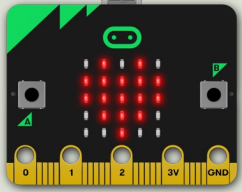


<https://www.halvorsen.blog>



micro:bit and TMP36 Temperature Sensor

Hans-Petter Halvorsen

Contents

- [Introduction to micro:bit and Python/MicroPython](#)
- [Using the built-in Temperature Sensor](#)
- [micro:bit I/O Pins](#)
 - Analog and Digital Pins used for communication with external components, like LEDs, Temperature Sensors, etc.
- [Using an external TMP36 Temperature Sensor](#)
 - Python Examples

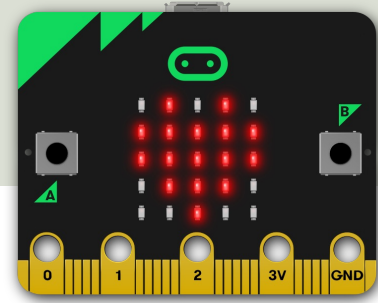


Introduction to micro:bit

Hans-Petter Halvorsen

[Table of Contents](#)

micro:bit

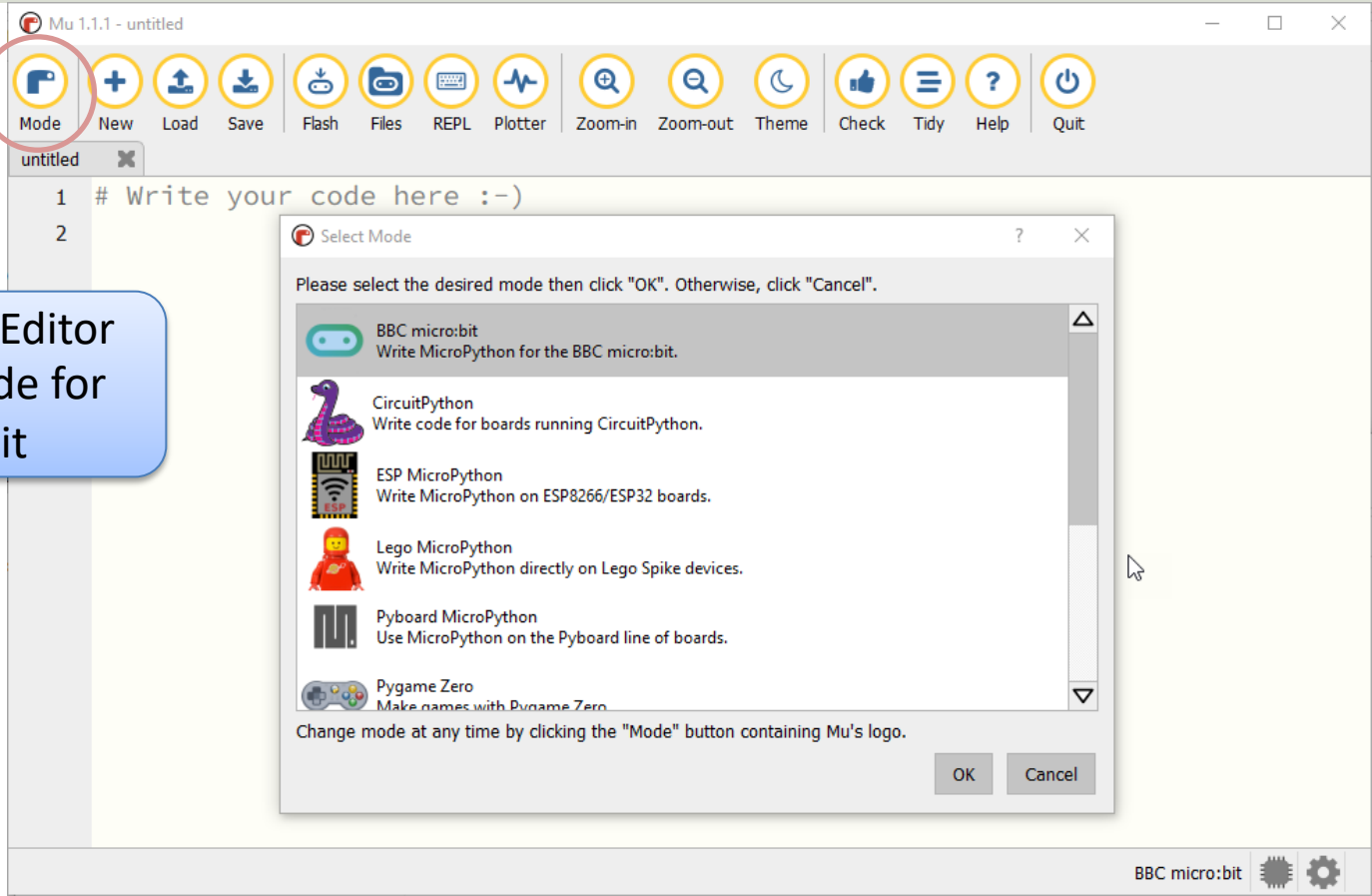


- micro:bit is a small microcontroller
- micro:bit is smaller than a credit card
- Price is about 150-400NOK (\$15-30)
- It can be used by kids and students to learn programming and technology
- micro:bit can run a special version of Python called MicroPython
- MicroPython is a down-scaled version of Python

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, CircuitPython, ESP MicroPython, Lego MicroPython, Pyboard MicroPython, and Pygame Zero. The dialog box has "OK" and "Cancel" buttons at the bottom.

Mu 1.1.1 - untitled

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

```
1 # Write your code here :-)
2
```

Select Mode

Please select the desired mode then click "OK". Otherwise, click "Cancel".

- 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.

Change mode at any time by clicking the "Mode" button containing Mu's logo.

OK Cancel

BBC micro:bit

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



Built-in 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:

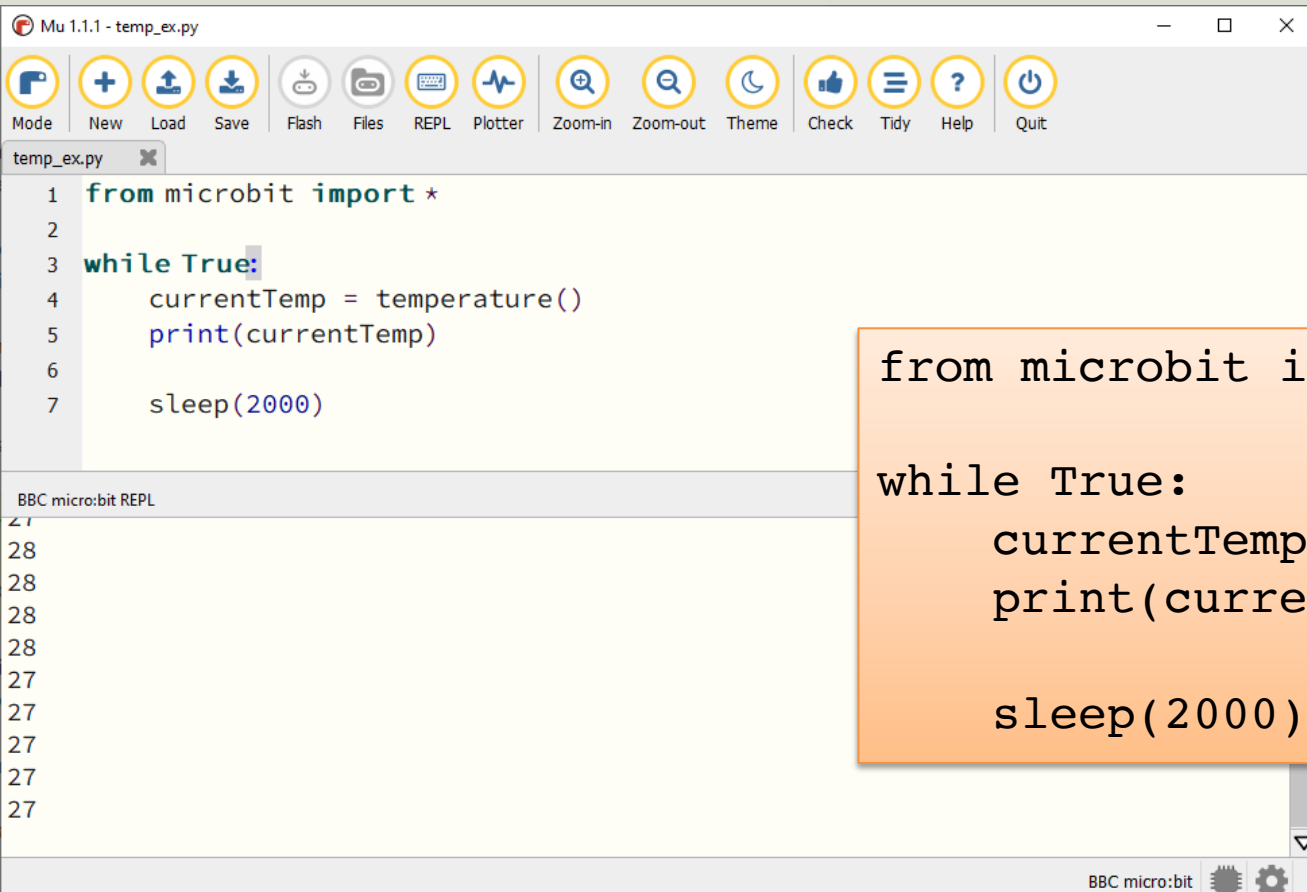
```
from microbit import *  
  
currentTemp = temperature()
```

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)
```

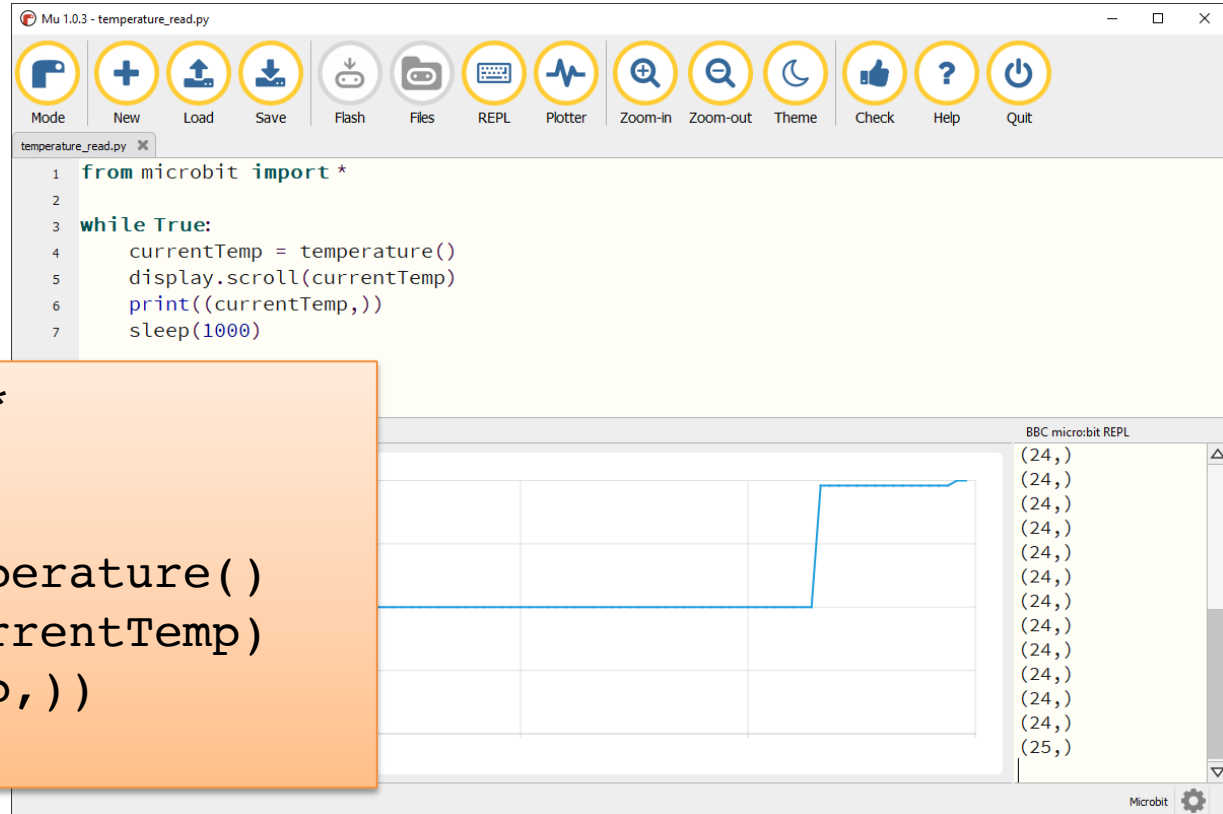
Below the editor is the "BBC micro:bit REPL" window, which shows a series of "27" characters, indicating the program is running and printing the temperature value.

```
from microbit import *

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

    sleep(2000)
```

Temperature Sensor



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

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

The bottom right panel shows the BBC micro:bit REPL with a list of output values: (24,) repeated 14 times, followed by (25,). The bottom left panel shows a plot of the temperature data, which is a horizontal line at 24 that steps up to 25.

```
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.

