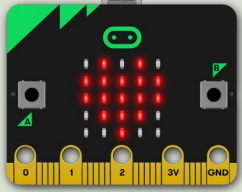


<https://www.halvorsen.blog>



micro:bit and Logging Sensor Data

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
- Data Logging using the TMP36 Sensor
 - Python Examples

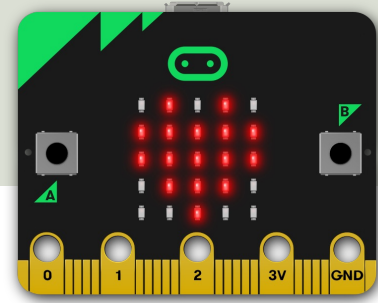


Introduction to micro:bit

Hans-Petter Halvorsen

[Table of Contents](#)

micro:bit

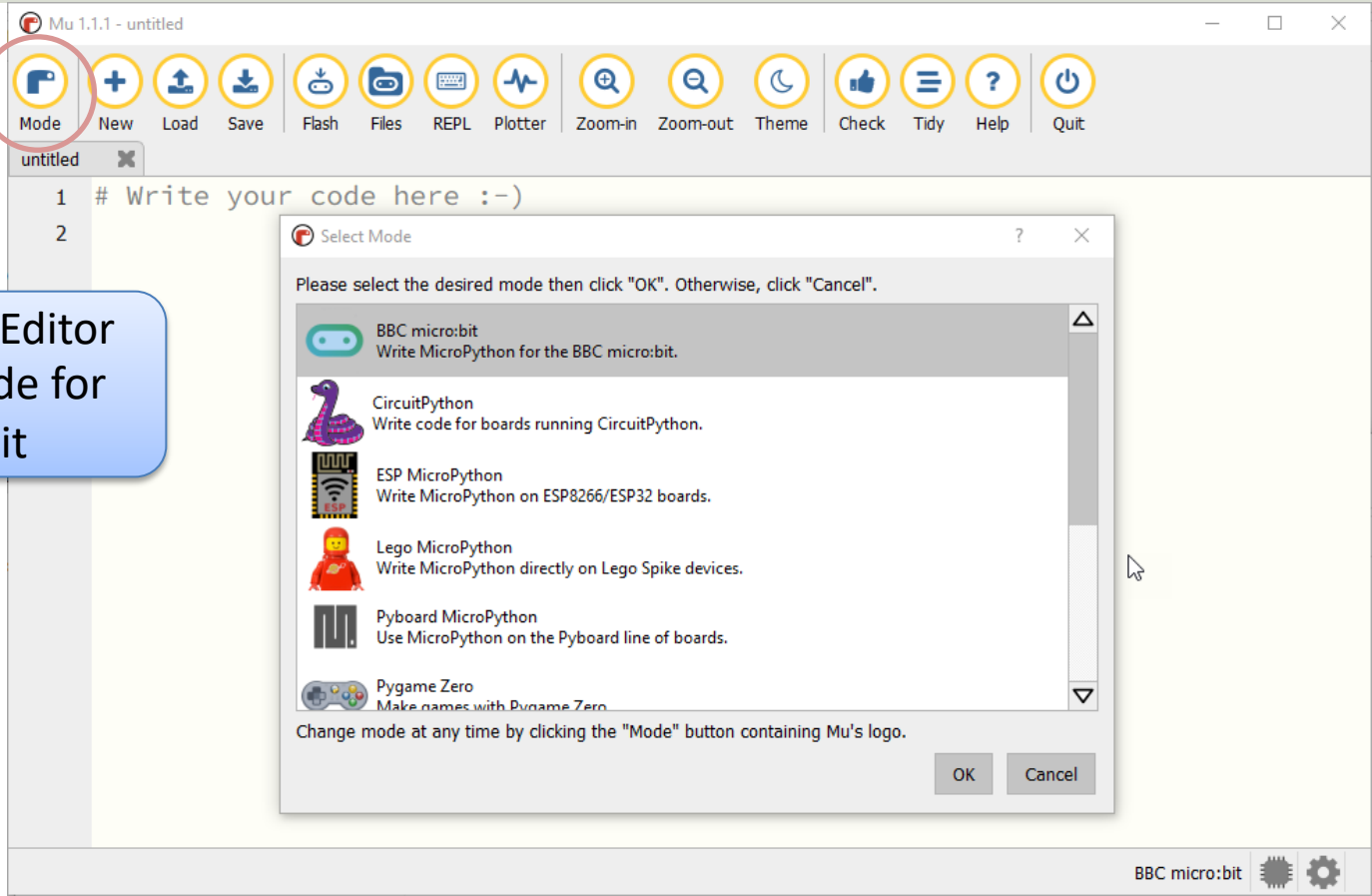


- micro:bit is a small microcontroller
- micro:bit is smaller than a credit card
- Price is about 150-400NOK (\$15-30)
- It can be used by kids and students to learn programming and technology
- micro:bit can run a special version of Python called MicroPython
- MicroPython is a down-scaled version of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



