Read First

This tutorial is an advanced tutorial for Micro:Rover. If you haven't yet studied the basic tutorial (Tutorial.pdf), we strongly recommend you to learn it (Tutorial.pdf) first. The basic tutorial (Tutorial.pdf) uses graphical block code programming, which is easier to understand and to get started.

The code involved in this tutorial is written in Python, and each project implements the same function as the one in the basic tutorial (Tutorial.pdf).

Freenove is an open-source electronics platform.
www.freenove.com

Warning

When you purchase or use Freenove Micro Rover, please note the following:

- This product contains small parts. Swallowing or improper operation can cause serious infections and death. Seek immediate medical attention when the accident happened.
- Do not allow children under 3 years old to play with or near this product. Please place this product in where children under 3 years of age cannot reach.
- Do not allow children lack of ability of safe to use this product alone without parental care.
- Never use this product and its parts near any AC electrical outlet or other circuits to avoid the potential risk of electric shock.
- Never use this product near any liquid and fire.
- Keep conductive materials away from this product.
- Never store or use this product in any extreme environments such as extreme hot or cold, high humidity and etc.
- Remember to turn off circuits when not in use this product or when left.
- Do not touch any moving and rotating parts of this product while they are operating.
- Some parts of this product may become warm to touch when used in certain circuit designs. This is normal. Improper operation may cause excessively overheating.
- Using this product not in accordance with the specification may cause damage to the product.

About

Freenove is an open-source electronics platform. Freenove is committed to helping customer quickly realize the creative idea and product prototypes, making it easy to get started for enthusiasts of programing and electronics and launching innovative open source products. Our services include:

- Electronic components and modules
- Learning kits for Arduino
- Learning kits for Raspberry Pi
- Learning kits for micro:bit
- Learning kits for Technology
- Product customization service
- Robot kits
- Auxiliary tools for creations

Our code and circuit are open source. You can obtain the details and the latest information through visiting the following web sites:
http://www.freenove.com
https://github.com/freenove
Your comments and suggestions are warmly welcomed, and please send them to the following email address:
support@freenove.com
If you have any business matters, please feel free to contact us:
sale@freenove.com

References

This product is named Freenove Micro Rover.
You can download the sketches and references used in this product in the following websites:
http://www.freenove.com
https://github.com/Freenove/Freenove_Micro_Rover
If you have any difficulties, you can send email to technical support for help.

Support

Freenove provides free and quick technical 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

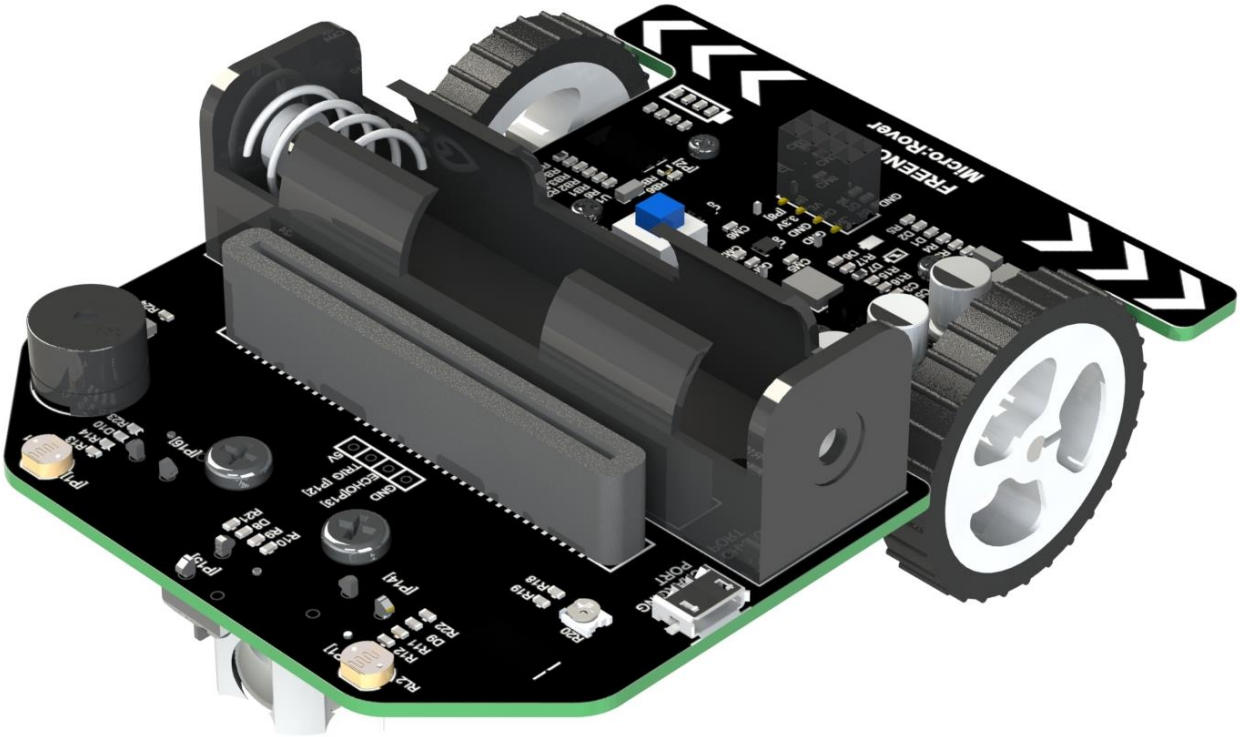
Please send email to:
support@freenove.com
On working day, we usually reply to you within 24 hours.

Copyright

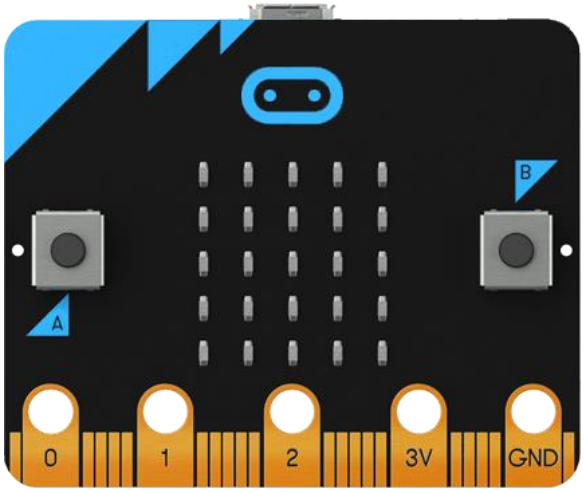
Contents

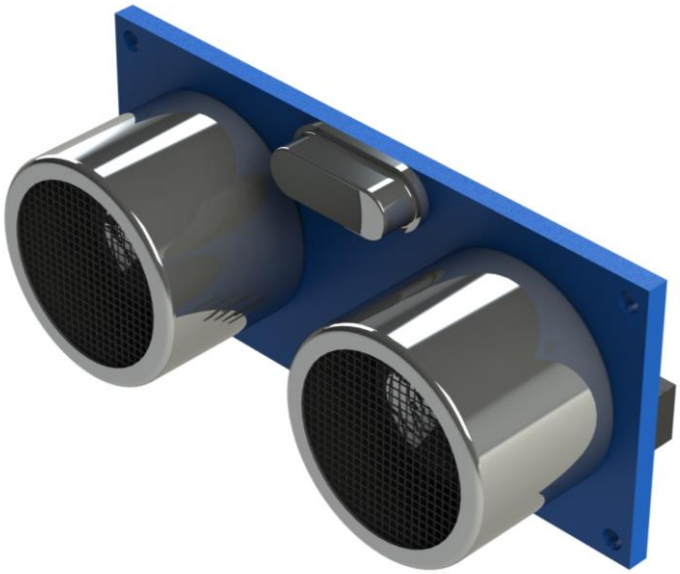
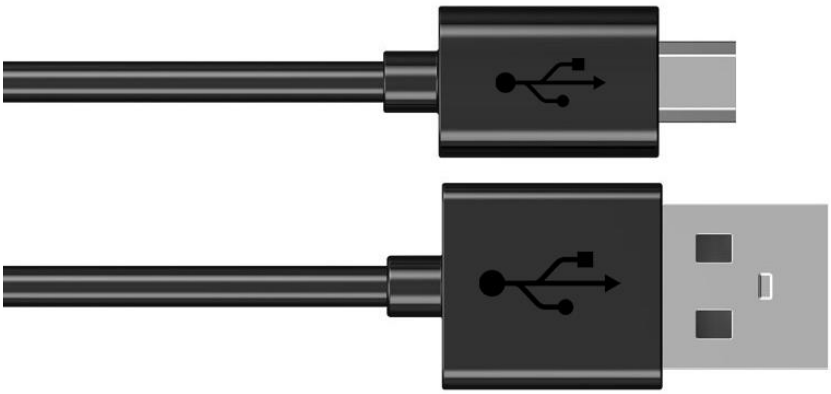
List

Micro:Rover



BBC micro:bit



Ultrasonic ranging module 
Micro USB cable 

Line-Tracking Map

Preface

Do you want to learn programming or get a robot programmed by yourself?
Currently, 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. It doesn't matter even though you haven't heard of them. With this product and the tutorial, you can easily complete a multi-functional programming car 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. So, no matter your computer system is Windows, Linux or Mac, they all work. 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, thus helping us realize our own creativity more efficiently by using these free resource. Of course, you can also make contributions to the resource.

And that's why Freenove Micro:Rover emerged. Rover is a programmable car developed by Freenove based on BBC micro:bit. Adopting integrated design, it does not require additional wiring, which makes it easy to use. There are ultrasonic, infrared and photosensitive sensors on it as well as buzzer, RGB LED and other peripherals. Rich hardware resources will help you master more knowledge and skills. You can use your imagination to create more ways to play the robot.

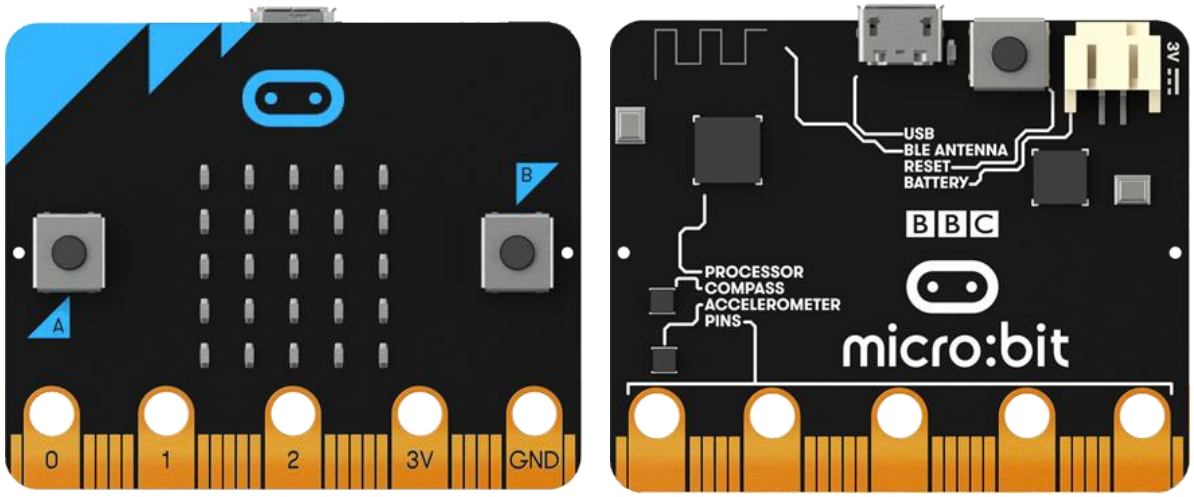
In each learning chapter of this tutorial, we provide program source code with detailed program explanations and burnable binaries, contributed to your understanding of the meaning of each section of program.

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

Welcome to the world of electronic and programming.
Our journey to build and explore Micro:bit and Micro:Rover electronic projects will start with this chapter.

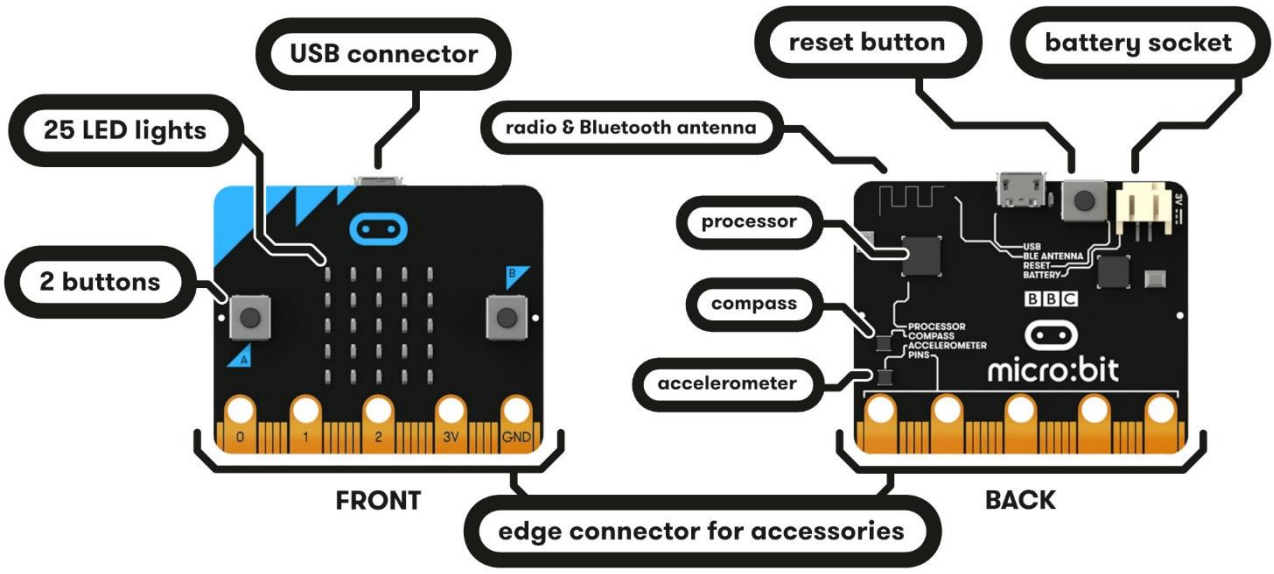
Meet micro:bit



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 individually-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 contents, please refer to:
https://microbit.org/guide/features/

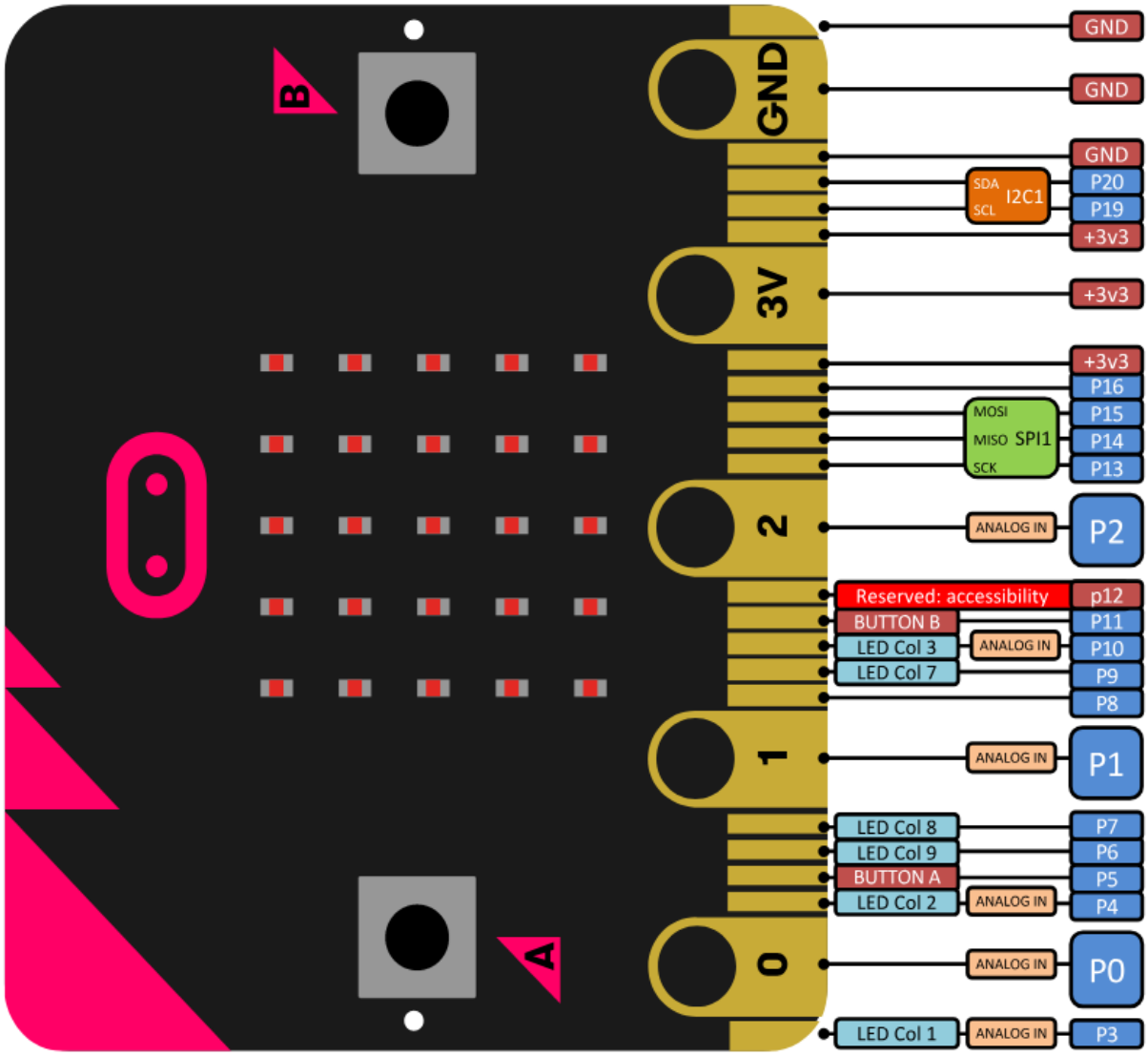
Hardware

It is not required that beginners 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/.

