

Table of Contents

1 Overview	2
1.1 What's in the Package:	3
1.2 Specification	3
1.3 Interface Introduction	4
2 Assemble the Joystick	6
Step 1 - Assemble the Acrylic Extension Handle	6
Step 2 - Install the batteries	7
Step 3 - Install the bottom acrylic battery cover	8
Step 4 - Attach the button caps	9
Step 5 - Turn off the power switch	9
Step 6 - Install the micro:bit	10
4.micro:bit Python Editor Quick Start	10
4.1 Create a new project	11
4.2 Save a project	12
4.3 Import Files	14
4.4 Upload code	15
4.5 Learning Basic Syntax of the micro:bit Python Editor	20
5. Basic Example Projects	21
5.1 Button	22
5.2 Joystick	23
5.3 Vibration motor	25
5.4 Battery voltage	26
5.5 Servo	28
5.6 Wireless control	31
6. Extended Projects	35
6.1 Control mCar	36
6.2 Control Pybit	42
6.3 Joystick Control Robot Arm	49
7. QA	50
7.1 Micro:bit Code Upload Failure	50
7.2 Abnormal Micro:bit Drive Presentation	50
7.3 Buttons or Joystick Not Responding	50
7.4 Other Issues	50
8.Contact Us	51

1. Overview

This expansion board is an interactive input peripheral designed for the BBC micro:bit. It integrates a dual-axis joystick, multiple function buttons, and a vibration feedback module, making it suitable for applications such as game controllers, robot remote control, interactive teaching, and more.

Features:

Dual-Axis Analog Joystick: Provides analog input in both X and Y axes, enabling real-time position detection.

Multi-Function Buttons: Includes four independent digital buttons (A/B/C/D) in addition to the joystick's built-in button, all programmable for custom actions.

Vibration Motor: Equipped with a 3610-type vibration motor for haptic feedback, triggered via program control.

Expansion Interfaces: Offers multiple digital I/O pins, servo control pins, and SPI communication support for connecting additional sensors or actuators.

Independent Power Supply: Can be powered by 4 AA batteries (nominal 6V), featuring a dedicated power switch—no need to rely on the micro:bit's onboard power.

Typical Applications:

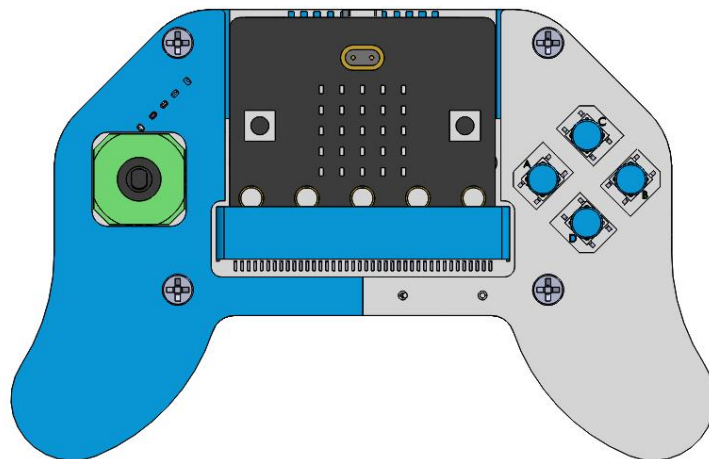
Game controller development

Robotic motion control

STEM education projects

Interactive art installations

Physical feedback simulation experiments

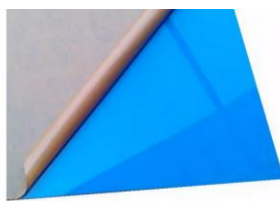


The Micro:bit must be purchased separately!

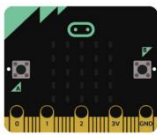



1.1 What's in the Package:

Joystick 1Pcs	3x18mm nylon standoffs 4Pcs	3x9mm screws 8Pcs
		
Acrylic plates 3Pcs	Screwdriver 1Pcs	Button caps 4Pcs
		

Note: The acrylic pieces are white and blue. Please remove the protective film covering their surfaces.



What you'll need (not included)

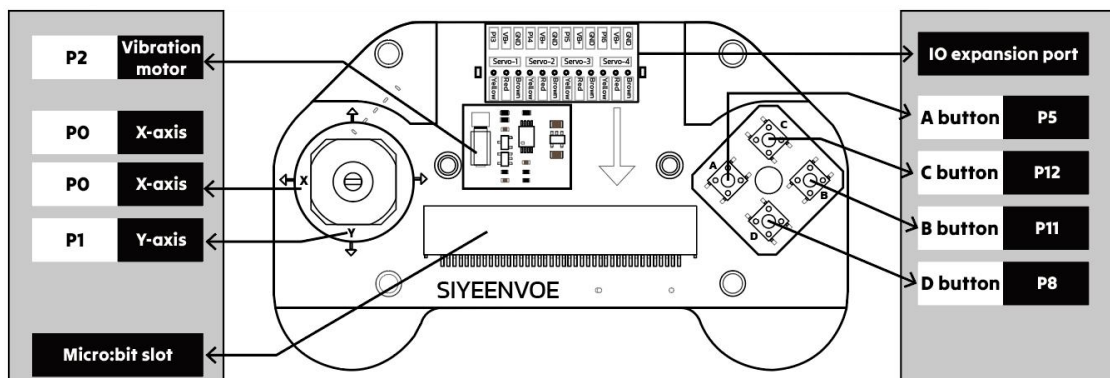
			
Micro:bit v2 1PCS	Micro USB Cable	computer with internet & a USB port	1.5V AA Battery 4 pcs

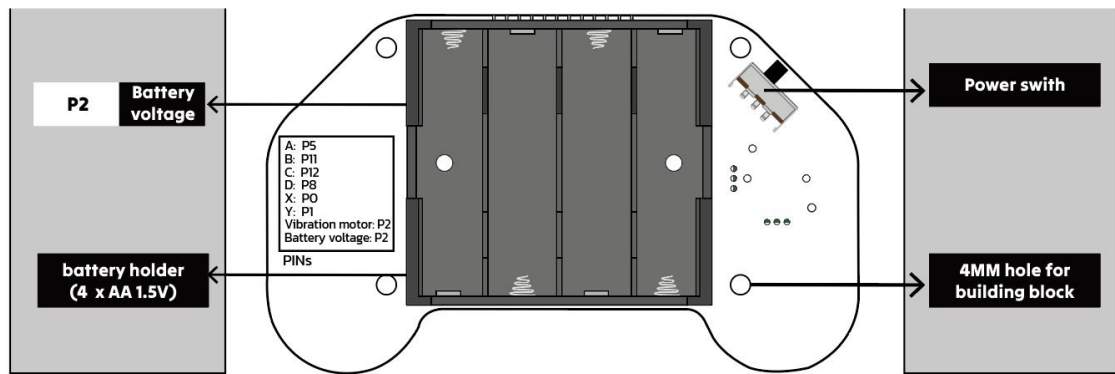
1.2 Specification

Shield	
Name	Joystick
SKU	M1K0000

Applicable motherboard	
Micro:bit	
Pins	
Digital I/O Pins	4 (P13, P14, P15, P16)
Servo pins	4 (P13, P14, P15, P16)
Communication	
SPI	Yes (P13, P14, P15, P16)
Power	
Input voltage (nominal)	6V (4 AA batteries, 1.5V/PCS)
VB(+/-)	6V (4 AA batteries, 1.5V/PCS)
Vibration motor	
Model	3610 Vibration motor
Rated voltage/current	2.7V/75mA
Rotation speed	14000±2500RPM
Dimensions	
Width	72.5 mm
Length	120 mm
Height	29mm
Weight	
PCBA	50 g
With Acrylic	82 g

1.3 Interface Introduction

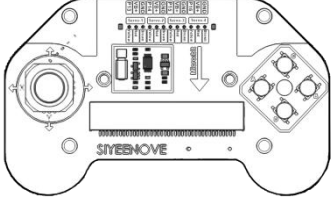




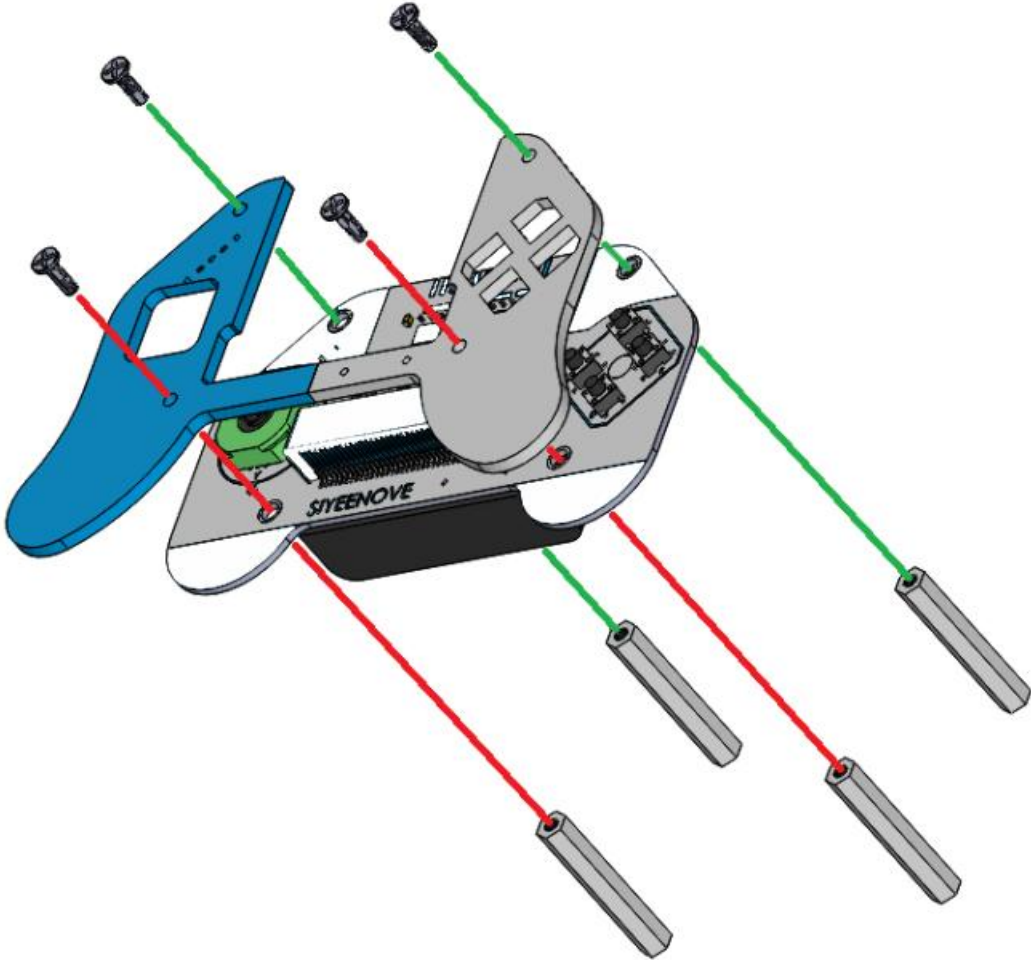


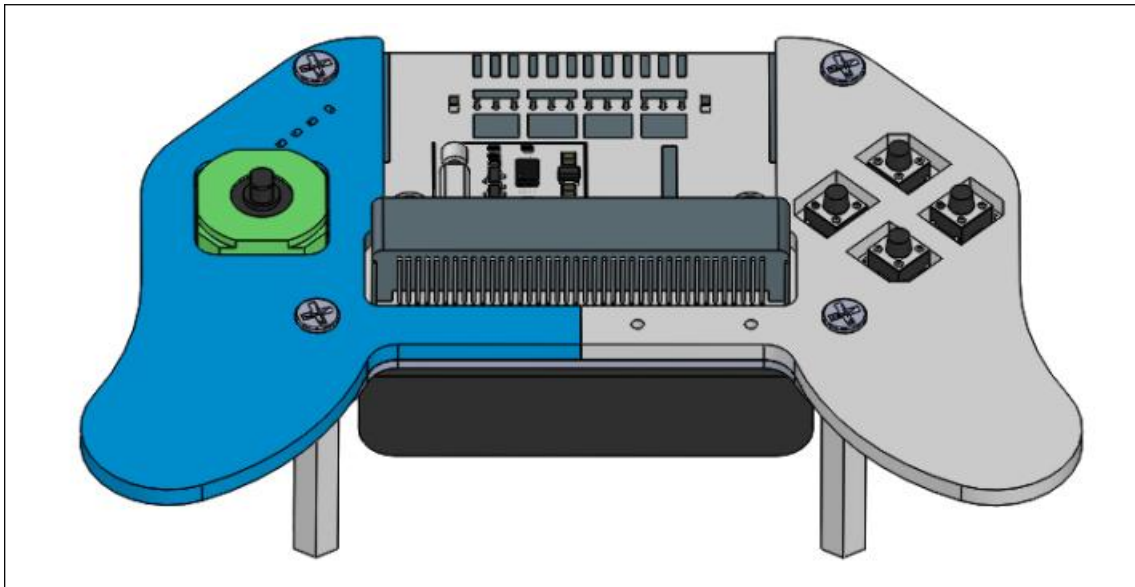


Name	Description
Vibration motor	Provides haptic feedback to the handle, controlled by Micro:bit's
X-axis	The joystick's X-axis, whose analog value can be read through Micro:bit's P0 pin
Y-axis	The joystick's Y-axis, whose analog value can be read through Micro:bit's P1 pin
Micro:bit slot	Designed for inserting the Micro:bit main control board
4 x AA 1.5V battery case	The battery case accommodates 4 batteries, each with standard 1.5V voltage, approximately 14.5mm in diameter and 50.5mm in height
Battery voltage	When 4 AA batteries are installed, the voltage can be read through Micro:bit's P2 pin
IO extension	For connecting external servos or sensors. The VB pin voltage equals the series voltage of 4 AA batteries ($1.5V \times 4 = 6V$)
A button	Its value can be read through Micro:bit's P5 pin
B button	Its value can be read through Micro:bit's P12 pin
C button	Its value can be read through Micro:bit's P11 pin
D button	Its value can be read through Micro:bit's P8 pin
Power switch	It controls the main power supply of the handle.
4MM hole	The handle features 4x4mm holes compatible with LEGO building block for expansion