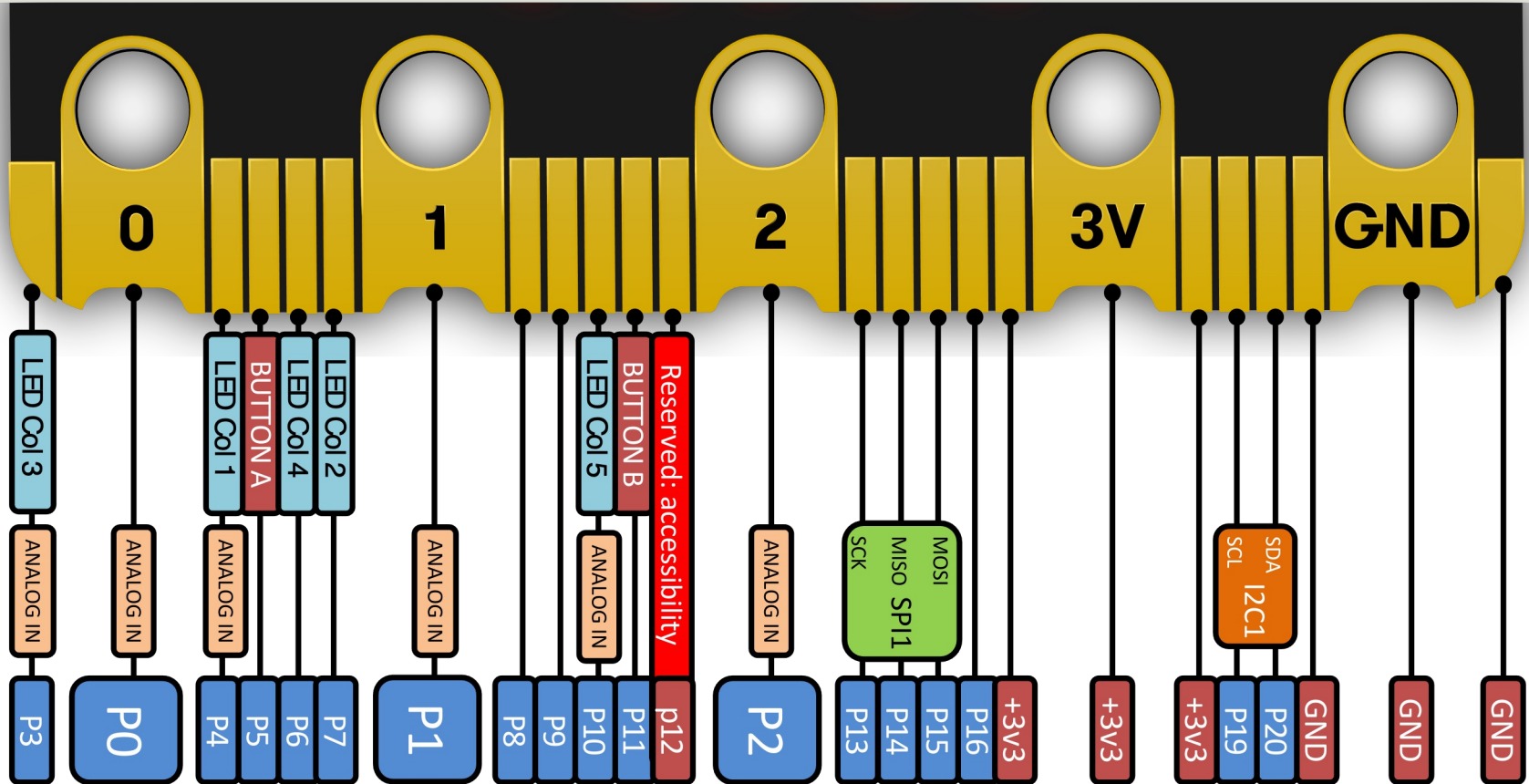
If you click A Button, the Minimum Temperature for the period (since you started the program/turned on the Micro:bit) is shown on the LED matrix