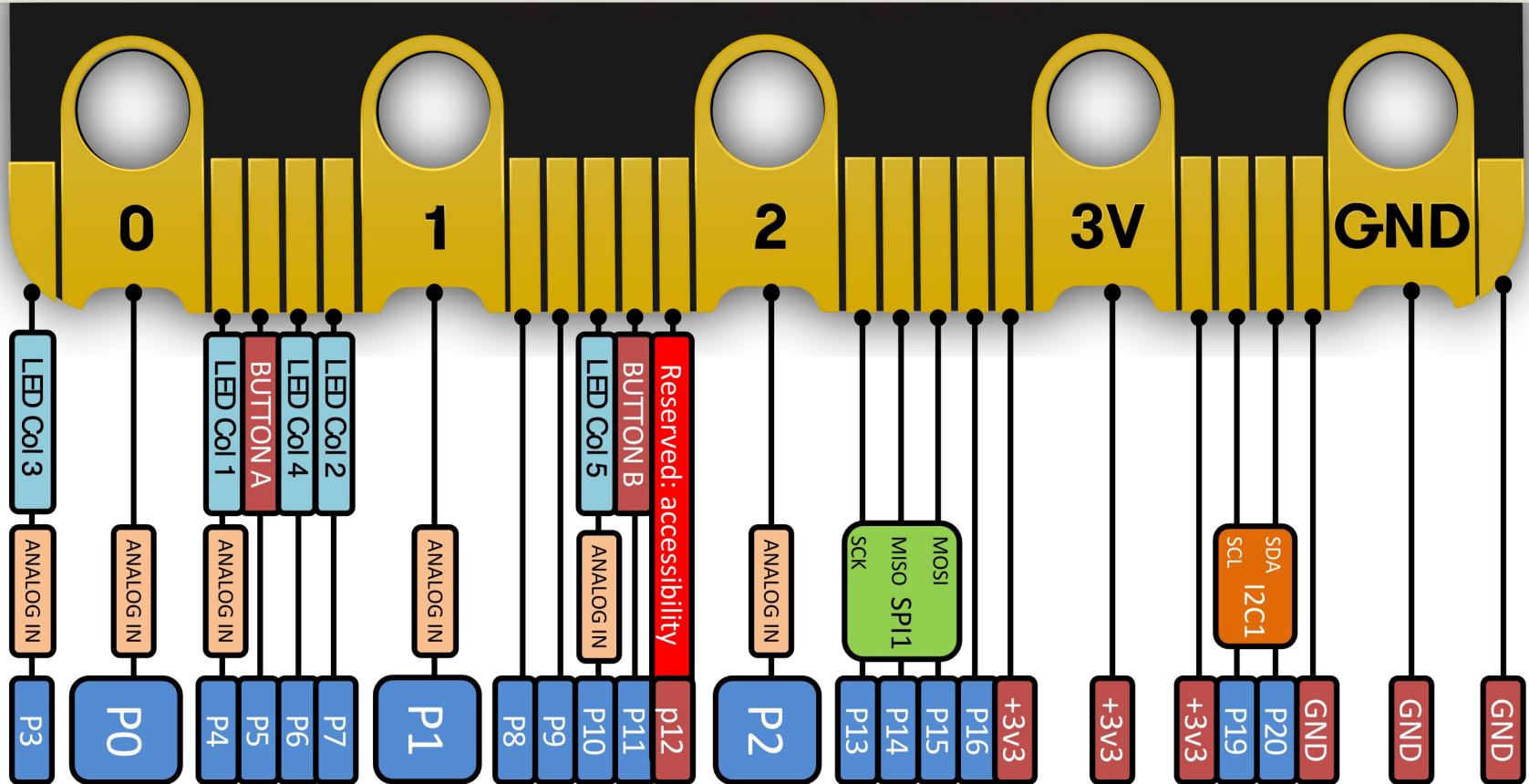
The screenshot shows the Mu Python Editor interface. The title bar reads "Mu 1.1.1 - untitled". The toolbar contains several icons, with the "Mode" icon (a blue folder with a white 'P') circled in red. Below the toolbar, the code editor shows two lines of text: "1 # Write your code here :-)" and "2".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