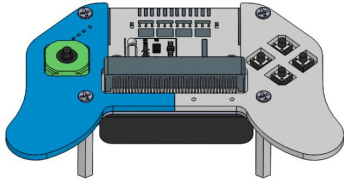

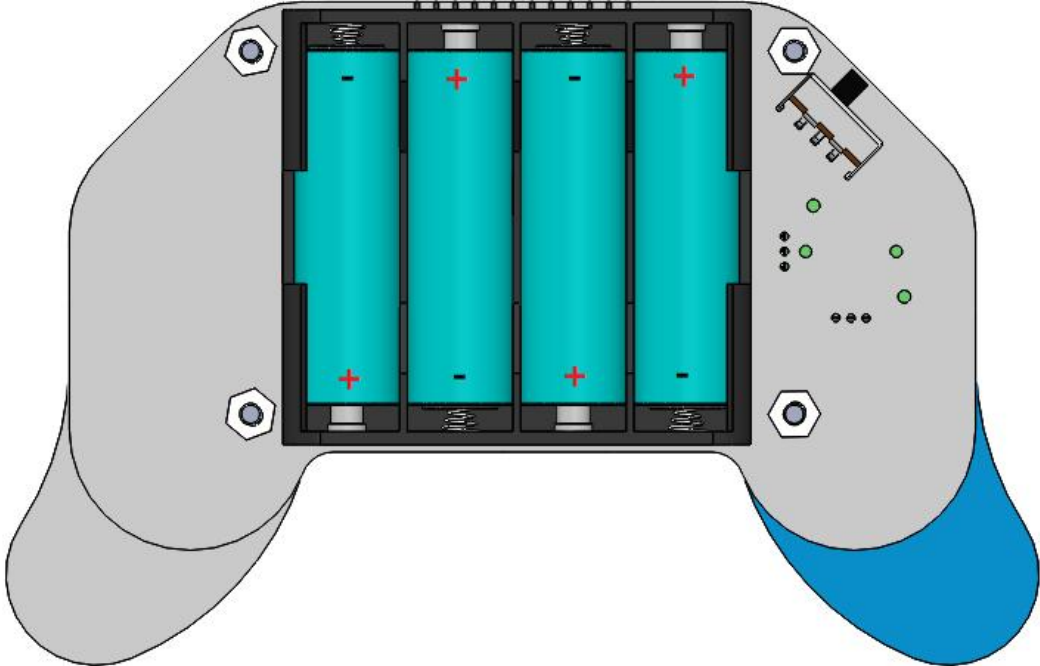
2 Assemble the Joystick

Step 1 - Assemble the Acrylic Extension Handle

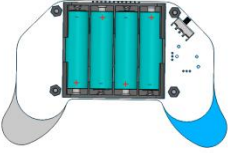


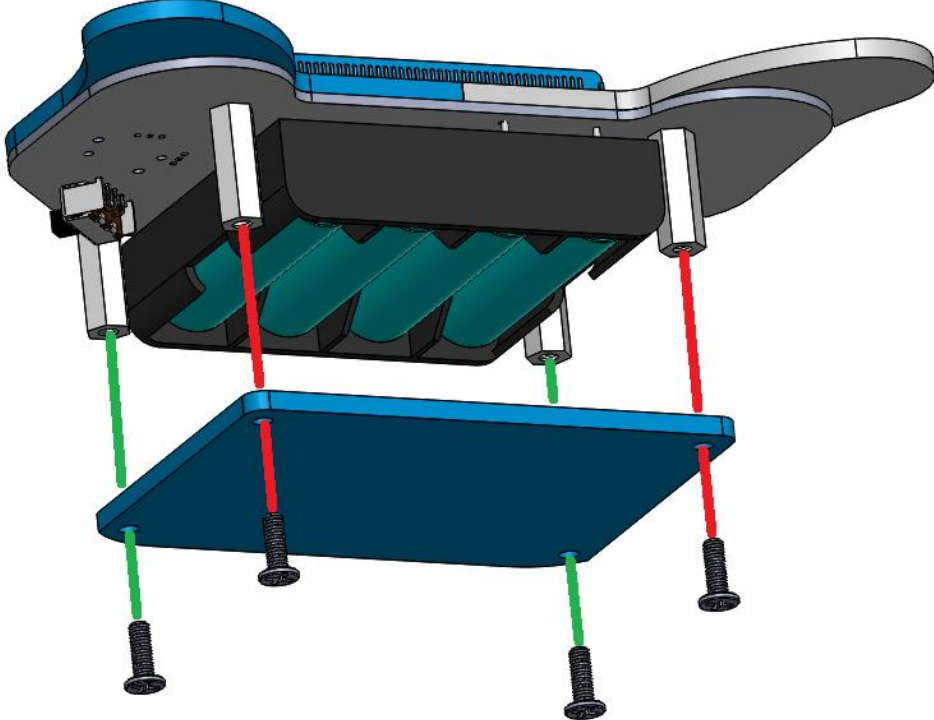
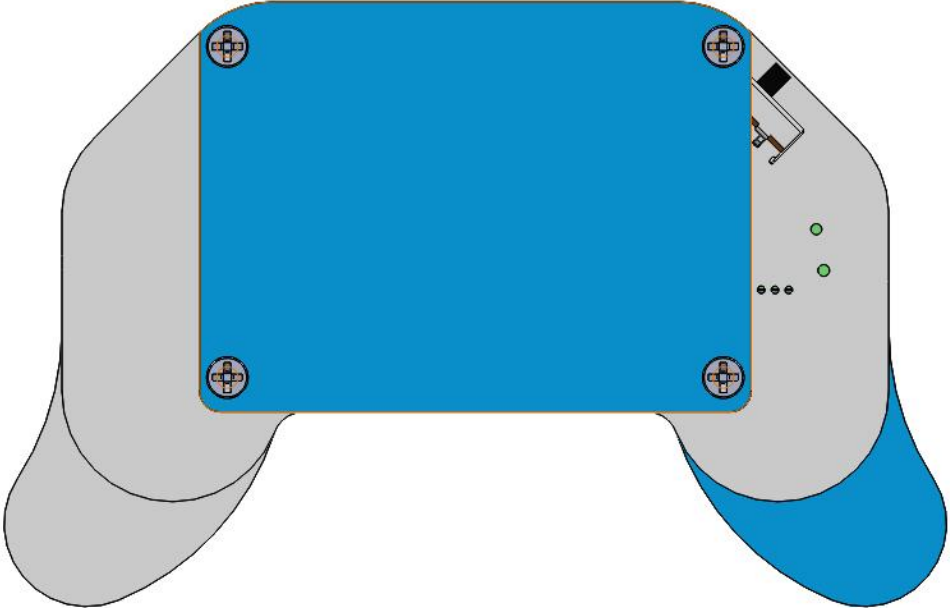
Joystick × 1	Acrylic plate × 1	Acrylic plate × 1
		
3×18mm nylon standoffs × 4	3×9mm screws × 4	Note
		Please remove the protective film covering the surface of the acrylic plates.
		



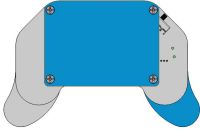
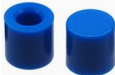
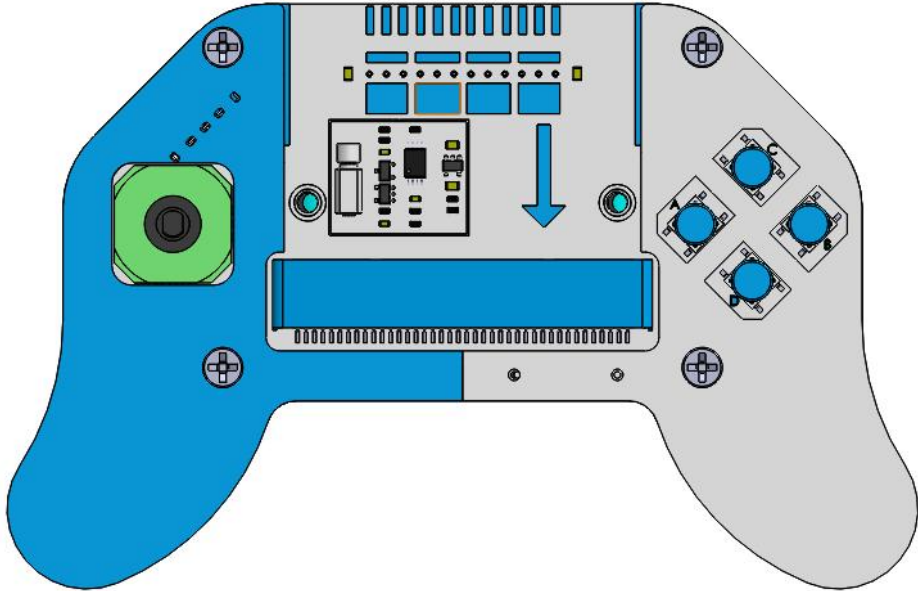
Step 2 – Install the batteries

Components from Step 1	AA batteries × 4
	
	

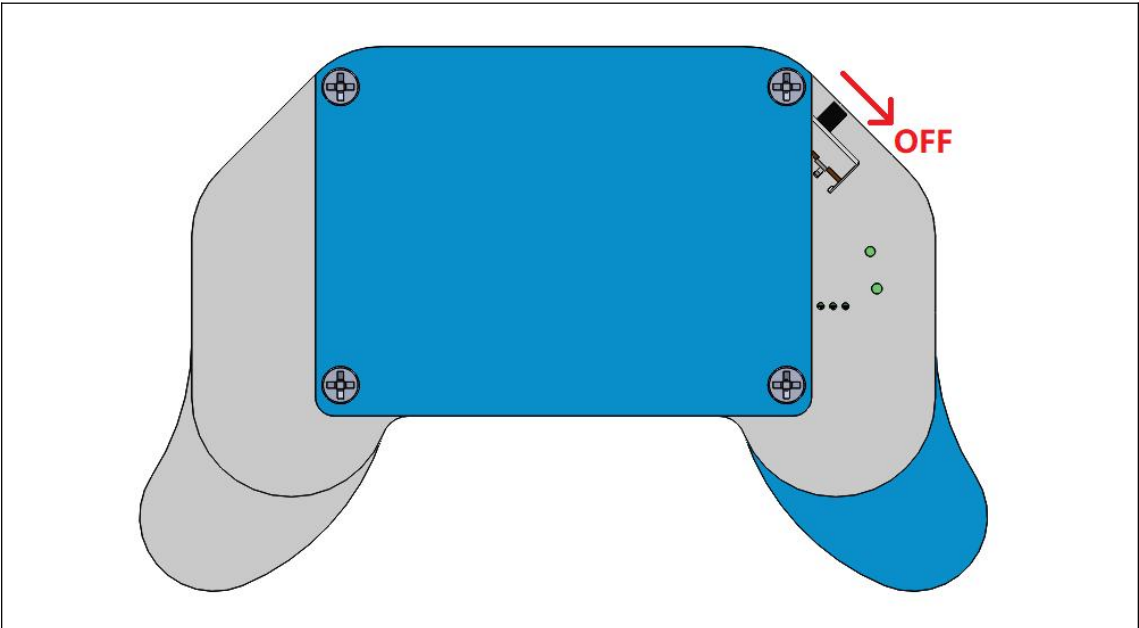
Step 3 – Install the bottom acrylic battery cover

Components from Step 2	Acrylic plate × 1	Screws × 4
		
		
		

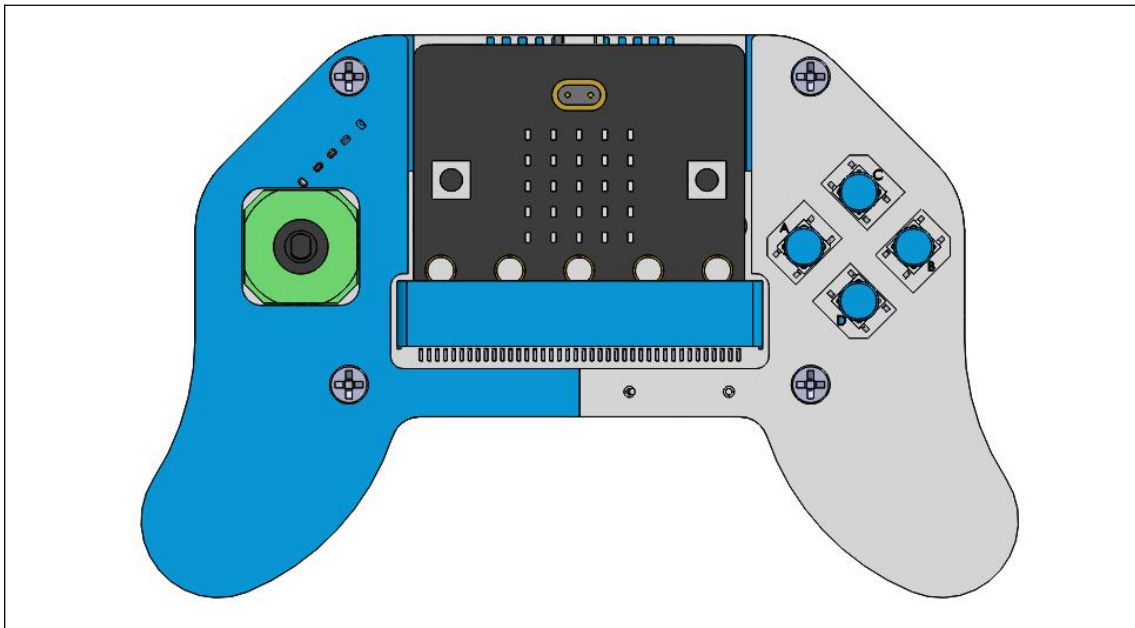
Step 4 – Attach the button caps

Components from Step 3	Button caps × 4
	
	

Step 5 – Turn off the power switch



Step 6 – Install the micro:bit



3. micro:bit Python Editor Quick Start

The [micro:bit Python Editor](#) is the perfect way to get started with MicroPython programming on the BBC micro:bit. It is free and works in browsers across all platforms.

We recommend using Chrome or Edge browsers. WebUSB is a recent and developing web feature that allows you to access a micro:bit directly from a web page. It also lets you directly receive data into the micro:bit Python editor from the micro:bit. It works in Google Chrome and Microsoft Edge browsers.

WebUSB support for your micro:bit

If you're not using a current version of the Chrome or Microsoft Edge browsers, make sure they are this version or newer:

Chrome (version 79 and newer) browser for Android, Chrome OS, Linux, macOS and Windows 10.

Microsoft Edge (version 79 and newer) browser for Android, Chrome OS, Linux, macOS and Windows 10.

