Welcome

Thank you for choosing Freenove products!

Get Support & Offer Input

You may find somethings missing or broken, or some difficulty to learn the kit.
Freenove provides free and quick support, including but not limited to:

- Quality problems of products
- Problems in using products
- Questions for learning and technology
- Opinions and suggestions
- Ideas and thoughts

If you have any concerns, please send email to us:

support@freenove.com

And suggestions and feedbacks are welcomed. Many customers offered great feedbacks. According to that, we are keeping updating the kit and the tutorial to make it better. Thank you.

Safety

Pay attention to safety when using and storing this product:

- Do not expose children under 6 years of age to this product. Put it out of their reach.
- Children lack safety ability should use this product under the guardianship of adults.
- This product contains small and sharp parts. Do not swallow, prick and scratch to avoid injury.
- This product contains conductive parts. Do not hold them to touch power supply and other circuits.
- Some parts will rotate or move when it works. Do not touch them to avoid being bruised or scratched.
- The wrong operation may cause overheat. Do not touch and disconnect the power supply immediately.
- Operate in accordance with the requirements of the tutorial. Otherwise, the parts may be damaged.
- Store the product in a dry place and avoid direct sunlight.
- Turn off the power of the circuit before leaving.

About

Freenove provides open source electronic products and services.

Freenove is committed to helping customers learn programming and electronic knowledge, quickly realize their creative ideas and product prototypes and launching innovative products. Our services include:

- Kits of robots, smart cars and drones
- Kits for learning Arduino, Raspberry Pi and micro:bit
- Electronic components and modules, tools
- **Product customization service**

You can learn more about us or get our latest information through our website:

http://www.freenove.com

Copyright

FREENOVE®

FREE YOUR INNOVATION

Contents

Preface

Do you want to learn programming?

Nowadays, Program is developed into the younger age group, and everyone programming is a trend. From Arduino and Raspberry Pi to micro:bit, simple graphical programming makes programming for kids possible. Maybe you haven't heard of them, it doesn't matter. With this product and the tutorial, you can easily complete a programming project and experience the fun as a Maker.

Micro:bit is a powerful and simple development board. Even if you've never programmed before, its simple graphical programming interface allows you to master it easily. It doesn't require any professional programming software; just simply a browser is enough to program it. So, no matter your computer system is Windows, Linux or Mac, you can program it. And you can also program it with Python.

It attracts a lot of fans in the world who are keen to exploration, innovation and DIY and have contributed a great number of high-quality open-source code, circuit and rich knowledge base. So we can realize our own creativity more efficiently by using these free resource. Of course, you can also contribute your own strength to the resource.

With Micro:bit, we can make a lot of projects and by adding kits to breadboard, we can carry out more interesting projects like ultrasonic ranging, gravity control, playing music, etc.

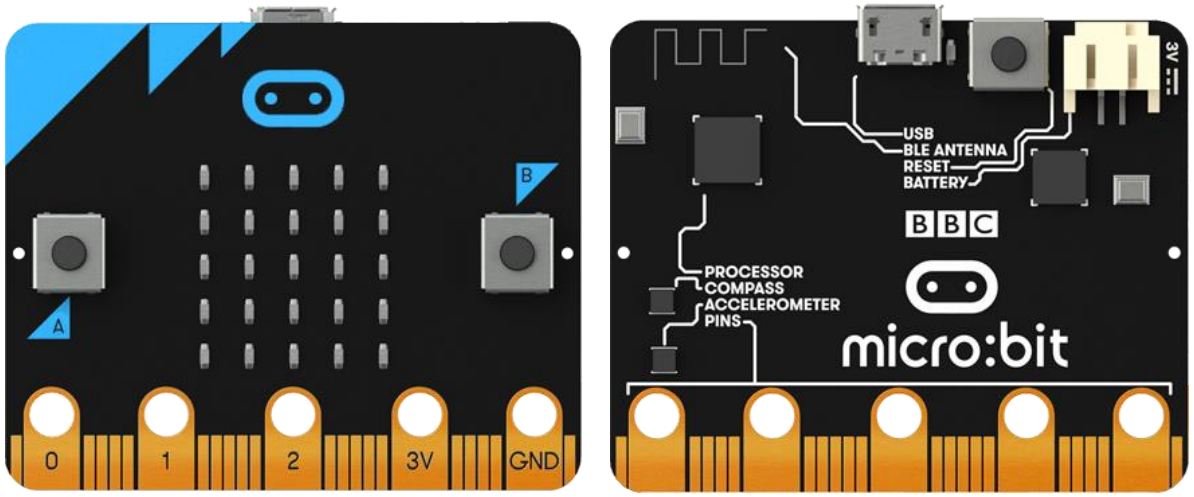
In each learning chapter of this tutorial, we provide program source code with detailed program explanations and burnable binaries, so that you can understand the meaning of each section of program.

Additionally, if you have any difficulties or questions about this tutorial and the kit, you can always ask us for quick and free technical support.

Micro:bit

This chapter is the Start Point in the journey to build and explore Micro:bit and Micro:Rover electronic projects.

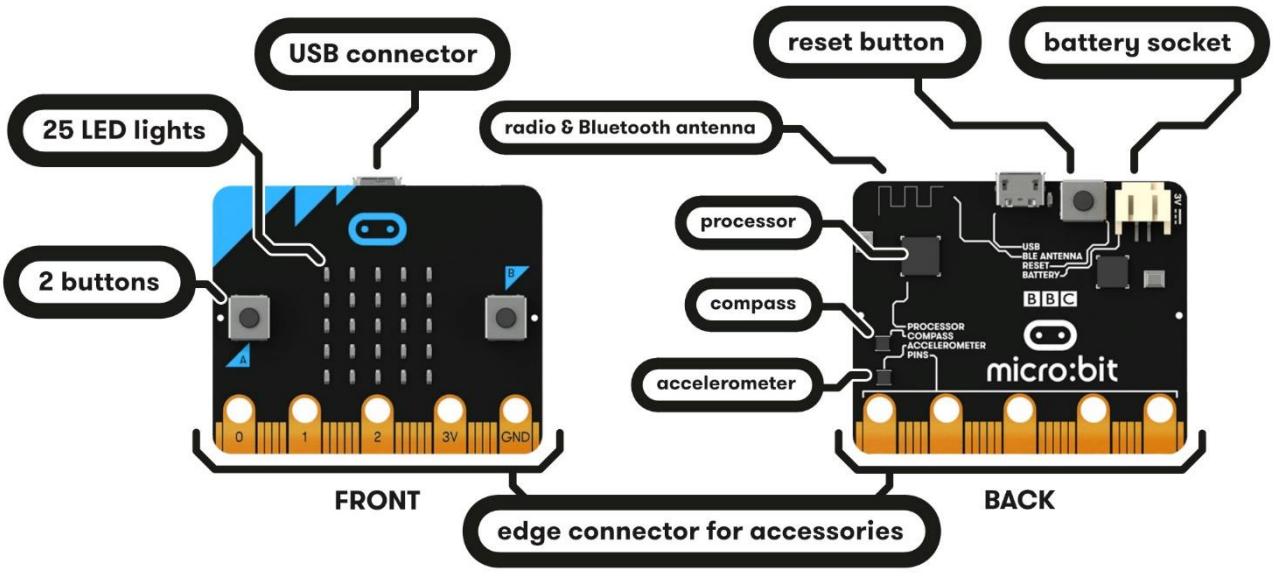
Meet micro:bit



The BBC micro:bit is a pocket-size, programmable micro-computer that can be used for all sorts of cool creations, from robots to musical instruments – the possibilities are infinite.

For more contents, please refer to:
https://microbit.org/guide/

Features



Your micro:bit has the following physical features:

- 25 individual programmable LEDs
- 2 programmable buttons
- Physical connection pins
- Light and temperature sensors
- Motion sensors (accelerometer and compass)
- Wireless Communication, via Radio and Bluetooth
- USB interface

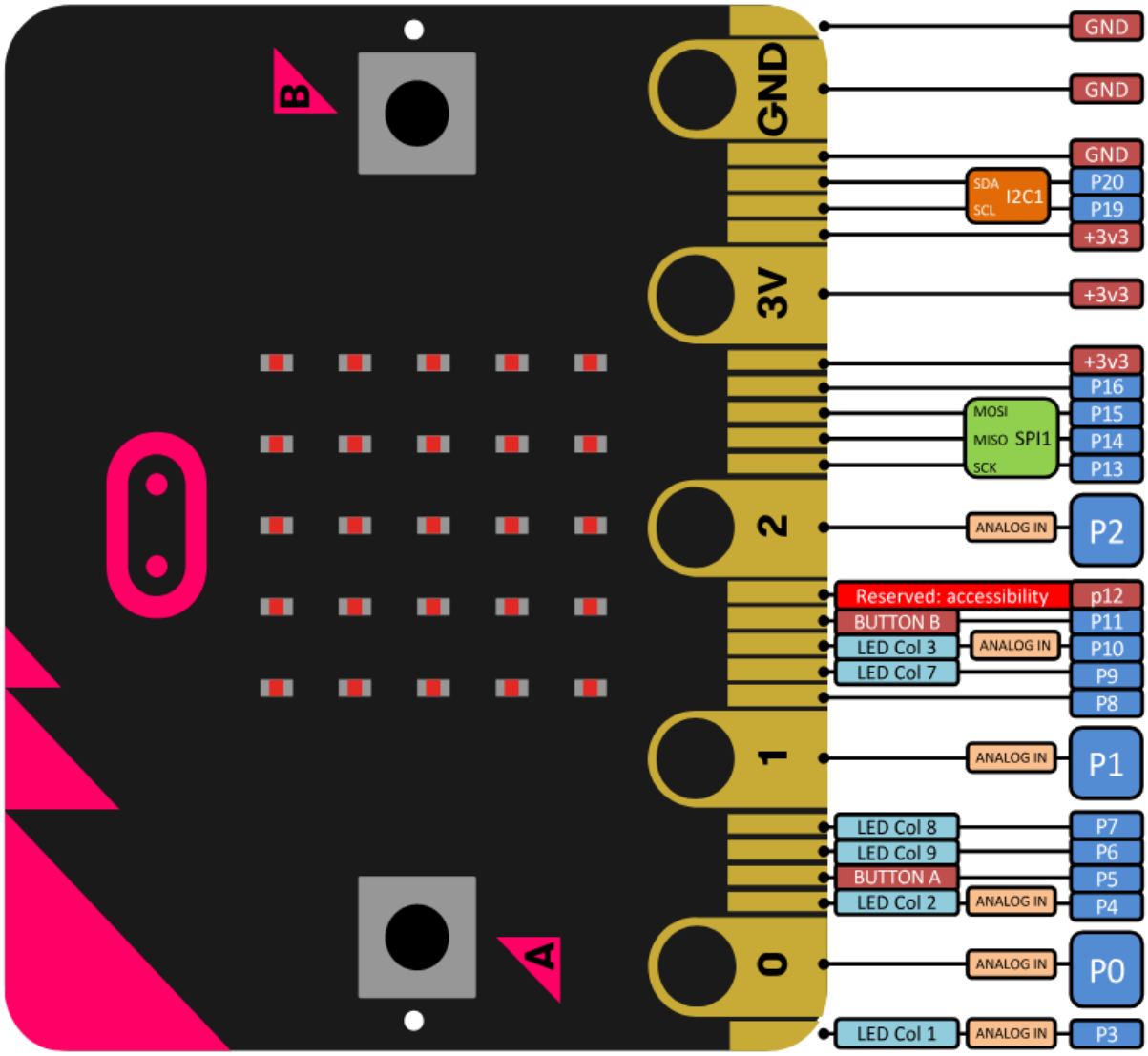
For more details, please refer to:
https://microbit.org/guide/features/

Hardware

It is not required for beginners to master this section, but a brief understanding is necessary. However, if you want to be a developer, hardware information will be very helpful. Detailed hardware information about micro:bit can be found here: https://tech.microbit.org/hardware/.
First, get to know the micro:bit GPIO.

GPIO

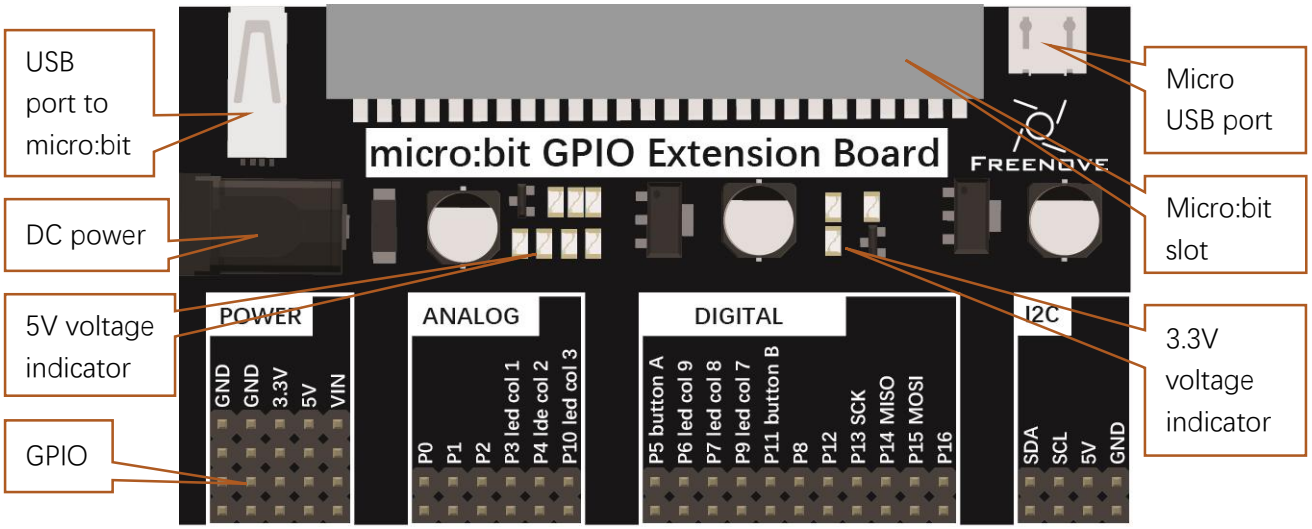
GPIO, namely General Purpose Input/output Pins, is an important part of micro:bit for connecting external devices. All sensors and devices on Rover communicate with each other through micro:bit GPIO. The following is the GPIO serial number and function diagram of micro:bit:



Micro:bit GPIO Extension Board

Hardware and Feature

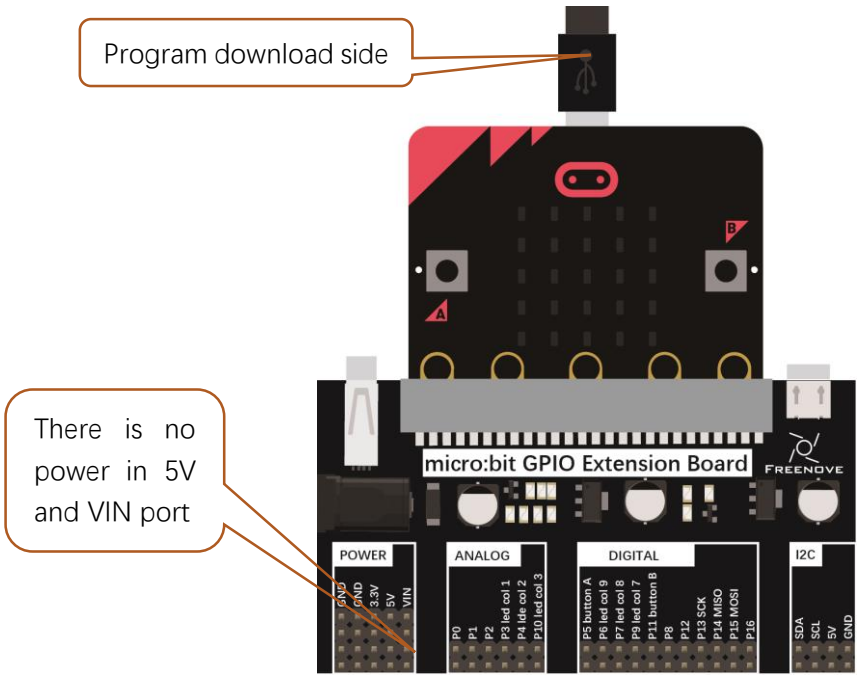
Micro:bit GPIO Extension Board is shown as below:



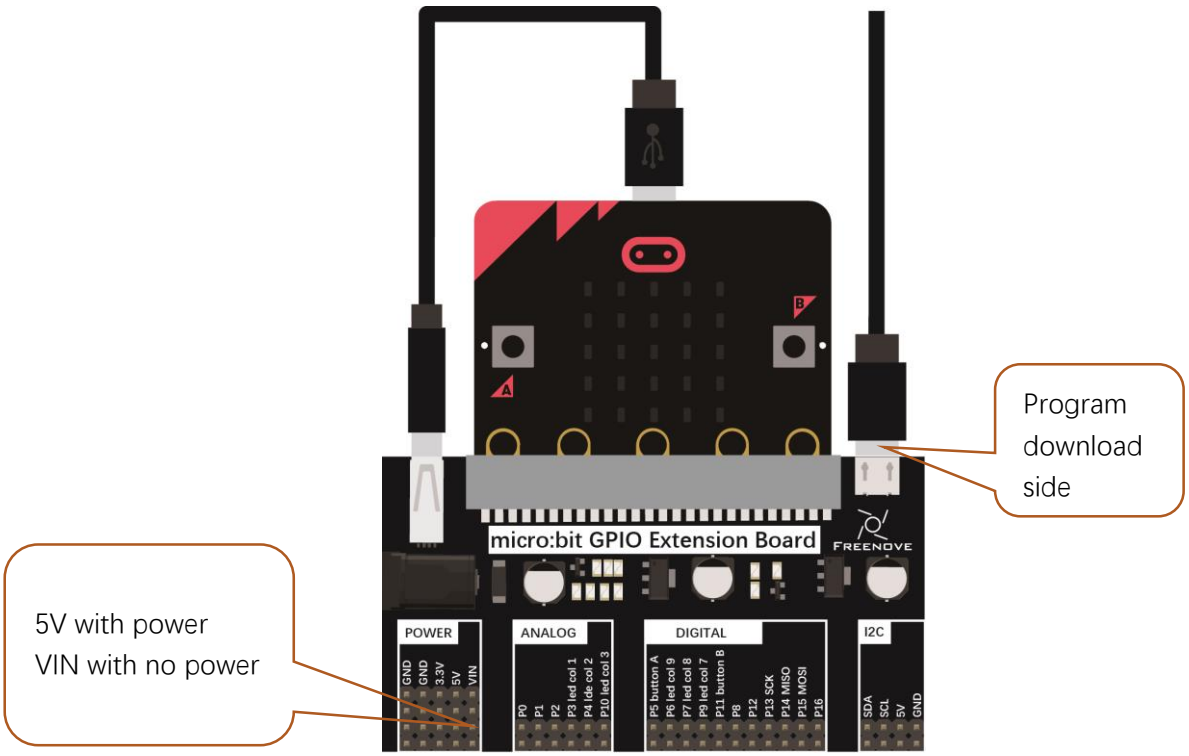
Micro:bit GPIO Extension Board is connected to micro:bit board via a slot with its GPIO connected to micro:bit's GPIO. In addition, there are also 5V and VIN(9V) IO port on the extension board to meet requirement of more devices.

How to use?

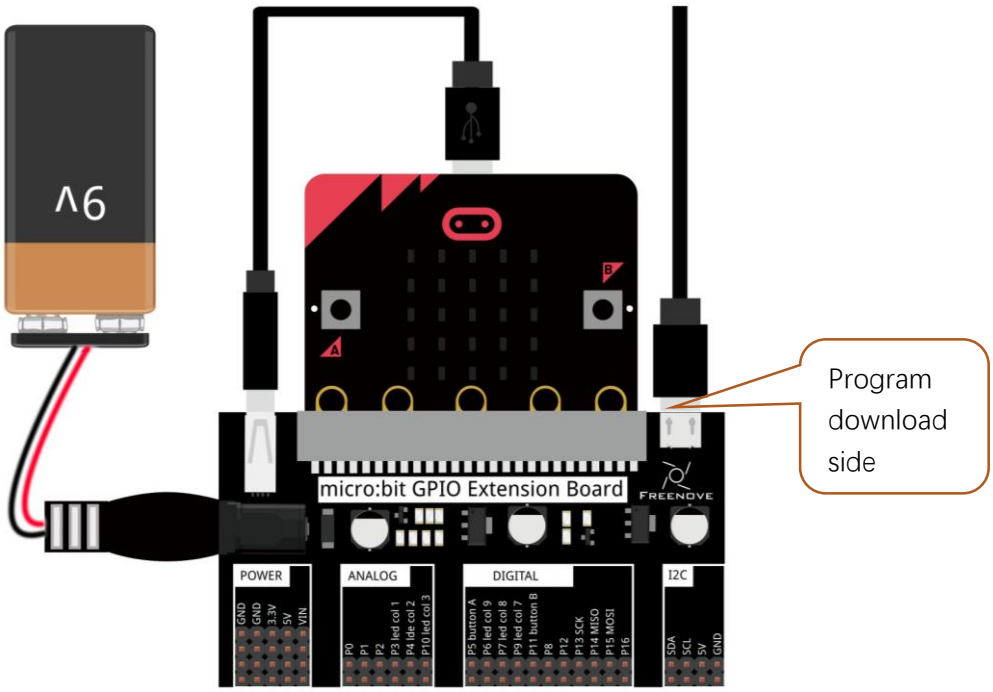
If external device doesn't require voltage of 5V and 9V, you can use the following wiring.



If external device uses 5v voltage, but power needed is not large, you can use the following wiring.



If external device uses 5v voltage, but power needed is large, you can use the following wiring.



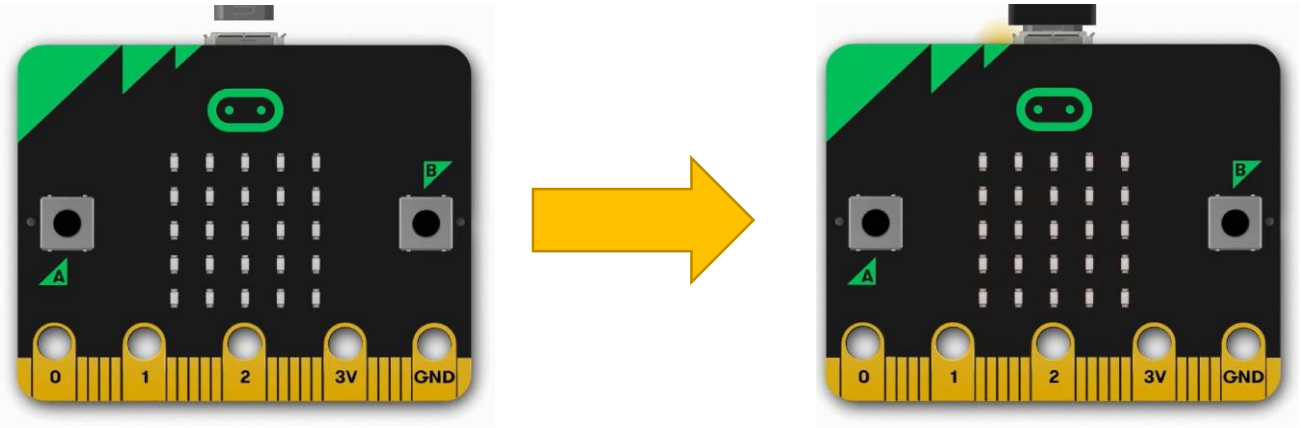
Code & Programming

Quick Start

This section describes how to write programs for micro:bit and how to download them to micro:bit. There are very detailed tutorials on the official website. You can refer to: Https://microbit.org/guide/quick/.

Step 1: Connecting Micro:bit

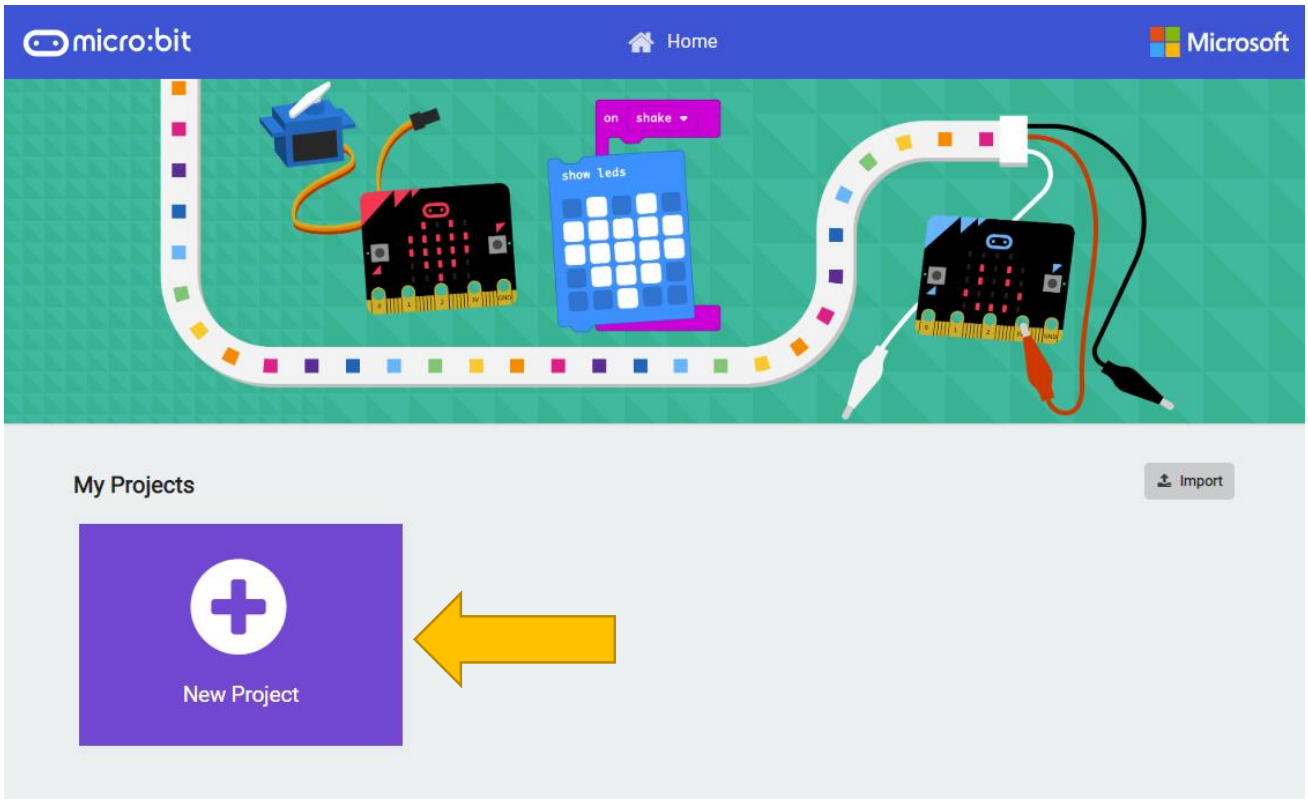
Connect the micro:bit to your computer via a micro USB cable. Macs, PCs, Chromebooks and Linux systems (including Raspberry Pi) are all supported.



Step 2: Write Program

Visit https://makecode.microbit.org/. Then click "New Project" and start programming.

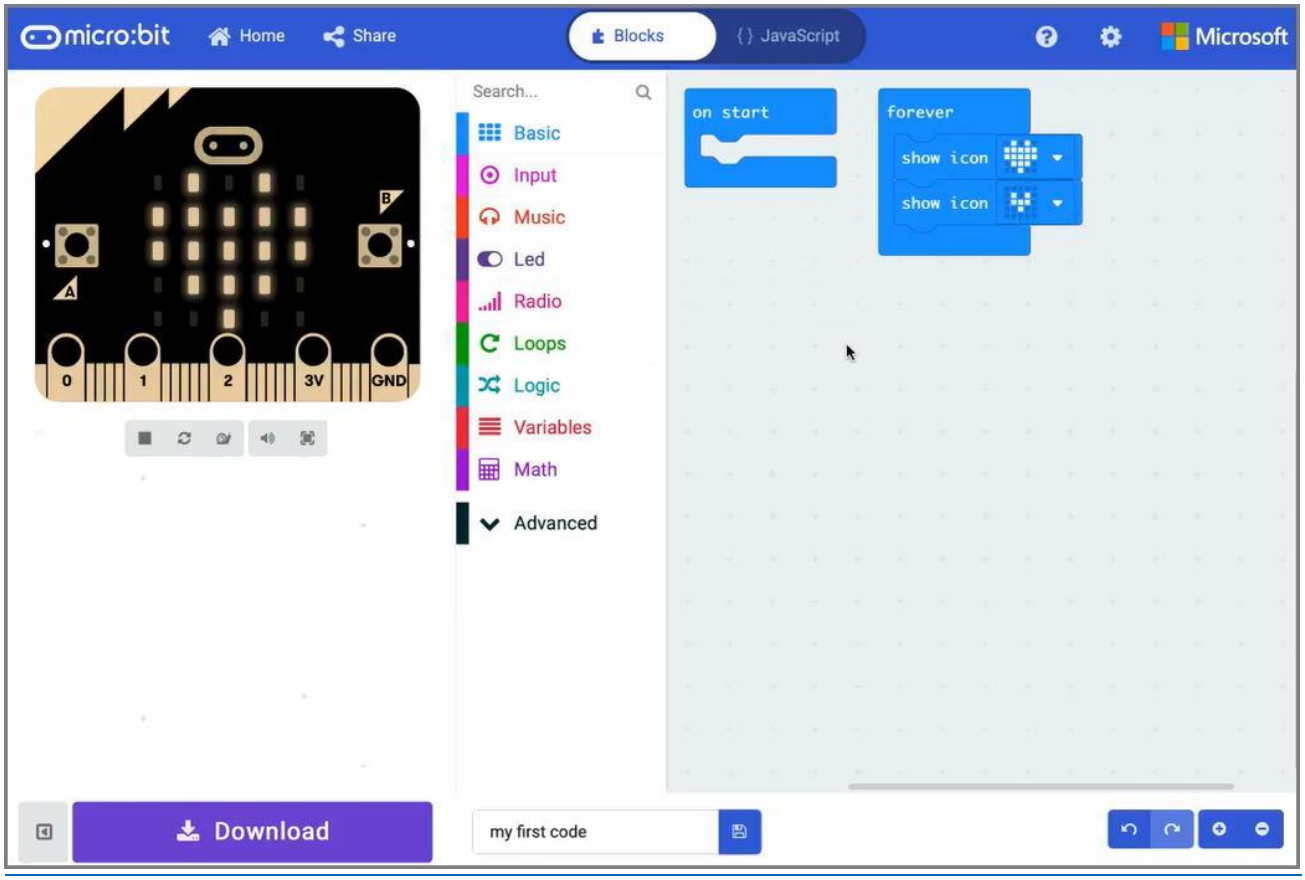
If your computer has Windows 10 operating system, you can also use Windows 10 App for programming, which is exactly the same as programming on browsers. Get windows 10 App(Click).



