目次

1. がいよう

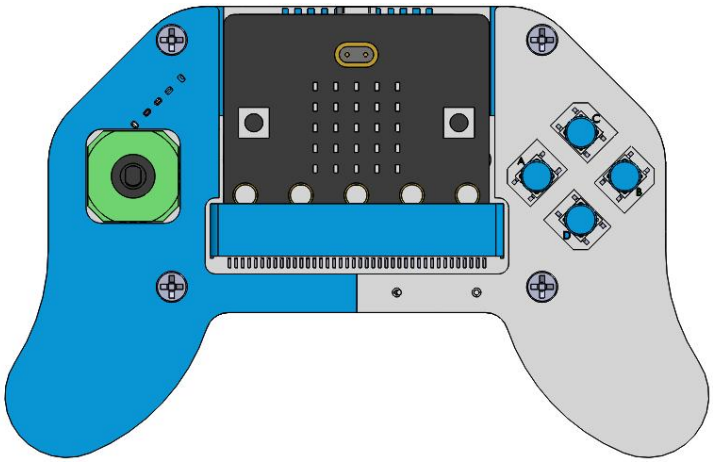
この拡張ボードは、BBC micro:bit用に設計されたインタラクティブ入力ペリフェラルです。デュアルアックスジョイスティック、複数の機能ボタン、振動フィードバックモジュールを統合しており、ゲームコントローラー、ロボットの遠隔操作、インタラクティブ教育などのアプリケーションに適しています。

特徴:

デュアルアックスアナログジョイスティック: X軸とY軸の両方でアナログ入力を提供し、リアルタイムの位置検出を可能にします。
マルチファンクションボタン: ジョイスティック内蔵ボタンに加えて、4つの独立したデジタルボタン（A/B/C/D）があり、カスタムアクション用にプログラム可能です。
振動モーター: 触覚フィードバック用の3610タイプの振動モーターを装備し、プログラム制御でトリガーされます。
拡張インターフェース: 複数のデジタルI/Oピン、サーボ制御ピン、SPI通信サポートを提供し、追加のセンサーやアクチュエーターを接続できます。
独立電源: 単三電池4本（公称6V）で動作可能で、専用の電源スイッチを備えています。micro:bitのオンボード電源に依存する必要はありません。

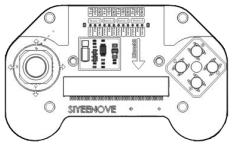
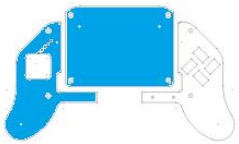
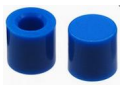
典型的な用途:

ゲームコントローラーの開発
ロボットの動作制御
STEM教育プロジェクト
インタラクティブアートインスタレーション
物理フィードバックシミュレーション実験

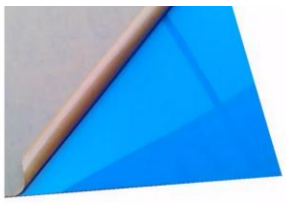


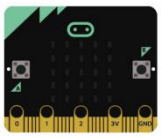
Micro:bitは別途ご購入ください！

1.1 パッケージ内容

Joystick 1Pcs	ナイロンスタンドオフ 3x18mm 4Pcs	ネジ 3x9mm 8Pcs
		
アクリル板 3枚	ドライバー 1Pcs	ボタンキャップ 4Pcs
		

アクリル部品は白と青です。表面の保護フィルムは剥がしてください。

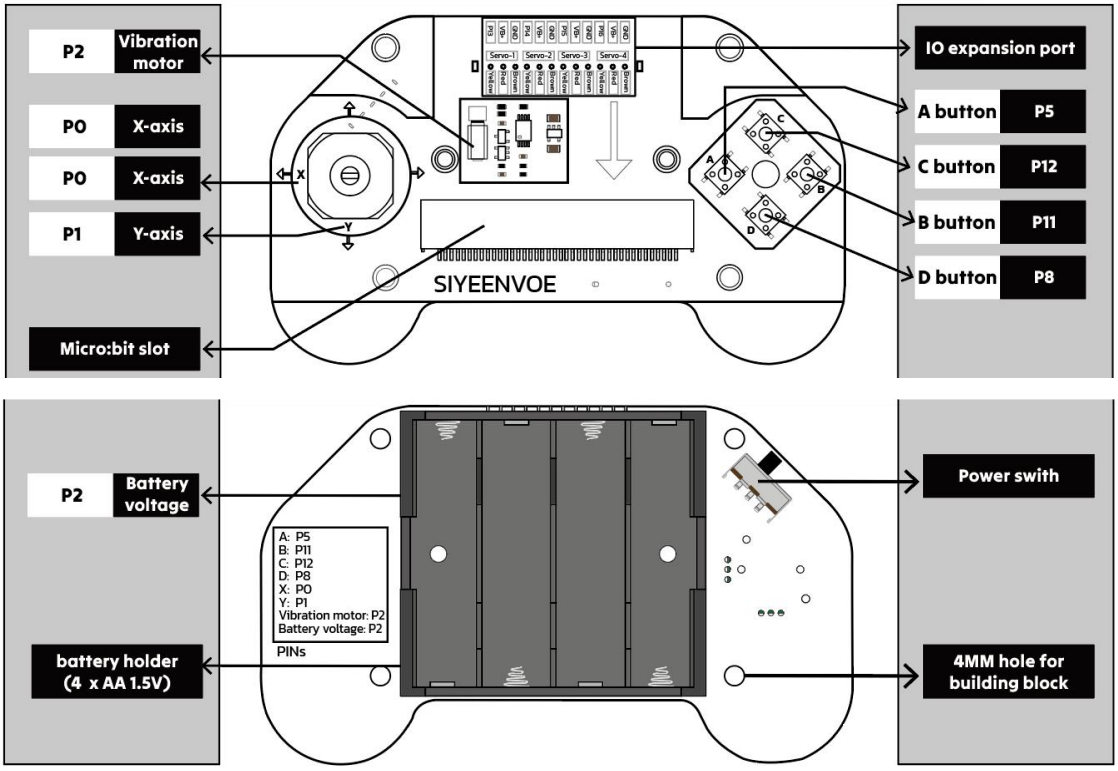


ご自身でご用意いただくもの（含まれていません）			
			
Micro:bit v2 1PCS	Micro USBケーブル	インターネット接続とUSB ポートを持つコンピュータ	単三電池 1.5V 4 本

1.2 仕様

シールド	
名称	Joystick
SKU	M1K0000
対応マザーボード	
Micro:bit	
Pins	
Digital I/O Pins	4 (P13, P14, P15, P16)
Servo pins	4 (P13, P14, P15, P16)
通信	
SPI	Yes (P13, P14, P15, P16)
電源	
入力電圧（公称）	6V (4 AA batteries, 1.5V/PCS)
VB(+/-)	6V (4 AA batteries, 1.5V/PCS)
振動モーター	
Model	3610 Vibration motor
定格電圧/電流	2.7V/75mA
回転速度	14000±2500RPM
Dimensions	
Width	72.5 mm
Length	120 mm
Height	29mm
重量	
PCBA	50 g
With Acrylic	82 g

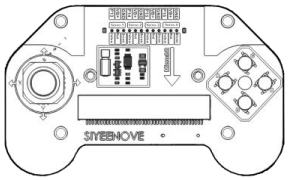
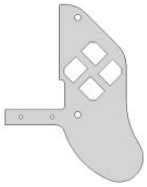
1.3 インターフェース紹介

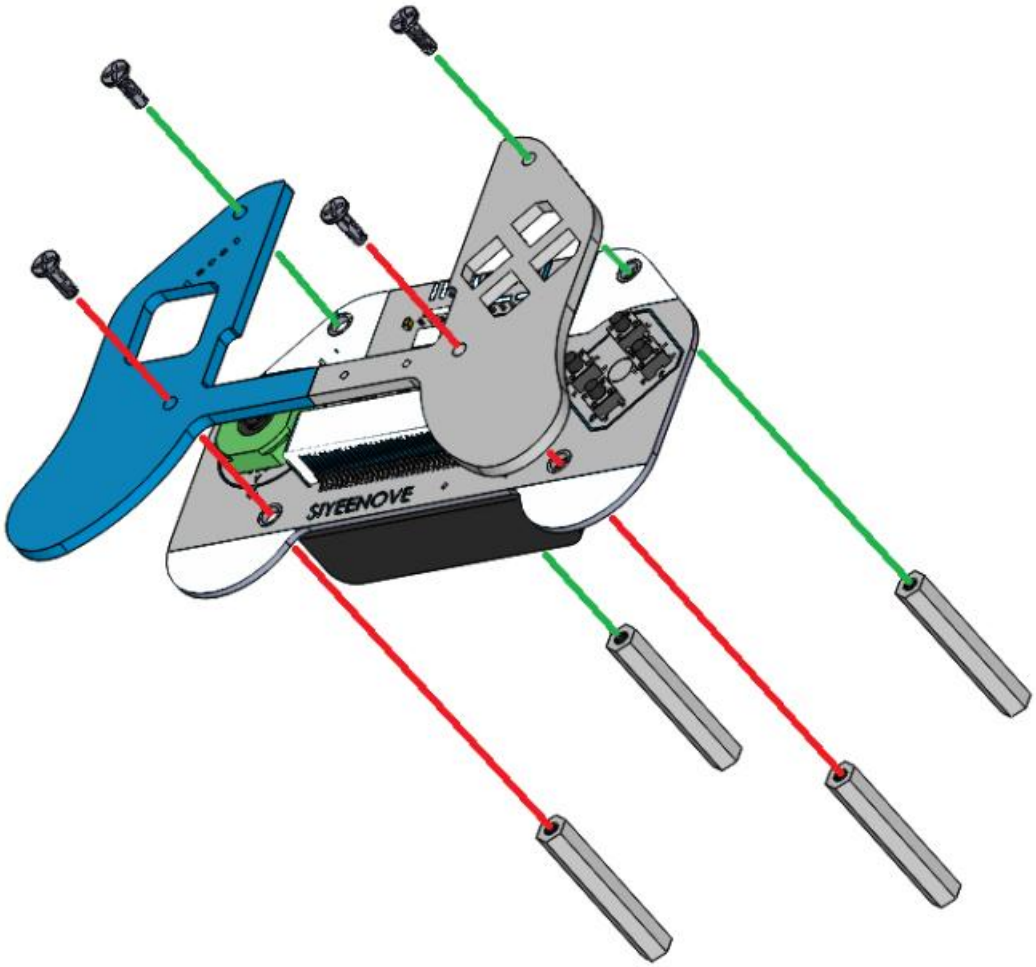


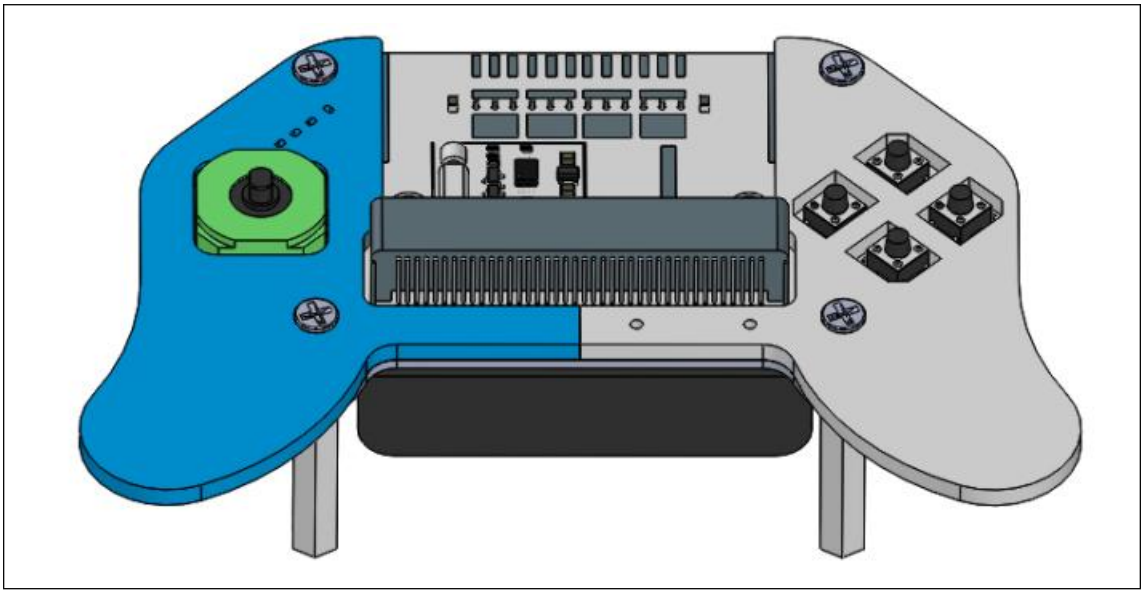
名称	説明
振動モーター r	ハンドルに触覚フィードバックを提供。Micro:bit の P2 ピンで制御。
X 軸	ジョイスティックの X 軸。アナログ値は Micro:bit の P0 ピンで読み取れます。
Y 軸	ジョイスティックの Y 軸。アナログ値は Micro:bit の P1 ピンで読み取れます。
Micro:bit slot	Micro:bit メインコントロールボードを挿入するための設計。
単三電池 1.5V 4 本用電池ケース	4 本の電池を収納。各電池は標準 1.5V で、直径約 14.5mm、高さ約 50.5mm。
電池電圧	単三電池 4 本が取り付けられている場合、Micro:bit の P2 ピンで電圧を読み取れます。
IO extension	外部サーボやセンサーを接続するため。VB ピンの電圧は 4 本の単三電池の直列電圧（1.5V×4 = 6V）に等しい。
A button	その値は Micro:bit の P5 ピンで読み取れます。
B button	その値は Micro:bit の P12 ピンで読み取れます。
C button	その値は Micro:bit の P11 ピンで読み取れます。
D button	その値は Micro:bit の P8 ピンで読み取れます。
Power switch	ハンドルのメイン電源を制御します。
4MM hole	ハンドルには 4×4mm 穴があり、LEGO ブロックとの互換性があり拡張用です。