To complete building Rover, we need a brief understanding of GPIO.

GPIO

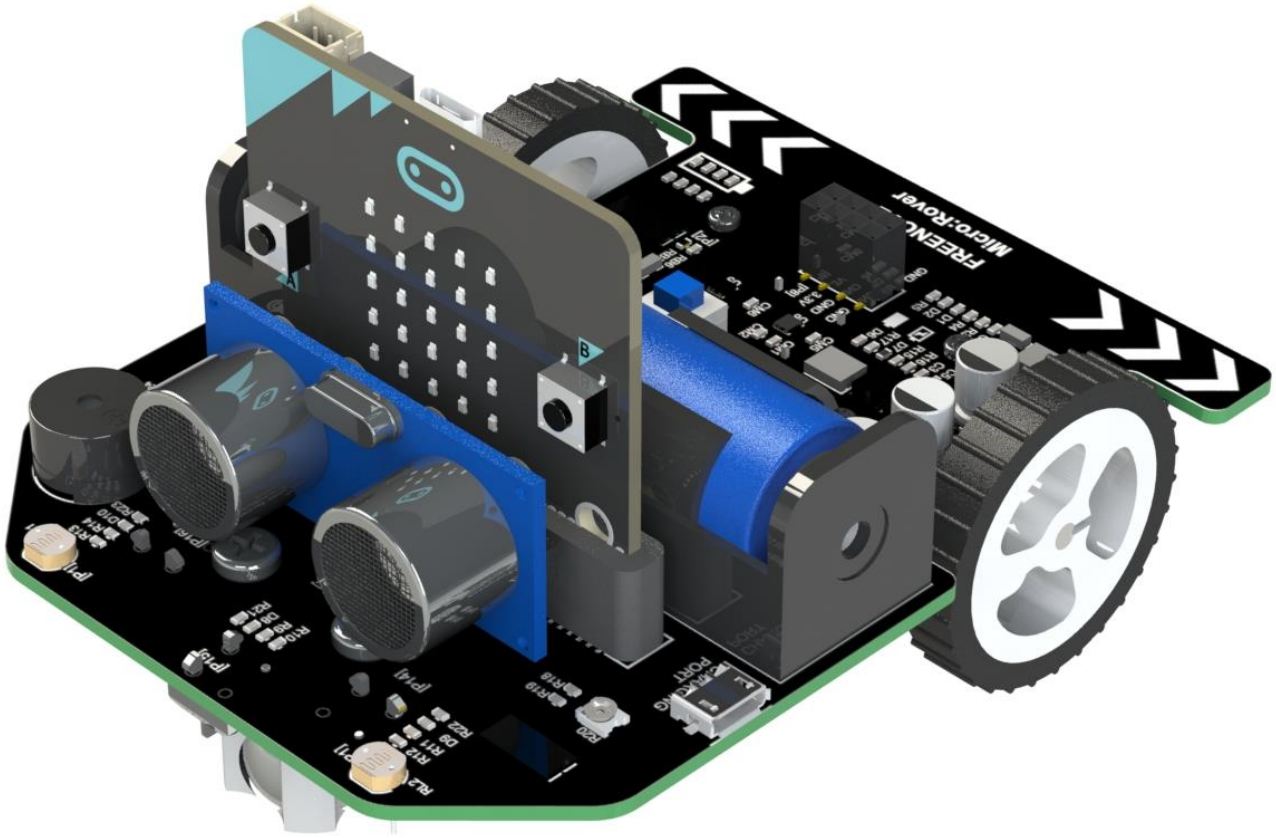
GPIO, namely General Purpose Input/output Pins, is a part of micro:bit. It is an important part for connecting external devices. All sensors and devices on Rover communicate with each other through micro:bit GPIO. The following is the GPIO serial number and function diagram of micro:bit:



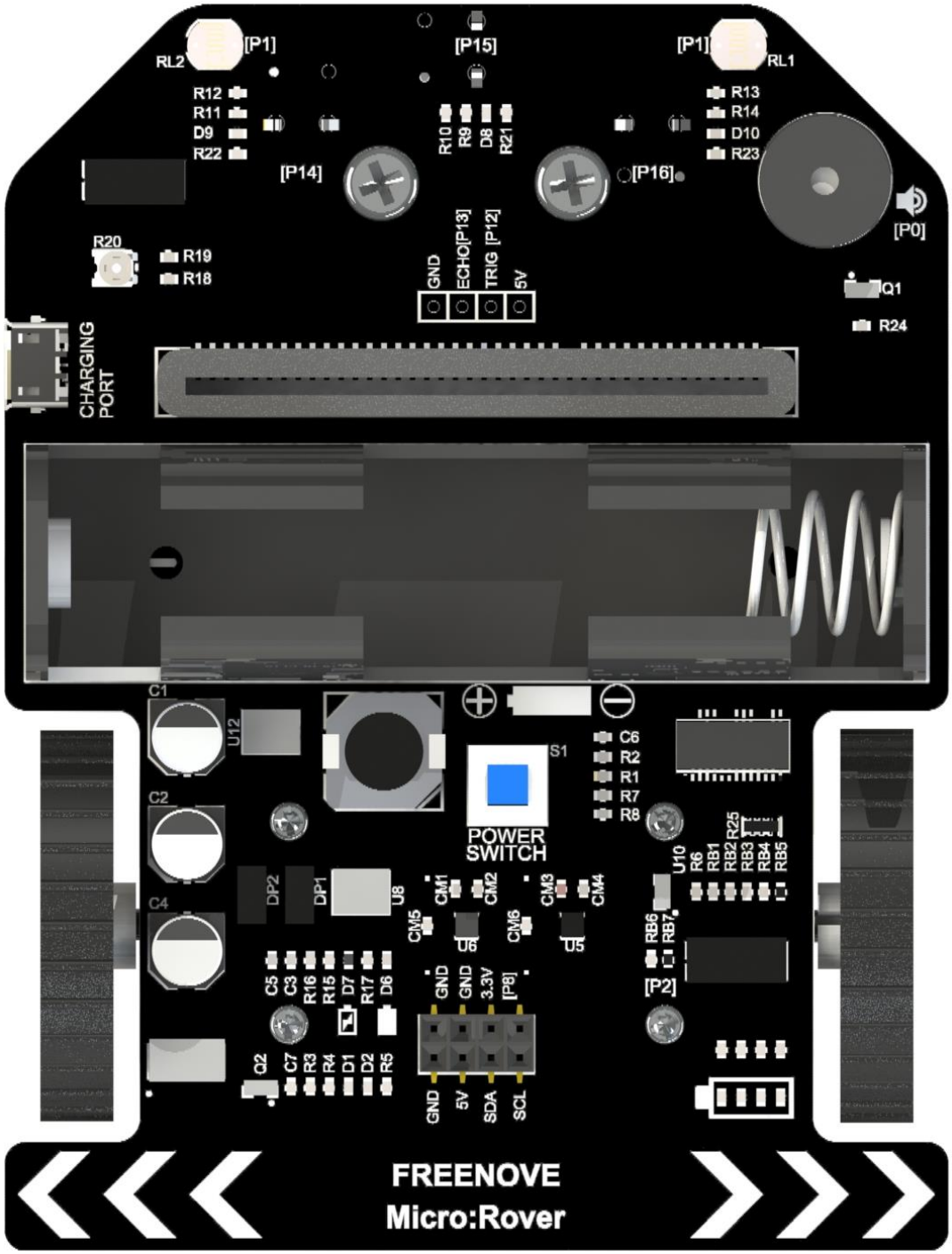
Micro:Rover

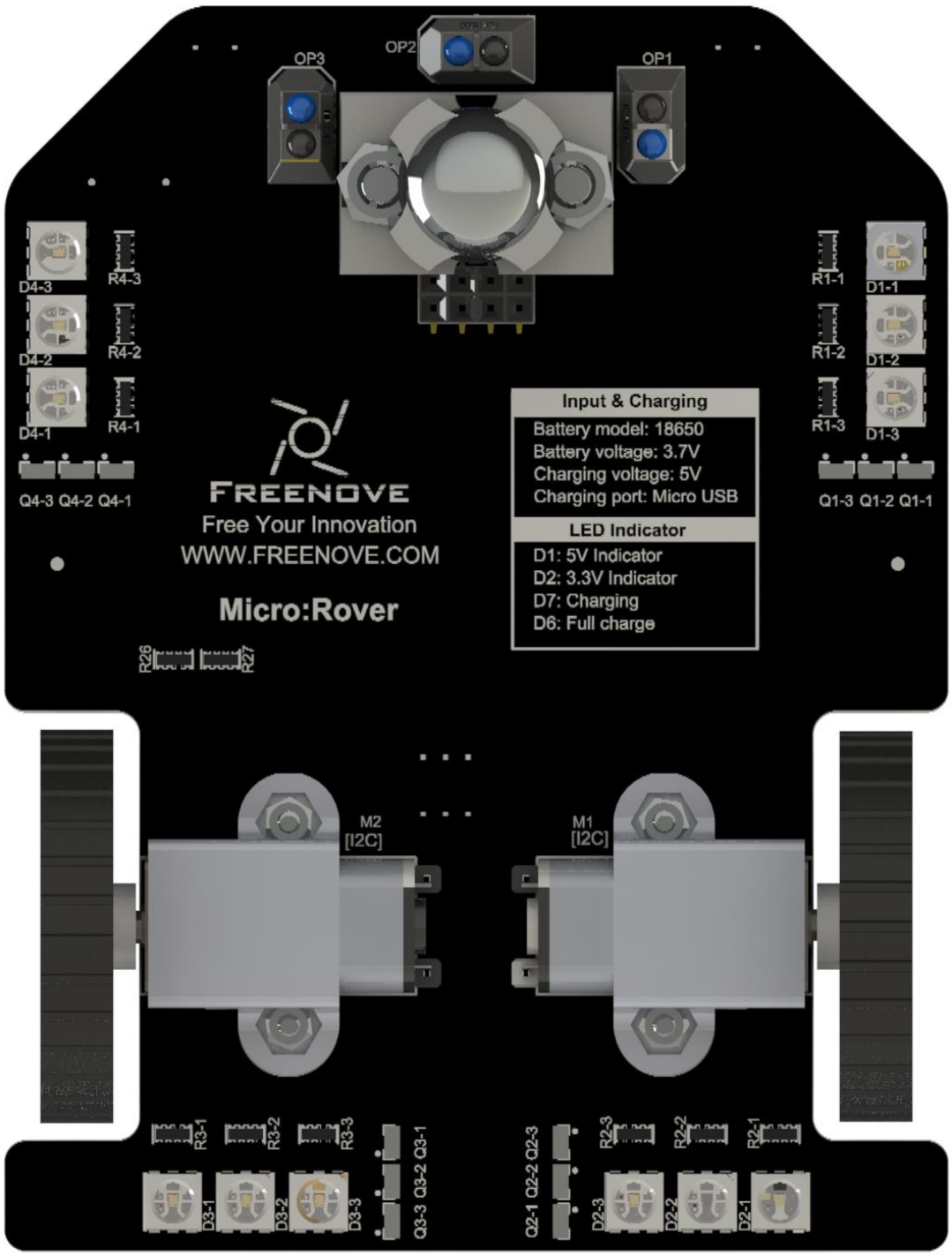
This chapter will introduce the functions and features of Micro:Rover.

Freenove Micro:Rover is a programmable car and it is based on BBC micro:bit. With integrated design, it does not require additional wiring, which makes it easy to use. There are ultrasonic, infrared and photosensitive sensors on it as well as buzzer, RGB LED and other peripherals. Rich hardware resources will help you master more knowledge and skills. You can use your imagination to create more ways to play the robot.



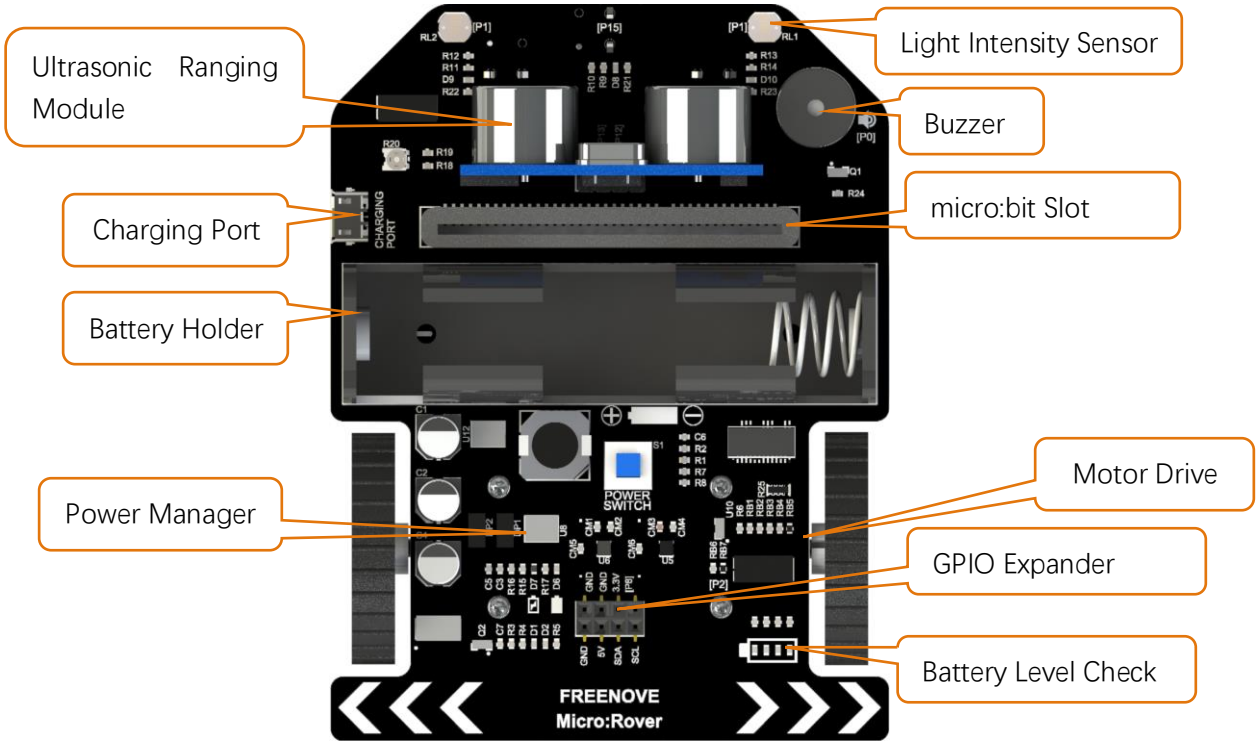
Meet micro:rover

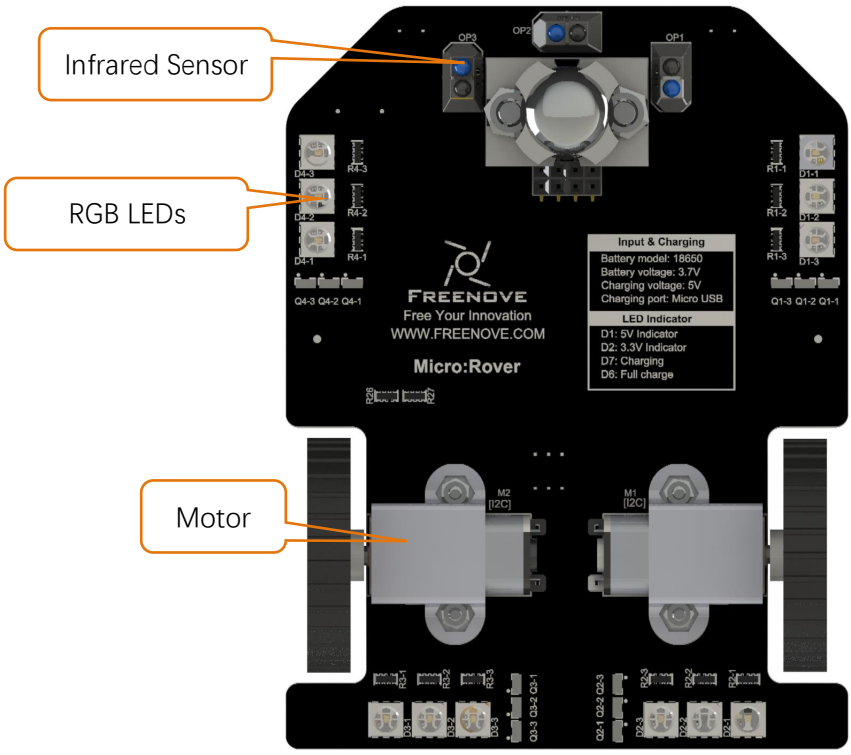




The Freenove Micro:Rover is a multi-functional car based on BBC micro:bit. It has rich sensors and peripherals to help you further learn how to use micro:bit and acquire knowledge about electronics.

