# 目次

# 1. チュートリアル説明

**1.1** Python は、強力なハードウェア（デスクトップやサーバーなど）で実行される汎用プログラミング言語です。MicroPython は、マイクロコントローラーや組み込み機器向けに設計された Python の簡略版です。

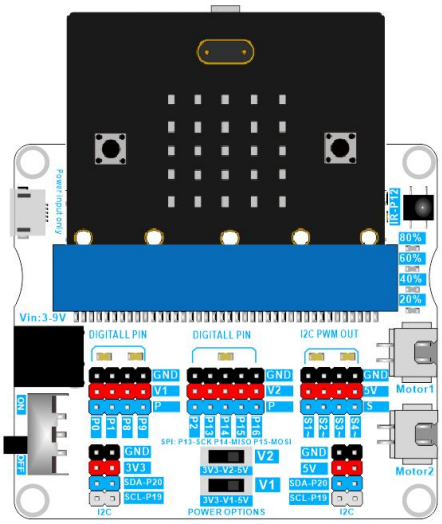
**1.2** このチュートリアルは、MicroPython を使用してプログラミングを行います。Micro:bit には専用の NEC 赤外線 MicroPython ライブラリがありません。Pin を使用して NEC 赤外線信号を読み取る場合、読み取り速度が遅すぎて、信号の損失が深刻になります。したがって、このチュートリアルでは NEC 赤外線リモコンを実装できません。



# 2. Pybit 紹介

mShield 拡張ボードは、当社が最新に発表した使いやすい micro:bit 拡張ボードです。2 チャンネルモータードライブ、赤外線受信、バッテリーレベル読み取り、4 チャンネルサーボドライブ、4 チャンネル PWM 信号出力、LED インジケーター、選択可能な 3V または 5V 電源出力などの強力な機能を統合しています。

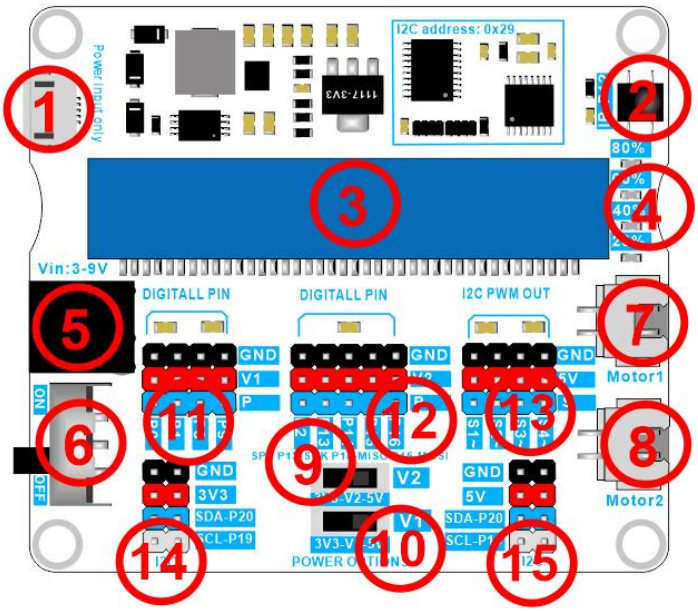
拡張ボードは、micro:bit の常用ピンをピンヘッダーの形で拡張し、他のセンサーに接続することができます。拡張ボードは 2 種類の多セルバッテリーに接続でき、また micro USB ポートを予備しているため、モバイルバッテリーにも接続でき、電源の適応性がより強く、より安全です。制御ボードを他の装置に固定するのを容易にするために、拡張ボードには 4 つの固定穴が付いており、他のプロジェクトで使用するのが簡単です。



# 3. 仕様パラメータ

| | |
|---|---|
| **Shield** | |
| Name | mShield |
| SKU | M1E0002 |
| **USB** | **Applicable motherboard** |
| Micro USB | Micro:bit |
| **Pins** | |
| Digital I/O Pins | 9 (P0, P1, P2, P8, P9, P13, P14, P15, P16) |
| Analog read pins | 3 (P0, P1, P2) |
| Analog write pins | 9 (P0, P1, P2, P8, P9, P13, P14, P15, P16) |
| PWM pins | 4 (S1~, S2~, S3~, S4~), period: 2ms. |
| Servo pins | 4 (S1~, S2~, S3~, S4~) |
| **Communication** | |
| UART | Yes |
| I2C | Yes |
| SPI | Yes |
| IR receiver | Yes (NEC) |
| **Power** | |
| Input voltage (nominal) | 3--9V |
| DC Current for 3.3V Pin | 500 mA |
| DC Current for 5V Pin | 2000 mA |
| **Motors** | |
| Connectors | Motor1, Motor2 (XH2.54 2P) |
| Output voltage | 3--9V |
| Maximum drive current | 1.2A |
| Recommended driving | 1A |
| **LEDs** | |
| Number | 4 (20%, 40%, 60%, 80%) |
| **Dimensions** | |
| Width | 62 mm |
| Length | 73 mm |
| Height | 14mm |
| **Weight** | 29.16 g |

# 4. インターフェース仕様




| 番号 | 説明 |
|---|---|
| 1 | Micro USB ポート、外部電源供給専用。 |
| 2 | 赤外線受信機、P12 ピンを使用。 |
| 3 | Micro:bit スロット。 |
| 4 | LED インジケーター、内部 I2C チップで制御。 |
| 5 | 電源ソケット(DC005)、3-9V 電圧を入力可能、3、4、5、6 本の単三電池、または 1、2 本のリチウム電池に接続可能、逆接続保護機能付き。 |
| 6 | 電源ソケットスイッチ。 |
| 7、8 | モーターインターフェース、3-9V DC モーターに接続可能、モーター電圧は電源ソケット電圧と等しい。内部 I2C チップで制御。 |
| 9、10 | V1 および V2 ピンヘッダー出力電圧選択スイッチ。 |
| 11、12 | Microbit IO 拡張ポート。 |
| 13 | mShield 内部拡張ポート、内部 I2C チップで制御。 |
| 14 | 3.3V 電源 I2C インターフェース、P19(SCL), P20(SDA)。 |
| 15 | 5V 電源 I2C インターフェース P19(SCL), P20(SDA)。 |

# 5. Python エディタの基本

Python エディタは、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。Python エディタは無料で、すべてのプラットフォームのブラウザで実行されます。

Google Chrome または Microsoft Edge ブラウザの使用を推奨します。WebUSB は、Web ページから直接 Micro:bit にアクセスできる新しく開発された Web 機能です。また、Micro:bit から直接 Python エディタにデータを送信することもできます。Google Chrome および Microsoft Edge ブラウザで動作します。