2 ジョイスティックの組み立て

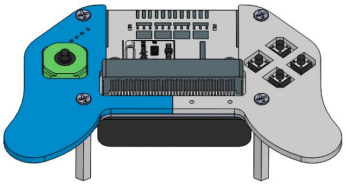
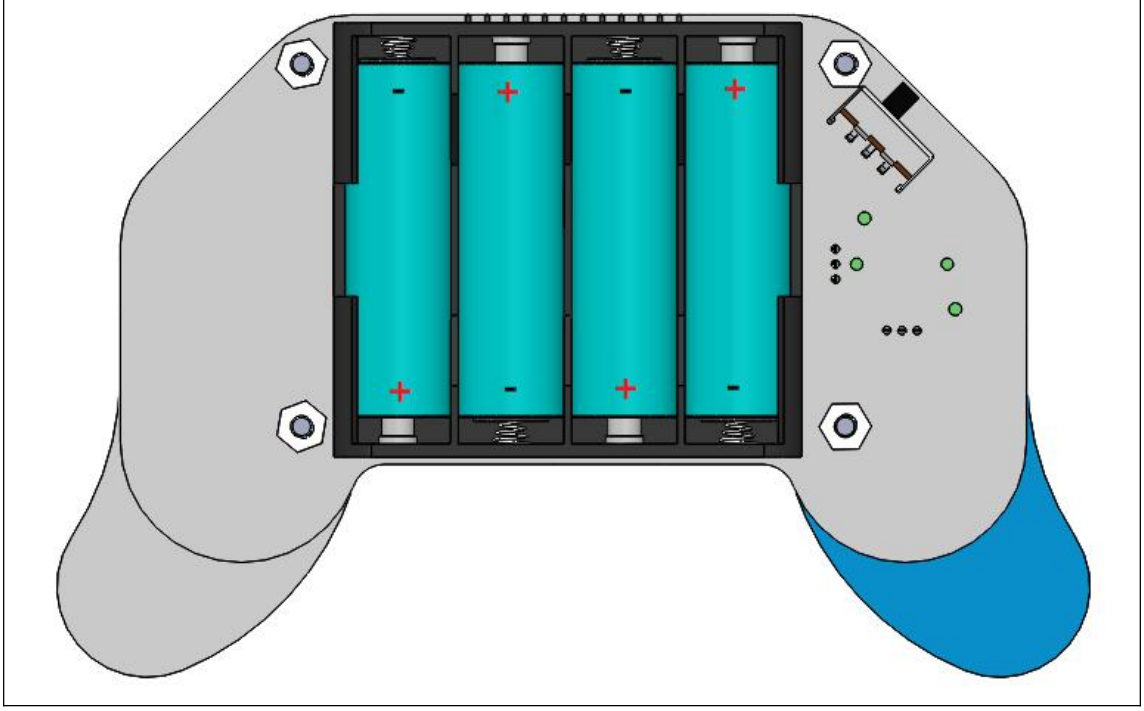
Step 1 - アクリル延長ハンドルの組み立て

Joystick × 1	アクリル板 × 1	アクリル板 × 1
		
ナイロンスタンドオフ 3×18mm × 4	ネジ 3×9mm × 4	注意
		アクリル板の表面を覆う保護フィルムは剥がしてください。

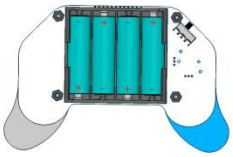
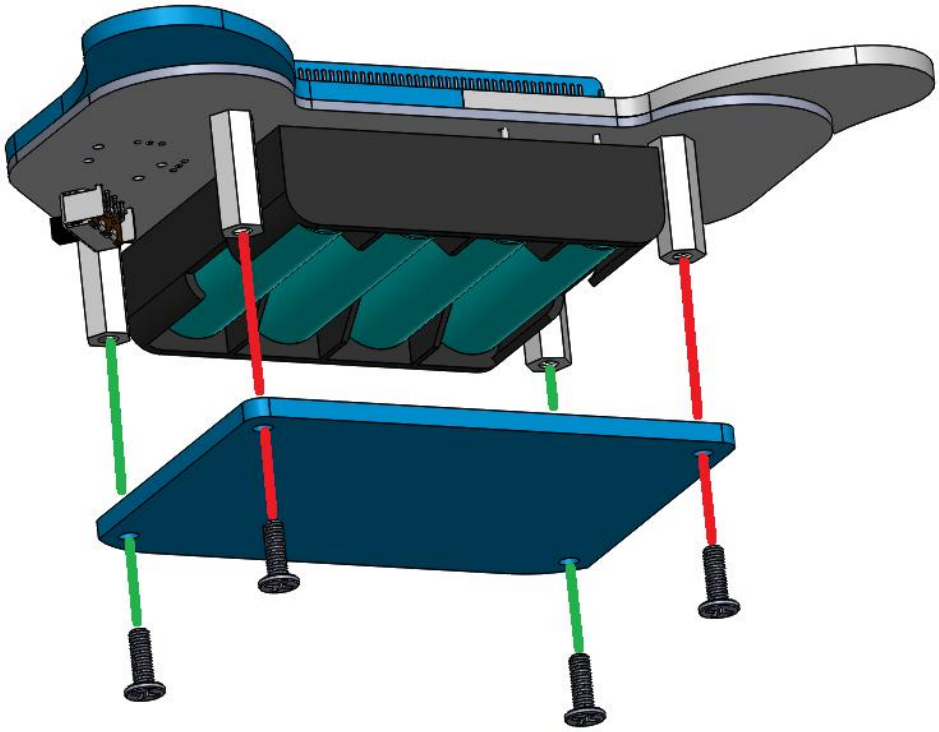
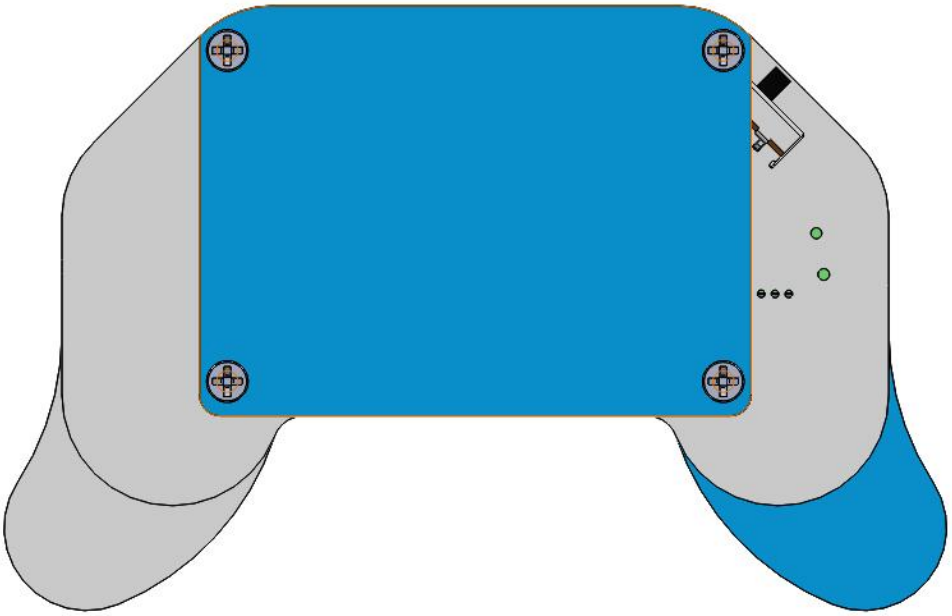




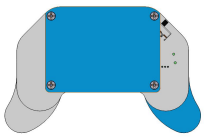
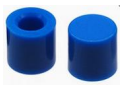
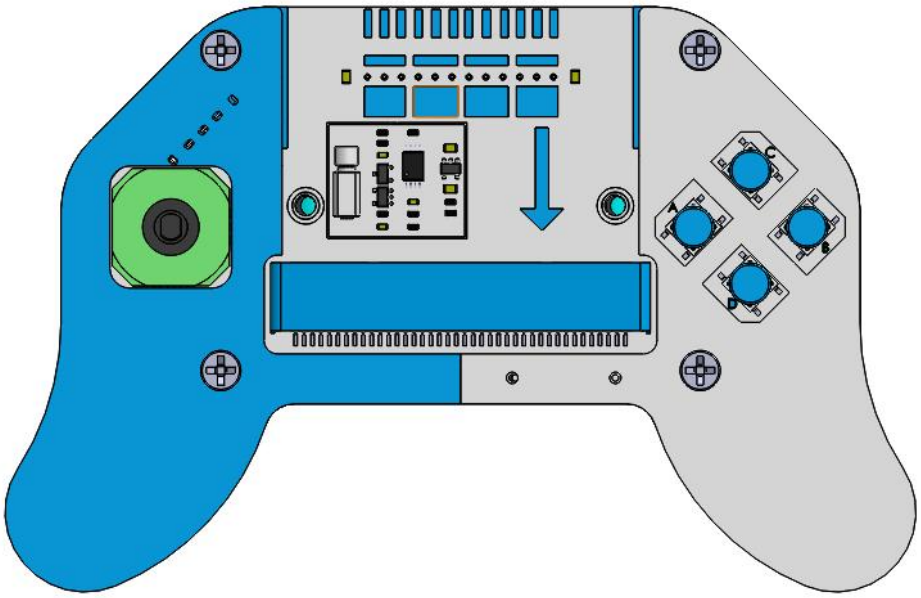
Step 2 – 電池の取り付け

ステップ**1**からの部品	単三電池 × **4**
	
	

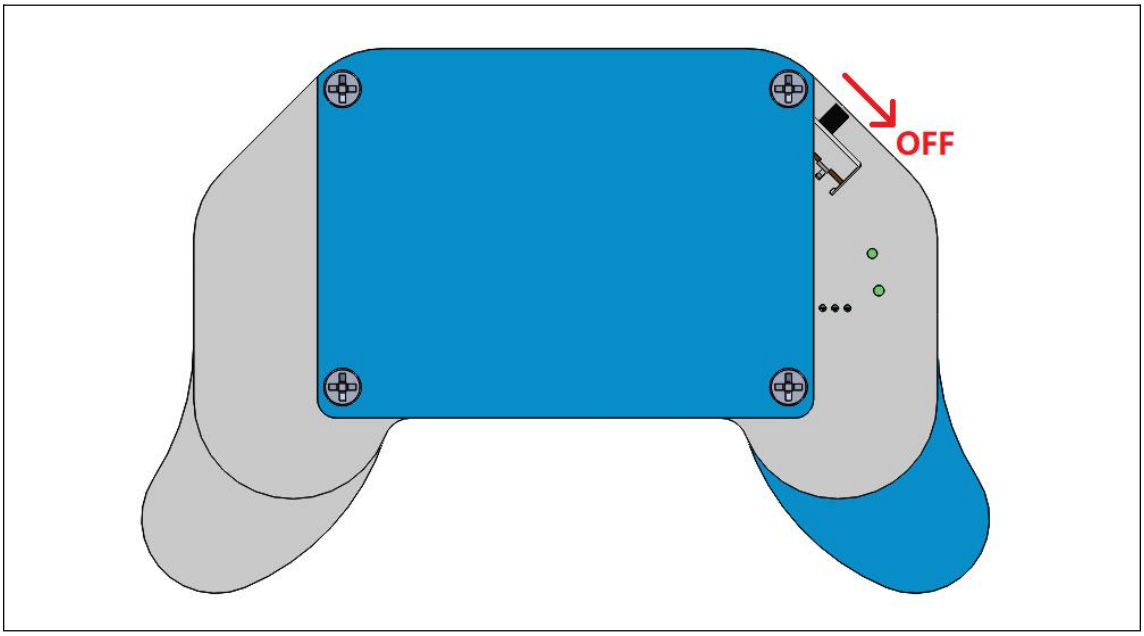
Step 3 – 底面アクリル電池カバーの取り付け

ステップ2からの部品	アクリル板 × 1	ネジ × 4
		
		
		