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

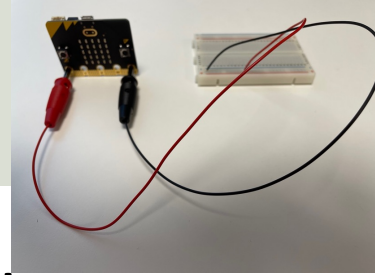


micro:bit I/O Pins

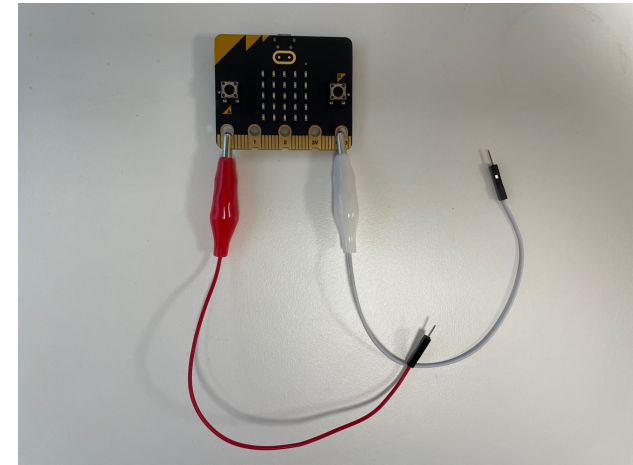
micro:bit I/O Pin Overview



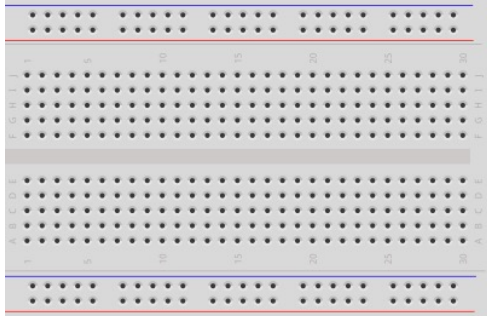
I/O Pins



- 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

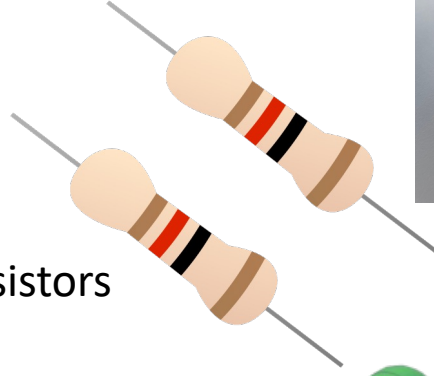


Breadboard

Temperature Sensor

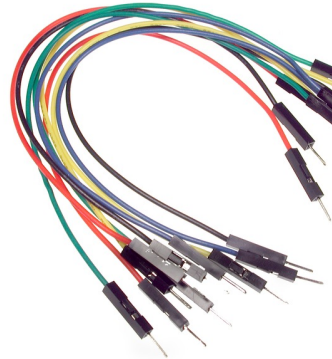


Resistors

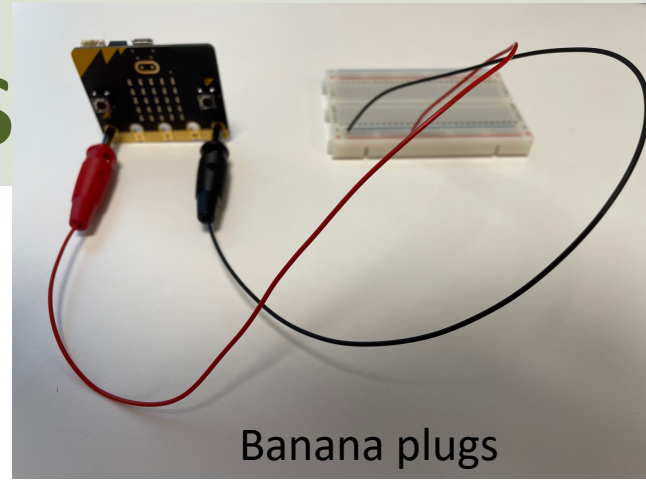


LEDs

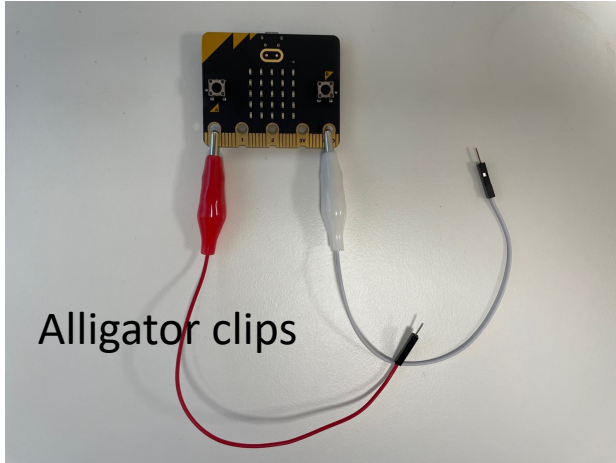
Wires



Multimeter



Banana plugs



Alligator clips

Types of I/O Pins

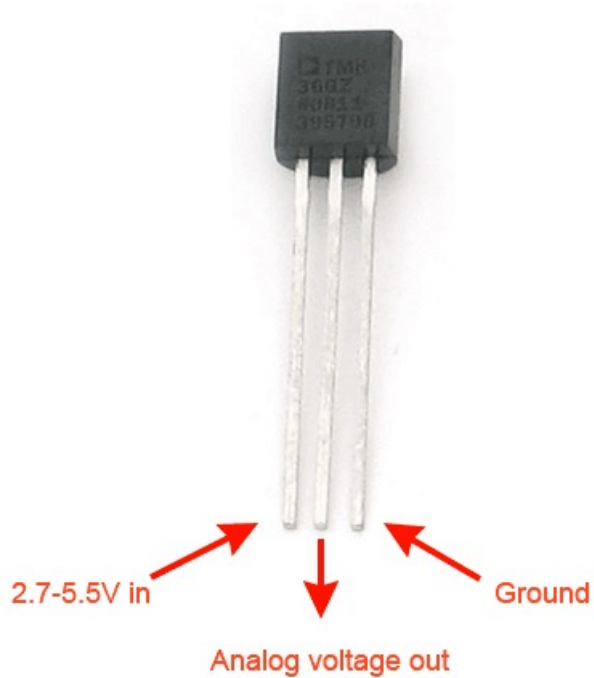
- Analog/Digital Input/Output Pins
- Pulse Width Modulation (PWM)
- SPI
- I2C
- UART (used for serial communication)