WebUSB サポート micro:bit バージョン

最新バージョンの Google Chrome または Microsoft Edge を使用していない場合は、以下のバージョン以上であることを確認してください：

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

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

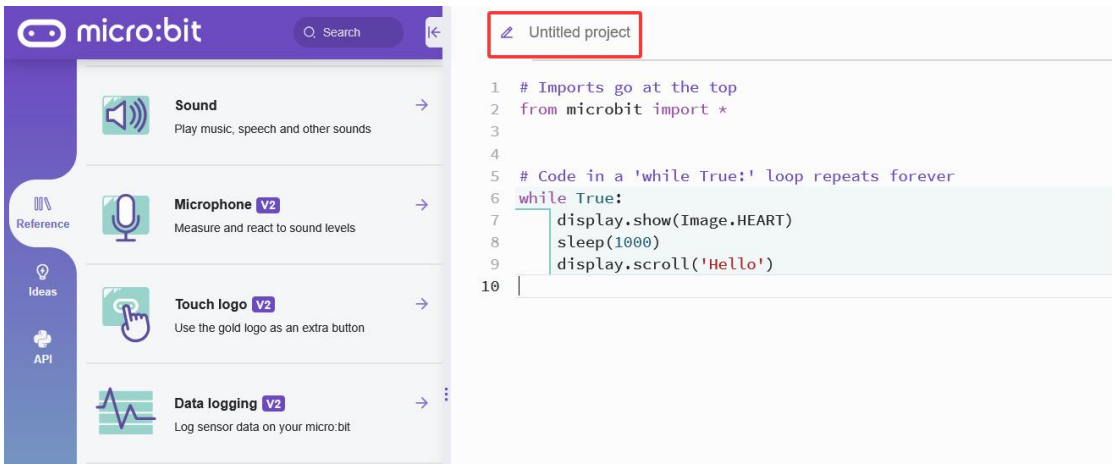
最新の Microsoft Edge をダウンロードするリンク：

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

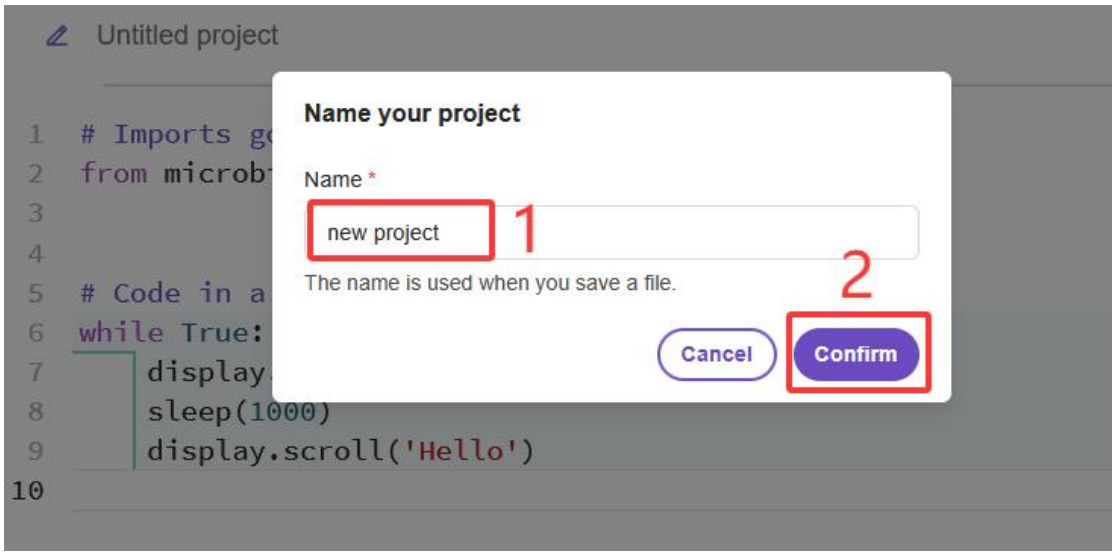
## 5.1 新規プロジェクトの作成

PC の Google Chrome ブラウザでオンラインプログラミングページを開く：

https://python.microbit.org/v/3 、赤枠をクリック：

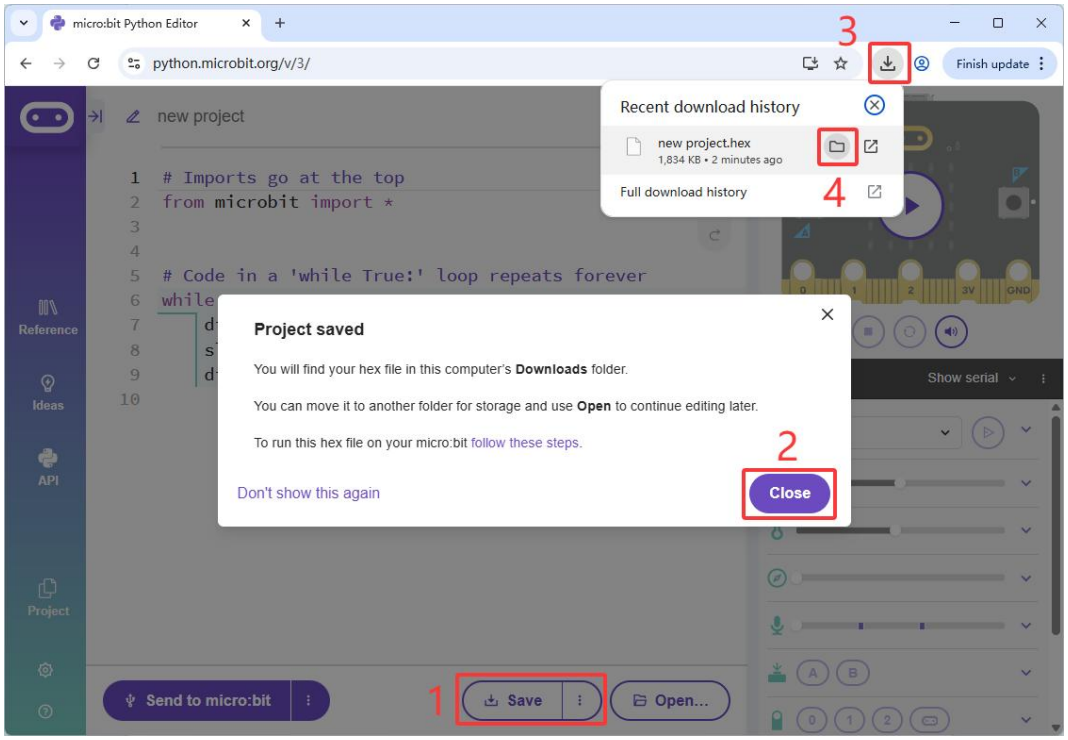


次にプロジェクト名を入力し、「確認」をクリック：




## 5.2 プロジェクトをローカルに保存

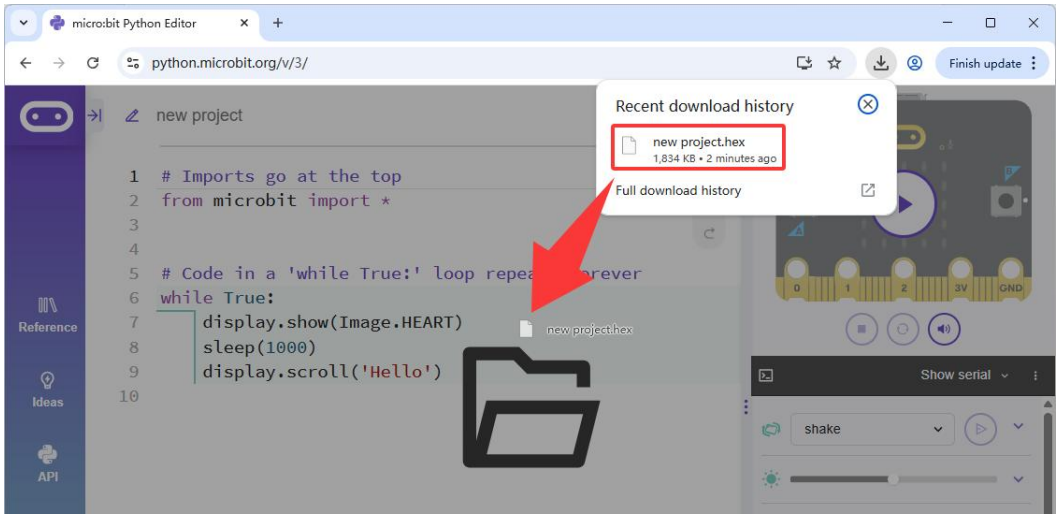
ステップ 1 の「保存」ボタンをクリックし、次にステップ 2 の「閉じる」ボタンをクリックしてプロジェクトをローカルに保存します。次にステップ 3 と 4 に従って保存された「.hex」ファイルを見つけます、下図のように：