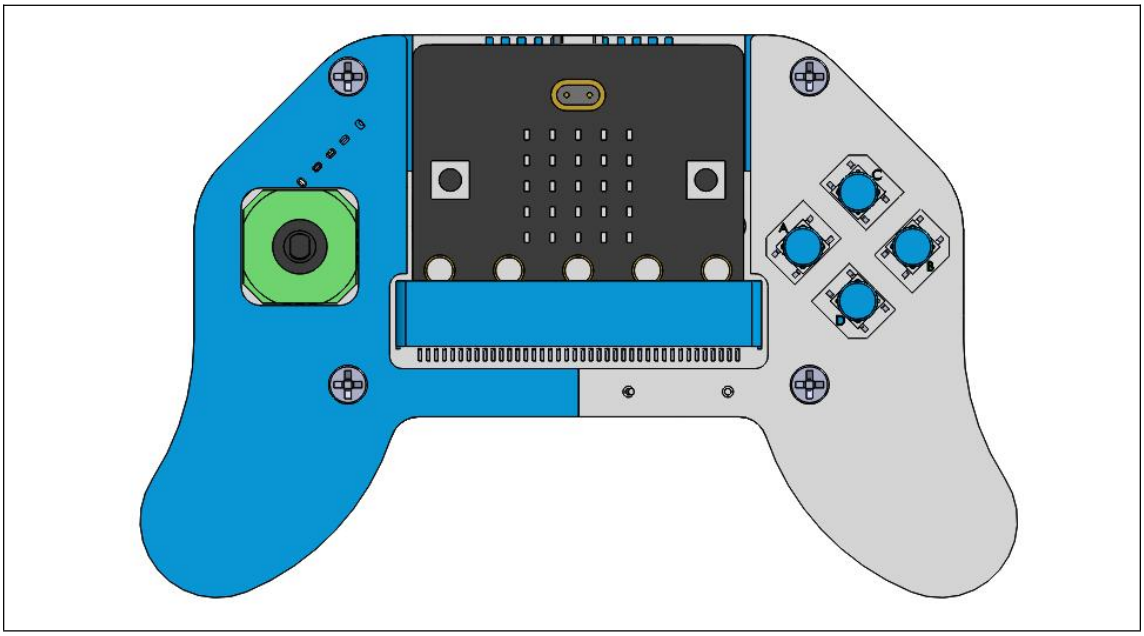
Step 4 – ボタンキャップの取り付け

ステップ3からの部品	ボタンキャップ × 4
	
	

Step 5 – 電源スイッチをオフにする



Step 6 – micro:bit の取り付け



3. micro:bit Python Editor クイックスタート

micro:bit Python Editorは、BBC micro:bitでMicroPythonプログラミングを始めるのに最適な方法です。無料で、すべてのプラットフォームのブラウザで動作します。

Google ChromeまたはMicrosoft Edgeブラウザの使用を推奨します。WebUSBは比較的新しく発展中のWeb機能で、Webページから直接micro:bitにアクセスできます。また、micro:bitからmicro:bit Pythonエディタにデータを直接受信することも可能です。Google ChromeとMicrosoft Edgeブラウザで動作します。

micro:bit用WebUSBサポート
ChromeまたはMicrosoft Edgeブラウザの最新バージョンを使用していない場合は、以下のバージョン以降であることを確認してください：

Android、Chrome OS、Linux、macOS、Windows 10用 Chrome（バージョン79以降）
Android、Chrome OS、Linux、macOS、Windows 10用 Microsoft Edge（バージョン79以降）
最新のGoogle Chromeをダウンロードするリンク：

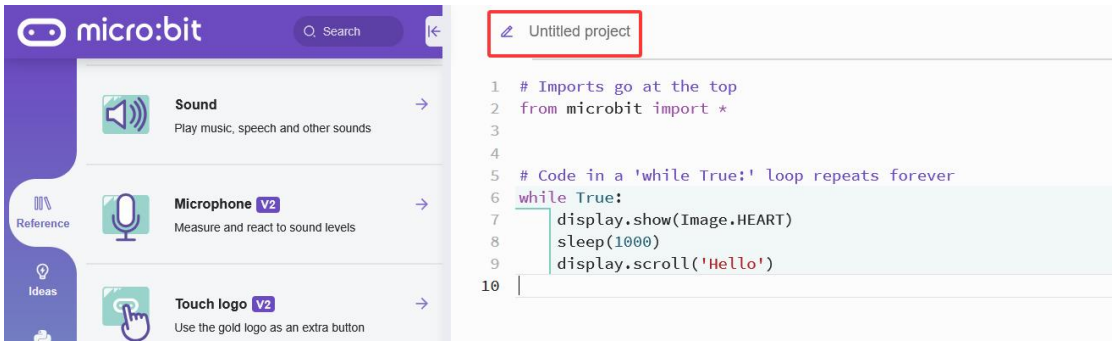
https://www.google.com/chrome/
最新のMicrosoft Edgeをダウンロードするリンク：
https://www.microsoft.com/en-us/edge/download

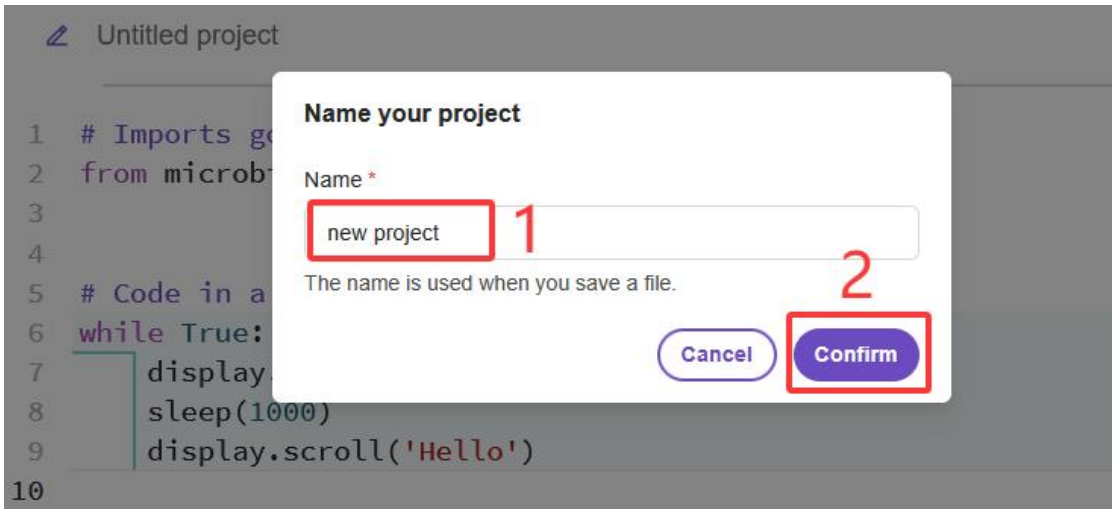
3.1 新規プロジェクトの作成

ブラウザでmicro:bit Python Editorを開きます：https://python.microbit.org/v/3.

エディタが読み込まれると、シンプルなスターターテンプレートを含むコード編集エリアが表示されます。エディタ上部の「Untitled project」をクリックし、プロジェクトに意味のある名前を入力します（例：new project）。

既存のテンプレートを編集するか、独自のPythonコードで完全に置き換えることができます。



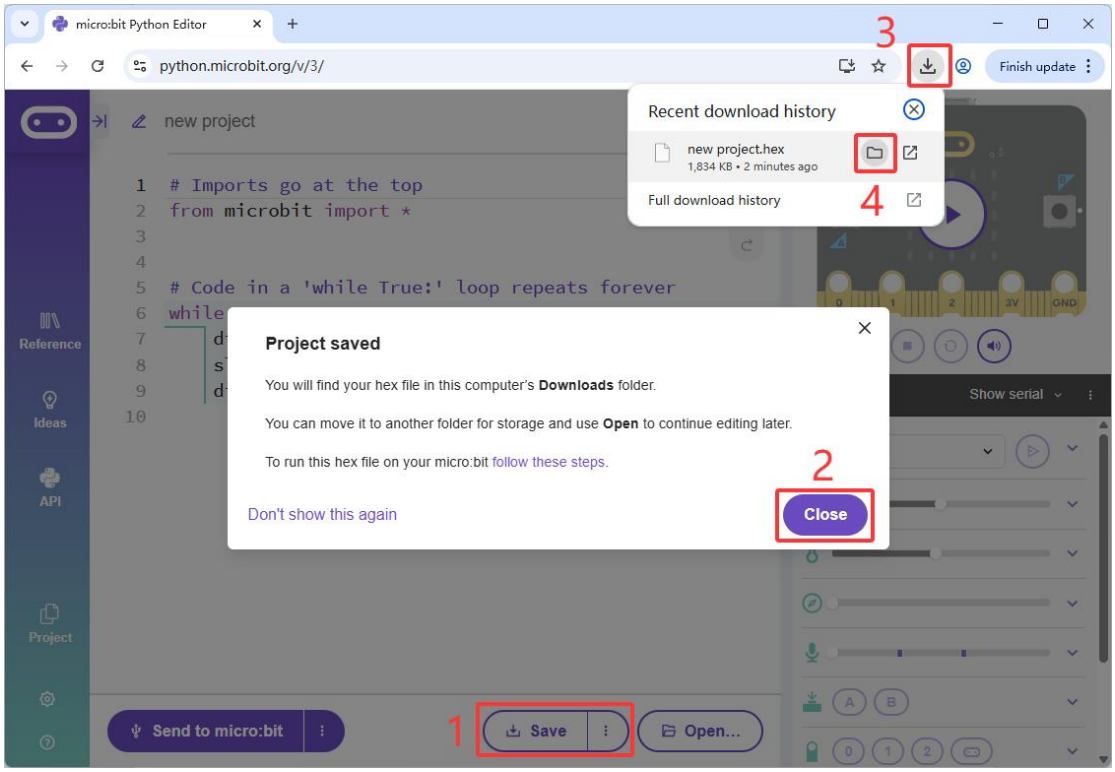


注意（Note）: micro:bit Python Editorは単一プロジェクトモデルで動作し、一度に編集できるプロジェクトは一つだけです。マルチタブ閲覧、複数ファイルの並列編集、組み込みのプロジェクト切り替えはサポートされていません。

3.2 プロジェクトの保存

1. プロジェクトをhexファイルとして保存

「Save」ボタンをクリックすると、エディタは作業内容をブラウザのローカルストレージに自動的に保存します。ダウンロードされたhexファイルは、ブラウザのデフォルトダウンロードフォルダに保存されます。