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

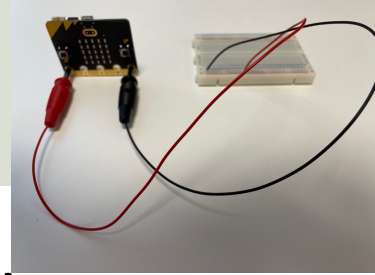


micro:bit I/O Pins

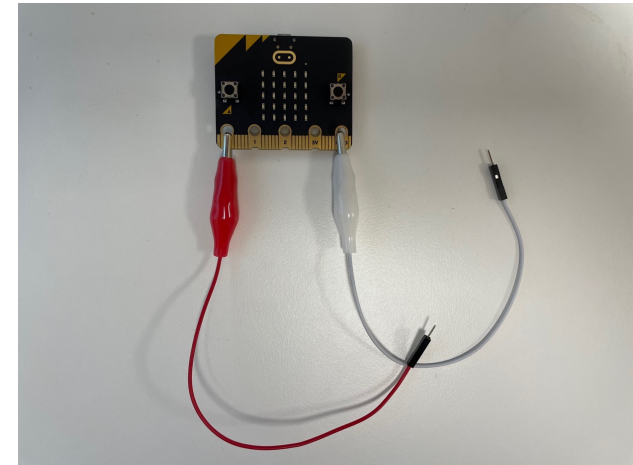
micro:bit I/O Pin Overview



I/O Pins



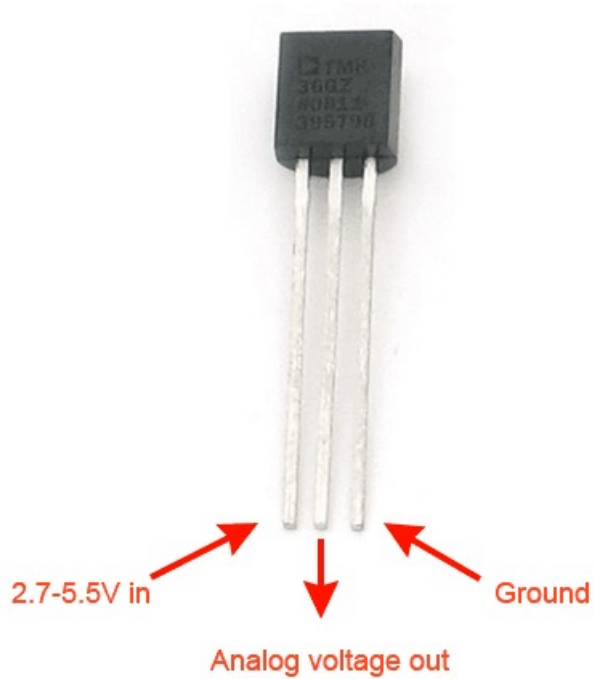
- We use the I/O pins to connect external components like LEDs, different types of Sensors, etc.
- You can use 4mm Banana plugs or Alligator/Crocodile clips
- Typically, you also want to use a Breadboard





