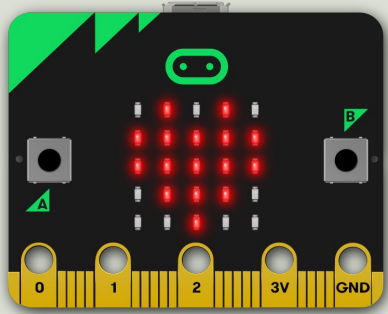


<https://www.halvorsen.blog>



micro:bit

Using the built-in Sensors and Interfaces

Hans-Petter Halvorsen

Contents

- [Introduction to micro:bit](#)
- [micro:bit and Python and MicroPython](#)
- [Mu Python Editor](#) - Python Editor with built-in support for micro:bit
- [micro:bit Interfaces with Python Examples](#)
 - [LED Matrix \(5x5\)](#)
 - [Buttons \(A and B\)](#)
 - [Temperature Sensor](#)
 - [Light Sensor](#)
 - [Accelerometer](#)
 - [Compass](#)
 - [I/O Pins](#)

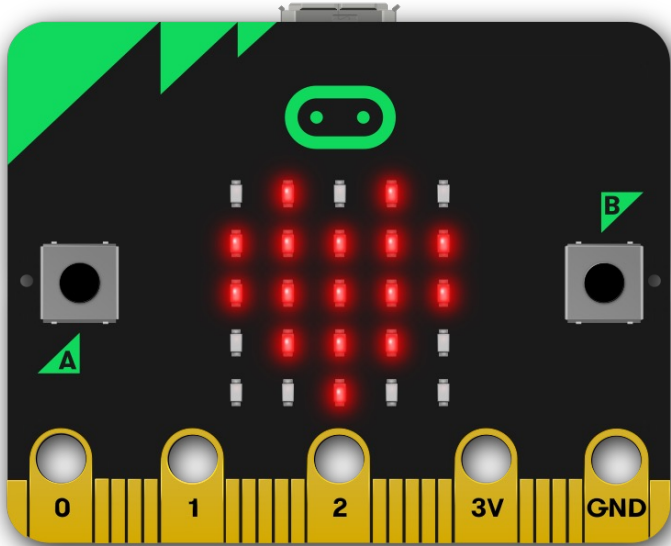


Introduction to micro:bit

Hans-Petter Halvorsen

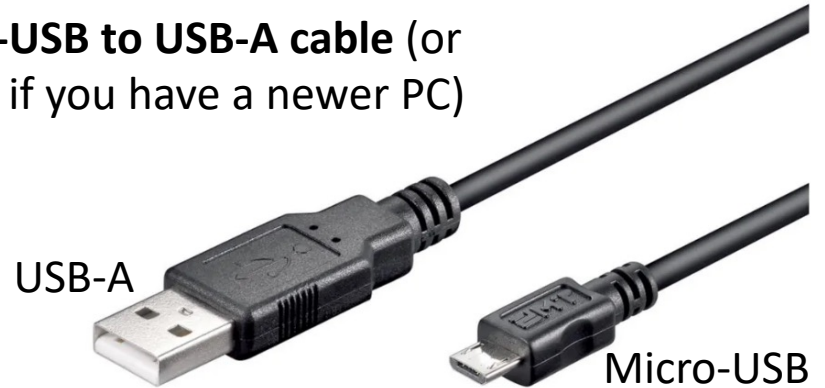
[Table of Contents](#)

What do you need?



micro:bit

Micro-USB to USB-A cable (or USB-C if you have a newer PC)



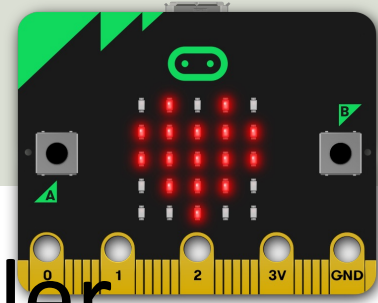
USB-A

Micro-USB

A PC

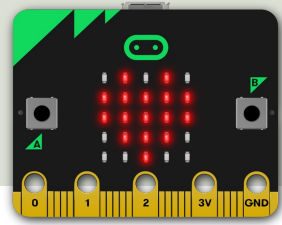


micro:bit



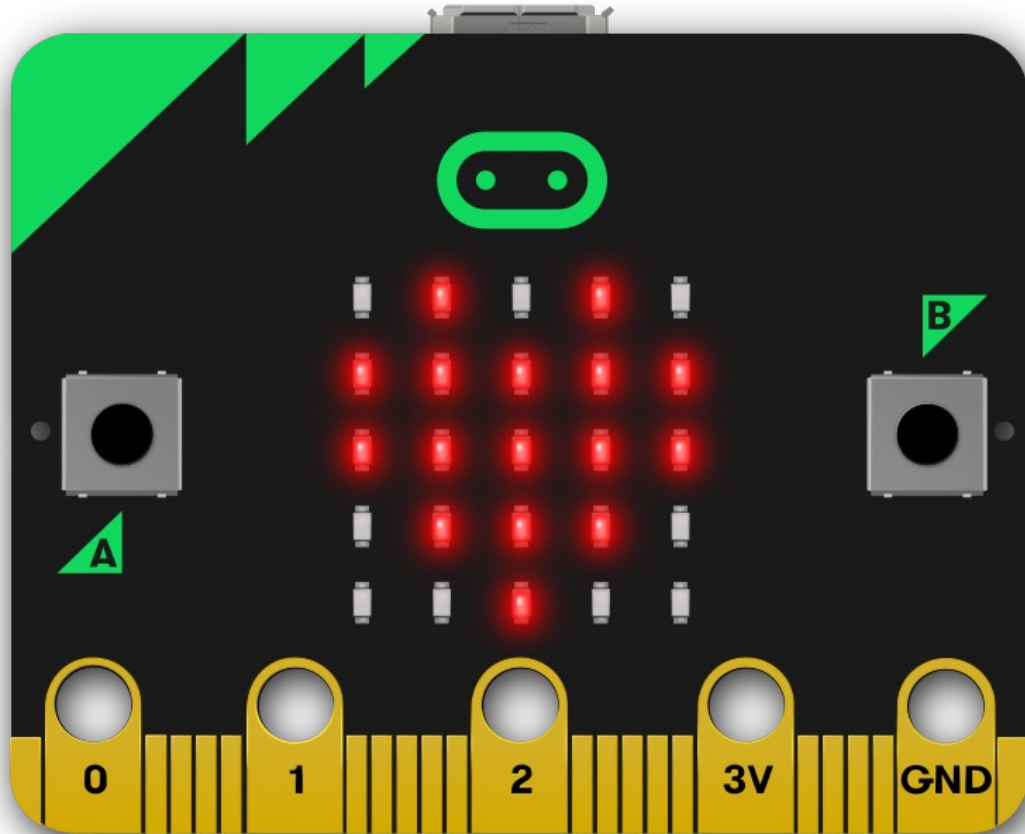
- micro:bit is a small microcontroller
- micro:bit is smaller than a credit card
- Price is about 150-400NOK (\$15-30)
- It can be used by kids and students to learn programming and technology
- micro:bit has Bluetooth but not WiFi