これはmicro:bitが理解・実行できる最終ファイル形式です。

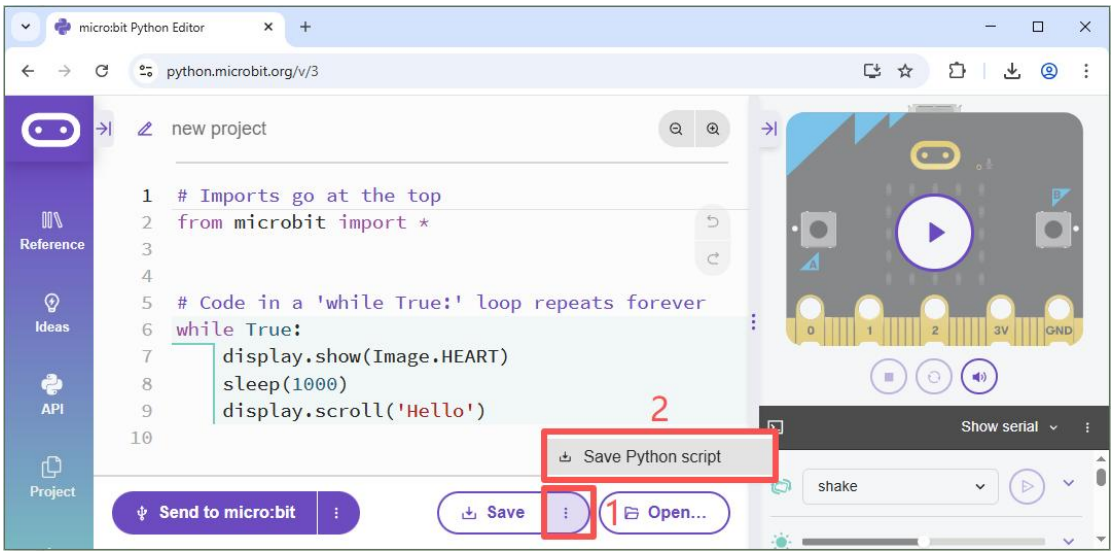
機能（Functions）：

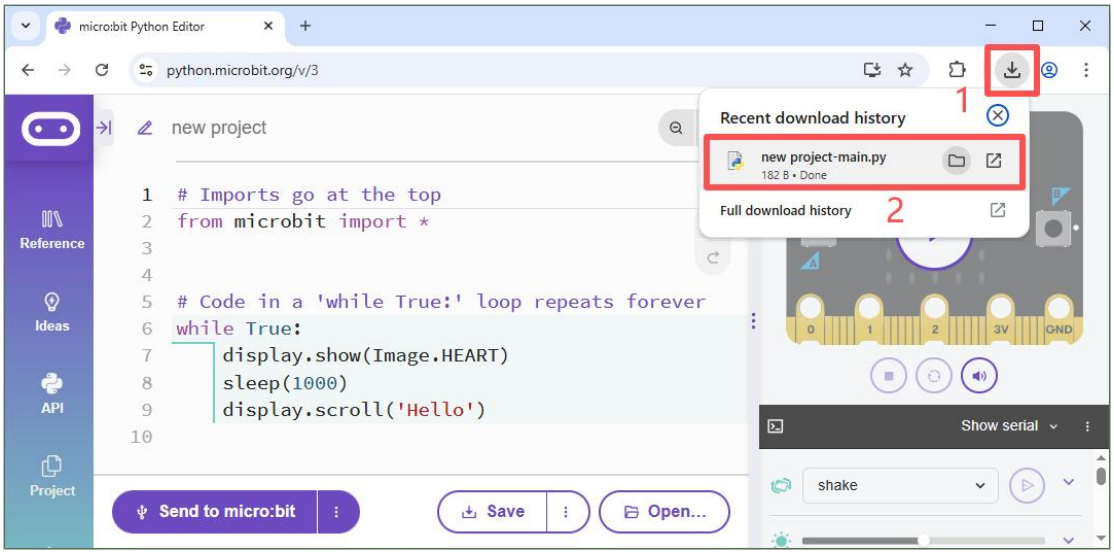
micro:bitへのアップロード: .hexファイルをmicro:bitのUSBドライブにドラッグ＆ドロップ（または送信）すると、micro:bitはそれをメモリにロードして実行を開始します。

プロジェクトのバックアップと共有: .hexファイルを保存して、すぐに実行可能なプロジェクトバージョンをバックアップしたり、他のユーザーと共有できます。他のユーザーはソースコードにアクセスせずに直接使用できます。

2. プロジェクトを.pyファイルとして保存

「**Save**」ボタン横の三点リーダーをクリックし、「**Save Python script**」ボタンをクリックすると、.pyファイルがコンピューターのダウンロードフォルダに保存されます。





これは記述されたPythonコードのテキストです。

機能（Functions）：

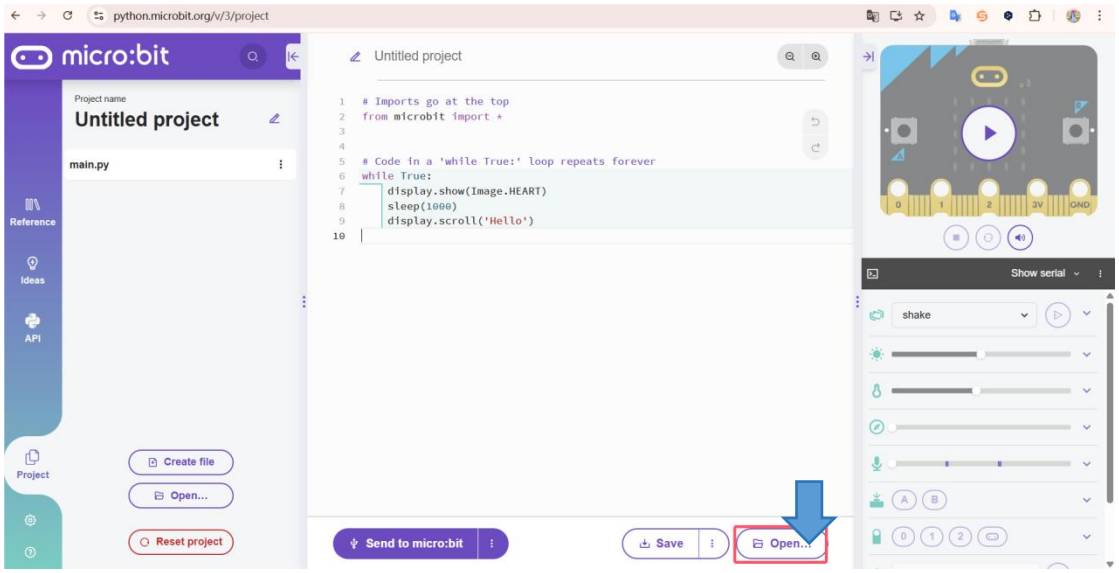
編集と作成: micro:bit Python Editorまたは任意のテキストエディタでプログラムの記述、修正、デバッグを行います。

覚えておいてください（Remember）: micro:bitは.hexファイルを実行しますが、開発は.pyファイルで行います。

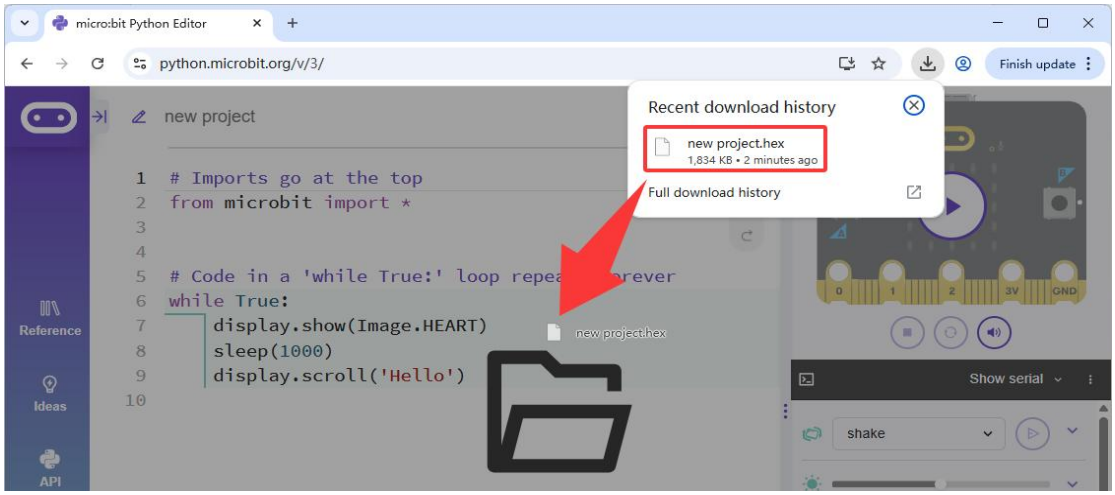
3.3 ファイルのインポート

プログラム（例：以前保存したもの、教師や第三者から提供されたもの）を開くには、「Open」ボタンを選択します。

ここから、開きたい.hexまたは.pyファイルを選択し、「open」をクリックします。



または、.hexファイルまたは.pyファイルをエディタにドラッグ＆ドロップすることもできます。



ファイルインポート時（When importing files）：

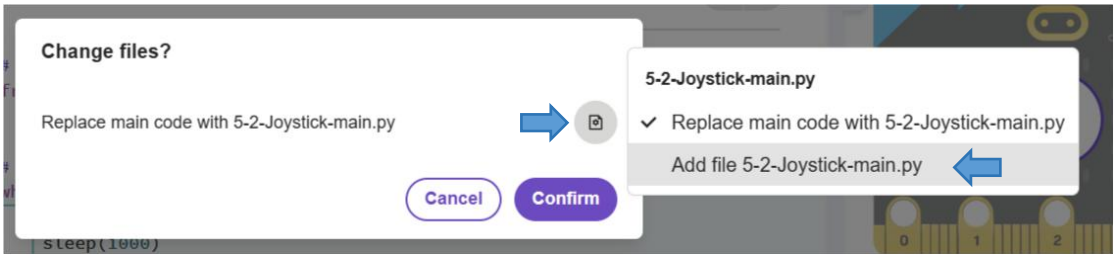
新しい.hexファイルを開くと、警告なしに現在のプロジェクトが直ちに置き換えられ、この操作は元に戻せません。コードの紛失を避けるため、必ず事前に作業内容を保存してください。

新しい.pyファイルを開くと、「Change files?」というプロンプトが表示されます。その後、以下の選択ができます：

Confirmを選択して現在のコードを置き換える、

Cancelを選択して現在のプロジェクトを維持する、

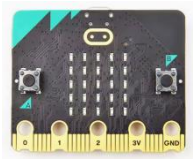
Addを選択して新しいファイルを既存のコードへの拡張として挿入する。



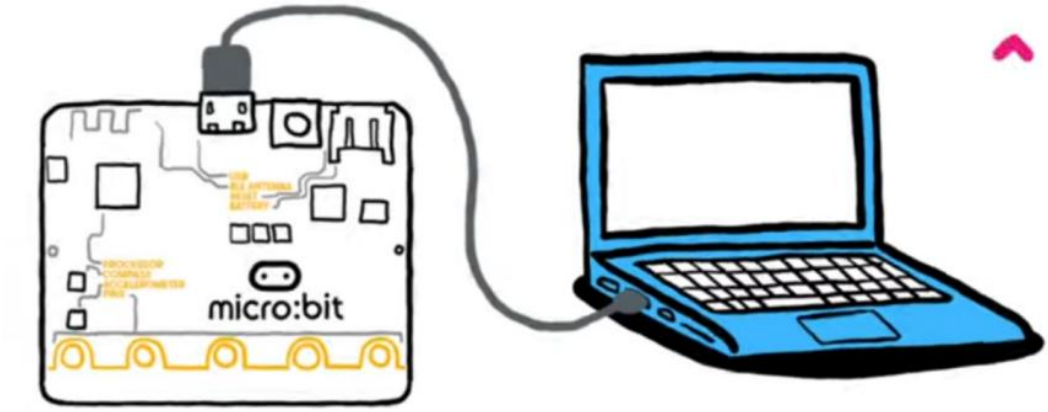
注意（Note）: micro:bit Python Editorは単一プロジェクトモデルで動作し、一度に開いてアクティブにできるプロジェクトは一つだけです。マルチタブ閲覧、複数ファイルの並列編集、組み込みのプロジェクト切り替えはサポートされていません。新しいプロジェクトをロードすると現在のプロジェクトが置き換えられます。この合理化された集中型のアプローチは、初心者向けにインターフェースを簡素化します。複数のプロジェクト間の切り替えによる混乱を避け、「一度に一つのこと」という教育哲学に沿うことで、生徒の認知的負荷を軽減し、現在のタスクに集中させることができます。

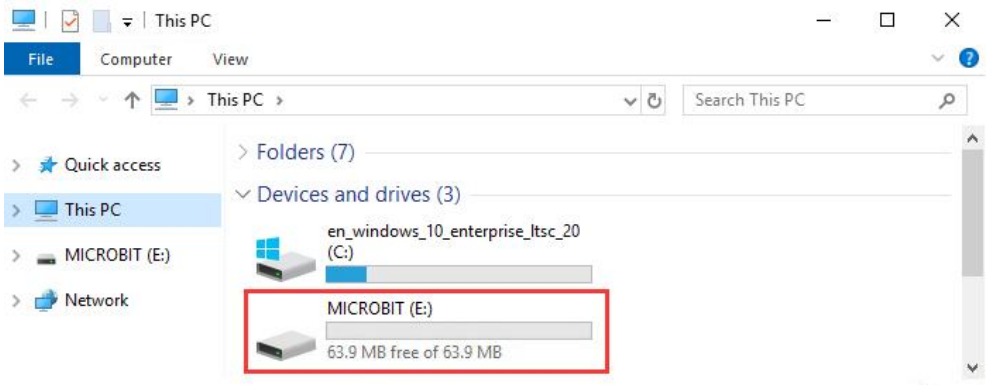
3.4 コードのアップロード

必要なもの

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

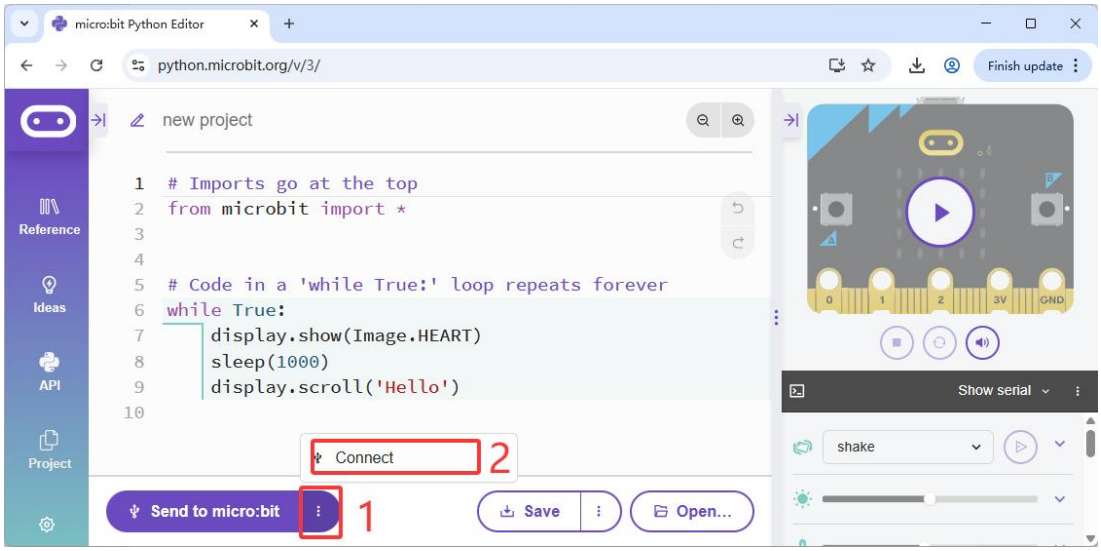
Micro USBケーブルを使用してmicro:bitボードをPCに接続します。PC上に「MICROBIT」という新しいUSBドライブが表示されます：