**注：ポップアップウィンドウを再表示させたくない場合は、ステップ 2 で「二度と表示しない」をクリックしてください！**

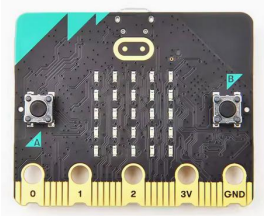
## 5.3 ローカルプロジェクトのインポート

ローカルの「HEX」プロジェクトファイルを Python エディタのワークスペースに直接ドラッグアンドドロップします、下図のように：

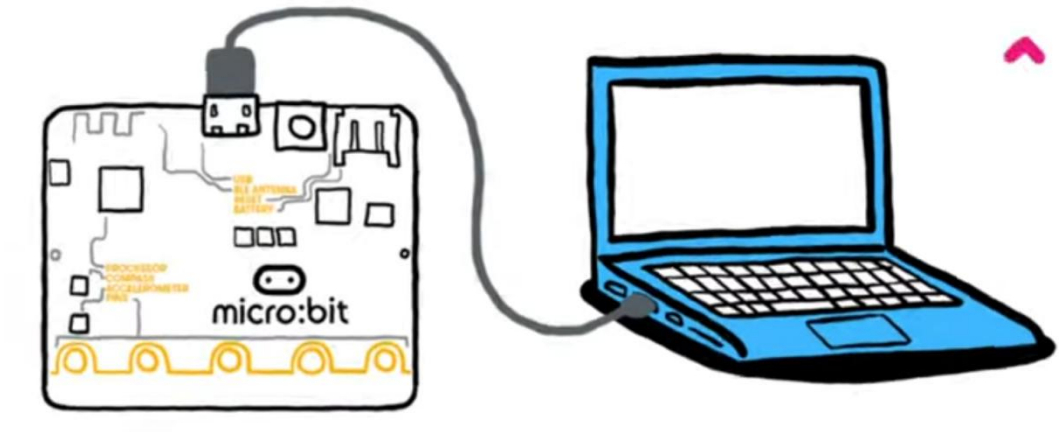


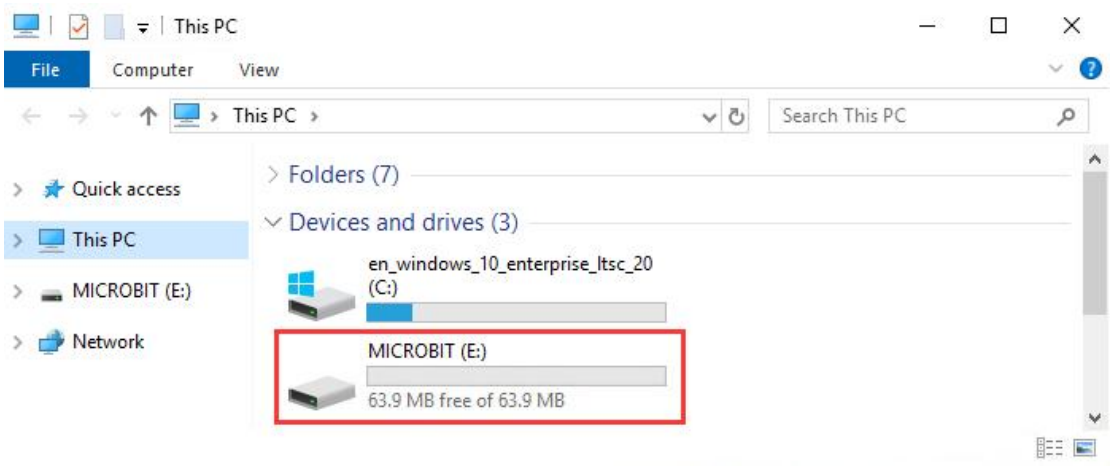

## 5.4 プロジェクトのアップロード

**ツール：**

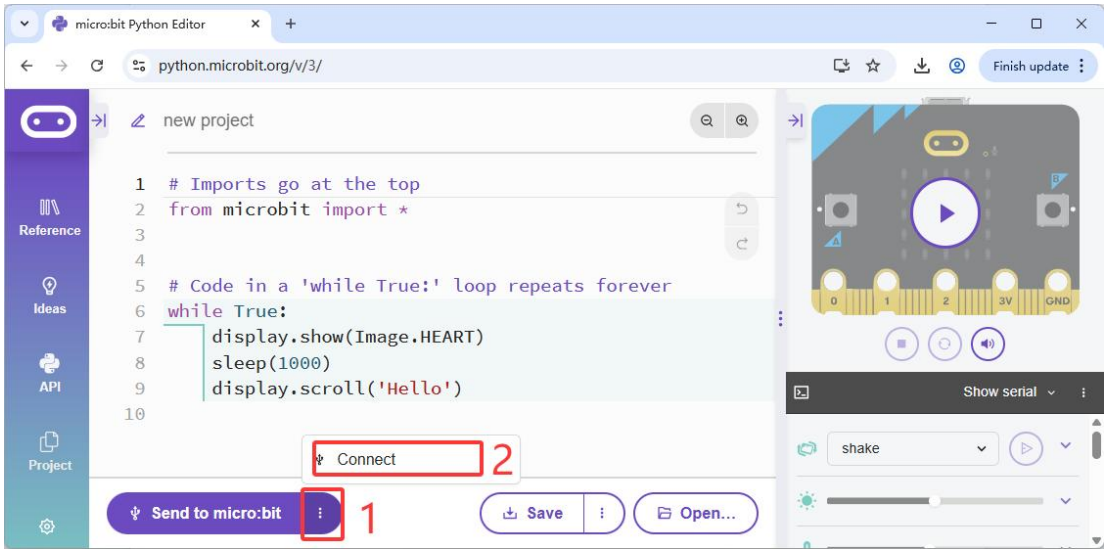
| PC | Micro:bit v2.x.x | Micro USB ケーブル |
|---|---|---|
|  |  |  |