Write your first micro:bit code. For example, drag and drop some blocks and try your program on the Simulator in the MakeCode Editor, like in the image below that shows how to program a Flashing Heart.

Here is a demo video: https://microbit.org/images/quickstart/makecode-heart.mp4

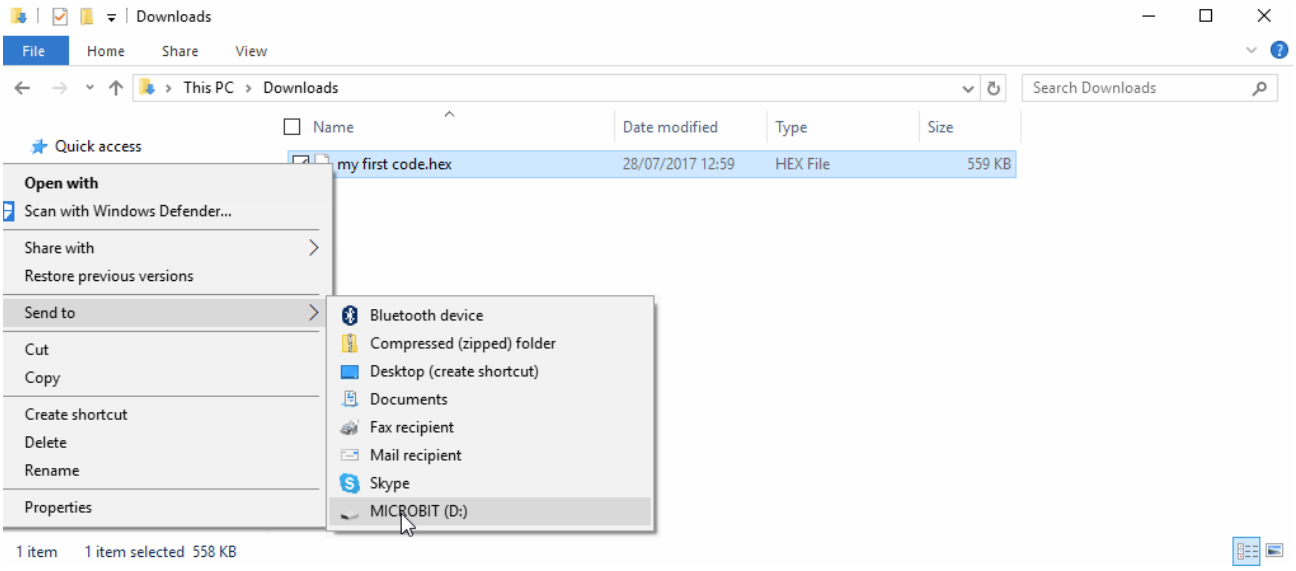
MakeCode will be further introduced in next section.



Step 3: Flashing Code to your Micro:bit

The process of transferring the .HEX file to the BBC micro:bit is called flashing.
If you write program using Windows 10 App, you just need to click the “Download” button, then the program will be downloaded directly to micro:bit without any other actions.

If you write program using browser, please follow steps below:
Click the Download button in the editor. This will download a 'hex' file, which is a compact format of your program that your micro:bit can read. Once the hex file has been downloaded, copy it to your micro:bit just like copying a file to a USB drive. On Windows you can right click and choose "Send to→MICROBIT."



Step 4: Run the Program

The micro:bit will pause and the yellow LED on the back of the micro:bit will blink while your code is flashed. Once that's finished the code will run automatically! The micro:bit can only run one program at a time - every time you drag-and-drop a hex file onto the device over USB it will erase the current program and replace it with the new one.

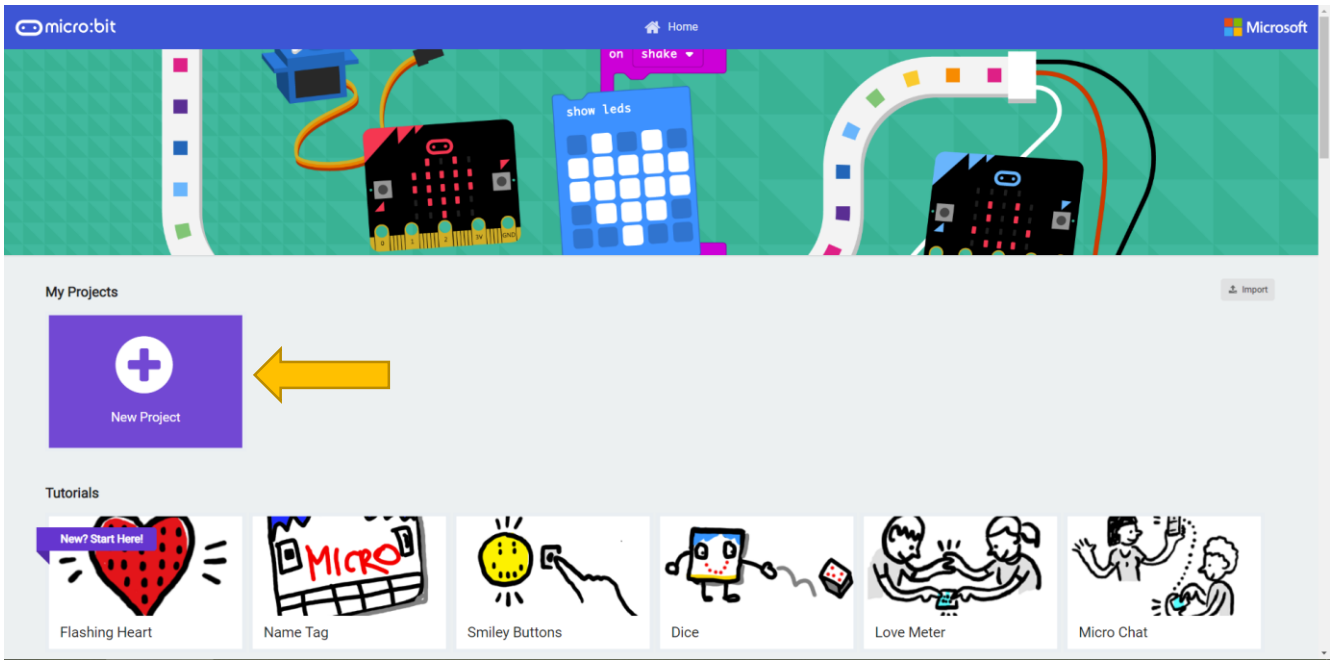
Warning

The MICROBIT drive will automatically eject and reconnect each time you program it, but your hex file will be gone. The micro:bit can only receive hex files and won't store anything else!

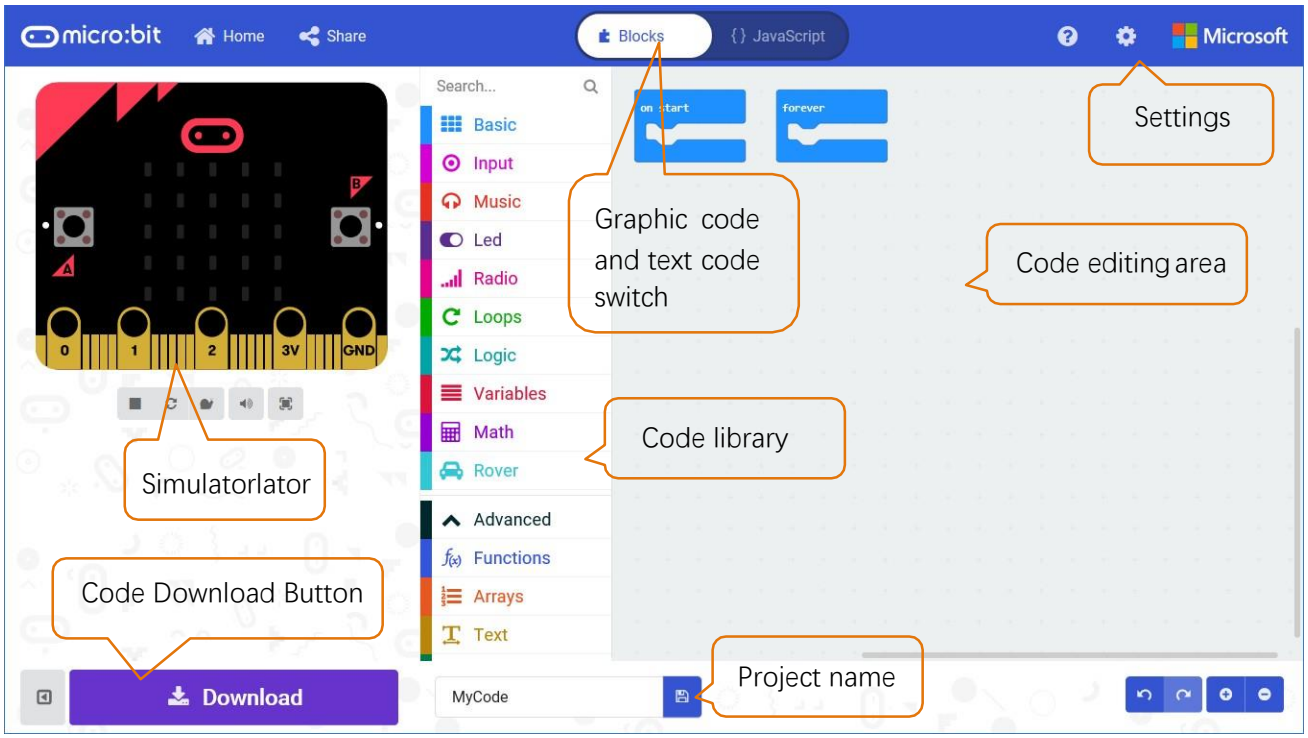
MakeCode

Open web version of MakeCode or **windows 10 app version of MakeCode**, which can be downloaded on **Microsoft store.**

https://makecode.microbit.org/



Click "New Project", MakeCode editor is as below:



In the code area, there are two fixed blocks "on start" and "forever".
The code in the "on start" block will be executed only once after power-on or reset. And the code in "forever" block will be executed circularly.

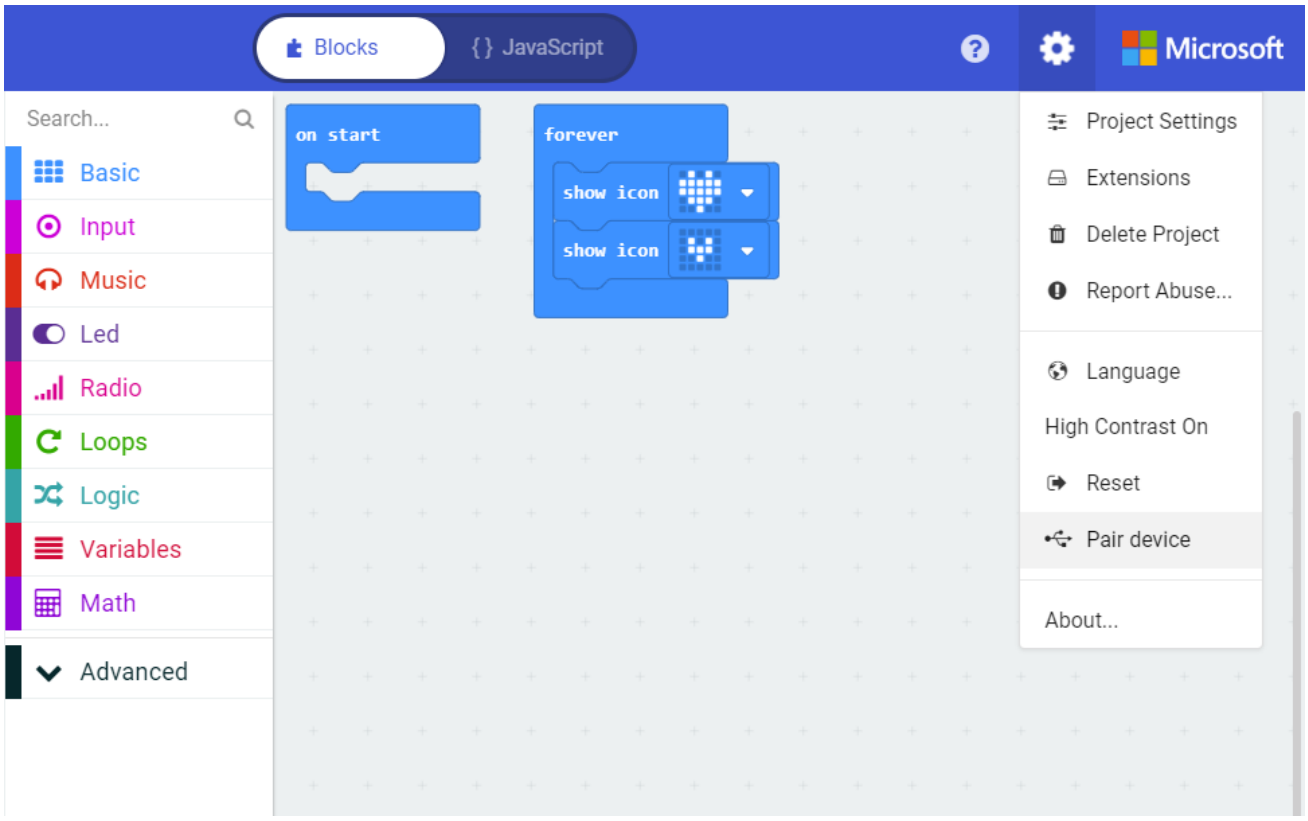
Quick Download

As mentioned earlier, if you use **Windows 10 App of MakeCode (recommended)**, you can **quickly** download the code to micro:bit by **clicking the download button**. Using **browser version of MakeCode** may require **more steps**.

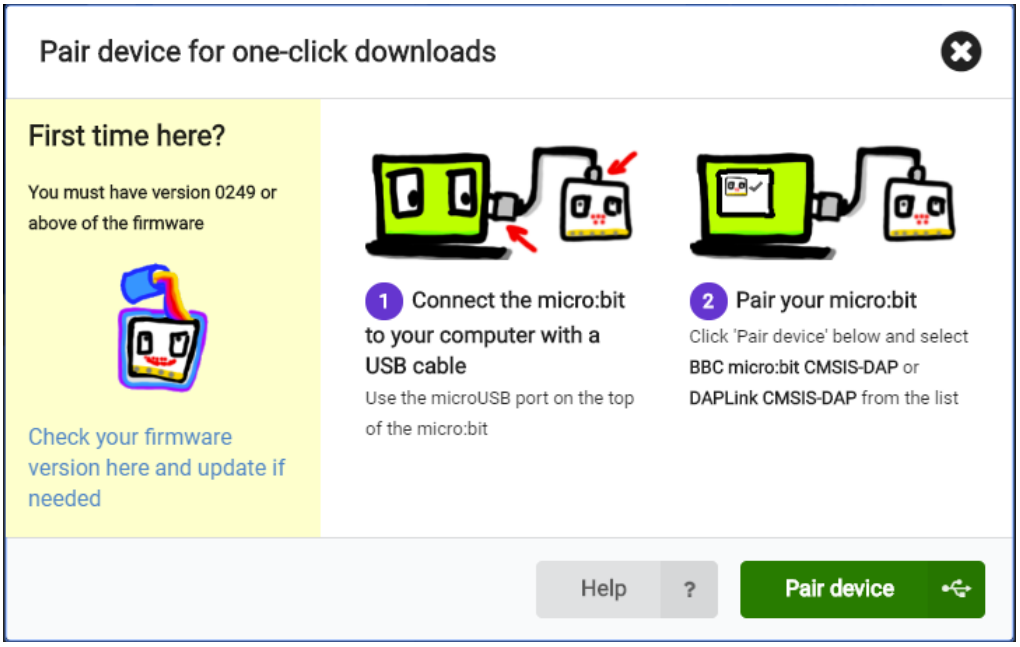
If you use browser version of MakeCode on **Google Chrome 65+ for platform Android, Chrome OS, Linux, macOS and Windows 10**, you can also download file quickly.
Here we use webUSB feature of Chrome, which allows web pages to access your USB hardware devices. We will complete the connection and pairing of the micro:bit device with the webpage in the following steps.

Pair device

Connect your computer and Micro:bit with a USB cable.
Click the gear menu in the top right corner and then click on "Pair device".



Then continue to click “Pair device” button.



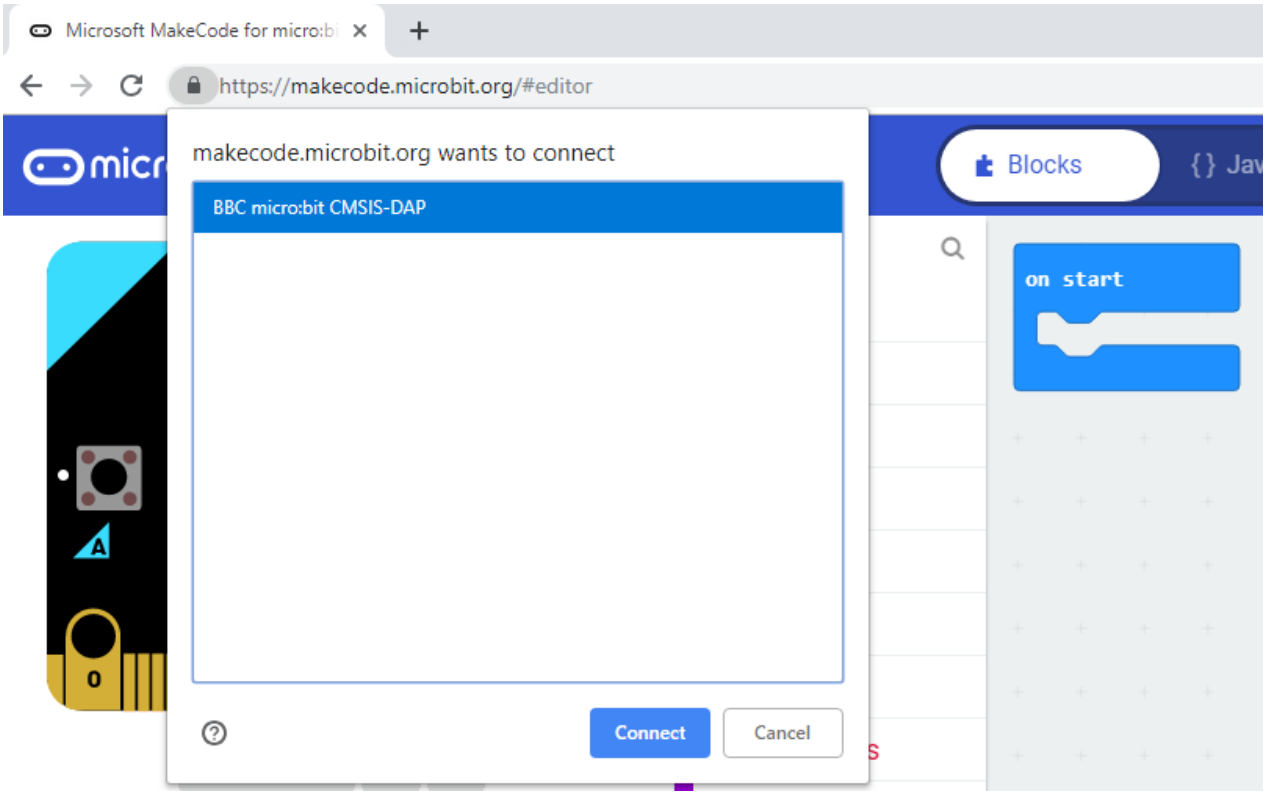
Select device on the pop-up window and click "Connect" button. If there are no devices shown on the pop-up window, please refer to following content:

https://makecode.microbit.org/device/usb/webusb/troubleshoot

We have save the page as a file "**Troubleshooting downloads with WebUSB - Microsoft MakeCode.pdf**". You can read it directly in the folder of this tutorial.

And the file "**Firmware microbit.pdf**" introduces how to update firmware of micro:bit. Its content come from:

https://microbit.org/guide/firmware/



After the connection succeeds, click the Download button and the program will be downloaded directly to Micro: bit.

Import Code

We provide hex file (project files) for each project, which contains all the contents of the project and can be imported directly. You can also complete the code of project manually. If you choose to complete the code by dragging code block, you may need to add necessary extensions.

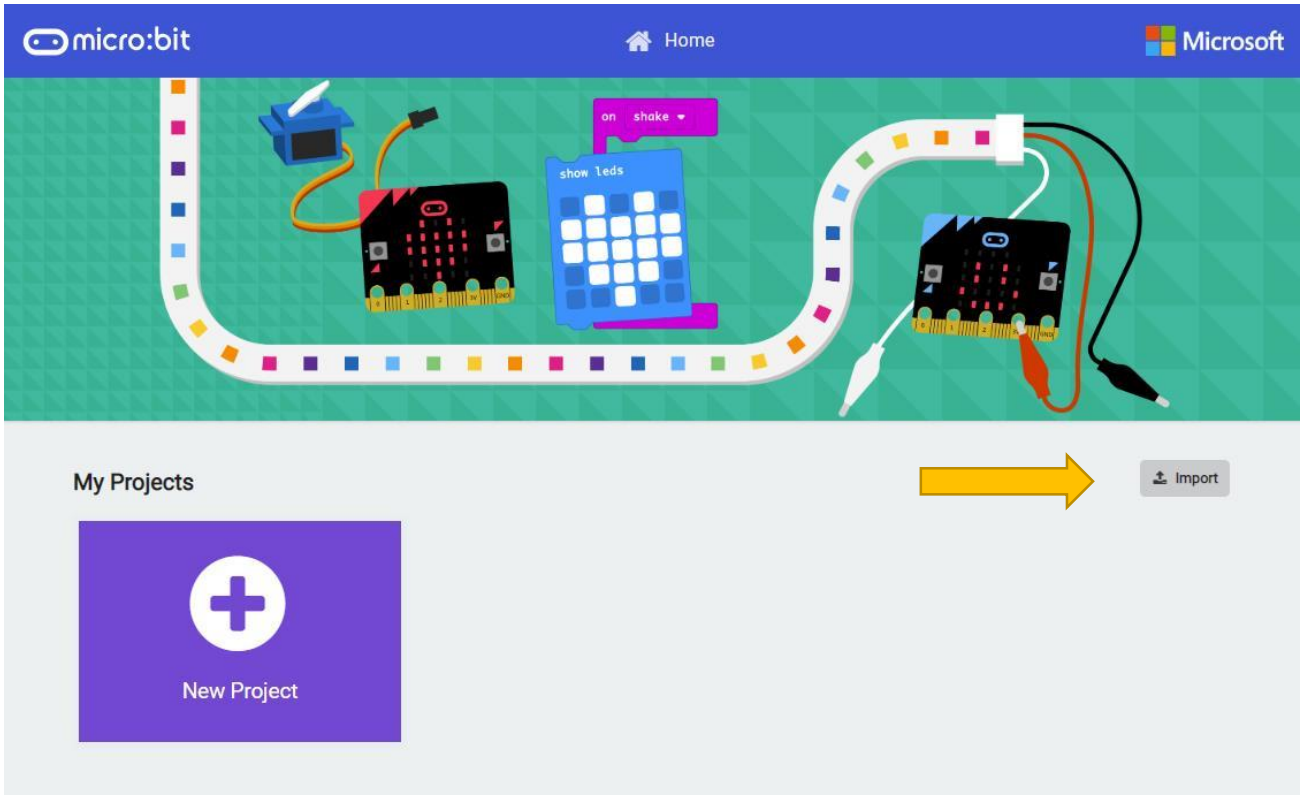
As for simple projects, it is recommended to complete the project by dragging code block.

As for complicated projects, it is recommended to complete the project by importing Hex code file.

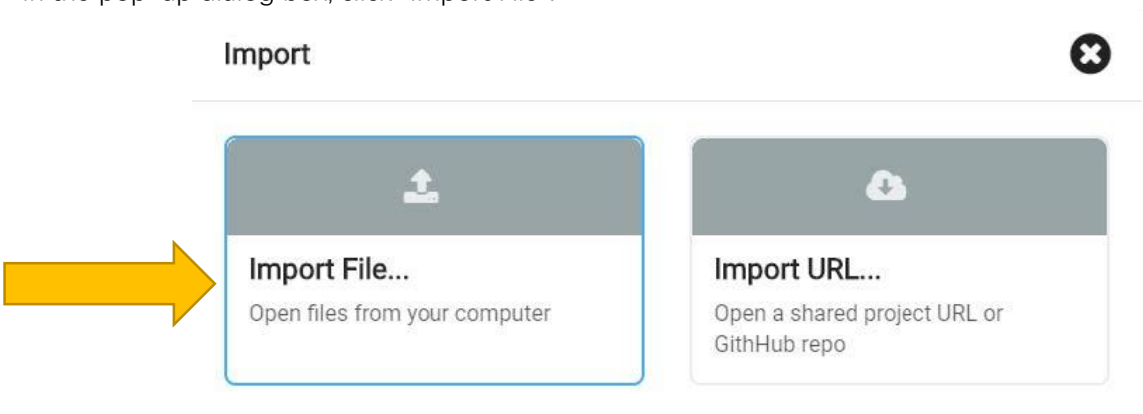
Next, we will take "Heartbeat" project as an example to introduce how to load code.

Open web version of makecode or windows 10 app version of MakeCode.

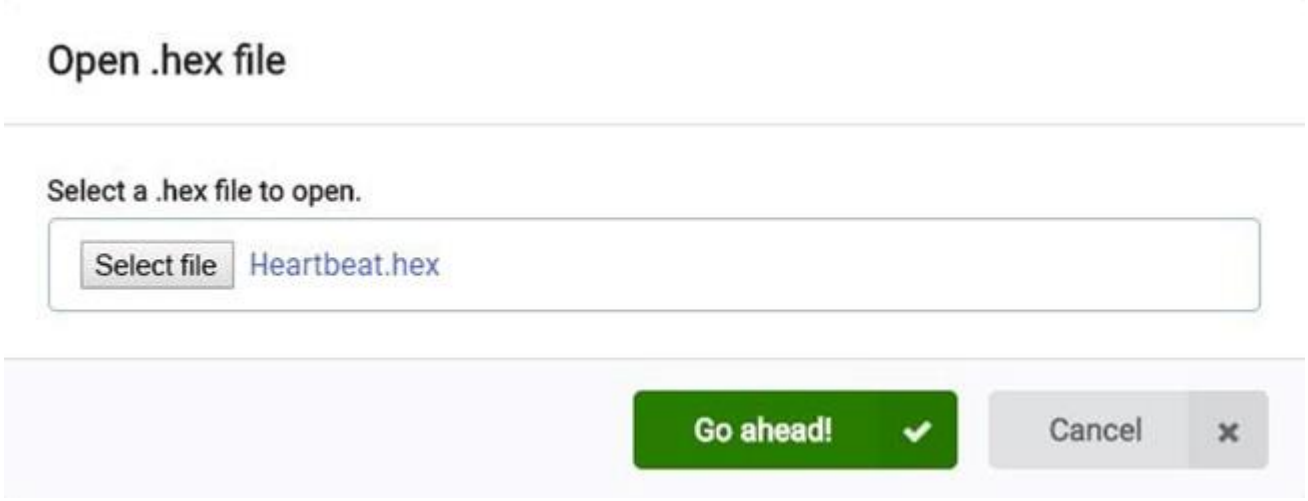
Click "Import" button on the right of HOME page.