Features


Ultrasonic Ranging Module
Light Intensity Sensor
Buzzer
Charging Port
micro:bit Slot
Battery Holder
Motor Drive
Power Manager
GPIO Expander
Battery Level Check
FREENOVE
Micro:Rover


Infrared Sensor
RGB LEDs
Motor
FREENOVE
Free Your Innovation
WWW.FREENOVE.COM
Micro:Rover
Input & Charging
Battery model: 18650
Battery voltage: 3.7V
Charging voltage: 5V
Charging port: Micro USB
LED Indicator
D1: 5V Indicator
D2: 3.3V Indicator
D7: Charging
D6: Full charge

Features of Micro:Rover:

- 4 Sets of controllable RGB LEDs (3 LEDs in each set)
- 2*Deceleration motors
- 1*Passive buzzer
- 1*Ultrasonic ranging module
- 2*Light intensity sensors
- 3*Infrared line-tracking sensors
- 1*micro:bit GPIO [P8]
- 1 Set of micro:bit I2C interface
- Battery power indication
- Battery charging and discharging management

Connection between Micro:Rover and micro:bit GPIO:

micro:bit GPIO	Micro:Rover
P0	Buzzer
P1	Light intensity sensor
P2	Battery voltage acquisition
P8	Expansion port
P12	Ultrasonic ranging module trig pin
P13	Ultrasonic echo pin
P14	Left Infrared line-tracing sensor
P15	Middle Infrared line-tracing sensor
P16	Right Infrared line-tracing sensor
P19-I2C SCL	Motors, RGBLEDs, I2C Extension port
P20-I2C SDA	

Battery & Charging

Freenove Micro:Rover is powered by one 18650 battery. The extended battery holder can be compatible with any type of 18650 batteries including pointed/flat/with or without protective plate. The discharge current of the battery should be at least 2A.

You can charge the battery with a USB port or a common USB charger. The maximum charging current is 700 mA.

Please note: this product does not contain battery.

Indicator

Some LEDs are integrated on Rover, different states of each have various meaning. You can know better about working state of Rover through them.

LED	Indicating content	LED On	LED Off
D1	5V circuit	5V circuit is working.	5V circuit is not working.
D2	3.3V circuit	3.3V circuit is working.	3.3V circuit is not working.
D8	Middle line-tracking sensor	Black object is detected or no object is detected.	White object is detected
D9	Left line-tracking sensor		
D10	Right line-tracking sensor		

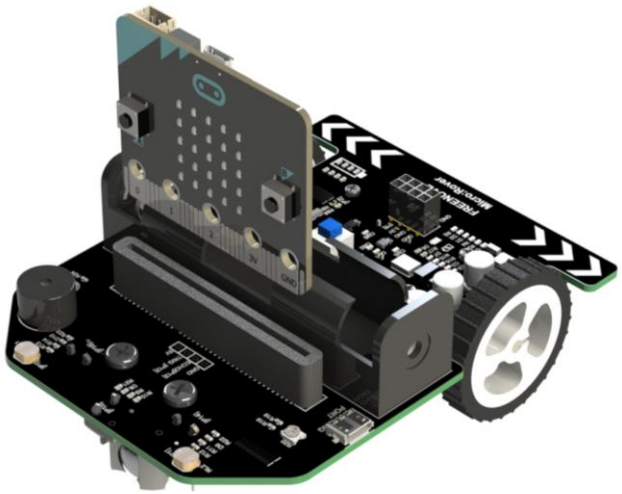
LEDs D6 and D7 indicate battery charging and discharging state respectively. Under normal working conditions, their states and corresponding meanings are shown below.

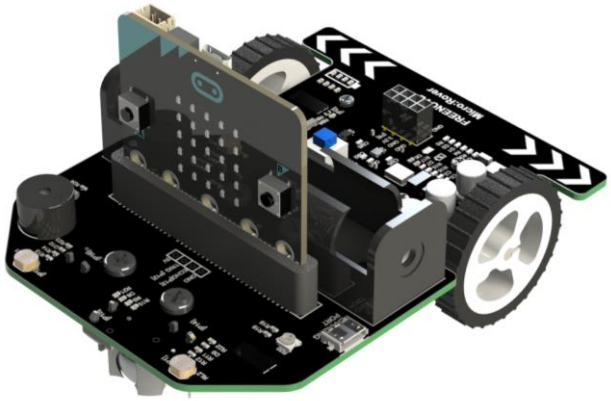
D7 State	D6 State	Indicating meaning
On	Off	Charging.
Off	On	Charging is complete and the battery is full.
Blink	On	The charger did not detect the battery.
Off	Off	The boost circuit is working.

On bottom right corner of Rover, four LEDs are used to indicate battery level, as shown blow.

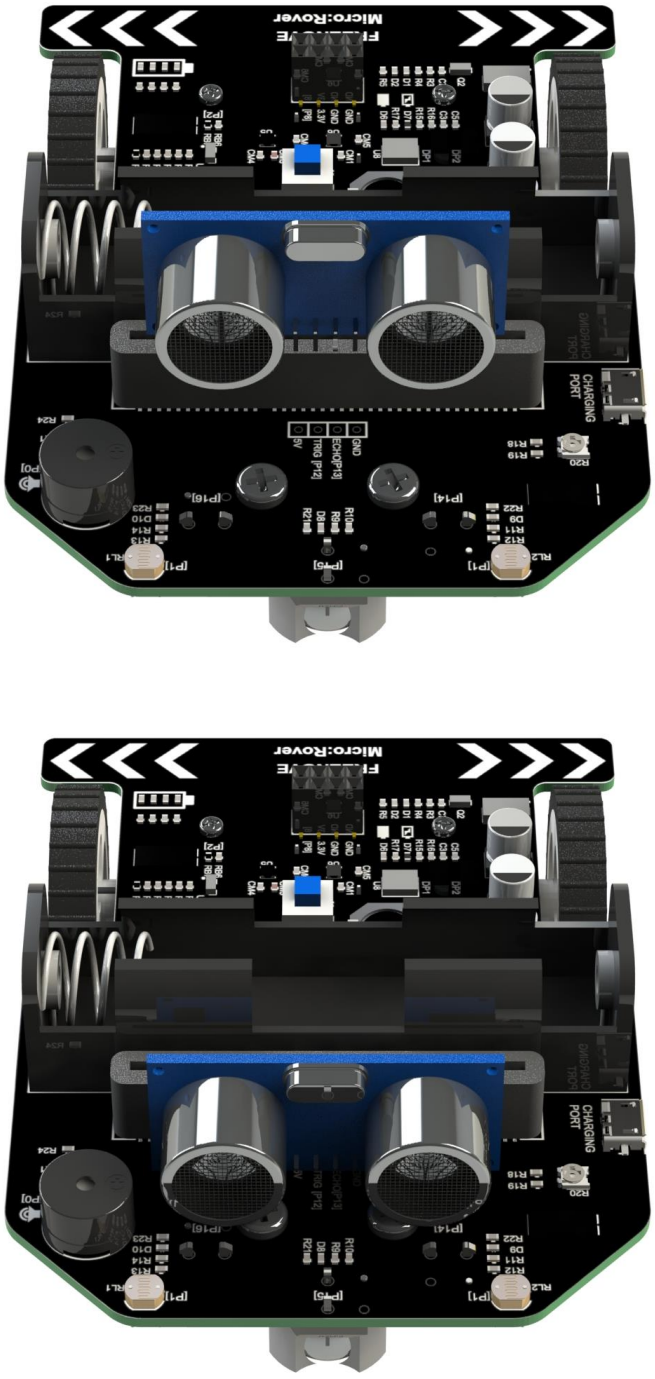
Assemble

Install micro:bit

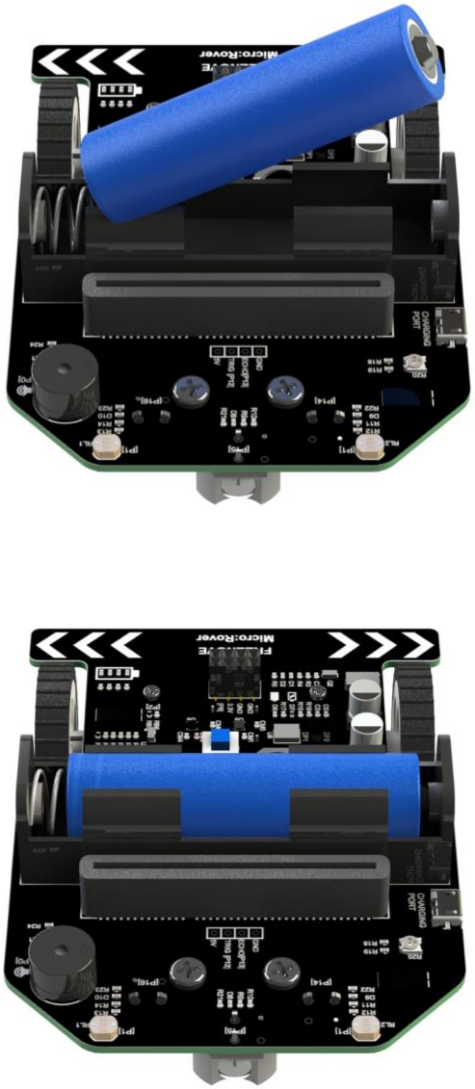




Install ultrasonic ranging module



Install battery



Code & Programming

This tutorial is written for Python language. If you want to program with graphical block code, please refer to the manual "Tutorial.pdf". In the root directory of the resource you download, there is a folder called "PythonProjects" that holds all the Python code for Rover. The name of Python code file ends with ".py".

Python

micro:bit can be programmed in Python. Since micro:bit is a microcontroller, the hardware difference makes it not fully support Python. Here is 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.
There are two types of editors for Python code.

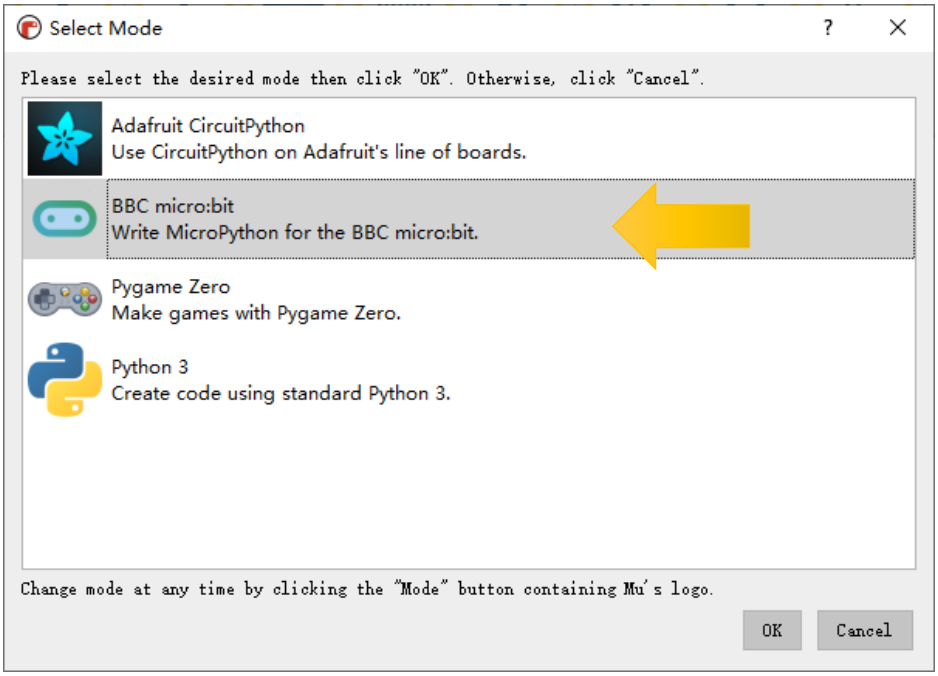
1. The web version for Python code editor is here: https://python.microbit.org/v/1.1
2. For Windows and MAC users, you can use the independent software Mu.

(https://codewith.mu/en/download)
We highly recommend using the software Mu as the Python code editor.

Mu

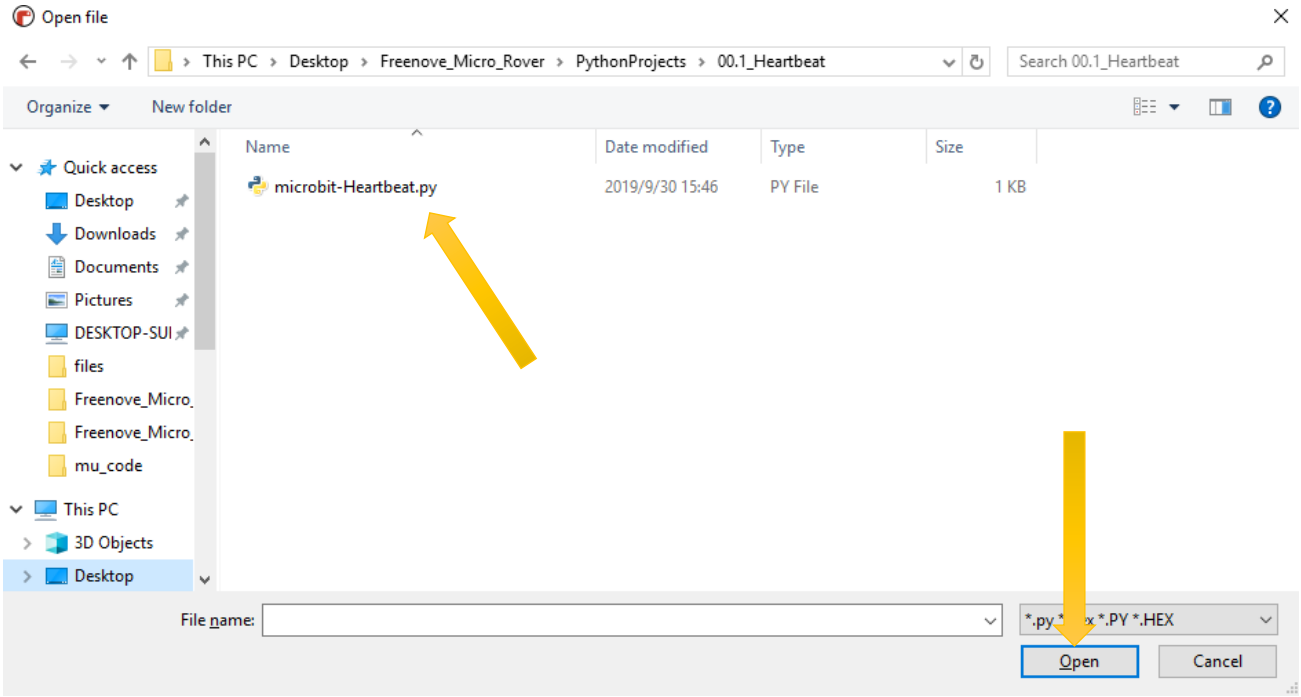
Mu is a Python code editor for beginner programmers based on extensive feedback given by teachers and learners.
Official website: https://codewith.mu/
You can download it here: https://codewith.mu/en/download
Download and install it. Then open it, following interface appears:

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

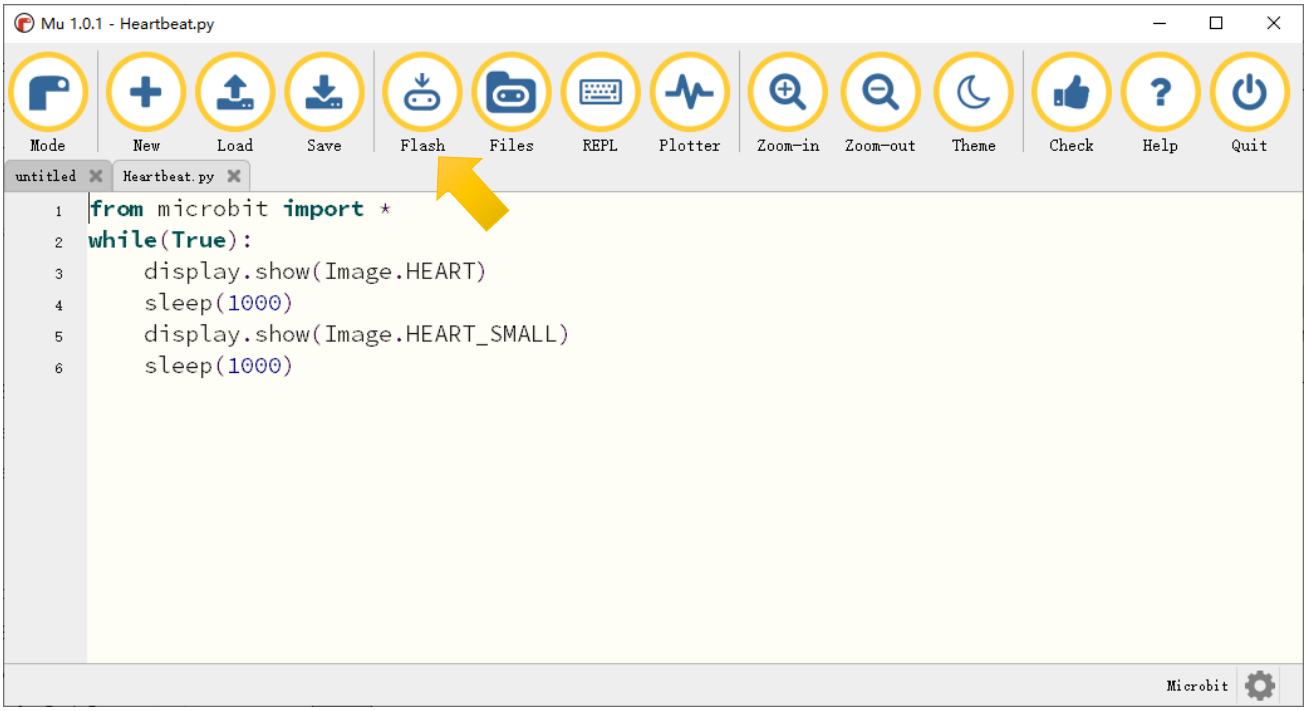


Click the "Load" button, select "microbit-Heartbeat.py" in the pop-up box, and click "Open". The file path is shown in the following table.