Micro USB ケーブルを使用して Micro:bit マザーボードを PC に接続します。PC にストレージドライブが表示されます：

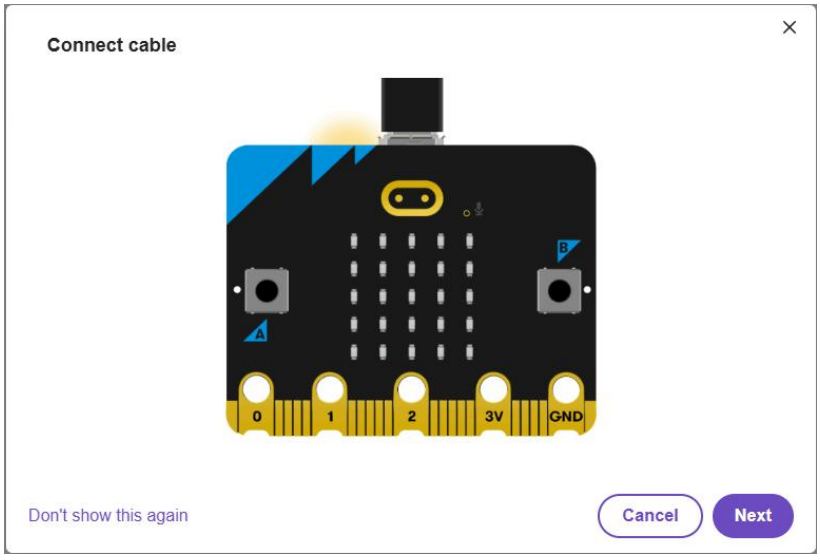




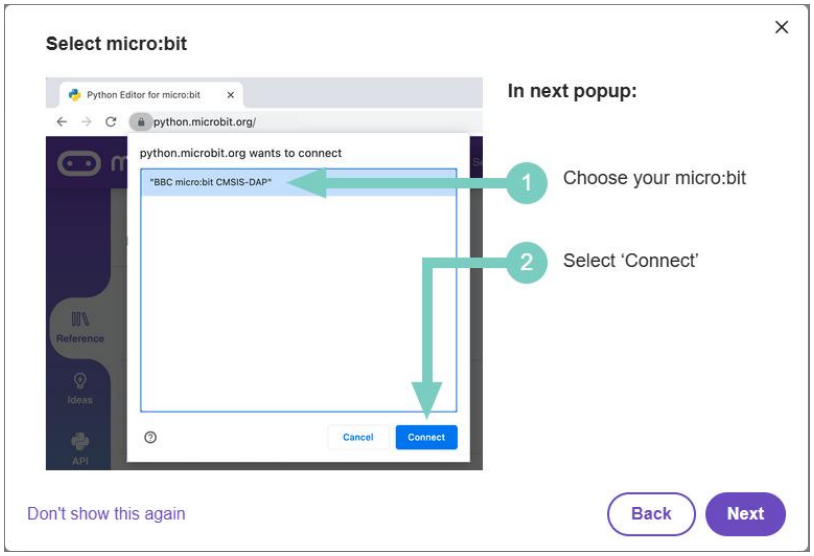

「接続」ボタンをクリック：



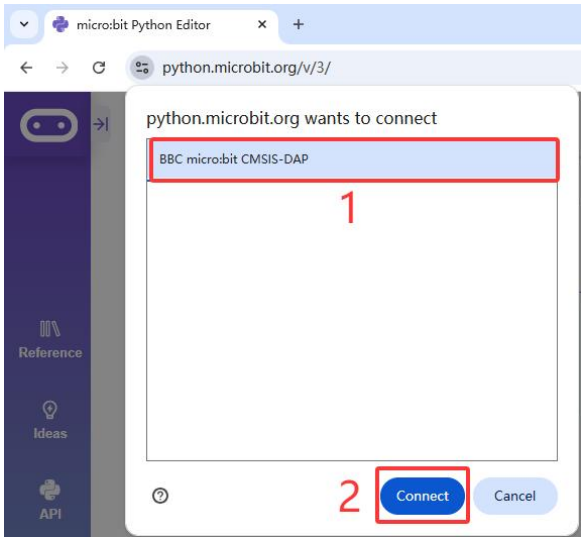

「次へ」をクリック：



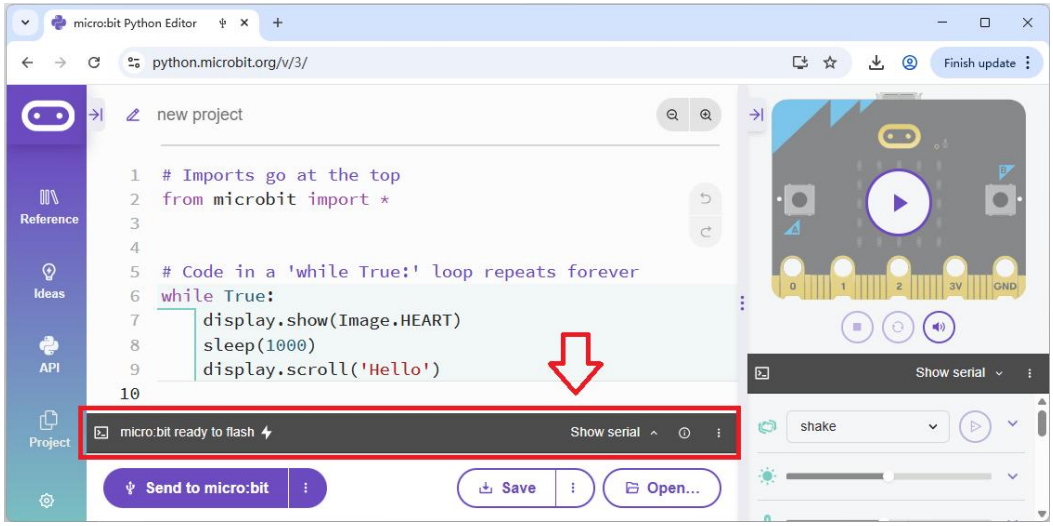
「次へ」をクリック：



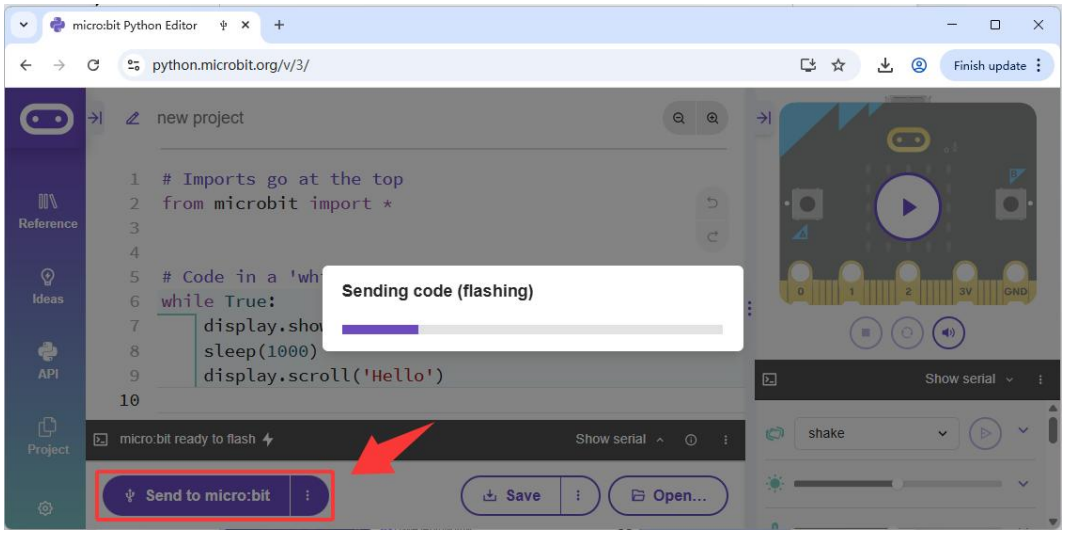

次に以下の手順に従います：



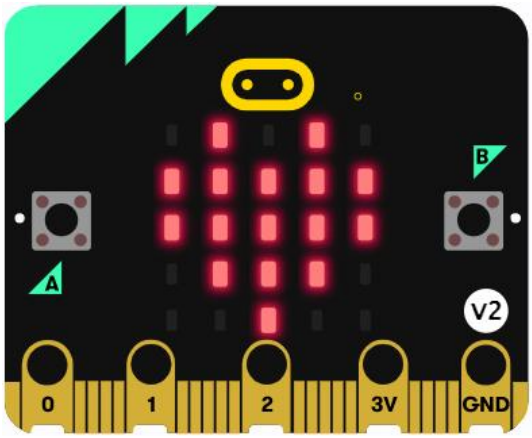

以下が表示されると、Micro:bitとPythonエディタ間の接続が成功したことを示します：



ここで「micro:bit に送信」をクリックして、Python エディタから Micro:bit にコードをアップロードします：

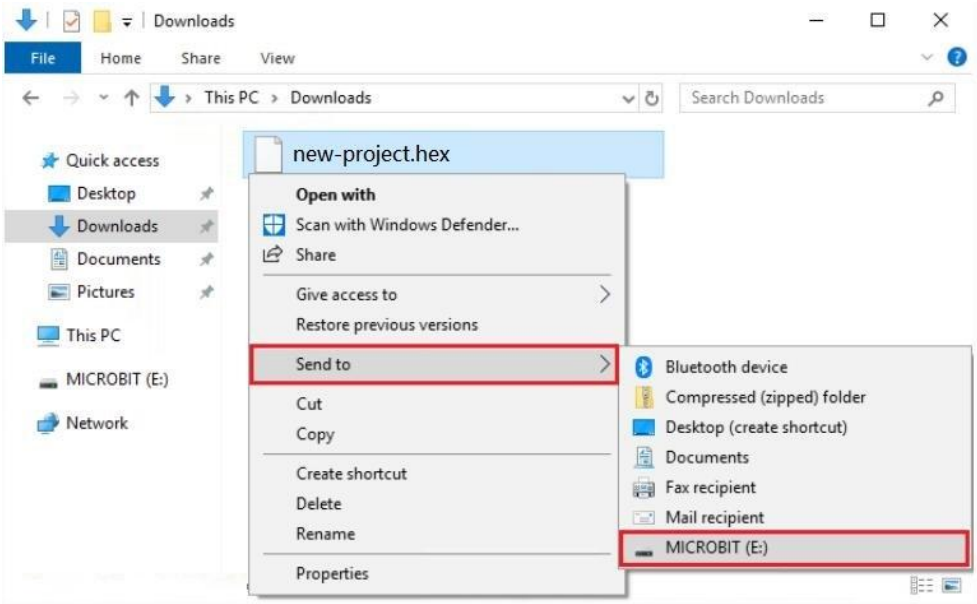



コードがアップロードされると、Micro:bit マザーボードのドットマトリックスがハート形を表示し、次に「Hello」をスクロール表示します：

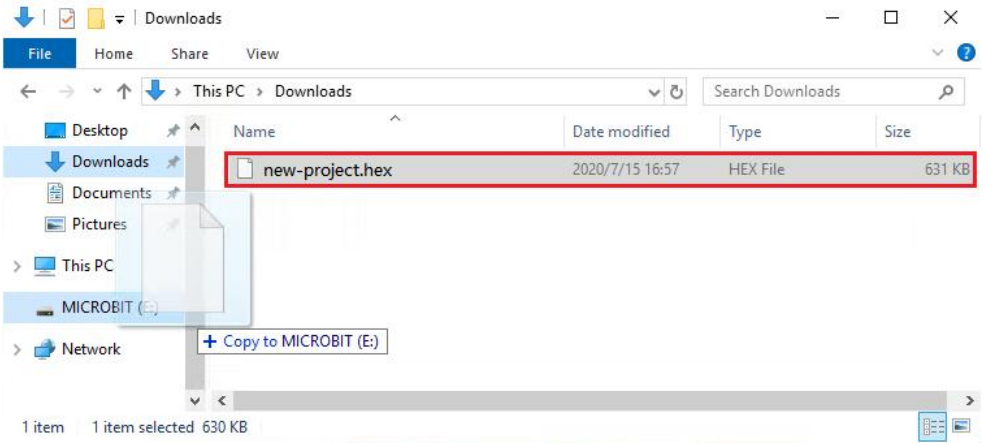


## コードをアップロードする他の方法

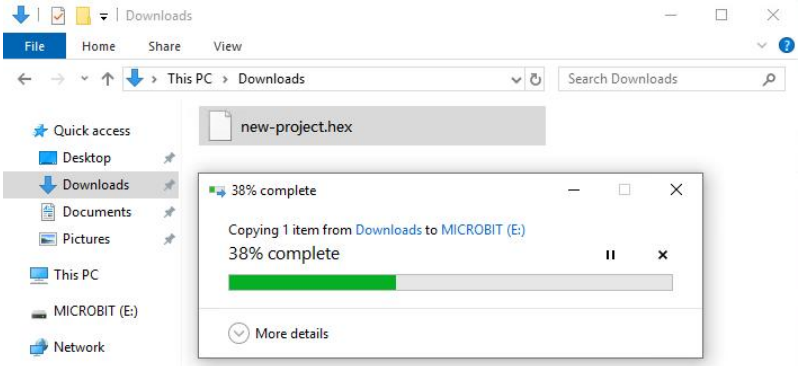
「HEX」ファイルを選択し、右クリックして「HEX」ファイルを Micro:bit マザーボードに送信します：