TMP36 Temperature Sensor

TMP36 Temperature Sensor

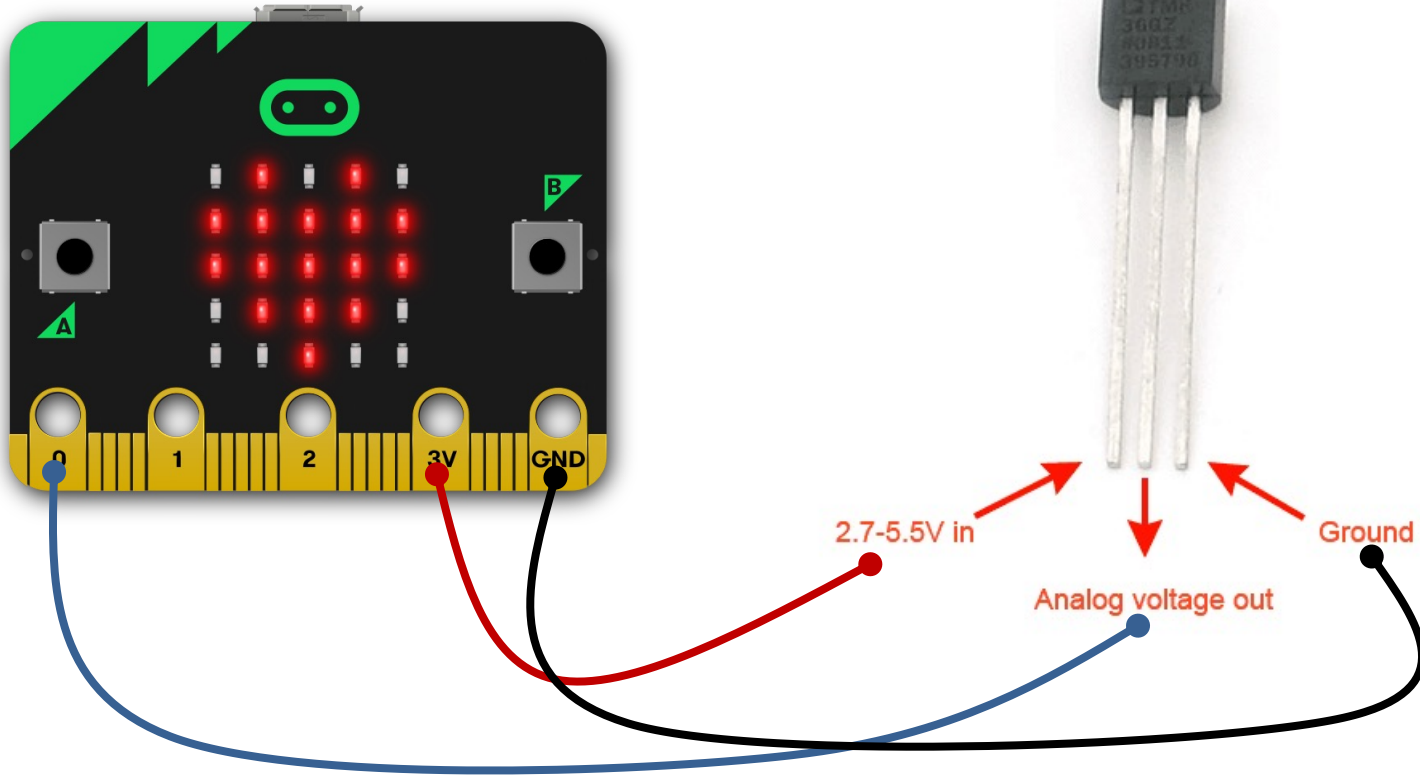


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

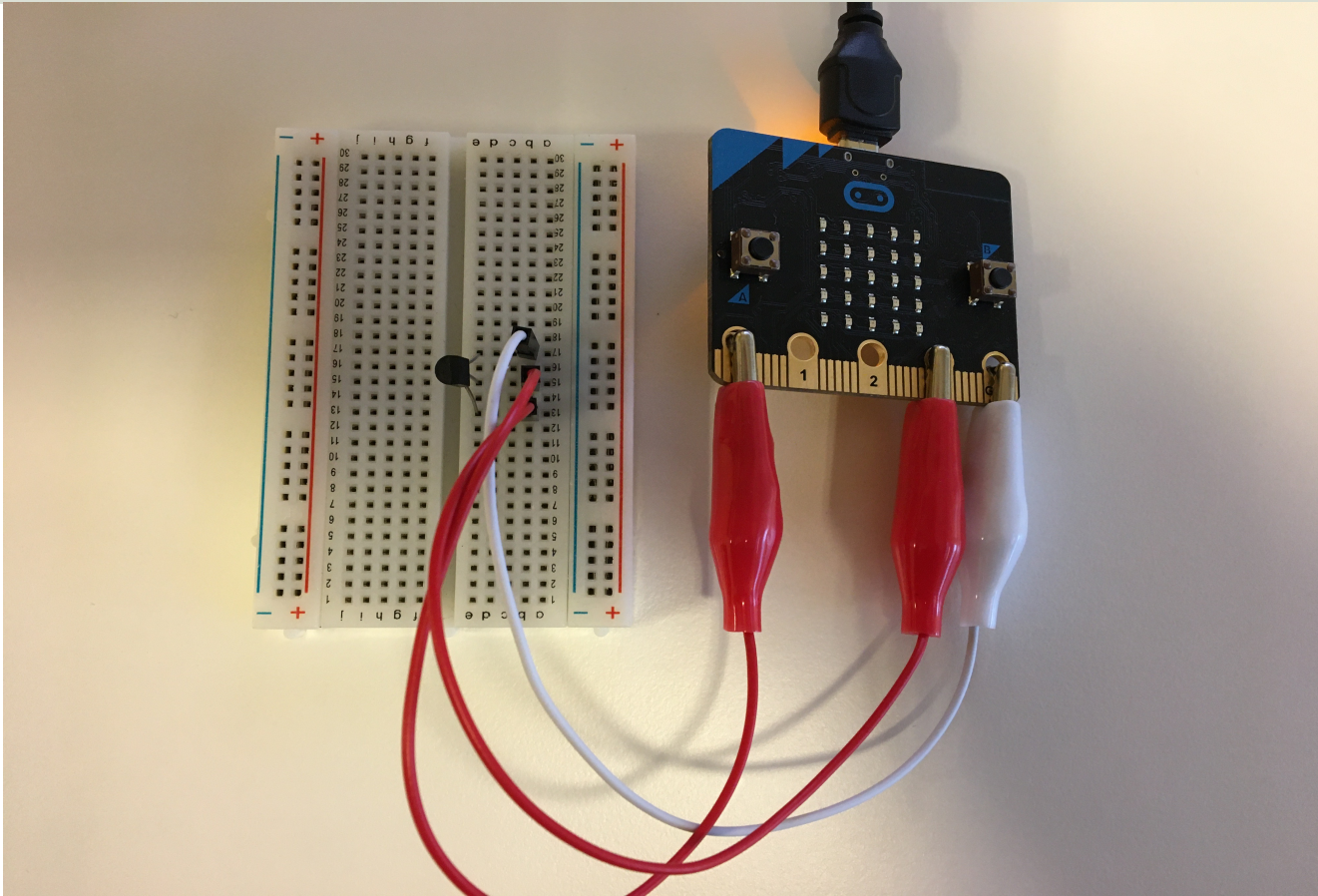
They use the fact as temperature increases, the voltage across a diode increases at a known rate.