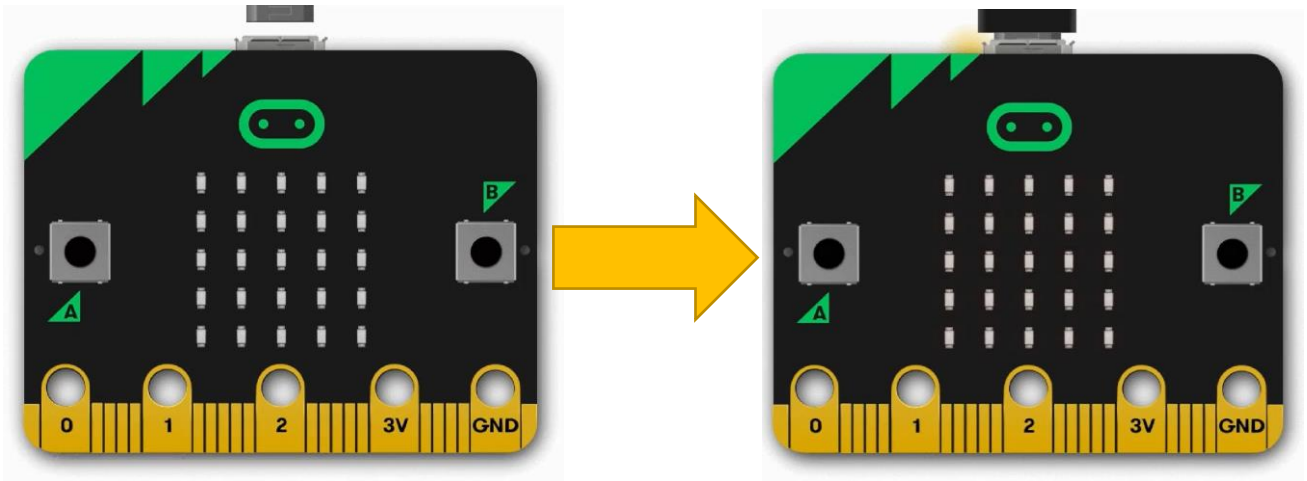
File type	Path	File name
Python file	../PythonProjects/00.1_Heartbeat	microbit-Heartbeat.py



Successful loading is shown below. You can also type the code by yourself.

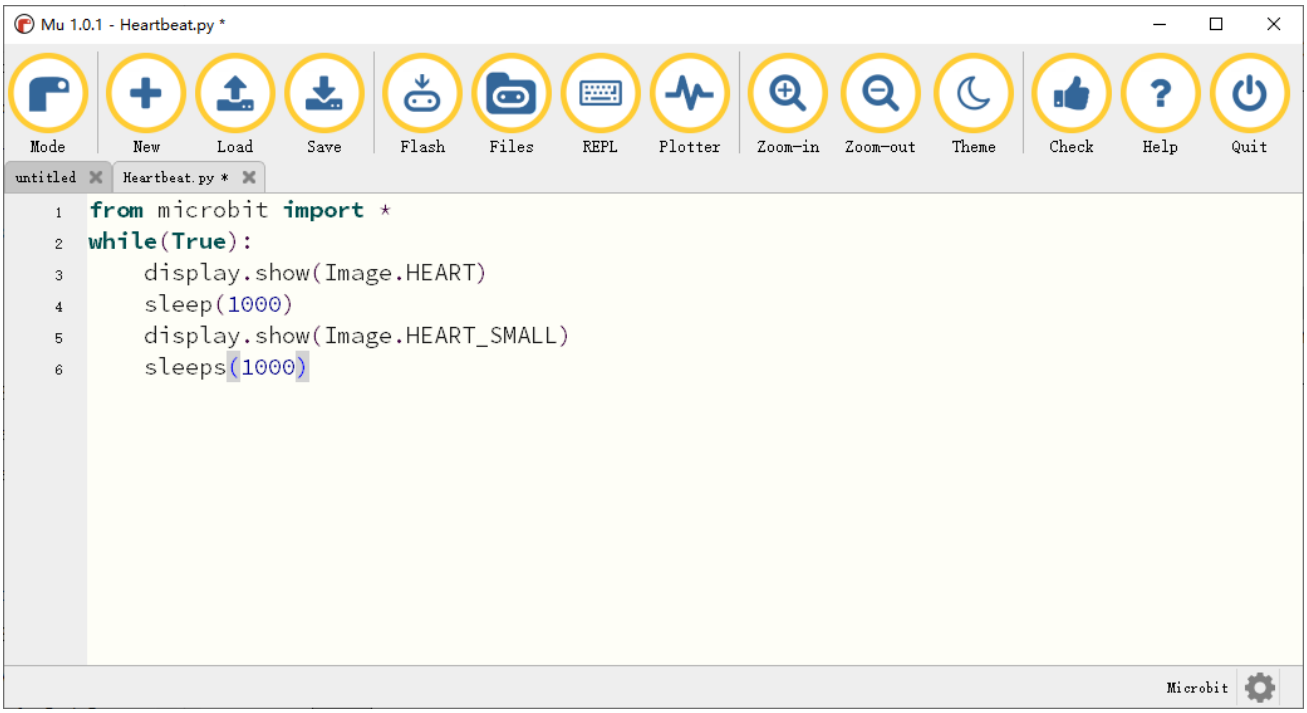


Connect the micro:bit to your computer via a micro USB cable. And click the **Flash** button to download the program into micro:bit.

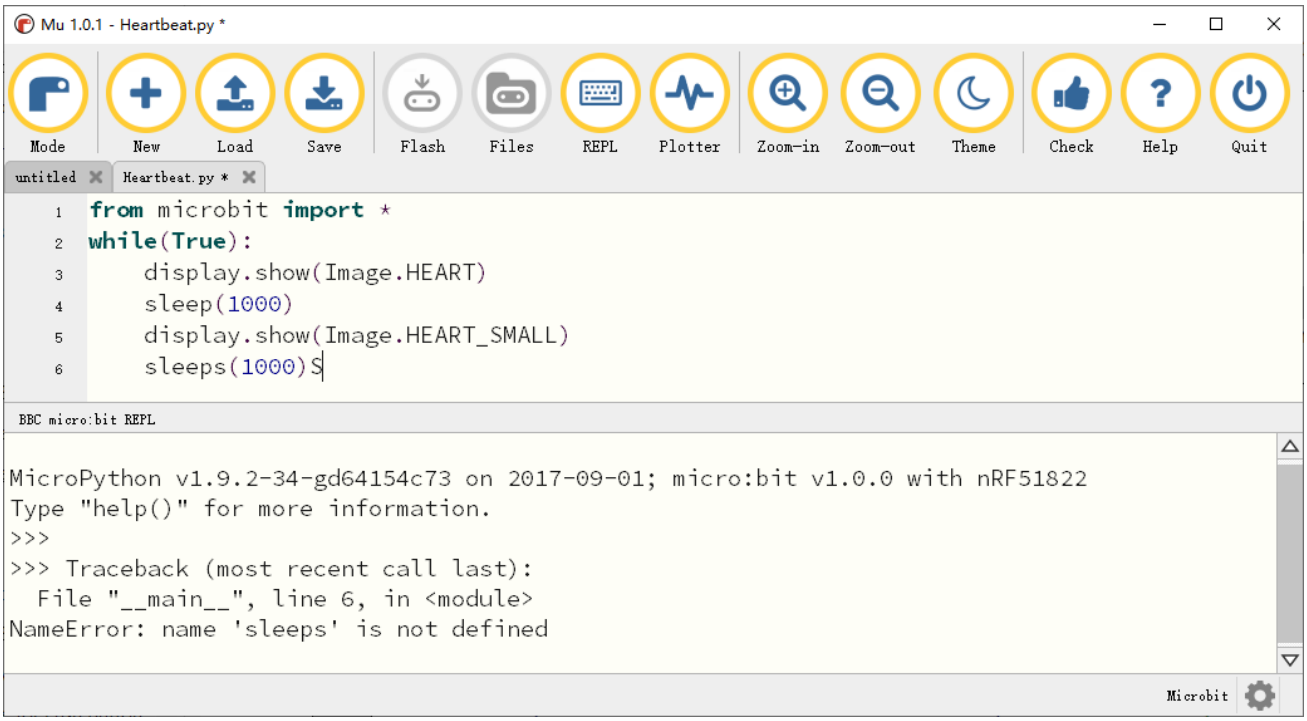


If there are errors in your code, you may be able to successfully download the code to micro:bit, but it will not work properly.

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.

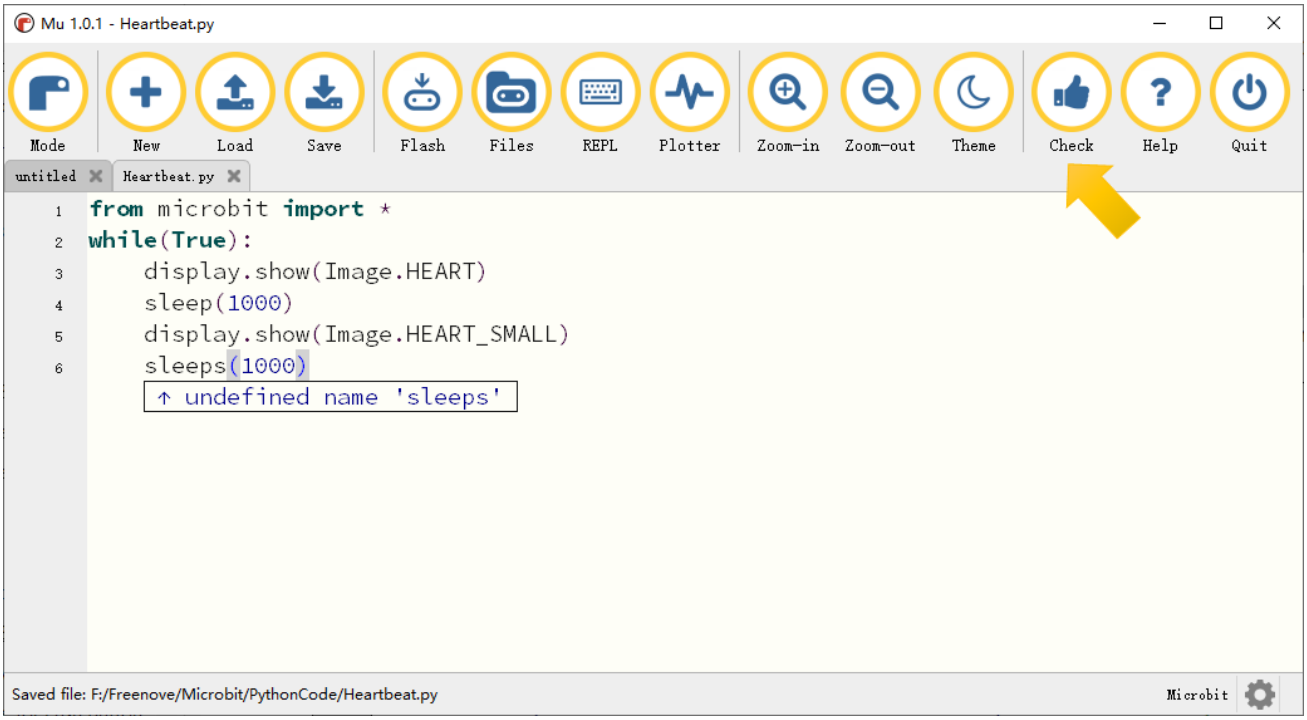


Click the "REPL" button and press the reset button (the button on the back, not A, B) on micro:bit. The error message will be displayed in the REPL box, as shown below:

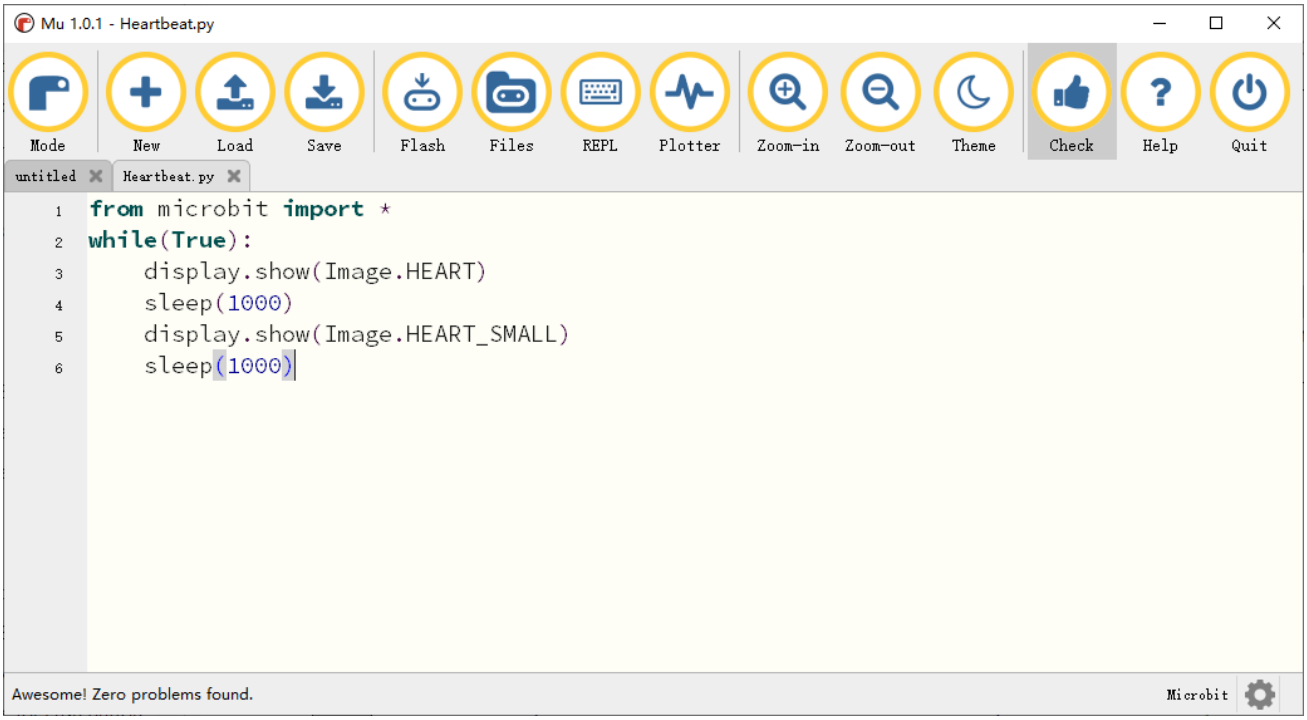


Click REPL again, you will close REPL mode. And then you can flash new code.

In order 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, and Mu will indicate the error of the code.



According to the error prompt, modify the code correctly. Then click the "Check" button again, Mu displays no error on the bar below.



For more tutorials for using Mu, please refer to: https://codewith.mu/en/tutorials/

Chapter 1 Music

You can use the buzzer on the car to play music.

Preparation

1. Insert micro:bit into Rover correctly.
2. Install battery into Rover.
3. Turn ON Rover power.
4. Connect micro:bit and computer through USB cable.

Play a note

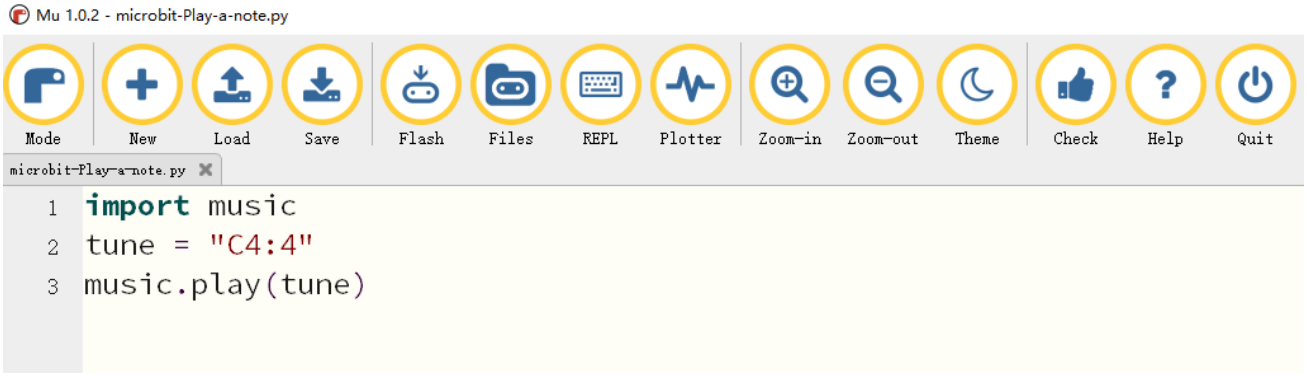
Let the car play a note.

Code

Open the "microbit-Play-a-note.py" with the Mu software. The path to the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	../PythonProjects/01.1_Play-a-note	microbit-Play-a-note.py

After the loading is completed, the interface of Mu is as shown below:



Download the code to micro:bit and the car will play a note. "C4:4" means that the note is C4 and the beat is 4.