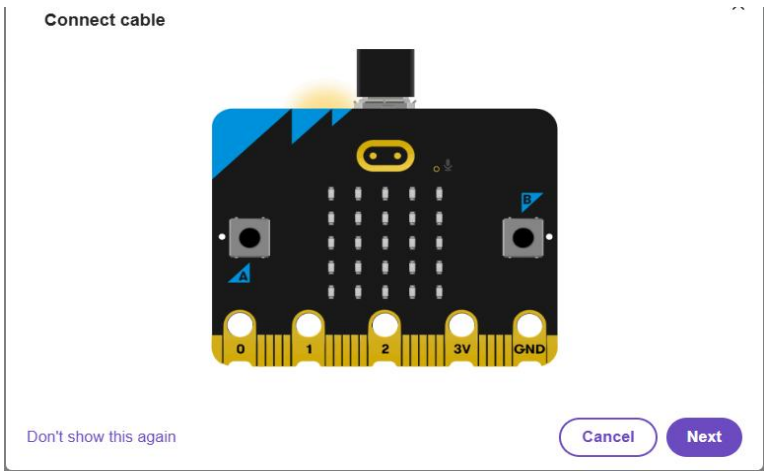


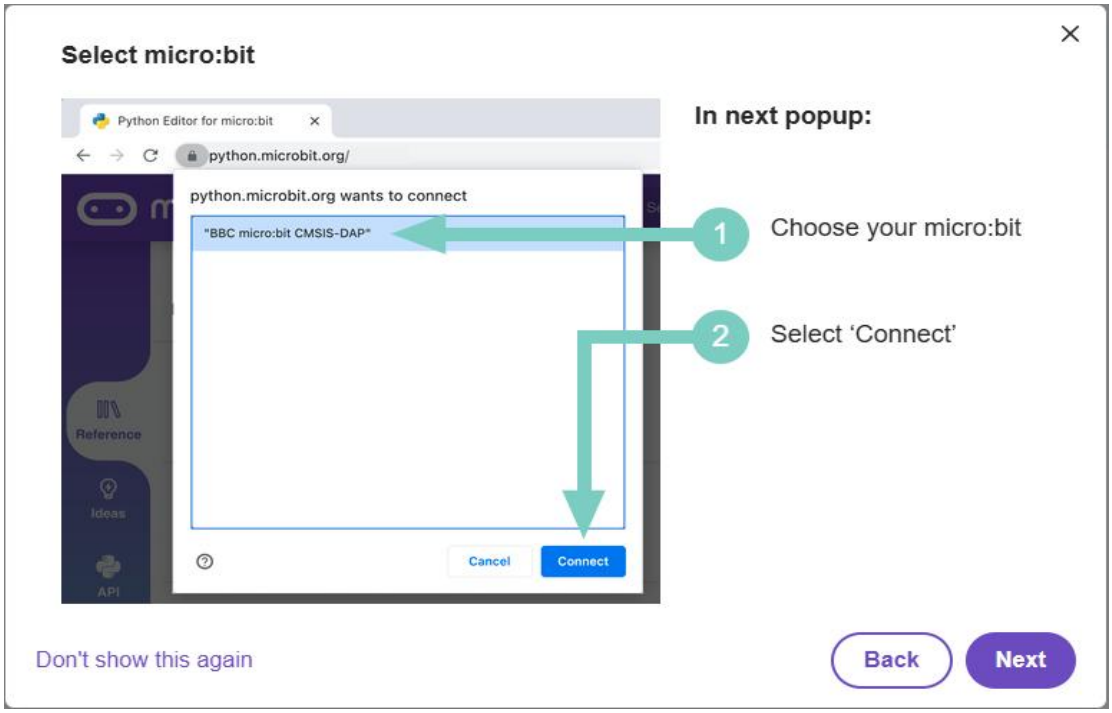
直接フラッシュ（WebUSB）：Google ChromeやMicrosoft Edgeなどの互換性のあるブラウザを使用している場合、WebUSBを使用してmicro:bitをエディタに直接接続し、ファイルのドラッグ＆ドロップなしでプログラムをフラッシュできます。

「Send to micro:bit」ボタン横の三点リーダーをクリックし、「Connect」ボタンをクリックします：

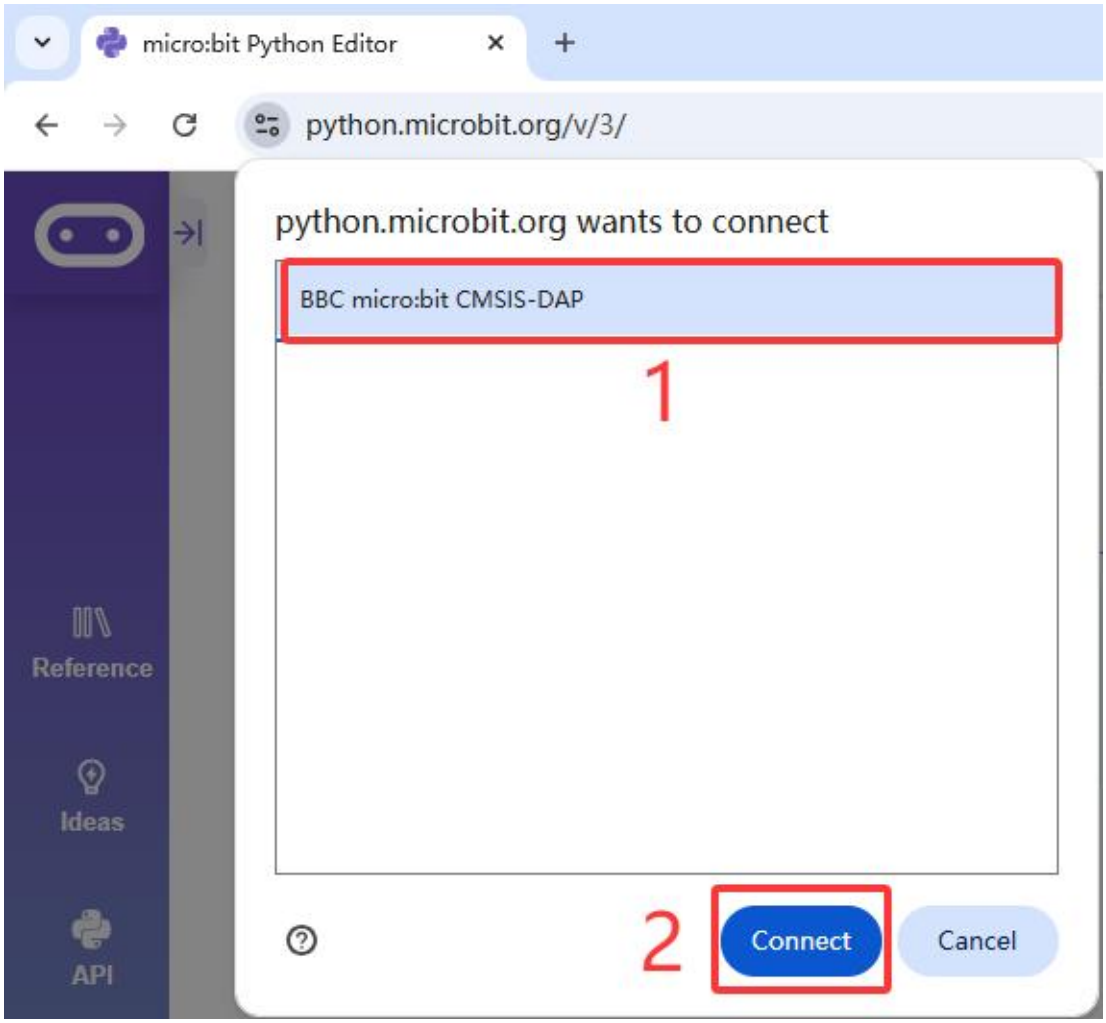


「Next」をクリックします：

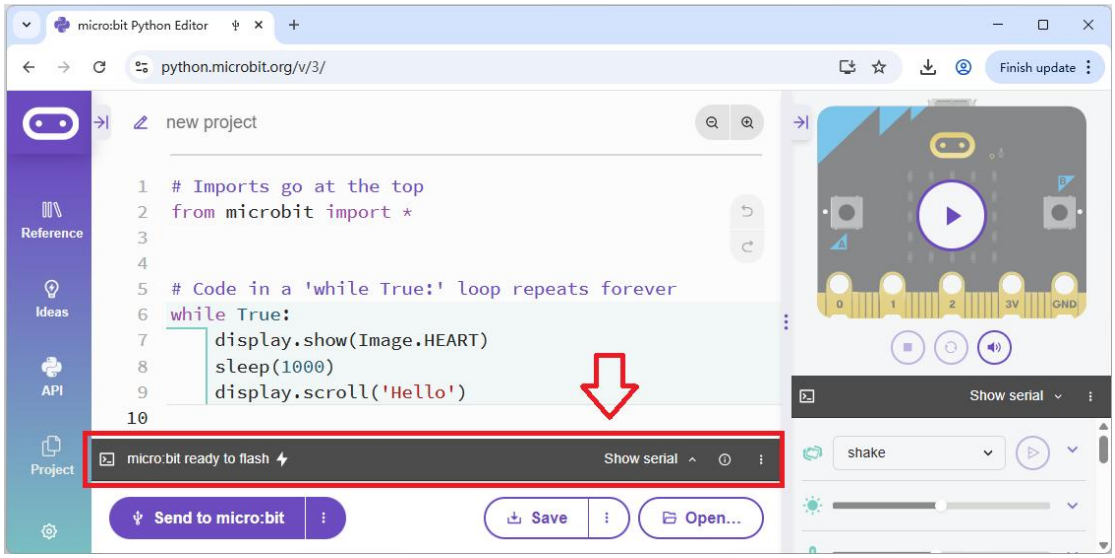




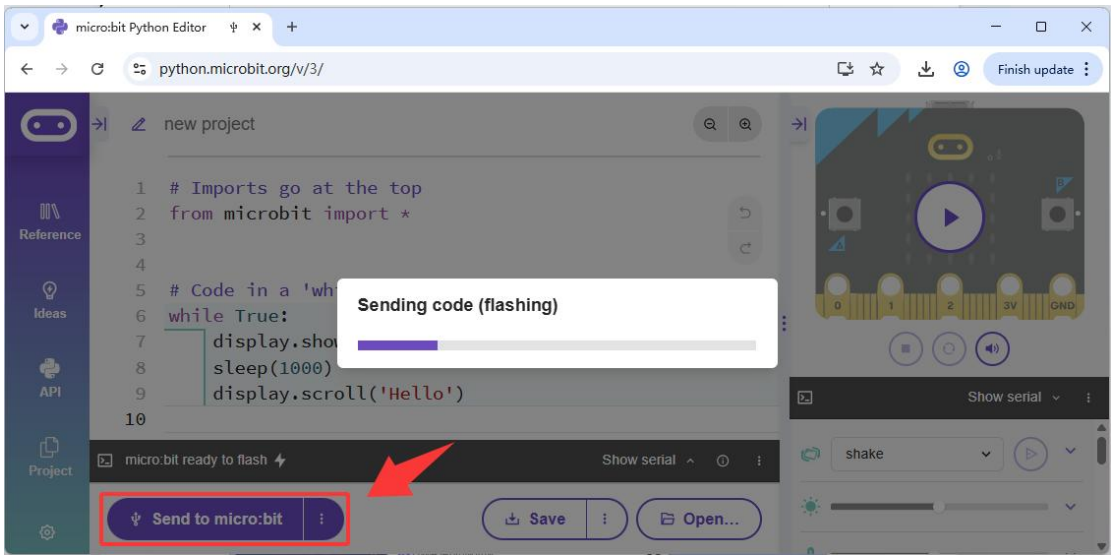
micro:bitを選択し、「**Connect**」をクリックします。



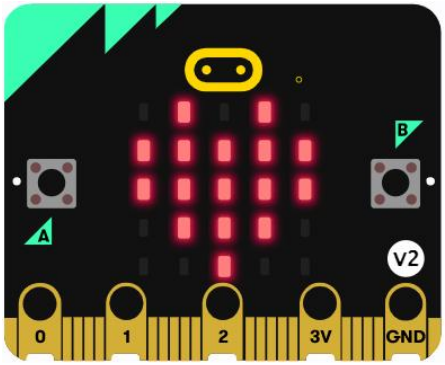
「micro:bit ready to flash」インターフェースが表示されたら、micro:bitがPython Editorに正常に接続されたことを示します。