micro:bit and Programming

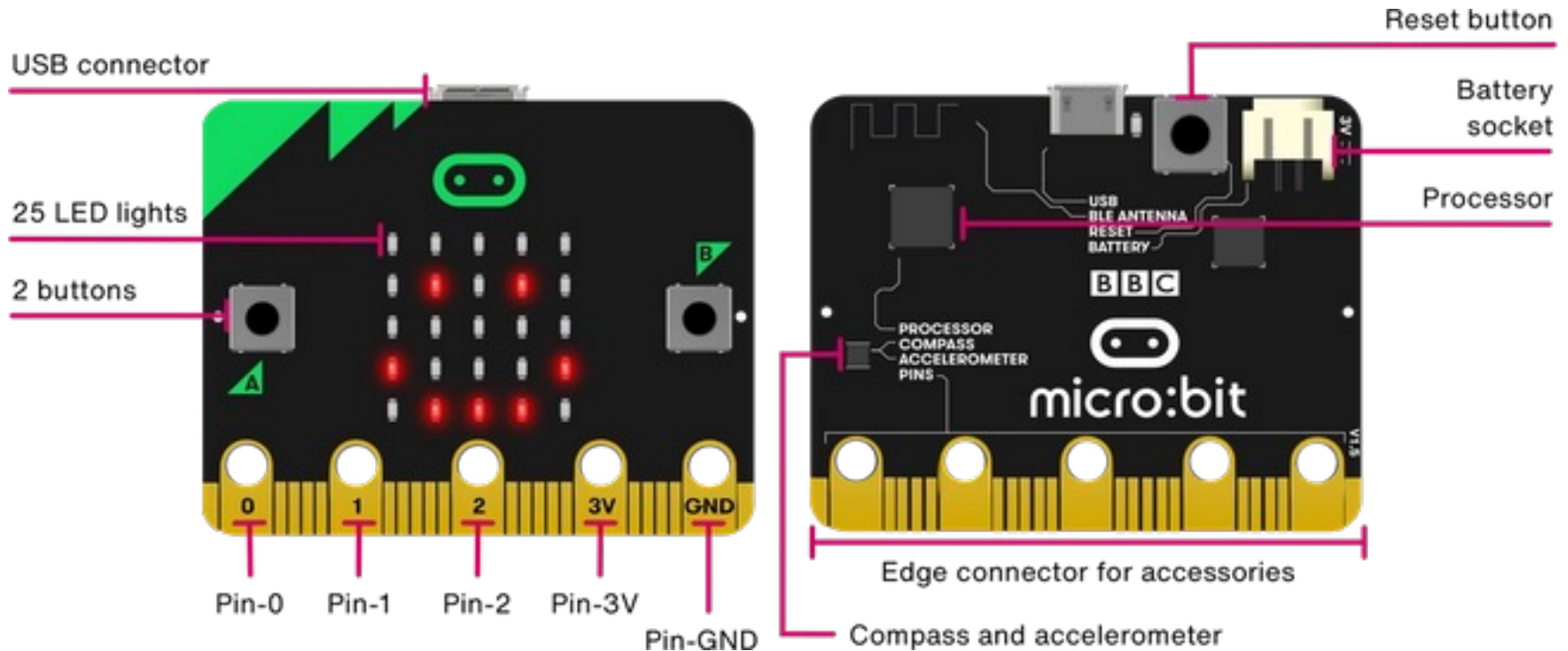


- micro:bit can run a special version of Python called **MicroPython**
- MicroPython is a down-scaled version of Python
- You can use different code editors and Programming Languages
 - Scratch, Microsoft MakeCode, Python, Swift Playground, etc.
- This Tutorial will use Python/MicroPython

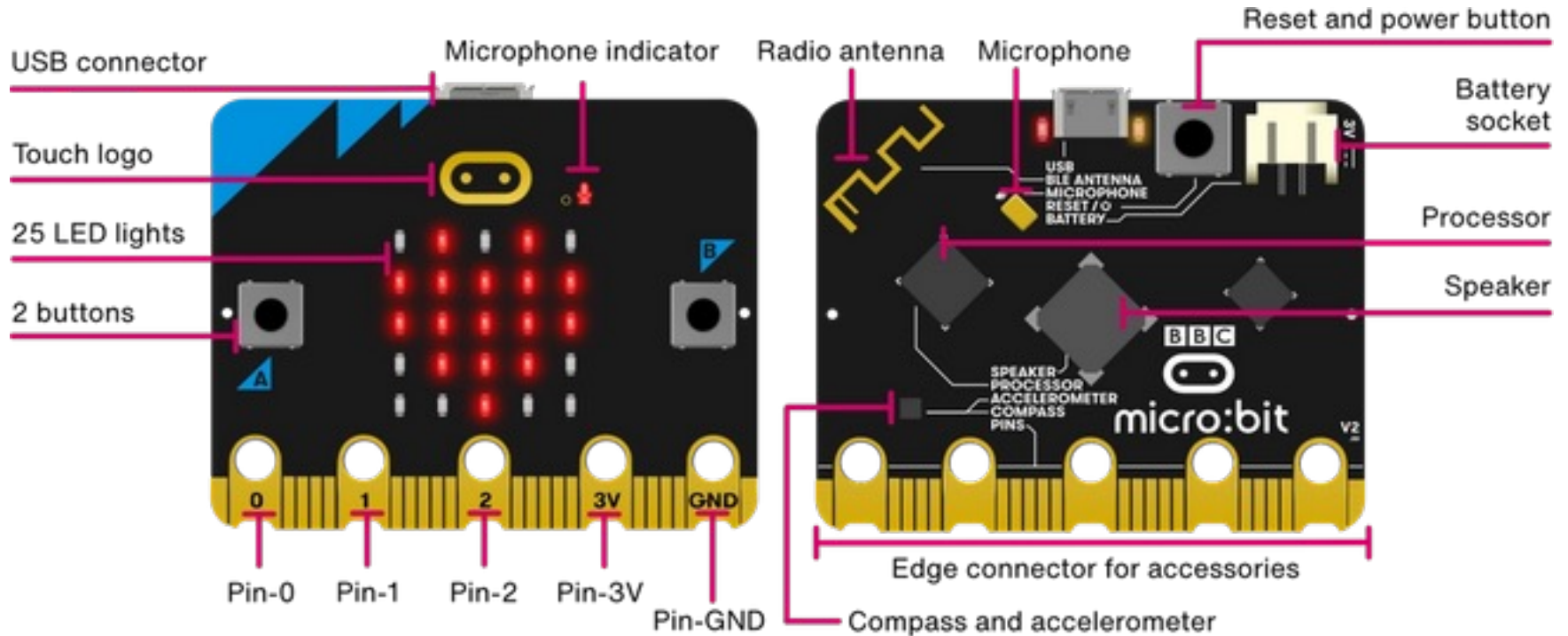
micro:bit



Original micro:bit



New micro:bit (micro:bit v2)



micro:bit Features

- USB Communication and Powered by micro-USB or JST Battery Connection
- Sensors: Motion, Temperature, Light, Magnetism, Microphone and Touch
- Push Buttons
- Lots of Analog/Digital Input/Output Pins
- Speaker
- Wireless Radio Communication
- Bluetooth Communication
- SPI, I2C and UART
- Pulse Width Modulation (PWM)

Micro USB

Front

Touch sensitive logo

LED matrix 5x5

Microphone

- LED indicator
- Hole for microphone input

User buttons

Analogue/Digital I/O

- Muxable to SPI, UART, I2C
- Notched pads for crocodile clips
- Holes for banana plugs

External supply

- Regulated 3.3V in or battery out

Edge Connector

Power indicator

USB activity indicator

Back

Battery connector

- JST connection for 3V

Reset/power button

NXP KL27Z

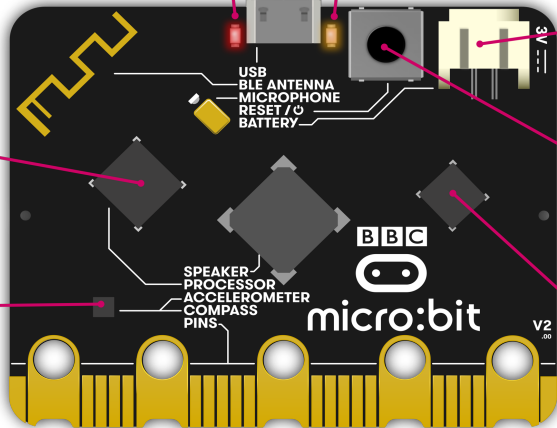
- USB interface chip

CPU

Nordic nRF52833

Motion sensor

ST LSM303AGR



<https://www.halvorsen.blog>



micro:bit and Python

Hans-Petter Halvorsen

[Table of Contents](#)

Python

- Python is a fairly old Programming Language (1991) compared to many other Programming Languages like C# (2000), Swift (2014), Java (1995), PHP (1995).
- Python has during the last 10 years become more and more popular.
- Today, Python has become one of the most popular Programming Languages.

micro:bit and Python

- The combination of the micro:bit Hardware and the Python Programming Language is very powerful
- micro:bit runs a special version of MicroPython
- MicroPython is a down-scaled version of Python
- You can use different Python Editors; e.g., Mu Python Editor or the Online Python Editor, etc.