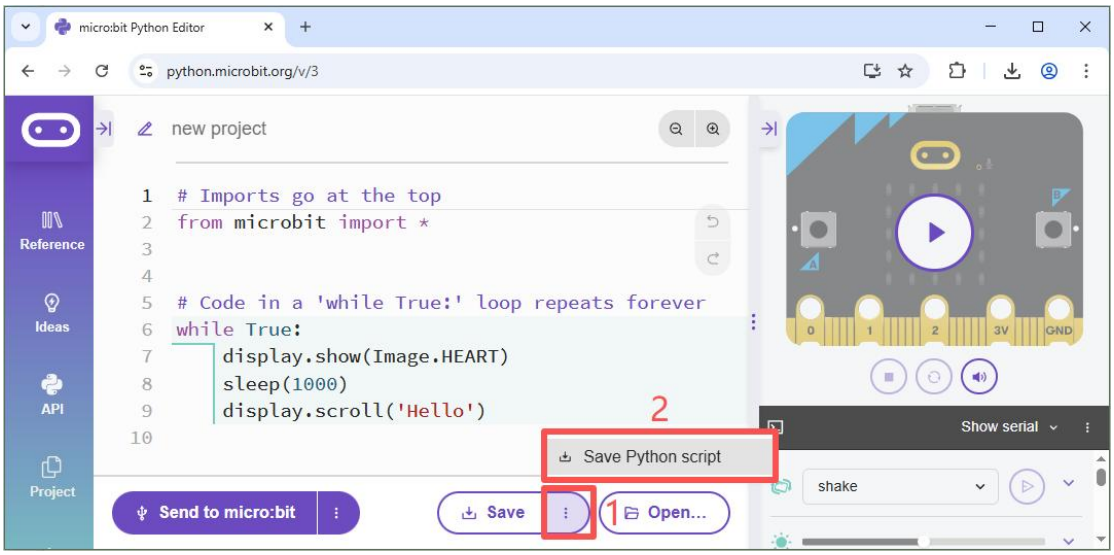

または直接 Micro:bit マザーボードにドラッグアンドドロップします：

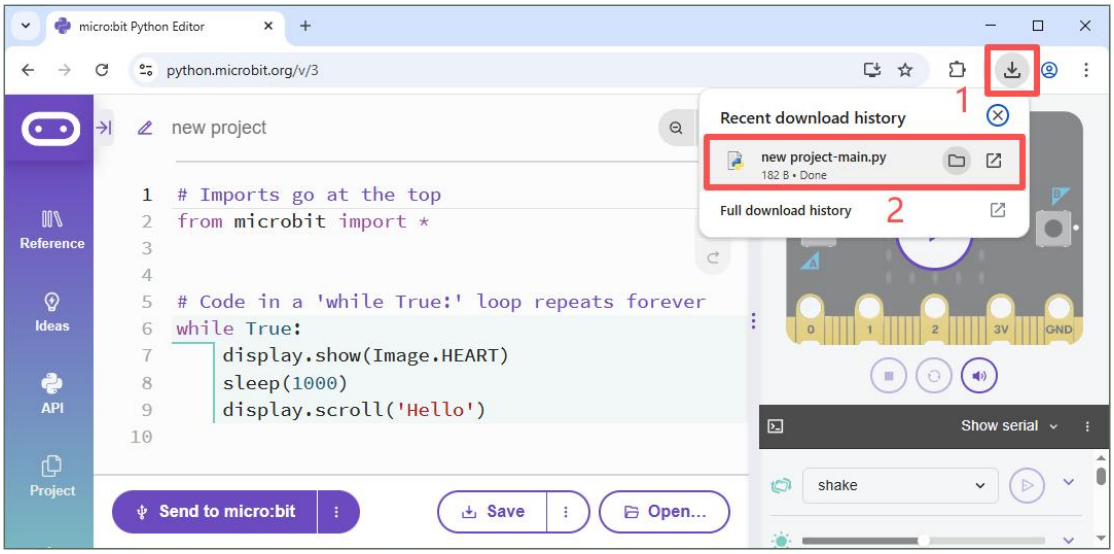



以下のインターフェースは、コードがアップロード中であることを示します：



## 5.5 プロジェクトを Python ファイルとして保存



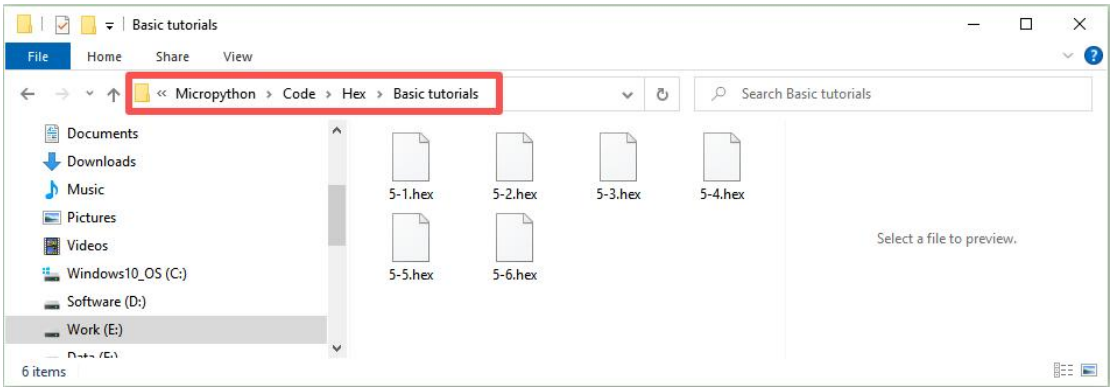

**注：Python「xx.py」ファイルはMicro:bitで直接実行できません；プロジェクトの「xx.hex」ファイルのみがMicro:bitで実行できます。「xx.py」ファイルは一般的なテキスト編集ソフトウェアで開くことができますが、「xx.hex」ファイルは開けません！**

## 5.6 Python エディタ基本構文学習

Micro:bit プラットフォームは、公式の Python エディタ使用参考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/

# 6. サンプルコード

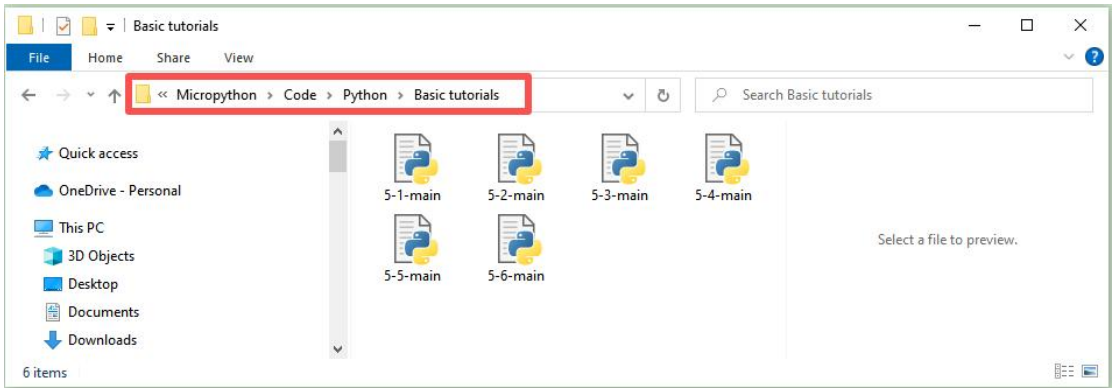
基本チュートリアルのサンプルプロジェクトはすべて、「Micropython -> Code -> Hex -> Basic tutorials」フォルダに保存されています：




特別な注意!!

以下のチュートリアルでは、提供されているローカルのサンプルプロジェクトをインポートする場合は、セクション 5.3 を参照してください。新しいプロジェクトを作成する場合は、セクション 5.1 を参照してください。

基本チュートリアルのサンプル Python ファイルは、「Micropython -> Code -> Python -> Basic tutorials」フォルダに保存されています：

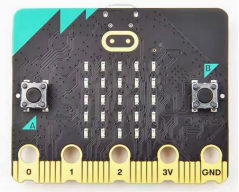


## 6.1 ファームウェアバージョン

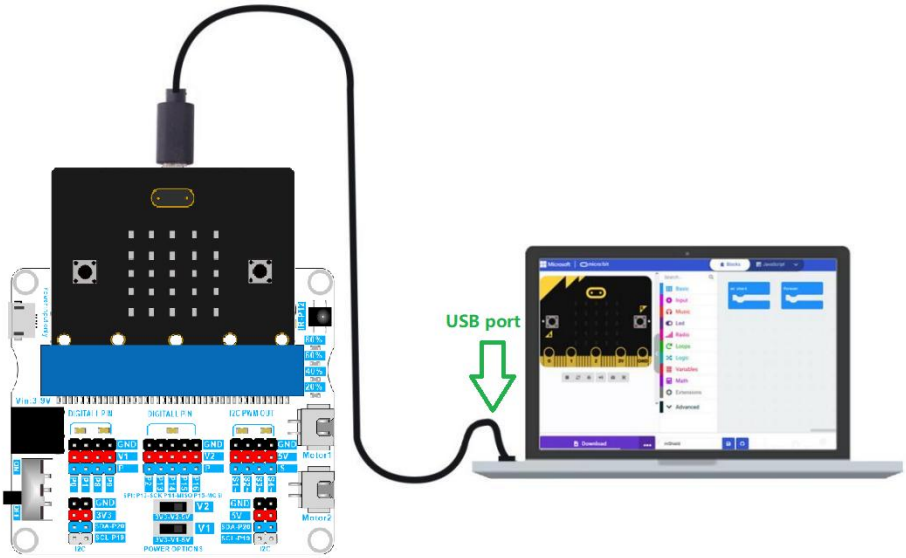
### 目的

mShield拡張ボードのファームウェアバージョンを読み取る。

### ツール

| PC | Micro:bit V2.x.x | USB cable |
|---|---|---|
|  |  |  |

### 配線



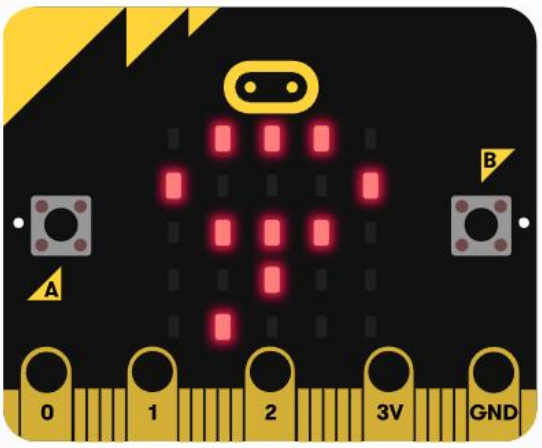

### プログラミング

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

# Car I2C address
i2cAddr = 0x29

# Firmware version register
ver = bytearray([0x00])

# Initialize the I2C communication interface
i2c.init()
```

```
# Code in a 'while True:' loop repeats forever
while True:
    # Read the firmware version
    i2c.write(i2cAddr, ver, True)
    version = i2c.read(i2cAddr, 1)

