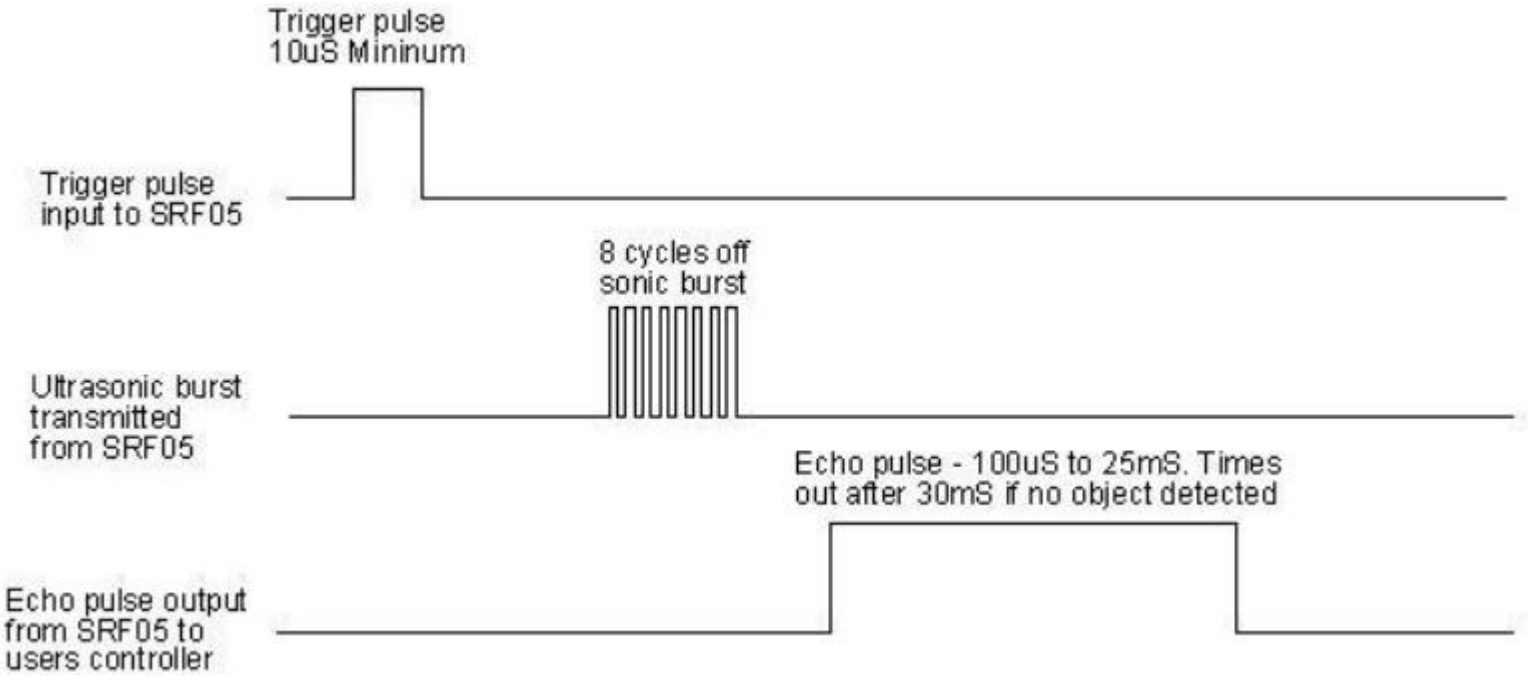


The ultrasonic module will emit the ultrasonic waves after trigger signals. When the ultrasonic waves encounter the object and are reflected back, the module outputs an echo signal, so it can determine the distance of object from the time difference between trigger signal (TRIG)and echo signal(ECHO).

As the picture shows, it is like two eyes. One is transmitting end, the other is receiving end.

According to the above wiring diagram, the integrated port of the ultrasonic sensor module is connected to the 5V G P15 P16 port on the micro:bit motor driver base plate. The Trig (T) pin is controlled by P15 of the micro:bit and the pin of Echo (E) the P16.

2. Working Principle



(1)Pull down TRIG then trigger high level signals with least 10us;

(2)After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return;

(3)If there is a signal return, when ECHO (E) outputs a high level, then the duration of the high level is the time from transmission to reception of the ultrasonic waves. Then test distance = high level duration *340m/s*0.5.

3. Parameters

- Working voltage: 3-5.5V (DC)
- Working current: 15mA
- Working frequency: 40khz
- Maximum detection distance: about 3m
- Minimum detection distance: 2-3cm
- Precision: up to 0.2cm
- Sensing angle: less than 15 degrees
- Input trigger pulse: 10us TTL level
- Output echo signal: output TTL level signal (high), proportional to range

4. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to the computer via an USB cable
- Open the offline version of Mu.

5. Test Code

Enter Mu software and open the file "Project 18.1: Ultrasonic Ranging.py" to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.18: Ultrasonic/8.18.1: Ultrasonic Ranging	microbit- Ultrasonic Ranging.py