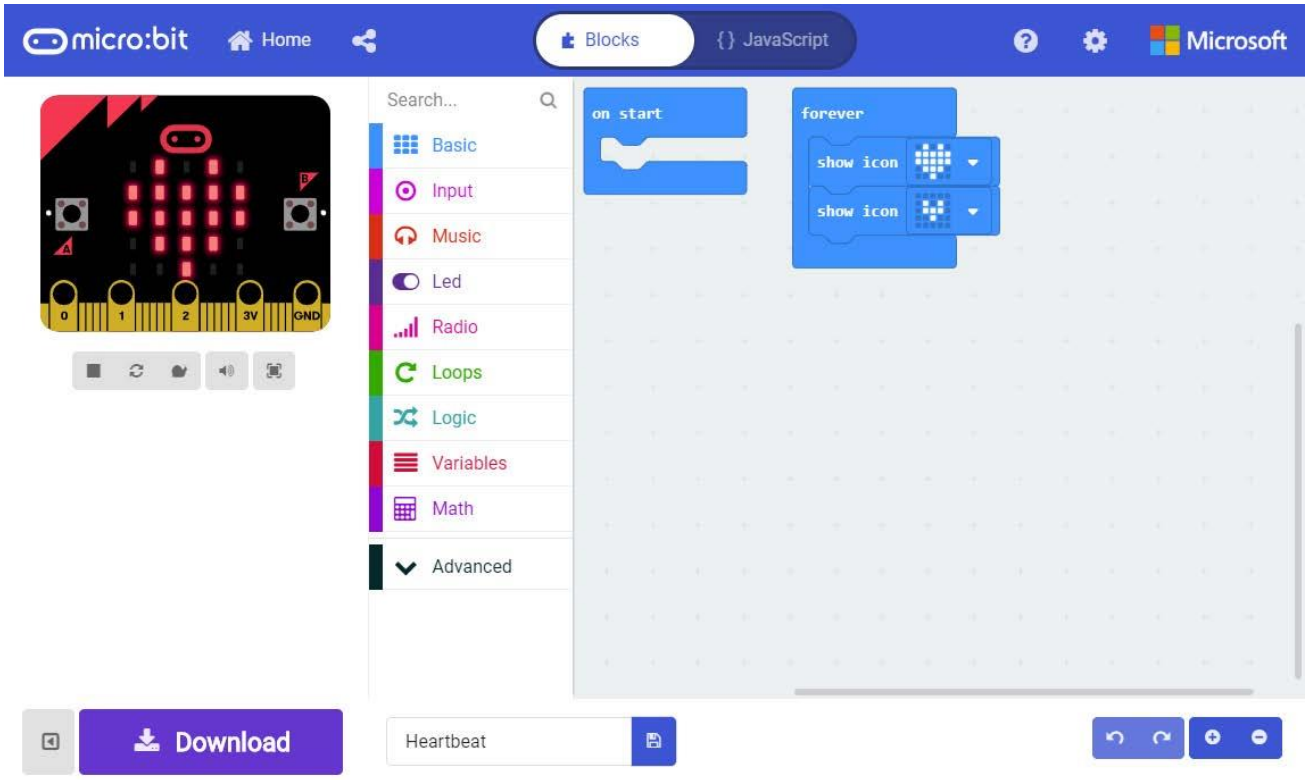
In the pop-up dialog box, click "Import File".



Select file".. Projects/BlockCode/01.1_Heartbeat/Heartbeat.hex". Then click "Go ahead!"



A few seconds later, the project is loaded successfully.

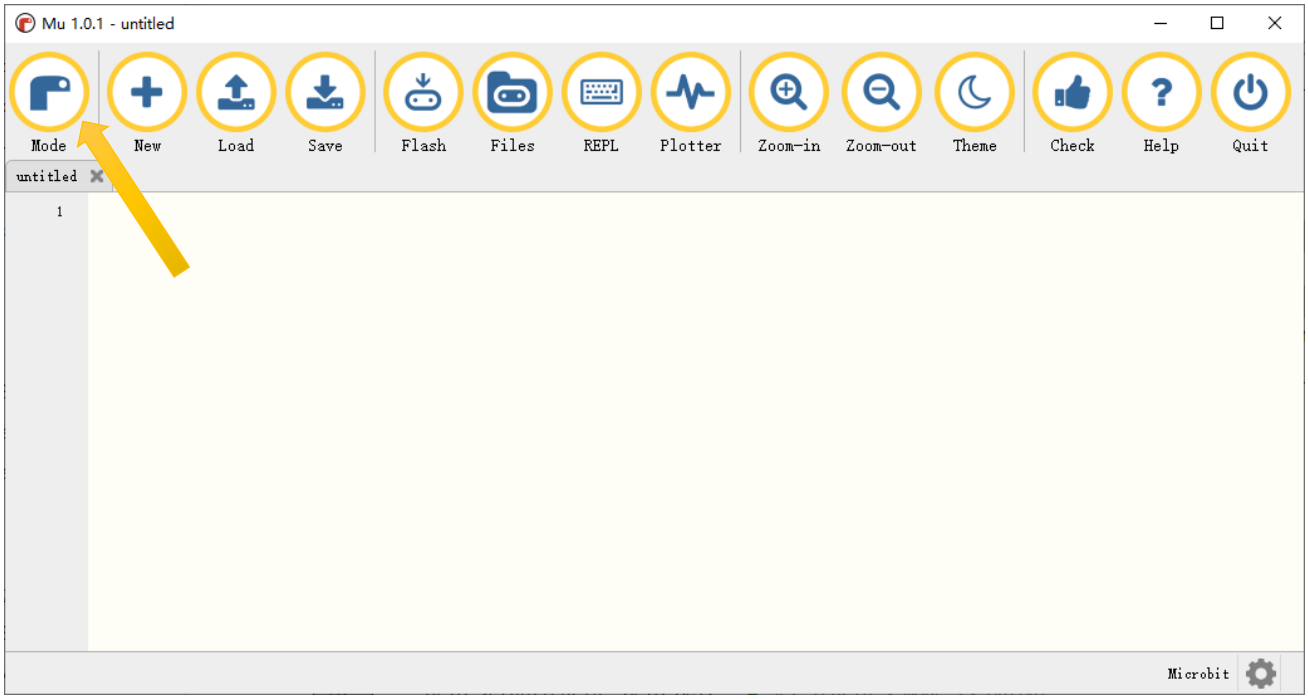


Python

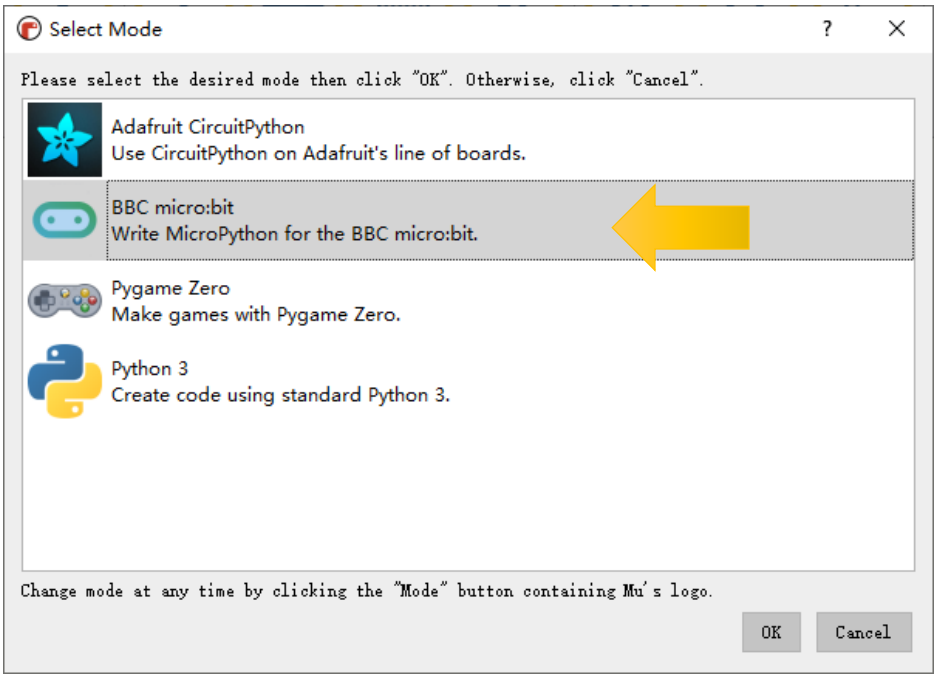
If you are not interested in python, you can skip this section.
Micro:bit can be programmed in Python. Since micro:bit is a microcontroller, the hardware difference makes it not support pure Python. Here we use MicroPython, which is specially designed for micro:bit.
MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments.
We designed block code and Python code with similar function for each project.
There are two kinds of python editors for micro:bit, web version and software.
It is highly recommended to use software Mu as a Python educator.
Mu. (https://codewith.mu/en/download)
Next, we will introduce Mu.

Mu

Mu is a Python code editor for beginner programmers based on extensive feedback given by teachers and learners.
Official website: https://codewith.mu/
You can download it here: https://codewith.mu/en/download
Download and install it.
And you will see the following interface when opening it.



Click the Mode button in the menu bar and select "BBC micro:bit" in the pop-up dialog box. Click "OK".

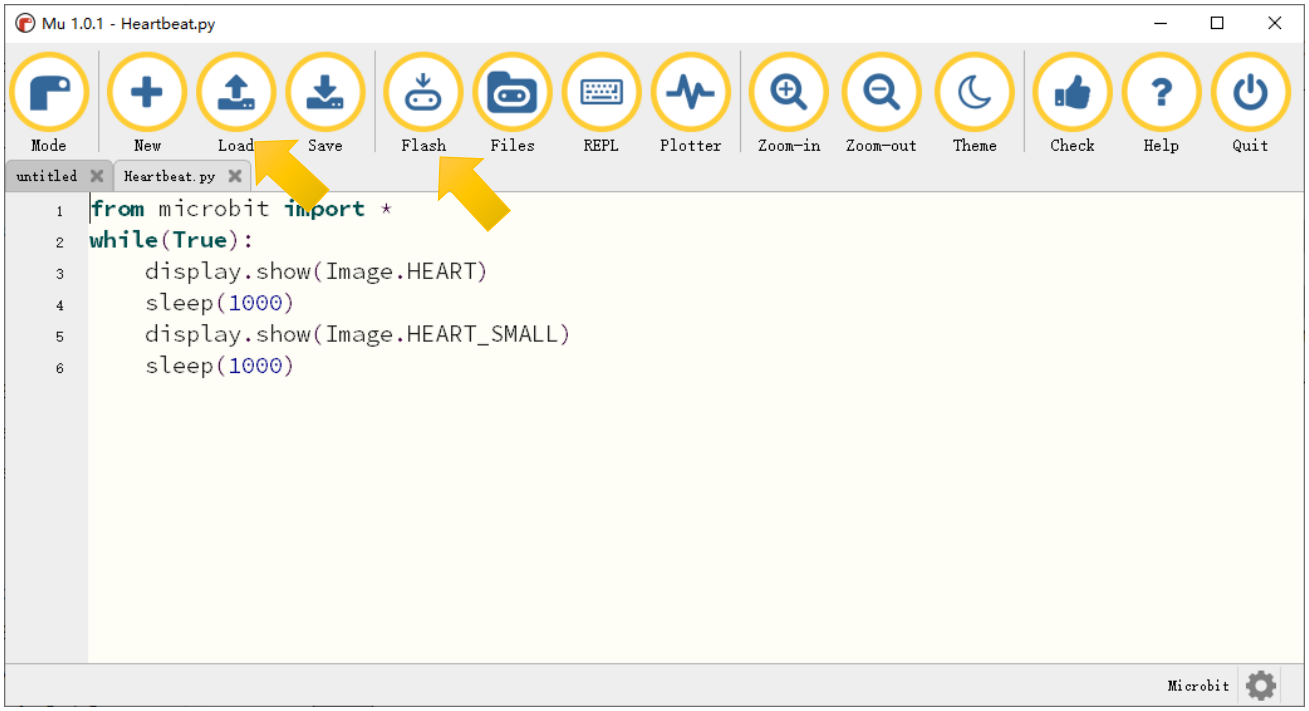


Import the .hex file. The path is as below:

File type	Path	File name
Python file	../project/1.1.Heartbeat	Heartbeat.py

Successful loading is shown below.

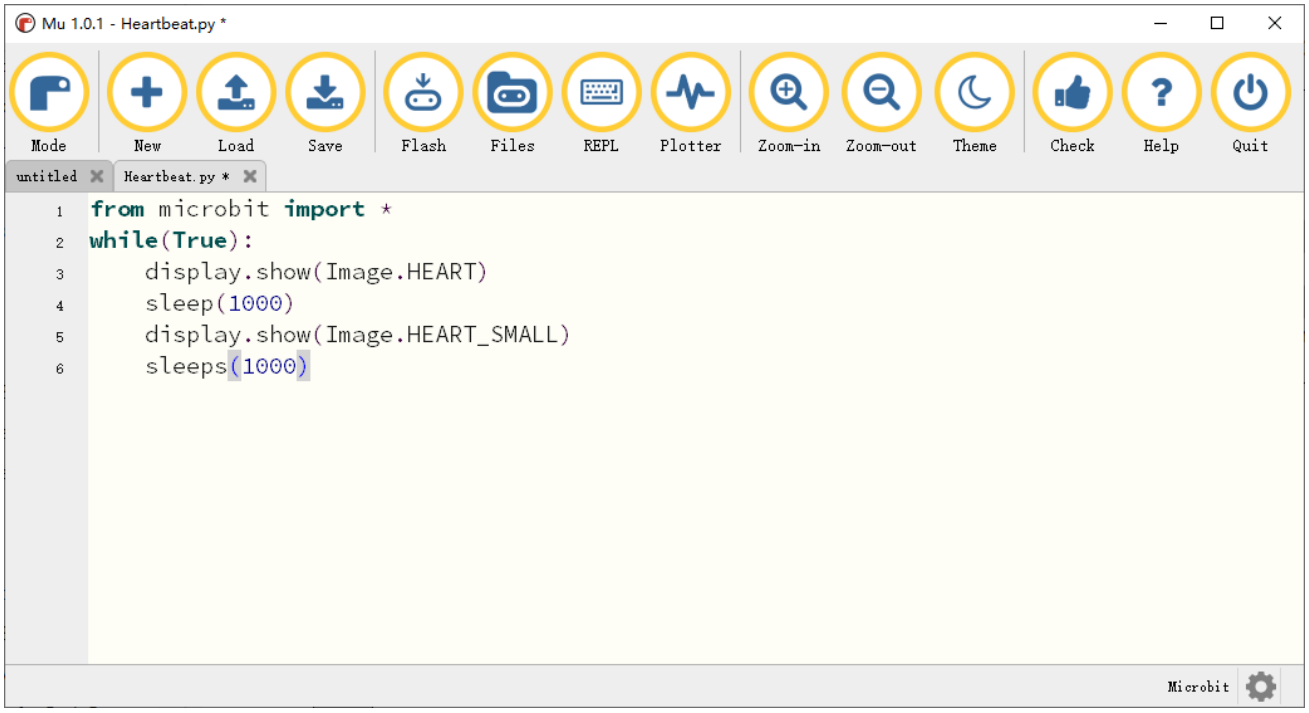
You can also type the code by yourself.



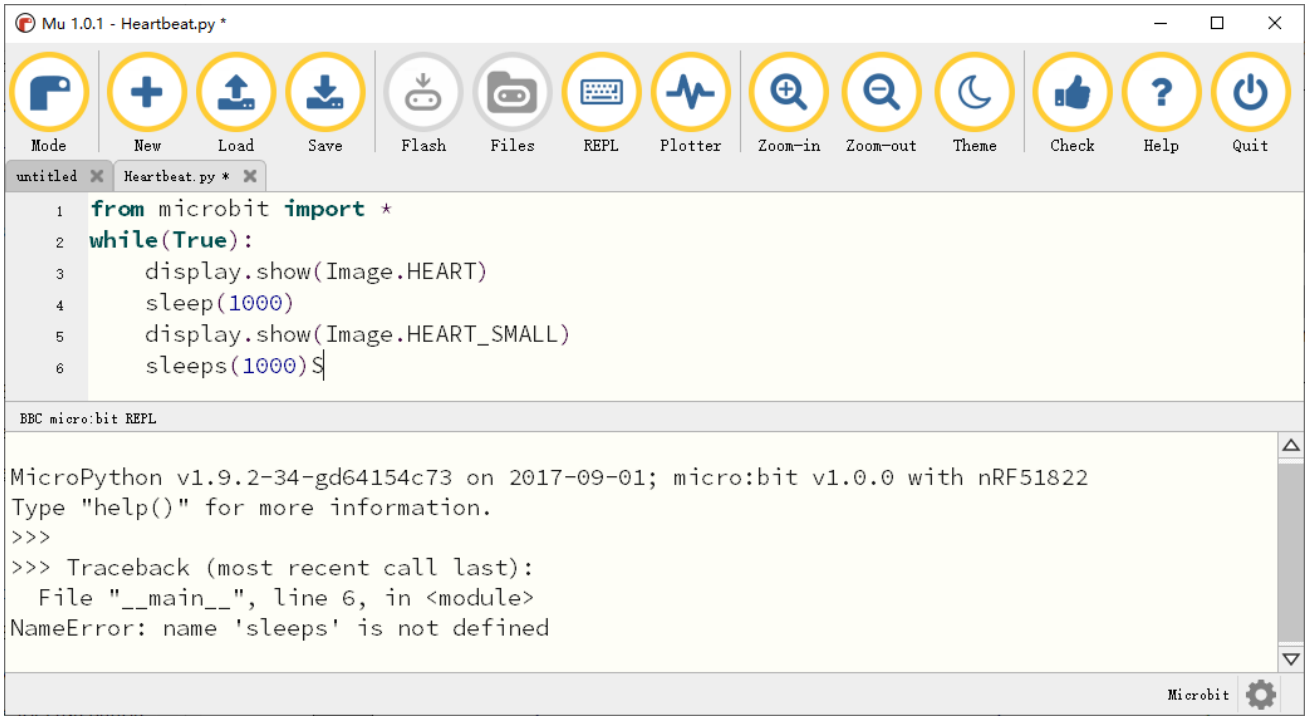
Use the micro USB cable to connect micro:bit and PC, and click the **Flash** button to download the program into micro:bit.

If there are errors in your code, you may be able to successfully download it to micro:bit, but it will not work properly.

For example, the function sleep() was written as sleeps() in the following illustration. Click the button and the code can be uploaded to Micro:bit successfully. However, after the downloading completes, LED matrix prompts some error information and the number of the wrong line.

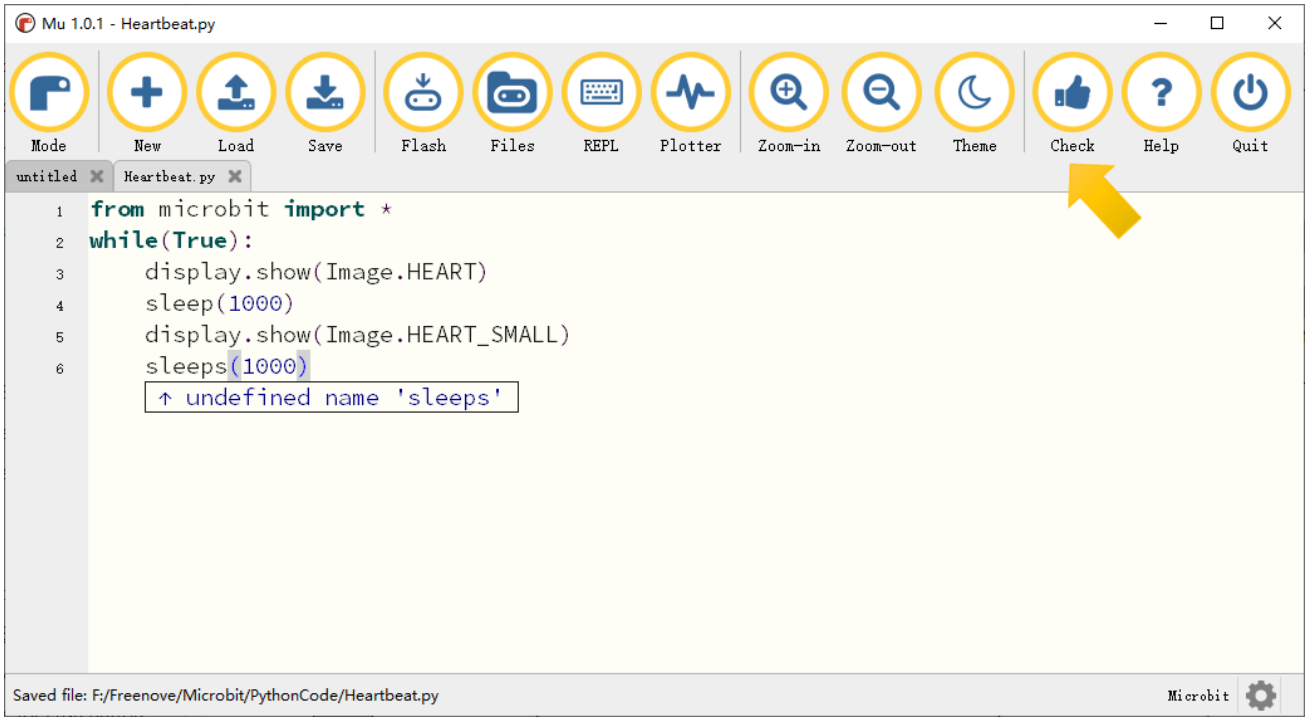


Click the “REPL” button and press the reset button (the button on the back, not A, B) on micro:bit. The error message will be displayed in the REPL box, as shown below:

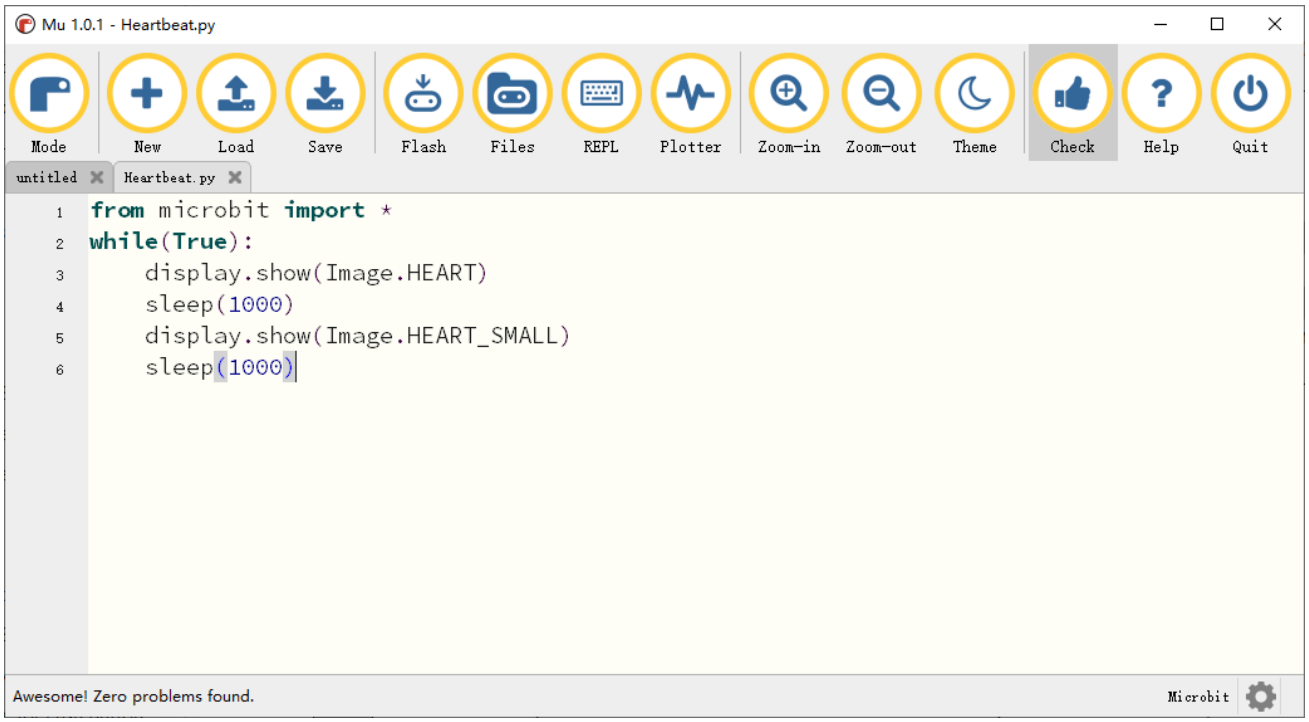


Click REPL again, you will close REPL mode. And then you can flash new code.

To ensure the code is correct, after completing the code, click the "**Check**" button to check the code for errors. As shown below, click the “Check” button. Then Mu will indicate the error of the code.



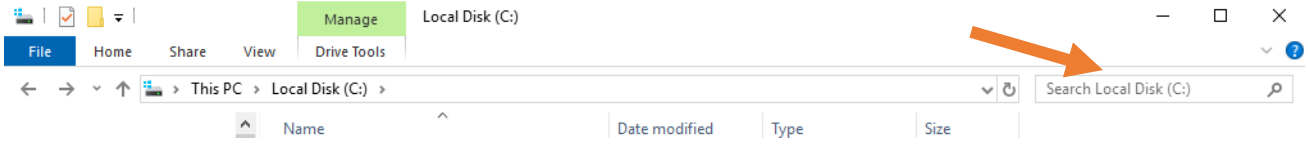
Correct the code according to the error prompt. Then click the "Check" button again, Mu displays no error on the bar below.



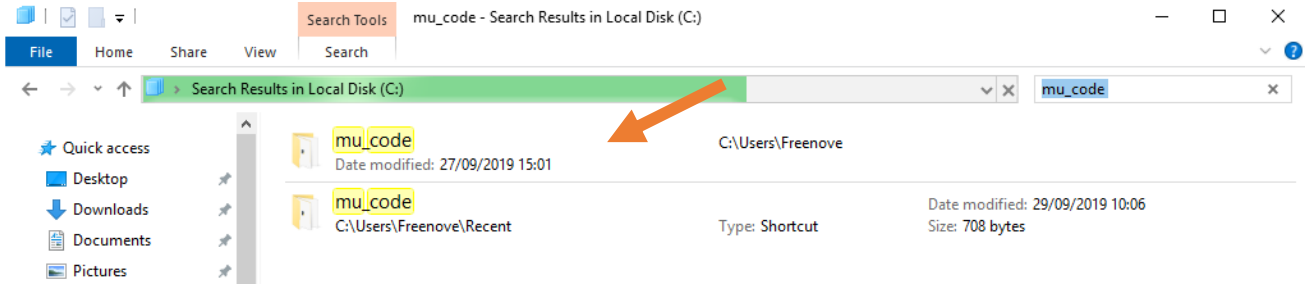
Import necessary Python file into micro:bit

In the code of this tutorial, the LCD1602 module and DHT11 module are used, so it is necessary to import "I2C_LCD1602_Class.py" and "DHT11_RW.py" into the micro:bit. You can skip this section if you don't use them. When you need, you can come back to import them.

The import method is as follows:
Search on the C drive and find the "mu_code" folder.



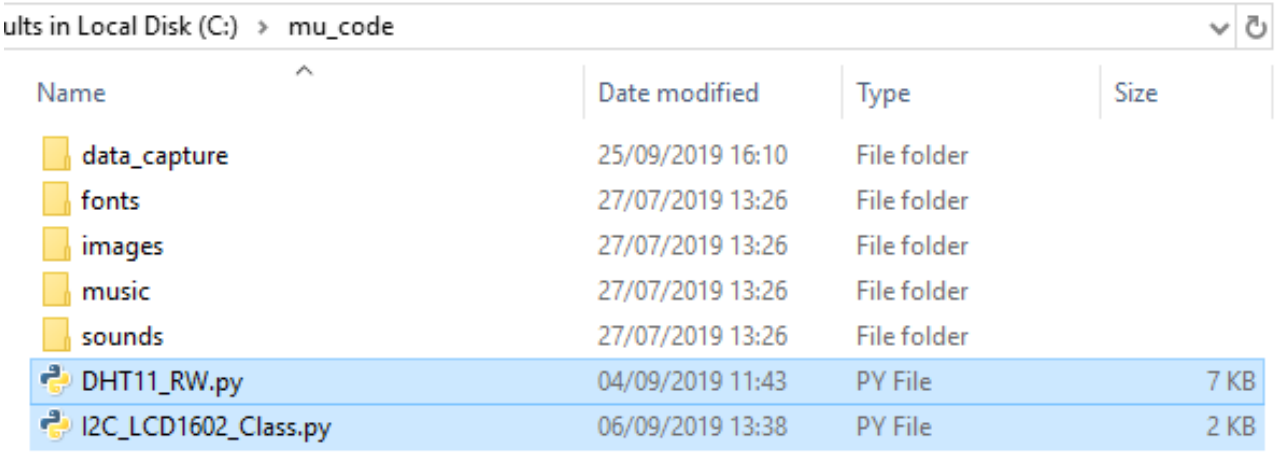
Double click on "mu_code" to enter the folder.



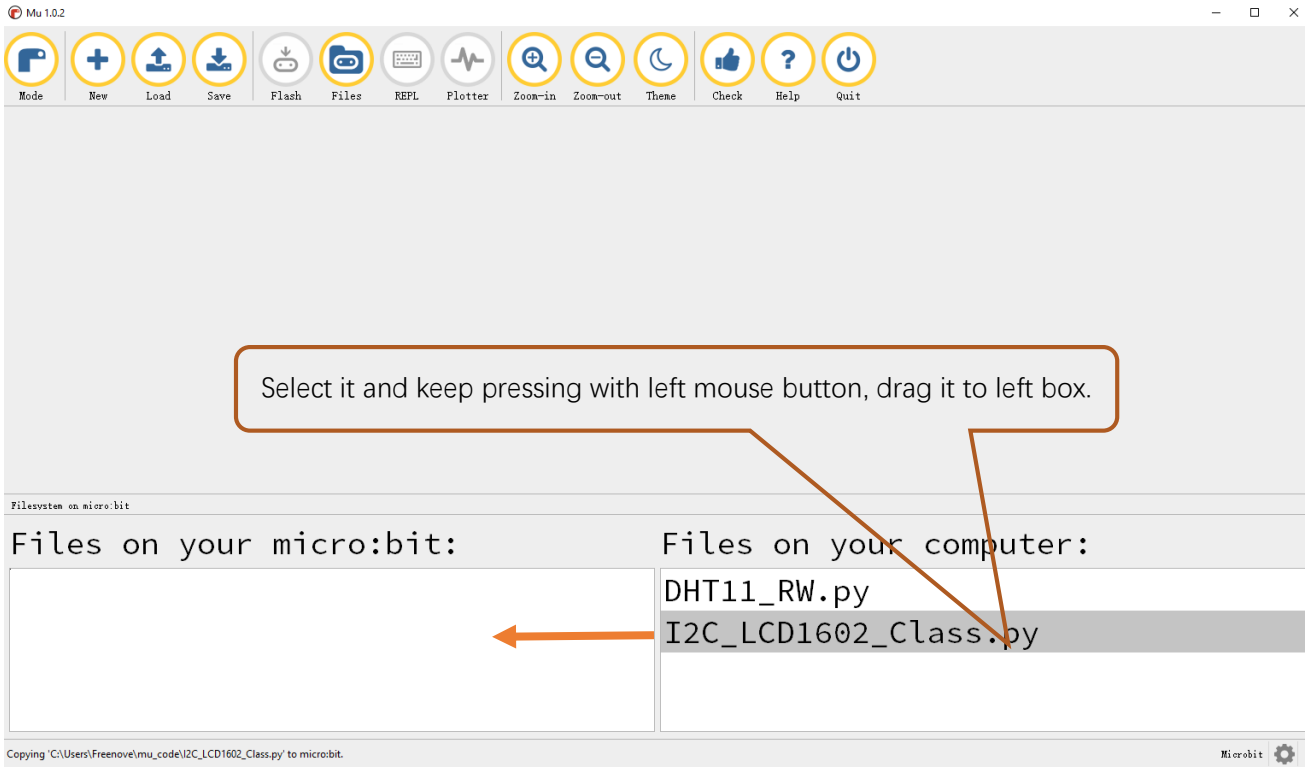
Copy "I2C_LCD1602_Class.py" and "DHT11_RW.py" from following path into "mu_code" directory.