We will only use an Analog Input pin in this Tutorial



TMP36 Temperature Sensor

TMP36 Temperature Sensor

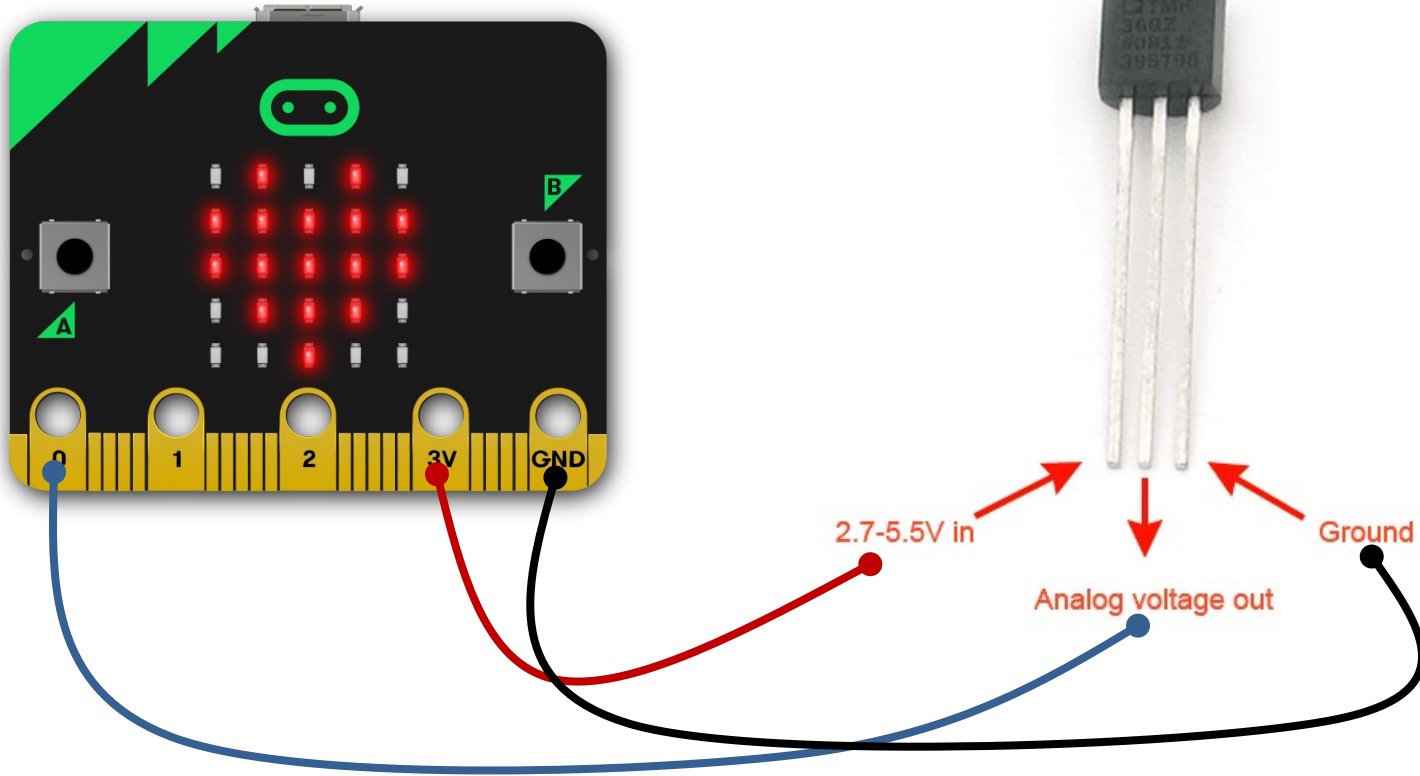


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

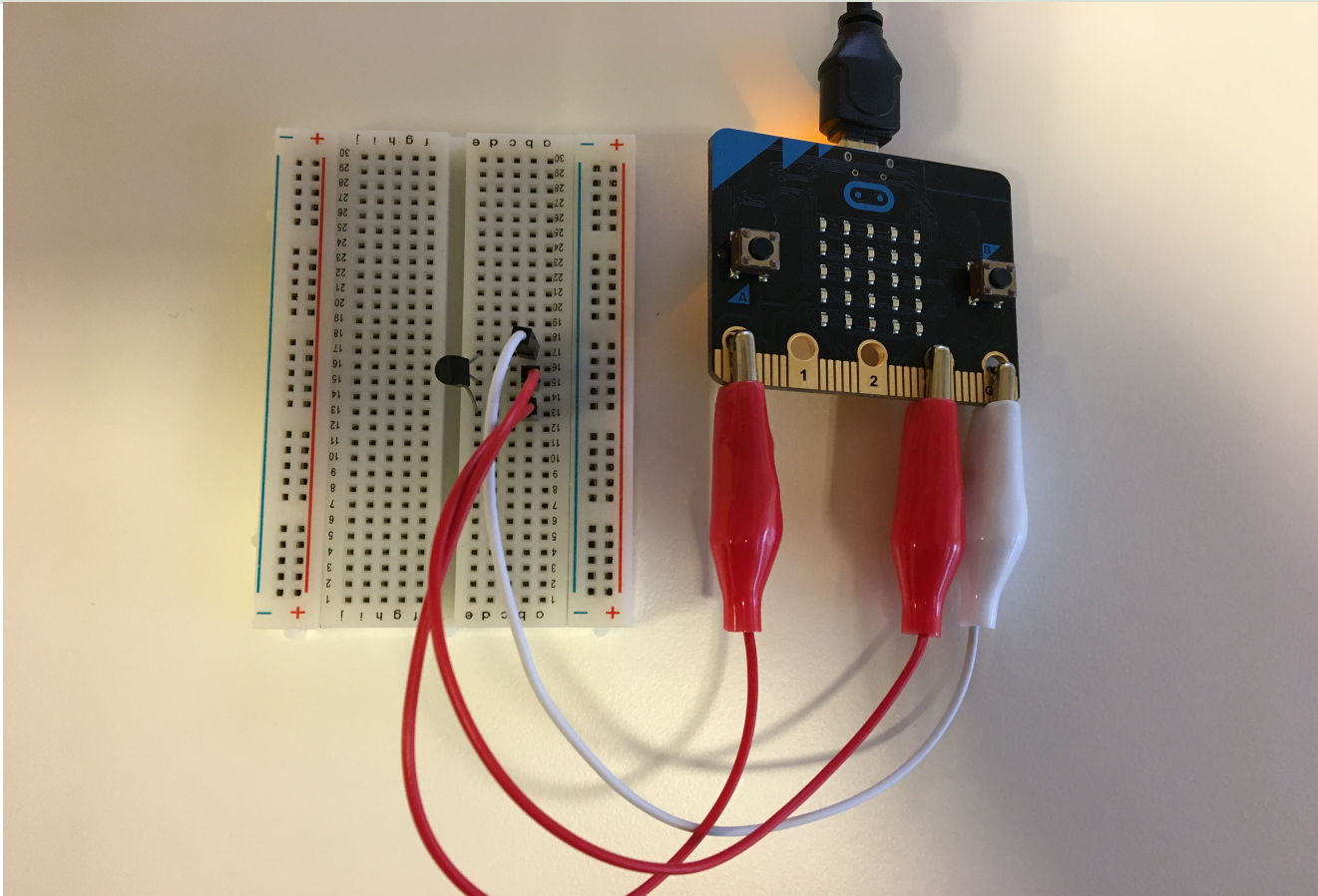
They use the fact as temperature increases, the voltage across a diode increases at a known rate.

<https://learn.adafruit.com/tmp36-temperature-sensor>

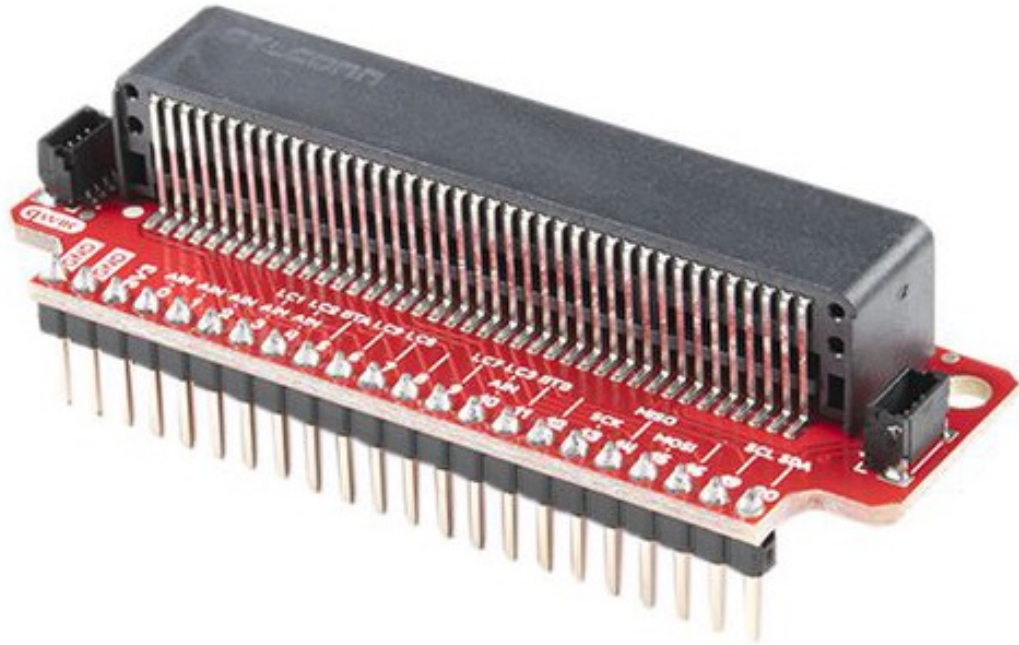
Wiring



Breadboard and Crocodile clips



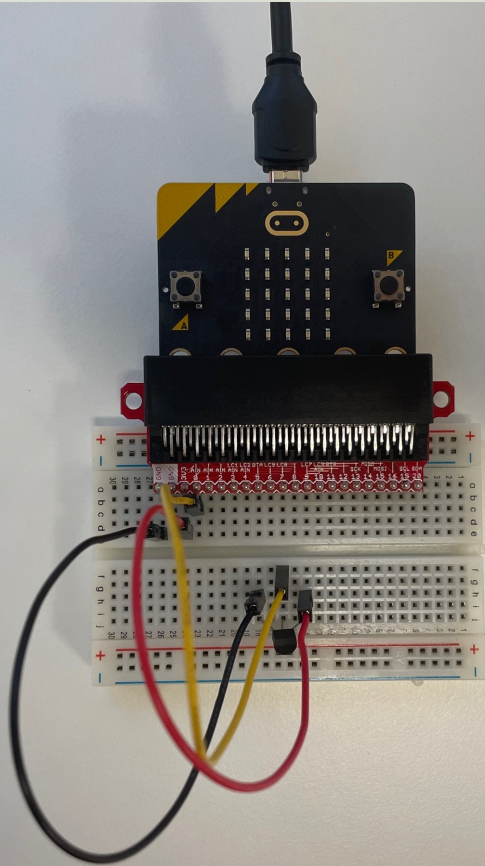
Adapter Breakout Board for micro:bit



We can also use an **Adapter Breakout Board for micro:bit** instead of Alligator/Crocodile clips

This makes it easier to wire for more advanced circuits and use of more in inputs/outputs pins

Adapter Breakout Board for micro:bit



Here you see see the wirings using an Adapter Breakout Board for micro:bit

Python

```
from microbit import *  
  
while True:  
    adc = pin0.read_analog()  
    display.scroll(adc)  
    sleep(5000)
```

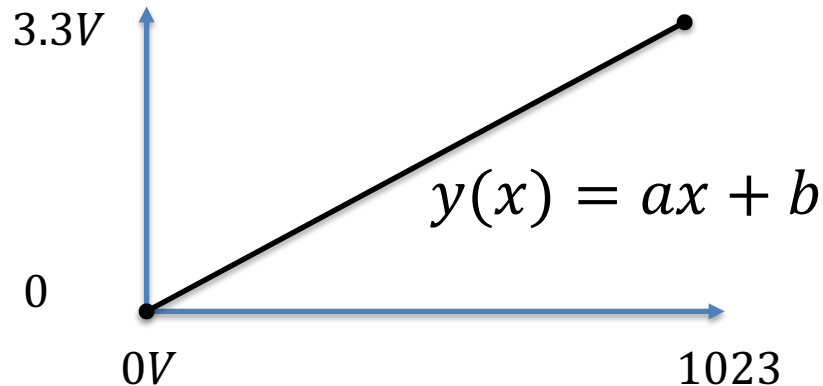

ADC Value to Voltage Value

Analog Pins: The the built-in analog-to-digital converter (ADC) on micro:bit is 10bit, producing values from 0 to 1023.