    sleep(1000)                 # Delay: 1000 milliseconds
    display.scroll(version[0])  # Scroll display of battery level
    sleep(3000)                 # Delay: 3000 milliseconds
```

## 結論

mShield 拡張ボードは、モーター、LED、および S1-S4 ポートを駆動するための 8 ビットマイクロコントローラーを統合しています。出荷前にファームウェアをプリロードしています。上記のコードを通じて、ファームウェアバージョンを読み取り、micro:bit LED マトリックスにバージョン番号を表示できます。




## 考察

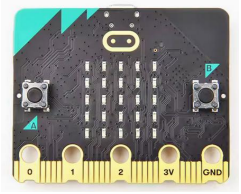
## よくある問題

## 6.2 LEDs

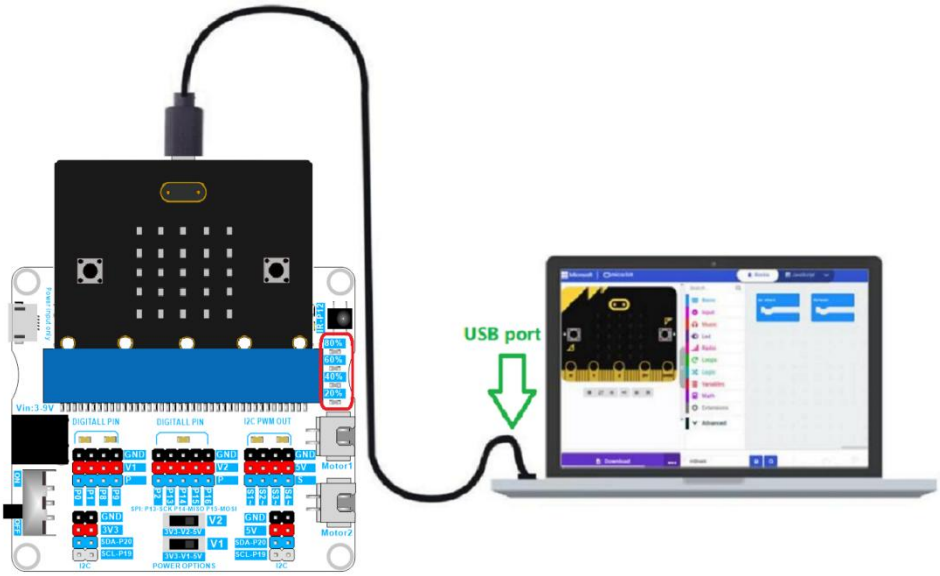
### 目的

拡張ボード上の4つのLEDを点灯させる。

### ツール

| PC | Micro:bit V2.x.x | USB cable |
| --- | --- | --- |
|  |  |  |

### 配線




### プログラミング

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

# Car I2C address
i2cAddr = 0x29
# For LEDs
led1  = 11       #20%
led2  = 12       #40%
led3  = 13       #60%
led4  = 14       #80%
i2cBuf = bytearray([0x00, 0x00])
```

```
# Control the LED function
# led: led1 -- led4
# onOff: 0 = off, 1 = on
def setLed(led, onOff):
    if led == led1:
        i2cBuf[0] = led1
    elif led == led2:
        i2cBuf[0] = led2
    elif led == led3:
        i2cBuf[0] = led3
    elif led == led4:
        i2cBuf[0] = led4
    else:
        return
    i2cBuf[1] = onOff
    i2c.write(i2cAddr, i2cBuf)

# Code in a 'while True:' loop repeats forever
while True:
    setLed(led1, 1)    # led1 on
    sleep(1000)
    setLed(led1, 0)    # led1 off
    setLed(led2, 1)    # led2 on
    sleep(1000)
    setLed(led2, 0)    # led2 off
    setLed(led3, 1)    # led3 on
    sleep(1000)
    setLed(led3, 0)    # led3 off
    setLed(led4, 1)    # led4 on
    sleep(1000)
    setLed(led4, 0)    # led4 off
```

## 結論

4 つの LED が順番に点灯します。

## 考察

上記のコードを修正して、点灯の循環をより速くしてください。

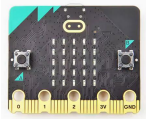
## よくある問題

## 6.3 PWM

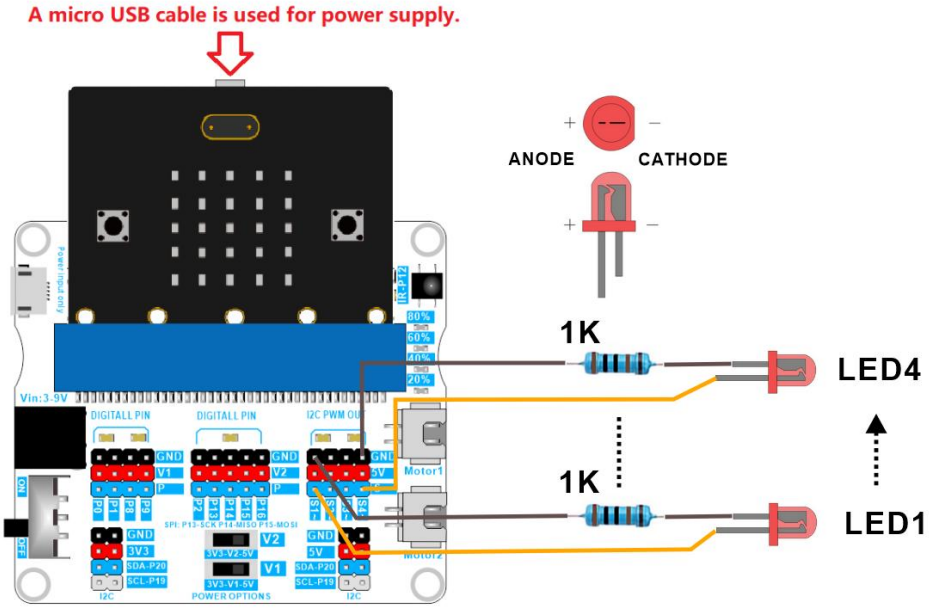
### 目的

S1 から S4 ピンを使用して LED を駆動し、LED の明るさを変更する。

### ツール

| PC | Micro:bit V2.x.x | USB cable | LED |
|---|---|---|---|
|  |  |  |  |
| 1K resistor | Female-to-female Dupont wires | | |
|  |  | | |

### 配線




### プログラミング

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

# Car I2C address
i2cAddr = 0x29

# For Pins
s1  = 16       #S1 Pin
```

```
s2  = 17      #S2 Pin
s3  = 18      #S3 Pin
s4  = 19      #S4 Pin
i2cBuf = bytearray([0x00, 0x00])

# S1--S4 pins mode
s1ToS4 = 15
pwmMode = 1
servoMode = 2

# Control S1--S4 pins output PWM signal function
# pin: s1 -- s4
# pwm: 0 -- 200
def setPinPwm(pin, pwm):
    if pin == s1:
        i2cBuf[0] = s1
    elif pin == s2:
        i2cBuf[0] = s2
    elif pin == s3:
        i2cBuf[0] = s3
    elif pin == s4:
        i2cBuf[0] = s4
    else:
        return
    i2cBuf[1] = pwm
    i2c.write(i2cAddr, i2cBuf)

# Set the S1--S4 pin mode
# mode: 1 = PWM mode, 2 = servo mode
def setPinMode(mode):
    if mode == servoMode:
        i2cBuf[1] = servoMode
    elif mode == pwmMode:
        i2cBuf[1] = pwmMode
    else:
        return
    i2cBuf[0] = s1ToS4
    i2c.write(i2cAddr, i2cBuf)

# Set the pins S1--S4 to PWM mode
setPinMode(pwmMode)

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

```
    for pwm in range(200):
        setPinPwm(s1, pwm)
        setPinPwm(s2, pwm)
        setPinPwm(s3, pwm)
        setPinPwm(s4, pwm)
        sleep(10)
```

### 結論

LED1、LED2、LED3、LED4の明るさが暗から明へゆっくり変化します。

### 考察

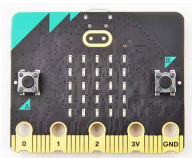
### よくある問題

## 6.4 バッテリーレベル

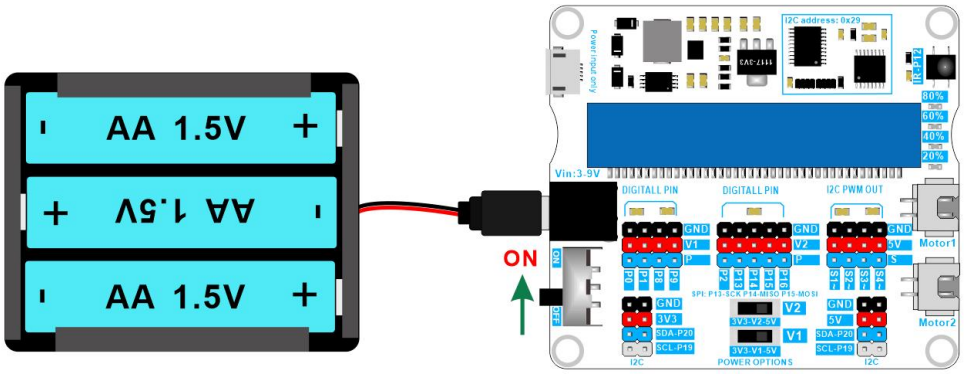
### 目的

mShieldのバッテリーレベル読み取り機能を使用して、接続されたバッテリーの電圧比を読み取る。

### ツール

| PC | Micro:bit V2.x.x | USB cable | 3xAA battery power |
|---|---|---|---|
|  |  |  |  |