MicroPython

- MicroPython is a small and efficient implementation of the Python 3 programming language
- MicroPython includes a small subset of the Python standard library
- MicroPython is optimized to run on microcontrollers and in constrained environments
- <https://microbit-micropython.readthedocs.io/>

micro:bit Python Documentation

- micro:bit Python User Guide

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

- micro:bit MicroPython documentation

<https://microbit-micropython.readthedocs.io>

micro:bit Python Editors

Here are some Editors:

- Online Editor (used in your Browser)

<https://python.microbit.org>

- Mu Python Editor

<https://codewith.mu>

This Tutorial will mainly use the Mu Python Editor



Online Python Editor

Online Python Editor

The screenshot displays the Python Editor for micro:bit interface. The browser address bar shows `python.microbit.org`. The main editor area contains the following Python code:

```
1 # Imports go at the top
2 from microbit import *
3
4
5 # Code in a 'while True:' loop repeats forever
6 while True:
7     display.show(Image.HEART)
8     sleep(1000)
9     display.scroll('Hello')
10
```

The left sidebar features a search bar and several categories:

- Variables**: Keep track of data that...
- Display**: The micro:bit's LED displa...
- Buttons**: Use button inputs in your...
- Loops**: Count and repeat sets of...
- Logic**: Making decisions in code
- Accelerometer**: Detect gestures and...
- Comments**: Explain your Python code
- Maths**: Basic maths in Python

The right-hand panel includes a visual micro:bit board, a "Show serial" dropdown, a "shake" event trigger, various sliders, and a "Radio message" section. At the bottom, there are buttons for "Send to micro:bit", "Save", and "Open...".

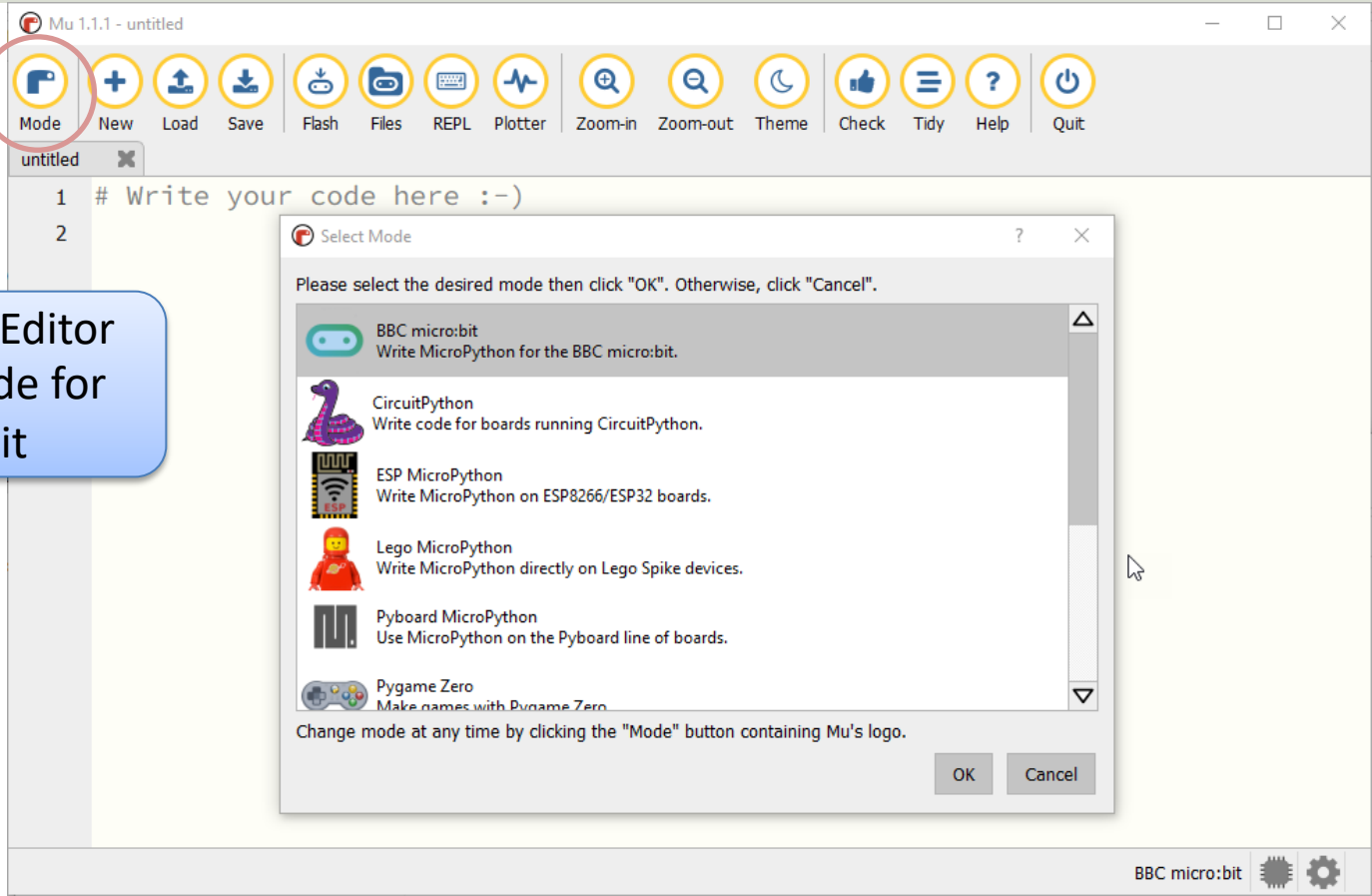


Mu Python Editor

Mu Python Editor

- Mu is a Python code editor for beginners
- It is tailor-made for micro:bit programming
- Mu has a “micro:bit mode” that makes it easy to work with micro:bit, download code to the micro:bit hardware, etc.
- Mu and micro:bit Tutorials:
<https://codewith.mu/en/tutorials/1.0/microbit>

Mu Python Editor



The screenshot shows the Mu Python Editor interface. The title bar reads "Mu 1.1.1 - untitled". The toolbar contains several icons, with the "Mode" icon (a blue folder with a white 'P') circled in red. Below the toolbar, the code editor shows two lines of text: "1 # Write your code here :-)" and "2". A "Select Mode" dialog box is open in the foreground, displaying a list of modes:

- BBC micro:bit
Write MicroPython for the BBC micro:bit.
- CircuitPython
Write code for boards running CircuitPython.
- ESP MicroPython
Write MicroPython on ESP8266/ESP32 boards.
- Lego MicroPython
Write MicroPython directly on Lego Spike devices.
- Pyboard MicroPython
Use MicroPython on the Pyboard line of boards.
- Pygame Zero
Make names with Pygame Zero.

At the bottom of the dialog box, there are "OK" and "Cancel" buttons. The status bar at the bottom right of the editor shows "BBC micro:bit" and a gear icon.