Link to download the latest Google Chrome:

<https://www.google.com/chrome/>

Link to download the latest Microsoft Edge:

<https://www.microsoft.com/en-us/edge/download>

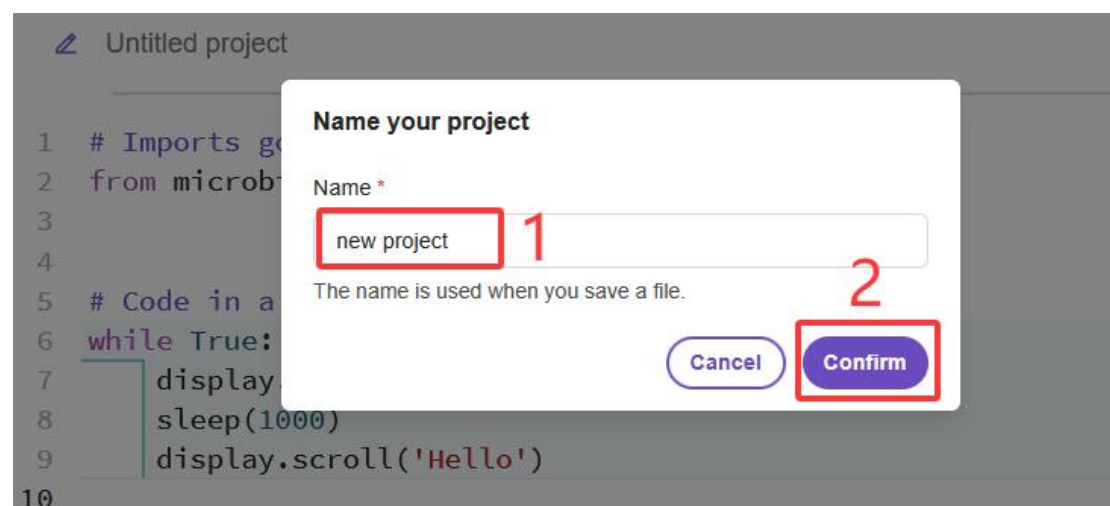
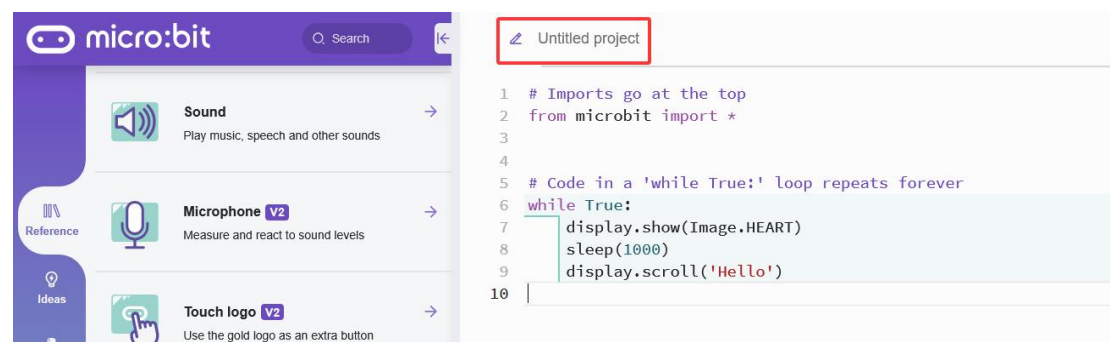
3.1 Create a new project

Open the micro:bit Python Editor on your browser:

<https://python.microbit.org/v/3>.

Once the editor loads, you will see a code editing area containing a simple starter template. Click on “[Untitled project](#)” at the top of the editor, then enter a meaningful name for your project (e.g., new project).

You can edit the existing template or replace it entirely with your own Python code.

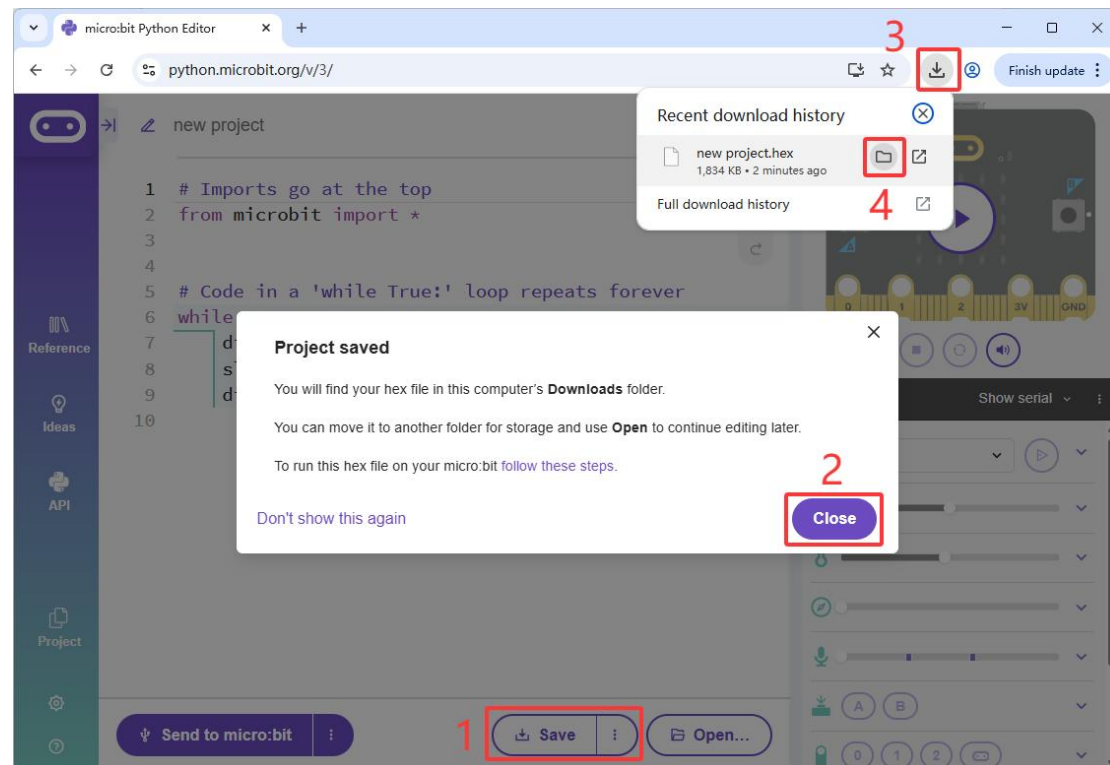


Note: The micro:bit Python Editor operates on a single-project model, meaning only one project can be edit at a time. It does not support multi-tab browsing, side-by-side editing of multiple files, or built-in project switching.

3.2 Save a project

1. Save Project as hex file

Click the **"Save"** button, the editor automatically saves your work in your browser's local storage. A downloaded hex file will be found in your browser's default download folder.



This is the final file format that the micro:bit can understand and execute.

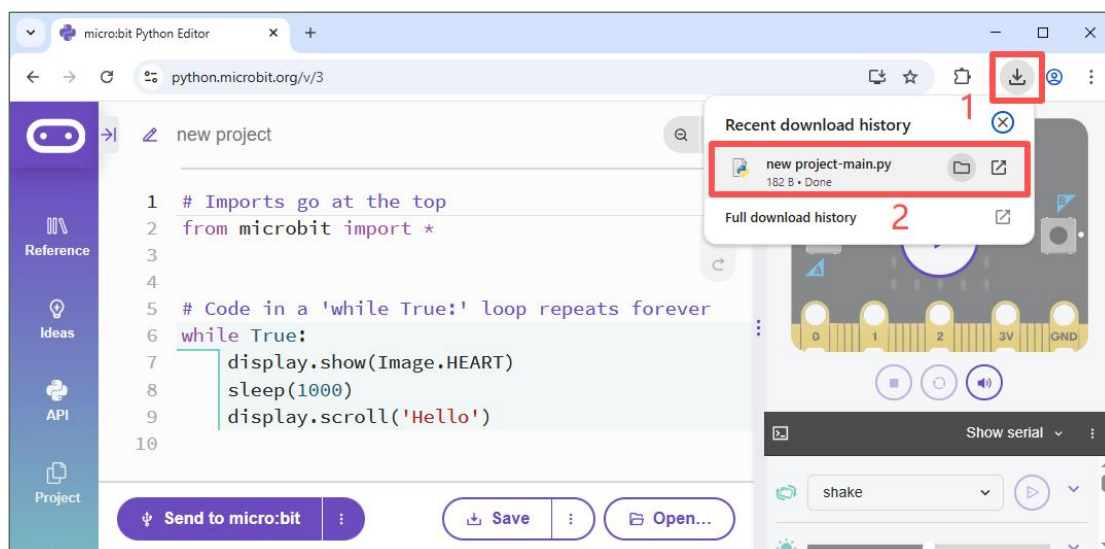
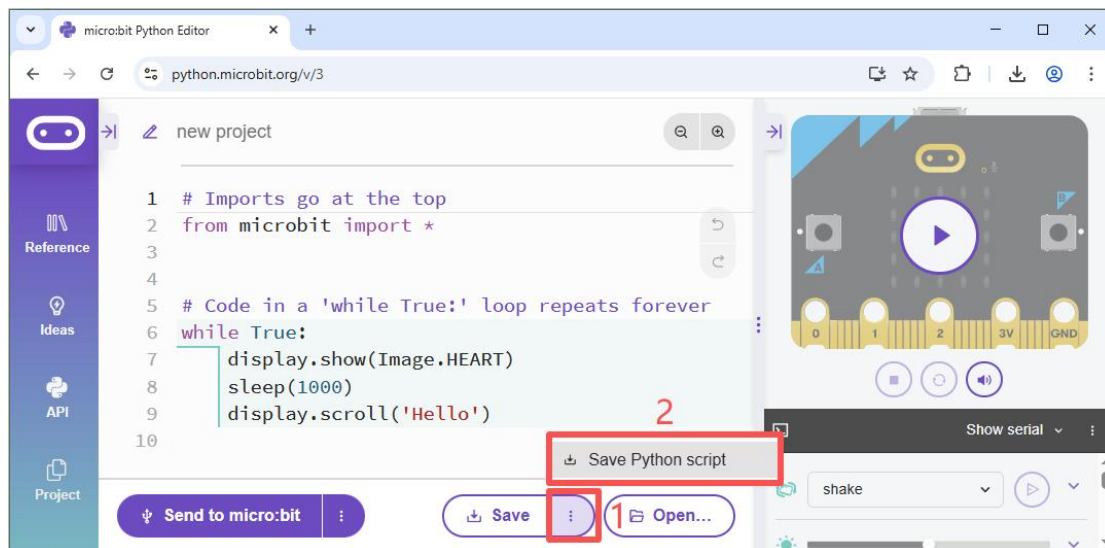
Functions:

Upload to micro:bit: Drag and drop the .hex file into the micro:bit's USB drive (or send it), and the micro:bit will load it into memory and start execution.

Project Backup and Sharing: You can save the .hex file to back up an immediately runnable version of the project or share it with others, who can use it directly without needing to access the source code.

2. Save Project as .py File

Click the three dots next to the "Save" button, then click the "Save Python script" button, the .py file will be saved to your computer's downloads folder.



This is the written Python code text.

Functions:

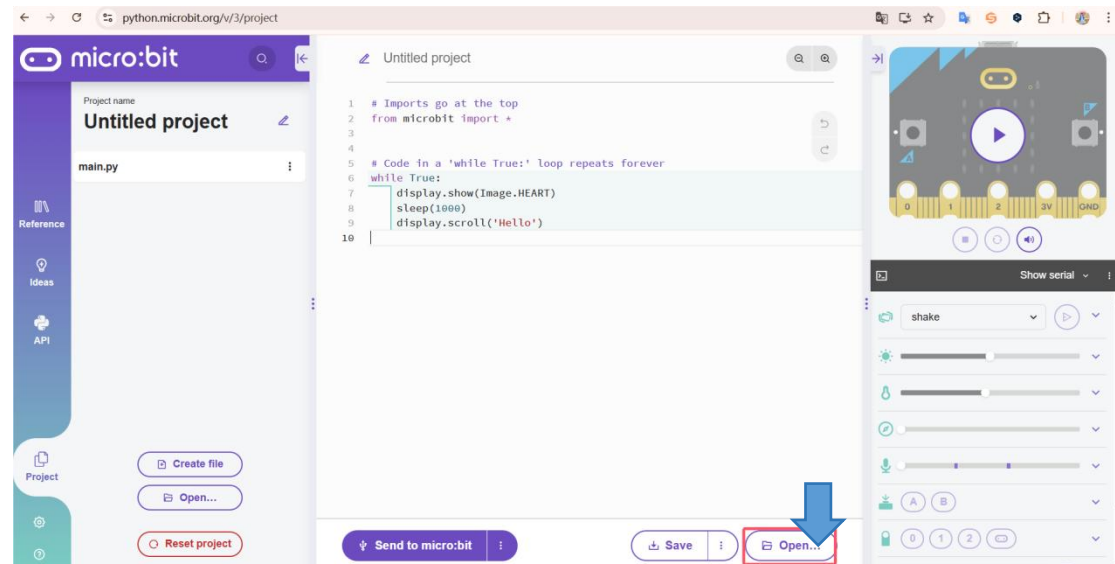
Editing and Creation: Write, modify, and debug your program in the micro:bit Python Editor or any text editor.

Remember: The micro:bit runs .hex files, but you develop in .py files.

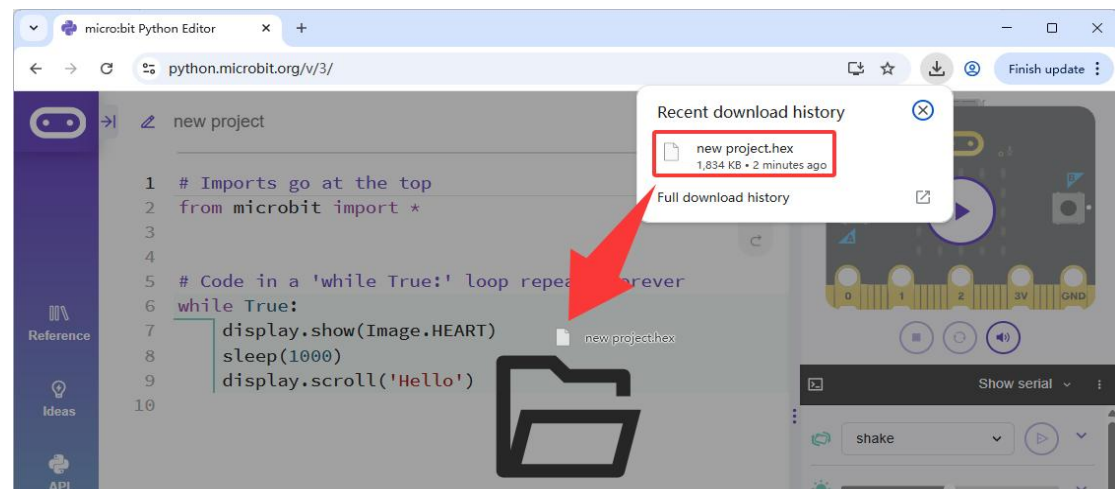
3.3 Import Files

To open a program (e.g. one you saved earlier or one provided by a teacher or third party), choose the 'Open' button.