これで「Send to micro:bit」をクリックして、Python Editorからmicro:bitにコードをアップロードできます：

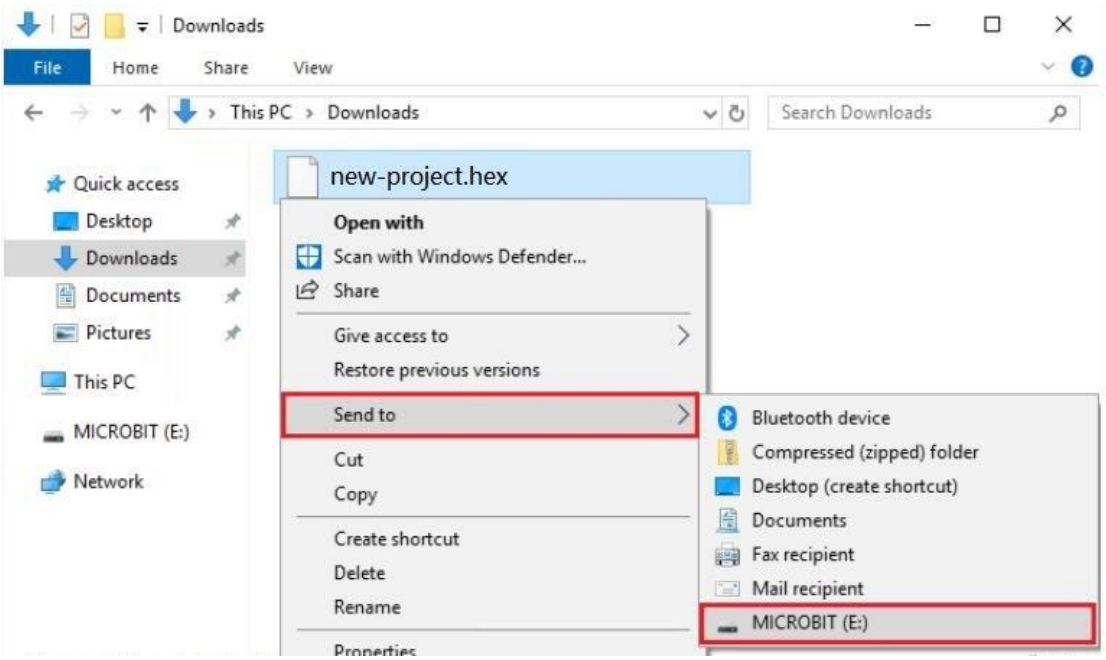


コードがアップロードされると、micro:bitのLEDマトリックスにハートアイコンが表示され、続いて「Hello」というテキストがスクロールします：



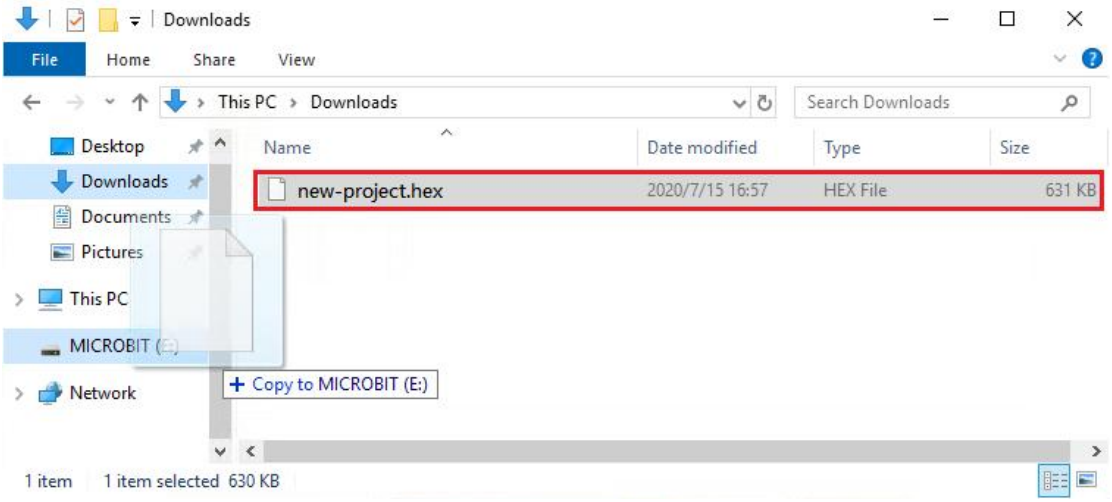
[コードアップロードの2つの代替方法

1. 右クリックで送信（Send via Right-Click）
2. 「.hex」ファイルを選択し、右クリックしてmicro:bitドライブに「Send to」を選択します：

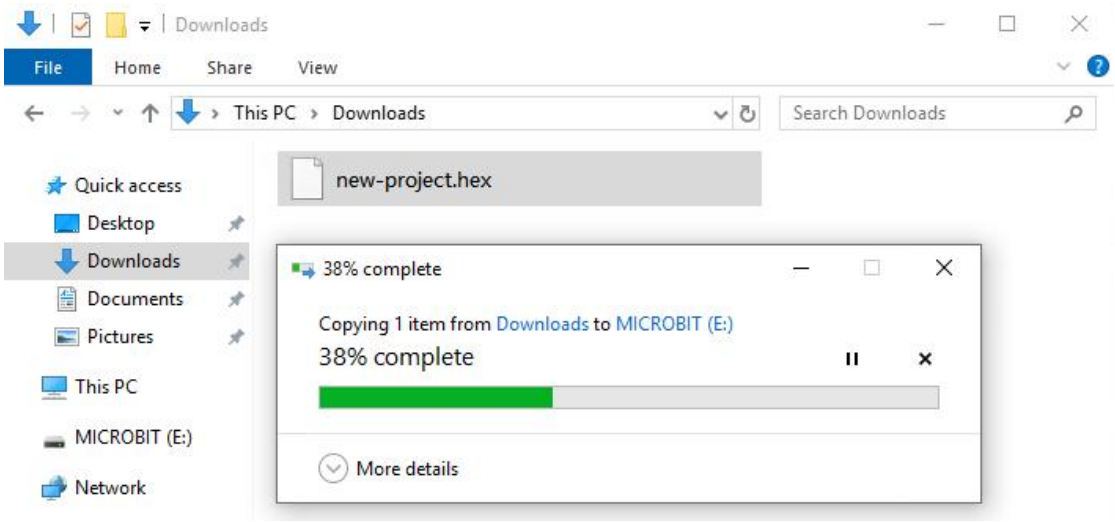


3. ドラッグ＆ドロップ（Drag and Drop）

または、直接「.hex」ファイルをmicro:bitドライブにドラッグ＆ドロップします：



次のインターフェースが表示され、コードがアップロード中であることを示します：



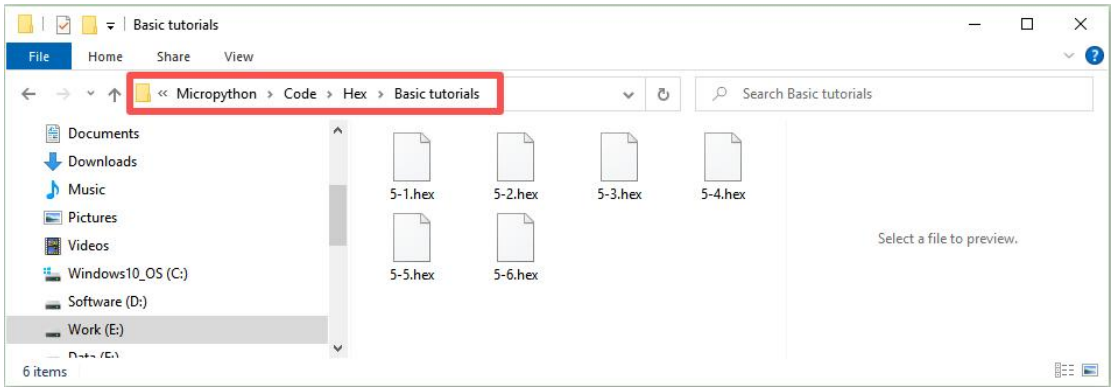
3.5 micro:bit Python Editor の基本構文を学ぶ

Micro:bitプラットフォームは、Python Editorの公式リファレンスドキュメントを提供しています。

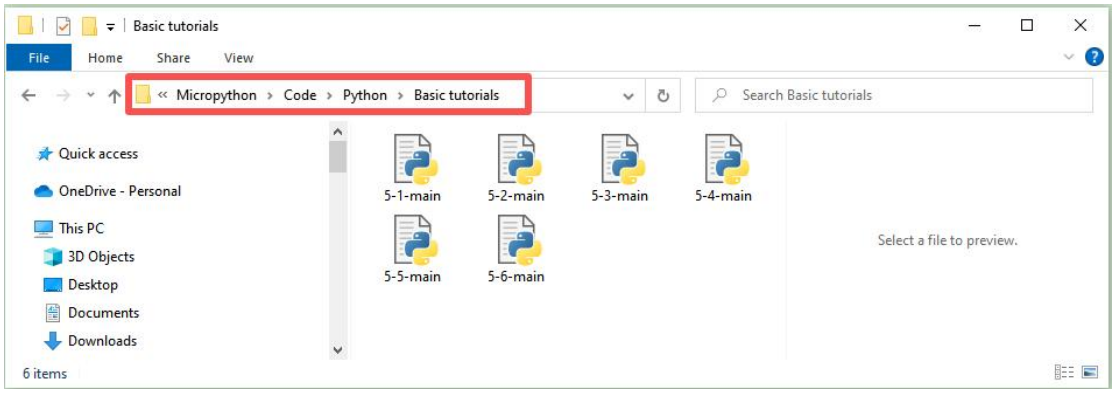
ユーザーガイド：
https://microbit.org/get-started/user-guide/python-editor/
MicroPythonドキュメント：
https://microbit-micropython.readthedocs.io/en/v2-docs/

4. 基本的なサンプルプロジェクト

すべての基本的なサンプルhexコードは、[Micropython → Code → Hex → Basic tutorials]フォルダに保存されています。



すべての基本的なサンプルPythonファイルは、[Micropython → Code → Python → Basic tutorials]フォルダに保存されています。



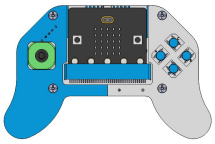
セクション**3.3:** ファイルのインポート（**Import Files**）を参照してください。チュートリアルパッケージからプロジェクトをエディタにインポートできます。

4.1 ボタン（Button）

目標

ジョイスティックのA、B、C、Dボタンの値を読み取ります。

必要なハードウェア

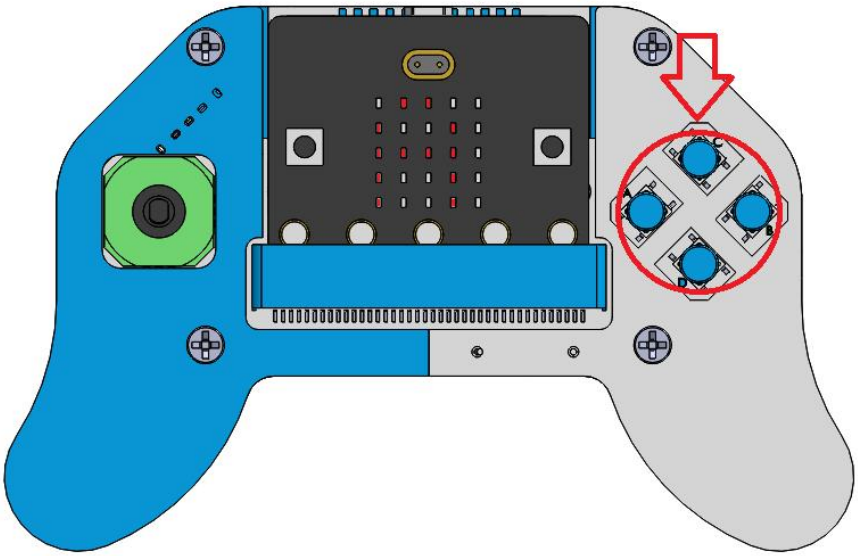
PC	Joystick with micro:bit v2	Micro USB cable
		