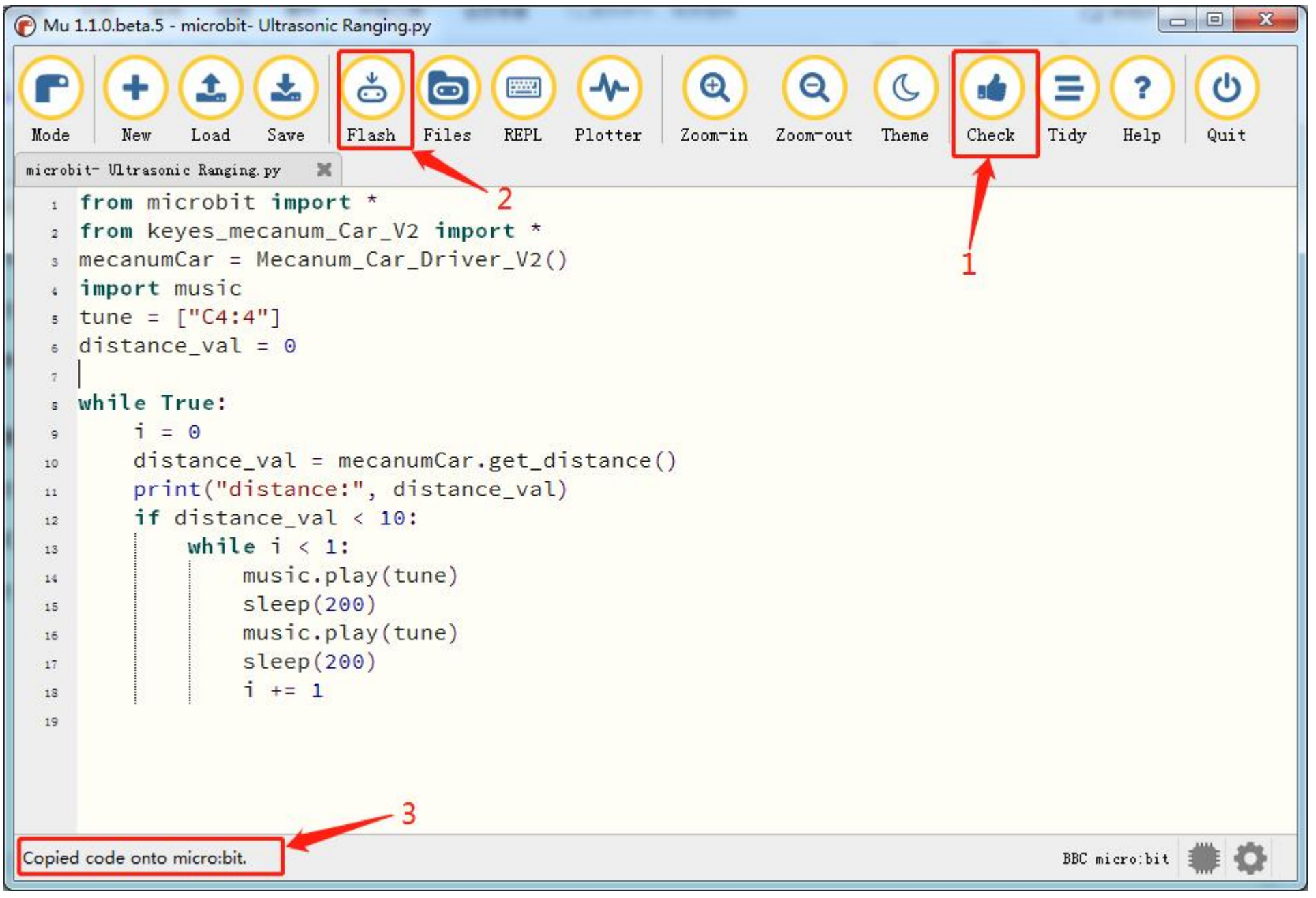
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Files" to import "keyes_mecanum_Car_V2.py "library file to micro:bit (How to import files?).

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

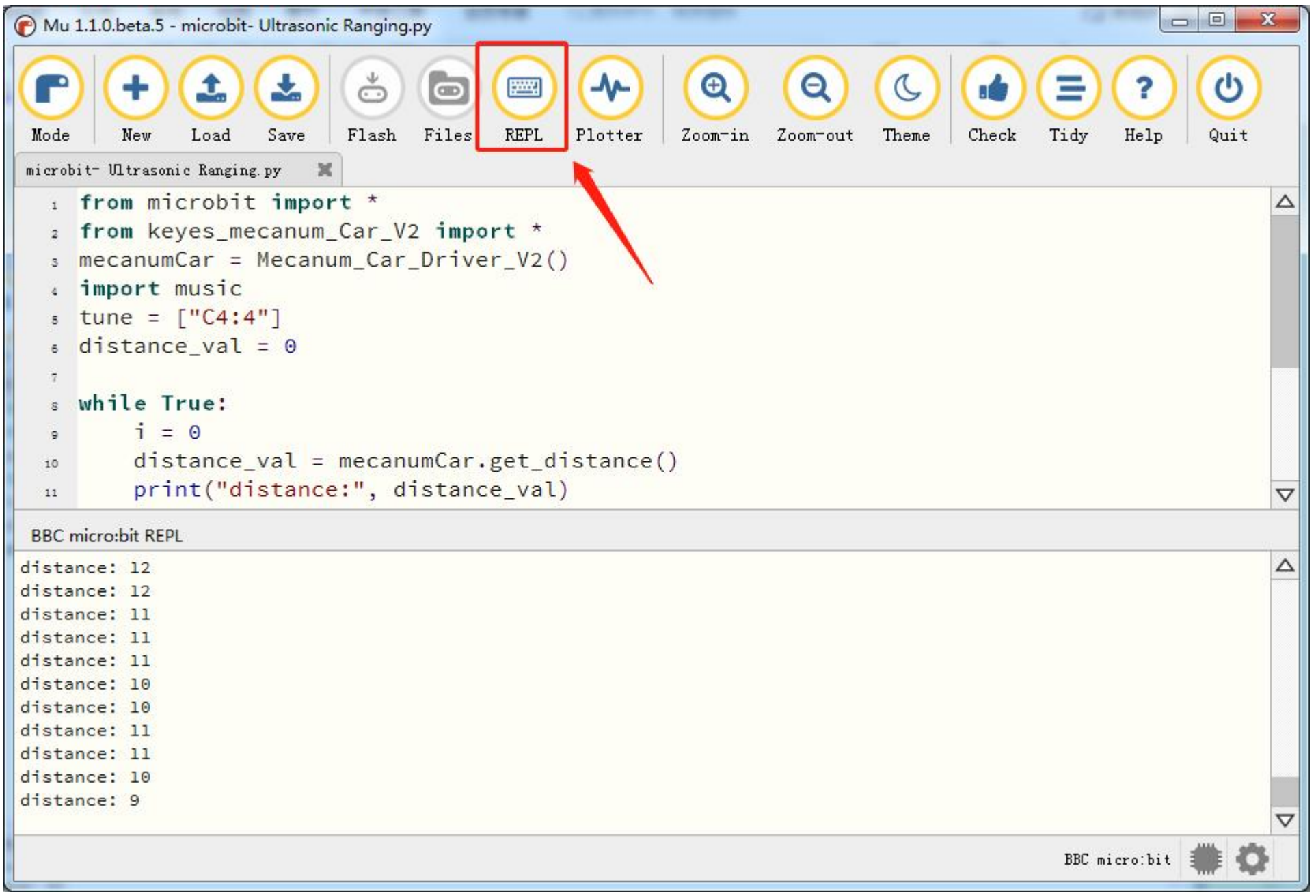
If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