File type	Path	File name
Python file	.. /Projects/PythonLibrary	I2C_LCD1602_Class.py DHT11_RW.py

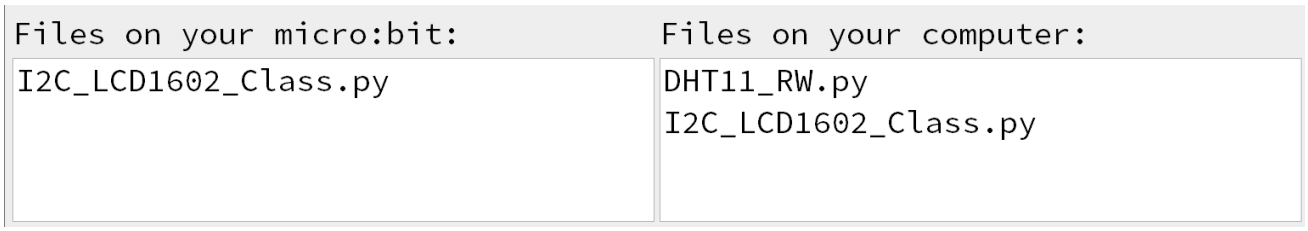
After pasting successfully, you can see them as below:



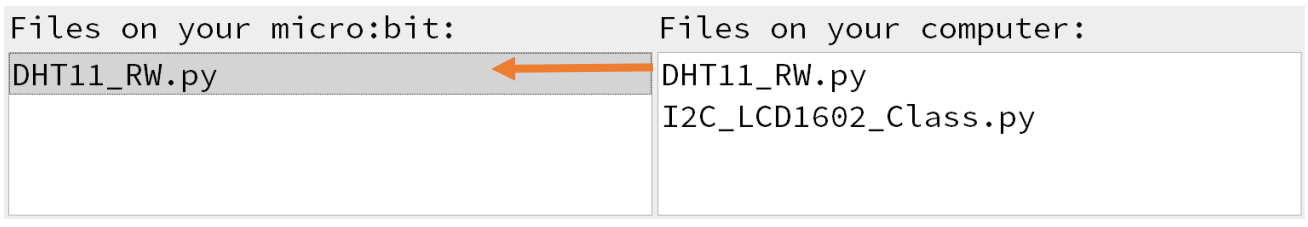
Open the Mu software, click "Files". Here we take "I2C_LCD1602_Class.py" as an example, drag "I2C_LCD1602_Class.py" into micro:bit.



After importing successfully, you will see it on the left.



The import method of "DHT11_RW.py" is the same as described above. You just need to import the one you need to use.



Note, after you upload other file into micro:bit, the original content will be covered. You need to import it next time you use it.

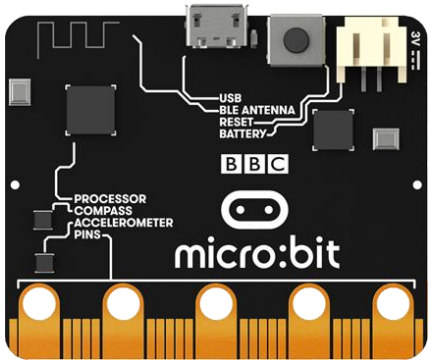
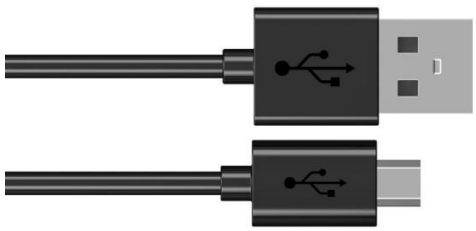
Chapter 1 LED matrix

The micro:bit integrates a 5x5 LED matrix, which is used as a display to display numbers, text, or simple images, which is useful and interesting.

Project 1.1 Heartbeat

This project uses a pattern built in MakeCode to make a heartbeat animation.

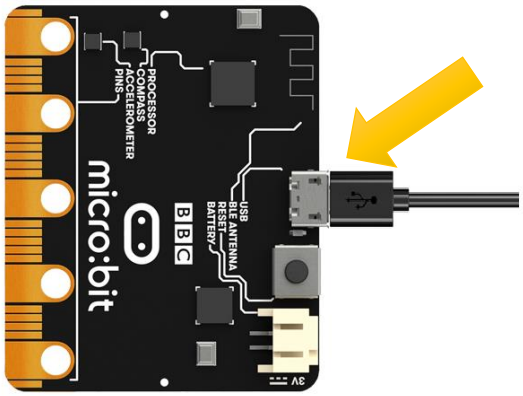
Component List

micro:bit x1 	micro USB cable x1 

Circuit

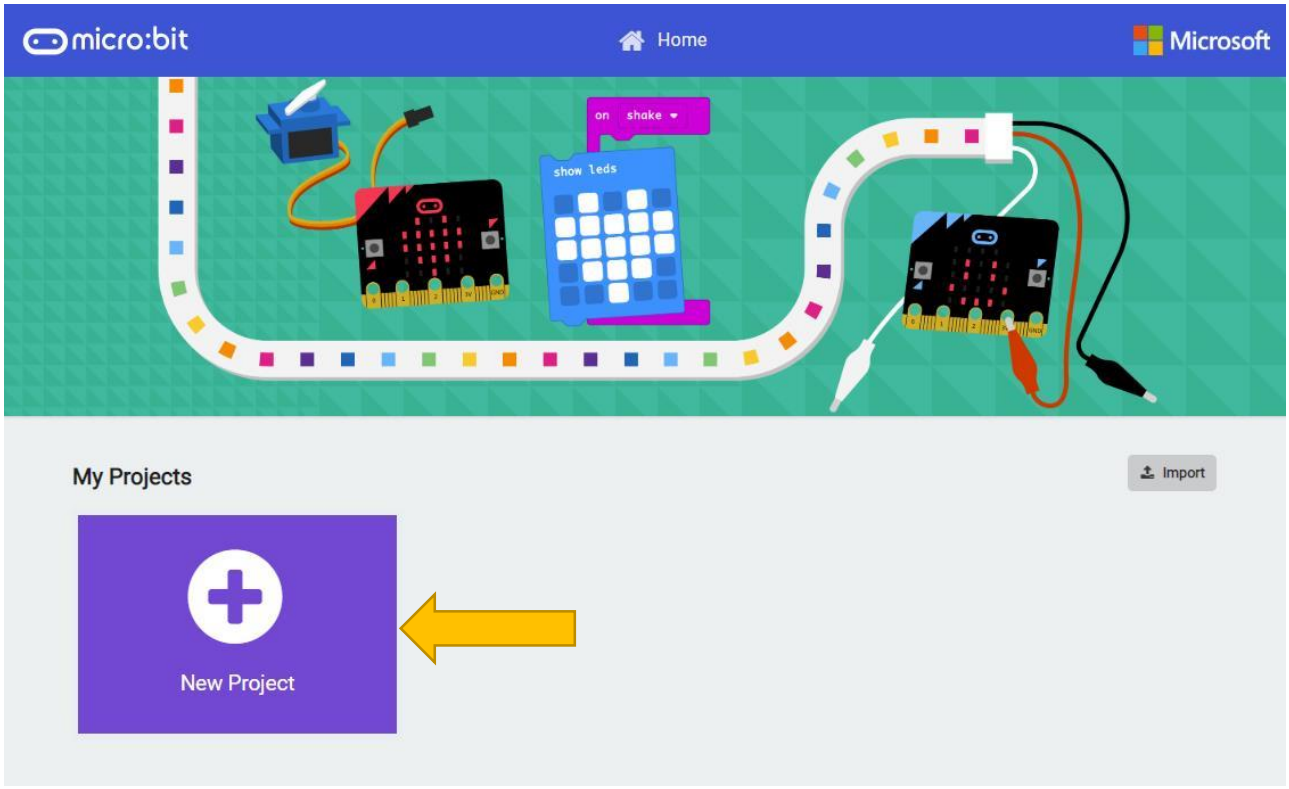
Connect micro:bit and PC via a micro USB cable.

Hardware connection

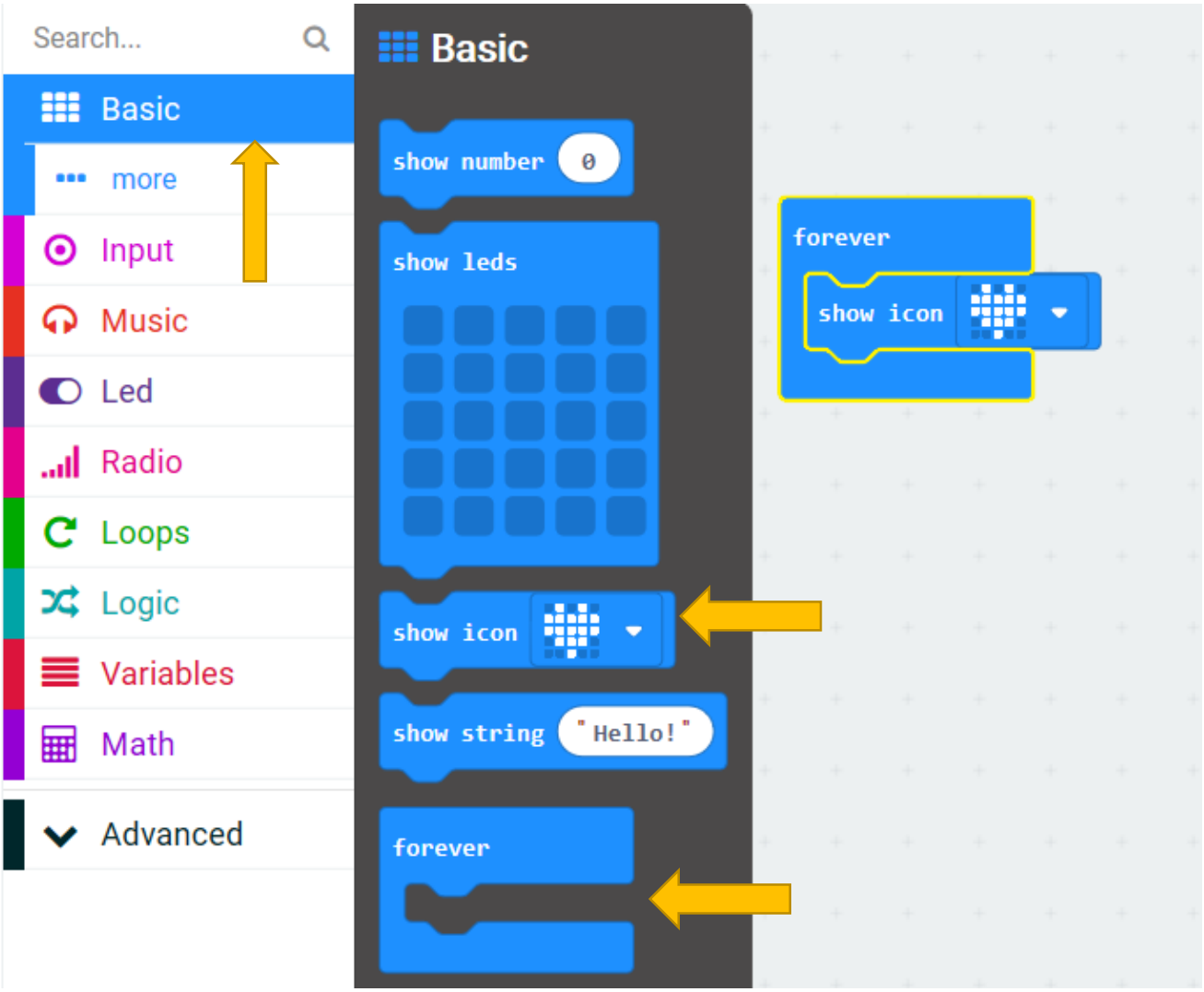


Block code

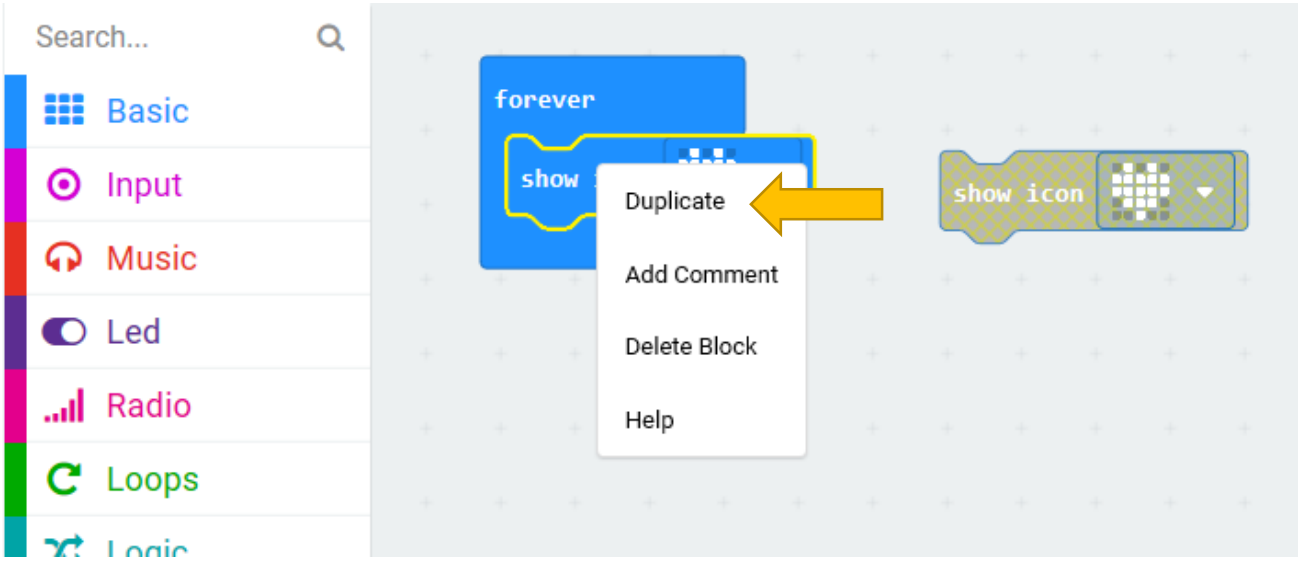
Open the MakeCode for the web version or MakeCode for the win10 version. Click on "New Project"



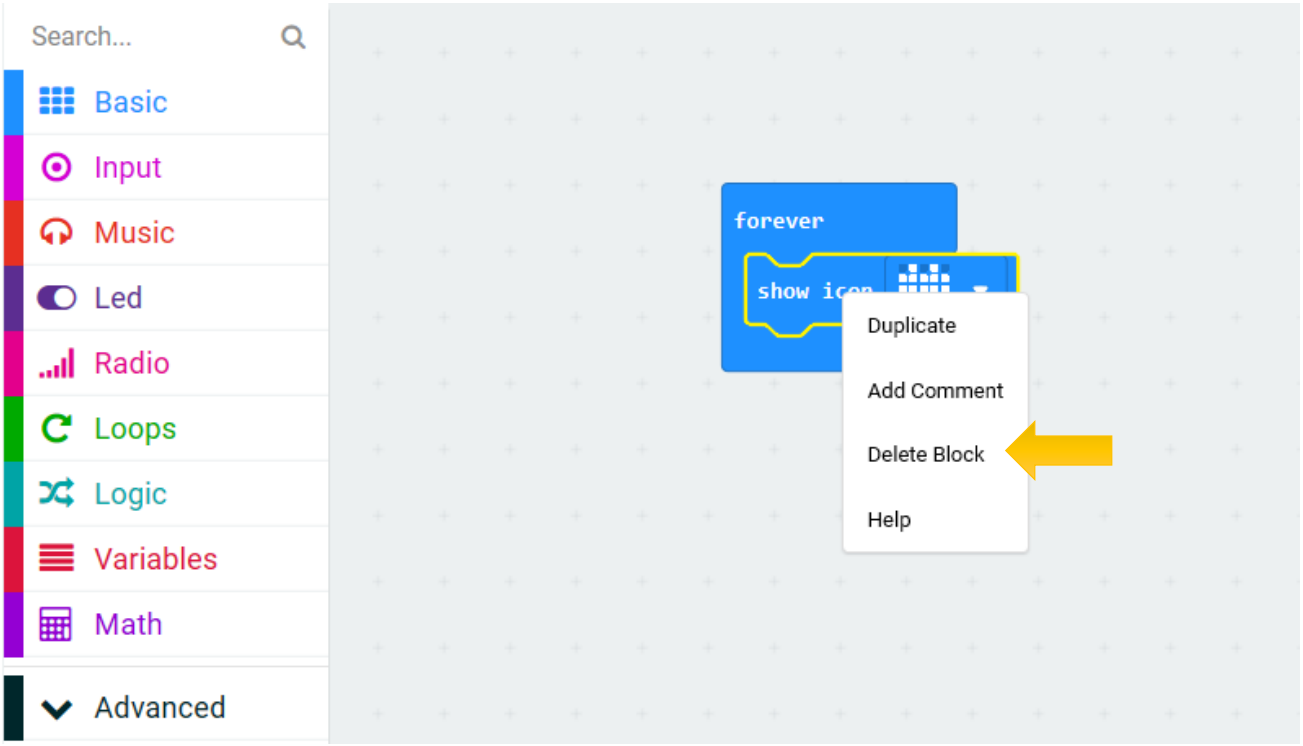
Click **basic** in the list on the left, select the desired code block, and drag it into the right code editing area.



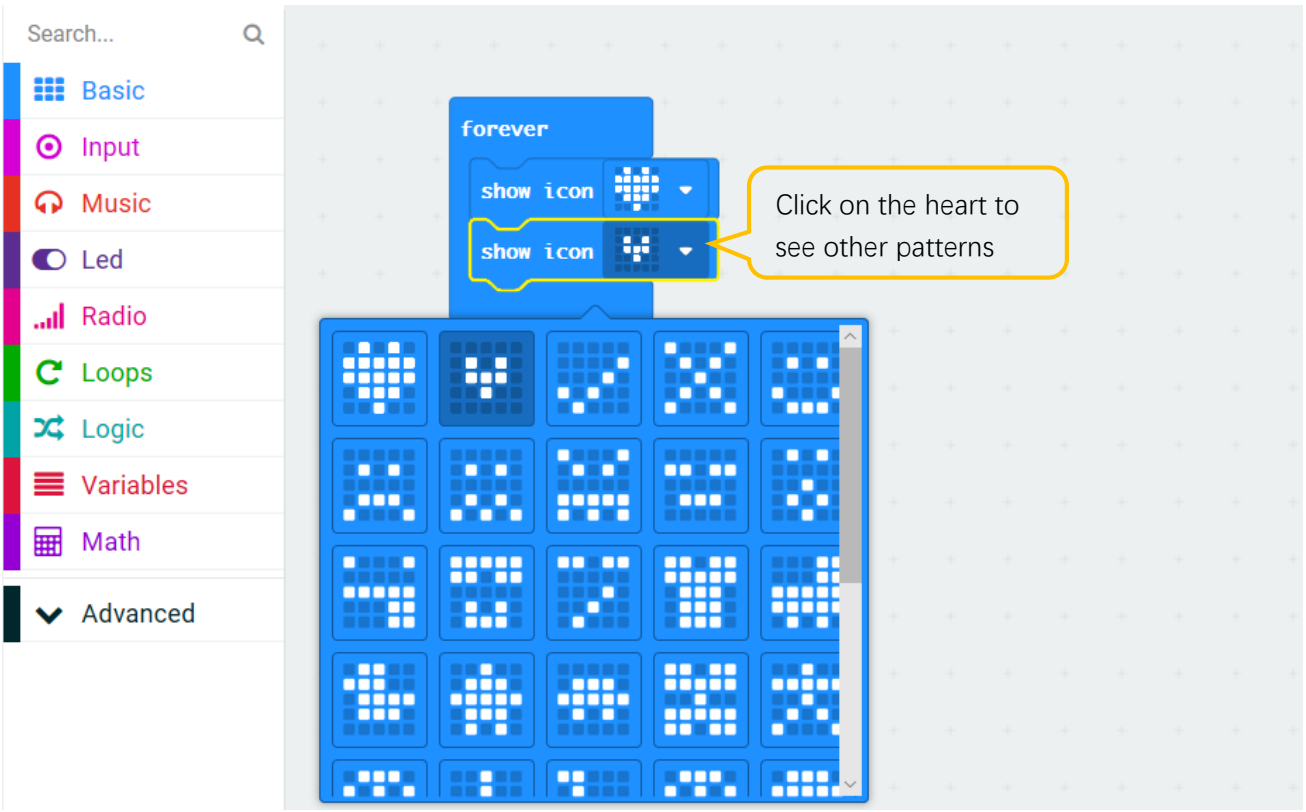
Right click the mouse and select Duplicate to duplicate the code block.



If you want to delete the block, you can right click on the block and select "Delete Block". **You can also drag it to left to delete it.**

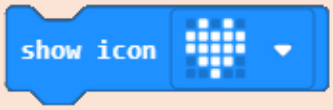


On the second block, click on the drop-down triangle next to the heart-shaped pattern on the block to display all the optional built-in patterns, select the second pattern, a small heart shape.



This completes the block code thie project.
Download the program to the microbit, the LED matrix on the micro:bit will continue to display a large heart-shaped pattern and a small heart-shaped pattern, just like heartbeating.
If you did not master downloading, please refer to contents (How to download? How to quick download?).

Reference

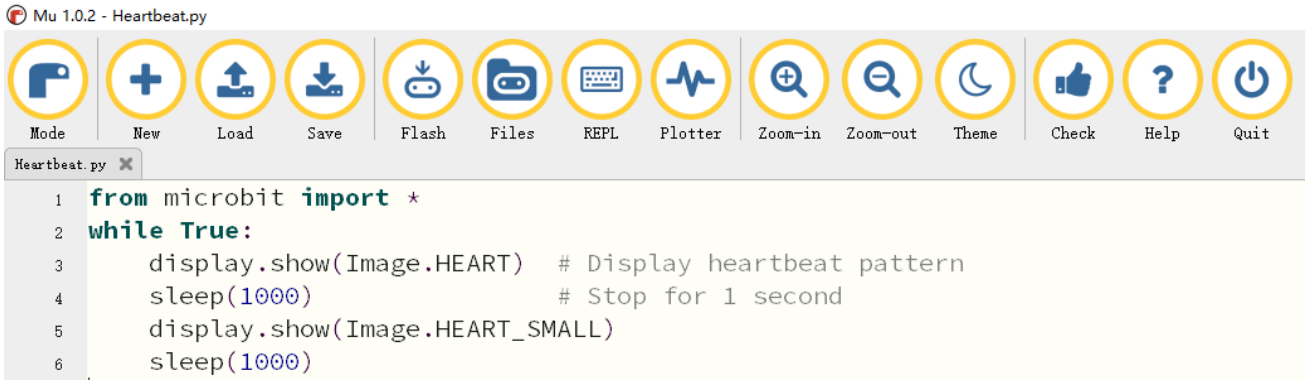
Block	Function
	Shows the selected icon on the LED screen

Python Code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/01.1_Heartbeat	Heartbeat.py

After loading successfully, the code is shown below:



Download the program to the microbit, theLED matrix on the micro:bit will continue to display a large heart-shaped pattern and a small heart-shaped pattern, just like heartbeating.

The following is the program code:

```
1 from microbit import *
2 while True:
3     display.show(Image.HEART)
4     sleep(1000)
5     display.show(Image.HEART_SMALL)
6     sleep(1000)
```

Python language is an interpreted language that is executed sequentially. In the code of this project, the micro:bit module is first imported, and then in a infinite loop statement, a large heart pattern and a small heart pattern are alternately displayed.
Next, we will explain the code line by line.

```
from microbit import *
```

Import everything in the microbit module, including functions, classes, variables, etc. You can also use **import microbit** directly. If you do this, you need to add "microbit." when you call the contents of this module in the program.

```
while True:
```

An infinite loop that will be executed circularly by microbit constantly.

```
display.show(Image.HEART)
```

Display heart pattern on LED matrix.

```
sleep(1000)
```

Delay for one second.

```
display.show(Image.HEART)
sleep(1000)
display.show(Image.HEART_SMALL)
sleep(1000)
```

Display the heart pattern on the LED matrix for one second, and then display the small heart pattern for another second.

Reference

from microbit import *

Import everything in the microbit module, including functions, classes, variables, etc. You can use all the available contents in the micro:bit module in the next program.

while True:

While is a loop statement, if the condition is true, the code in while is executed.
This code with True means that the code in the while is always executed circularly.

display.show(image)

Display the image.
For more details about display,
please refer to: https://microbit-micropython.readthedocs.io/en/latest/display.html
For more details about image,
Please refer to: https://microbit-micropython.readthedocs.io/en/latest/image.html

sleep(t)

Delay for given number of milliseconds, should be positive or 0.
For more details about sleep function, please refer to:
https://microbit-micropython.readthedocs.io/en/latest/utime.html

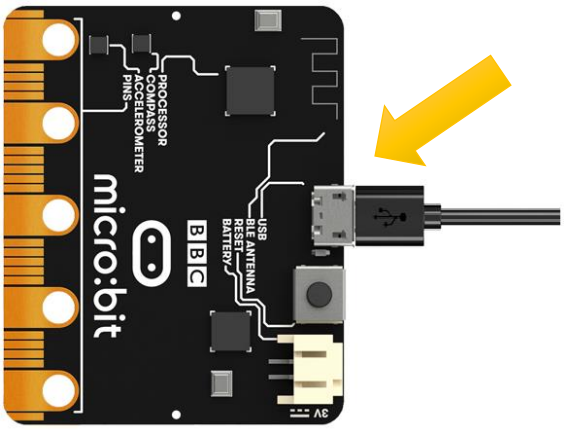
Project 1.2 Displaying Number

In this project, we will use the LED matrix of the micro:bit to display numbers.

Circuit

Connect micro:bit and PC via a micro USB cable.

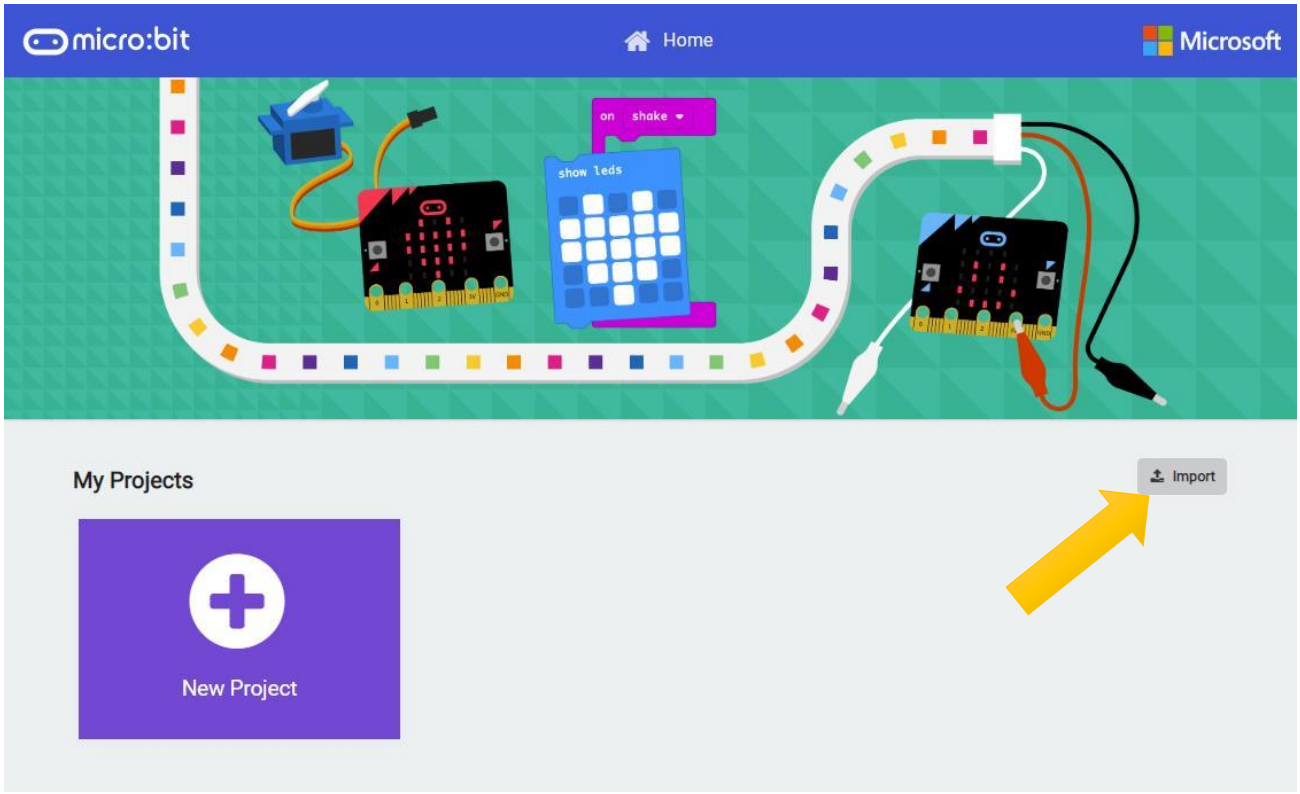
Hardware connection



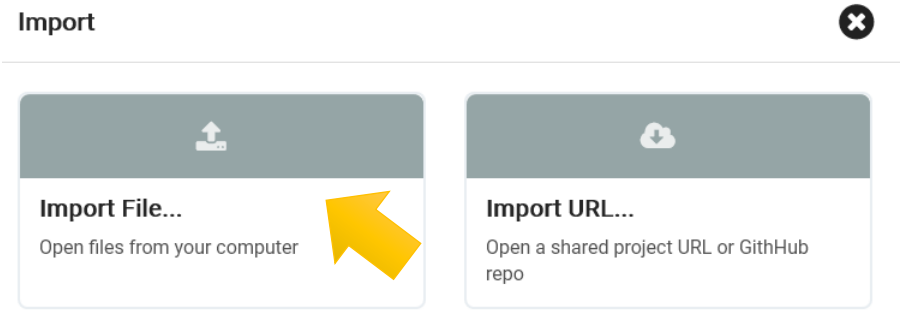
Block code

Open MakeCode first.