From here, select the .hex or .py file that you wish to open and then click open.



Alternatively, you can drag and drop a .hex file or .py file into the editor.

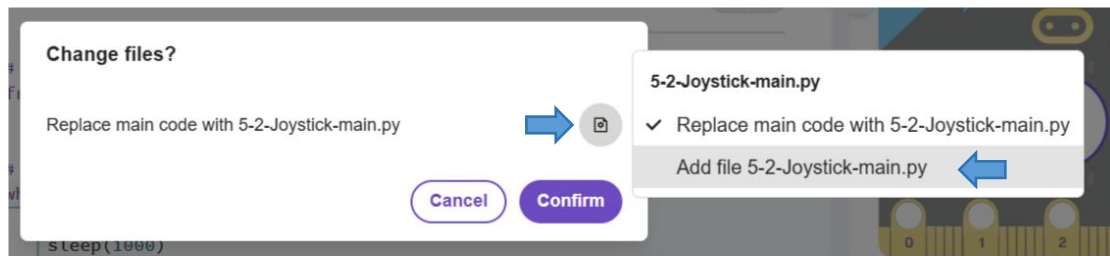


When importing files:

Opening a **new .hex file** will replace the current project immediately without any **warning**, and this action is irreversible. **Be sure to save your work beforehand to avoid losing code.**

Opening a **new .py file** will trigger a prompt: "**Change files?**" You can then choose to:


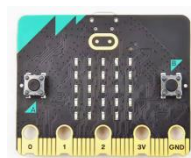

- Confirm** to replace the current code,
- Cancel** to keep the current project, or
- Add** to insert the new file as an **extension** to the existing code.



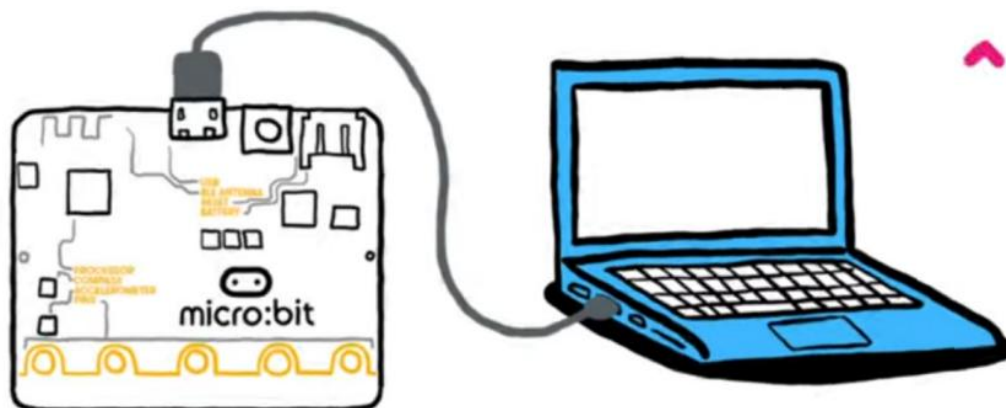
Note: The micro:bit Python Editor operates on a single-project model, meaning only one project can be open and active at a time. It does not support multi-tab browsing, side-by-side editing of multiple files, or built-in project switching. Loading a new project will replace the current project. This streamlined, focused approach simplifies the interface for beginners. It reduces cognitive load by letting students focus solely on the current task, avoiding confusion from switching between multiple projects and aligning with the "one thing at a time" teaching philosophy.

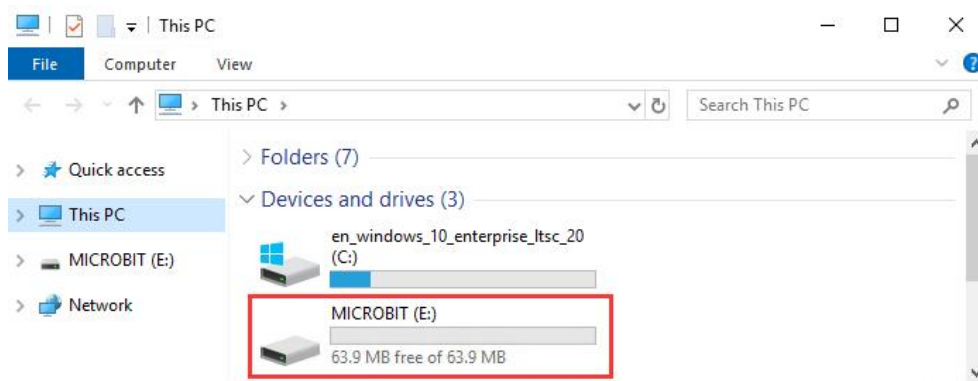
3.4 Upload code

Things you need:

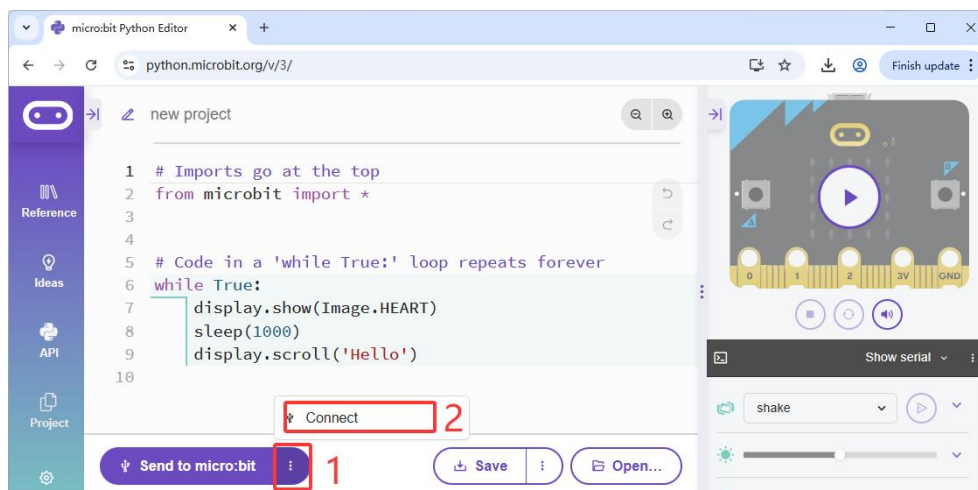
PC	Micro:bit v2.x.x	Micro USB Cable
		

Use a Micro USB cable to connect the micro:bit board to the PC. You will find a new USB disk called MICROBIT on the PC:

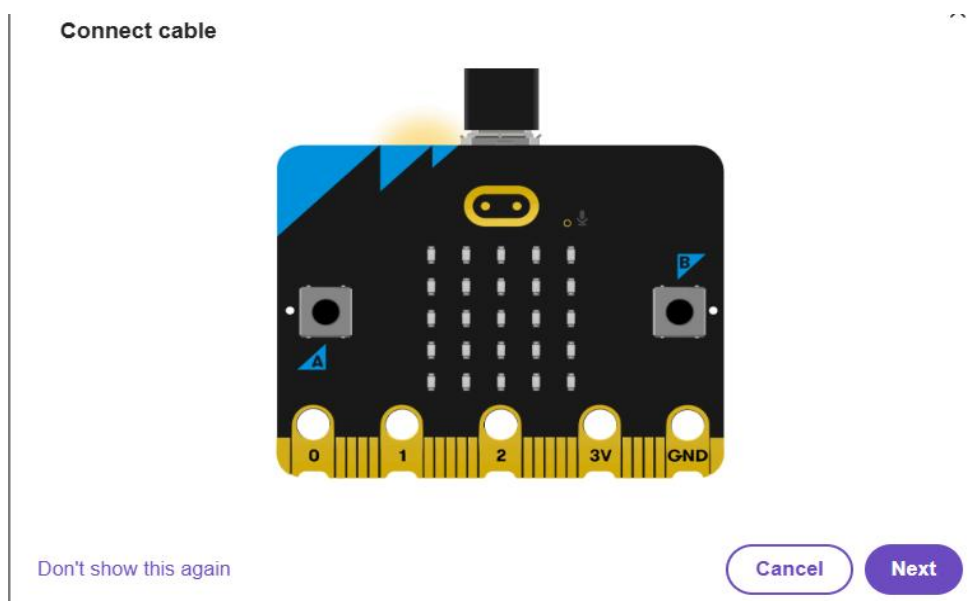


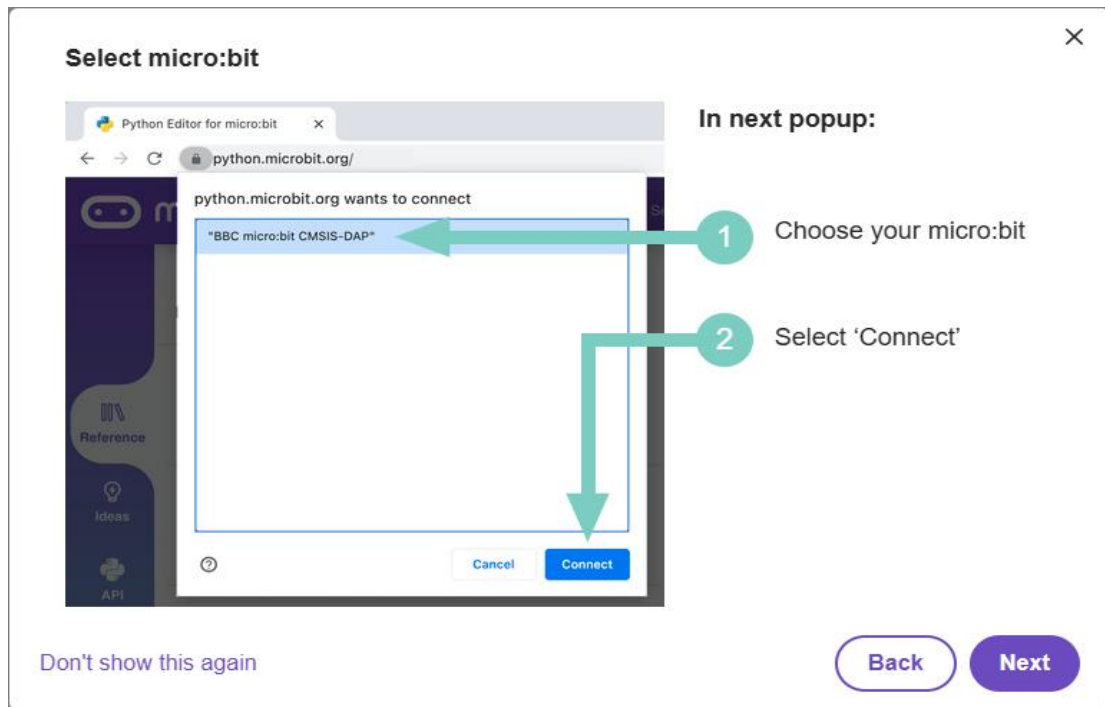


Direct Flashing (WebUSB): If you are using a compatible browser like Google Chrome or Microsoft Edge, you can use WebUSB to connect your micro:bit directly to the editor and flash the program without dragging and dropping files. Click the three dots next to the **"Send to micro:bit"** button, then click the **"Connect"** button:

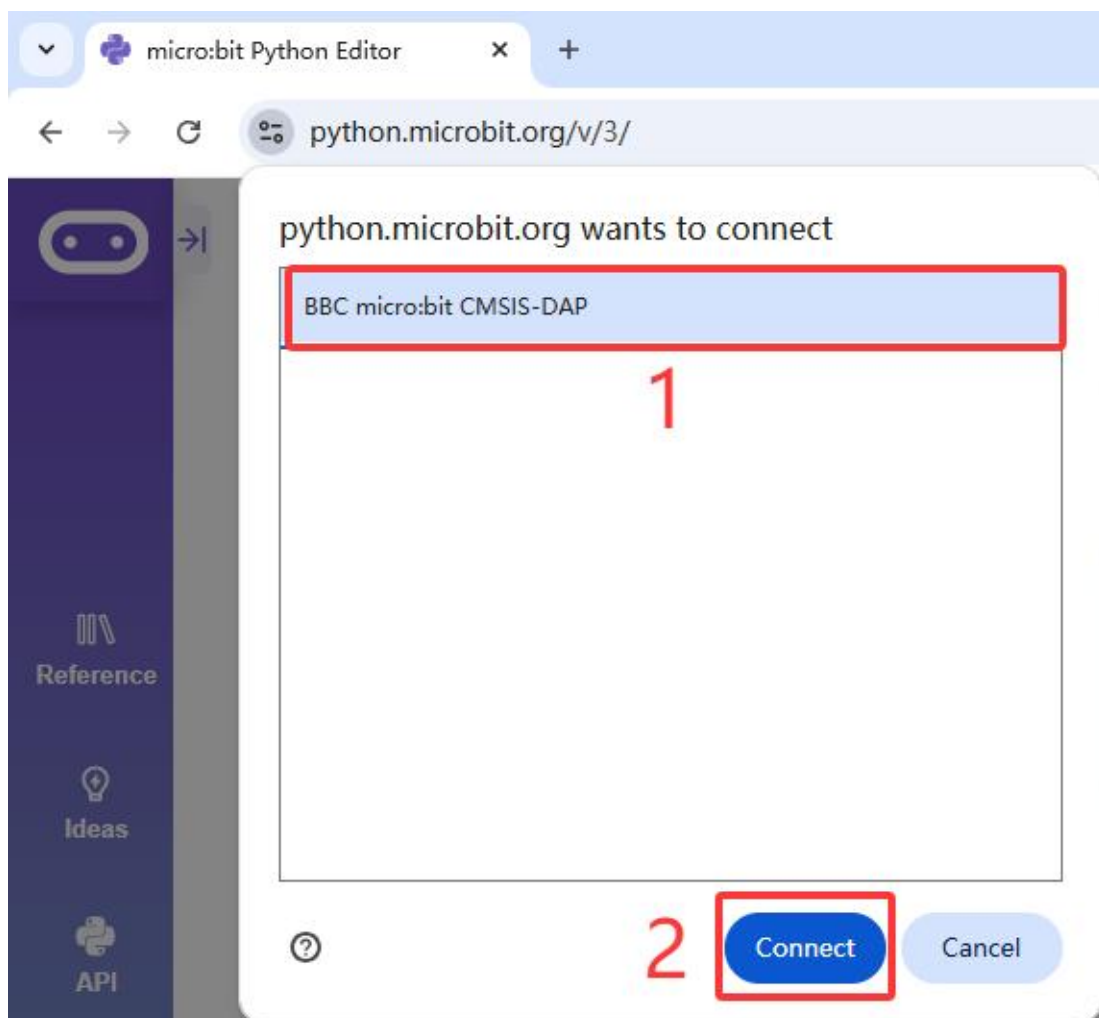


Click **"Next"**:

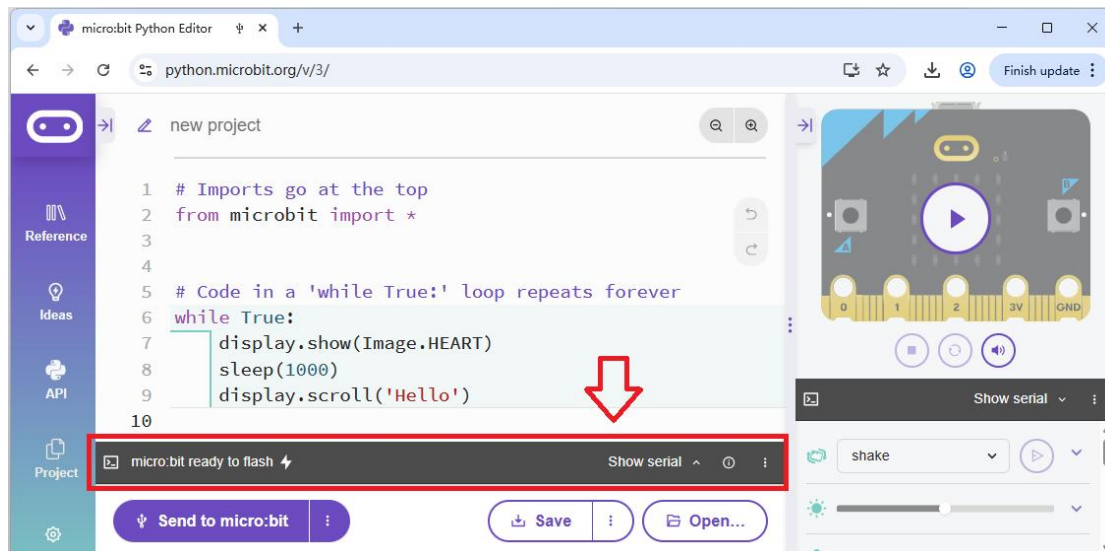




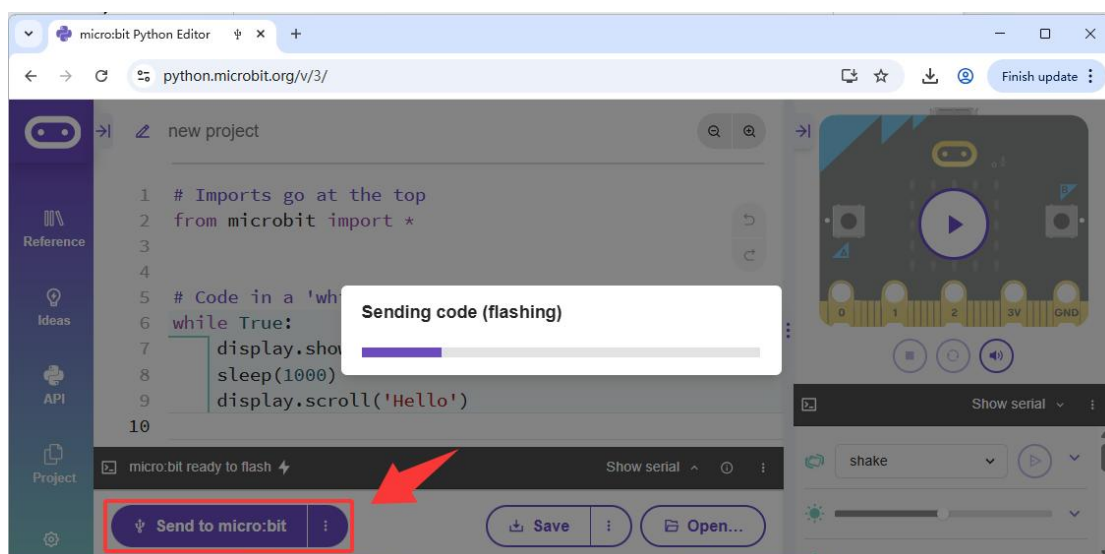
Choose your micro:bit, then click **“Connect”**.



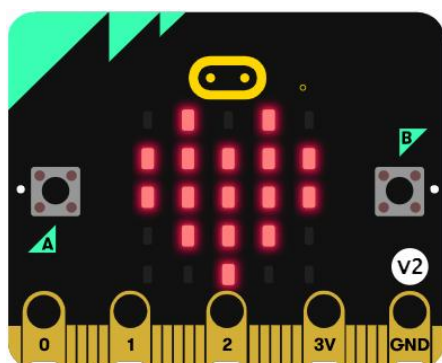
When the **micro:bit ready to flash** interface appears, it indicates that the micro:bit is successfully connected to the Python Editor.



Now you can click **"Send to micro:bit"** to upload the code from the Python Editor to the micro:bit:



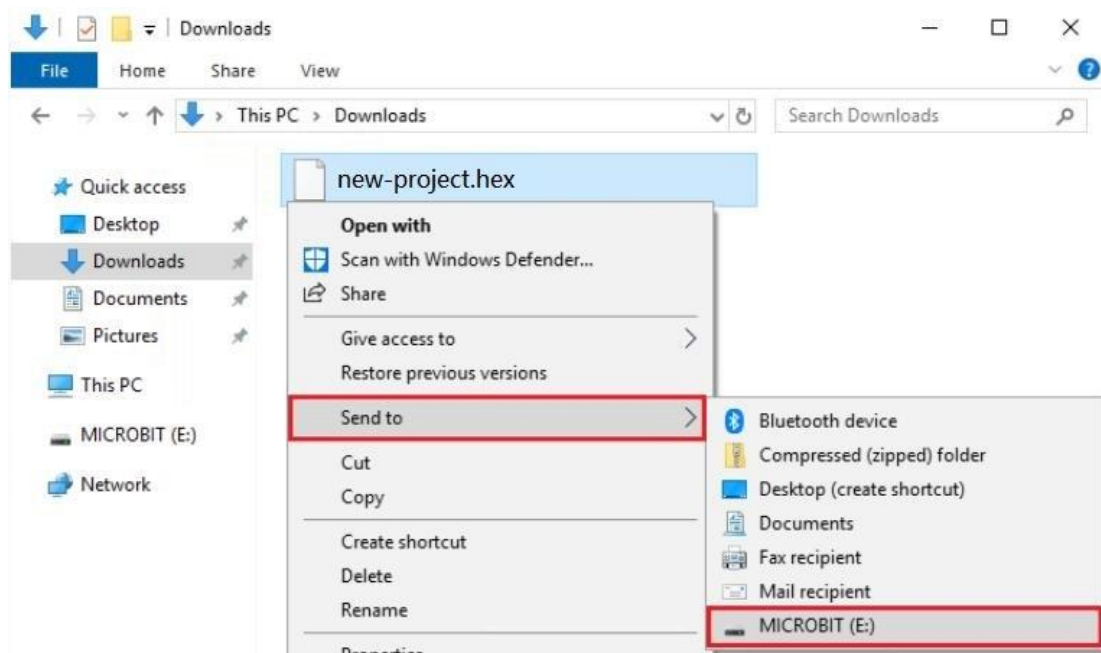
After the code is uploaded, the micro:bit's LED matrix will display a **heart** icon, followed by scrolling the text **"Hello"**:



Two Alternative Methods for Uploading Code

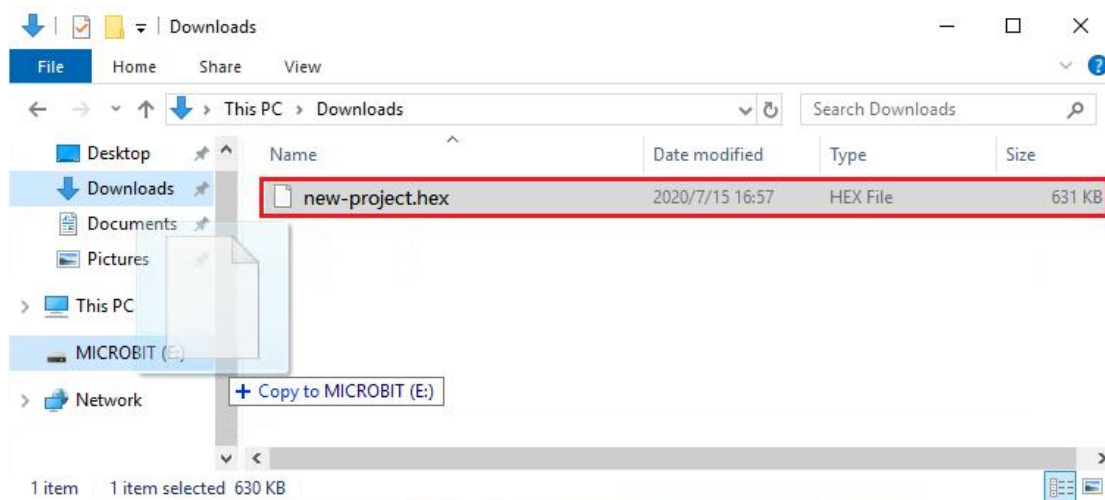
1. Send via Right-Click

Select the ".hex" file, right-click, and choose "Send to" the micro:bit drive:

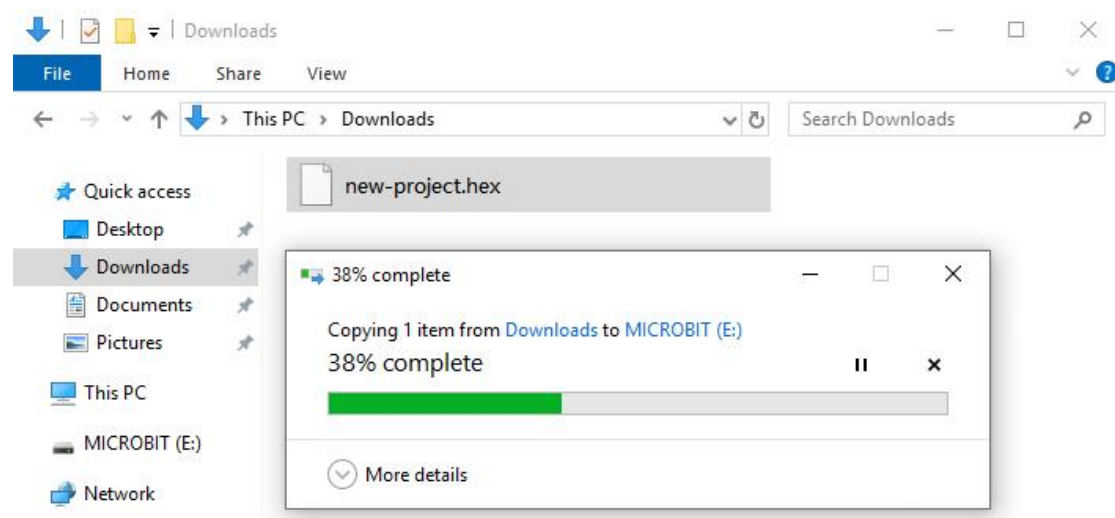


2. Drag and Drop

Alternatively, directly **drag and drop** the ".hex" file onto the micro:bit drive:



The following interface will appear, indicating that the code is being uploaded:



3.5 Learning Basic Syntax of the micro:bit Python Editor

The Micro:bit platform provides official reference documentation for the Python Editor.

User Guide:

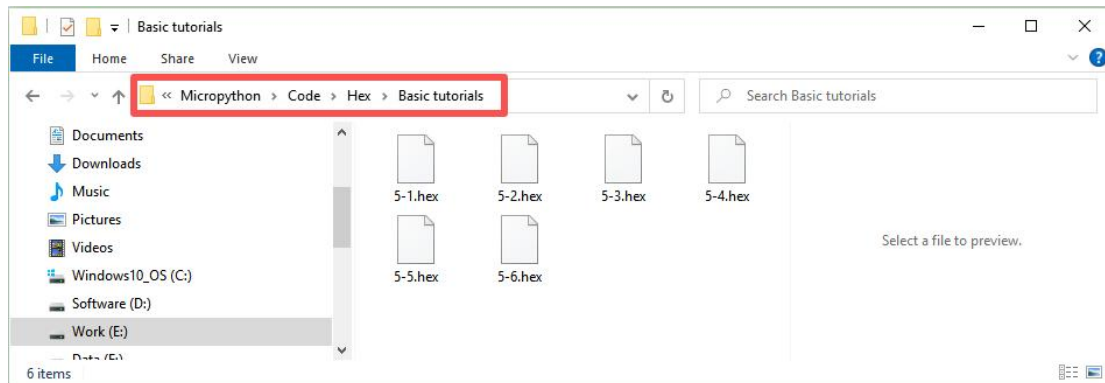
<https://microbit.org/get-started/user-guide/python-editor/>

MicroPython Documentation:

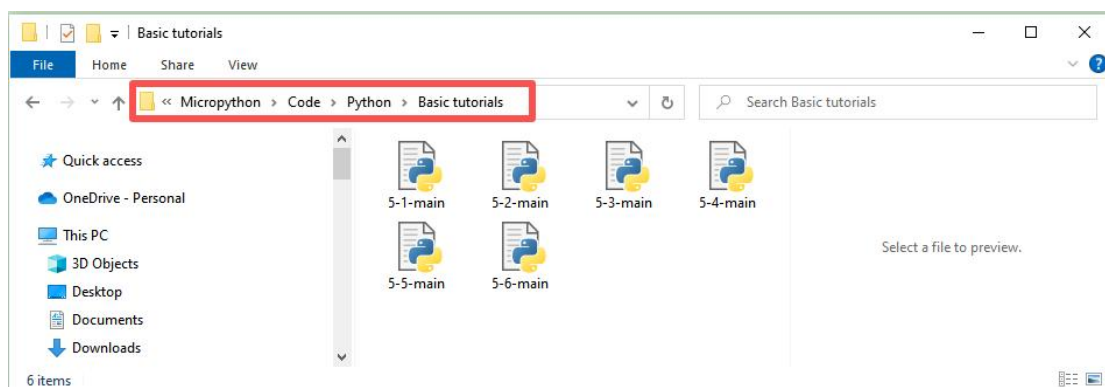
<https://microbit-micropython.readthedocs.io/en/v2-docs/>

4. Basic Example Projects

All basic example hex codes are stored in the **Micropython → Code → Hex → Basic tutorials** folder.