6. Test Result

Download the code to micro:bit board and don’ t plug off the USB cable. Click “REPL” and then press the reset button, the distance value of obstacle will be displayed, as shown below.

When the distance is less than 10cm, the passive buzzer of smart will emit sound.

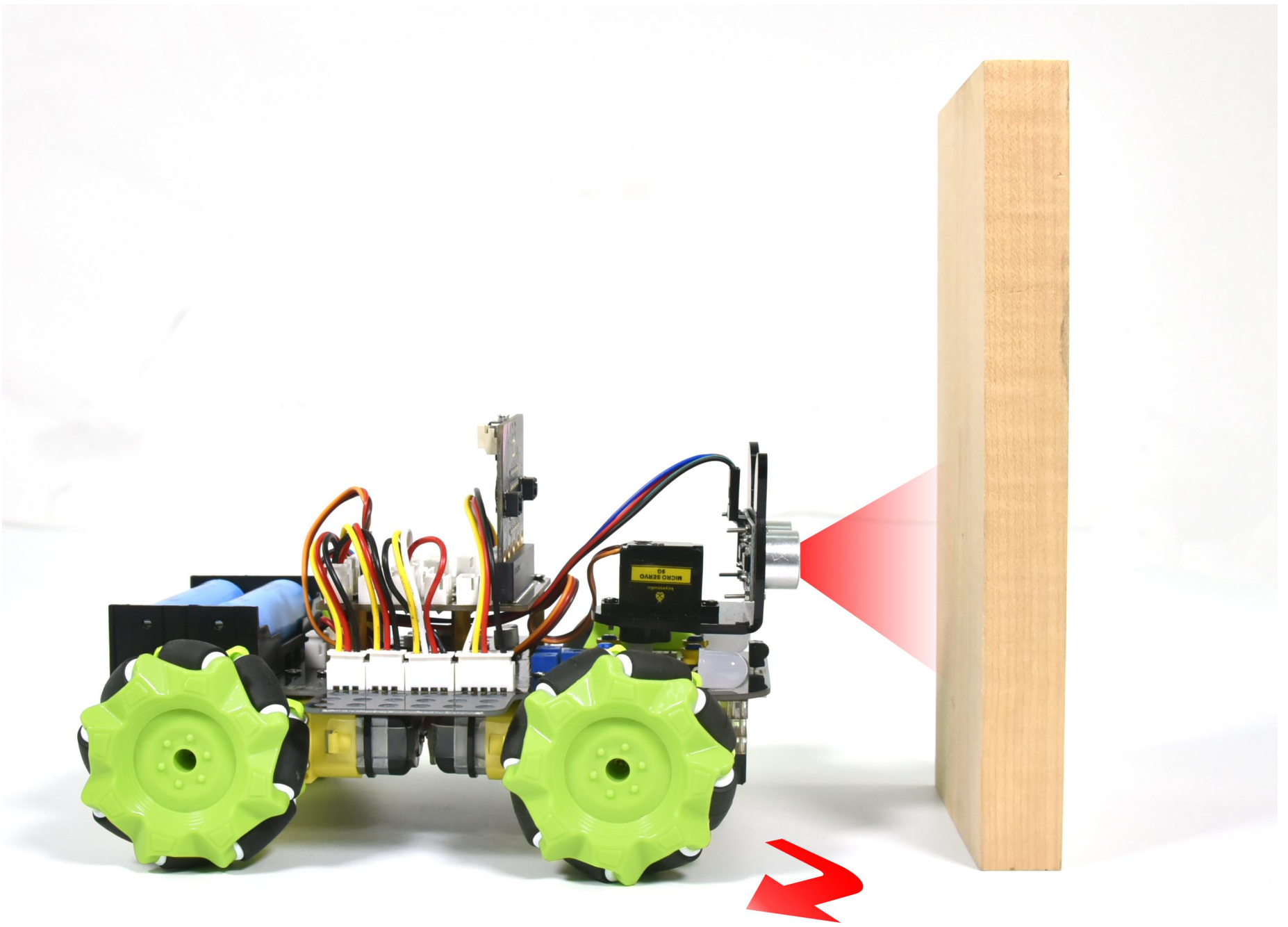


7. Code Explanation

from microbit **import ***	Import the library of micro:bit
from keyes_mecanum_Car **import ***	Import the library of keyes_mecanum_Car_V2
mecanumCar = Mecanum_Car_Driver()()	instantiate Mecanum_Car_Driver_V2()to mecanumCar
import music	Import the library of music
tune = ["C4:4"]	Create tune to save
while True:	This is a permanent loop that makes micro:bit execute the code of it.
i = 0	Set variable i=0

distance_val = mecanumCar.get_distance()	Assign mecanumCar.get_distance()to variable distance_val
print("distance:", distance_val)	BBC microbit REPL window shows the distance value between the ultrasonic sensor and the obstacle
if distance < 10:	if distance < 10
while i < 1:	When i < 1
music.play(tune) sleep(200) music.play(tune) sleep(200)	Passive buzzer emits sound
i += 1	Variable i adds 1 gradually