In this project, we will import the block code.



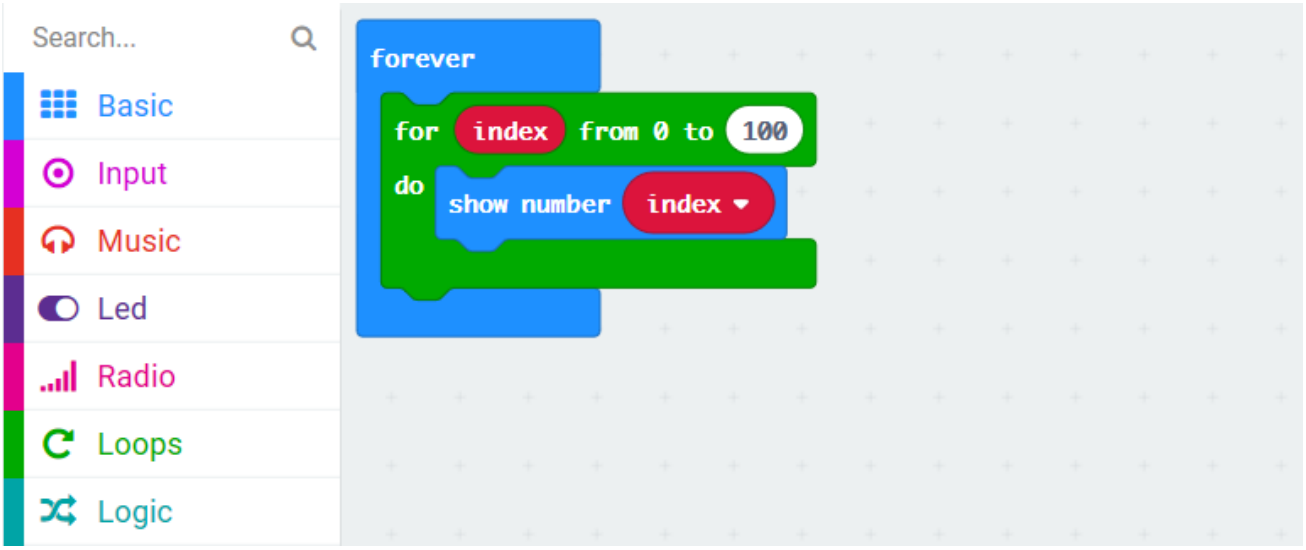
Click **Import**. Then click **Import File**.



Import the .hex file. The path is as below:

File type	Path	File name
HEX file	../ Projects/BlockCode/01.2_ShowNumber	ShowNumber.hex

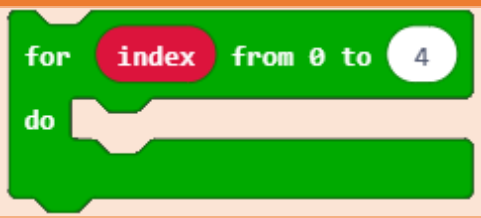
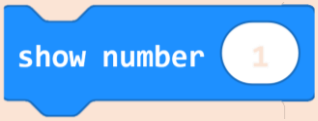
After load successfully, the code is shown as below:



Download the code into the micro:bit. After the downloading completes, the micro:bit LED matrix will start to display the numbers 1, 2, 3, 4...100. Then start again from 1 to 100, so that it will cycle permanently.

In this code, a for loop is used. Each time the loop is executed, the value of the variable index is increased by 1. When the value is greater than 100, the for loop is exited. In the body of the loop, the value of the numeric index is displayed.

Reference

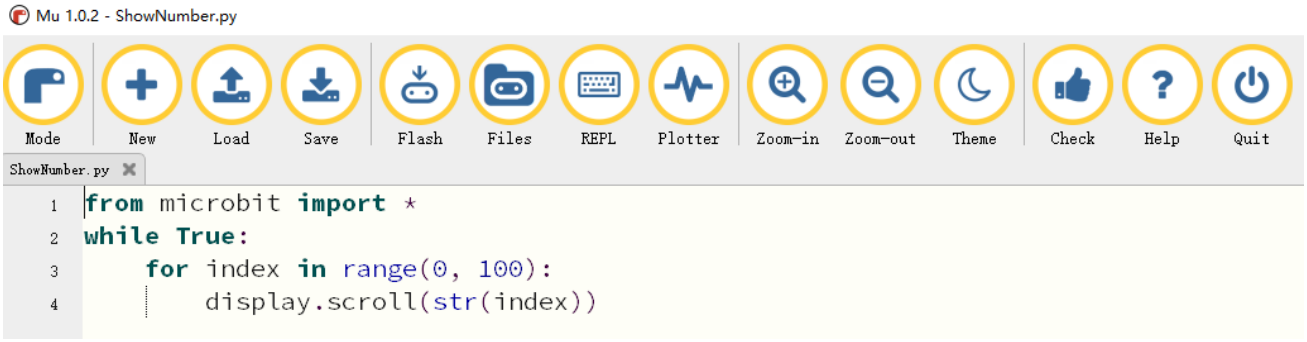
Block	Function
	This is a for loop, the number (4) of loops can be changed, each time the index is incremented by 1. The loop won't end until the index is greater than the set value.
	Show a number on the LED screen. It will slide left if the number is more than one digit..

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/01.2_ShowNumber	ShowNumber.py

After loading successfully, the code is shown as below:



Download the code into the microbit. After the downloading completes, the micro:bit LED matrix will start to display the numbers 1, 2, 3, 4...100. Then start again from 1 to 100, so that it will repeat endlessly.

The following is the program code:

```
1 from microbit import *
2 while True:
3     for index in range(0, 100):
4         display.scroll(str(index))
```

The code of this project, in a 0-100 for loop, scrolls through the cyclic number index, which is incremented by 1.

Reference

display.scroll(value)

Scrolls value horizontally on the display. If value is an integer or float it is first converted to a string using str().

For more information, please refer to: https://microbit-micropython.readthedocs.io/en/latest/utime.html

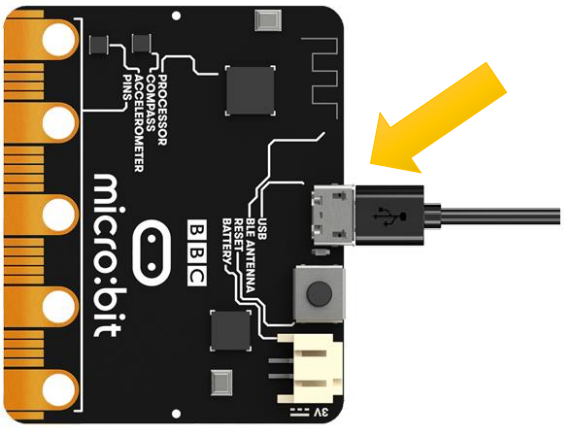
Project 1.3 Displaying Text

This project uses the LED matrix of micro:bit to display text (ASCII).

Circuit

Connect micro:bit and PC via micro USB cable.

Hardware connection



Block code

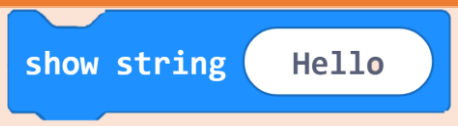
Open MakeCode first.Import the .hex file. The path is as below:(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/01.3_ShowText	ShowText.hex

After loading successfully, the code is shown as below:

Download the code into the microbit, the micro:bit LED matrix will scroll from left to right to display "Hello, Freenove!"

Reference

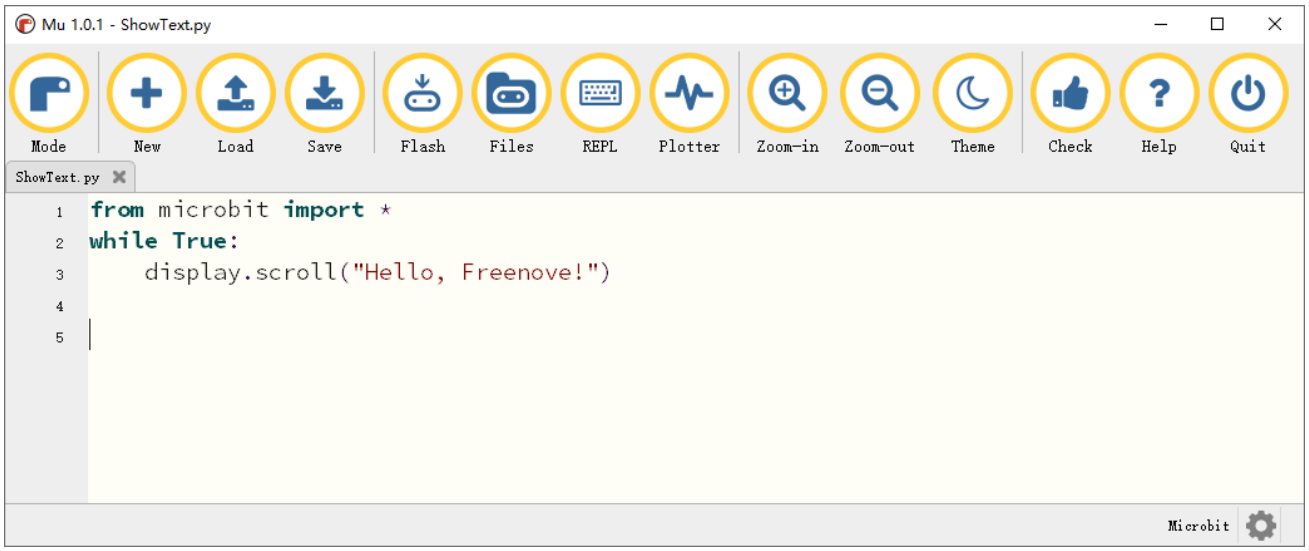
Block	Function
	Displays a string on the LED screen. It will scroll to left if it's beyond the screen.

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/01.3_ShowText	ShowText.py

After loading successfully, the code is shown as below:



Download the code into the microbit, the micro:bit LED matrix will scroll from left to right to display "Hello, Freenove!"

The following is the program code:

```
1 from microbit import *
2 while True:
3     display.scroll("Hello, Freenove!")
```

This code scrolls through the text "Hello, Freenove!" in a while loop.

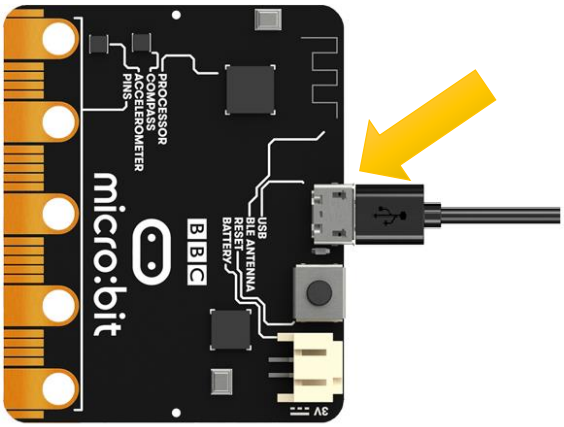
Project 1.4 Displaying Custom

This project uses a micro:bit LED matrix to display a custom pattern.

Circuit

Connect micro:bit and PC via micro USB cable.

Hardware connection



Block code

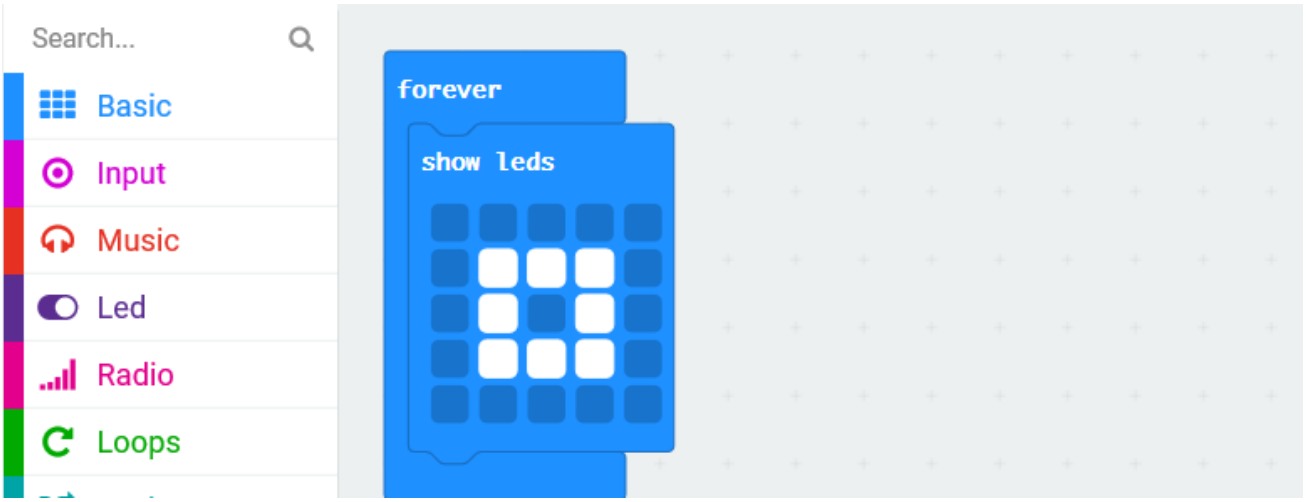
Open MakeCode first.

Import the .hex file. The path is as below:

(How to import project)

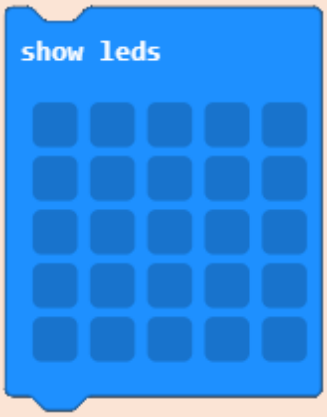
File type	Path	File name
HEX file	../Projects/BlockCode/01.4_ShowCustom	ShowCustom.hex

After loading successfully, the code is shown as below:



Check the connection of the circuit and verify it correct.. Then download the code into the microbit, and the square pattern shown above will appear on the LED matrix of the micro:bit.

Reference

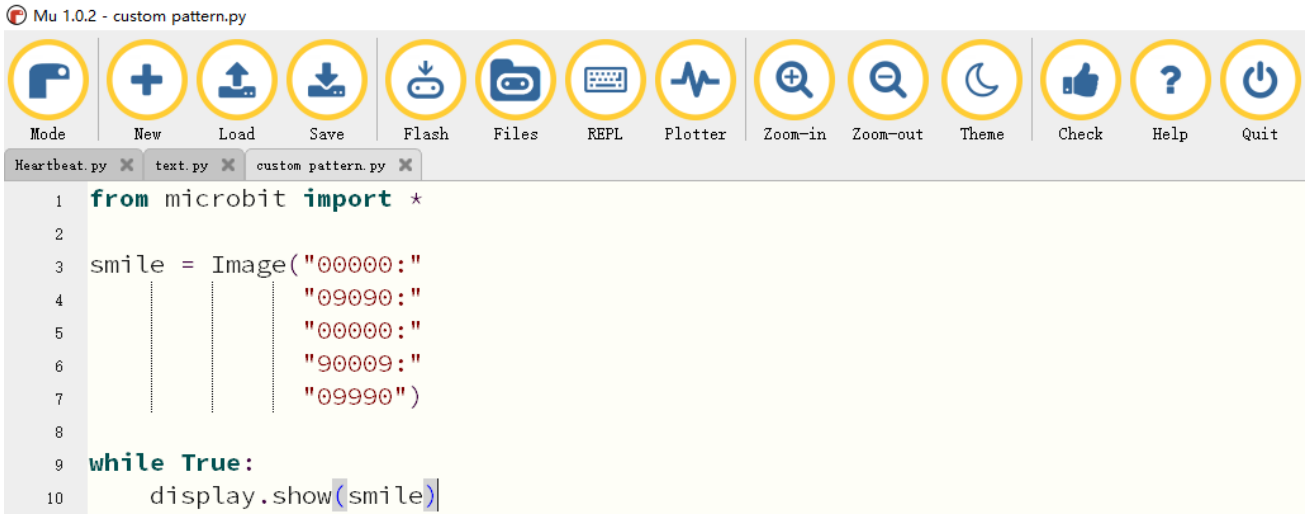
Block	Function
	Shows a picture on the LED screen.

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/01.4_ShowCustom	ShowCustom.py

After loading successfully, the code is shown as below:



Download the code into the microbit, and a square pattern will appear on the micro:bit LED matrix.
(How to download?)

The following is the program code:

```
from microbit import *

img = Image("00000:"
            "09990:"
            "09090:"
            "09990:"
            "00000")

while True:
    display.show(img)
```

Create an image in the code and define it as **img**, then display the defined image in a while loop. As shown in the code below, the parameters in Image consist of 5 strings. Each line of characters corresponds to a row of LEDs. Each digit represents the brightness of an LED. The value ranges from 0 to 9, the larger the number, the brighter the LED.

```
img = Image("00000:"
            "09990:"
            "09090:"
            "09990:"
            "00000")
```

Reference

```
img = Image("00000:"
            "09990:"
            "09090:"
            "09990:"
            "00000")
```

Create an image of LED, and set brightness of LED.

For more information, please refer to:

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

Chapter 2 Built-in Button

Keyboards or buttons are important tools for human-computer interaction. We often use keyboards to enter text, type commands, control devices, etc. Two programmable buttons A and B are integrated on the micro:bit to easily control the micro:bit to make actions.

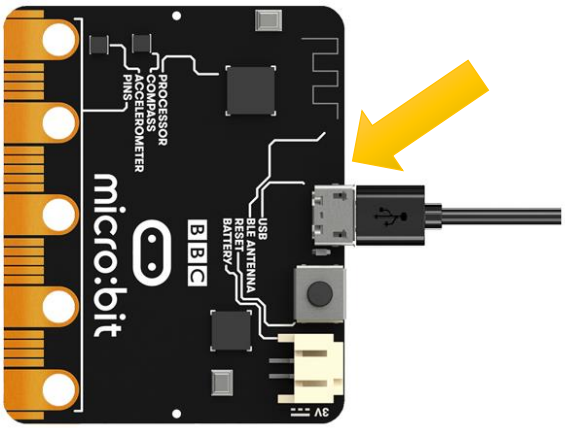
Project 2.1 Button A and B

This project uses micro:bit integrated buttons A and B. When different buttons are pressed, micro:bit displays different patterns.

Circuit

Connect micro:bit and PC via micro USB cable.

Hardware connection



Block code

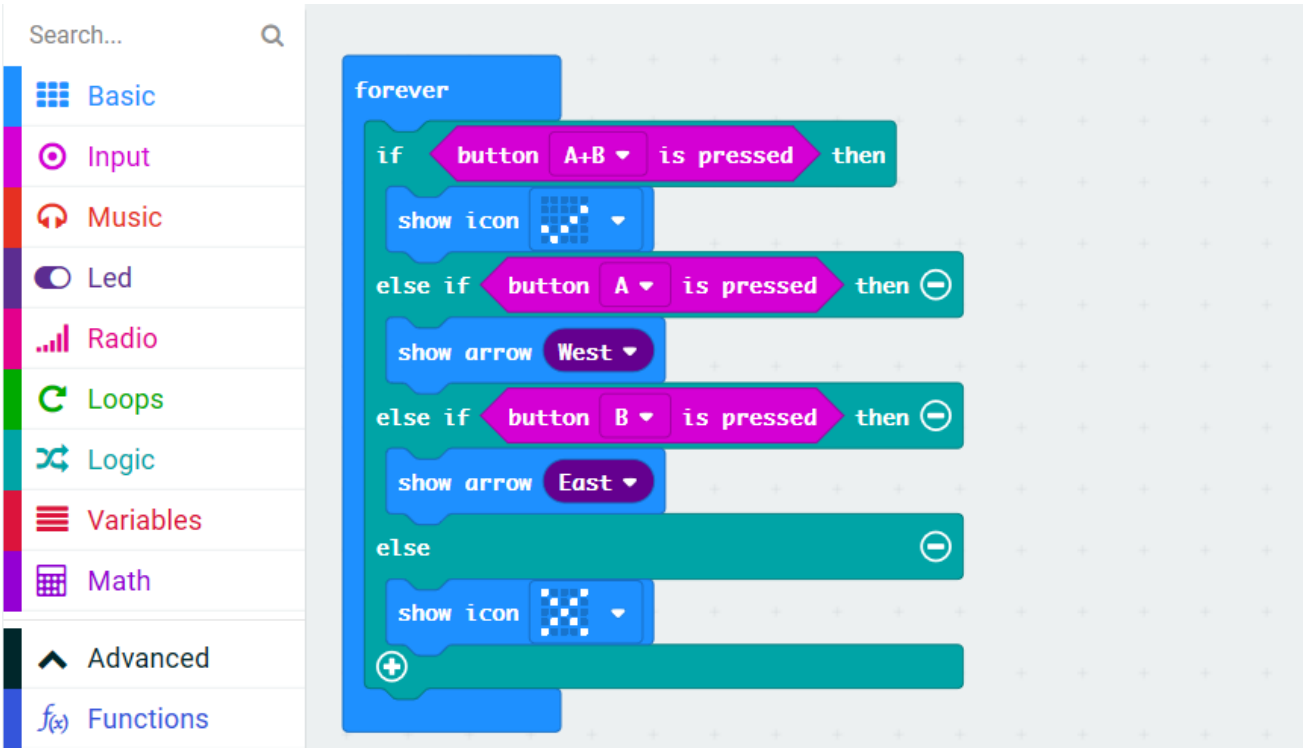
Open MakeCode first.

Import the .hex file. The path is as below:

(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/02.1_BuiltInButton	BuiltInButton.hex

After loading successfully, the code is shown as below:



Download the code into micro:bit. When button A is pressed, the micro:bit LED matrix will display an arrow pointing to button A. When button B is pressed, the micro:bit LED matrix will display an arrow pointing to button B. When the buttons A and B are pressed at the same time, the micro:bit LED matrix will display a check mark. When no button is pressed, the micro:bit LED matrix displays a cross.

Reference

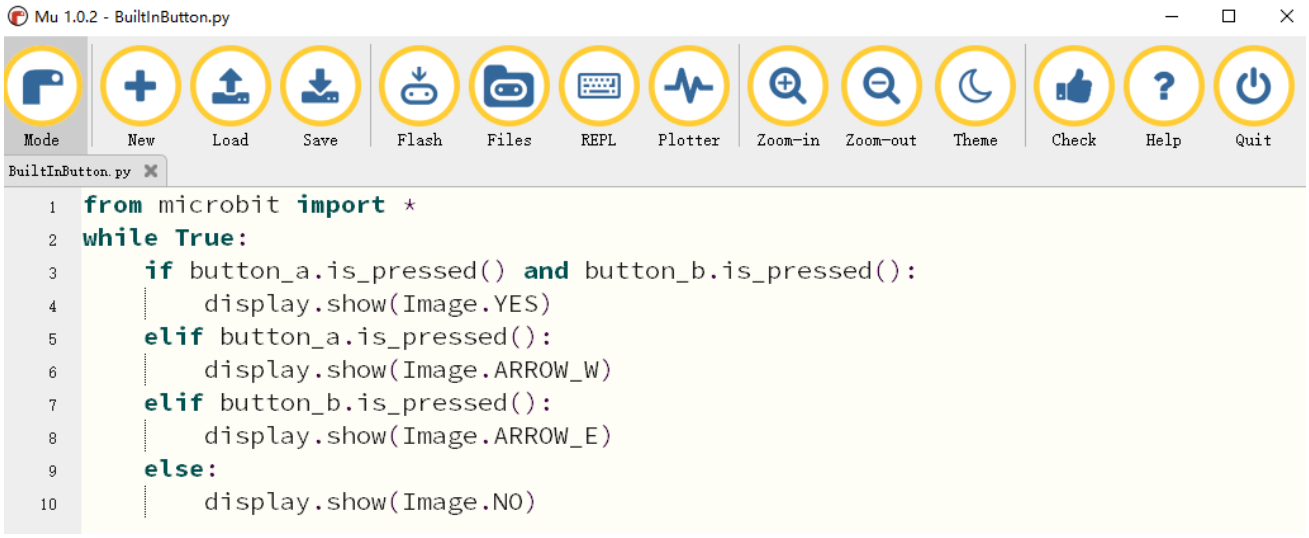
Block	Function
button A is pressed	Check whether a button is pressed at the moment. The micro:bit has two buttons: button A and button B.
on button A pressed	This handler works when button A or B is pressed, or A and B together.

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/02.1_BuiltInButton	BuiltInButton

After loading successfully, the code is shown as below:



Download the code into micro:bit. When button A is pressed, the micro:bit LED matrix will display an arrow pointing to button A. When button B is pressed, the micro:bit LED matrix will display a an arrow pointing to button B.When the buttons A and B are pressed at the same time, the micro:bit LED matrix will display a check mark. When no button is pressed, the micro:bit LED matrix displays a cross.

The following is the program code:

```
1  from microbit import *
2
3  while True:
4      if button_a.is_pressed() and button_b.is_pressed():
5          display.show(Image.YES)
6      elif button_a.is_pressed():
7          display.show(Image.ARROW_W)
8      elif button_b.is_pressed():
9          display.show(Image.ARROW_E)
10     else:
11         display.show(Image.NO)
```

Use the if-elif-else statement to determine when the button is pressed. First, when the buttons A and B are pressed at the same time, a check mark is displayed.

```
    if button_a.is_pressed() and button_b.is_pressed():
        display.show(Image.YES)
```

Then, determine in turn if the buttons A or B is pressed seperately, and the case where no button is pressed.

```
    elif button_a.is_pressed():
        display.show(Image.ARROW_W)
    elif button_b.is_pressed():
        display.show(Image.ARROW_E)
    else:
        display.show(Image.NO)
```

Note that it is necessary to first determine if buttons A and B are pressed at the same time. If-elif-else statement will make the micro:bit execute only one situation. If the state with two buttoon pressed is placed in last, the result of pressing A or B will appear first, then the statement will end, and then sentence met the state with two button pressed will never be executed.

Reference

is_pressed()
Returns True if the specified button is currently being pressed, and False otherwise. For more information, please refer to https://microbit-micropython.readthedocs.io/en/latest/button.html

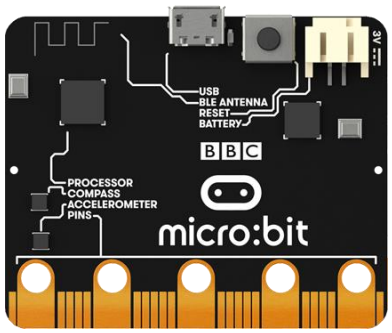
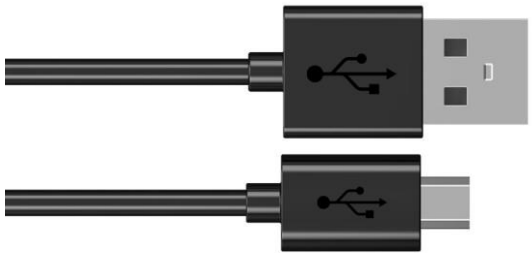
Chapter 3 Serial Communication

In this chapter, we will learn how to use serial port.

Project 3.1 Display the Data

This project uses serial ports to transmit data and display data.

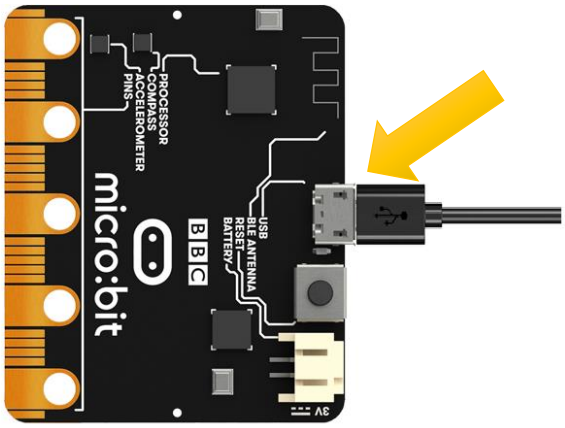
Component list

Microbit x1	USB cable x1
	

Circuit

Connect micro:bit and PC via a micro USB cable.