The following is the program code:

```
1 import music
2 tune = "C4:4"
3 music.play(tune)
```

Import the music module, which contains functions that control the sound and make sounds.

```
1 import music
```

Create a variable tune to store notes.

```
1 tune = "C4:4"
```

Call the play() function to play the notes stored in the variable tune.

```
1 music.play(tune)
```

Reference

music.play()

It is used to play music. MicroPython has quite a lot of built-in melodies.

For more information, please refer to:

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

Play a melody

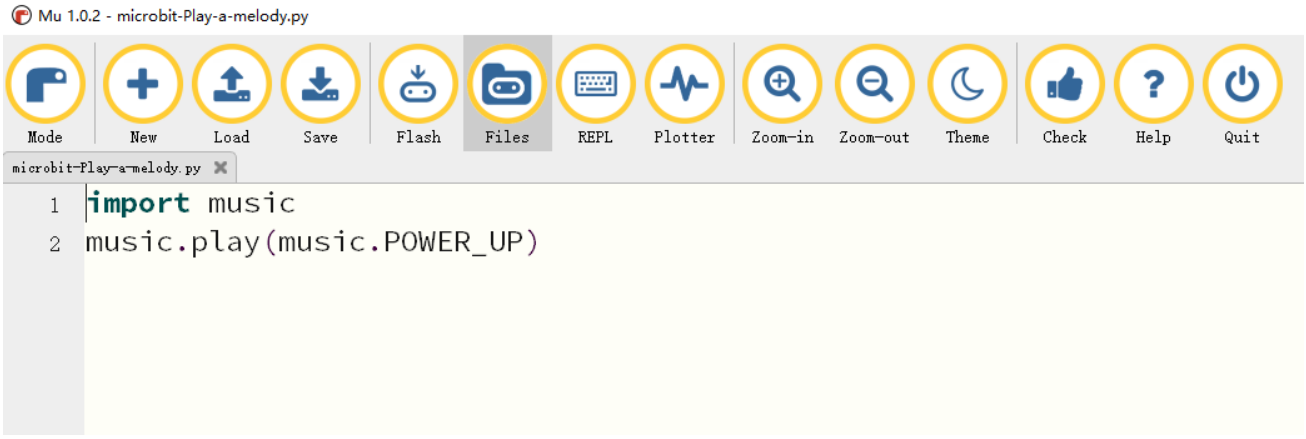
Let the car play a melody.

Code

Open the "microbit-Play-a-melody.py" with the Mu software. The path of the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	.. /PythonProjects/01.2_Play-a-melody	microbit-Play-a-melody.py

After the loading is completed, the interface of Mu is as shown below:



Download the code to micro:bit and the car will play a melody called "POWER_UP".

The following is the program code:

```
1  import music
2  music.play(music.POWER_UP)
```

Import music module.

```
1  import music
```

Call the play() function to play a melody called "POWER_UP"

```
1  music.play(music.POWER_UP)
```

Play custom melody

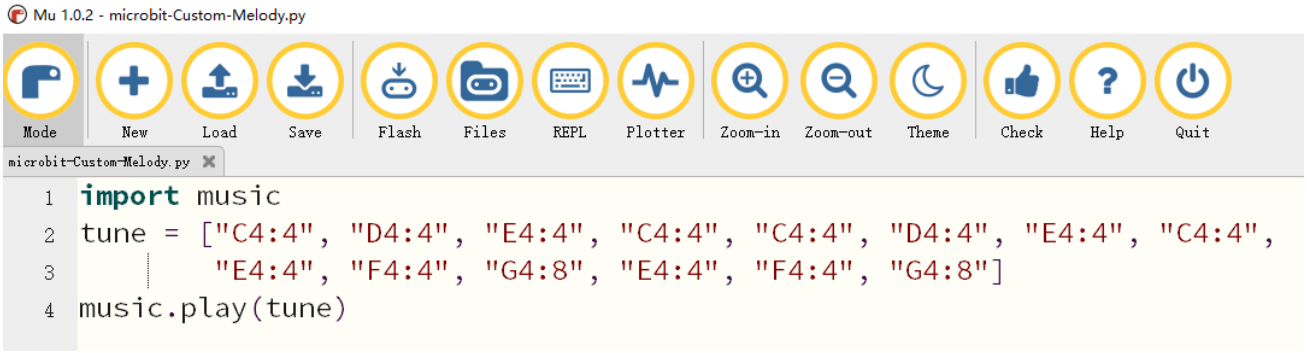
Let the car play a custom melody.

Code

Open the "microbit-Custom-Melody.py" with the Mu software. The path of the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	.. /PythonProjects / 01.3_Custom-Melody	microbit-Custom-Melody.py

After the loading is completed, the interface of Mu is as shown below:



Download the code to micro:bit and the car will play a custom melody.

The following is the program code:

```
1 import music
2 tune = ["C4:4", "D4:4", "E4:4", "C4:4", "C4:4", "D4:4", "E4:4", "C4:4",
3         "E4:4", "F4:4", "G4:8", "E4:4", "F4:4", "G4:8"]
  music.play(tune)
```

Import the music module, which contains functions that control the sound and make sounds.

```
1 import music
```

Create a variable tune to store notes.

```
1 tune = ["C4:4", "D4:4", "E4:4", "C4:4", "C4:4", "D4:4", "E4:4", "C4:4",
          "E4:4", "F4:4", "G4:8", "E4:4", "F4:4", "G4:8"]
```

Call the play() function to play the notes stored in the variable tune.

```
1 music.play(tune)
```

Chapter 2 RGB LED

There are four sets of RGBLEDs integrated on Rover, each has three RGBLEDs and can be controlled independently. Now create your own RGB light.

Preparation

1. Insert micro:bit into Rover correctly.
2. Install battery into Rover.
3. Turn ON Rover power.
4. Connect micro:bit and computer through USB cable.

Emitting one color of light

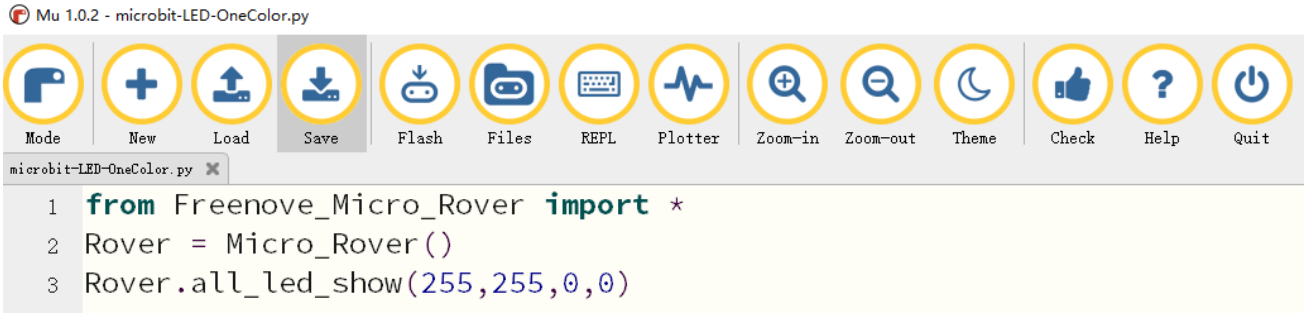
Let all the LEDs on the car display the same color with same brightness.

Code

Open the "microbit-LED-OneColor.py" with the Mu software. The path of the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	.. /PythonProjects / 02.1_LED-OneColor	microbit-LED-OneColor.py

After the loading is completed, the interface of Mu is as shown below::



Don't rush to click on "Flash", because you still need some extra work: import the "Freenove_Micro_Rover.py" file into micro:bit. This file contains methods to control Rover, which were integrated into a class "Micro_Rover", making it easier for you to control Rover through Python code. This is similar to the extension in the block code.

Import the "Freenove_Micro_Rover.py" file.

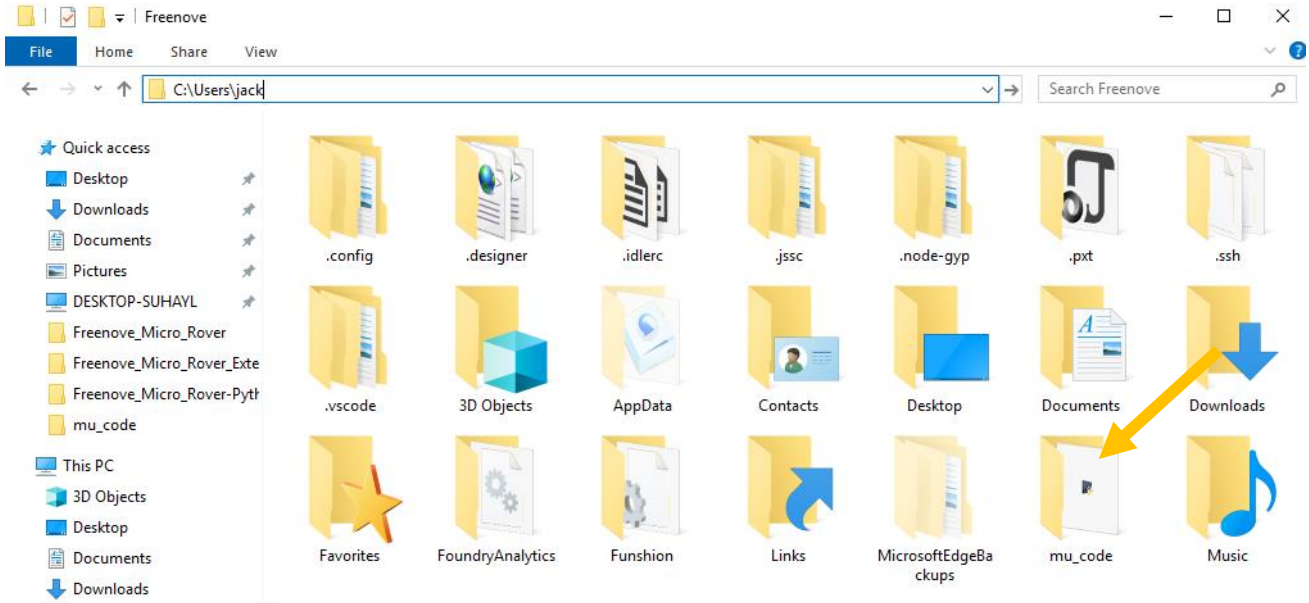
The default directory for Mu to save files is "mu_code", which is in the root of your user directory.

Reference: https://codewith.mu/en/tutorials/1.0/files

For example, in the windows system, assuming your system is installed on the C drive, the user name is "jack", then the path of the "mu_code" directory is "C:\Users\jack\mu_code".

On Linux systems, the path to the "mu_code" directory is "~/home/mu_code"

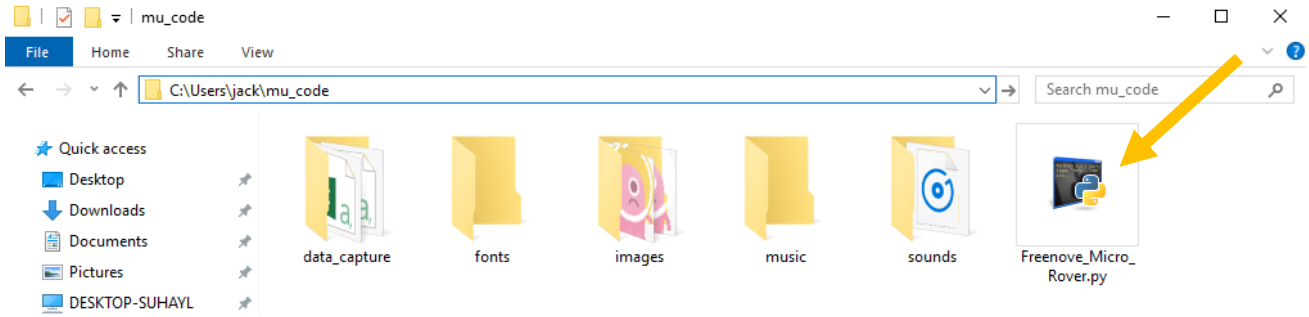
Enter the "mu_code" folder.



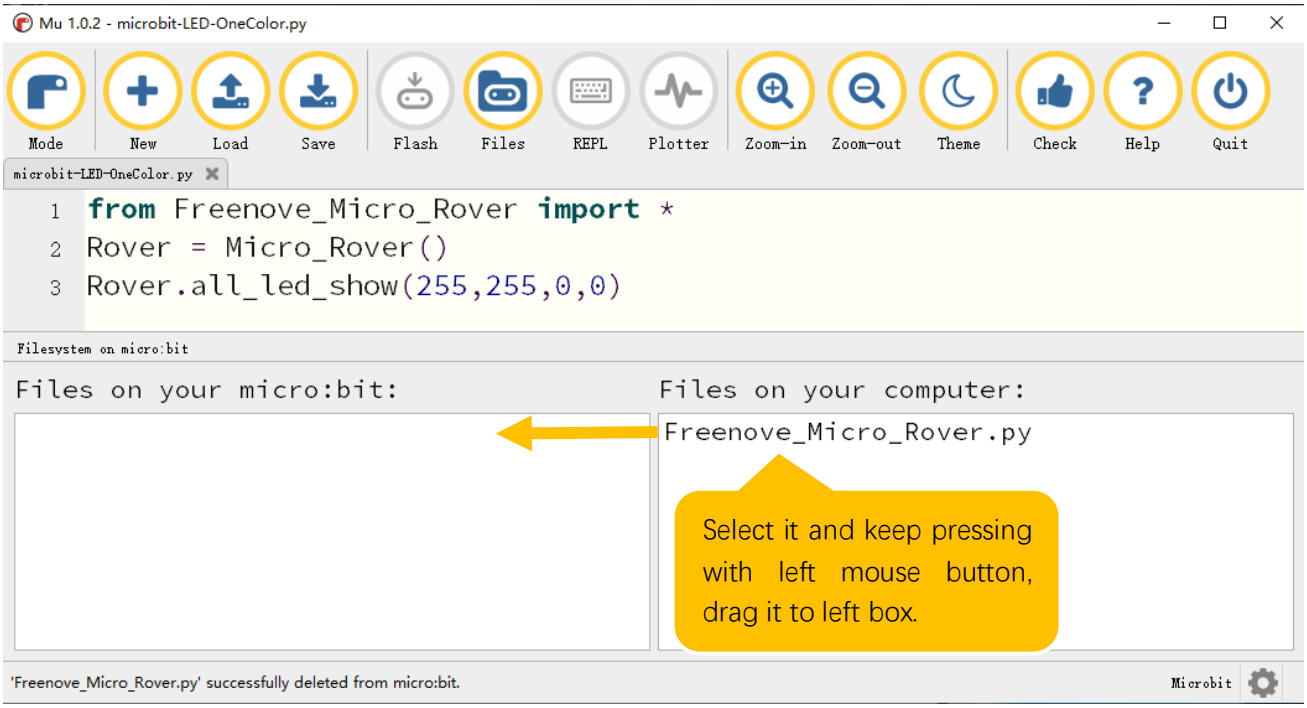
Copy the "Freenove_Micro_Rover.py" file to the "mu_code" folder. The path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects / Libraries	Freenove_Micro_Rover.py

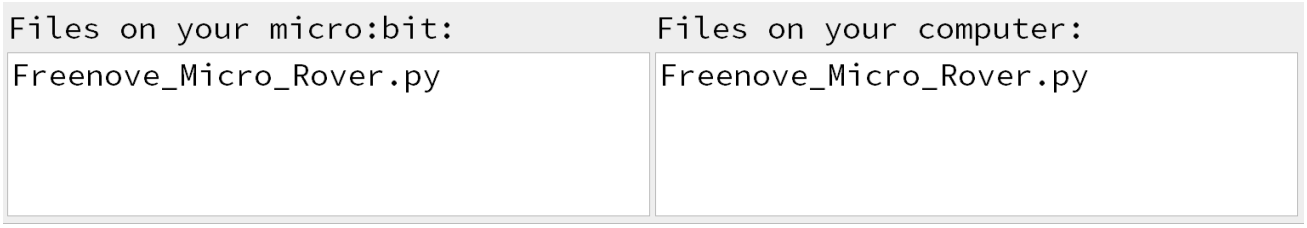
The copy is successful as shown below:



Open the Mu software and connect micro:bit to the computer. Click on "Files".
Import"Freenove_Micro_Rover.py" into micro:bit.



After a few seconds, the import is completed.
After the import is completed, click the "Flash" button to download the code to micro:bit. All LEDs on the car emit red light.
After importing successfully, you will see it on left.



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

The following is the program code:

```
1 from Freenove_Micro_Rover import *
2 Rover = Micro_Rover()
3 Rover.all_led_show(255,255,0,0)
```

Import the Freenove_Micro_Rover module, which contains functions that control LEDs, drive motors, etc.

```
1 from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
1 Rover = Micro_Rover()
```

Call the all_led_show() function in the Micro_Rover class to turn ON all the LEDs on the car.

```
1 Rover.all_led_show(255,255,0,0)
```

Reference

`all_led_show(Brightness，R,G,B)`

Description:

Set all RGB LEDs on the Rover to the same brightness and color. This function belongs to the Micro_Rover class in the Freenove_Micro_Rover module.

Parameter:

Brightness: Specifies the brightness of the LED, the value ranges from 0-255.

R, G, B: Specifies the color value, the value ranges from 0-255.

Return:

Nothing

Example:

All_led_show(255,0,255,0), sets all RGB LEDs to green (0, 255, 0) with maximum brightness (255).