Project 18.2：Ultrasonic Avoidance



1. Description

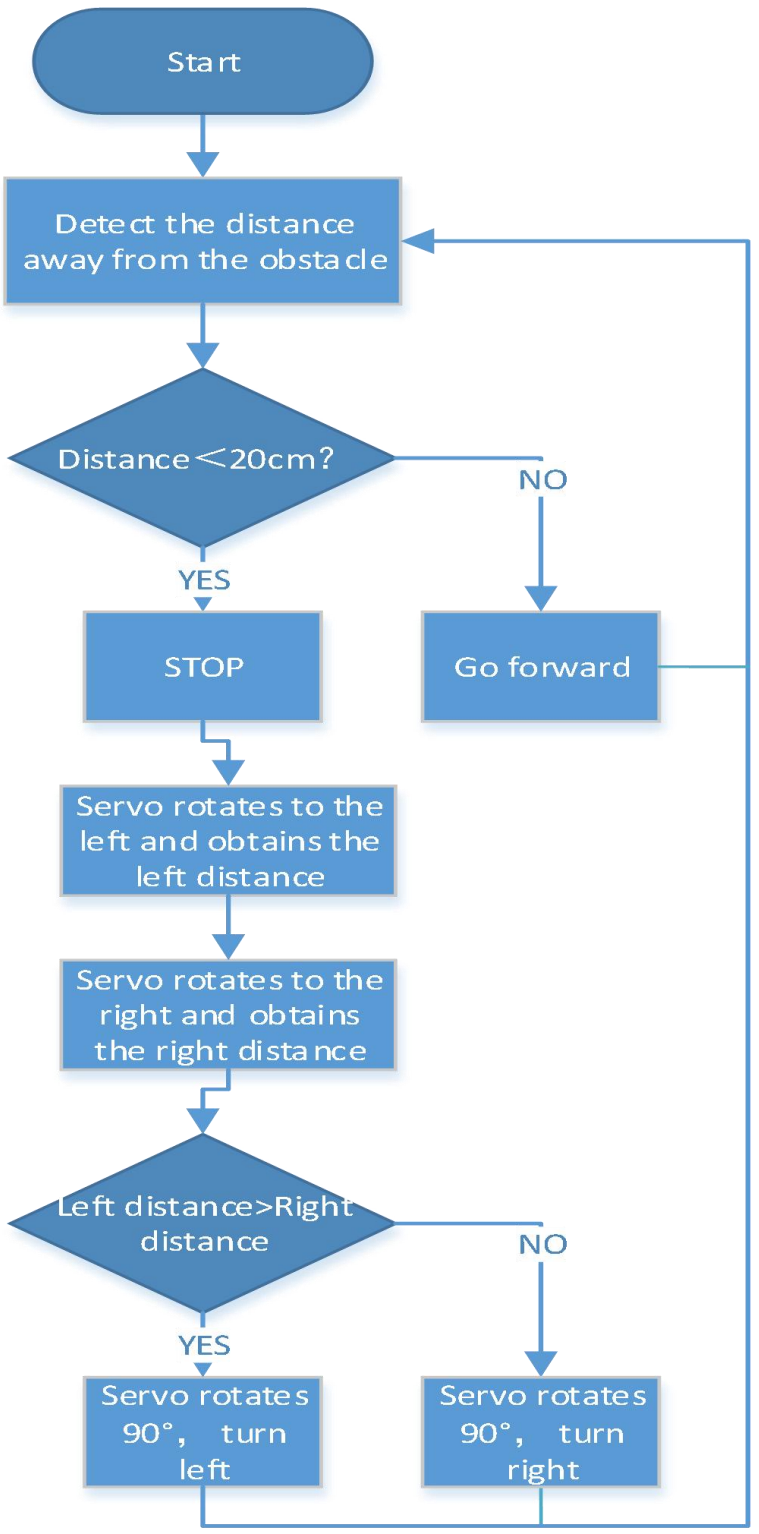
In this project, we will integrate an ultrasonic sensor and a car to make an ultrasonic avoidance car.

Its principle is to detect the distance between the car and obstacle via the ultrasonic sensor to control the motion of smart car.

2. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to the computer via an USB cable
- Open the offline version of Mu.

3. Flow Chart



4. Test Code

Enter Mu software and open the file "Project 18.2: Ultrasonic Avoid Smart Car.py" to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.18: Ultrasonic/8.18.2: Ultrasonic Avoidance Car	microbit- Ultrasonic Avoid Smart Car.py