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

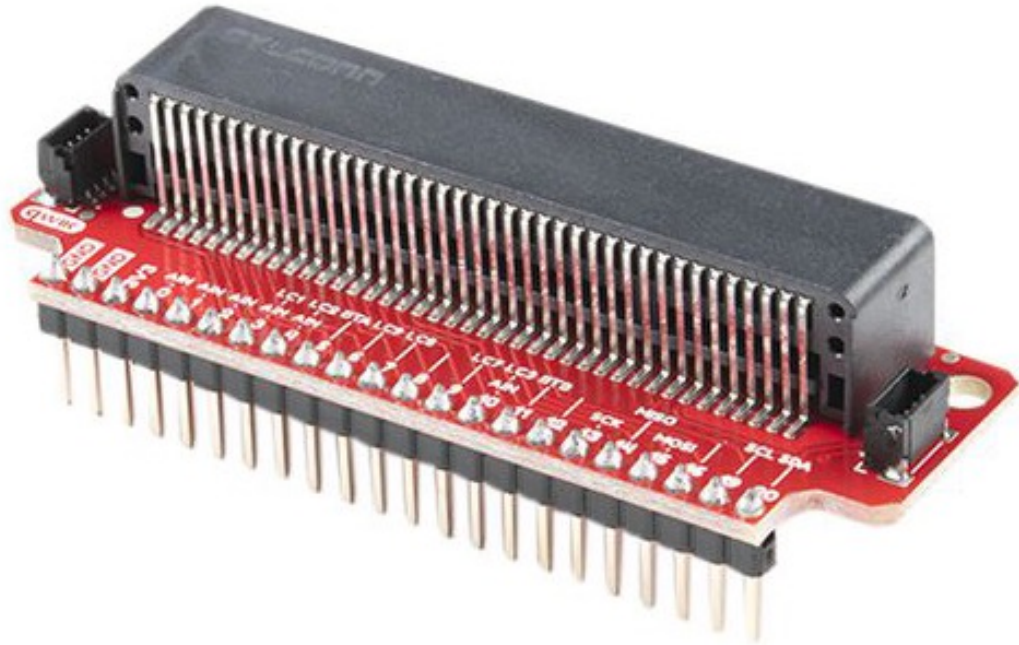
Wiring



Breadboard and Crocodile clips



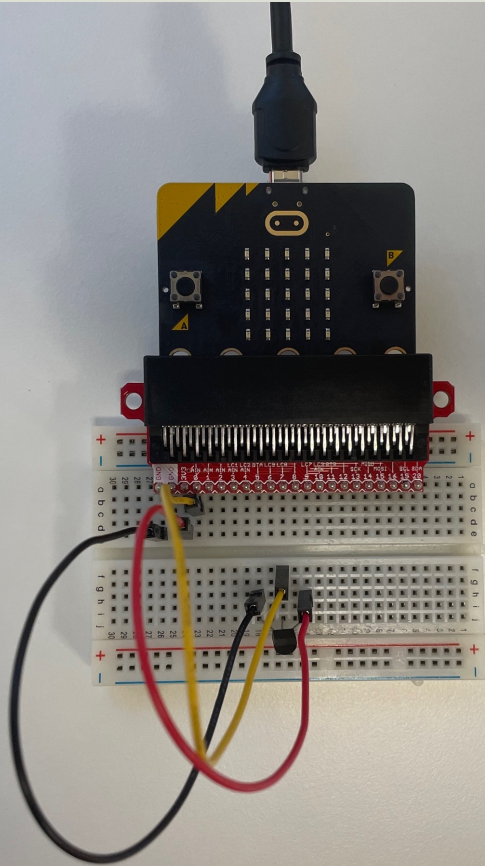
Adapter Breakout Board for micro:bit



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)
```

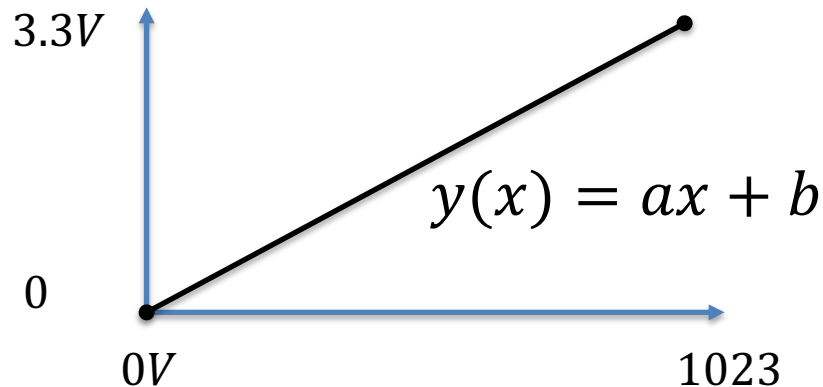

ADC Value to Voltage Value

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