Emitting different colors of light

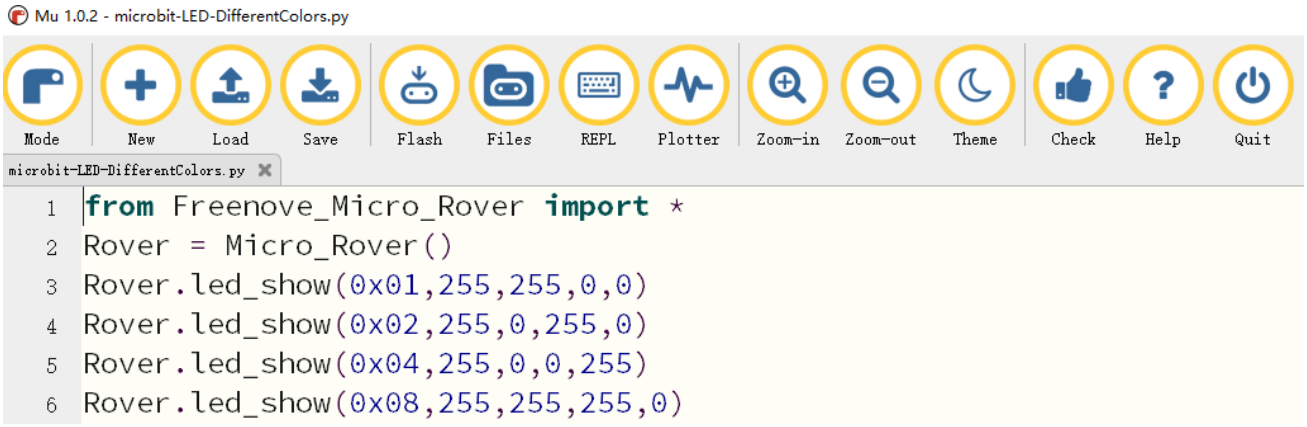
Let the LEDs on the car display different colors.

Code

Open the "microbit-LED-DifferentColors.py" with the Mu software. The path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects / 02.2_LED-DifferentColors	microbit-LED-DifferentColors.py

After the loading completes, the interface of Mu is as shown below::



Import the "Freenove_Micro_Rover.py" file into micro:bit, (How to import files?), then download the code to micro:bit. LED1 on the car emits red, LED2 emits green, LED3 emits blue, and LED4 emits yellow.

The following is the program code:

```
1  from Freenove_Micro_Rover import *
2  Rover = Micro_Rover()
3  Rover.led_show(0x01, 255, 255, 0, 0)
4  Rover.led_show(0x02, 255, 0, 255, 0)
5  Rover.led_show(0x04, 255, 0, 0, 255)
6  Rover.led_show(0x08, 255, 255, 255, 0)
```

Import the Freenove_Micro_Rover module.

```
1 from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
1 Rover = Micro_Rover()
```

Call the led_show () function in the Micro_Rover class to illuminate the LED on the car. The brightness is the maximum brightness (255), LED1 emits red (255,0,0), LED2 emits green (0,255,0), LED3 emits blue (0,0,255), and LED4 emits yellow (255,255,0).

```
1 Rover.led_show(0x01,255,255,0,0)
  Rover.led_show(0x02,255,0,255,0)
  Rover.led_show(0x04,255,0,0,255)
  Rover.led_show(0x08,255,255,255,0)
```

Reference

`led_show(Index,Brightness,R,G,B)`

Description:

Set the brightness and color of selected RGBLED.

This function belongs to the Micro_Rover class in the Freenove_Micro_Rover module.

Parameter:

Index: Specify the LED sequence number. LED1: 0x01, LED2: 0x02, LED3: 0x04, LED4: 0x08.

Brightness: Specifies the brightness of the LED, the value range is 0-255.

R, G, B: Specify the color value, the value range is 0-255.

Return:

Nothing

Example:

Led_show (0x01, 255, 0, 255, 0), set LED1 with the sequence number 0x01 to light green (0, 255, 0) with maximum brightness (255).

Emitting random color of light

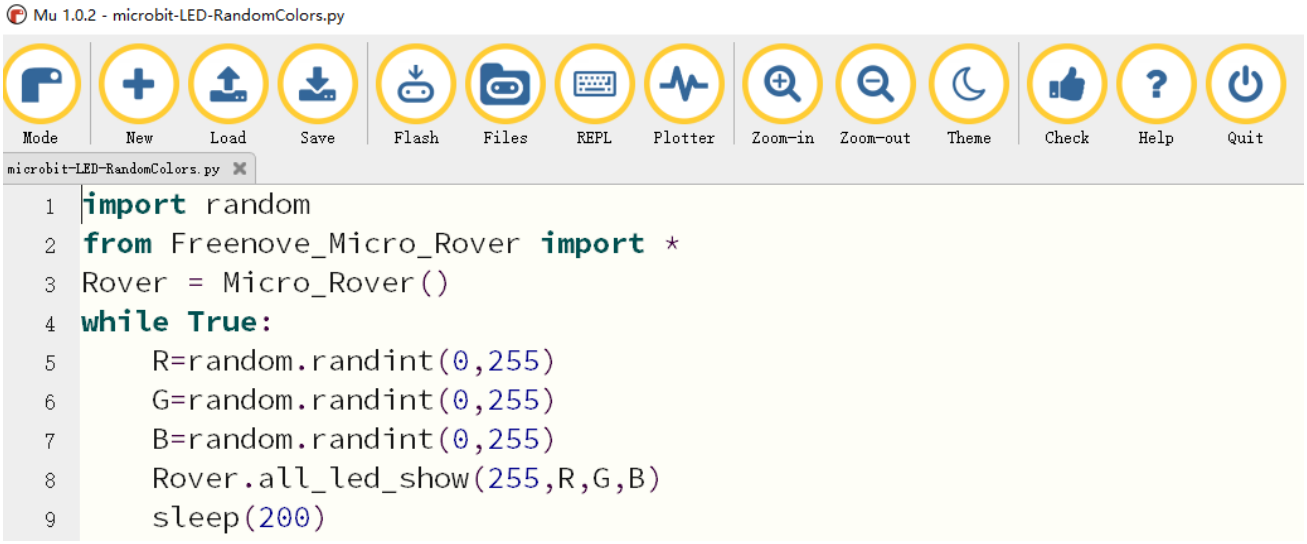
Let the LED on the car show a random color.

Code

Open the "microbit-LED-DifferentColors.py" with the Mu software. The path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects / 02.2_LED-DifferentColors	microbit-LED-DifferentColors.py

After the loading completes, the interface of Mu is as shown below::



Import the "Freenove_Micro_Rover.py" file into micro:bit (How to import files?) and download the code into micro:bit. The LEDs on the car will show random colors.

The following is the program code:

```
1 import random
2 from Freenove_Micro_Rover import *
3 Rover = Micro_Rover()
4 while True:
5     R=random.randint(0,255)
6     G=random.randint(0,255)
7     B=random.randint(0,255)
8     Rover.all_led_show(255,R,G,B)
9     sleep(200)
```

Import the random module. MicroPython comes with a random module to make it easy to introduce chance and a little chaos into your code.

```
import random
```

Import the Freenove_Micro_Rover module.

```
from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
Rover = Micro_Rover()
```

This is a permanent loop that will cause the micro:bit to execute the code in this loop forever.

```
1 while True:
```

Call the randint() function in the random module to generate a random number within 0-255. Assign the obtained random number to the R, G, and B variables. Then call the all_led_show() function to let the LED show the color corresponding to the RGB value.

```
1 R=random.randint(0,255)
  G=random.randint(0,255)
  B = random.randint (0,255)
  Rover.all_led_show(255,R,G,B)
```

The delay is 0.2S, so the color is changed every 0.2S.

```
1 sleep(200)
```

Reference

randint(a,b)

Return a random integer N such that a <= N <= b. Alias for randrange(a, b+1).

Example: randint(0,100), which returns an integer in the range 0-100.

For more information: https://microbit-micropython.readthedocs.io/en/latest/random.html

sleep(ms)

sleep for the given number of milliseconds.

For more details about sleep function,

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

Emittinng soft colors

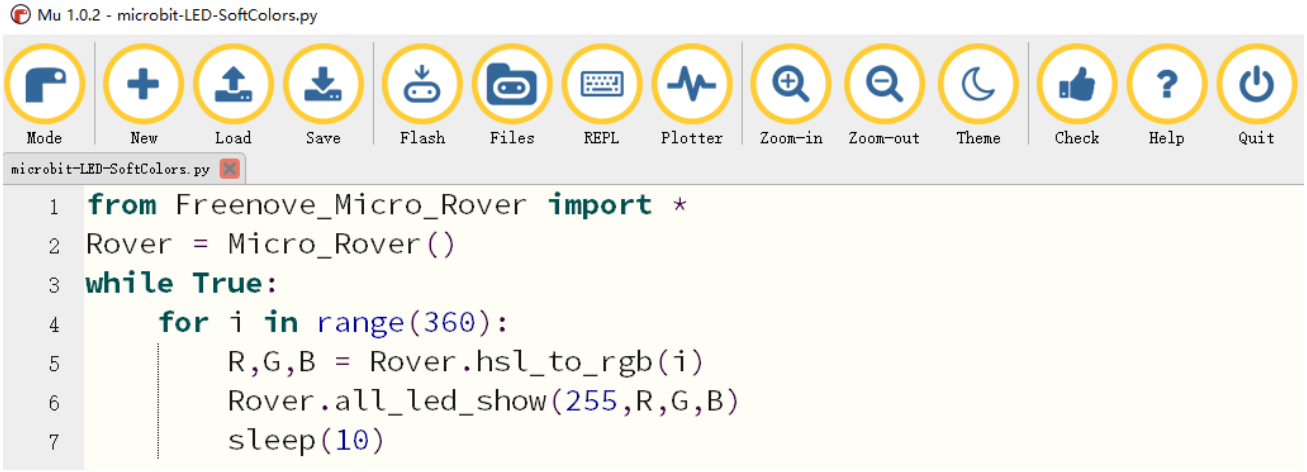
Let the LEDs on the car emit soft colors.

Code

Open the "microbit-LED-RandomColors.py" with the Mu software. The path of the code is as follows:

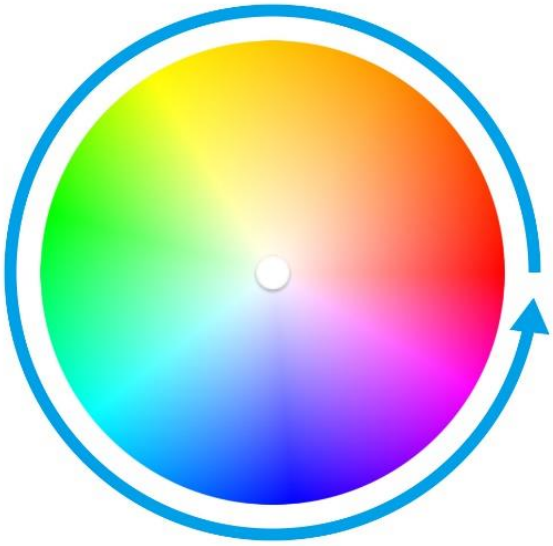
File type	Path	File name
Python file	.. /PythonProjects / 02.3_LED-RandomColors	microbit-LED-RandomColors.py

After the loading completes, the interface of Mu is as shown below:



Import the "Freenove_Micro_Rover.py" file into micro:bit(How to import files?), then download the code to micro:bit, and LEDs emit soft colors on the car.

In this code, the HSL color model is used. The hue range is 0-360, and when the hue is different, the colors will be different, as shown in the following figure:



The following is the program code:

```
1  from Freenove_Micro_Rover import *
2  Rover = Micro_Rover()
3  while True:
4      for i in range(360):
5          R,G,B = Rover.hsl_to_rgb(i)
6          Rover.all_led_show(255,R,G,B)
7          sleep(10)
```

Import the Freenove_Micro_Rover module.

```
from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
Rover = Micro_Rover()
```

Create a for loop that loops 360 times in the while loop, simulating the hue of 0-360. Call the hsl_to_rgb () function to convert the color value of the corresponding hue to RGB values. Then turn ONthe LED and change the color every 10ms.

```
while True:
    for i in range(360):
        R,G,B = Rover.hsl_to_rgb(i)
        Rover.all_led_show(255,R,G,B)
        sleep(10)
```

Reference

`hsl_to_rgb(h)`

Description:

Convert the HSL color model to an RGB color model. This function belongs to the Micro_Rover class in the Freenove_Micro_Rover module.

Parameter:

h: Hue value of the HSL color model, ranging from 0-360.

Return:

R, G, B: Three values representing the color values of the three channels R, G, and B.

Example:

hsl_to_rgb(0). The return color is red, and the RGB value of the hue 0 is (255, 0, 0).

Chapter 3 Ultrasonic Ranging

The ultrasonic module on Rover can measure the distance between Rover and the obstacle in front of it. Therefore, take advantage of this characteristic to make Rover avoid obstacle When Rover is close to the obstacle in the front, let it turn to avoid the obstacle.

Preparation

1. Insert micro:bit into Rover correctly.
2. Install battery into Rover.
3. Turn ON Rover power.
4. Connect micro:bit and computer through USB cable.

Obtain value of ultrasonic ranging

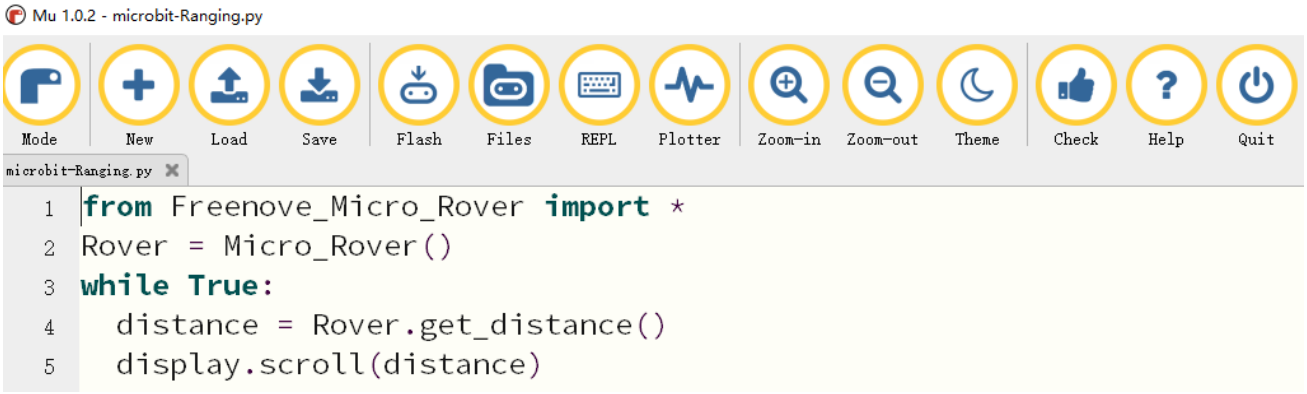
Obtain the distance measurement value of the ultrasonic module on the car, and then display it with LED matrix on the micro:bit.

Code

Open the "microbit-Rangingr.py" with the Mu software. The path of the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	.. /PythonProjects / 03.1_Ranging	microbit-Ranging.py

After the loading completes, the interface of Mu is as shown below:



Import the “Freenove_Micro_Rover.py” file into micro:bit (How to import files?) and download the code into micro:bit. The LED matrix on the micro:bit will display the distance value of the ultrasonic module.

The following is the program code:

```
1  from Freenove_Micro_Rover import *
2  Rover = Micro_Rover()
3  while True:
4    distance = Rover.get_distance()
5    display.scroll(distance)
```

Import the Freenove_Micro_Rover module, which contains functions that control LEDs, drive motors,etc.

```
   from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
   Rover = Micro_Rover()
```

Call the custom get_distance () function in the while loop to get the value of ultrasonic module. Finally, call the display.scroll() function to display the value on the LED matrix.

```
   while True:
     distance = Rover.get_distance()
     display.scroll(distance)
```

Reference

get_distance()

Obtain the value of the ultrasonic module and return it with integer in CM.

display.scroll()

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

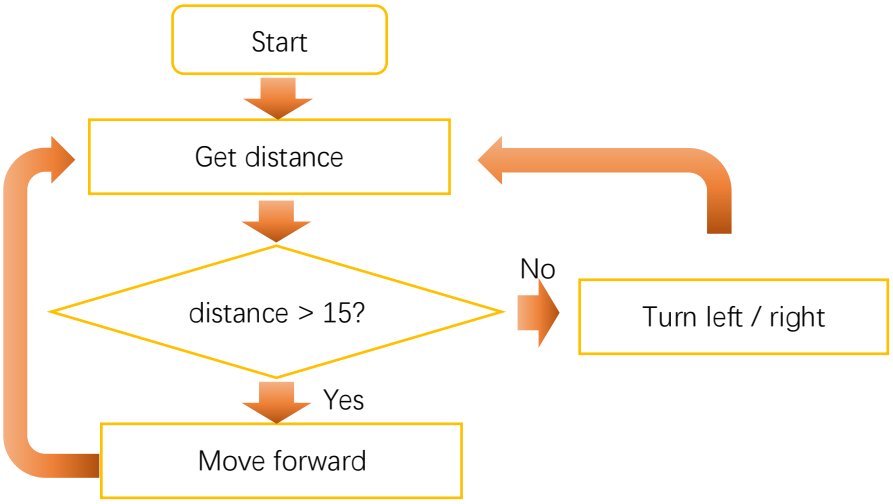
For more information, please refer to: https://microbit-micropython.readthedocs.io/en/latest/utime.html

Rover-Obstacle avoidance mode -1

In this project, we will realize the obstacle avoidance mode of Rover.