All basic example Python files are stored in the **Micropython → Code → Python → Basic tutorials** folder.




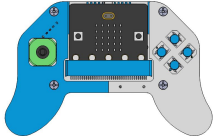

Please refer to [Section 3.3: Import Files](#). You can import the projects from the tutorial package into the editor.

4.1 Button

Goal

Read the values of the joystick's A, B, C, and D buttons.

Required Hardware

PC	Joystick with micro:bit v2	Micro USB cable
		

Code

```
# Imports go at the top
from microbit import *

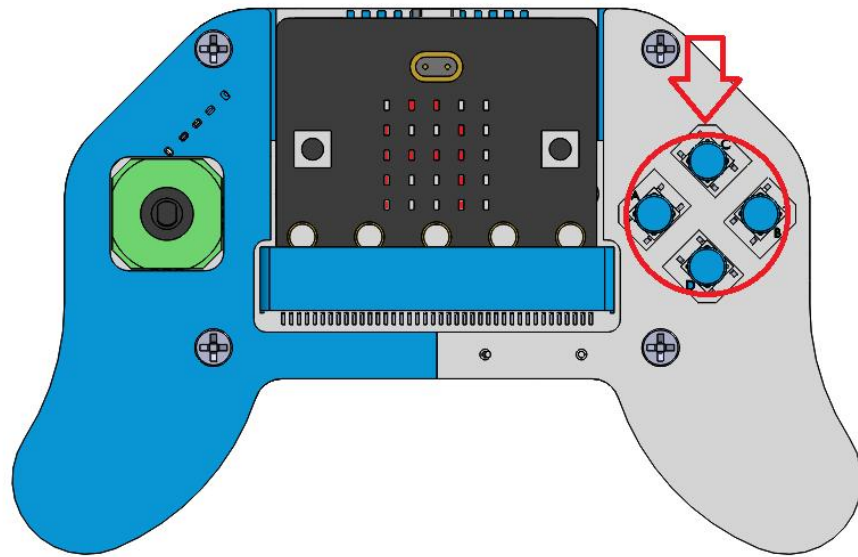
# Button
button_a = pin5
button_b = pin11
button_c = pin12
button_d = pin8

def readButtonValue(button):
    return button.read_digital()

# Code in a 'while True:' loop repeats forever
while True:
    if readButtonValue(button_a) == 0:
        display.show('A')
    if readButtonValue(button_b) == 0:
        display.show('B')
    if readButtonValue(button_c) == 0:
        display.show('C')
    if readButtonValue(button_d) == 0:
        display.show('D')
    sleep(100)
```

Result

When button A, B, C, or D is pressed respectively, the Micro:bit's LED matrix displays 'A', 'B', 'C', or 'D'.


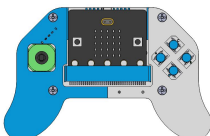



4.2 Joystick

Goal

Read the analog values of the joystick in real time and displays the corresponding directional letters on the micro:bit LED matrix based on the tilt direction of the joystick.

Required Hardware:

PC	Joystick with micro:bit v2	Micro USB cable
		

Code

```
# Imports go at the top
from microbit import *
```

```

# Joystick
x_axis = pin1
y_axis = pin0

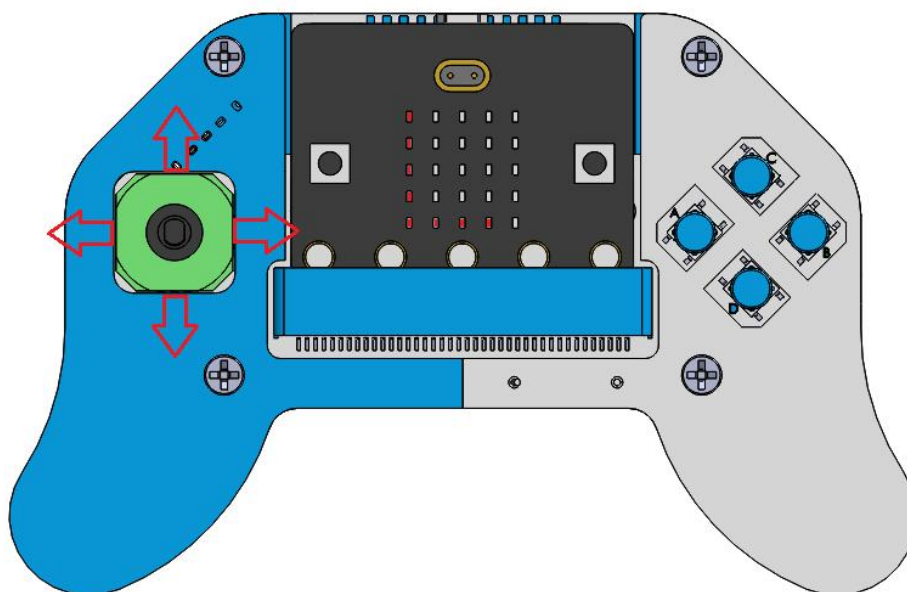
def readJoystickValue(axis):
    # Map 0-1023 values to -100 - +100
    return scale(axis.read_analog(), from_=(0, 1023), to=(-100, 100))

# Code in a 'while True:' loop repeats forever
while True:
    if readJoystickValue(x_axis) > 80:
        display.show('L')
    if readJoystickValue(x_axis) < -80:
        display.show('R')
    if readJoystickValue(y_axis) > 80:
        display.show('U')
    if readJoystickValue(y_axis) < -80:
        display.show('D')
    sleep(100)

```

Result

When the joystick is pushed left, right, up, or down, the Micro:bit's LED matrix displays 'L', 'R', 'U', or 'D' respectively.


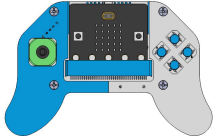



4.3 Vibration motor

Goal

Activate the vibration motor of the joystick.

Required Hardware:

PC	Joystick with micro:bit v2	Micro USB cable
		

Code

```
# Imports go at the top
from microbit import *

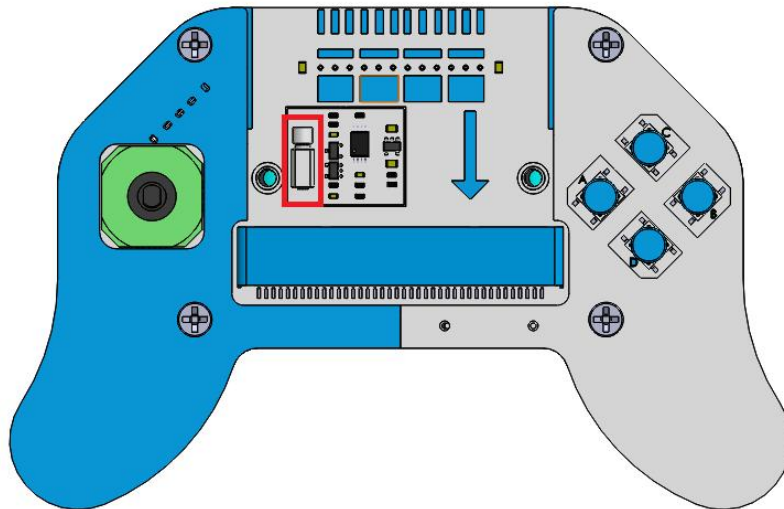
ON = 1
OFF = 0

def setVibrationMotor(OnOrOff):
    if OnOrOff == ON:    # Turn on the vibration motor
        pin2.write_digital(1)
    if OnOrOff == OFF:  # Turn off the vibration motor
        pin2.write_digital(0)

# Code in a 'while True:' loop repeats forever
while True:
    setVibrationMotor(ON)
    sleep(1000)
    setVibrationMotor(OFF)
    sleep(1000)
```

Result

The vibration motor on the joystick vibrates for 1 second every second.


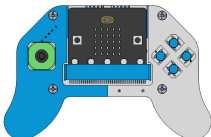



4.4 Battery voltage

Goal

Read the battery level of the joystick.

Required Hardware:

PC	Joystick with micro:bit v2	Micro USB cable
		

Code

```
# Imports go at the top
from microbit import *

# Read the battery level.
# 4 AA batteries
def readBatteryLevel():
    batLevel = pin2.read_analog()
```

```

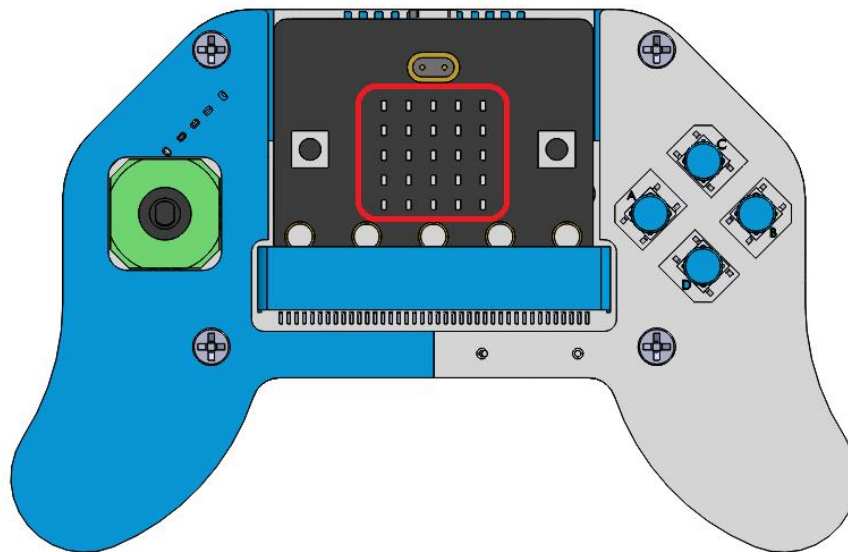
if batLevel > 310: # 310=6V/6/0.0032226, 100%
    batLevel = 310
if batLevel < 232: # 232=4.5V/6/0.0032226, 0%
    batLevel = 232
# Map 232-310 values to 0-100
batLevel = scale(batLevel, from_=(232, 310), to=(0, 100))
return batLevel

# Code in a 'while True:' loop repeats forever
while True:
    display.show(readBatteryLevel())
    sleep(1000)

```

Result

The Micro:bit LED matrix displays the battery level value respectively, within a range of 0-100



Note

Reading the battery voltage and controlling the vibration motor share the same P2 pin on the Micro:bit. Do not switch too frequently between reading the battery voltage and controlling the vibration motor, as this may affect the accuracy of the battery voltage reading. It is recommended to read the voltage continuously multiple times after controlling the vibration motor and take the last value.

4.5 Servo

Goal

Control the rotation of the servo motor.

Required Hardware:

PC	Joystick with micro:bit v2	Micro USB Cable	Servo*4
			

Code

```
# Imports go at the top
from microbit import *

# Define servo pin
servo1 = pin13
servo2 = pin14
servo3 = pin15
servo4 = pin16

# Servo control function
# Set steering Angle (0-180 degrees)
def setServoAngle(pin, angle):
    # The period of the PWM signal is set to 20ms.
    pin.set_analog_period(20)

    # Convert Angle to pulse width (0.5ms-2.5ms)
    pulse_width = 500 + (angle * 2000 / 180)

    # Set PWM signal (20ms period)
    pin.write_analog(int(pulse_width / 20000 * 1023))

# Code in a 'while True:' loop repeats forever
```

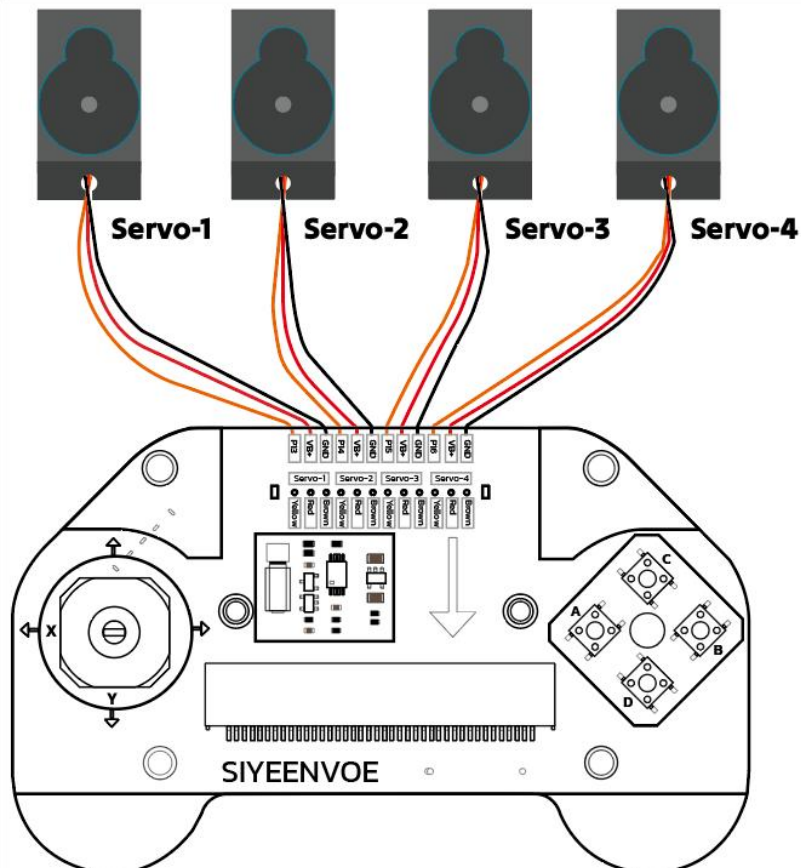
```

while True:
    # 0 degrees
    setServoAngle(servo1, 0)
    setServoAngle(servo2, 0)
    setServoAngle(servo3, 0)
    setServoAngle(servo4, 0)
    sleep(2000)
    # 180 degrees
    setServoAngle(servo1, 180)
    setServoAngle(servo2, 180)
    setServoAngle(servo3, 180)
    setServoAngle(servo4, 180)
    sleep(2000)

```

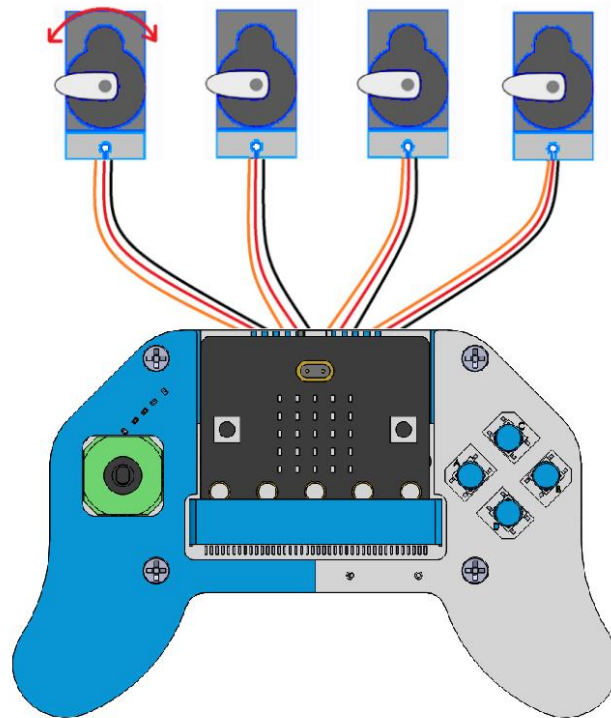
Wiring:

Servo 1			Servo 2			Servo 3			Servo 4		
P13	VB+	GND	P14	VB+	GND	P15	VB+	GND	P16	VB+	GND



Result

The four servos move synchronously, completing one full cycle every 4 seconds ($0^\circ \rightarrow$ hold for 2 seconds $\rightarrow 180^\circ \rightarrow$ hold for 2 seconds \rightarrow return to 0°), then return to step 1 and repeat indefinitely in an infinite loop.