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)
```

結果

ボタンA、B、C、またはDがそれぞれ押されると、Micro:bitのLEDマトリックスに「A」、「B」、「C」、または「D」が表示されます。

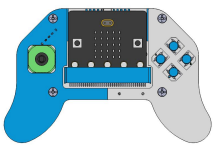


4.2 Joystick

目標

ジョイスティックのアナログ値をリアルタイムで読み取り、ジョイスティックの傾き方向に基づいてmicro:bit LEDマトリックスに対応する方向の文字を表示します。

必要なハードウェア：

PC	Joystick with micro:bit v2	Micro USB cable
		

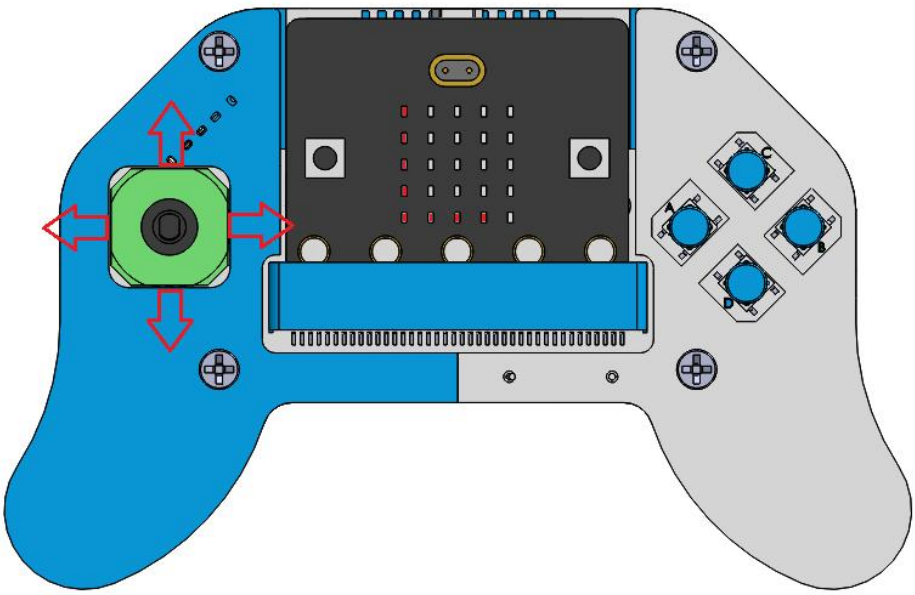
Code

```
# 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)
```

結果

ジョイスティックを左、右、上、下に押すと、Micro:bitのLEDマトリックスにそれぞれ「L」、「R」、「U」、「D」が表示されます。

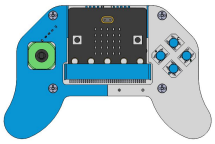


4.3 振動モーター

目標

ジョイスティックの振動モーターを作動させます。

必要なハードウェア：

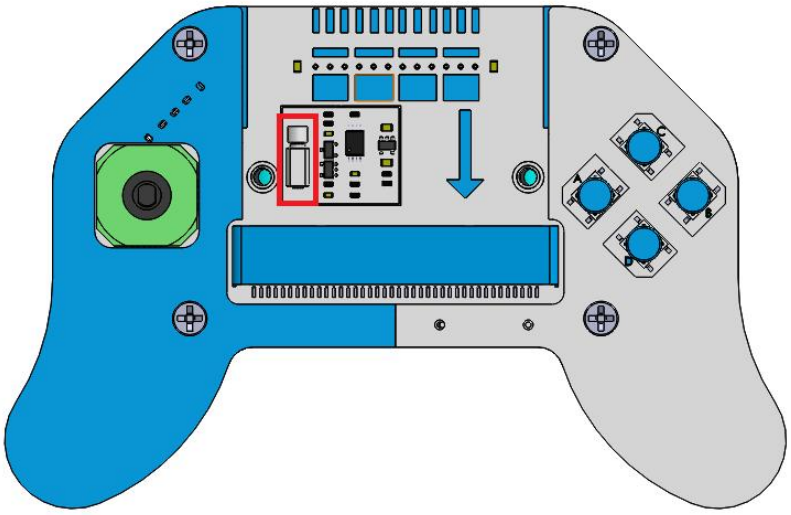
PC	Joystick with micro:bit v2	Micro USB cable
		

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)
```

結果

ジョイスティック上の振動モーターが、1秒ごとに1秒間振動します。

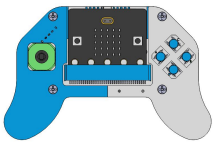


4.4 電池電圧

目標

ジョイスティックの電池残量を読み取ります。

必要なハードウェア：

PC	Joystick with micro:bit v2	Micro USB cable
		

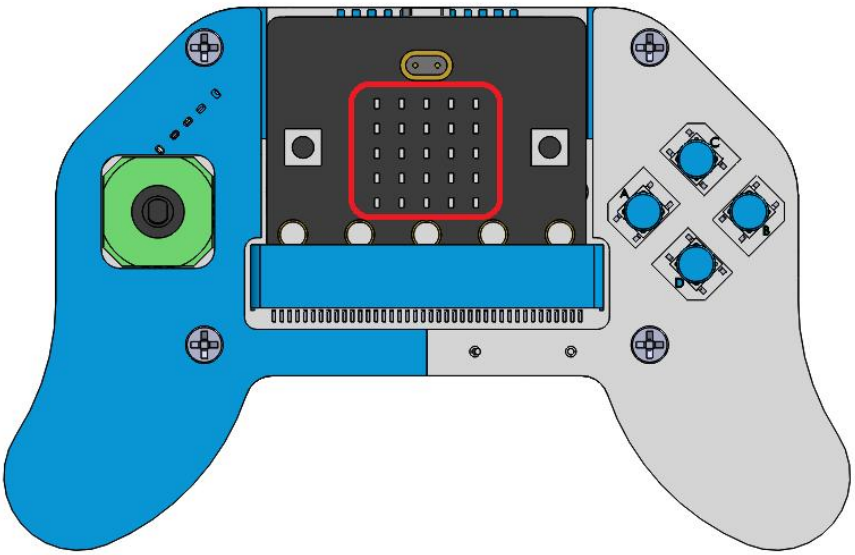
Code

```
# 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)
```

結果

Micro:bit LEDマトリックスに電池残量の値がそれぞれ表示されます。範囲は0～100です。



注意

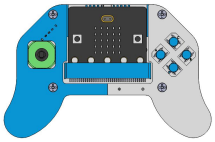
電池電圧の読み取りと振動モーターの制御は、Micro:bitの同じP2ピンを共有しています。電池電圧の読み取りと振動モーターの制御を頻繁に切り替えると、電池電圧読み取りの精度に影響する可能性があります。振動モーターを制御した後、電圧を連続して複数回読み取り、最後の値を取ることを推奨します。

4.5 サーボ

目標

サーボモーターの回転を制御します。

必要なハードウェア：

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

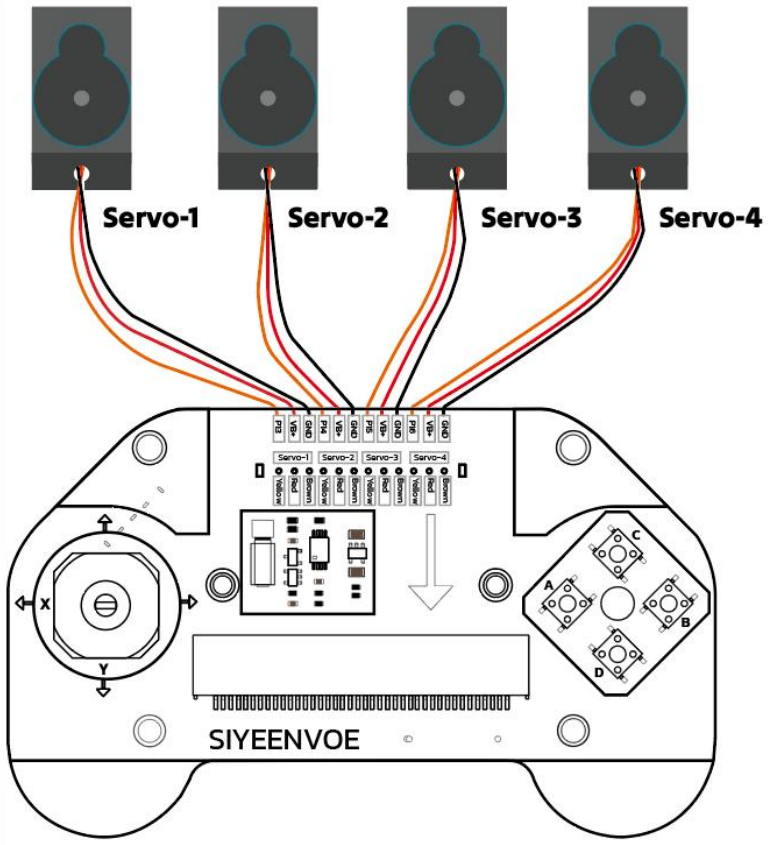
Code

```
# 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)
```

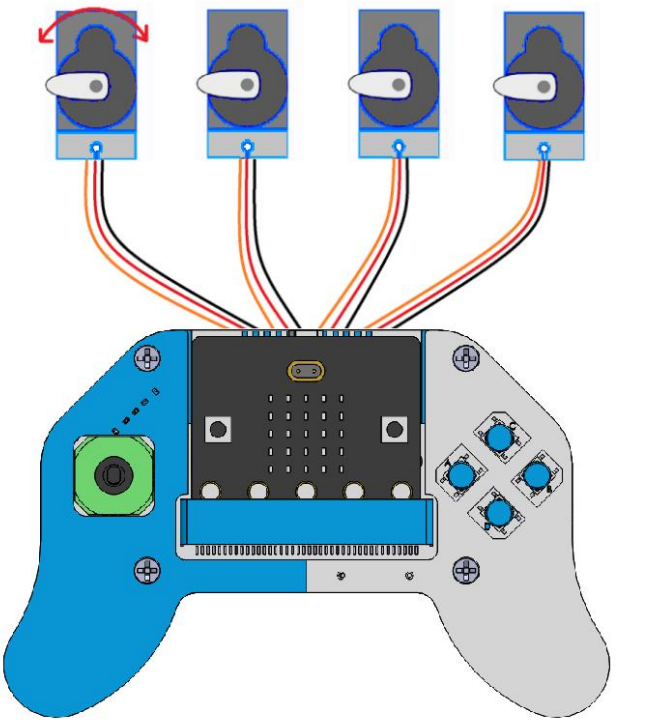
配線：

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



結果

4つのサーボが同期して動作し、4秒ごとに1サイクル（0° → 2秒間保持 → 180° → 2秒間保持 → 0°に戻る）を完了し、その後ステップ1に戻り、無限ループで永久に繰り返します。



注意:

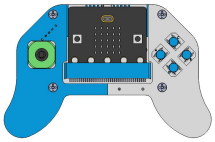
ジョイスティックには単三電池4本を装備し、電源スイッチをオンにする必要があります！

4.6 無線制御

目標

2台のMicro:bitを使用して、無線で相互にデータを送信します。

必要なハードウェア：

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
```

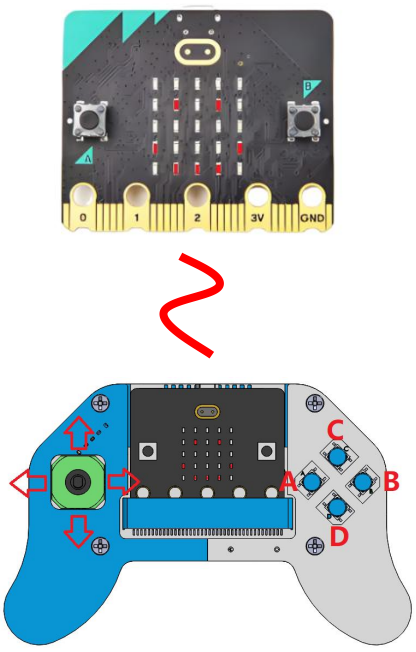
```
display.show(Image.HAPPY)
sleep(400)