Flow chart

The program code is written according to flow chart, as shown below.

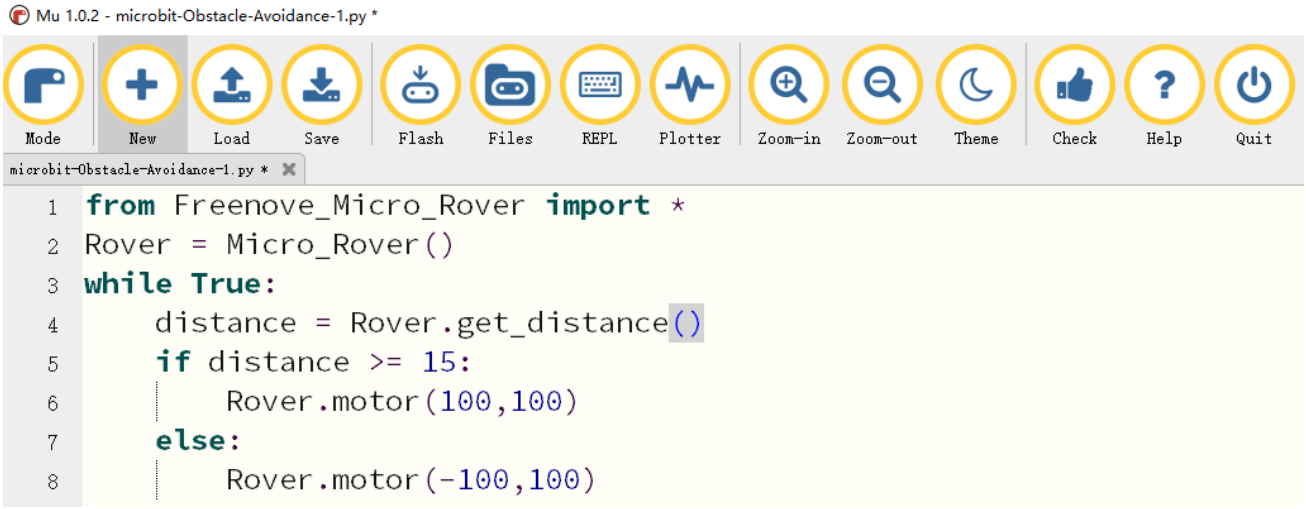


Code

Open the "microbit-Obstacle-Avoidance-1.py" with the Mu software. The path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects /03.2_Obstacle-Avoidance-1	microbit-Obstacle-Avoidance-1.py

After the loading completes, the interface of Mu is as shown below::



Import the "Freenove_Micro_Rover.py" file into micro:bit, then download the code into micro:bit and observe the movement of the car.

The following is the program code:

```
1 from Freenove_Micro_Rover import *
2 Rover = Micro_Rover()
3 while True:
4     distance = Rover.get_distance()
5     if distance >= 15:
6         Rover.motor(100,100)
7     else:
8         Rover.motor(-100,100)
```

Import the Freenove_Micro_Rover module.

```
1 from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
1 Rover = Micro_Rover()
```

Call the custom get_distance () function in the while loop to get the value of the ultrasonic module. The car will make movements according to the distance value received. If the distance is greater than 15 cm, the car advances. If the distance is less than 15cm, the car will turn left.

```
1  while True:
       distance = Rover.get_distance()
       if distance >= 15:
           Rover.motor(100,100)
       else:
           Rover.motor(-100,100)
```

Reference

`motor(left,right)`

Description:

Set the speed and direction of the two motors on the Rover. This function belongs to the Micro_Rover class in the Freenove_Micro_Rover module.

Parameter:

Left: Refers to the speed and direction of the motor on the left side, positive values indicate forward and negative values indicate backward. The absolute value of the value indicates the speed. Value range: -255~255.

Right: Refers to the speed and direction of the motor on the right side, positive values indicates forward, negative values indicate backwards, and the absolute value of the value indicates the speed. Value range: -255~255.

Return:

None

Example:

Motor (-255, -255), Rover will move back at maximum speed.

Rover-Obstacle avoidance mode -2

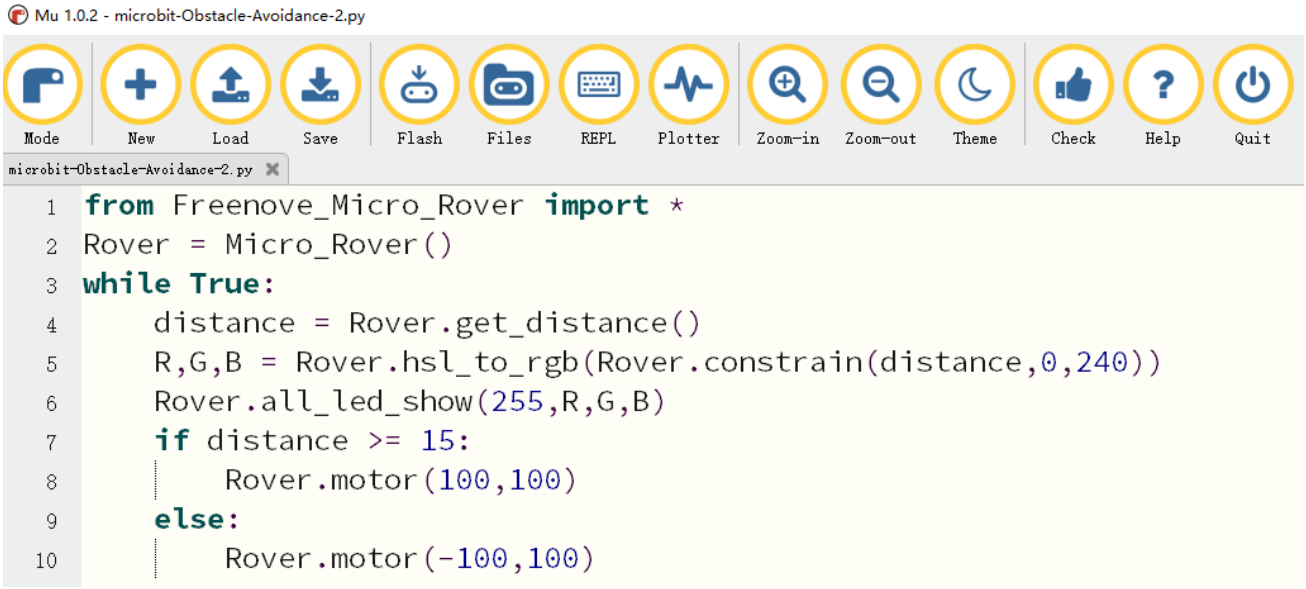
In this project, we will combine RGBLED and LED matrix to provide some auxiliary instructions for obstacle avoidance mode.

Code

Open the "microbit-Obstacle-Avoidance-2.py" with the Mu software. The path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects / 03.3_Obstacle-Avoidance-2	microbit-Obstacle-Avoidance-2.py

After the loading completes, the interface of Mu is as shown below:



Import the "Freenove_Micro_Rover.py" file into micro:bit and download the code into micro:bit, and then observe the movement of the Rover. When the measured distance is different, the LEDs display different colors.

The following is the program code:

```
1  from Freenove_Micro_Rover import *
2  Rover = Micro_Rover()
3  while True:
4      distance = Rover.get_distance()
5      R, G, B = Rover.hsl_to_rgb(Rover.constrain(distance, 0, 240))
6      Rover.all_led_show(255, R, G, B)
7      if distance >= 15:
8          Rover.motor(100, 100)
9      else:
10         Rover.motor(-100, 100)
```

Import the Freenove_Micro_Rover module.

```
from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
Rover = Micro_Rover()
```

Call the custom get_distance () function in the while loop to get the value of the ultrasonic module. Pass this value into the hsl_to_rgb() function, returning an RGB value to light up LED. Finally, the car will make movements according to the distance value received. If the distance is greater than 15 cm, the car moves forward. If the distance is less than 15cm, the car will turn left.

```
1  while True:
       distance = Rover.get_distance()
       R,G,B = Rover.hsl_to_rgb(Rover.constrain(distance,0,240))
       Rover.all_led_show(255,R,G,B)
       if distance >= 15:
           Rover.motor(100,100)
       else:
           Rover.motor(-100,100)
```

Reference

`constrain (value,low,high)`

Description:

Constrain the value between the values “low” and “high” and return the value. This function belongs to the Micro_Rover class in the Freenove_Micro_Rover module.

Parameter:

Value: the value to be constrained

low: the minimum value of the constraint interval

high: the maximum value of the constraint interval

Return:

value after constrained

Example:

Constrain(-1,0,10), the return value is 0.

Constrain(5,0,10), the return value is 5.

Chapter 4 Light tracing

There are two light intensity sensors on Rover, so we can learn which sensor receives more intensive light based on the difference between the values of the two sensors. In this way, Rover can achieve the function of tracking or avoiding light.

Preparation

1. Insert micro:bit into Rover correctly.
2. Install battery into Rover.
3. Turn ON Rover power.
4. Connect micro:bit and computer through USB cable.

Get value of light intensity sensor

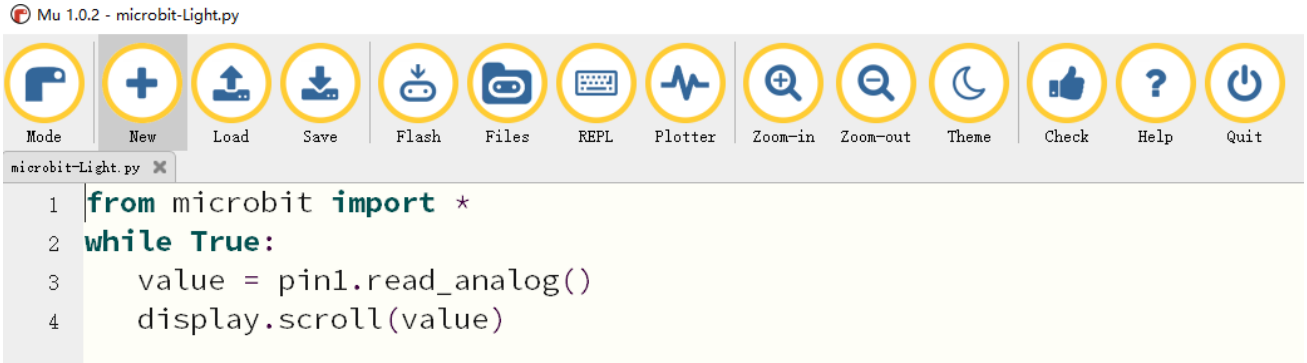
Get the value of the onboard light intensity sensor and display it on the LED matrix of the micro:bit.

Code

Open the "microbit-Light.py" with the Mu software. The path of the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	.. /PythonProjects / 04.1_Light	microbit-Light.py

After the loading completes, the interface of Mu is as shown below:



Download the code into micro:bit. Illuminate the light intensity sensor with a light source or cover it with your hand to observe the change of value on LED matrix of the micro:bit

The following is the program code:

```
1  from microbit import *
2  while True:
3      value = pin1.read_analog()
4      display.scroll(value)
```

Import everything in the microbit module, including functions, classes, variables, and so on. 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.

```
1  from microbit import *
```

The analog voltage value of the P1 pin is read in the while loop and displayed in the LED matrix. In the code, the variable value obtains the light intensity value measured by two sensors and combined them into one, and the range is 0-1023. In theory, if the value is equal to 512, then the two sensors receive the same intensity of light. If the value is greater than 512, the light intensity of the right sensor is larger than the left sensor. If the value is less than 512, the light intensity of the left sensor is larger than that of the right sensor. But in reality, it is affected by many factors. When the light intensity on both the left and the right is equal, the intensity value may not be 512, but a number close to 512. Therefore, we should calibrate when we use it.

```
while True:
    value = pin1.read_analog()
    display.scroll(value)
```

Reference

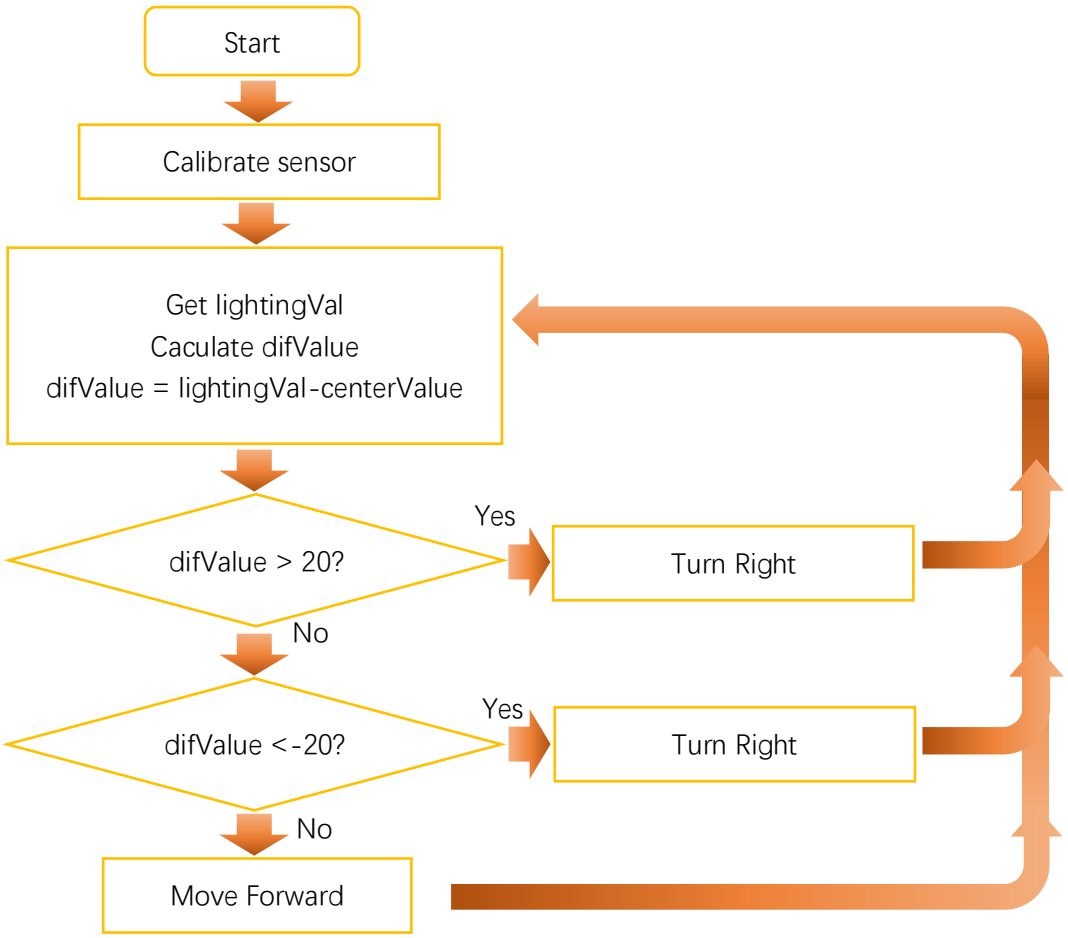
read_analog()
Read an analog signal (0 through 1023) from the pin you set.

Rover-light tracing mode

In this project, we will realize light tracing mode of Rover.

Flow chart

The program code is written according to flow chart, as shown below.

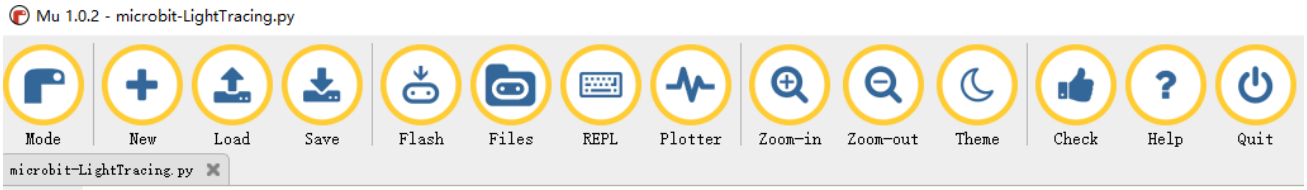


Code

Open the "microbit-LightTracing" with the Mu software. The path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects / 04.2_LightTracing	microbit-LightTracing.py

After the loading completes, the interface of Mu is as shown below:




```
1  from Freenove_Micro_Rover import *
2  Rover = Micro_Rover()
3  sumValue=0
4  for i in range(10):
5      sumValue += pin1.read_analog()
6  centerValue=round(sumValue/10)
7  display.show(Image.HAPPY)
8  while True:
9      LightingValue=pin1.read_analog()
10     R,G,B=Rover.hsl_to_rgb(Rover.map(LightingValue,0,1023,0,360))
11     Rover.all_led_show(255,R,G,B)
12     difValue=LightingValue-centerValue
13     if difValue > 20:
14         Rover.motor(150,0)
15     elif difValue < -20:
16         Rover.motor(0,150)
17     else:
18         Rover.motor(100,100)
```

Import the "Freenove_Micro_Rover.py" file into micro:bit (How to import files?), and download the code into micro:bit. Illuminate the light intensity sensor with a flashlight or other light source device to observe the movement of the car.