The function `pin0.read_analog()` gives a value between 0 and 1023. It must be converted to a Voltage Signal 0 - 3.3v

ADC = 0 -> 0v

ADC = 1023 -> 3.3v



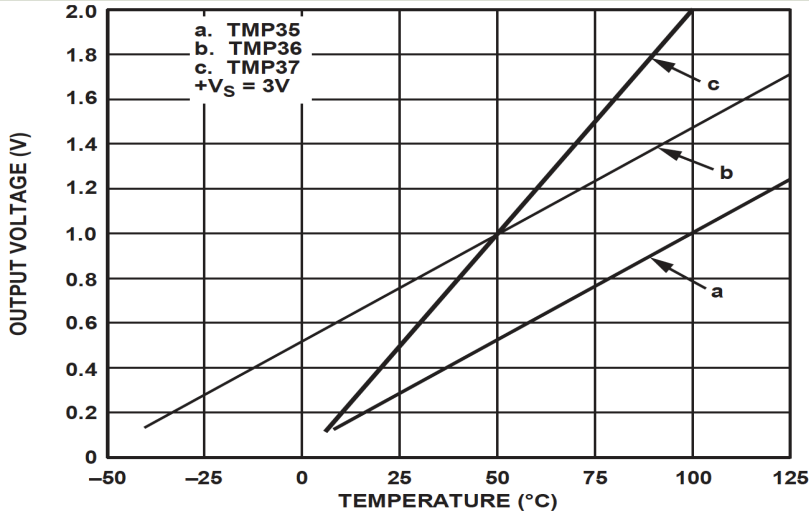
This gives the following conversion formula:

$$y(x) = \frac{3.3}{1023}x$$

Python

```
from microbit import *  
  
while True:  
    adc = pin0.read_analog()  
    volt = (3.3/1023)*adc  
    display.scroll(volt)  
    sleep(5000)
```

Voltage to degrees Celsius



Convert from Voltage (V) to degrees Celsius

From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25^{\circ}C)$$

$$(x_2, y_2) = (1V, 50^{\circ}C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

This gives:

$$y - 25 = \frac{50 - 25}{1 - 0.75} (x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

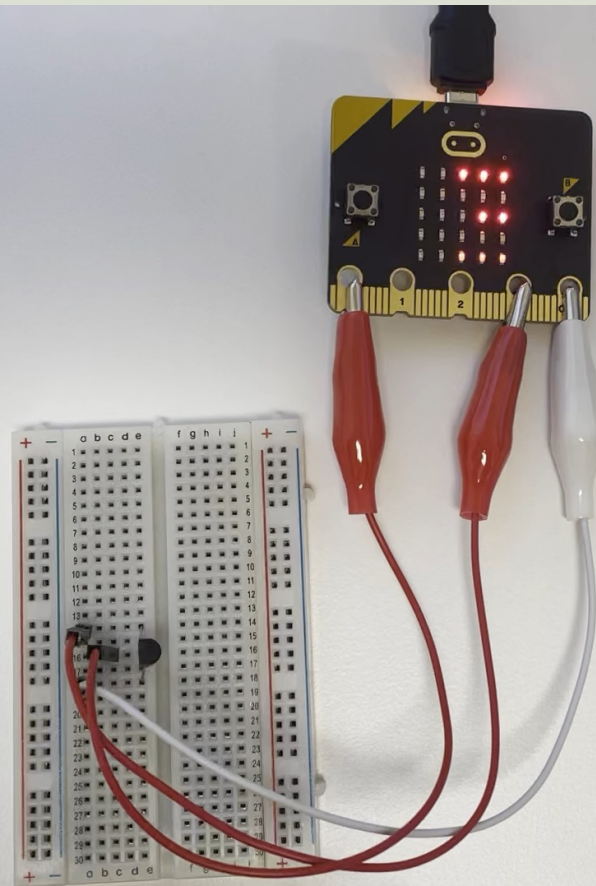
We can find a and b using the following known formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

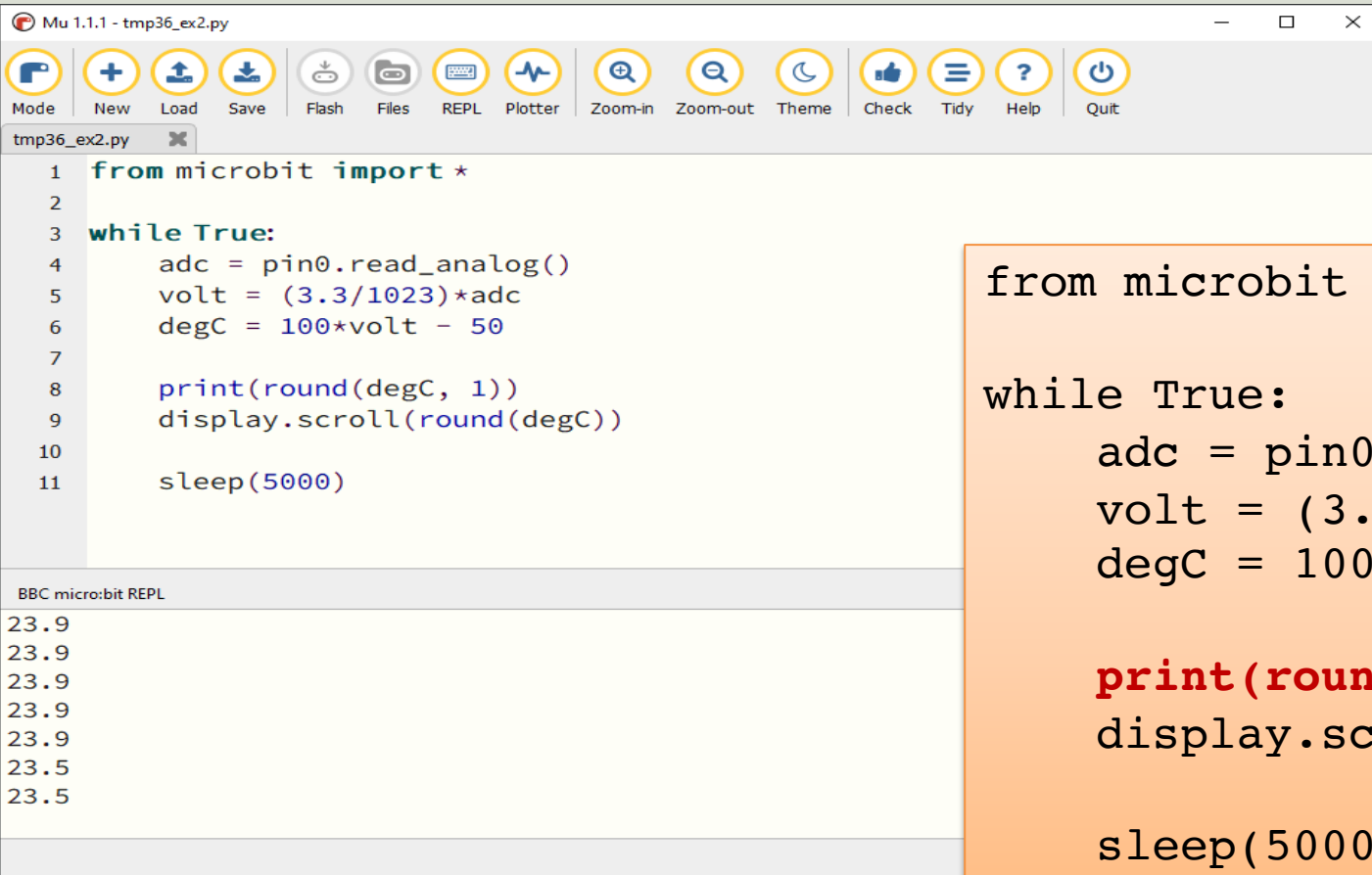
Python

```
from microbit import *  
  
while True:  
    adc = pin0.read_analog()  
    volt = (3.3/1023)*adc  
    degC = 100*volt - 50  
    display.scroll(round(degC))  
    sleep(5000)
```