# 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)
```

結果

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

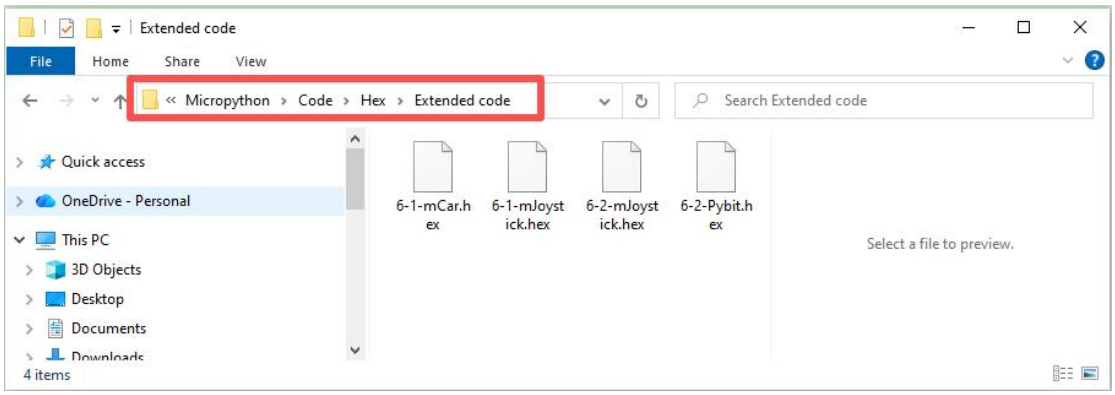


注意:

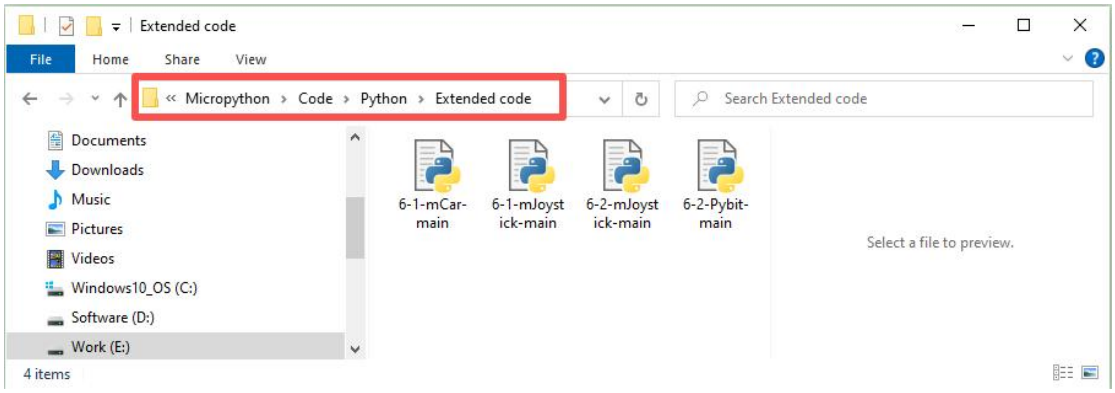
ジョイスティックには単三電池4本を装備し、電源スイッチをオンにする必要があります！

5. 拡張プロジェクト

拡張プロジェクト（Extended Projects）のすべてのサンプルhexファイルは、[Micropython → Code → Hex → Extended code]フォルダに保存されています。



拡張プロジェクト（Extended Projects）のすべてのサンプルPythonファイルは、[Micropython → Code → Python → Extended code]フォルダに保存されています



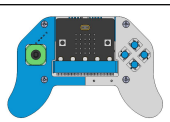
セクション3.3: ファイルのインポート（Import Files）を参照してください。チュートリアルパッケージからプロジェクトをエディタにインポートできます。

5.1 mCar の制御

目標

mCar ロボットを制御します（Micro:bit v2 は含まれません、型番: M1C0000）。

必要なハードウェア：

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

mCarのチュートリアルはこちらからダウンロードできます：

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

mCar ロボットに興味がある場合は、お問い合わせいただき購入してください。

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:

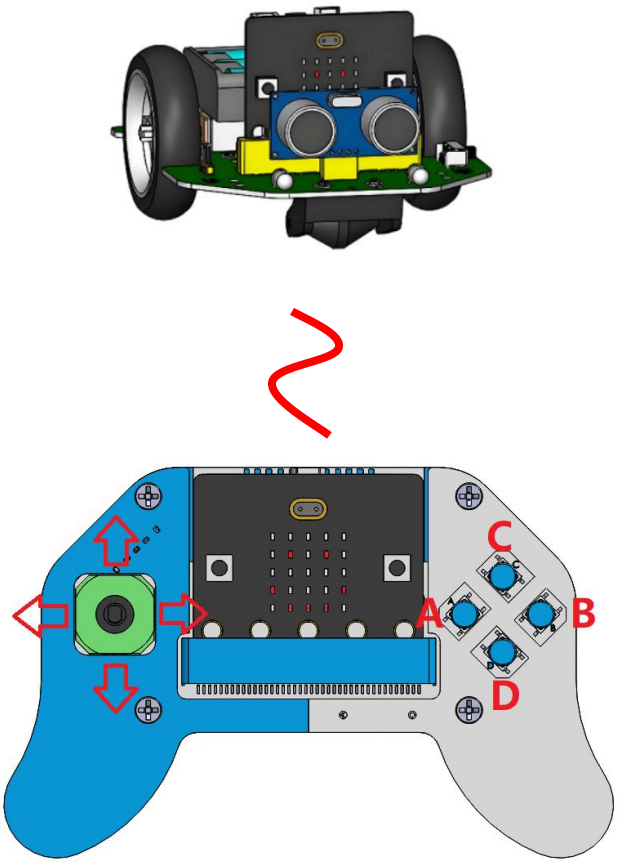
```
# Imports go at the top
from microbit import *
import radio
import speech

# 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)
```

結果



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

注意：

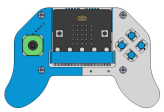
ジョイスティックには単三電池**4**本を装備し、電源スイッチをオンにする必要があります！

5.2 Pybit の制御

目標

Pybit ロボットを制御します（Micro:bit v2 は含まれません、型番: M1C0001）。

必要なハードウェア

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

Pybitのチュートリアルはこちらからダウンロードできます：

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

Pybitロボットに興味がある場合は、お問い合わせいただき購入してください。

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:

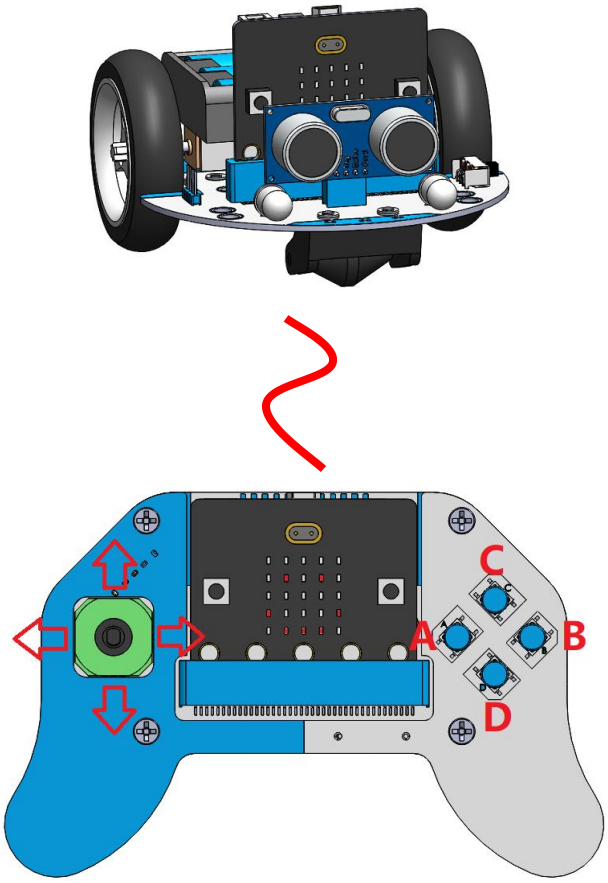
```
# Imports go at the top
from microbit import *
import radio
import speech

# 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 == 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)
```

結果



Joystick Action	Pybit Function
Push joystick up	Car moves forward
Push joystick down	Car moves backward
Push joystick left	Car turns left

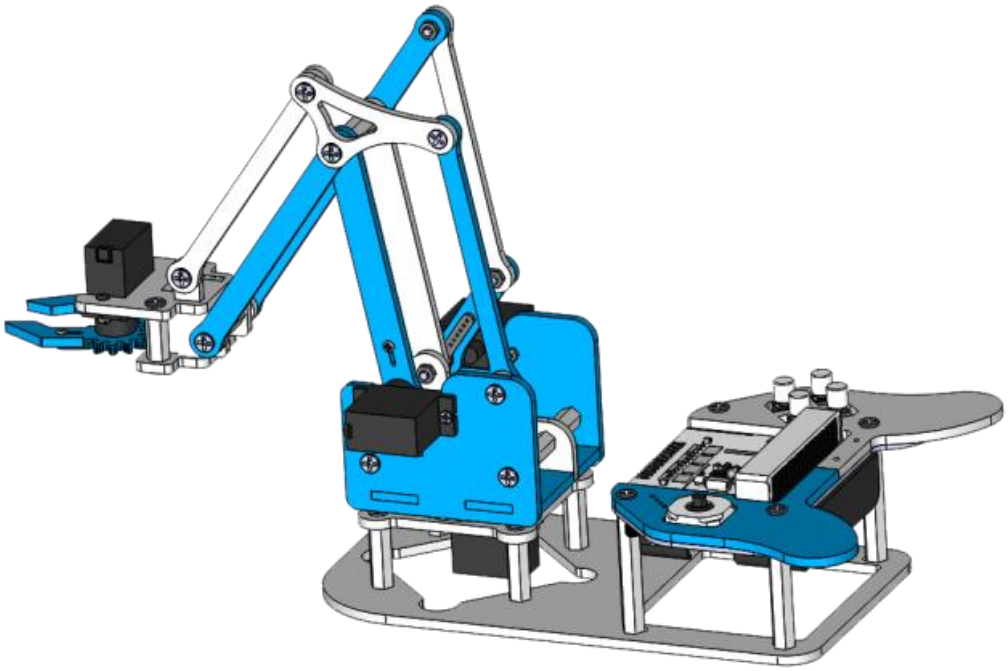
Push joystick right	Car turns right
Press button A	Car beeps
Press button B	Headlights turn green
Press button C	Headlights turn blue
Press button D	Headlights turn off
No action	Car stops

注意：

ジョイスティックには単三電池4本を装備し、電源スイッチをオンにする必要があります！

5.3 ジョイスティックでロボットアームを制御

このジョイスティックは、ロボットアームの制御にも使用できます。こちらが当社で開発した、このジョイスティックを含むmArm 4DF ロボットアームです（Micro:bit v2は含まれません、型番: M1R0000）。mArmキットまたはジョイスティックを除いた付属品をご購入希望の場合は、お問い合わせください。



mArmロボットアームのチュートリアルはこちらからダウンロードできます：
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 コードのアップロード失敗

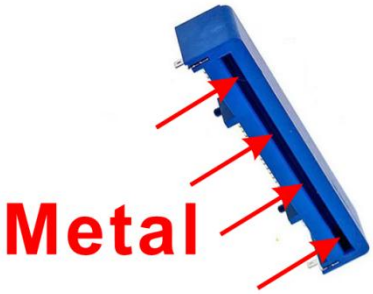
☛ USBケーブルがデータ転送をサポートしているか確認してください（充電専用ケーブルは動作しません）
☛ USBケーブルとポートの物理的な損傷や接続問題を点検してください

6.2 Micro:bit ドライブの異常表示

☛ ドライブが「MAINTENANCE」（「MICROBIT」ではなく）と表示される場合は、誤ってファームウェア回復モードに入ったことを示します。解決方法：
☛ 公式ツールを使用してファームウェアを再フラッシュしてください：
https://microbit.org/get-started/user-guide/firmware/
☛ 電源投入時にリセットボタンを押さないようにして再発を防止してください

6.3 ボタンまたはジョイスティックが反応しない

☛ Micro:bitがmJoystickスロットに完全に挿入されていることを確認してください。
☛ Micro:bitのエッジコネクタが清潔であることを確認してください。
☛ mJoystickスロットが清潔であることを確認してください



6.4 その他の問題

☛ 電池の電力が十分か確認してください。
☛ 使用している電池が要求仕様を満たしているか確認してください

7.お問い合わせ

上記で解決策が見つからなかった場合は、サポートチームまでお問い合わせください。

迅速なサポートのために、以下の情報をご用意ください：
ご注文番号。
製品モデル（例：Robot Arm Kit M1K0000）とソフトウェアバージョン
問題や質問の詳細な説明。
既に試した手順。
関連するエラーメッセージのスクリーンショット、写真、コードスニペット。

その他のお問い合わせ？
以下の事項についてもお気軽にお問い合わせください：
チュートリアルの誤りとフィードバック：ドキュメントの改善にご協力ください。
製品アイデアと提案：素晴らしいアイデアをお聞かせください。
パートナーシップとコラボレーション：一緒に働きませんか？ 話しましょう。
割引とプロモーション：教育用または大口価格についてお問い合わせください。
その他：技術的な問題以外のご質問はすべて。

✉ support@siyeenove.com

https://siyeenove.com