### 配線




プログラミング

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

# Car I2C address
i2cAddr = 0x29

# Type of battery
aaBattery3 = bytearray([0x01])        # 3 AA batteries
aaBattery4 = bytearray([0x02])        # 4 AA batteries
aaBattery5 = bytearray([0x03])        # 5 AA batteries
aaBattery6 = bytearray([0x04])        # 6 AA batteries
lithiumBattery1 = bytearray([0x05])   # 1 lithium battery
lithiumBattery2 = bytearray([0x06])   # 2 lithium batteries

# Initialize the I2C communication interface
i2c.init()


# Code in a 'while True:' loop repeats forever
while True:
    # Read the level of 3 AA batteries
    i2c.write(i2cAddr, aaBattery3, True)
    batLevel = i2c.read(i2cAddr, 1)

    sleep(1000)                 # Delay: 1000 milliseconds
    display.scroll(batLevel[0])  # Scroll display of battery level
    sleep(3000)                 # Delay: 3000 milliseconds
```

結論

mShield に単三電池を 3 本直列接続した場合、micro:bit は電池の電圧比率（0〜100%）を読み取り、LED マトリックスにこの値を表示します。

### 思考

6.2 章に基づき、バッテリー残量を 4 段階（20%、40%、60%、80%）で表示する電量計プログラムを作成します。各 LED はそれぞれの電量レベルに対応します。

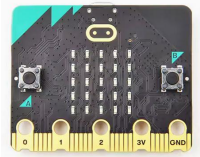
### よくある問題

## 6.5 モーター

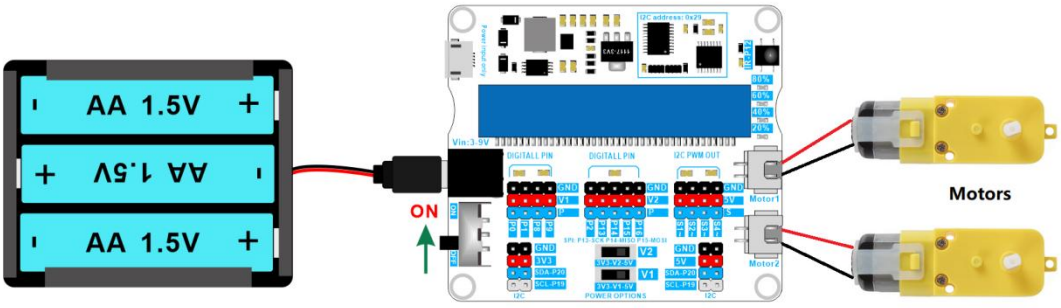
### 目标

mShield 拡張ボードの 2 つのモーターを制御する。

### ツール

| PC | Micro:bit V2.x.x | USB cable | Motor |
|---|---|---|---|
|  |  |  |  |
| 3xAA battery power | | | |
|  | | | |

### 配線




プログラミング

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

# Car I2C address
i2cAddr = 0x29

# For motor
motor1  = 1
motor2 = 2
CW   = 1    # clockwise
CCW  = 2    # counterclockwise
i2cBuf = bytearray([0x00, 0x00])

# Set the motor speed function
# motor: 1 = motor1, 2 = motor2
# direction: 1 = CW, 2 = CCW
# speed: 0--100
def setMotorSpeed(motor, direction, speed):
    # Important! The speed is between 0 and 100.
    if speed > 100:
        speed = 100
    elif speed < 0:
        speed = 0

    if motor == motor1:
        i2cBuf[0] = 0x09  # motor1 register
        if direction == CW:
            i2cBuf[1] = speed
        elif direction == CCW:
            # speed value, 101 is the default required data.
            i2cBuf[1] = speed + 101
        i2c.write(i2cAddr, i2cBuf)
```

```
    if motor == motor2:
        i2cBuf[0] = 0x0a  # motor2 register
        if direction == CW:
            i2cBuf[1] = speed
        elif direction == CCW:
            # speed value, 101 is the default required data.
            i2cBuf[1] = speed + 101
        i2c.write(i2cAddr, i2cBuf)

# Code in a 'while True:' loop repeats forever
while True:
    # CW
    setMotorSpeed(motor1, CW, 100)
    setMotorSpeed(motor2, CW, 100)
    sleep(1000)
    # stop
    setMotorSpeed(motor1, CW, 0)
    setMotorSpeed(motor2, CW, 0)
    sleep(1000)
    # CCW
    setMotorSpeed(motor1, CCW, 100)
    setMotorSpeed(motor2, CCW, 100)
    sleep(1000)
    # stop
    setMotorSpeed(motor1, CCW, 0)
    setMotorSpeed(motor2, CCW, 0)
    sleep(1000)
```

## 結論

外部モーターを Motor1 および Motor2 インターフェースに接続した場合、モーター 1 とモーター 2 は以下の動作を繰り返します：1 秒間の正回転 → 1 秒間の停止 → 1 秒間の逆回転 → 1 秒間の停止。

## 考察

## よくある問題

工場出荷時に補正値が設定されていない場合、モーターが回転しない可能性があります。補正値をリセットするだけでこの問題は解決します。

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


# Car I2C address
```

```
i2cAddr = 0x29
i2cBuf = bytearray([0x00, 0x00])

# For wheel
leftwheel  = 0
rightwheel = 1

# Calibrate the wheel speed function
# leftWheelOffset:  0--10
# rightWheelOffset: 0--10
def wheelsAdjustment(leftWheelOffset, rightWheelOffset):
    # The limit variable is between 0 and 10.
    leftWheelOffset = max(0, min(10, leftWheelOffset))
    rightWheelOffset = max(0, min(10, rightWheelOffset))

    i2cBuf[0] = 0x07  # Left wheel offset register
    i2cBuf[1] = leftWheelOffset
    i2c.write(i2cAddr, i2cBuf)
    i2cBuf[0] = 0x08  # Right wheel offset register
    i2cBuf[1] = rightWheelOffset
    i2c.write(i2cAddr, i2cBuf)

# The compensation values of the left and right motors are set to 0
wheelsAdjustment(0, 0)

while True:
    None
```

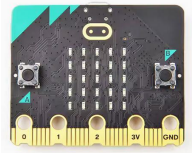
モーター速度補正は、一方のモーター速度を低下させることで 2 つのモーター速度を同期させる仕組みです。補正値の設定範囲は 0～10 です。この設定コードはプログラム実行時に 1 度だけ実行すればよく、補正後の速度パラメータは mShield に永続的に保存されます。以降のコードにこの文を含める必要はなく、モーター速度補正を再調整する場合にのみ追加します。

## 6.6 180 度サーボ

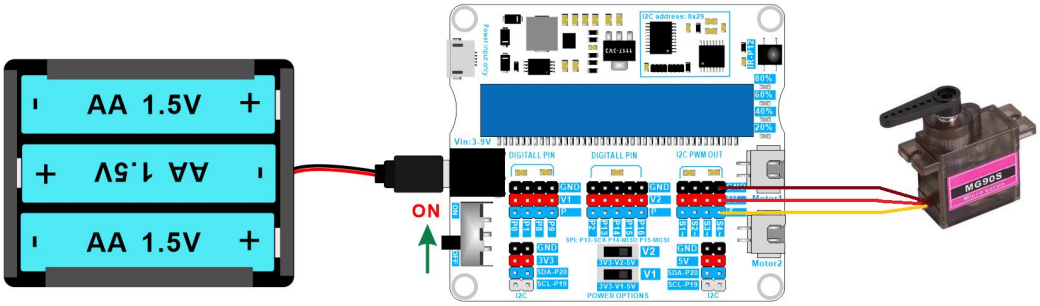
### 目的

180度サーボモーターの駆動方法を理解する。

### ツール

| PC | Micro:bit V2.x.x | USB cable | 180 度 サーボ |
|---|---|---|---|
|  |  |  |  |
| 3xAA battery power | | | |
|  | | | |

### 配線




### プログラミング

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

# Car I2C address
i2cAddr = 0x29
i2cBuf = bytearray([0x00, 0x00])

# S1--S4 pins mode
s1ToS4 = 15
pwmMode = 1
servoMode = 2
```

```
# Servo type
servo90  = 0
servo180 = 1
servo270 = 2
pinS1 = 20    # S1 pin
pinS2 = 21    # S2 pin
pinS3 = 22    # S3 pin
pinS4 = 23    # S4 pin

# Set the S1--S4 pin mode
# mode: 1 = PWM mode, 2 = servo mode
def setPinMode(mode):
    if mode == servoMode:
        i2cBuf[1] = servoMode
    elif mode == pwmMode:
        i2cBuf[1] = pwmMode
    else:
        return
    i2cBuf[0] = s1ToS4
    i2c.write(i2cAddr, i2cBuf)

# Set the Angle function of 90, 180 and 270 servo.
# index: 20 = S1 pin, 21 = S2 pin, 22 = S3 pin, 23 = S4 pin
# servoType: 0 = 90 servo, 1 = 180 servo, 2 = 270 servo
# angle: 90 servo -> 0-90, 180 servo -> 0-180, 270 servo -> 0-270
def setServo(index, servoType, angle):
    angleMap = 0
    if servoType == servo90:
        # Map 0-90 to 50-200
        angleMap = scale(angle, from_=(0, 90), to=(50, 200))
    if servoType == servo180:
        # Map 0-90 to 50-200
        angleMap = scale(angle, from_=(0, 180), to=(50, 200))
    if servoType == servo270:
        # Map 0-90 to 50-200
        angleMap = scale(angle, from_=(0, 270), to=(50, 200))

    if index == pinS1:
        i2cBuf[0] = pinS1  # S1 pin
    if index == pinS2:
        i2cBuf[0] = pinS2  # S2 pin
    if index == pinS3:
        i2cBuf[0] = pinS3  # S3 pin
    if index == pinS4:
```

```
        i2cBuf[0] = pinS4  # S4 pin

    i2cBuf[1] = angleMap
    i2c.write(i2cAddr, i2cBuf)

# Set the pins S1--S4 to servo mode
setPinMode(servoMode)

# Code in a 'while True:' loop repeats forever
while True:
    # The 180 degree servo of the S1 pin turns to the 0 degree position
    setServo(pinS1, servo180, 0)
    # The 180 degree servo of the S2 pin turns to the 0 degree position
    setServo(pinS2, servo180, 0)
    # The 180 degree servo of the S3 pin turns to the 0 degree position
    setServo(pinS3, servo180, 0)
    # The 180 degree servo of the S4 pin turns to the 0 degree position
    setServo(pinS4, servo180, 0)
    sleep(1000)

    # The 180 degree servo of the S1 pin turns to the 180 degree position
    setServo(pinS1, servo180, 180)
    # The 180 degree servo of the S2 pin turns to the 180 degree position
    setServo(pinS2, servo180, 180)
    # The 180 degree servo of the S3 pin turns to the 180 degree position
    setServo(pinS3, servo180, 180)
    # The 180 degree servo of the S4 pin turns to the 180 degree position
    setServo(pinS4, servo180, 180)
    sleep(1000)
```