The Mu Python Editor has built-in Mode for the micro:bit

Hello World

Mu 1.1.1 - HelloWorld.py

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

HelloWorld.py

```
1 from microbit import *
2
3 while True:
4     display.show('Hello World')
5     sleep(2000)
6
```

Click on the “Flash” button to download the code to the micro:bit

You need to start all code by importing the microbit Python library

```
from microbit import *

while True:
    display.show('Hello World')
    sleep(2000)
```

BBC micro:bit

REPL

The screenshot shows the Mu Python IDE interface. At the top, a toolbar contains various icons for file operations and development tools. The 'REPL' icon, which depicts a computer keyboard, is circled in red. Below the toolbar, a code editor window titled 'HelloWorld.py' contains the following Python code:

```
1 from microbit import *
2
3 while True:
4     print('Hello World')
5     sleep(2000)
6
```

At the bottom of the IDE, a 'BBC micro:bit REPL' window is open, showing the output of the code:

```
>>>
>>> Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

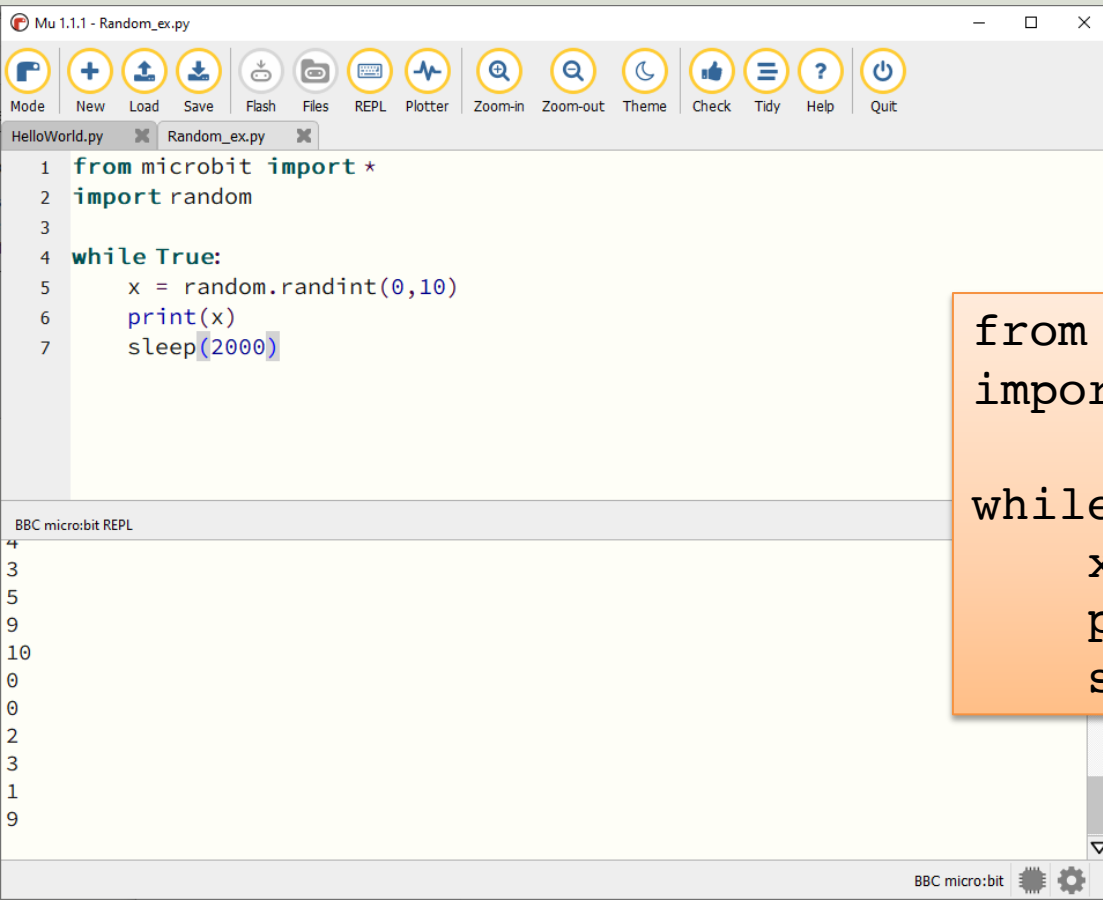
Click on the “REPL” button to interact with the MicroPython installed on the micro:bit

Here is the standard Python **print()** command used

```
from microbit import *

while True:
    print('Hello World')
    sleep(2000)
```


REPL – Random Numbers



The screenshot shows the Mu Python IDE interface. The title bar reads "Mu 1.1.1 - Random_ex.py". The menu bar includes icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The code editor displays the following Python code:

```
1 from microbit import *
2 import random
3
4 while True:
5     x = random.randint(0,10)
6     print(x)
7     sleep(2000)
```

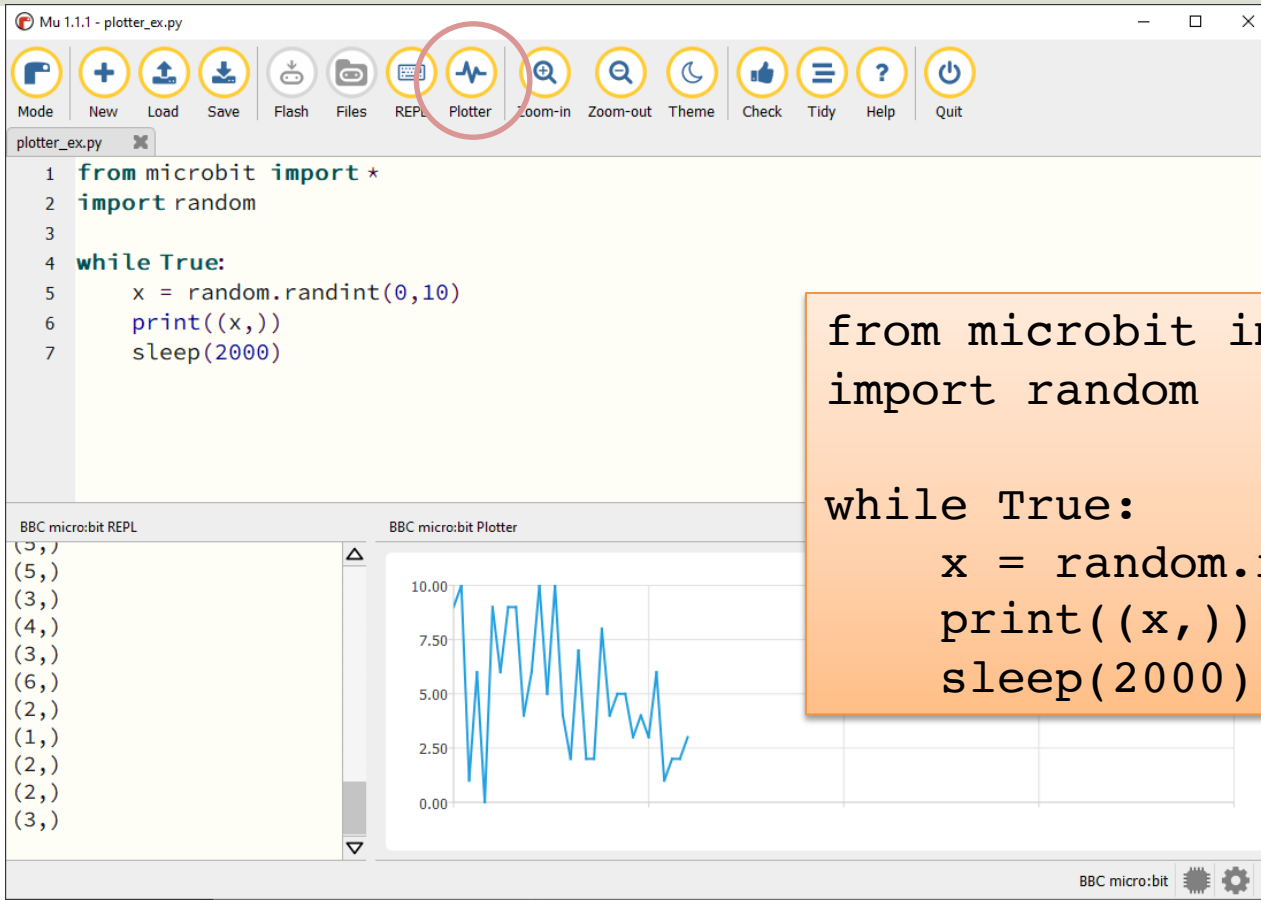
Below the code editor is the "BBC micro:bit REPL" window, which shows a vertical list of numbers: 4, 3, 5, 9, 10, 0, 0, 2, 3, 1, 9.

At the bottom of the window, the text "BBC micro:bit" is visible next to a gear icon.