Note:

The joystick must be equipped with 4 AA batteries and the power switch must be turned on!

4.6 Wireless control

Goal

Use two Micro:bit to send data to each other via radio.

Required Hardware:

PC	Joystick with Micro:bit v2	Micro USB Cable	Micro:bit v2
			

Code for the Joystick (Transmitter)

```
# Imports go at the top
from microbit import *
import radio

# Joystick
x_axis = pin1
y_axis = pin0

# Button
button_a = pin5
button_b = pin11
button_c = pin12
button_d = pin8

def readButtonValue(button):
    return button.read_digital()

def readJoystickValue(axis):
    # Map 0-1023 values to -100 - +100
    return scale(axis.read_analog(), from_=(0, 1023), to=(-100, 100))
```

```
# Display image
display.show(Image.HAPPY)
sleep(400)

# Configure the radio
radio.config(group=1, power=3)
radio.on()

# Code in a 'while True:' loop repeats forever
while True:
    if readJoystickValue(x_axis) > 80:
        radio.send('1')
    if readJoystickValue(x_axis) < -80:
        radio.send('2')
    if readJoystickValue(y_axis) > 80:
        radio.send('3')
    if readJoystickValue(y_axis) < -80:
        radio.send('4')

    if readButtonValue(button_a) == 0:
        radio.send('5')
    if readButtonValue(button_b) == 0:
        radio.send('6')
    if readButtonValue(button_c) == 0:
        radio.send('7')
    if readButtonValue(button_d) == 0:
        radio.send('8')
```

Code for the Receiver micro:bit

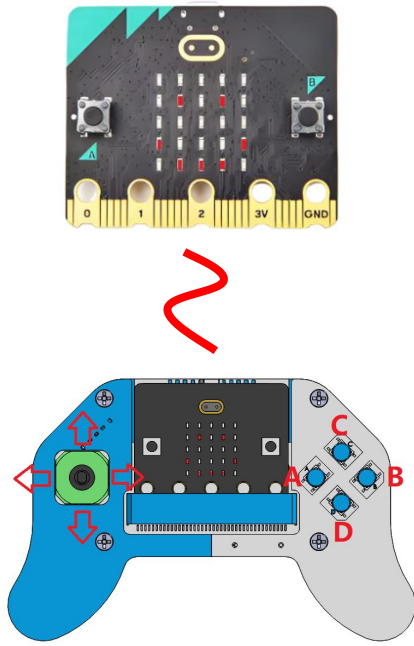
```
# Imports go at the top
from microbit import *
import radio

# Display image
display.show(Image.HEART)
sleep(400)

# Configure the radio
radio.config(group=1)
radio.on()

# Code in a 'while True:' loop repeats forever
while True:
    message = radio.receive()
    if message == '1':
        display.show(1)
    elif message == '2':
        display.show(2)
    elif message == '3':
        display.show(3)
    elif message == '4':
        display.show(4)
    elif message == '5':
        display.show(5)
    elif message == '6':
        display.show(6)
    elif message == '7':
        display.show(7)
    elif message == '8':
        display.show(8)
```

Result

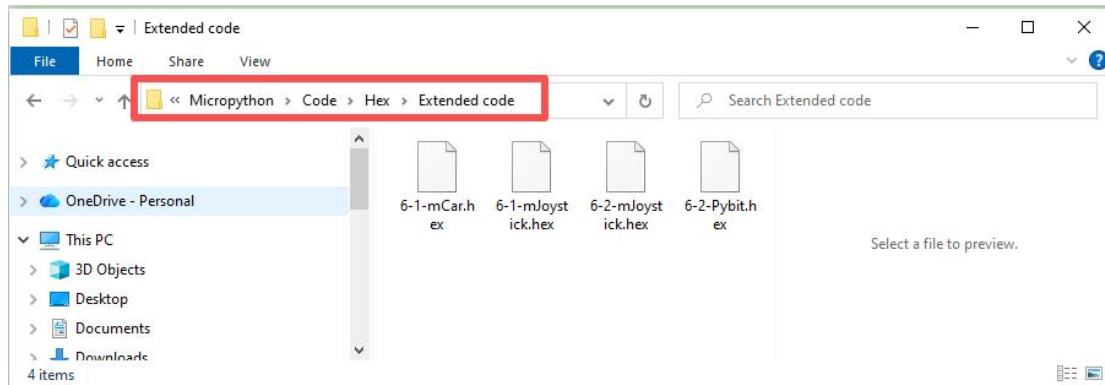
Transmitter Action	Receiver Display	
Push joystick right	1	
Push joystick left	2	
Push joystick up	3	
Push joystick down	4	
Press button A	5	
Press button B	6	
Press button C	7	
Press button D	8	

Note:

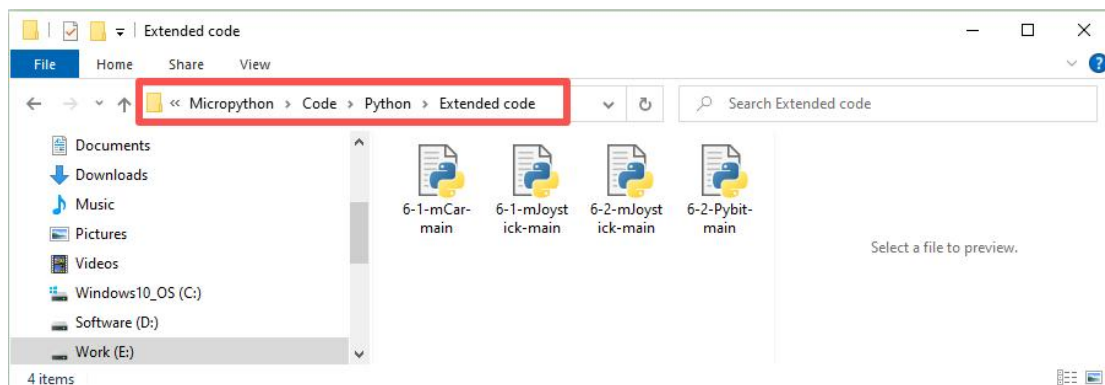
The joystick must be equipped with 4 AA batteries and the power switch must be turned on!

5. Extended Projects

All example hex files for Extended Projects are saved in the **Micropython → Code → Hex → Extended code** folder.



All example Python files for the Extended Projects are saved in the **Micropython → Code → Python → Extended code** folder.



Please refer to [Section 3.3: Import Files](#). You can import the projects from the tutorial package into the editor.

5.1 Control mCar

Goal

Control mCar robot(Micro:bit v2 **not included**, Model No.: M1C0000).

Required Hardware:

PC	Joystick with Micro:bit v2	mCar M1C0000 with Micro:bit v2	Micro USB Cable
			

The tutorial for the mCar can be downloaded here:

<https://siyeenove.com/tutorial/>

SKU	Product-Name	View on GitHub	Download from Github	Download from Cloud
A1D0000	Leonardo R3	→ Go to view	↓ Download	↓ Download
E1R0000	ESP32 C3 eArm	→ Go to view	↓ Download	↓ Download
M1C0000	Micro:bit mCar	→ Go to view	↓ Download	↓ Download
M1C0001	Micro:bit Pybit	→ Go to view	↓ Download	↓ Download
M1E0001	Micro:bit uShield	→ Go to view	↓ Download	↓ Download
M1E0002	Micro:bit mShield	→ Go to view	↓ Download	↓ Download
M1K0000	Micro:bit mJoystick	→ Go to view	↓ Download	↓ Download
M1R0000	Micro:bit mArm	→ Go to view	↓ Download	↓ Download

If you are interested in the mCar robot, you may contact us to purchase it.

Code for the Joystick (Transmitter)

```
# Imports go at the top
from microbit import *
import radio

# Joystick
x_axis = pin1
y_axis = pin0

# Button
button_a = pin5
button_b = pin11
button_c = pin12
button_d = pin8
```

```

def readButtonValue(button):
    return button.read_digital()

def readJoystickValue(axis):
    # Map 0-1023 values to -100 - +100
    return scale(axis.read_analog(), from_=(0, 1023), to=(-100, 100))

# Display image
display.show(Image.HAPPY)
sleep(400)

# Configure the radio
radio.config(group=1, power=3)
radio.on()

# Code in a 'while True:' loop repeats forever
while True:
    if readJoystickValue(x_axis) > 80:
        radio.send('1')
    if readJoystickValue(x_axis) < -80:
        radio.send('2')
    if readJoystickValue(y_axis) > 80:
        radio.send('3')
    if readJoystickValue(y_axis) < -80:
        radio.send('4')

    if readButtonValue(button_a) == 0:
        radio.send('5')
    if readButtonValue(button_b) == 0:
        radio.send('6')
    if readButtonValue(button_c) == 0:
        radio.send('7')
    if readButtonValue(button_d) == 0:
        radio.send('8')

```

Code for mCar:

```
# Imports go at the top
from microbit import *
import radio
import speech

# Car I2C address
i2cAddr = 0x2a

# For wheels
leftwheel = 0
rightwheel = 1
backward = 0
forward = 1
i2cBuf = bytearray([0x00, 0x00])

# For head RGB LED
leftLed = 0
rightLed = 1

# Set the wheel speed function
# wheel: 0 = left wheel, 1 = right wheel
# direction: 0 = backward, 1 = forward
# speed: 0--100
def setWheelSpeed(wheel, direction, speed):
    # Important! The speed is between 0 and 100.
    if speed > 100:
        speed = 100
    elif speed < 0:
        speed = 0

    if wheel == leftwheel:
        i2cBuf[0] = 0x05 # left wheel register
        if direction == forward:
            # speed value, 101 is the default required data.
            i2cBuf[1] = speed + 101
```

```

        elif direction == backward:
            i2cBuf[1] = speed
            i2c.write(i2cAddr, i2cBuf)

    if wheel == rightwheel:
        i2cBuf[0] = 0x06 # right wheel register
        if direction == forward:
            i2cBuf[1] = speed
        elif direction == backward:
            # speed value, 101 is the default required data.
            i2cBuf[1] = speed + 101
        i2c.write(i2cAddr, i2cBuf)

# Set the head RGB LED function
# led: 0 is the left led and 1 is the right LED.
# r: red brightness value
# g: green brightness value
# b: blue brightness value
def setHeadRgbLed(led, r, g, b):
    if led == leftLed:
        i2cBuf[0] = 0x07 # red register
        i2cBuf[1] = r     # red brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x08 # green register
        i2cBuf[1] = g     # green brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x09 # blue register
        i2cBuf[1] = b     # blue brightness value
        i2c.write(i2cAddr, i2cBuf)
    if led == rightLed:
        i2cBuf[0] = 0x0a # red register
        i2cBuf[1] = r     # red brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x0b # green register
        i2cBuf[1] = g     # green brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x0c # blue register

```

```

        i2cBuf[1] = b      # blue brightness value
        i2c.write(i2cAddr, i2cBuf)

# Display image
display.show(Image.HAPPY)
sleep(400)

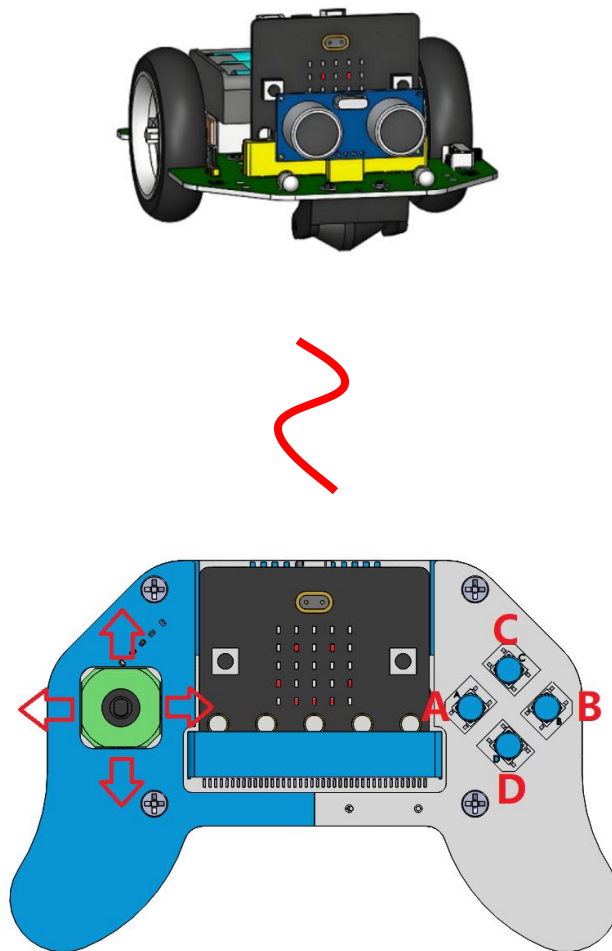
# Configure the radio
radio.config(group=1)
radio.on()

# Code in a 'while True:' loop repeats forever
while True:
    message = radio.receive()
    if message == '1':
        # Turn right at place
        setWheelSpeed(leftwheel, forward, 60)
        setWheelSpeed(rightwheel, backward, 60)
    elif message == '2':
        # Turn left at place
        setWheelSpeed(leftwheel, backward, 60)
        setWheelSpeed(rightwheel, forward, 60)
    elif message == '3':
        # Forward
        setWheelSpeed(leftwheel, forward, 100)
        setWheelSpeed(rightwheel, forward, 100)
    elif message == '4':
        # Backward
        setWheelSpeed(leftwheel, backward, 100)
        setWheelSpeed(rightwheel, backward, 100)
    elif message == '5':
        # Speech
        speech.say('Hello, world. How are you?')
    elif message == '6':
        setHeadRgbLed(leftLed, 0, 255, 0)    # green
        setHeadRgbLed(rightLed, 0, 255, 0)
    elif message == '7':

```

```
setHeadRgbLed(leftLed, 0, 0, 255)    # blue
setHeadRgbLed(rightLed, 0, 0, 255)
elif message == '8':
    setHeadRgbLed(leftLed, 0, 0, 0)    # back
    setHeadRgbLed(rightLed, 0, 0, 0)
else:
    # Stop
    setWheelSpeed(leftwheel, backward, 100)
    setWheelSpeed(rightwheel, backward, 100)
sleep(200)
```

Result



Joystick Action	mCar Function
Push joystick up	Car moves forward
Push joystick down	Car moves backward
Push joystick left	Car turns left
Push joystick right	Car turns right
Press button A	Car beeps
Press button B	Headlights turn green
Press button C	Headlights turn blue
Press button D	Headlights turn off
No action	Car stops

Note:

The joystick must be equipped with 4 AA batteries and the power switch must be turned on!