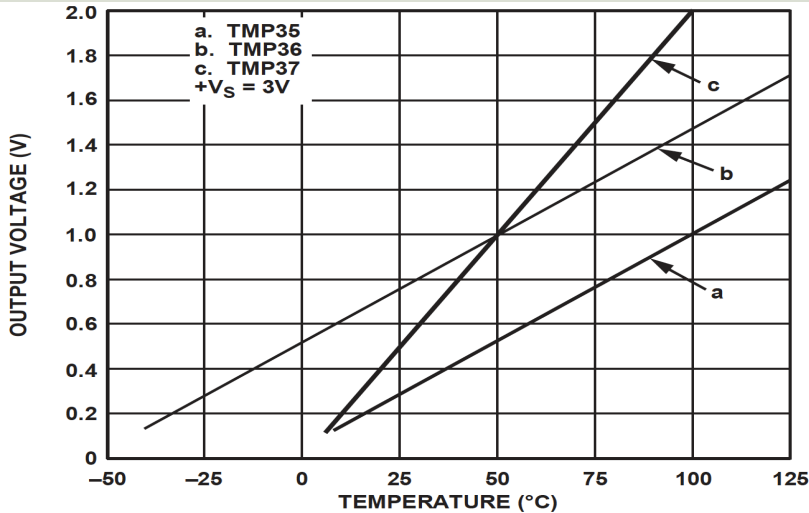
This gives the following conversion formula:

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

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:

$$y = 100x - 50$$

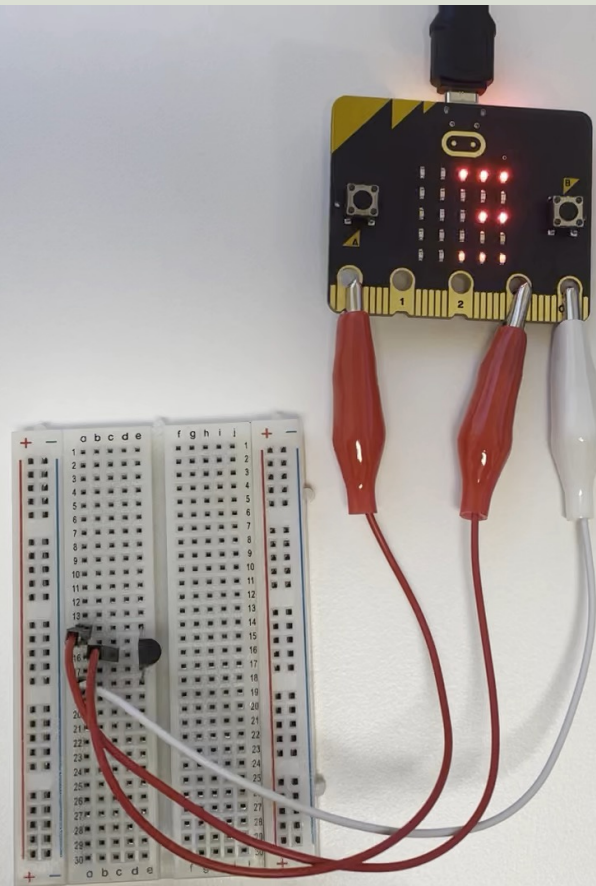
We can find a and b using the following known formula:

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

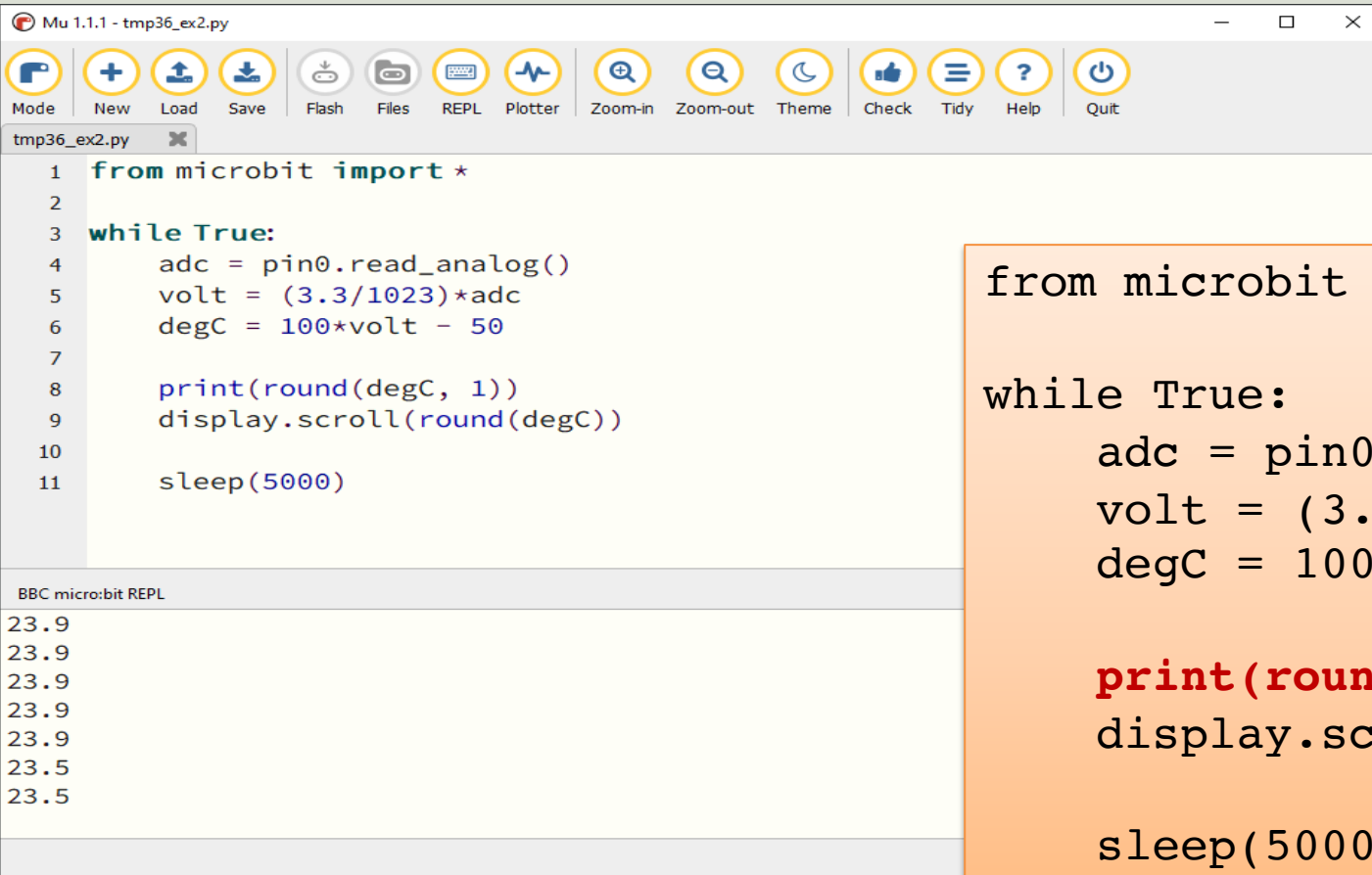
Python