```
from microbit import *
import random

while True:
    x = random.randint(0,10)
    print(x)
    sleep(2000)
```

Plotter



The screenshot shows the Mu Python IDE interface. At the top, a toolbar contains icons for various functions: Mode, New, Load, Save, Flash, Files, REPL, Plotter (circled in red), Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. Below the toolbar, a code editor displays a Python script. The script is as follows:

```
1 from microbit import *
2 import random
3
4 while True:
5     x = random.randint(0,10)
6     print((x,))
7     sleep(2000)
```

Below the code editor, there are two panels. The left panel, titled "BBC micro:bit REPL", shows the output of the script: a series of coordinate pairs (x, y) where x is the value of the random number and y is the current line number. The right panel, titled "BBC micro:bit Plotter", displays a line graph of the data. The x-axis represents the line number (1 to 7), and the y-axis represents the value of x (0.00 to 10.00). The plot shows a jagged line representing the random values generated by the script.

BBC micro:bit

```
from microbit import *
import random

while True:
    x = random.randint(0,10)
    print((x,))
    sleep(2000)
```

Files

The screenshot displays the Mu 1.1.1 IDE interface. At the top, the title bar reads "Mu 1.1.1 - HelloWorld.py". Below it is a toolbar with icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The "Files" icon is circled in red. The main editor area shows a Python script with the following code:

```
1 from microbit import *
2
3 while True:
4     display.show('Hello World')
5     sleep(2000)
6
```

Below the editor is a "Filesystem on micro:bit" window. It has two panes: "Files on your device:" containing "main.py" and "Files on your computer:". A blue callout box points to the "Files on your computer:" pane with the text "C:\Users\\mu_code".

It is possible to drag and drop Files between the micro:bit and your computer

C:\Users\\mu_code

BBC micro:bit



micro:bit Interfaces

with Python Examples

Hans-Petter Halvorsen

[Table of Contents](#)

micro:bit Interfaces

- LED Matrix (5x5)
- Buttons (A and B)
- Temperature Sensor
- Light Sensor
- Accelerometer
- Compass
- Touch (only available for new micro:bit)
- Microphone (only available for new micro:bit)
- I/O Pins: Analog/Digital Input/Output Pins



LED Matrix (5x5)

LED Matrix (5x5)

- An LED, or light-emitting diode is an output device that gives off light.
- The Micro:bit has a display of 25 (5x5) LEDs for you to program.
- You can use the LED matrix to show images or show text or numbers

LED Matrix - Text

```
from microbit import *  
  
display.show("WELCOME")
```

This will show one letter at the time on the LED matrix

```
from microbit import *  
while True:  
display.show("WELCOME")  
sleep(1000)
```

It will do it "Forever"

```
from microbit import *  
  
display.scroll("WELCOME")
```

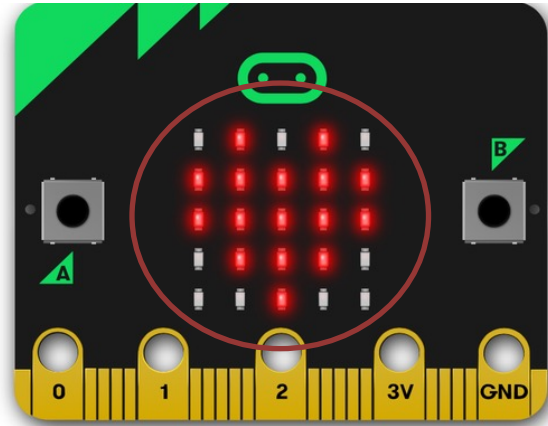
The word "WELCOME" will scroll over the LED matrix

```
from microbit import *  
while True:  
display.scroll("WELCOME")  
sleep(1000)
```


LED Matrix - Images

The micro:bit has a set of other built-in images that you can use

```
from microbit import *  
  
display.show(Image.HEART)
```

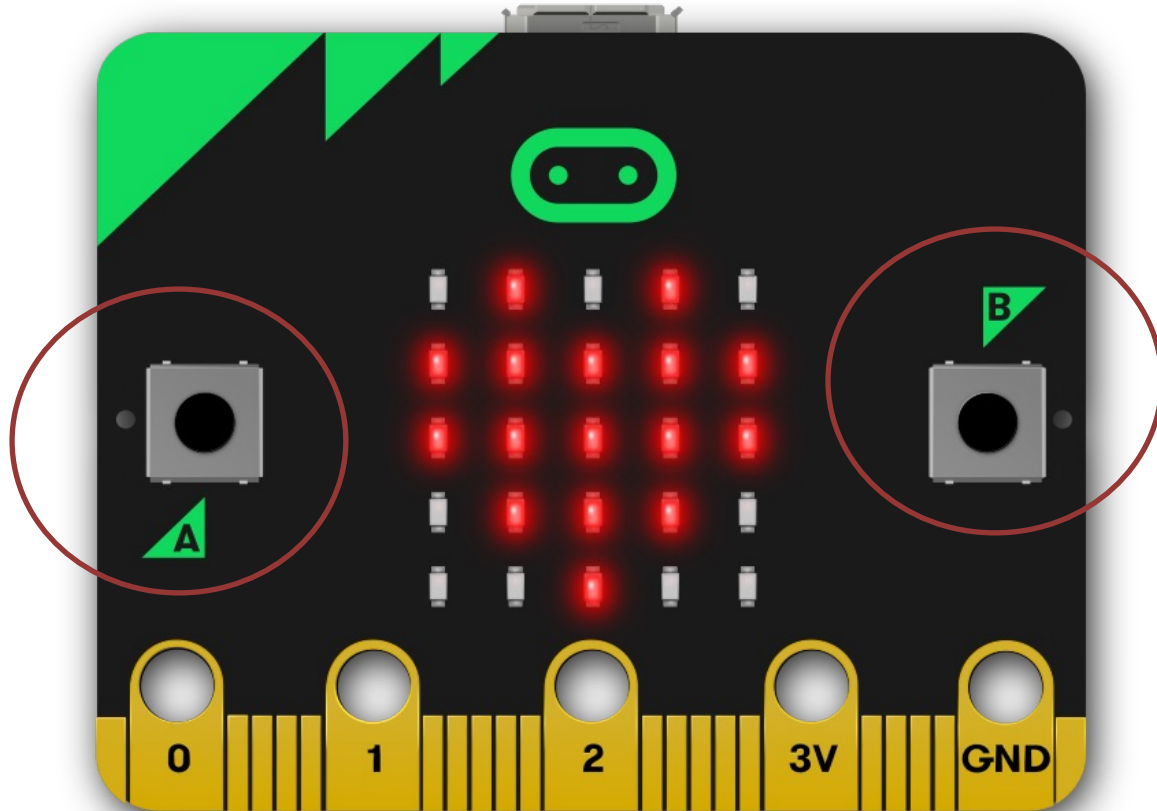


There are almost 100 built-in images that you can use. Just enter “Image.” and the Intellisense will list all available Images that you can use.



Buttons (A and B)

Buttons (A and B)



Buttons (A and B)

```
from microbit import *

while True:
    if button_a.was_pressed():
        display.scroll("A")
    elif button_b.was_pressed():
        display.scroll("B")
    else:
        display.scroll("?")