Results



Printing to REPL



The screenshot shows the Mu Python IDE interface. The title bar reads "Mu 1.1.1 - tmp36_ex2.py". The menu bar includes: Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The editor window shows the following Python code:

```
1 from microbit import *
2
3 while True:
4     adc = pin0.read_analog()
5     volt = (3.3/1023)*adc
6     degC = 100*volt - 50
7
8     print(round(degC, 1))
9     display.scroll(round(degC))
10
11     sleep(5000)
```

The bottom panel, labeled "BBC micro:bit REPL", shows the output of the program:

```
23.9
23.9
23.9
23.9
23.5
23.5
```

```
from microbit import *
```

```
while True:
```

```
    adc = pin0.read_analog()
```

```
    volt = (3.3/1023)*adc
```

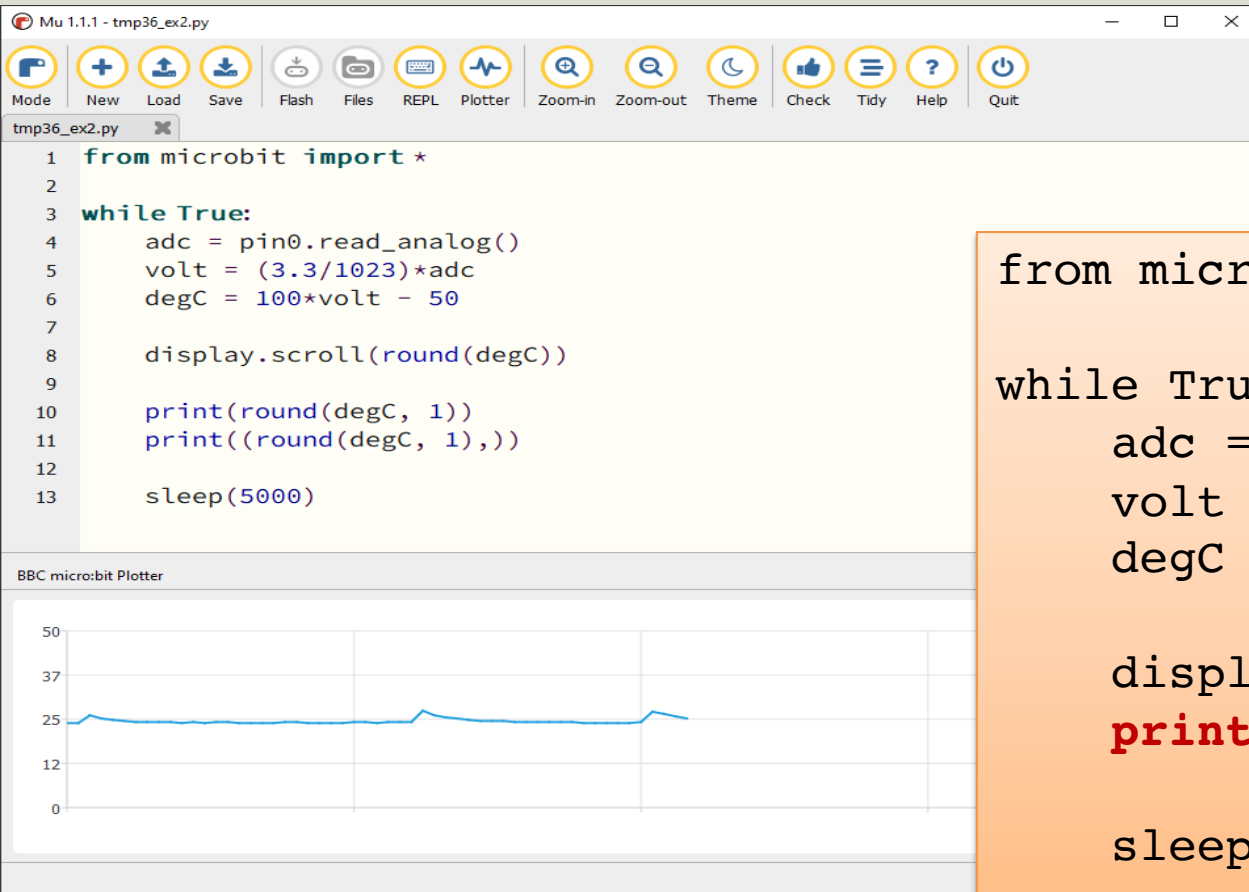
```
    degC = 100*volt - 50
```

```
    print(round(degC, 1))
```

```
    display.scroll(round(degC))
```

```
    sleep(5000)
```

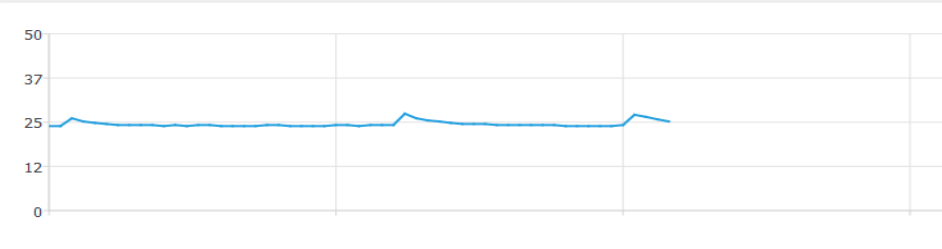
Plotting



The screenshot shows the Mu Python IDE interface. The top toolbar includes icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The main editor displays a Python script for temperature measurement. Below the editor is the BBC micro:bit Plotter window, which shows a line graph of the temperature data over time. The y-axis ranges from 0 to 50, and the x-axis represents time. The plot shows a blue line fluctuating around a value of approximately 25 degrees Celsius.

```
1 from microbit import *
2
3 while True:
4     adc = pin0.read_analog()
5     volt = (3.3/1023)*adc
6     degC = 100*volt - 50
7
8     display.scroll(round(degC))
9
10    print(round(degC, 1))
11    print((round(degC, 1),))
12
13    sleep(5000)
```