The following is the program code:

```
from Freenove_Micro_Rover import *
Rover = Micro_Rover()
sumValue=0
for i in range(10):
    sumValue += pin1.read_analog()
centerValue=round(sumValue/10)
display.show(Image.HAPPY)
while True:
    LightingValue=pin1.read_analog()
    R,G,B=Rover.hsl_to_rgb(Rover.map(LightingValue,0,1023,0,360))
    Rover.all_led_show(255,R,G,B)
    difValue=LightingValue-centerValue
    if difValue > 20:
        Rover.motor(150,0)
    elif difValue < -20:
        Rover.motor(0,150)
    else:
        Rover.motor(100,100)
```

Import the Freenove_Micro_Rover module.

```
from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
Rover = Micro_Rover()
```

As mentioned earlier, when using light sensors, we need to calibrate them first. Therefore, after starting, the sensor value is read 10 times, and then their sum is averaged as the center value after calibration.

```
sumValue=0
for i in range(10):
    sumValue += pin1.read_analog()
centerValue=round(sumValue/10)
```

Display a smile in the LED matrix

```
display.show(Image.HAPPY)
```

Read the value of the light sensor at the P1 pin in the while loop, and call the map function to map the value ranging from 0-1023 to 0-360. Then pass the converted value to the hsl_to_rgb () function, return the corresponding RGB value, and turn ON the LED on the car.

The value read is subtracted from the center value. If the difference is greater than 20, the light received by the right side is more intensive than the left side, and the car turns right. If the difference is less than -20, the light received by the left side is more intensive than the right side, and the car turns left. If the difference is between -20-20, the difference of light intensity between the left and right sides is so small that can be ignore, and the car goes straight.

```
while True:
    LightingValue=pin1.read_analog()
    R,G,B=Rover.hsl_to_rgb(Rover.map(LightingValue,0,1023,0,360))
    Rover.all_led_show(255,R,G,B)
    difValue=LightingValue-centerValue
    if difValue > 20:
        Rover.motor(150,0)
    elif difValue < -20:
        Rover.motor(0,150)
    else:
        Rover.motor(100,100)
```

Reference

`display.show(image)`

Display the image.

`map(value,fromLow,fromHigh,toLow,toHigh)`

Description:

Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

Parameters:

value: the number to map.
fromLow: the lower bound of the value's current range.
fromHigh: the upper bound of the value's current range.
toLow: the lower bound of the value's target range.
toHigh: the upper bound of the value's target range.

Returns:

The mapped value.

Example:

Map(5,0,10,0,50), return 25.

Chapter 5 Line tracking

There are three Reflective Optical Sensors on Rover. When the infrared light emitted by infrared diode shines on the surface of different objects, the sensor will receive light with different intensities after reflection.
As we know, black objects absorb light better. So when black lines are drawn on the white plane, the sensor can detect the difference. In this way, we can realize Line tracking mode for Rover. The sensor can also be called Line Tracking Sensor.

Preparation

1. Insert micro:bit into Rover correctly.
2. Install battery into Rover.
3. Turn ON Rover power.
4. Connect micro:bit and computer through USB cable.

Warning

Reflective Optical Sensor(including Line Tracking Sensor) should be avoided using in environment with infrared interference, like sunlight, which contains a lot of invisible light such as infrared and ultraviolet. Under environment with intense sunlight, Reflective Optical Sensor cannot work normally.

Get value of LineTrackingSensor

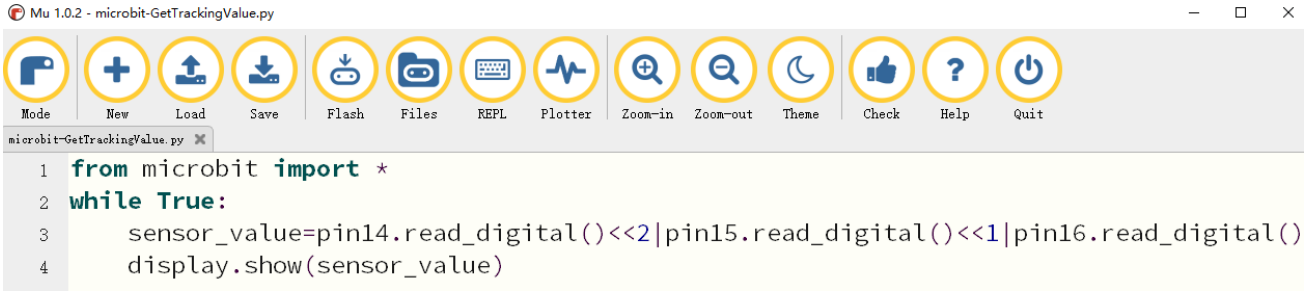
Obtain the value of the Reflective Optical Sensor and display the LED matrix on the micro:bit.

Code

Open the "microbit-GetTrackingValue.py" with the Mu software. The path of the code is as follows: (How to load the project code?)

File type	Path	File name
Python file	.. /PythonProjects / 05.1_GetTrackingValue	microbit-GetTrackingValue.py

After the loading completes, the interface of Mu is as shown below:



Import the “Freenove_Micro_Rover.py” file into micro:bit (How to import files?) and download the code into micro:bit. Use black and white objects to block the sensor (the shielding distance is better in 2-3cm), observe the LED matrix of micro:bit and indicator LED on the car.

The following table shows the values of all cases when three Tracking Sensors detect objects of different colors. Among them, 1 represents black objects or no objects are detected., and 0 represents white objects are detected.

Left	Middle	Right	Value(binary)	Value(decimal)
0	0	0	000	0
0	0	1	001	1
0	1	0	010	2
0	1	1	011	3
1	0	0	100	4
1	0	1	101	5
1	1	0	110	6
1	1	1	111	7

You can verify the running result of Rover with this table.

The following is the program code:

```
1  from microbit import *
2  while True:
3      sensor_value=pin14.read_digital()<<2|pin15.read_digital()<<1|pin16.read_digital()
4      display.show(sensor_value)
```

Import microbit module.

```
from microbit import *
```

Read the values of the three sensors in the while loop. Returns high when the sensors (P14, P15, P16 pins) detect black or no object. Otherwise, returns low level. And the acquired values will be displayed on the LED matrix.

```
while True:
    sensor_value=pin14.read_digital()<<2|pin15.read_digital()<<1|pin16.read_digital()
    display.show(sensor_value)
```

Reference

read_digital()

Return 1 if the pin is high, and 0 if it's low.

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

| operator

| is a bitwise OR operation that performs an OR operation in binary bits. Operation rules:

0 | 0=0;

0 | 1=1;

1 | 0=1;

1 | 1=1

For example:

A=0011 1100 (0x3c)

B=0011 0101 (0x35)

A | B=0011 1101 (0x3d)

Rover-light tracing mode

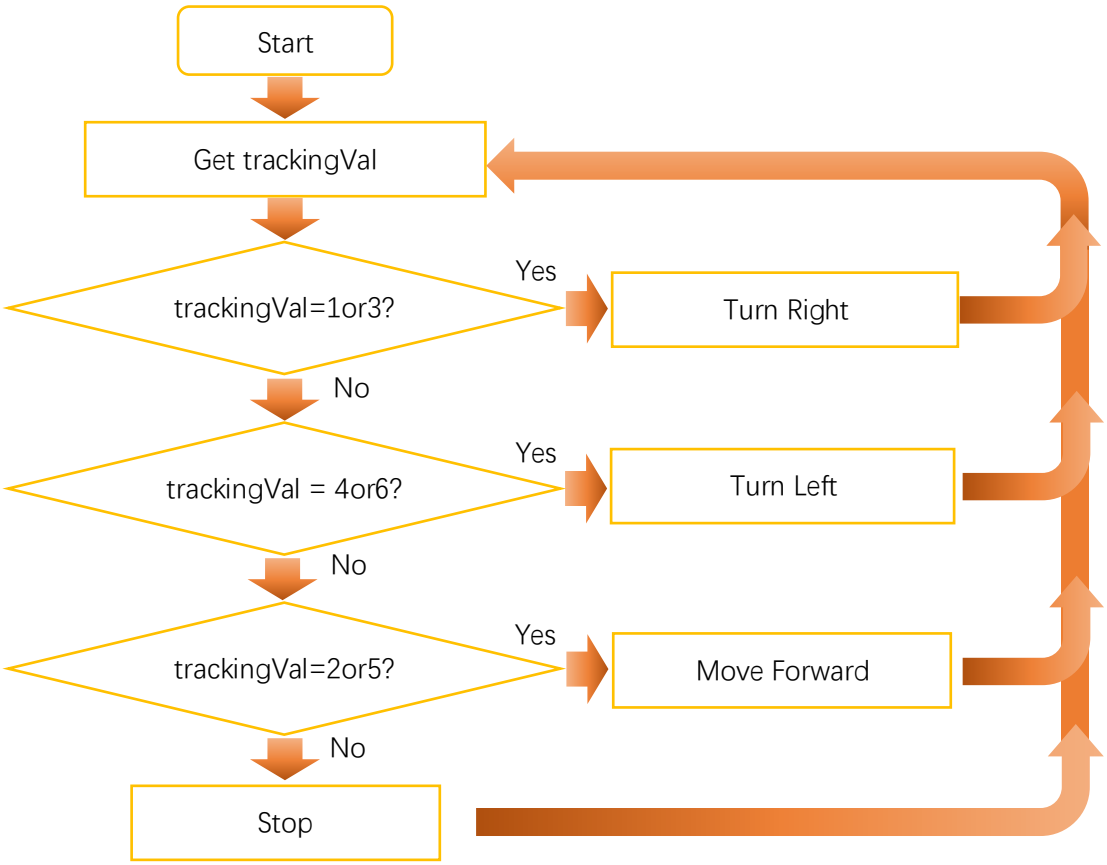
In this project, we will realize line tracking mode of Rover.

Flow chart

Rover will make different actions according to the value transmitted by the line tracking sensor.

Left	Middle	Right	Value(binary)	Value(decimal)	Rover Action
0	0	0	000	0	Stop
0	0	1	001	1	Turn Right
0	1	0	010	2	Move Forward
0	1	1	011	3	Turn Right
1	0	0	100	4	Turn Left
1	0	1	101	5	Move Forward
1	1	0	110	6	Turn Left
1	1	1	111	7	Stop

Flow chart is as below:

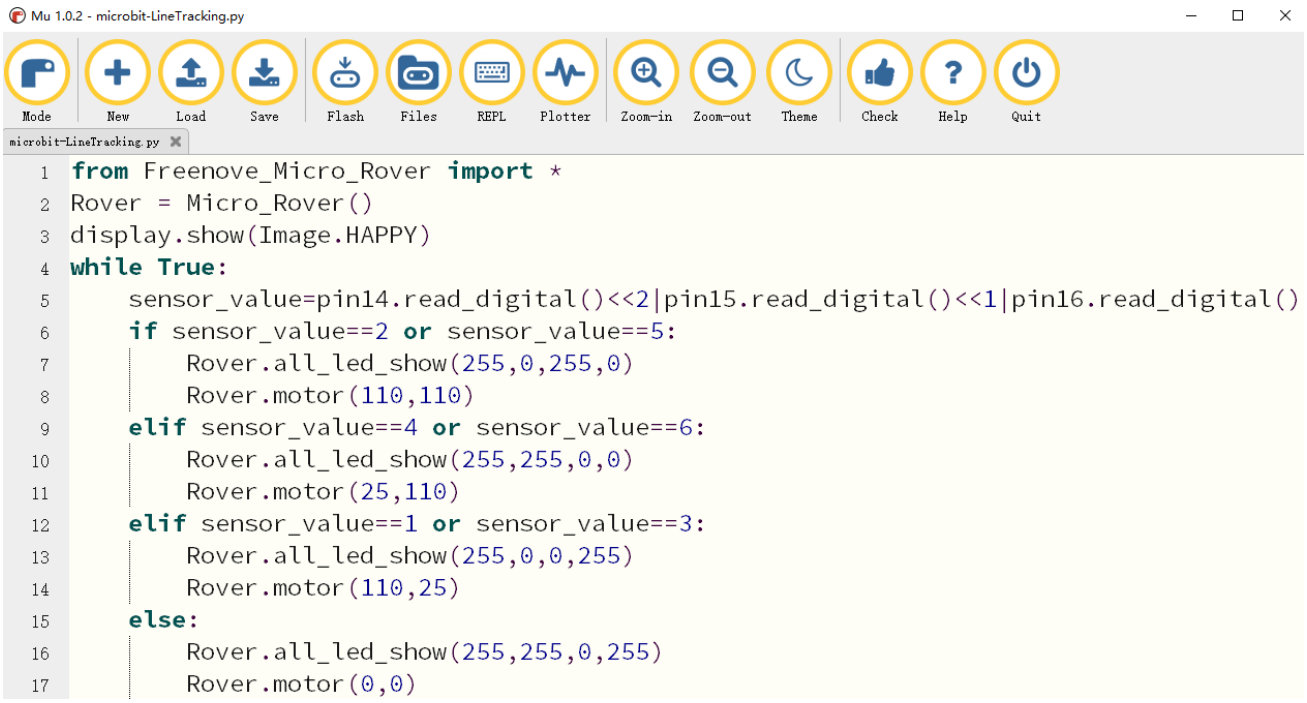


Code

Open microbit-LineTracking.py with Mu software, the path of the code is as follows:

File type	Path	File name
Python file	.. /PythonProjects / 05.2_LineTracking	microbit-LineTracking.py

After the loading completes, the interface of Mu is as shown below::




```
from Freenove_Micro_Rover import *
Rover = Micro_Rover()
display.show(Image.HAPPY)
while True:
    sensor_value=pin14.read_digital()<<2|pin15.read_digital()<<1|pin16.read_digital()
    if sensor_value==2 or sensor_value==5:
        Rover.all_led_show(255,0,255,0)
        Rover.motor(110,110)
    elif sensor_value==4 or sensor_value==6:
        Rover.all_led_show(255,255,0,0)
        Rover.motor(25,110)
    elif sensor_value==1 or sensor_value==3:
        Rover.all_led_show(255,0,0,255)
        Rover.motor(110,25)
    else:
        Rover.all_led_show(255,255,0,255)
        Rover.motor(0,0)
```

Import the “Freenove_Micro_Rover.py” file into micro:bit(How to import files?) and download the code into micro:bit. Place the car on the map we provide and observe the movement of the car.

The following is the program code:

```
from Freenove_Micro_Rover import *
Rover = Micro_Rover()
display.show(Image.HAPPY)
while True:
    sensor_value=pin14.read_digital()<<2|pin15.read_digital()<<1|pin16.read_digital()
    if sensor_value==2 or sensor_value==5:
        Rover.all_led_show(255,0,255,0)
        Rover.motor(110,110)
    elif sensor_value==4 or sensor_value==6:
        Rover.all_led_show(255,255,0,0)
        Rover.motor(25,110)
    elif sensor_value==1 or sensor_value==3:
        Rover.all_led_show(255,0,0,255)
        Rover.motor(110,25)
    else:
        Rover.all_led_show(255,255,0,255)
        Rover.motor(0,0)
```

Import the Freenove_Micro_Rover module.

```
from Freenove_Micro_Rover import *
```

Create an object of the Micro_Rover class in the Freenove_Micro_Rover module.

```
Rover = Micro_Rover()
```

Display a smile on LED matrix.

```
display.show(Image.HAPPY)
```

Read the values of the three sensors in the while loop. Returns high when the sensors (P14, P15, P16 pins) detect black or no object. Otherwise, returns low level. Assign the integrated value to variable “sensor_value”.

```
while True:
    sensor_value=pin14.read_digital()<<2|pin15.read_digital()<<1|pin16.read_digital()
```

In the code, the value of l_value is determined.
If l_value is equal to 2 or 5, the car moves forward while the LEDs emitting bright green.
If l_value is equal to 4 or 6, the car turns left and the LEDs emit red.
If l_value is equal to 1 or 3, the car turns right and the LEDs emit blue.
If l_value is equal to 0 or 7, the car stops and the LEDs emit purple.

```
if l_value==2 or l_value==5:
    Rover.all_led_show(255,0,255,0)
    Rover.motor(110,110)
elif l_value==4 or l_value==6:
    Rover.all_led_show(255,255,0,0)
    Rover.motor(25,110)
elif l_value==1 or l_value==3:
    Rover.all_led_show(255,0,0,255)
    Rover.motor(110,25)
else:
    Rover.all_led_show(255,255,0,255)
    Rover.motor(0,0)
```

Chapter 6 Bluetooth

Though the BBC micro:bit has hardware capable of allowing the device to work as a Bluetooth Low Energy (BLE) device, it only has 16k of RAM. The BLE stack alone takes up 12k RAM which means there's not enough memory for MicroPython to support Bluetooth.
https://microbit-micropython.readthedocs.io/en/latest/ble.html

Next

Here, all projects that do not require additional components have been completed.

We have also prepared an additional free tutorial. It contains several projects. But they need additional components or modules that are not included in the Rover kit. If you are interested in it, you can find it here:
View: https://github.com/Freenove/Freenove_Micro_Rover_Extended_Projects
Download: https://github.com/Freenove/Freenove_Micro_Rover_Extended_Projects/archive/master.zip

If you do not have these components, you will not be able to complete the next projects.

If you're not interested in extra projects. Never mind. Just leave them alone.

Enjoy your Rover.

Appendix

Following are details of functions in "Freenove_Micro_Rover.py" file.

Function	Description
constrain(Value,Low,High)	Used to constrain a value within an interval.
all_led_show(Brightness,R,G,B)	Light up all the LEDs on the body, and you can set the brightness and color.
led_show(Index, Brightness ,R,G,B)	Light up the specified LED on the car, and you can set the brightness and color
hsl_to_rgb(Degree)	Converts the color value of the corresponding hue to an RGB value.
motor(Left,Right)	Control the left and right motors. Positive integer makes motor move forward, negative integer makes motor move back. The larger the absolute value of the number, the faster the speed.
get_distance()	Obtain the distance measurement value of the ultrasonic module.

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 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.