    sleep(1000)
```



Temperature Sensor

Temperature Sensor

- Micro:bit has a built-in Temperature Sensor (that is located on the CPU)
- This sensor can give an approximation of the air temperature.
- Just use the built-in `temperature()` function in order to get the temperature value from the sensor

Temperature Sensor

In order to read the temperature, you just use the built-in `temperature()` function:

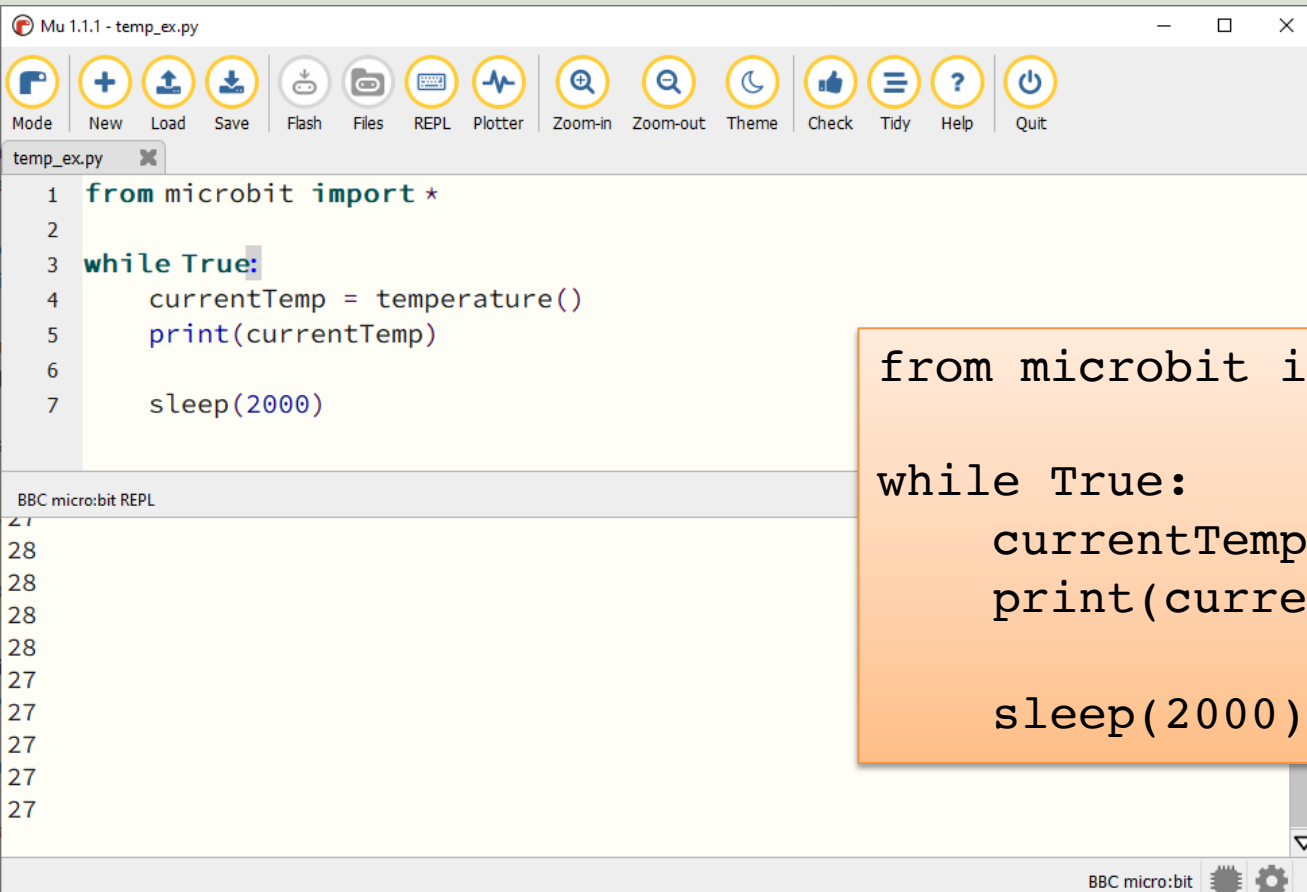
```
from microbit import *  
  
currentTemp = temperature()
```

This examples displays the temperature on the LED matrix:

```
from microbit import *  
  
while True:  
    if button_a.was_pressed():  
        display.scroll(temperature())
```

<https://microbit.org/get-started/user-guide/features-in-depth/#temperature-sensor>

Temperature Sensor



The screenshot shows the Mu Python IDE interface. The title bar reads "Mu 1.1.1 - temp_ex.py". The toolbar contains icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The main editor window displays the following Python code:

```
1 from microbit import *
2
3 while True:
4     currentTemp = temperature()
5     print(currentTemp)
6
7     sleep(2000)
```

Below the editor is the "BBC micro:bit REPL" window, which shows a series of "27" characters, indicating the program is running and printing the temperature value.

```
from microbit import *

while True:
    currentTemp = temperature()
    print(currentTemp)

    sleep(2000)
```


Temperature Sensor

The screenshot shows the Mu Python IDE interface. The main window displays a Python script named 'temperature_read.py' with the following code:

```
1 from microbit import *
2
3 while True:
4     currentTemp = temperature()
5     display.scroll(currentTemp)
6     print((currentTemp,))
7     sleep(1000)
```

The IDE's toolbar includes icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Help, and Quit. Below the code editor, the 'BBC micro:bit REPL' window shows a series of output values: (24,) repeated 13 times, followed by (25,). To the right of the REPL, a plotter window displays a blue line graph that remains constant at a value of 24 for most of the duration, then jumps to 25 for the final few data points.

```
from microbit import *

while True:
    currentTemp = temperature()
    display.scroll(currentTemp)
    print((currentTemp,))
    sleep(1000)
```

Display Min/Max Temperature

```
from microbit import *

currentTemp = temperature()
maxTemp = currentTemp
minTemp = currentTemp

while True:
    currentTemp = temperature()

    if currentTemp < minTemp:
        minTemp = currentTemp
    if currentTemp > maxTemp:
        maxTemp = currentTemp

    if button_a.was_pressed():
        display.scroll(minTemp)
    elif button_b.was_pressed():
        display.scroll(maxTemp)
    else:
        display.scroll(currentTemp)

    print((currentTemp, minTemp, maxTemp))
    sleep(2000)
```

If you do nothing, the LED matrix shows the Current Temperature.

If you click A Button, the Minimum Temperature for the period (since you started the program/turned on the Micro:bit) is shown on the LED matrix

If you click B Button, the Maximum Temperature for the period (since you started the program/turned on the Micro:bit) is shown on the LED matrix



Light Sensor

Light Sensor

The LED matrix display on the front of your micro:bit can also **detect** light

```
from microbit import *

lightlimit = 100

while True:
    if display.read_light_level() > lightlimit:
        display.show(Image.HAPPY) #Happy because sunny
    else:
        display.show(Image.SAD) #Sad because cloudy
        sleep(2000)
```

In this Example, hold your micro:bit in front of a light source (e.g., a flashlight) and turn it on and off. The Image on the LED matrix should go from Sad to Happy or opposite.

Light Sensor

```
from microbit import *

lightlimit = 100

def sunlight():
    display.show(Image(
        "00000:"
        "00900:"
        "09990:"
        "00900:"
        "00000"))
    sleep(500)
    display.show(Image(
        "00000:"
        "09990:"
        "09990:"
        "09990:"
        "00000"))
    sleep(500)
    display.show(Image(
        "90909:"
        "09990:"
        "99999:"
        "09990:"
        "90909"))

while True:
    if display.read_light_level() > lightlimit:
        sunlight()
    else:
        display.show(Image.SAD) #Sad because cloudy weather
        sleep(2000)
```

Shows a flashing sunny image



Accelerometer

Accelerometer

Basic Example using the Accelerometer functionality:

```
from microbit import *  
import random  
  
while True:  
    if accelerometer.was_gesture('shake'):  
        display.show(random.randint(1, 6))
```

After shaking the micro:bit, a number between 1 and 6 is shown

Dices

```
from microbit import *
import random

while True:
    if accelerometer.was_gesture('shake'):
        number = random.randint(1, 6)
        if number == 1:
            display.show(Image(
                "00000:"
                "00000:"
                "00900:"
                "00000:"
                "00000"))
        elif number == 2:
            display.show(Image(
                "00000:"
                "00000:"
                "90009:"
                "00000:"
                "00000"))
        elif number == 3:
            display.show(Image(
                "00009:"
                "00000:"
                "00900:"
                "00000:"
                "90000"))
        elif number == 4:
            display.show(Image(
                "90009:"
                "00000:"
                "00000:"
                "00000:"
                "90009"))
        elif number == 5:
            display.show(Image(
                "90009:"
                "00000:"
                "00900:"
                "00000:"
                "90009"))
        else:
            display.show(Image(
                "90009:"
                "00000:"
                "90009:"
                "00000:"
                "90009"))
```

After shaking the micro:bit, a dice is shown with 1, 2, 3, 4, 5, or 6 eyes

Dices Improved