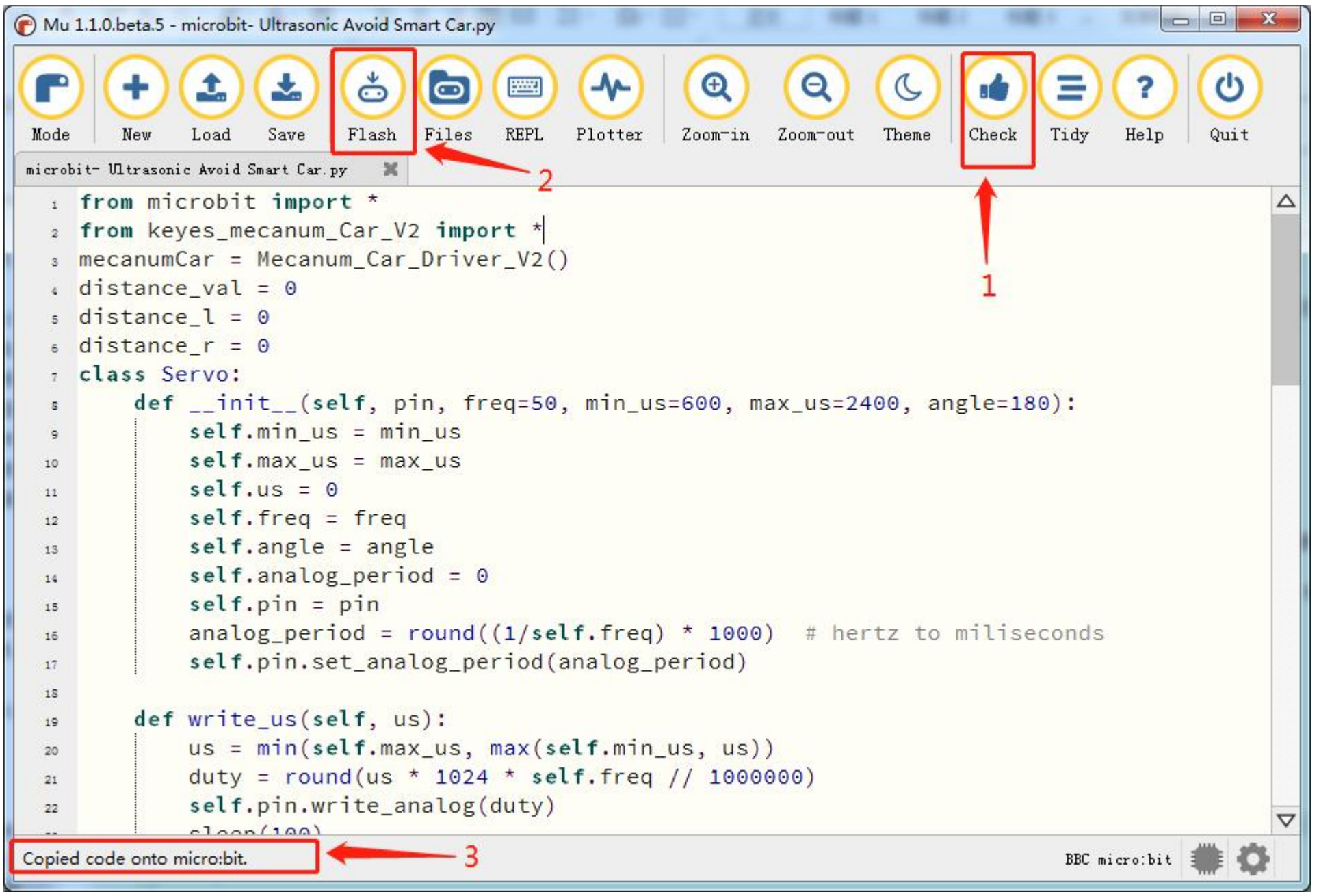
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Files" to import "keyes_mecanum_Car_V2.py "library file to micro:bit (How to import files?).

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



5. Test Result

Download the code to micro:bit, and dial POWER to ON end. When the obstacle distance is greater than 20cm, the car goes forward ; on the contrary, the smart car turns left.

6. Code Explanation

from microbit **import ***	Import the library of micro:bit
from keyes_mecanum_Car **import ***	Import the library of keyes_mecanum_Car
mecanumCar = Mecanum_Car_Driver()()	Instantiate Mecanum_Car_Driver()()to mecanumCar
distance_val = 0	Set the initial value of the variable distance_val to 0
distance_l = 0	Set the initial value of the variable distance_l to 0
distance_r = 0	Set the initial value of the variable distance_r to 0

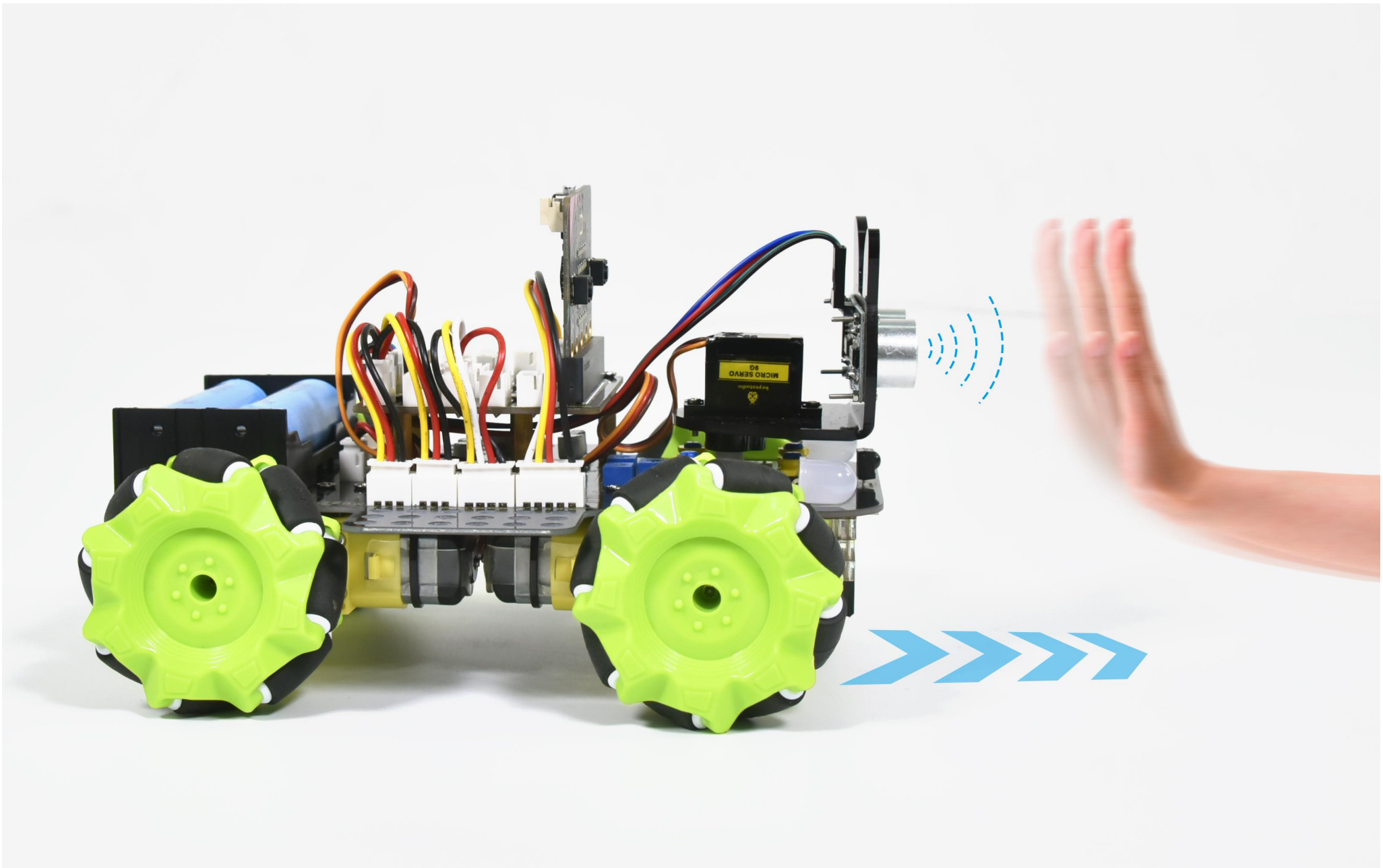
Servo(pin14).write_angle(90)	The steering gear is connected to P14, the rotation angle is 90 degrees
while True:	This is a permanent loop that makes micro:bit execute the code of it.
distance_val = mecanumCar.get_distance()	Assign mecanumCar.get_distance()to variable distance_val