```
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 icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The editor window displays 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
```

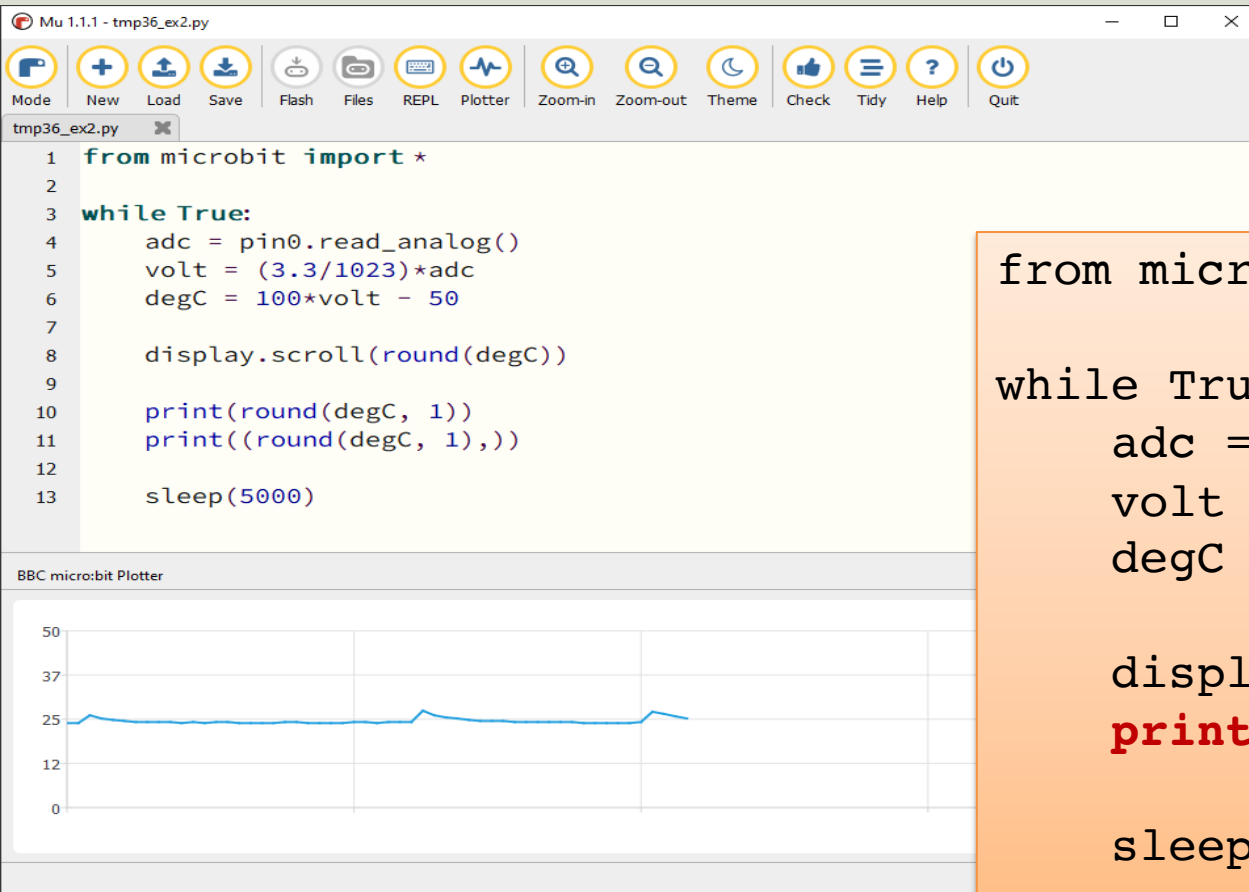
```
from microbit import *