```
from microbit import *

def dice(number):

    if number == 1:
        diceimage = Image("00000:"
                           "00000:"
                           "00900:"
                           "00000:"
                           "00000")

    elif number == 2:
        diceimage = Image("00000:"
                           "00000:"
                           "90009:"
                           "00000:"
                           "00000")

    elif number == 3:
        diceimage = Image("00009:"
                           "00000:"
                           "00900:"
                           "00000:"
                           "90000")

    elif number == 4:
        diceimage = Image("90009:"
                           "00000:"
                           "00000:"
                           "00000:"
                           "90009")

    elif number == 5:
        diceimage = Image("90009:"
                           "00000:"
                           "00900:"
                           "00000:"
                           "90009")

    else:
        diceimage = Image("90009:"
                           "00000:"
                           "90009:"
                           "00000:"
                           "90009")

    return diceimage
```

dice.py

```
from microbit import *
import random
from dice import *

while True:
    if accelerometer.was_gesture('shake'):
        number = random.randint(1, 6)
        display.show(dice(number))
```

Dices

The screenshot shows the Mu Python IDE interface. At the top, there is a toolbar with icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. Below the toolbar, the code editor displays the following Python code:

```
1 from microbit import *
2 import random
3 from dice import *
4
5 while True:
6     if accelerometer.was_gesture('shake'):
7         number = random.randint(1, 6)
8         display.show(dice(number))
```

At the bottom of the IDE, there is a file manager section with two panes: "Files on your device:" and "Files on your computer:". The "Files on your device:" pane lists "dice.py" and "main.py". The "Files on your computer:" pane lists "dice.py". A red box highlights the "Files on your device:" pane. In the bottom right corner, there is a blue box containing the path "C:\Users\\mu_code".

Note! The Python file “dice.py” needs to be copied to the micro:bit

C:\Users\\mu_code



Compass

Hans-Petter Halvorsen

[Table of Contents](#)

Compass

The micro:bit has a built-in compass sensor called a magnetometer. You can use it to measure the Earth's magnetic field and use it as a compass.

When you first use the micro:bit compass you must calibrate it – a little game appears on the screen where you must tilt the micro:bit to light up every LED, then you're ready to go.

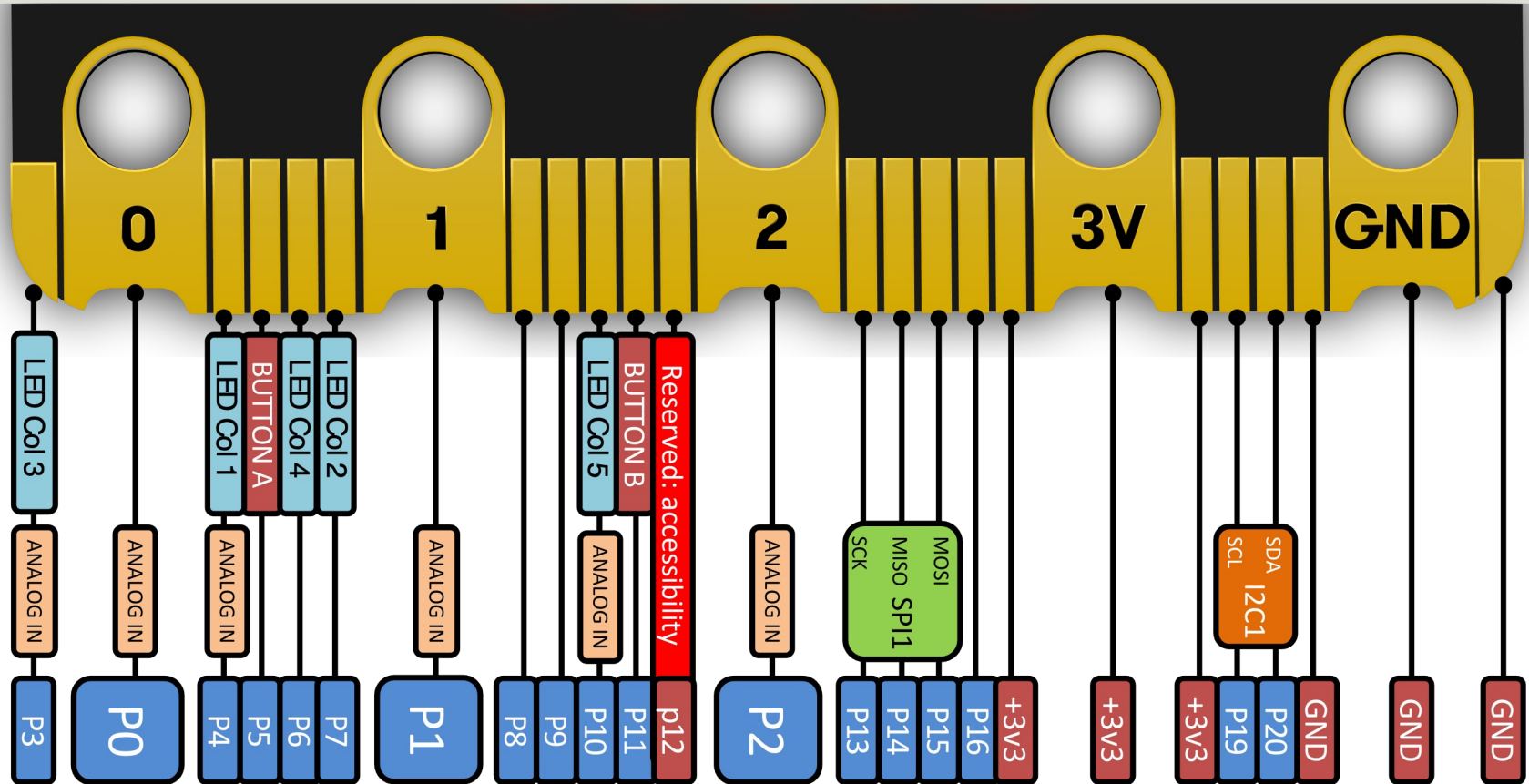
```
from microbit import *  
  