Hardware connection

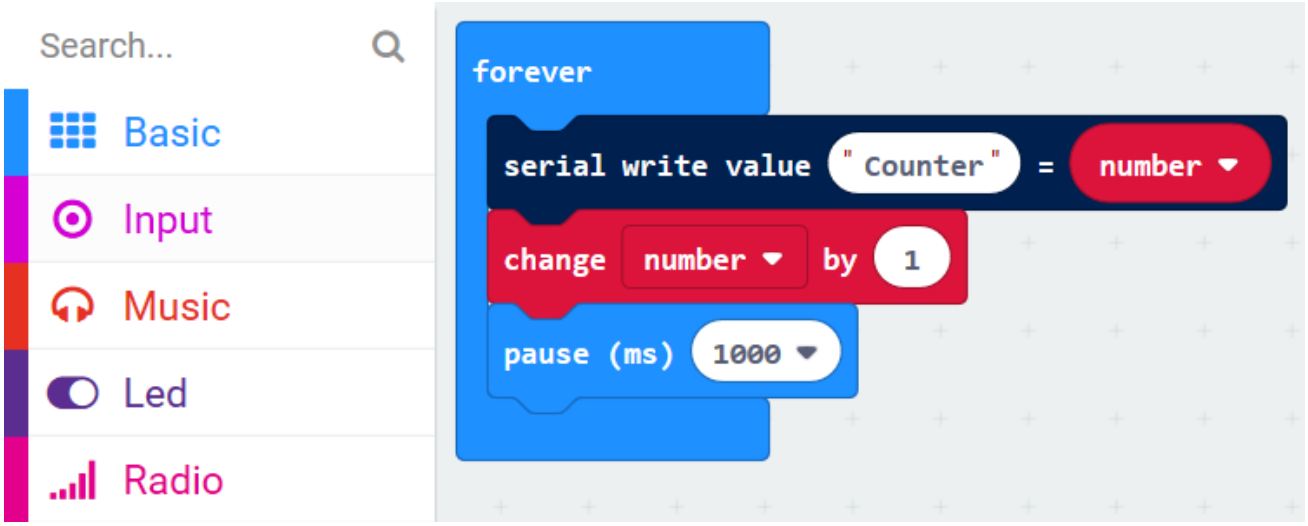


Block code

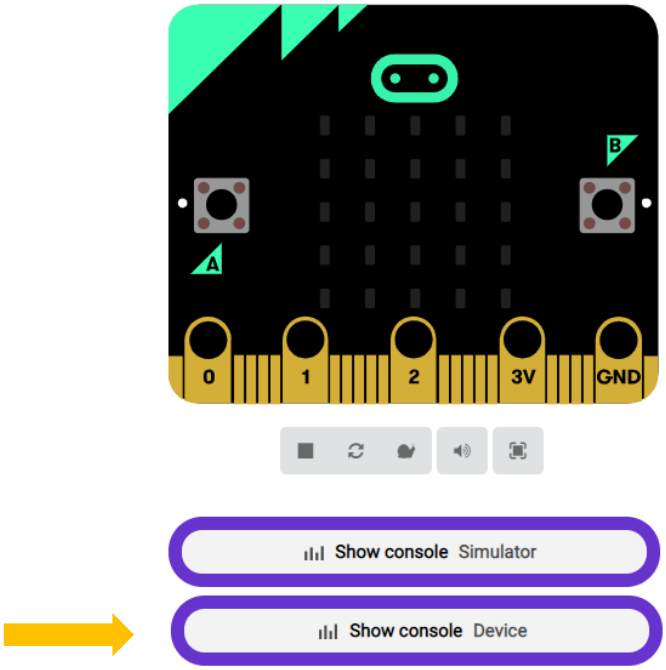
Open MakeCode first. Import the .hex file. The path is as below:
(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/03.1_SerialPort	SerialPort.hex

After importing successfully, the code is shown as below:



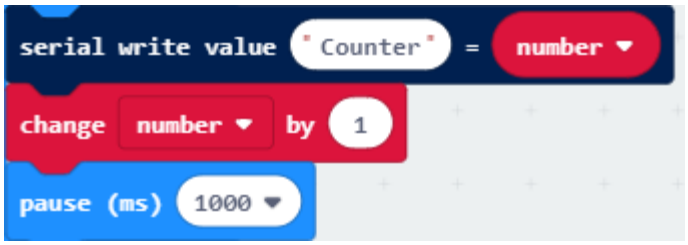
Check the connection of the circuit and verify it correct download the code into the micro:bit, and then open the serial controller, as shown below:



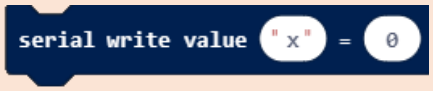
On the serial console, you can see the data sent by the microbit.


```
Counter:162
Counter:163
Counter:164
Counter:165
Counter:166
Counter:167
Counter:168
```

Every 1 second, the value of the variable number is incremented by 1, and the new value will be sent to the serial port.



Reference

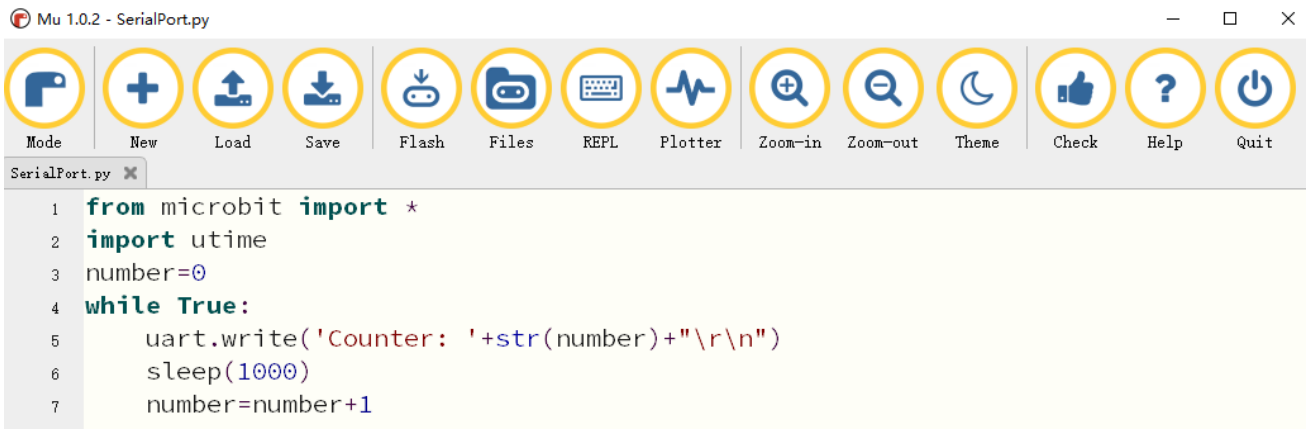
Block	Function
	Write a name:value pair and a newline character (\r\n) to the serial port.
	The change blocks increase the value in the variable by the amount you want. This is also known as an addition assignment operation.

Python code

Open the .py file with Mu. Code, the path is as below:

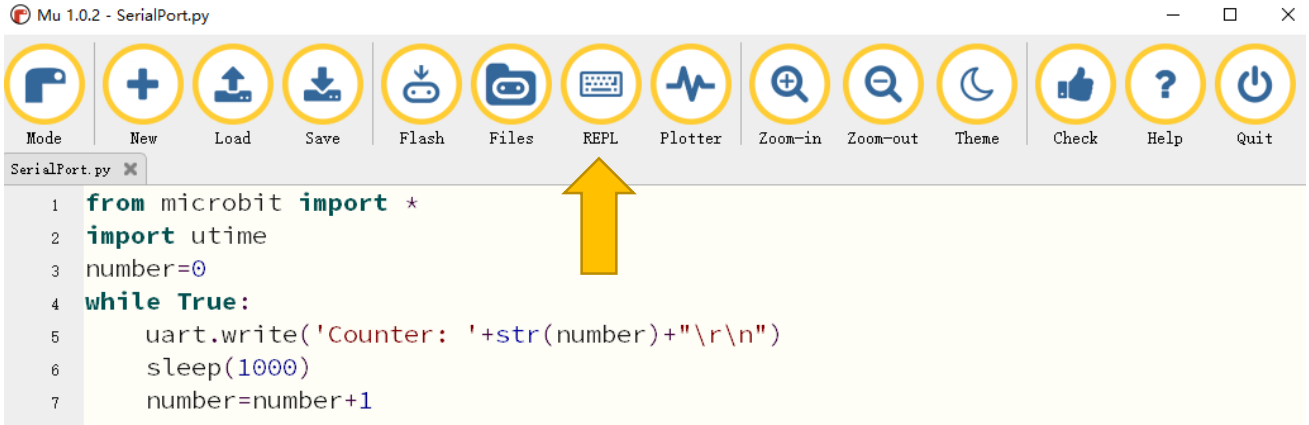
File type	Path	File name
Python file	../Projects/PythonCode/03.1_SerialPort	SerialPort.py

After loading successfully, the code is shown as below:

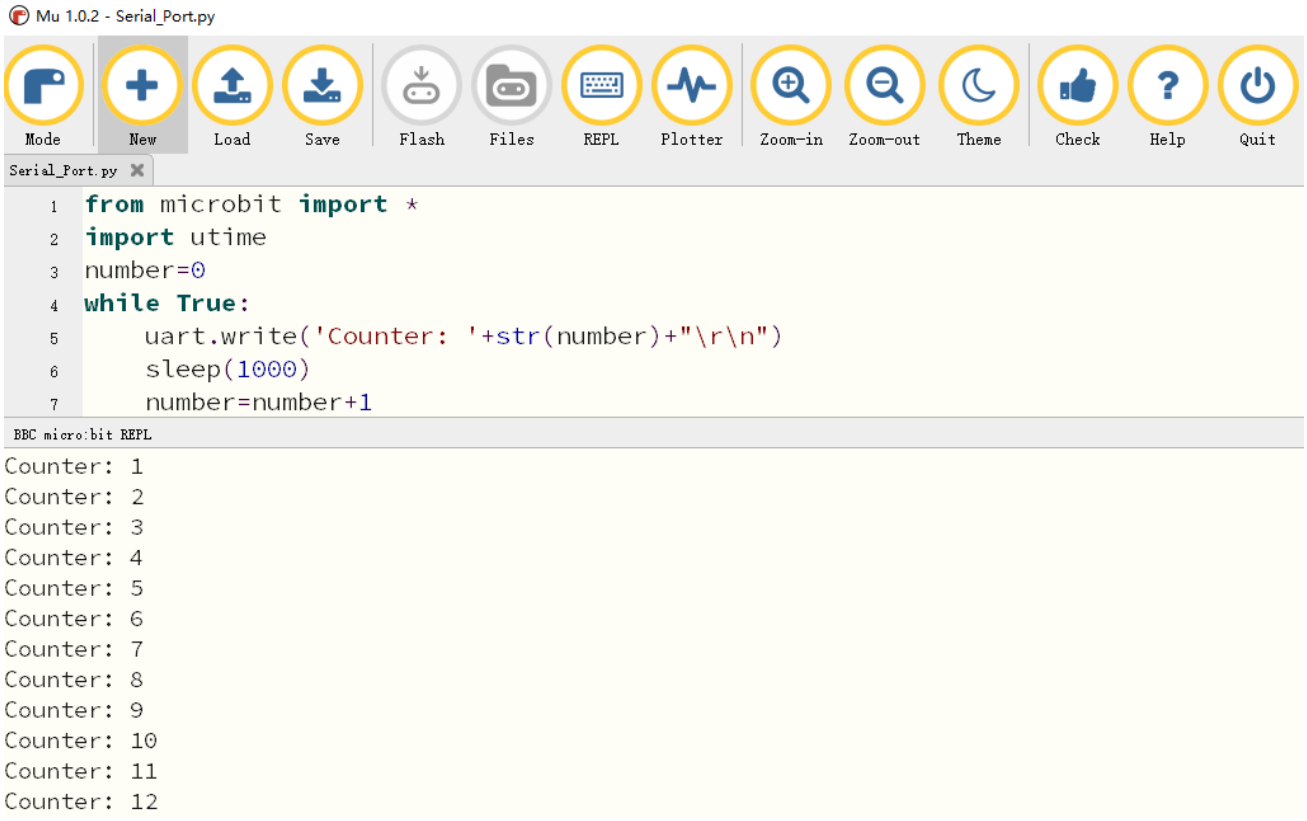


Check the connection of the circuit and verify it correct and then download the code into the micro:bit.

After the program is downloaded, click on REPL as shown below.



Then press the reset button (the button on the back) of the Micro:bit and we will see the change of value.



The following is the program code:

```
1 from microbit import *
2 import utime
3 number=0
4 while True:
5     uart.write('Counter: '+str(number)+ "\r\n")
6     sleep(1000)
7     number=number+1
```

Every 1 second, the value of the variable number is incremented by 1, and the new value will be sent to the serial port, where "\r\n" is the meaning of the newline.

```
uart.write('Counter: '+str(number)+ "\r\n")
sleep(1000)
number=number+1
```

Reference

uart.write(x)

Write the buffer to the bus, it can be a bytes object or a string:
uart.write('hello world')
uart.write(b'hello world')
uart.write(bytes([1, 2, 3]))
For more information, please refer to:
https://microbit-micropython.readthedocs.io/en/latest/uart.html

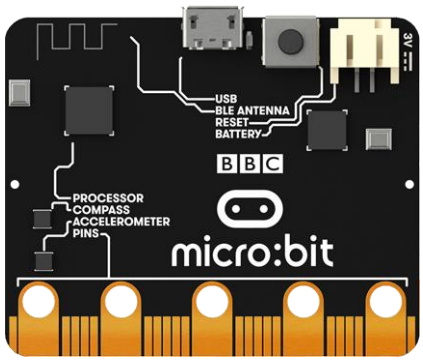
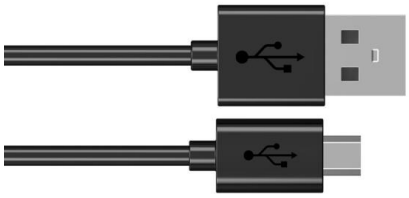
Chapter 4 Magnetometer

In this chapter, we will learn the micro:bit built-in magnetometer chip.

Project 4.1 Display Magnetometer Data

This project will print the data obtained from the magnetometer chip on the serial console.

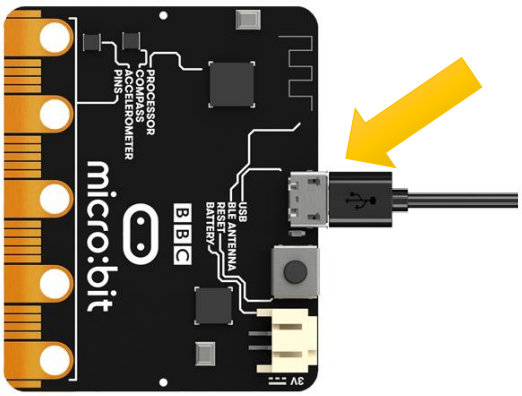
Component list

Microbit x1 	USB cable x1 

Circuit

Connect micro:bit and PC via micro USB cable.

Hardware connection



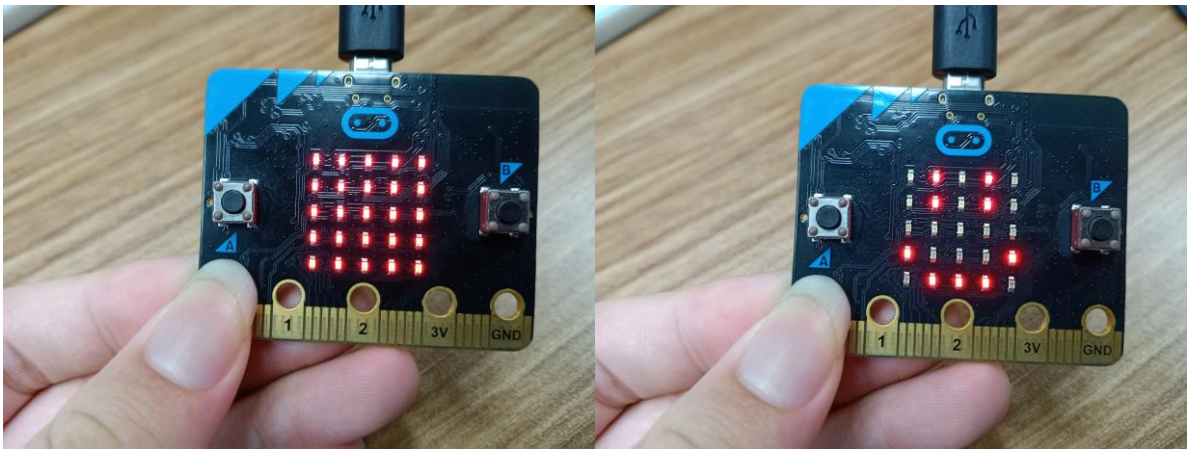
Block code

Open MakeCode first. Import the .hex file. The path is as below:
(How to import project)

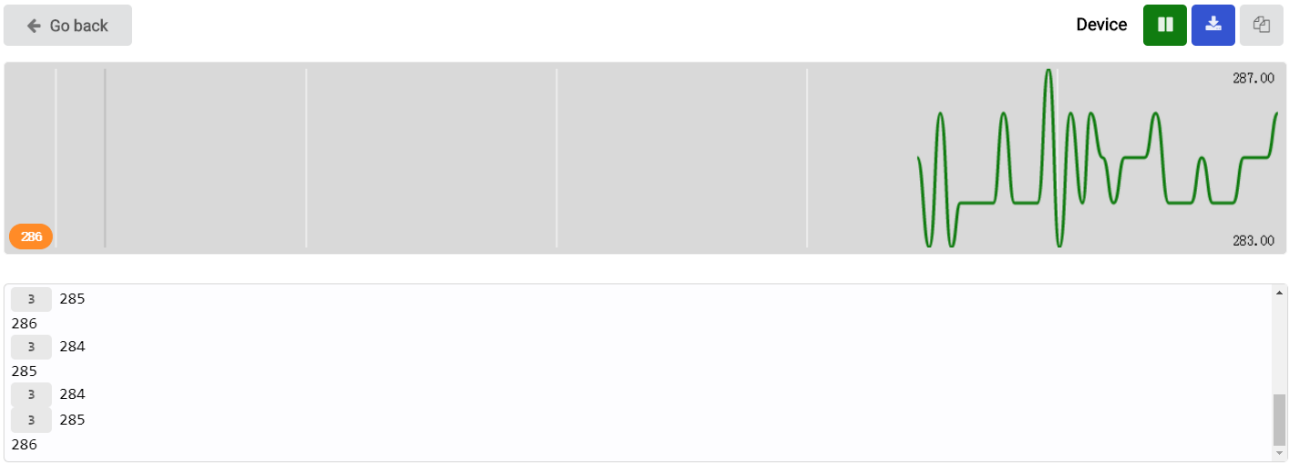
File type	Path	File name
HEX file	../Projects/BlockCode/04.1_DisplayMagnetometerData	DisplayMagnetometerData.hex

After importing successfully, the code is shown as below:

Check the connection of the circuit and verify it correct, and then download the code into the micro:bit. After completing downloading, the magnetometer needs to be calibrated (calibration must be performed using the magnetometer program). Calibrating the magnetometer will cause the program to pause until the calibration is completed. Start the calibration process, a prompt will scroll on the LED matrix, which indicates that you need to rotate the micro:bit until **all** LEDs on the LED screen are illuminated, and then a smile is displayed which means the calibration is completed, as shown below:

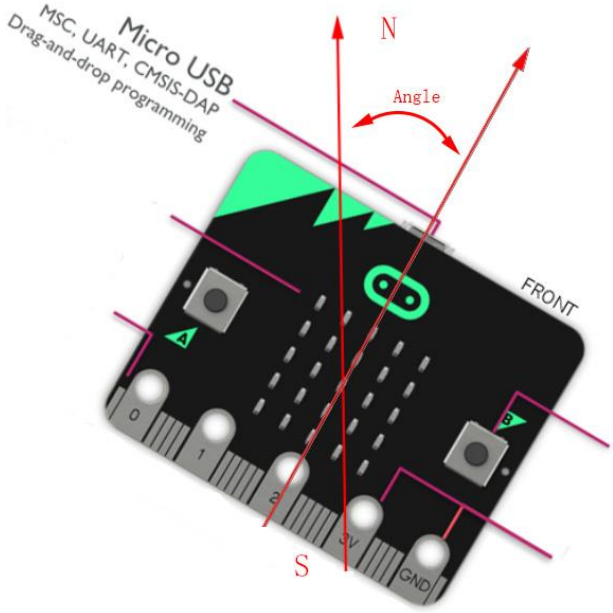


Then open the serial console (Open the Serial Port), place the micro:bit horizontally on the desktop, and rotate the micro:bit (clockwise or counterclockwise) to see the angular offset read from the magnetometer chip. As shown below:



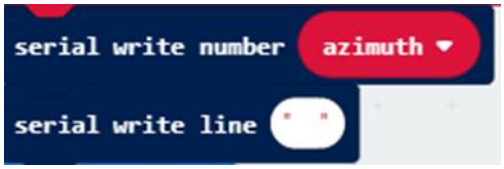
3 indicates the number of times of consecutive readings of the same value.

The angular offset is the angle between the directions of the micro:bit and the geographic North Pole, as shown in the following figure.

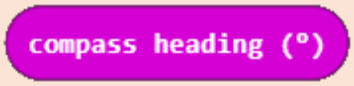


The angular offset read from the magnetometer chip is stored in the variable azimuth.

Then the value of the variable azimuth is printed on the serial port interface.



Reference

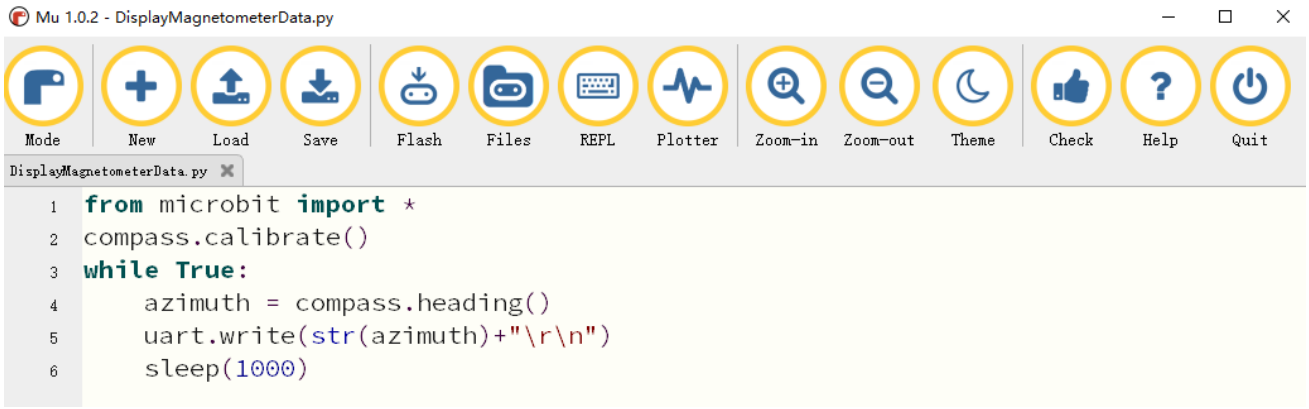
Block	Function
	Write a string to the serial port and start a new line of text by writing \r\n.
	Write a number to the serial port.
	The micro:bit measures the compass heading from 0 to 359 degrees with its magnetometer chip. Different numbers mean north, east, south, and west.

Python code

Open the .py file with Mu. Code, the path is as below:

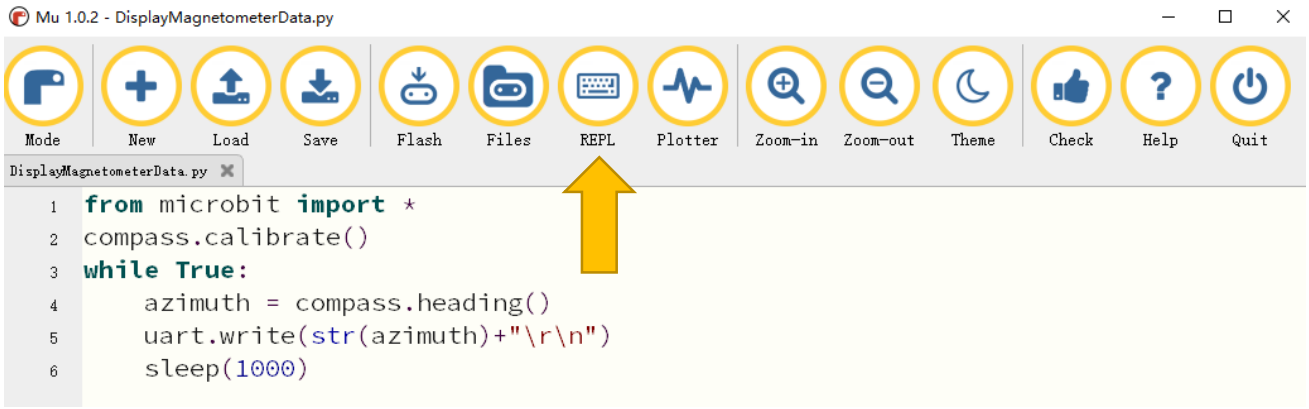
File type	Path	File name
Python file	../Projects/PythonCode/04.1_DisplayMagnetometerData	DisplayMagnetometerData.py

After loading successfully, the code is shown as below:



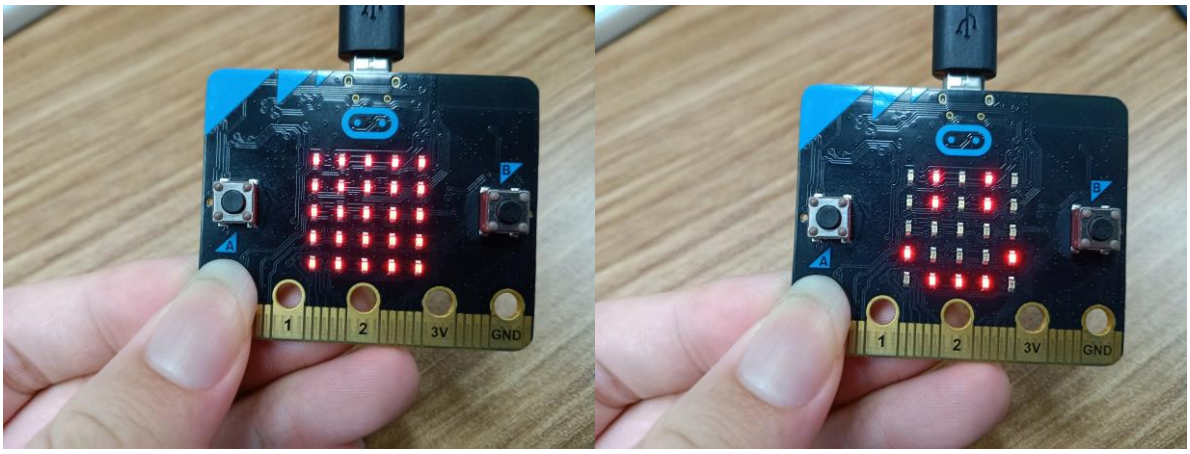
After checking the connection of the circuit and verify it correct, download the code into micro:bit.

After downloading the program, click REPL, as shown below.



Then press the reset button of the Micro:bit, you need to calibrate the magnetometer (the calibration must be performed when downloading using the magnetometer program).

Calibrating the magnetometer will cause the program to pause until the calibration is complete. Start the calibration process, a prompt will scroll on the LED matrix, which indicates that you need to rotate the micro:bit until **all** LEDs on the LED screen are illuminated, and then a smile is displayed which means the calibration is completed, as shown below:



Place the micro:bit horizontally on the desktop, and rotate the micro:bit (clockwise or counterclockwise) to see the angular offset read from the magnetometer chip. As shown below:

Mu 1.0.2 - compass.py

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

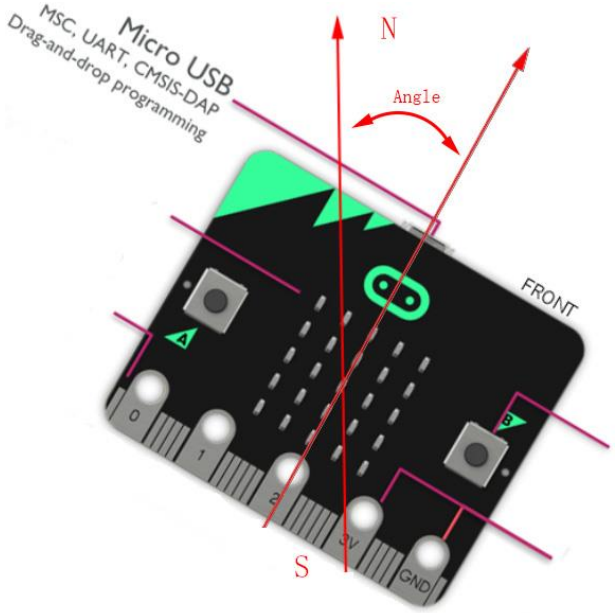
untitled compass.py

```
1 from microbit import *
2 compass.calibrate()
3 while True:
4     azimuth = compass.heading()
5     uart.write(str(azimuth)+"\r\n")
6     sleep(1000)
```

BBC micro:bit REPL

```
304
304
304
302
304
304
303
```

The angular offset is the angle between the direction of the micro:bit and the geographic north pole, as shown in the following figure.