5.2 Control Pybit

Goal

Control Pybit robot(Micro:bit v2 **not included**, Model No.: M1C0001).

Required Hardware:

PC	Joystick with Micro:bit v2	Pybit M1C0001 with Micro:bit v2	Micro USB Cable
			

The tutorial for the Pybit can be downloaded here:

<https://siyeenove.com/tutorial/>

SKU	Product-Name	View on GitHub	Download from Github	Download from Cloud
A1D0000	Leonardo R3	→ Go to view	↓ Download	↓ Download
E1R0000	ESP32 C3 eArm	→ Go to view	↓ Download	↓ Download
M1C0000	Micro:bit mCar	→ Go to view	↓ Download	↓ Download
M1C0001	Micro:bit Pybit	→ Go to view	↓ Download	↓ Download
M1E0001	Micro:bit uShield	→ Go to view	↓ Download	↓ Download
M1E0002	Micro:bit mShield	→ Go to view	↓ Download	↓ Download
M1K0000	Micro:bit mJoystick	→ Go to view	↓ Download	↓ Download
M1R0000	Micro:bit mArm	→ Go to view	↓ Download	↓ Download

If you are interested in the Pybit robot, you may contact us to purchase it.

Code for the Joystick (Transmitter)

```
# Imports go at the top
from microbit import *
import radio

# Joystick
x_axis = pin1
y_axis = pin0

# Button
button_a = pin5
button_b = pin11
button_c = pin12
button_d = pin8

def readButtonValue(button):
    return button.read_digital()

def readJoystickValue(axis):
    # Map 0-1023 values to -100 - +100
    return scale(axis.read_analog(), from_=(0, 1023), to=(-100, 100))

# Display image
display.show(Image.HAPPY)
sleep(400)

# Configure the radio
```

```

radio.config(group=1, power=3)
radio.on()

# Code in a 'while True:' loop repeats forever
while True:
    if readJoystickValue(x_axis) > 80:
        radio.send('1')
    if readJoystickValue(x_axis) < -80:
        radio.send('2')
    if readJoystickValue(y_axis) > 80:
        radio.send('3')
    if readJoystickValue(y_axis) < -80:
        radio.send('4')

    if readButtonValue(button_a) == 0:
        radio.send('5')
    if readButtonValue(button_b) == 0:
        radio.send('6')
    if readButtonValue(button_c) == 0:
        radio.send('7')
    if readButtonValue(button_d) == 0:
        radio.send('8')

```

Code for the Pybit:

```

# Imports go at the top
from microbit import *
import radio
import speech

# Car I2C address
i2cAddr = 0x2a

# For wheels
leftwheel = 0
rightwheel = 1
backward = 0

```

```

forward = 1
i2cBuf = bytearray([0x00, 0x00])

# For head RGB LED
leftLed = 0
rightLed = 1

# Set the wheel speed function
# wheel: 0 = left wheel, 1 = right wheel
# direction: 0 = backward, 1 = forward
# speed: 0--100
def setWheelSpeed(wheel, direction, speed):
    # Important! The speed is between 0 and 100.
    if speed > 100:
        speed = 100
    elif speed < 0:
        speed = 0

    if wheel == lefthewheel:
        i2cBuf[0] = 0x05 # left wheel register
        if direction == forward:
            # speed value, 101 is the default required data.
            i2cBuf[1] = speed + 101
        elif direction == backward:
            i2cBuf[1] = speed
        i2c.write(i2cAddr, i2cBuf)

    if wheel == righthewheel:
        i2cBuf[0] = 0x06 # right wheel register
        if direction == forward:
            i2cBuf[1] = speed
        elif direction == backward:
            # speed value, 101 is the default required data.
            i2cBuf[1] = speed + 101
        i2c.write(i2cAddr, i2cBuf)

# Set the head RGB LED function

```

```

# led: 0 is the left led and 1 is the right LED.
# r: red brightness value
# g: green brightness value
# b: blue brightness value
def setHeadRgbLed(led, r, g, b):
    if led == leftLed:
        i2cBuf[0] = 0x07 # red register
        i2cBuf[1] = r     # red brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x08 # green register
        i2cBuf[1] = g     # green brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x09 # blue register
        i2cBuf[1] = b     # blue brightness value
        i2c.write(i2cAddr, i2cBuf)
    if led == rightLed:
        i2cBuf[0] = 0x0a # red register
        i2cBuf[1] = r     # red brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x0b # green register
        i2cBuf[1] = g     # green brightness value
        i2c.write(i2cAddr, i2cBuf)
        i2cBuf[0] = 0x0c # blue register
        i2cBuf[1] = b     # blue brightness value
        i2c.write(i2cAddr, i2cBuf)

# Display image
display.show(Image.HAPPY)
sleep(400)

# Configure the radio
radio.config(group=1)
radio.on()

# Code in a 'while True:' loop repeats forever
while True:
    message = radio.receive()

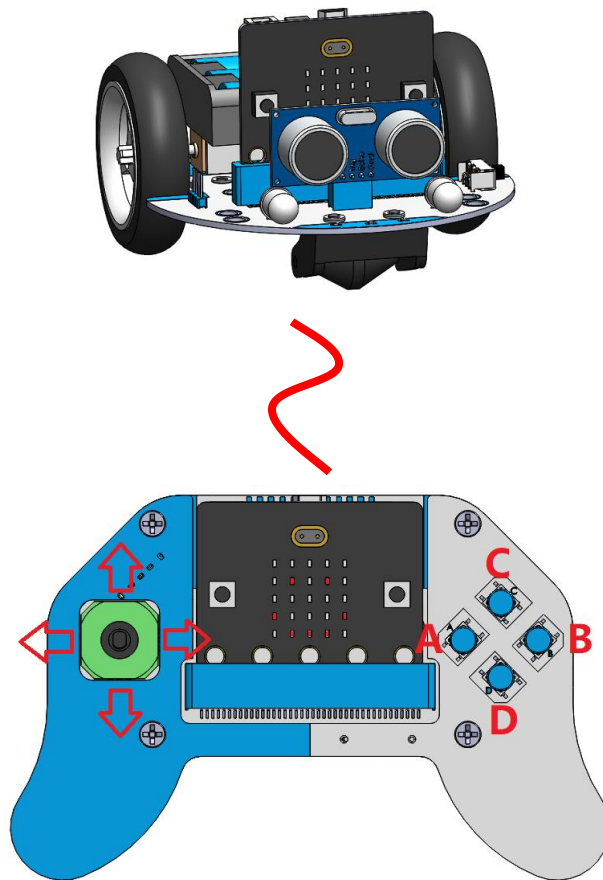
```

```

if message == '1':
    # Turn right at place
    setWheelSpeed(leftwheel, forward, 60)
    setWheelSpeed(rightwheel, backward, 60)
elif message == '2':
    # Turn left at place
    setWheelSpeed(leftwheel, backward, 60)
    setWheelSpeed(rightwheel, forward, 60)
elif message == '3':
    # Forward
    setWheelSpeed(leftwheel, forward, 100)
    setWheelSpeed(rightwheel, forward, 100)
elif message == '4':
    # Backward
    setWheelSpeed(leftwheel, backward, 100)
    setWheelSpeed(rightwheel, backward, 100)
elif message == '5':
    # Speech
    speech.say('Hello, world. How are you?')
elif message == '6':
    setHeadRgbLed(leftLed, 0, 255, 0)    # green
    setHeadRgbLed(rightLed, 0, 255, 0)
elif message == '7':
    setHeadRgbLed(leftLed, 0, 0, 255)    # blue
    setHeadRgbLed(rightLed, 0, 0, 255)
elif message == '8':
    setHeadRgbLed(leftLed, 0, 0, 0)      # back
    setHeadRgbLed(rightLed, 0, 0, 0)
else:
    # Stop
    setWheelSpeed(leftwheel, backward, 100)
    setWheelSpeed(rightwheel, backward, 100)
sleep(200)

```

Result



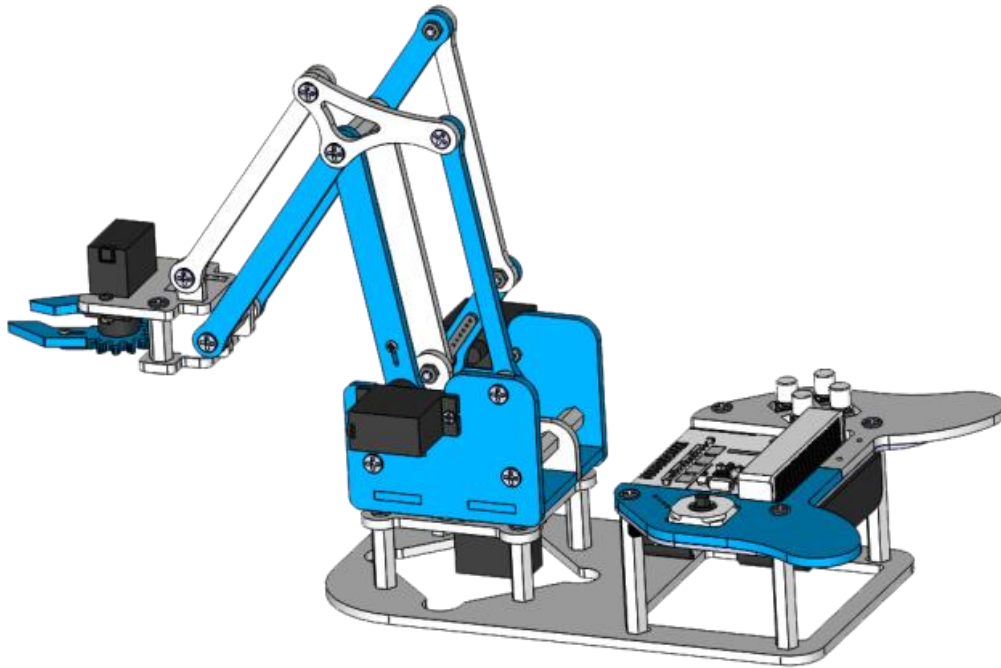
Joystick Action	Pybit Function
Push joystick up	Car moves forward
Push joystick down	Car moves backward
Push joystick left	Car turns left
Push joystick right	Car turns right
Press button A	Car beeps
Press button B	Headlights turn green
Press button C	Headlights turn blue
Press button D	Headlights turn off
No action	Car stops

Note:

The joystick must be equipped with 4 AA batteries and the power switch must be turned on!

5.3 Joystick Control Robot Arm

The Joystick can also be used to control a robotic arm. Here is the mArm 4DF Robotic Arm we developed that includes this joystick (**Micro:bit v2 not included**, Model No.: M1R0000). You can contact us to purchase the mArm kit or the accessories of it excluding the joystick.



The tutorial for the mArm robot arm can be downloaded here:

<https://siyeenove.com/tutorial/>

SKU	Product-Name	View on GitHub	Download from Github	Download from Cloud
A1D0000	Leonardo R3	→ Go to view	↓ Download	↓ Download
E1R0000	ESP32 C3 eArm	→ Go to view	↓ Download	↓ Download
M1C0000	Micro:bit mCar	→ Go to view	↓ Download	↓ Download
M1C0001	Micro:bit Pybit	→ Go to view	↓ Download	↓ Download
M1E0001	Micro:bit uShield	→ Go to view	↓ Download	↓ Download
M1E0002	Micro:bit mShield	→ Go to view	↓ Download	↓ Download
M1K0000	Micro:bit mJoystick	→ Go to view	↓ Download	↓ Download
M1R0000	Micro:bit mArm	→ Go to view	↓ Download	↓ Download

6. QA

6.1 Micro:bit Code Upload Failure

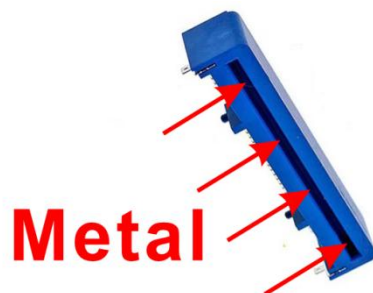
- ☛ Verify USB cable supports data transmission (charge-only cables will not work)
- ☛ Inspect USB cable and ports for physical damage or connectivity issues

6.2 Abnormal Micro:bit Drive Presentation

- ☛ Drive labeled "MAINTENANCE" (instead of "MICROBIT") indicates accidental entry to firmware recovery mode. Resolution:
 - ☛ Reflash firmware using official tool:
<https://microbit.org/get-started/user-guide/firmware/>
- ☛ Prevent recurrence by avoiding reset button depression during power cycling

6.3 Buttons or Joystick Not Responding

- ☛ Ensure the Micro:bit is fully inserted into the mJoystick slot.
- ☛ Ensure the edge connector of the Micro:bit is clean.
- ☛ Ensure the mJoystick slot is clean



6.4 Other Issues

- ☛ Please check whether the batteries have sufficient power.
- ☛ Please check whether the batteries used meet the required specifications.

7.Contact Us

If you couldn't find a solution above, please contact our support team.

To help us assist you quickly, please have the following information ready:

Your order number.

Product model (e.g., Robot Arm Kit M1K0000) and software version

A detailed description of the issue or your question.

Steps you have already tried.

Any relevant error message screenshots, photos, or code snippets.

Other Inquiries?

Please feel free to reach out for:

Tutorial Errors & Feedback: Help us make our documentation better.

Product Ideas & Suggestions: We'd love to hear your great ideas.

Partnerships & Collaboration: Interested in working together? Let's talk.

Discounts & Promotions: Inquire about educational or bulk pricing.

Anything Else: For all other non-technical questions.

 support@siyeenove.com

 <https://siyeenove.com>