## 結論

サーボシステムは 0 度から 180 度の間で循環動作します。

## 考察

90度と270度のサーボモーターを駆動する方法

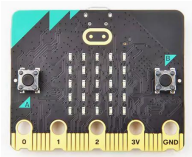
## よくある問題

## 6.7 360 度サーボ

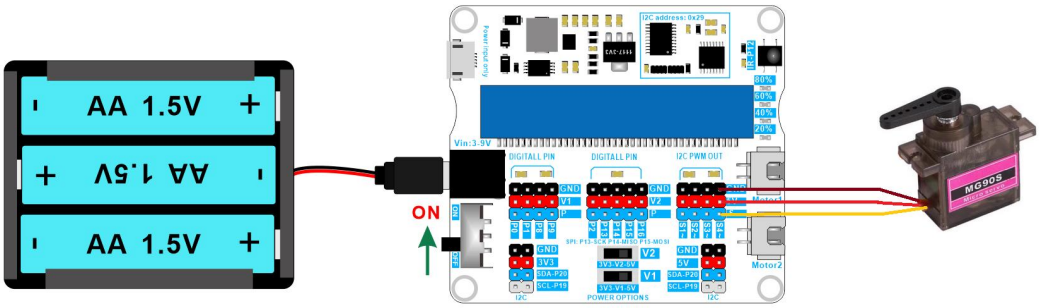
### 目的

360度サーボモーターの駆動方法を理解する

### ツール

| PC | Micro:bit V2.x.x | USB cable | 360 度 サーボ |
|---|---|---|---|
|  |  |  |  |
| 3xAA battery power | | | |
|  | | | |

### 配線




### プログラミング

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

# Car I2C address
i2cAddr = 0x29
i2cBuf = bytearray([0x00, 0x00])

# S1--S4 pins mode
s1ToS4 = 15
pwmMode = 1
servoMode = 2

# Servo type
```

```
servo90  = 0
servo180 = 1
servo270 = 2
pinS1 = 20    # S1 pin
pinS2 = 21    # S2 pin
pinS3 = 22    # S3 pin
pinS4 = 23    # S4 pin

# Set the S1--S4 pin mode
# mode: 1 = PWM mode, 2 = servo mode
def setPinMode(mode):
    if mode == servoMode:
        i2cBuf[1] = servoMode
    elif mode == pwmMode:
        i2cBuf[1] = pwmMode
    else:
        return
    i2cBuf[0] = s1ToS4
    i2c.write(i2cAddr, i2cBuf)

# Set the Angle function of 90, 180 and 270 servo.
# index: 20 = S1 pin, 21 = S2 pin, 22 = S3 pin, 23 = S4 pin
# servoType: 0 = 90 servo, 1 = 180 servo, 2 = 270 servo
# angle: 90 servo -> 0-90, 180 servo -> 0-180, 270 servo -> 0-270
def setServo(index, servoType, angle):
    angleMap = 0
    if servoType == servo90:
        # Map 0-90 to 50-200
        angleMap = scale(angle, from_=(0, 90), to=(50, 200))
    if servoType == servo180:
        # Map 0-90 to 50-200
        angleMap = scale(angle, from_=(0, 180), to=(50, 200))
    if servoType == servo270:
        # Map 0-90 to 50-200
        angleMap = scale(angle, from_=(0, 270), to=(50, 200))

    if index == pinS1:
        i2cBuf[0] = pinS1  # S1 pin
    if index == pinS2:
        i2cBuf[0] = pinS2  # S2 pin
    if index == pinS3:
        i2cBuf[0] = pinS3  # S3 pin
    if index == pinS4:
        i2cBuf[0] = pinS4  # S4 pin
```

```
    i2cBuf[1] = angleMap
    i2c.write(i2cAddr, i2cBuf)

# Set the speed of 360 servo
# index: 20 = S1 pin, 21 = S2 pin, 22 = S3 pin, 23 = S4 pin
# speed: -100 to +100
def setServo360(index, speed):
    # Map -100 - 100 to 0 - 180
    angle = scale(speed, from_=(-100, 100), to=(0, 180))
    setServo(index, servo180, angle)

# Set the pins S1--S4 to servo mode
setPinMode(servoMode)

# Code in a 'while True:' loop repeats forever
while True:
    # The 360-degree servo of the S1--S4 pins is forward
    # at the maximum speed
    setServo360(pinS1, 100)
    setServo360(pinS2, 100)
    setServo360(pinS3, 100)
    setServo360(pinS4, 100)
    sleep(1000)

    # The 360-degree servo of the S1--S4 pins is backward
    # at the maximum speed
    setServo360(pinS1, -100)
    setServo360(pinS2, -100)
    setServo360(pinS3, -100)
    setServo360(pinS4, -100)
    sleep(1000)
```

## 結論

S1、S2、S3、S4 ピンに 360 度サーボモーターを接続時、サーボモーターは 1 秒間隔で最大速度の正逆回転を繰り返します。

## 考察

360 度サーボモーターの緩やかな均一加速起動方法

## よくある問題

# 7. QA

## 7.1 micro:bit にコードをアップロードできない

☛ データ通信機能対応の USB ケーブルをご使用ですか?

☛ USB ケーブルの状態は良好ですか?

## 7.2 Micro:bit のドライブレターが誤って表示されています

☛表示内容：MAINTENANCE（正常場合は MICROBIT）

この現象は、Micro:bit の電源投入時に誤ってリセットボタンを押し続けたため、ファームウェア更新モードに移行したことが原因です。ファームウェアを再更新することで正常状態に復旧できます。以下のリンクから操作手順を参照してください：

https://microbit.org/get-started/user-guide/firmware/

## 7.3 コードを再アップロード後もモーターが回転を継続

☛ 原因：5.6 節で定義したモーター停止関数が再呼び出されていないため

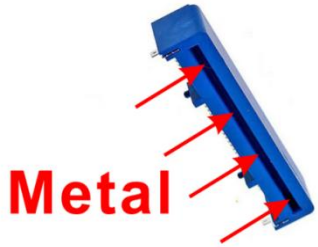
```
setWheelSpeed(wheel, direction, 0)
```

## 7.4 モーターが作動しない場合の確認事項

☛micro:bit が Pybit 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 エッジコネクタの清潔さを確保してください

☛Pybit スロット内部の清掃状態を確認してください




☛6.5節のFAQを参照の上、ホイール補正値を0にリセットしてください。

## 7.5 その他の故障

☛組み立てが正しく行われているか確認してください

☛请 バッテリー残量が十分か確認してください

☛使用中の電池が規定規格に適合しているか確認してください

# 8. お問い合わせ

上記の解決策で問題が解決しない場合は、サポートチームまでご連絡ください。

迅速な対応をさせていただくために、以下の情報をご準備ください：

ご注文番号

製品型番（例：mShield 拡張ボード M1E0002）および使用ソフトウェア（例：MakeCode または MicroPython）

問題や疑問点の詳細な説明

実施済みの対応手順

関連するエラーメッセージのスクリーンショット、写真、またはコード断片

### その他のお問い合わせ

当社はお客様のご意見を重視し、常に改善に努めております。以下のお問い合わせ窓口までお気軽にご連絡ください：

**チュートリアルの誤記とフィードバック：**ドキュメント改善のためのご指摘
**製品アイデアとご提案：**優れたアイデアをぜひお聞かせください
**パートナーシップとコラボレーション：**ご協働にご興味がございましたら、ぜひご相談ください
**割引とプロモーション：**教育機関向けまたは大口購入時の価格について
**その他：**上記に該当しない非技術的なお問い合わせ全般

### サポートチャネル


support@siyeenove.com


https://siyeenove.com