The following is the program code:

```
1  from microbit import *
2  compass.calibrate()
3  while True:
4      azimuth = compass.heading()
5      uart.write(str(azimuth)+"\r\n")
6      sleep(1000)
```

Magnetometer calibration.

```
compass.calibrate()
```

The angular offset read from the magnetometer chip is stored in the variable azimuth and then printed out every 1s through a serial port.

```
azimuth = compass.heading()
uart.write(str(azimuth)+"\r\n")
sleep(1000)
```

Reference

compass.calibrate()

Starts the calibration process. An instructive message will scroll on the LED matrix, which indicates that you need to rotate the micro:bit until all LEDs are illuminated.

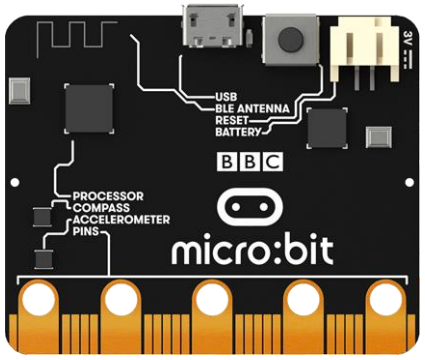
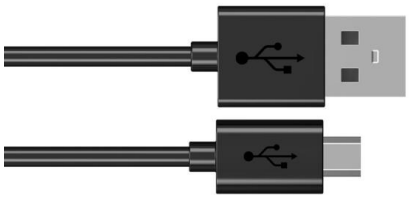
compass.heading()

Gives the compass heading, calculated from the above readings, as an integer in the range from 0 to 360, representing the angle in degrees, clockwise, with north as 0.

Project 4.2 Electronic Compass

In this project, we will use micro:bit to make an electronic compass, displaying an arrow on the micro:bit, and the arrow always points to the geographic north pole.

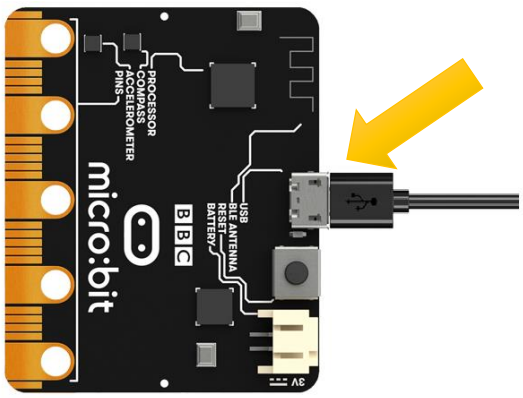
Component list

Microbit x1 	USB cable x1 

Circuit

Connect micro:bit and PC via micro USB cable.

Hardware connection



Block code

Open MakeCode first. Import the .hex file. The path is as below:
(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/04.2_ElectronicCompass	ElectronicCompass.hex

After importing successfully, the code is shown as below:

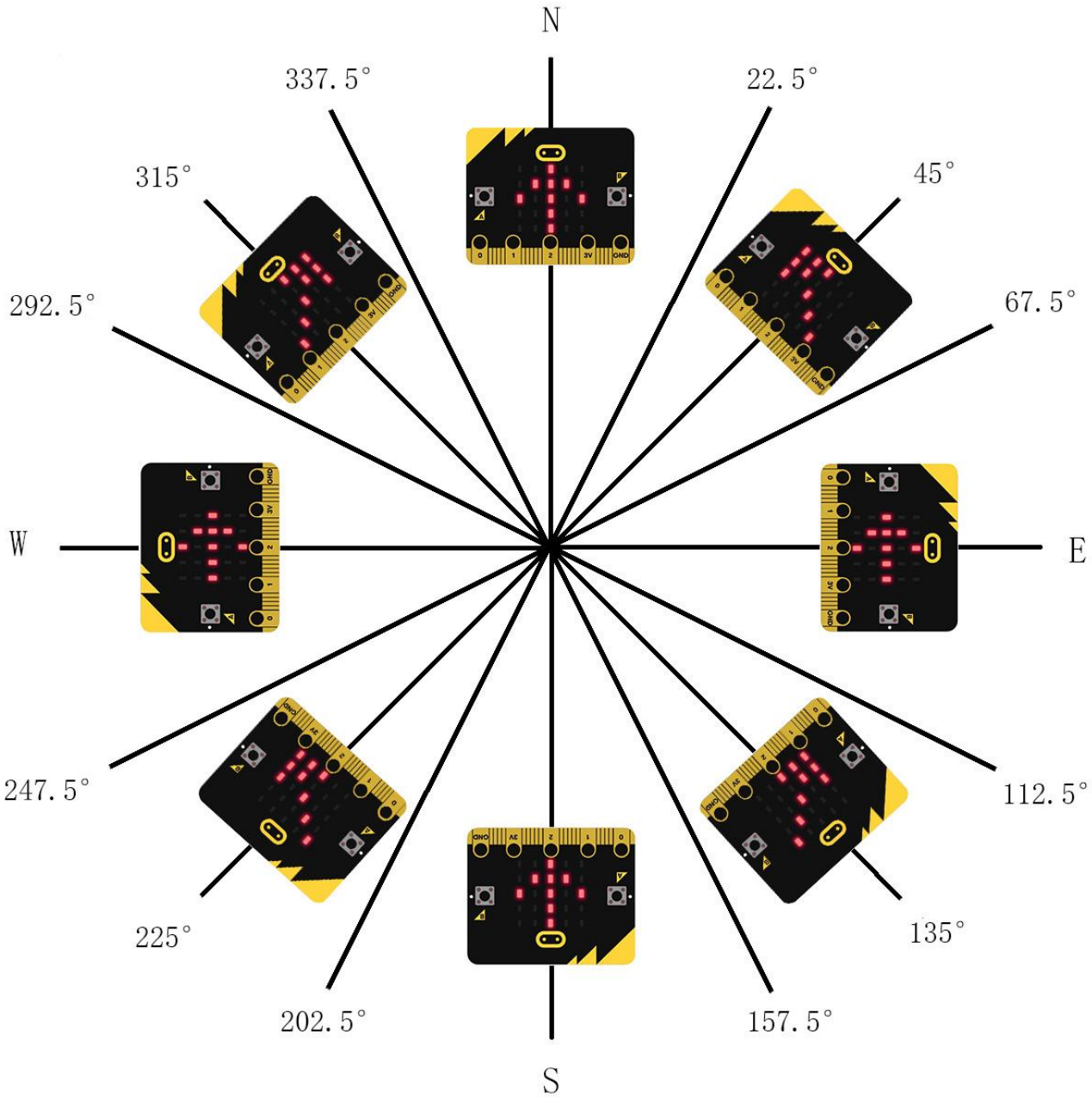
Check the connection of the circuit and verify it correct, and then download the code into the micro:bit. Calibrate the electronic compass. After the calibration is successful, place the micro:bit horizontally and turn the micro:bit to see that the arrow points to the geography Arctic.

The arrow will point to eight directions: northwest, west, southwest, south, southeast, east, northeast, north, each direction is 45 degrees apart. Assuming that the direction of the micro:bit is rotated 45 degrees from the north to the northeast of the geography, the arrow shown should be reversed, that is, it rotates -45 degrees, pointing to the northwest of the micro:bit, which is the geographic north pole. Therefore, we can adjust the direction of the arrow according to its angular offset from the geographic North Pole.

When the variable azimuth is less than 22.5 or greater than 337.5, the arrow points to the due north of the micro:bit.

When the variable azimuth is greater than 22.5 or less than 67.5, the arrow points to the northwest of the micro:bit.

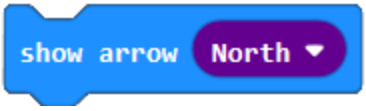
And so on in the same fashion, in every 45 degrees, the arrow points to a particular direction indicating the geographic north, as shown in the following illustration:



The angular offset read from the magnetometer chip is stored in the variable azimuth

Determine the value of the variable azimuth to change the direction of the arrow.

Reference

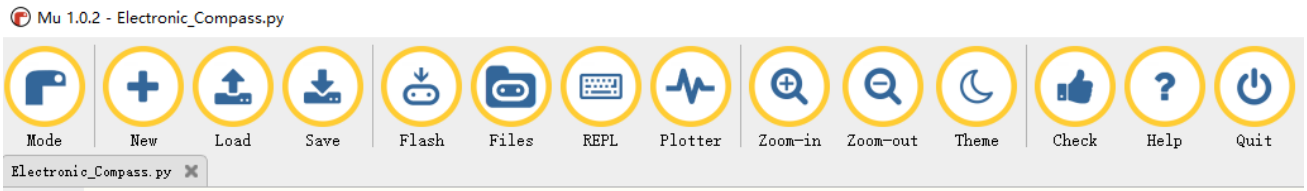
Block	Function
	Run code depending on whether a Boolean condition is true or false.
	Shows the selected arrow on the LED screen

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/04.2_ElectronicCompass	ElectronicCompass.py

After loading successfully, the code is shown as below:




```
from microbit import *
compass.calibrate()
while True:
    azimuth = compass.heading()
    if azimuth<22.5 and azimuth<67.5:
        display.show(Image.ARROW_NW)
    elif azimuth<67.5 and azimuth<112.5:
        display.show(Image.ARROW_W)
    elif azimuth<112.5 and azimuth<157.5:
        display.show(Image.ARROW_SW)
    elif azimuth<157.5 and azimuth<202.5:
        display.show(Image.ARROW_S)
    elif azimuth<202.5 and azimuth<247.5:
        display.show(Image.ARROW_SE)
    elif azimuth<247.5 and azimuth<292.5:
        display.show(Image.ARROW_E)
    elif azimuth<292.5 and azimuth<337.5:
        display.show(Image.ARROW_NE)
    elif azimuth<22.5 or azimuth>337.5:
        display.show(Image.ARROW_N)
```

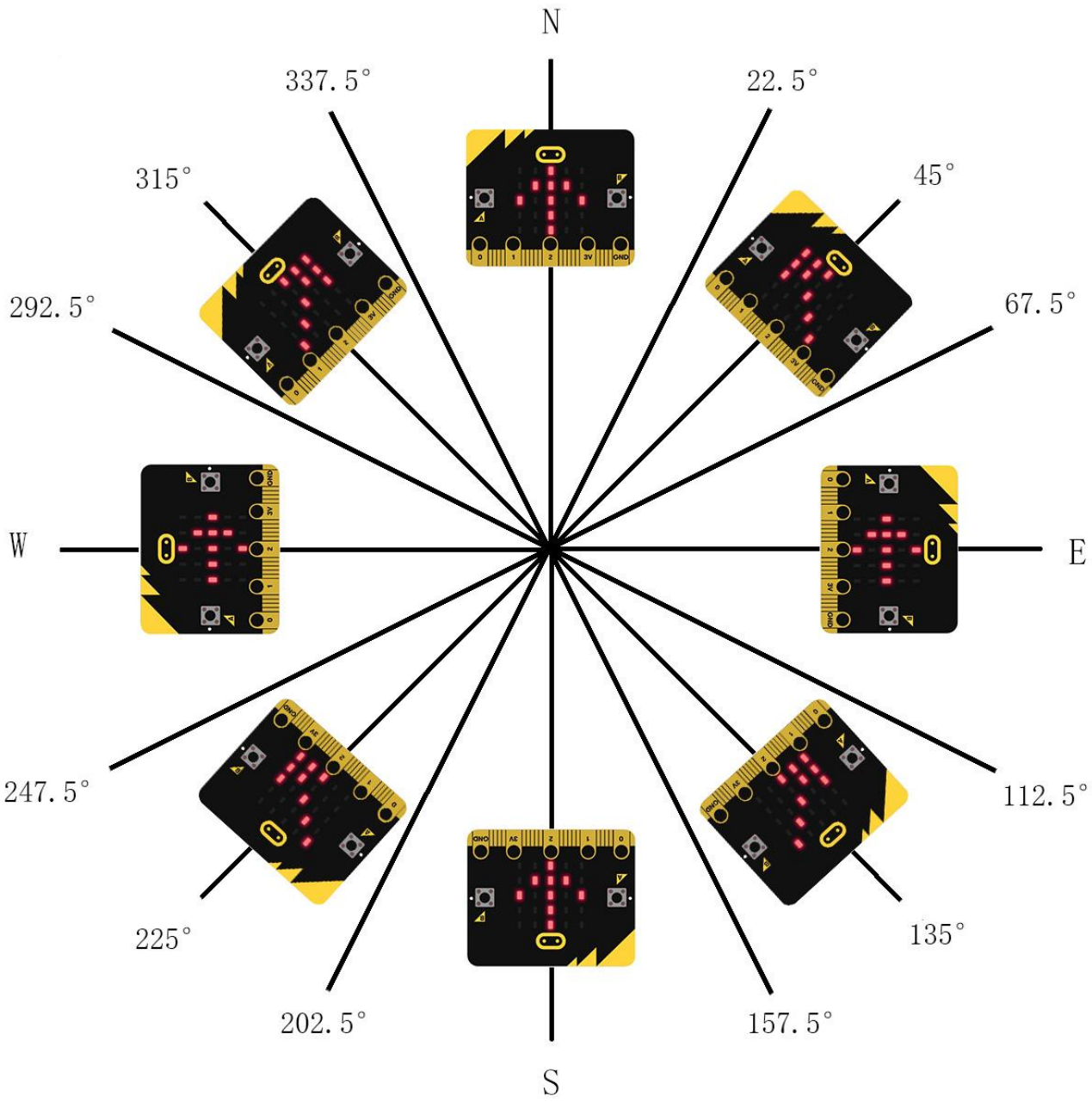
Check the connection of the circuit and verify it correct,, download the code into the micro:bit and calibrate the electronic compass. After the calibration is successful, place the micro:bit horizontally and rotate the micro:bit to see that the arrow points to the geography Arctic.

The arrow will point to eight directions:: northwest, west, southwest, south, southeast, east, northeast, north, each direction is 45 degrees apart. Assuming that the direction of the micro:bit is rotated 45 degrees from the north to the northeast of the geography, the arrow shown should be reversed, that is, rotated -45 degrees, pointing to the northwest of the micro:bit, which is the geographic north pole. Therefore, the direction of the arrow is adjusted according to the angular offset from the geographic North Pole.

When the variable azimuth is less than 22.5 or greater than 337.5, the arrow points to the true north of the micro:bit.

When the variable azimuth is greater than 22.5 and less than 67.5, the arrow points to the northwest of the micro:bit.

And so on in the same fashion, in every 45 degrees, the arrow points to a particular direction indicating the geographic north, as shown in the following figure:



The following is the program code:

```
from microbit import *
compass.calibrate()
while True:
    azimuth = compass.heading()
    if azimuth<22.5 and azimuth<67.5:
        display.show(Image.ARROW_NW)
    elif azimuth<67.5 and azimuth<112.5:
        display.show(Image.ARROW_W)
    elif azimuth<112.5 and azimuth<157.5:
        display.show(Image.ARROW_SW)
    elif azimuth<157.5 and azimuth<202.5:
        display.show(Image.ARROW_S)
    elif azimuth<202.5 and azimuth<247.5:
        display.show(Image.ARROW_SE)
    elif azimuth<247.5 and azimuth<292.5:
        display.show(Image.ARROW_E)
    elif azimuth<292.5 and azimuth<337.5:
        display.show(Image.ARROW_NE)
    elif azimuth<22.5 and azimuth>337.5:
        display.show(Image.ARROW_N)
```

Calibrate the electronic compass first and store the data on the variable azimuth.

```
compass.calibrate()
azimuth = compass.heading()
```

Determine the value of the variable azimuth and change the direction of the arrow.

```
if azimuth<22.5 and azimuth<67.5:
    display.show(Image.ARROW_NW)
elif azimuth<67.5 and azimuth<112.5:
    display.show(Image.ARROW_W)
elif azimuth<112.5 and azimuth<157.5:
    display.show(Image.ARROW_SW)
elif azimuth<157.5 and azimuth<202.5:
    display.show(Image.ARROW_S)
elif azimuth<202.5 and azimuth<247.5:
    display.show(Image.ARROW_SE)
elif azimuth<247.5 and azimuth<292.5:
    display.show(Image.ARROW_E)
elif azimuth<292.5 and azimuth<337.5:
    display.show(Image.ARROW_NE)
elif azimuth<22.5 and azimuth>337.5:
    display.show(Image.ARROW_N)
```

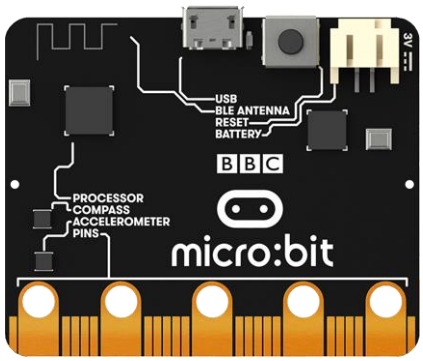
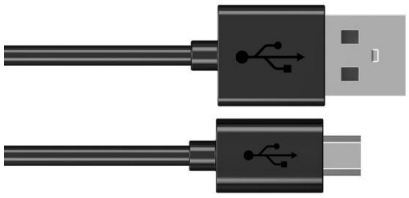
Chapter 5 Accelerometer

In this chapter, we will learn about the built-in accelerometer sensor of micro:bit.

Project 5.1 Display Accelerometer Data

In this project, we will obtain data from the accelerometer sensor and print it on the serial console.

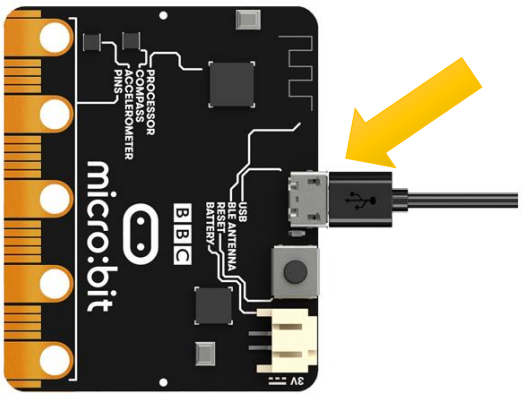
Component list

Microbit x1	USB cable x1
	

Circuit

Connect micro:bit and PC via a micro USB cable.

Hardware connection

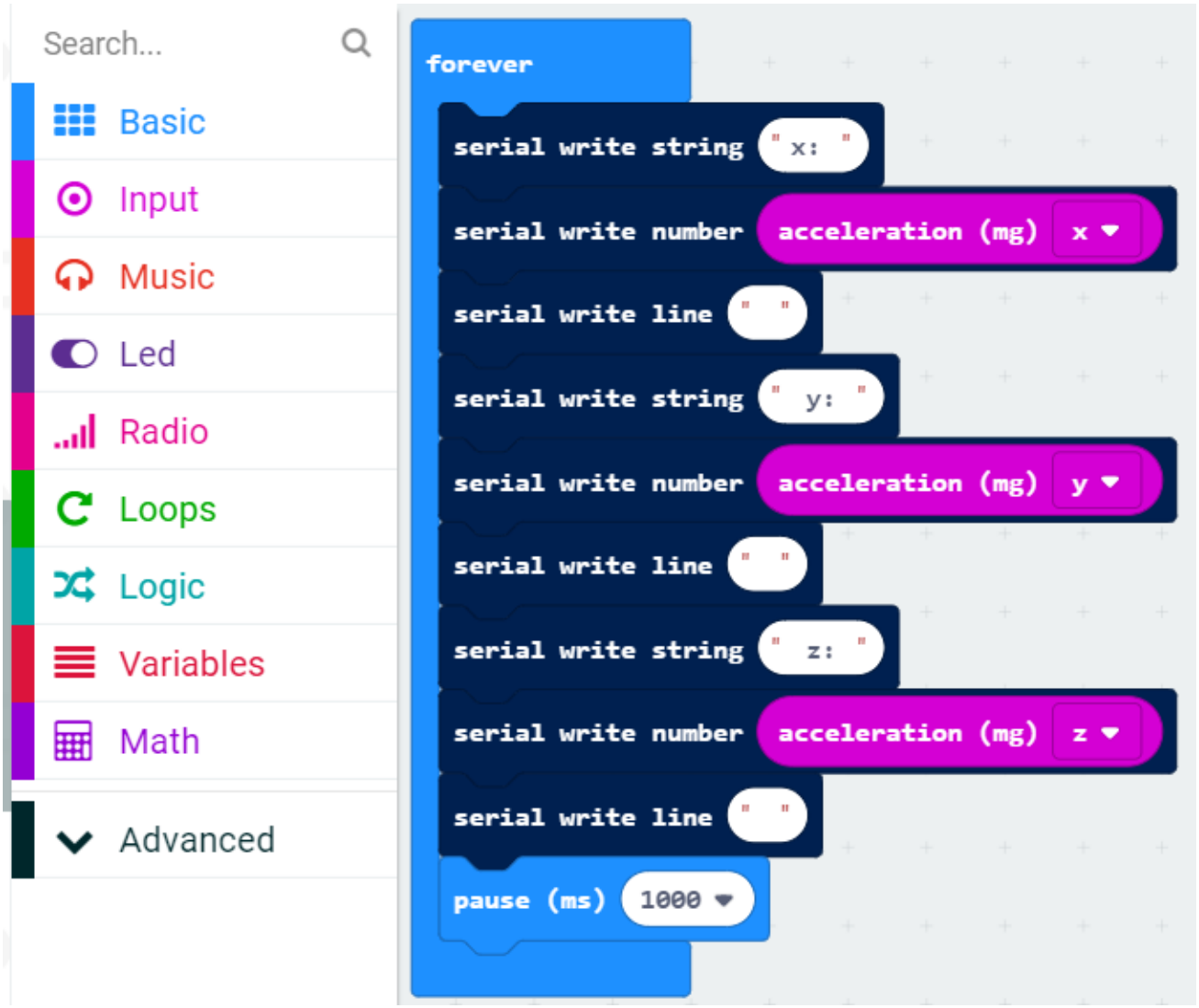


Block code

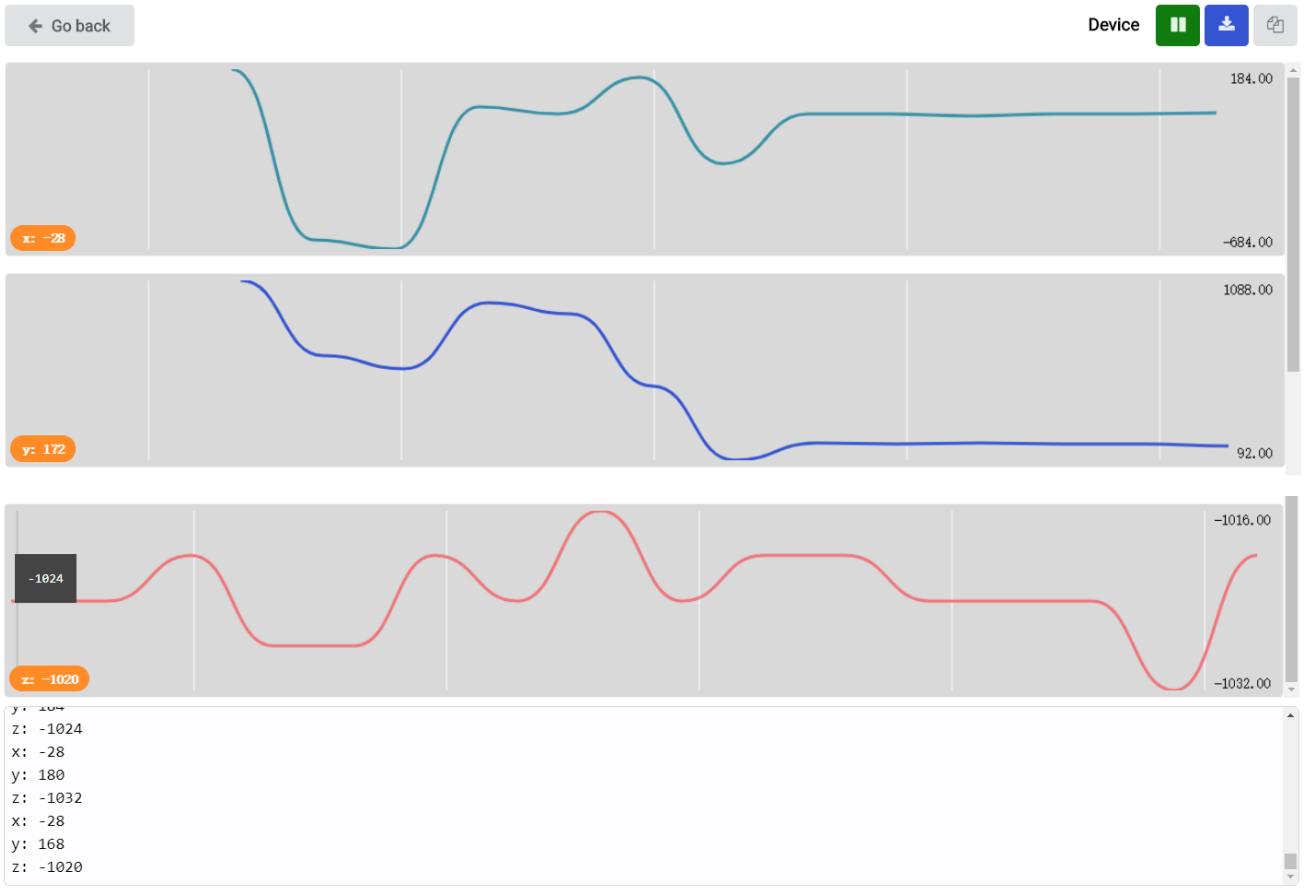
Open MakeCode first.
Import the .hex file. The path is as below:
(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/05.1_DisplayAccelerometerData	DisplayAccelerometerData.hex

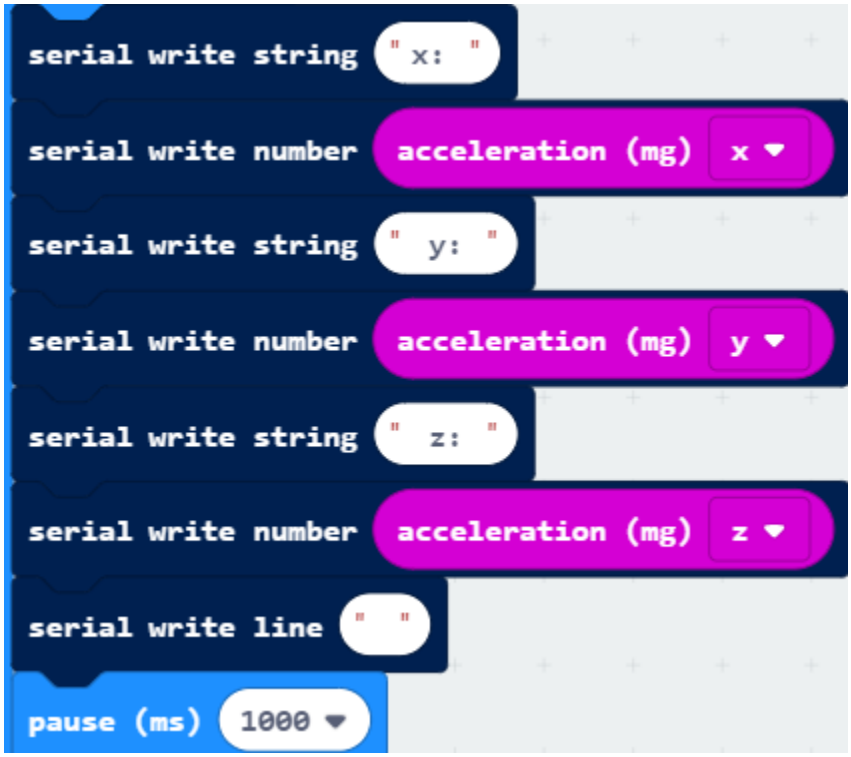
After importing successfully, the code is shown as below:



Check the connection of the circuit and verify it correct, download the code into the micro:bit, and then open the serial console, you can see the data of the accelerometer, as shown below:



Read the value of the accelerometer in three directions and print it out through the serial port every second.



Reference

Block	Function
acceleration (mg) x	Get the acceleration value in one of three dimensions, or the combined value in all directions (x, y, and z).

Python code

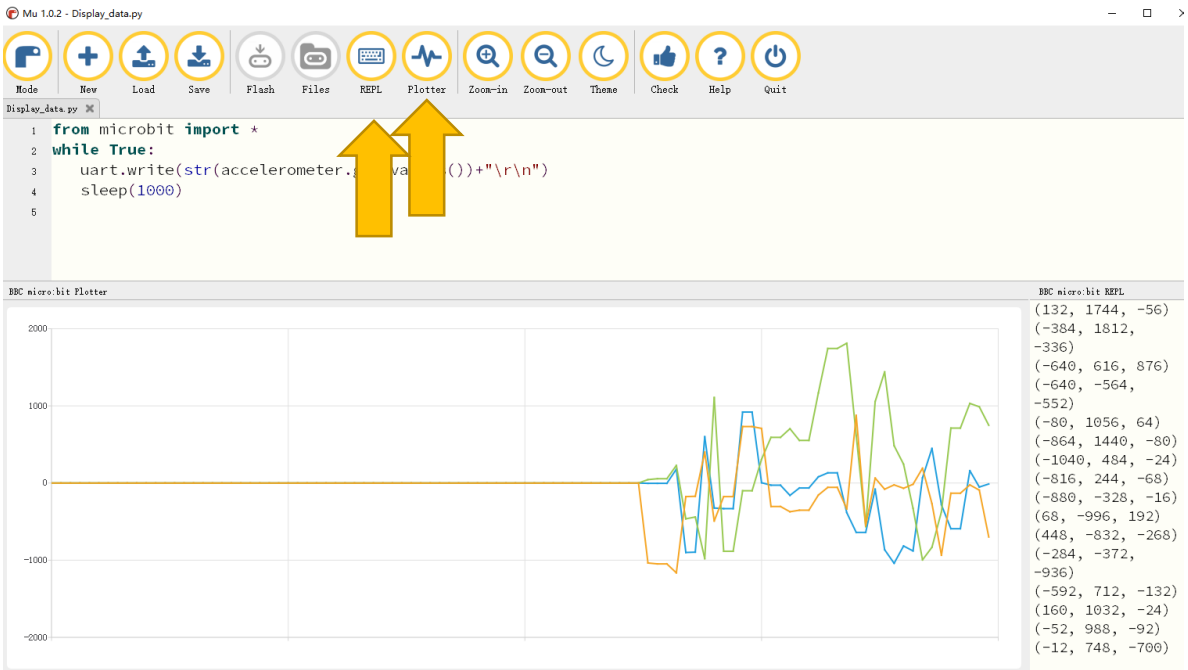
Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../PythonCode/05.1_DisplayAccelerometerData	DisplayAccelerometerData.py

After loading successfully, the code is shown as below:

Check the connection of the circuit and verify it correct, and then download the code into the micro:bit.