if distance_val < 20:
mecanumCar.Motor_Upper_L(0, 0)
mecanumCar.Motor_Lower_L(0, 0)
mecanumCar.Motor_Upper_R(0, 0)
mecanumCar.Motor_Lower_R(0, 0)
sleep(500)
Servo(pin14).write_angle(180)
distance_l = mecanumCar.get_distance()
Servo(pin14).write_angle(0)
distance_r = mecanumCar.get_distance()
if distance_l > distance_r:
mecanumCar.Motor_Upper_L(0, 150)
mecanumCar.Motor_Lower_L(0, 150)
mecanumCar.Motor_Upper_R(1, 150)
mecanumCar.Motor_Lower_R(1, 150)
Servo(pin14).write_angle(90)
sleep(300)
else:
mecanumCar.Motor_Upper_L(1, 150)
	If distance_val <20 is established (stop)
	The left front motor of the Mecanum wheel smart car stops rotating;
	The left rear motor of the Mecanum wheel smart car stops rotating;
	The right front motor of the Mecanum wheel smart car stops rotating;
	The right rear motor of the Mecanum wheel smart car stops rotating;
	Delay in 500ms
	The servo connected to P14 rotates to 180 degrees;
	Assign mecanumCar.get_distance() to the variable distance_l;
	The servo connected to P14 turns to 0 degrees;
	Assign mecanumCar.get_distance() to the variable distance_r;
	If distance_l> distance_r condition is true (turn left)
	The left front motor of car rotates anticlockwise at the speed of PWM100.
	The left rear motor of car rotates

mecanumCar.Motor_Lower_L(1, 150) mecanumCar.Motor_Upper_R(0, 150) mecanumCar.Motor_Lower_R(0, 150) Servo(pin14).write_angle(90) sleep(300) else: mecanumCar.Motor_Upper_L(1, 150) mecanumCar.Motor_Lower_L(1, 150) mecanumCar.Motor_Upper_R(1, 150) mecanumCar.Motor_Lower_R(1, 150)	anticlockwise at the speed of PWM100. The right front motor of car rotates clockwise at the speed of PWM100. The right rear motor of car rotates clockwise at the speed of PWM100. The servo connected to P14 turns to 90 degrees; Delay in 300ms; If distance_l> distance_r condition is not true (turn right), The left front motor of car rotates clockwise at the speed of PWM100. The left rear motor of car rotates anticlockwise at the speed of PWM100. The right front motor of car rotates clockwise at the speed of PWM100. The right rear motor of car rotates anticlockwise at the speed of PWM100. The servo connected to P14 turns to 90 degrees; Delay in 300ms; If condition distance_val <20 is not true (move forward) The left motor front of car rotates clockwise at the speed of PWM100. The left rear motor of car rotates clockwise at the speed of PWM100. The right front motor of car rotates clockwise at the speed of PWM100.

	The right rear motor of car rotates clockwise at the speed of PWM100.