BBC micro:bit Plotter



Time	Temperature (degC)
0	25.0
1	25.0
2	25.0
3	25.0
4	25.0
5	25.0
6	25.0
7	25.0
8	25.0
9	25.0
10	25.0
11	25.0
12	25.0
13	25.0
14	25.0
15	25.0
16	25.0
17	25.0
18	25.0
19	25.0
20	25.0
21	25.0
22	25.0
23	25.0
24	25.0
25	25.0
26	25.0
27	25.0
28	25.0
29	25.0
30	25.0
31	25.0
32	25.0
33	25.0
34	25.0
35	25.0
36	25.0
37	25.0
38	25.0
39	25.0
40	25.0
41	25.0
42	25.0
43	25.0
44	25.0
45	25.0
46	25.0
47	25.0
48	25.0
49	25.0

```
from microbit import *

while True:
    adc = pin0.read_analog()
    volt = (3.3/1023)*adc
    degC = 100*volt - 50

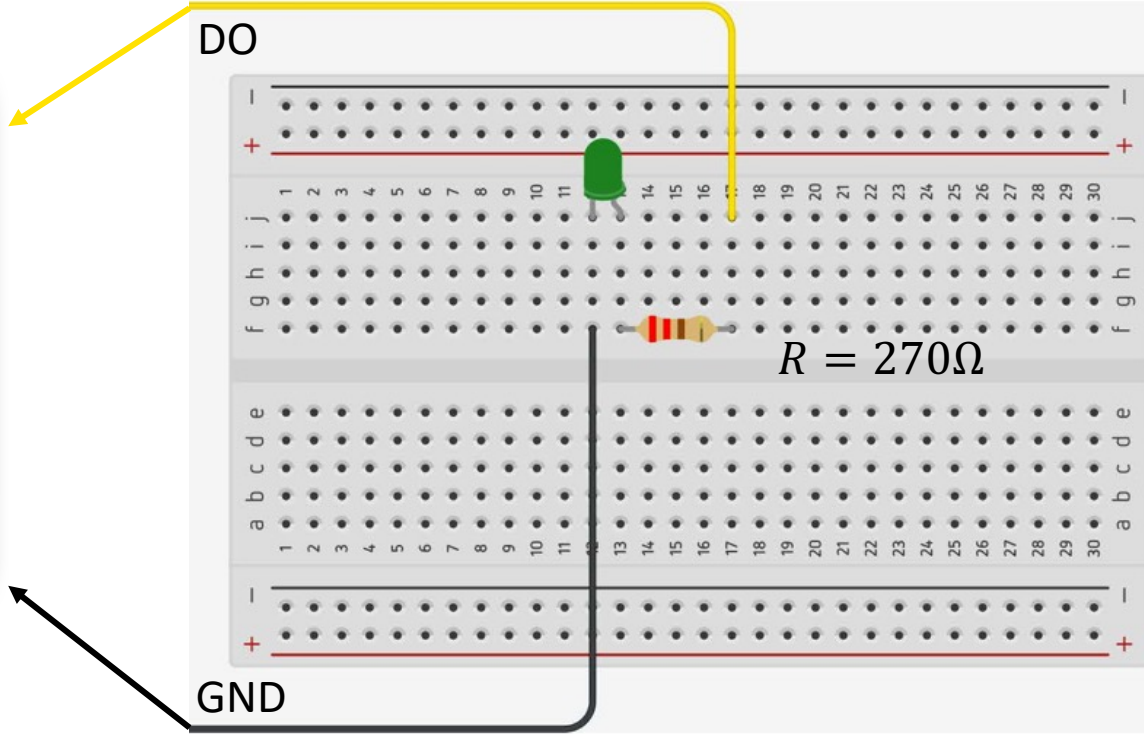
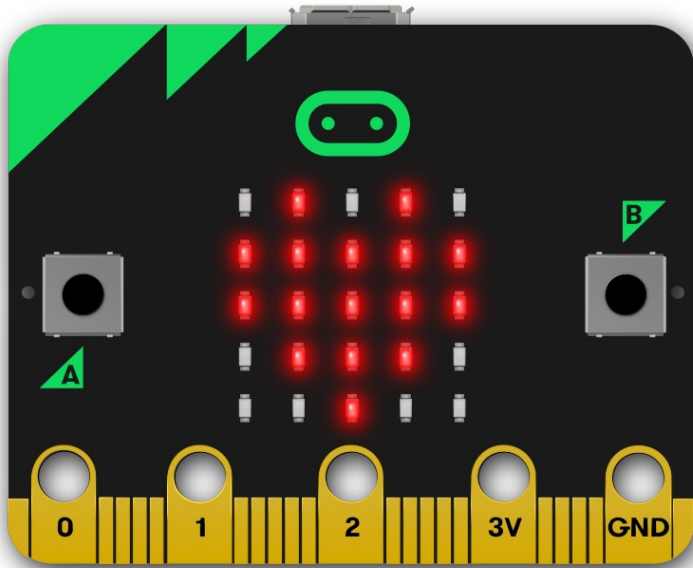
    display.scroll(round(degC))
    print((round(degC, 1),))

    sleep(5000)
```



Temperature with Alarm

LED Wiring



Python



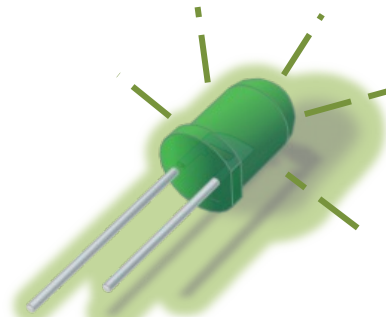
Temperature > Limit?

No



LED OFF

Yes



LED ON

```
from microbit import *

alarmLimit = 28

while True:
    adc = pin0.read_analog()
    volt = (3.3/1023)*adc
    degC = 100*volt - 50

    display.scroll(round(degC))

    print(round(degC, 1))

    if degC > alarmLimit:
        print("Alarm")
        pin1.write_digital(1)
    else:
        pin1.write_digital(0)

    sleep(5000)
```



Mode



New



Load



Save



Flash



Files



REPL



Plotter



Zoom-in



Zoom-out



Theme



Check



Tidy



Help



Quit

tmp36_led.py

```
1 from microbit import *
2
3 alarmLimit = 28
4
5 while True:
6     adc = pin0.read_analog()
7     volt = (3.3/1023)*adc
8     degC = 100*volt - 50
9
10    display.scroll(round(degC))
11
12    print(round(degC, 1))
13
14    if degC > alarmLimit:
15        print("Alarm")
16        pin1.write_digital(1)
17    else:
18        pin1.write_digital(0)
19
20    sleep(5000)
```

BBC micro:bit REPL

```
27.7
27.7
27.7
28.1
Alarm
27.7
27.4
```



Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

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