while True:  
    if button_a.was_pressed():  
        display.scroll(str(compass.heading()))
```

<https://microbit.org/projects/make-it-code-it/compass-bearing/?editor=python>



I/O Pins

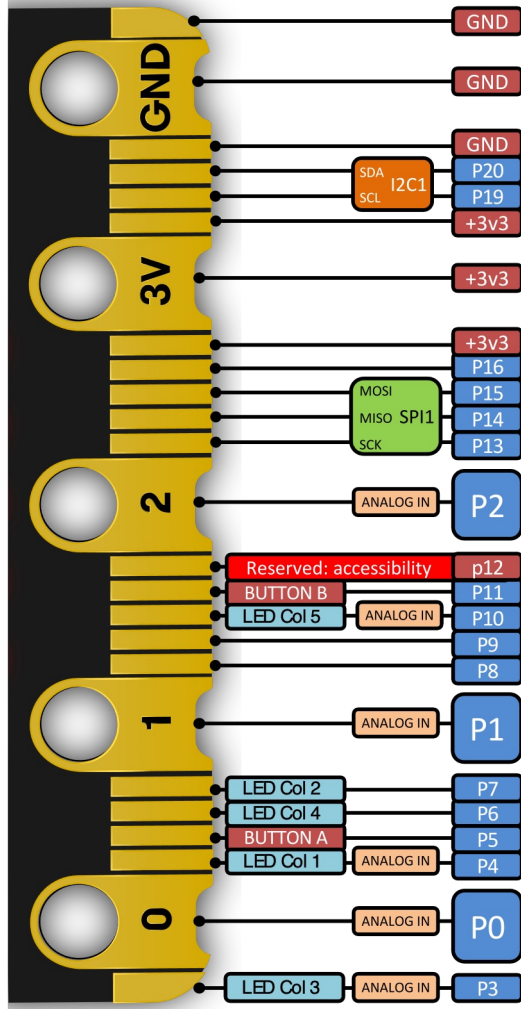
I/O Pin Overview v.2



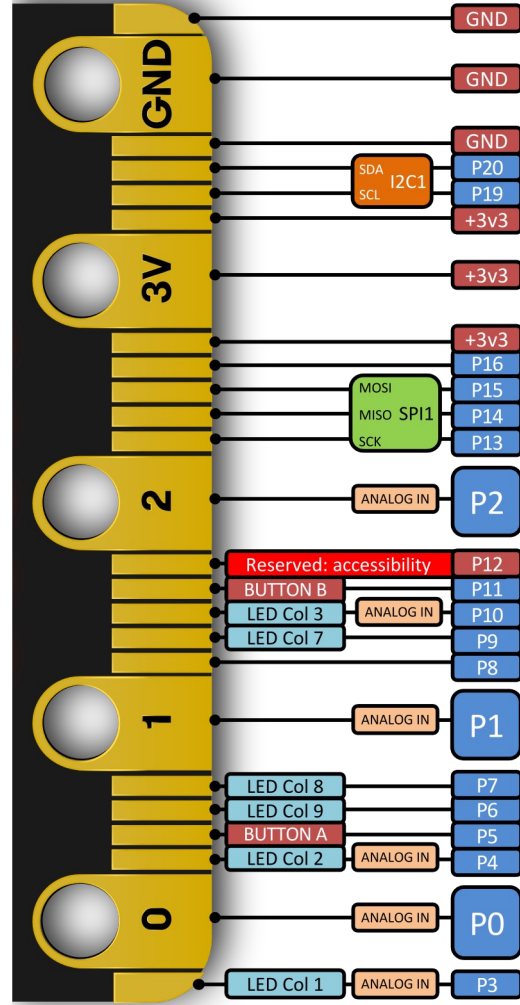
I/O Pin Overview

<https://microbit.pinout.xyz/>

New micro:bit (micro:bit v2)

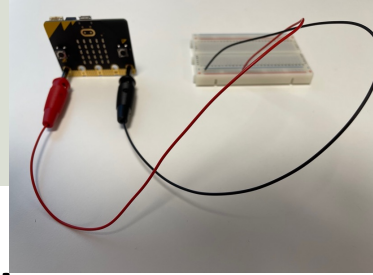


Original micro:bit

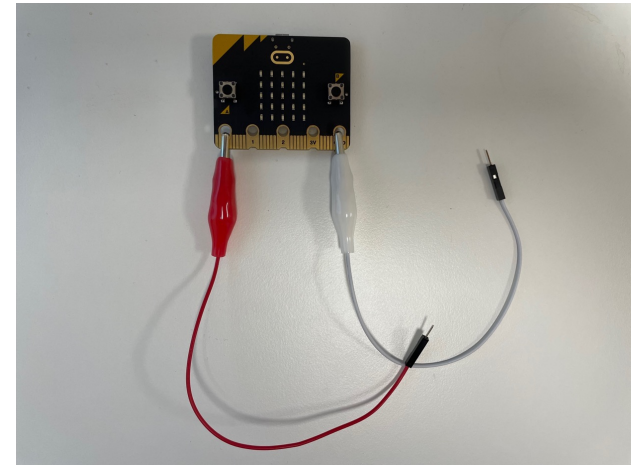


<https://tech.microbit.org/hardware/edgeconnector/>

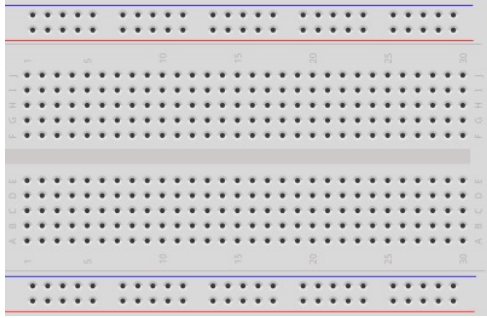
I/O Pins



- We use the I/O pins to connect external components like LEDs, different types of Sensors, etc.
- You can use 4mm Banana plugs or alligator/crocodile clips
- Typically you also want to use a Breadboard

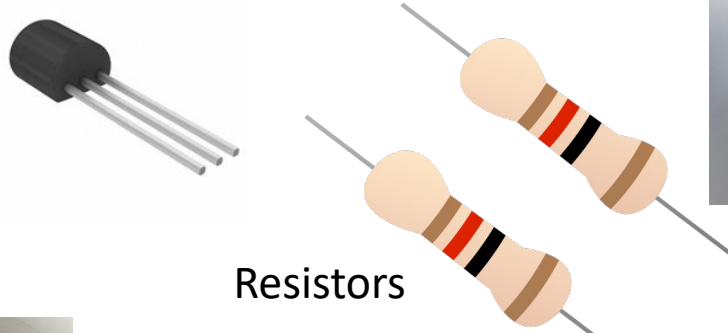


Component Examples

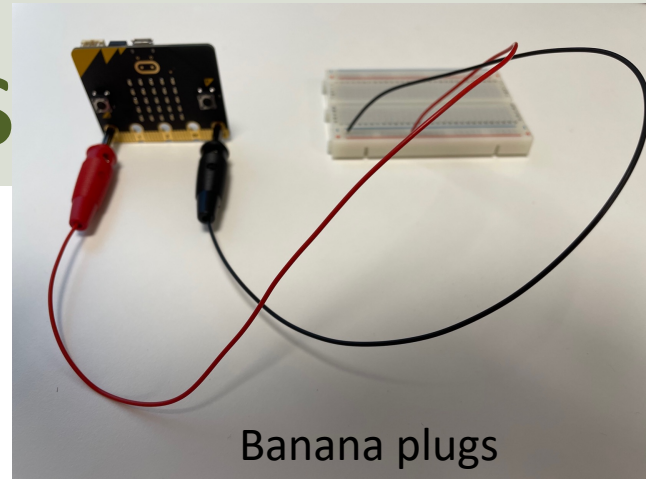


Breadboard

Temperature Sensor



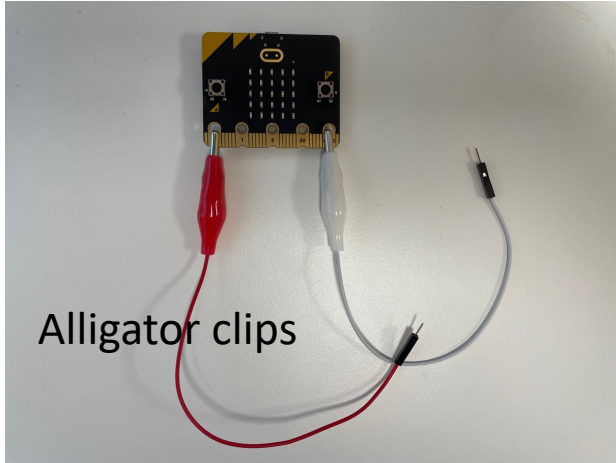
Resistors



Banana plugs

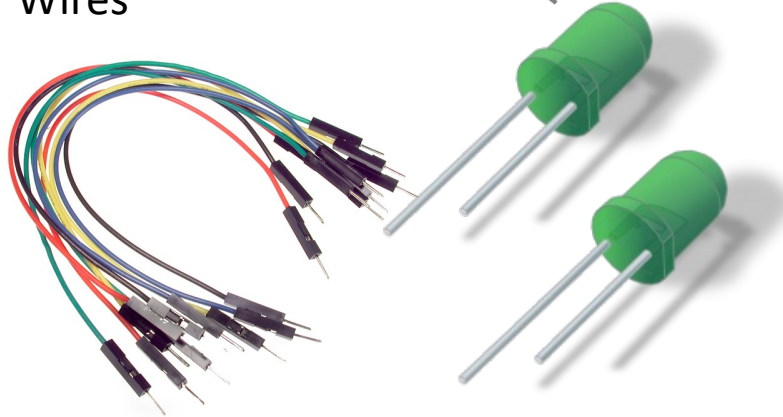
LEDs

Multimeter



Alligator clips

Wires



Types of I/O Pins

- Analog/Digital Input/Output Pins
- Pulse Width Modulation (PWM)
- SPI
- I2C
- UART (used for serial communication)

We will not use the I/O pins in this Tutorial, but I will give an overview and give examples in other micro:bit/Python Tutorials that goes in more depth regarding these I/O pins with lots of practical examples

micro:bit Resources and References

- micro:bit Python User Guide
<https://microbit.org/get-started/user-guide/python/>
- micro:bit MicroPython documentation
<https://microbit-micropython.readthedocs.io>
- Learn micro:bit (Adafruit):
<https://learn.adafruit.com/bbc-micro-bit-lesson-number-0>
- Online Python Editor: <https://python.microbit.org>
- Mu Python Editor: <https://codewith.mu>

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