Project 18.3：Ultrasonic Following



1. Description

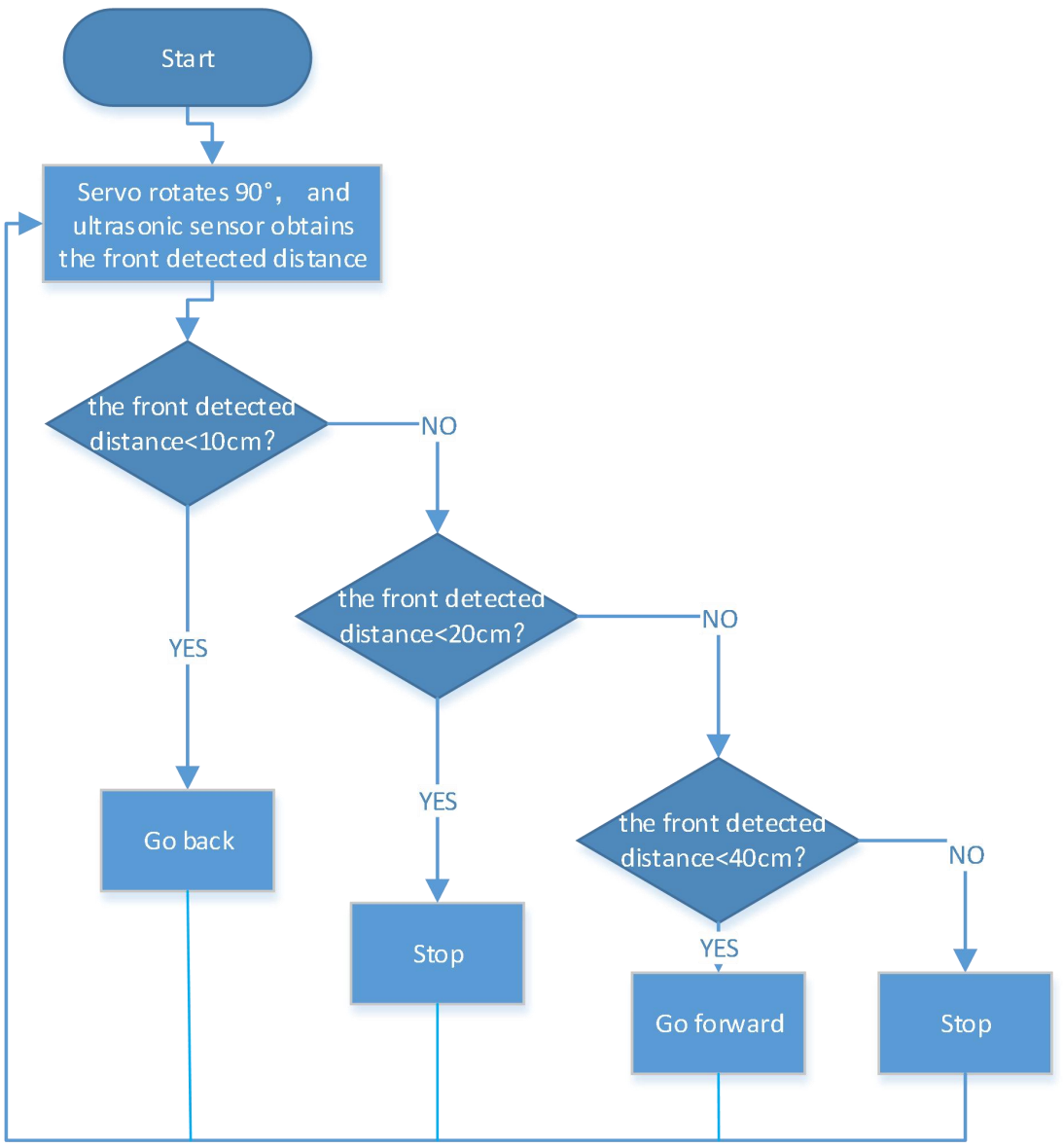
In previous lesson, we’ ve learned the basic principle of line tracking sensor. Next, we will combine the ultrasonic sensor with the car to make an ultrasonic following car.

The ultrasonic sensor detects the obstacle distance and control the motion status of car.

2. Preparation

- Insert micro:bit board into the slot of keyestudio 4WD Mecanum Robot CarV2.0
- Place batteries into battery holder
- Dial power switch to ON end
- Connect micro:bit to the computer via an USB cable
- Open the offline version of Mu.

3. Flow Chart



4. Test Code

Enter Mu software and open the file "Project 18.3: Ultrasonic Follow Smart Car.py" to import code:

(How to load the project code?)

File Type	Route	File Name
Python file	../Python code/8.18: Ultrasonic/8.18.3: Ultrasonic Follow Smart Car	microbit- Ultrasonic Follow Smart Car..py

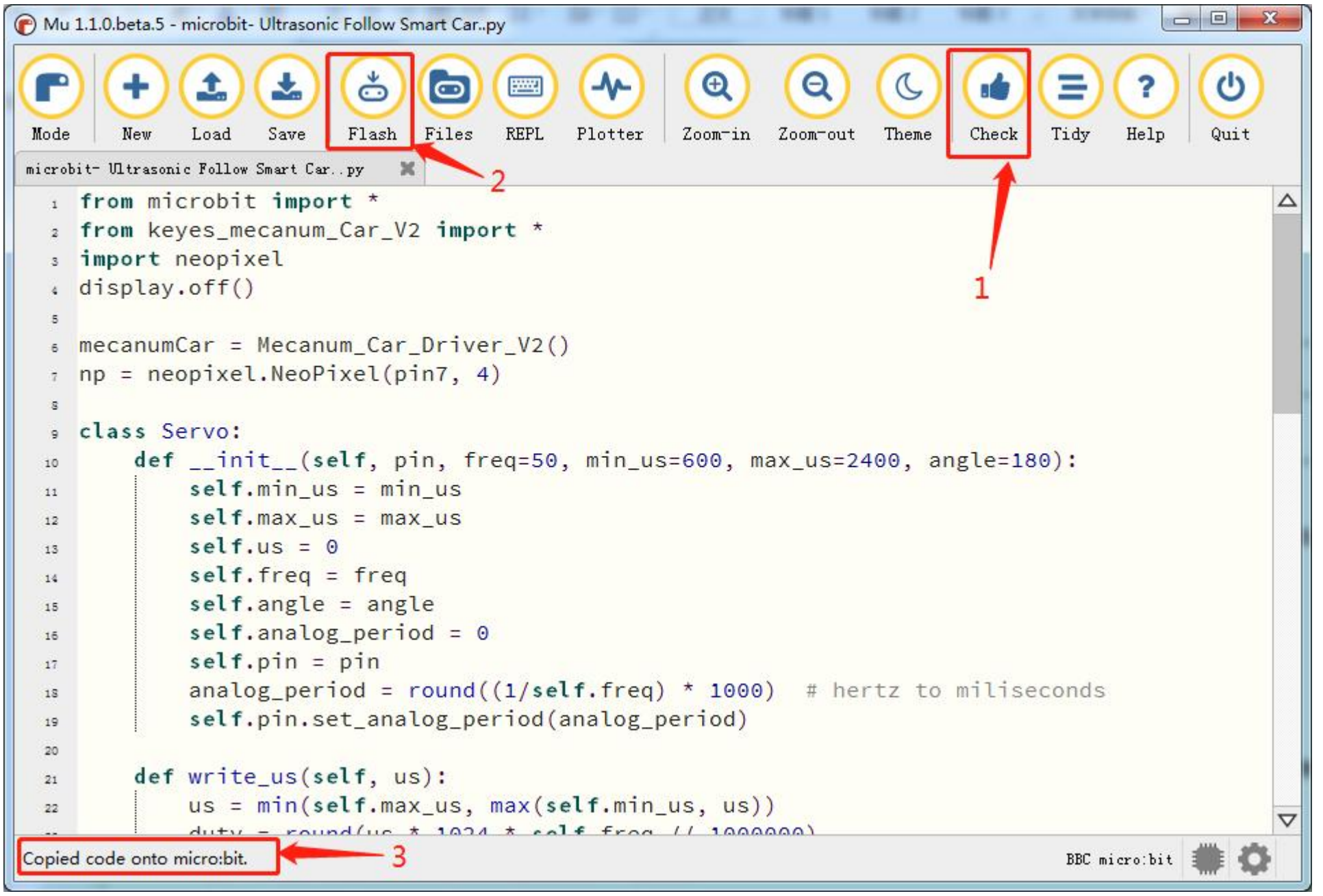
You can also input code in the edit window yourself.