After the program is downloaded, open the plotter (Plotter), click on the REPL, you can see the x-axis, y-axis, z-axis data collected by the accelerometer, as shown below:



The following is the program code:

```
1  from microbit import *
2  while True:
3      uart.write(str(accelerometer.get_values())+"\r\n")
4      sleep(1000)
```

Every 1 second, the accelerometer data will be obtained and printed through the serial port.

```
1  uart.write(str(accelerometer.get_values())+"\r\n")
2  sleep(1000)
```

Reference

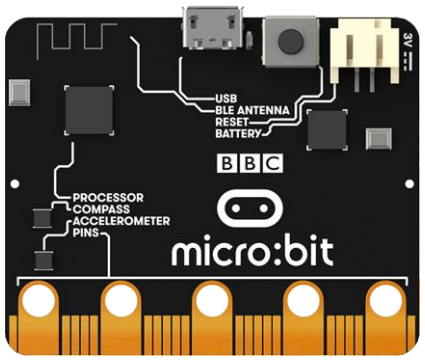
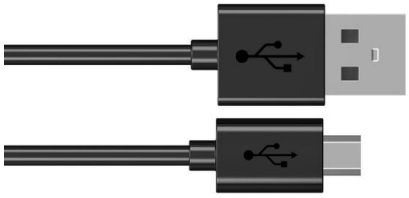
accelerometer.get_values()

Get the acceleration measurements in all axes at once, as a three-element tuple of integers ordered as X, Y, Z. By default the accelerometer is configured with a range of +/- 2g, so X, Y, and Z will be within the range of +/-2000mg.

Project 5.2 Gradiometer

In this project, we will use the accelerometer to make a level instrument.

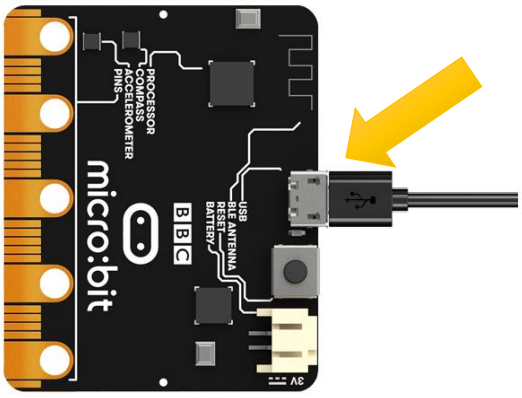
Component list

Microbit x1	USB cable x1
	

Circuit

Connect micro:bit and PC via a micro USB cable.

Hardware connection

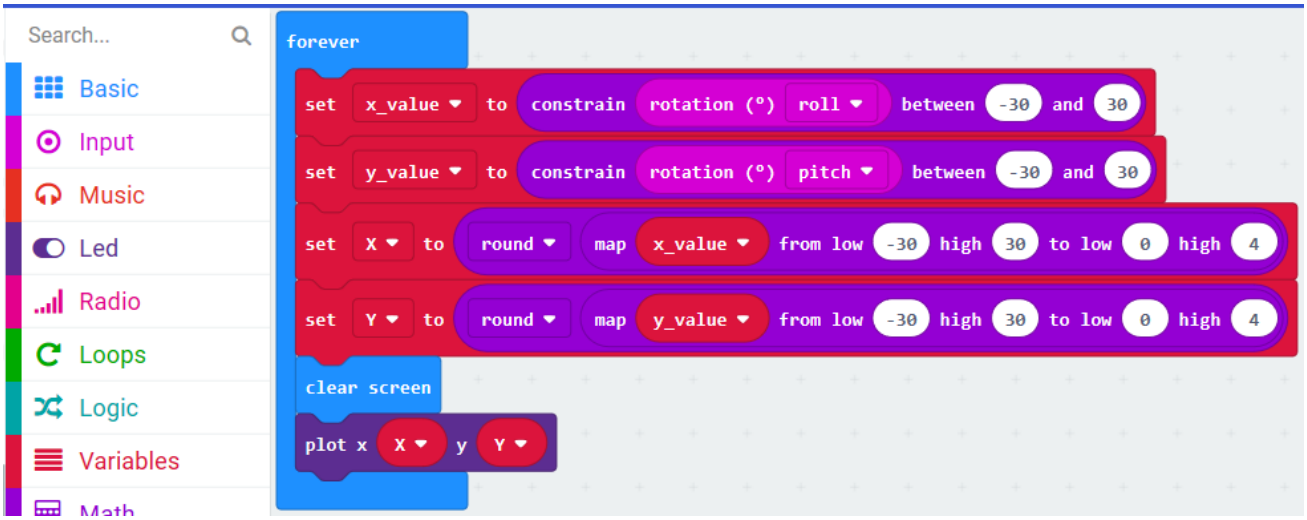


Block code

Open MakeCode first.
Import the .hex file. The path is as below:
(How to import project)

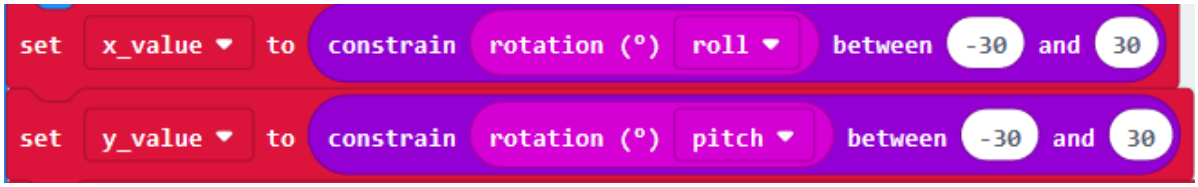
File type	Path	File name
HEX file	../Projects/BlockCode/05.2_Gradienter	Gradienter.hex

After importing successfully, the code is shown as below:



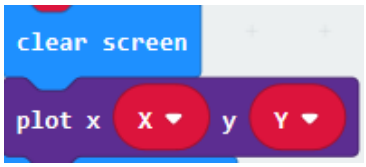
Check the connection of the circuit and verify it correct and download the code into micro:bit, you will observe that the LED dot matrix will change with the tilt of micro:bit.

Detect the flip angle of the microbit in the x-axis and the y-axis. The return value ranges from -180 to 180 degrees. This project does not require such a wide range of flip angles, so we just set it within -30 to 30 degrees.

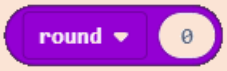
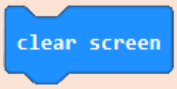


Since the LED screen is 5x5, map the range of -30-30 to the range of 0-4, and assign it to the X, Y variable.

Turn OFF all the LED first, then turn ON the corresponding LED according to the value of the X, Y variables.



Reference

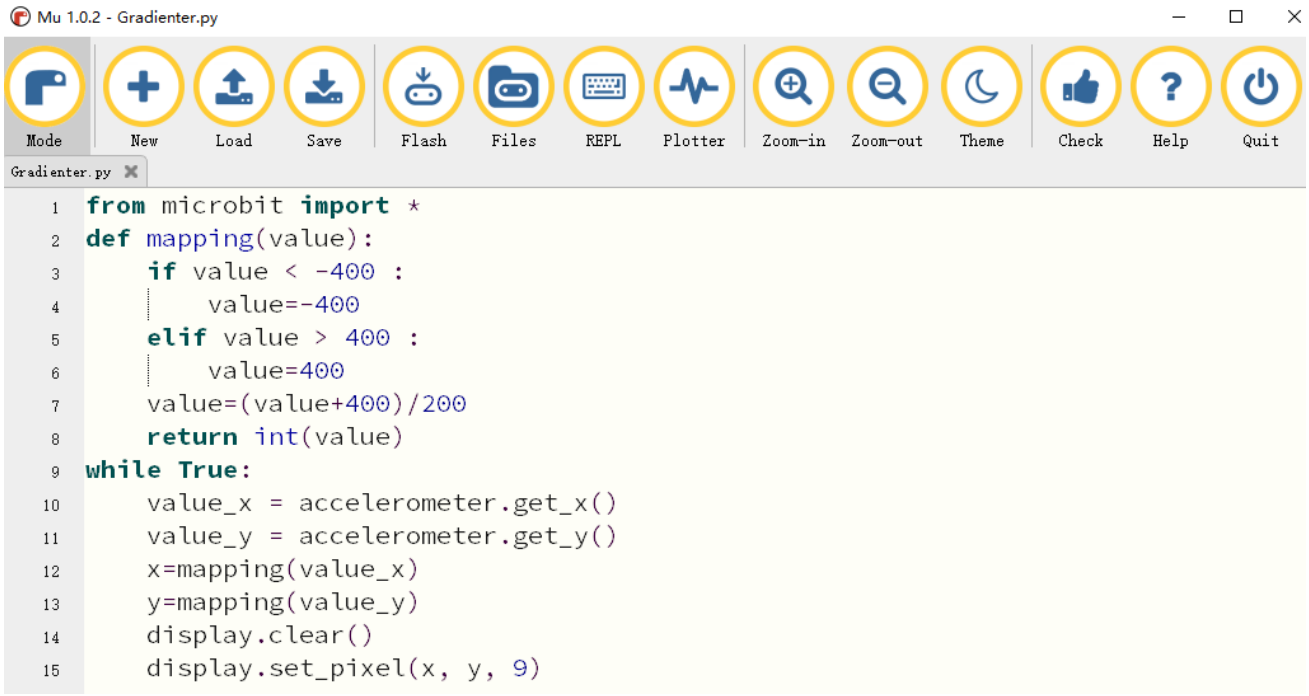
Block	Function
	Find how much the micro:bit is tilted in different directions.
	Turn ON the LED you set on the LED screen.
	If a number has a fractional part, you can change the number to the nearest integer value.
	A map is a conversion of one span of numbers to another.
	Turn OFF all the LED lights on the LED screen.
	Make sure that the value of the number you give is within the range.

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/05.2_Gradienter	Gradienter.py

After loading successfully, the code is shown as below:



Check the connection of the circuit and verify it correct, download the code into micro:bit, you will observe that the LED dot matrix will change with the tilt of micro:bit.

The following is the program code:

```
1  from microbit import *
2  def mapping(value):
3      if value < -400 :
4          value=-400
5      elif value > 400 :
6          value=400
7      value=(value+400)/200
8      return int(value)
9  while True:
10     value_x = accelerometer.get_x()
11     value_y = accelerometer.get_y()
12     x=mapping(value_x)
13     y=mapping(value_y)
14     display.clear()
15     display.set_pixel(x, y, 9)
```

A custom mapping() function limits the input value to a range of -400 to 400 and maps to a range of 0-4.

```
def mapping(value):
    if value < -400 :
        value=-400
    elif value > 400 :
        value=400
    value=(value+400)/200
    return int(value)
```

Read the value of the accelerometer X, Y-axis direction. The return value range is -2000-2000. This project does not require such a wide range, So we set it to the range of -400to 400. Call the mapping() function to return the value ranging from 0-4 , lighting the LED corresponding to the x row and the y column.

```
while True:
    value_x = accelerometer.get_x()
    value_y = accelerometer.get_y()
    x=mapping(value_x)
    y=mapping(value_y)
    display.clear()
    display.set_pixel(x, y, 9)
```

Reference

`display.clear()`
Set the brightness of all LEDs to 0 (off).
`display.set_pixel(x,y,9)`
Set the brightness value of the LED at column x and row y, which has to be an integer between 0 and 9.
`accelerometer.get_x()`
Get the acceleration measurement in the x axis, as a positive or negative integer, depending on the direction. The measurement is given in milli-g. By default the accelerometer is configured with a range of +/- 2g, and so this method will return a value within the range of +/- 2000mg
`accelerometer.get_y()`
Get the acceleration measurement in the y axis, as a positive or negative integer, depending on the direction. The measurement is given in milli-g. By default the accelerometer is configured with a range of +/- 2g, and so this method will return a value within the range of +/- 2000mg.

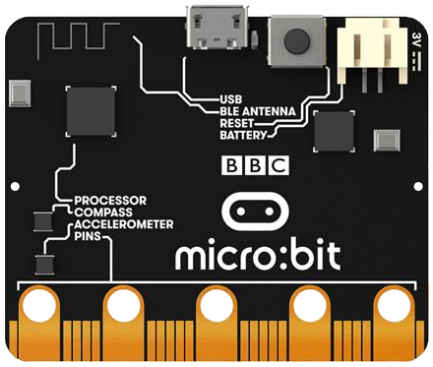
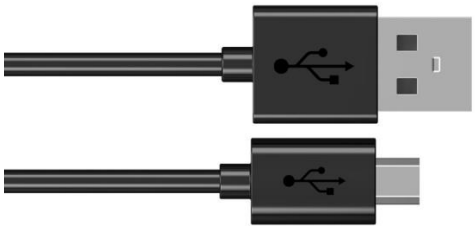
Chapter 6 Light Sensor

In this chapter, we will learn the micro:bit built-in light sensor and photoresistor.

Project 6.1 Built-in Light Sensor

In this project, we use the micro:bit built-in light sensor to measure the brightness of light.

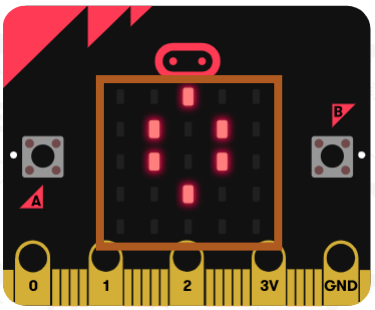
Component list

Microbit x1 	USB cable x1 

Component knowledge

Light sensor

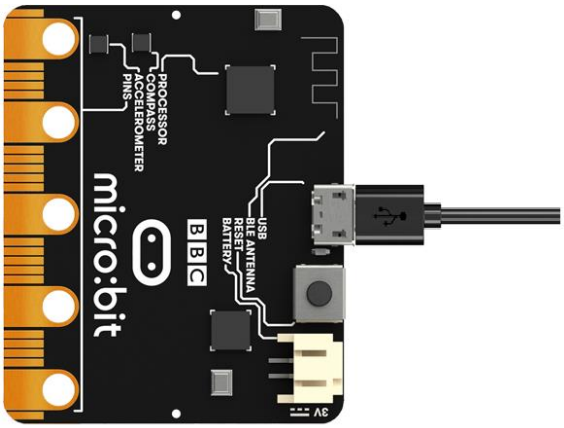
Micro:bit detects the ambient light intensity through the LED matrix. In forward bias mode, the LED screen works as a display. In reverse bias mode, the LED screen works as a basic light sensor that can be used to detect ambient light.



Circuit

Connect micro:bit and PC via a micro USB cable.

Hardware connection



Block code

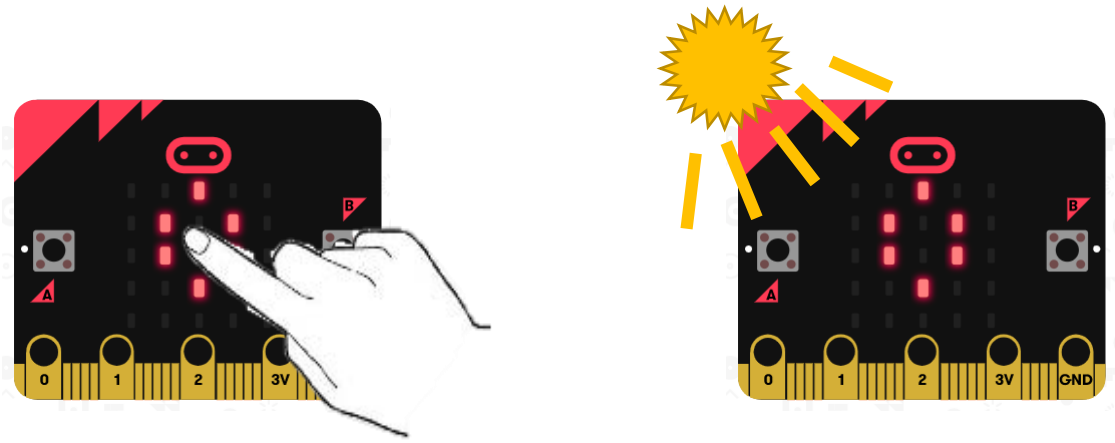
Open MakeCode first. Import the .hex file. The path is as below:

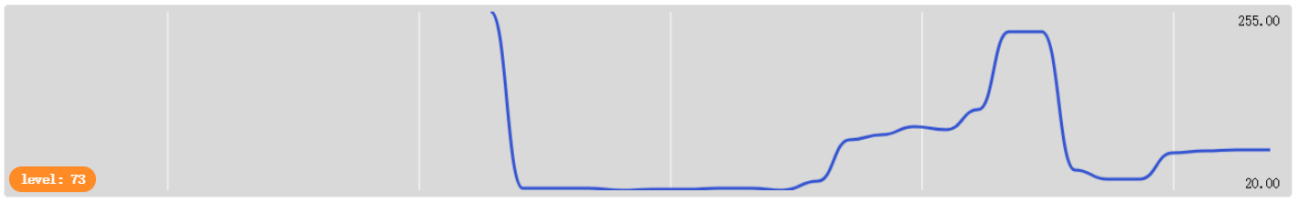
(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/06.1_LightIntensityMeter	LightIntensityMeter.hex

After importing successfully, the code is shown as below:

Check the connection of the circuit, verify it correct and download the code into the micro:bit. Open the serial port console, then you can see the reading light intensity. Cover the LED screen with your hand or increase the light shining on it, you can see the change in value, the range of values is 0-255, 0 is dark, 255 is the brightest, as shown below.






```
level:228
level:229
level:46
level:35
level:34
level:69
level:72
2 level:73
```

Reference

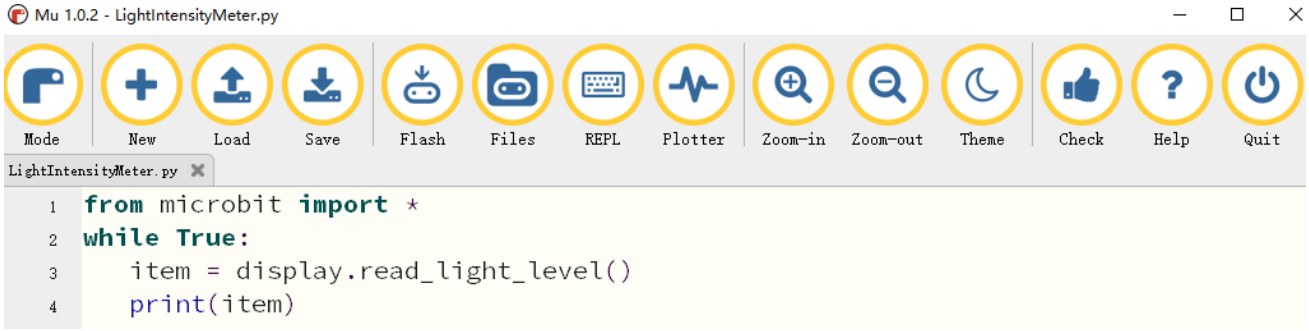
Block	Function
light level	Detect the light level (how bright or dark it is) of the environment where you are. The light level 0 means darkness and 255 means bright light.

Python code

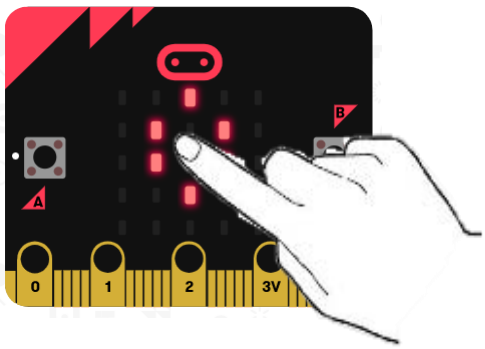
Open the .py file with Mu. Code, the path is as below:

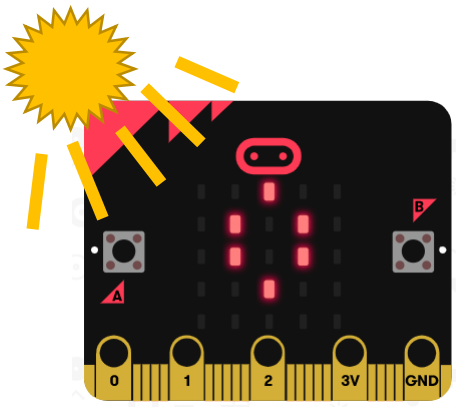
File type	Path	File name
Python file	../Projects/PythonCode/06.1_LightIntensityMeter	LightIntensityMeter.py

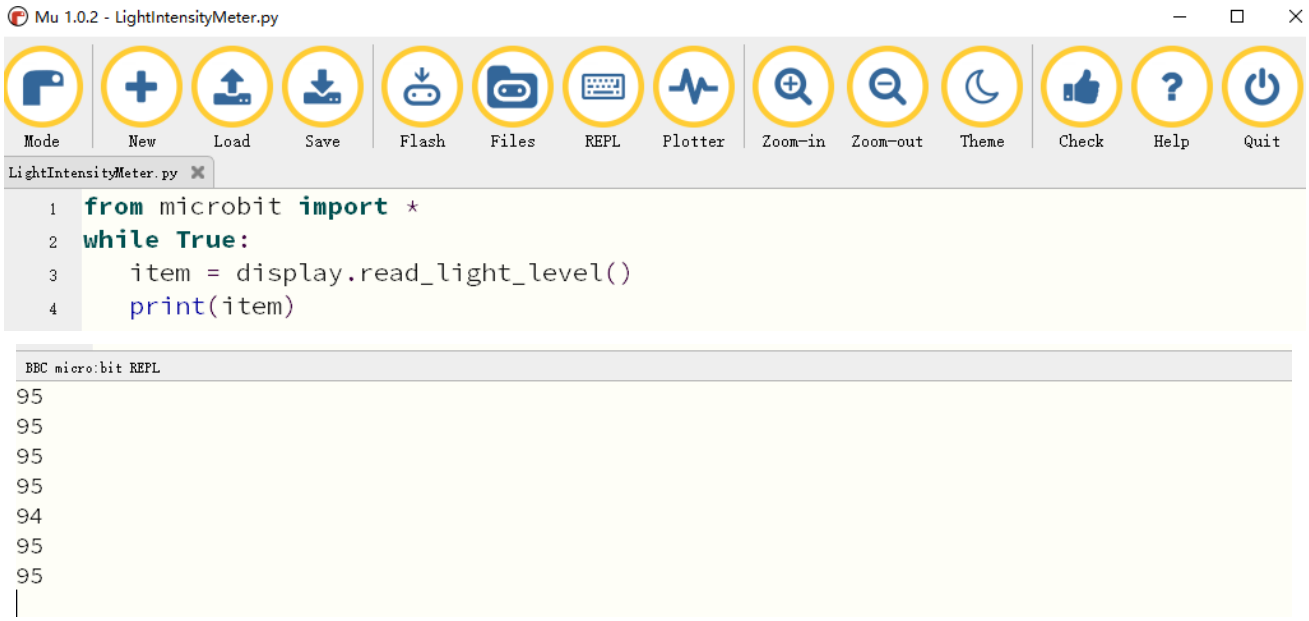
After loading successfully, the code is shown as below:



Check the connection of the circuit, verify it correct, download the code into the micro:bit. Open the serial port console, then you can see the reading light intensity. Cover the LED screen with your hand or increase the light shining on it, you can see the change in value, the range of values is 0-255, 0 is dark, 255 is the brightest, as shown below.







The following is the program code:

```
1 from microbit import *
2 while True:
3     item = display.read_light_level()
4     print (item)
```

Reference

display.read_light_level()

Use the display's LEDs in reverse-bias mode to sense the amount of light falling on the display, return an integer between 0 and 255 representing the light level. The larger the value, the brighter the light..

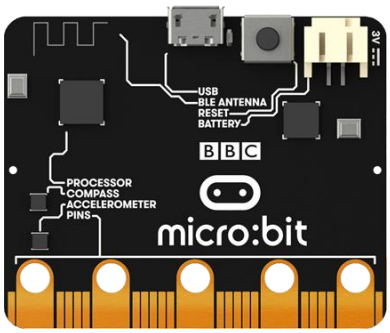
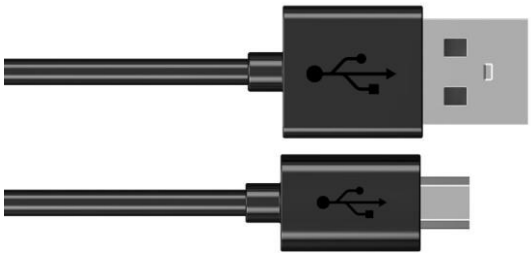
Chapter 7 Temperature Sensor

In this chapter, we will learn the micro:bit built-in temperature sensor and thermistor.

Project 7.1 Built-in Temperature Sensor

In this project, we measure the temperature with the micro:bit's built-in temperature sensor.

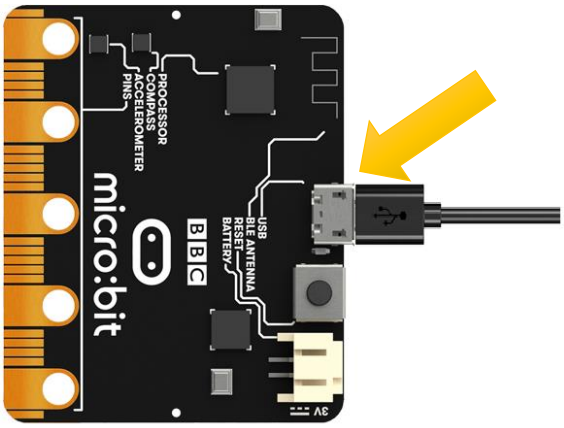
Component list

Microbit x1 	USB cable x1 

Circuit

Connect micro:bit and PC via micro the USB cable.

Hardware connection



Block code

Open MakeCode first. Import the .hex file. The path is as below:
(How to import project)

File type	Path	File name
HEX file	../Projects/BlockCode/07.1_BuiltInThermometer	BuiltInThermometer.hex

After importing successfully, the code is shown as below:

Check the connection of the circuit, verify it correct, and download the code into micro:bit, and then LED dot matrix screen will display the current detected temperature.

Reference

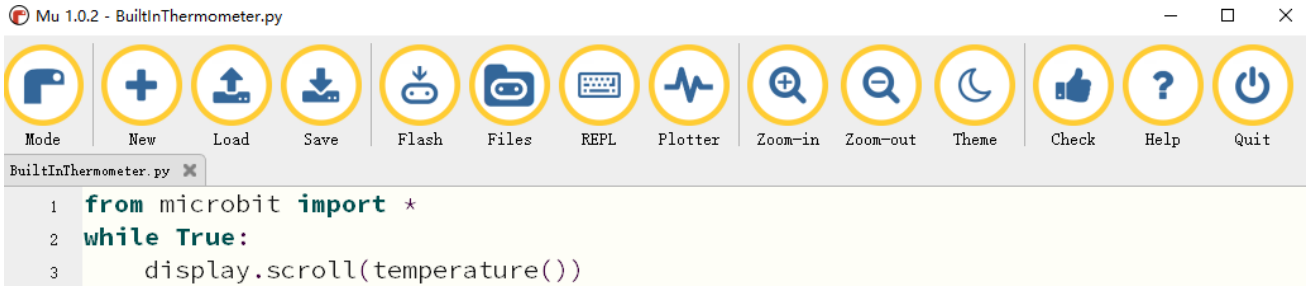
Block	Function
temperature (°C)	Detect the temperature of the environment where you are. The temperature is measured in Celsius (metric).

Python code

Open the .py file with Mu. Code, the path is as below:

File type	Path	File name
Python file	../Projects/PythonCode/07.1_BuiltInThermometer	BuiltInThermometer.py

After loading successfully, the code is shown as below:



Check the connection of the circuit, verify it correct, and download the code into micro:bit, and then LED dot matrix screen will display the current detected temperature.

The following is the program code:

```
1 from microbit import *
2 while True:
3     display.scroll(temperature())
```

Display the detected temperature on the LED dot matrix.

```
    display.scroll(temperature())
```

Reference

`temperature()`
Return the temperature of the micro:bit in Celcius.
`display.scroll()`
scrolls a string across the display

What's Next?

THANK YOU for participating in this learning experience!

We have reached the end of this Tutorial. If you find errors, omissions or you have suggestions and/or questions about the Tutorial or component contents of this Kit, please feel free to contact us: support@freenove.com
We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.
If you want to learn more about micro:bit, we have a smart Car named Micro:bit Rover. You can visit our website to purchase. http://www.freenove.com/store.html

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our website. We will continue to launch fun, cost-effective, innovative and exciting products.
http://www.freenove.com/

Thank you again for choosing Freenove products.