(Note: All English words and symbols must be written in English.)

Click "Files" to import "keyes_mecanum_Car_V2.py "library file to micro:bit (How to import files?).

Click "Check" to examine errors in the code. The program proves wrong if underlines and cursors are shown.

If the code is correct, connect the micro:bit to your computer and click "Flash" to download the code to the micro:bit board.



5. Test Result

Download code to the micro:bit, dial POWER switch to ON end on shield, the smart car could follow the obstacle to move and 4 WS2812 RGB lights will display different colors.

Note: the obstacle can only move in front of smart car.

6. Code Explanation

from microbit **import ***	Import the library of micro:bit
from keyes_mecanum_Car_V2 **import ***	Import the library of keyes_mecanum_Car
mecanumCar = Mecanum_Car_Driver_V2()	Instantiate Mecanum_Car_Driver()to mecanumCar
import neopixel	Import the library of neopixel
display.off()	Close the dot matrix
np = neopixel.NeoPixel(pin7, 4)	Set Neopixel to pin P7, and initialize 4 LEDs
Servo(pin14).write_angle(90)	The servo connected to P14 pin turns to 90 degrees
while True:	This is a permanent loop that makes micro:bit execute the code of it.

distance_val = 0	Set the initial value of the variable distance_val to 0
distance_val = mecanumCar.get_distance()	Assign mecanumCar.get_distance() to the variable distance_val
if distance_val >= 20 **and** distance_val <= 40: mecanumCar.Motor_Upper_L(1, 80) mecanumCar.Motor_Lower_L(1, 80) mecanumCar.Motor_Upper_R(1, 80) mecanumCar.Motor_Lower_R(1, 80) **for** pixel_id1 **in** range(0, len(np)): np[pixel_id1] = (255, 0, 0) np.show()	If distance_val ≥20 and distance_val ≤ 40 are true, The car moves forward. RGB pixels in the range of (0, len(np)) are pixel_id1 Set pixel_id1 to light up red Display pixels on Neopixel strip
if distance_val <= 10: mecanumCar.Motor_Upper_L(0, 80) mecanumCar.Motor_Lower_L(0, 80) mecanumCar.Motor_Upper_R(0, 80) mecanumCar.Motor_Lower_R(0, 80) **for** pixel_id1 **in** range(0, len(np)): np[pixel_id1] = (255, 255, 0) np.show()	If distance_val ≤10 holds The car moves back. RGB pixels in the range of (0, len(np)) are pixel_id1 Set pixel_id1 to bright yellow light Display pixels on Neopixel strip
if distance_val > 10**and** distance_val < 20 **or** distance_val > 40: mecanumCar.Motor_Upper_L(0, 0) mecanumCar.Motor_Lower_L(0, 0) mecanumCar.Motor_Upper_R(0, 0) mecanumCar.Motor_Lower_R(0, 0) **for** pixel_id1 **in** range(0, len(np)): np[pixel_id1] = (255, 255, 255) np.show()	If distance_val>10 and distance_val<20 or distance_val>40 is true The car stops. RGB pixels in the range of (0, len(np)) are pixel_id1 Set pixel_id1 to bright white light Display pixels on Neopixel strip

9. Resources

1. Download PDF files: https://fs.keyestudio.com/KS4034-4035

2. BBC microbit MicroPython：

https://microbit-micropython.readthedocs.io/en/latest/tutorials/introduction.html

3.MicroPython：

https://docs.openmv.io/reference/index.html

4.ustruct library：

https://docs.openmv.io/library/ustruct.html

5.math library：

https://docs.openmv.io/library/math.html

6.utime(sleep_us,tick_us) library：

https://docs.openmv.io/library/utime.html#

10. Common Problems

1. The car has no reaction

Please check whether the batteries are sufficient

Please check whether the wirings are correct

2. Computers can't recognize the USB ports

Please ensure whether the microbit driver is installed

Please check whether the USB wire is in good condition.

3. Code fails to burn and dot matrix displays expressions

Please check whether the keyes_mecanum_Car_V2.py library file is imported