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 window displays a Python script for measuring temperature using a microbit's ADC. The script is as follows:

```
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)
```

Below the code editor is the BBC micro:bit Plotter window, which displays a line graph of the temperature data. The y-axis ranges from 0 to 50 degrees Celsius, and the x-axis represents time. The plot shows a blue line that fluctuates slightly around a mean value of approximately 25 degrees Celsius.

```
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)
```



Data Logging

Data Logging

It is possible to Log Data from different Sensors, etc. to a Text File that is stored on the micro:bit device

Main Functions:

Replace x, y, etc. with the name you will use

```
log.set_labels( 'x', 'z', 'z' )
```

```
log.add( {  
    'x': datax,  
    'y': datay,  
    'z': dataz  
})
```

```
log.delete( )
```

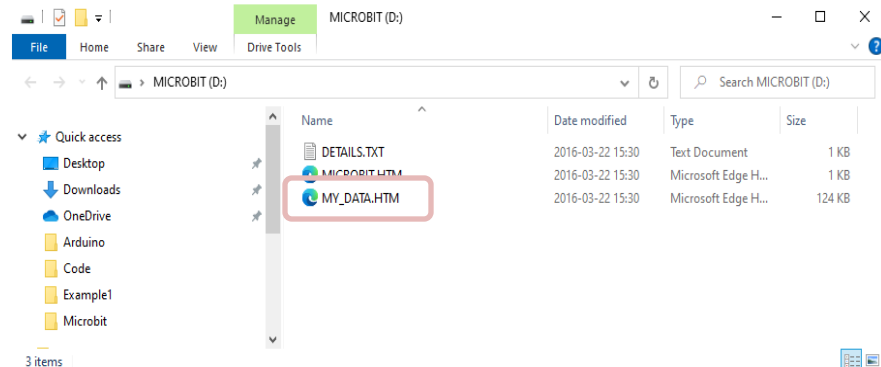
You can also choose mirror the data logging to the serial output (REPL):

```
log.set_mirroring(True)
```

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

Data Logging

- The micro:bit can log sensor data to a file on the micro:bit.
- After logging your data, plug the micro:bit into a computer and open **MY_DATA.HTM** on the MICROBIT drive to view your data in a web browser
- You can also download it as a CSV file for use in a spreadsheet like Excel, etc.



Python

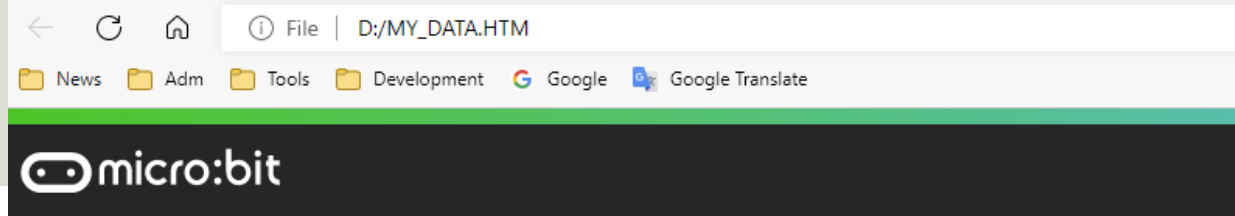
```
def tempTmp36():  
    adc = pin0.read_analog()  
    volt = (3.3/1023)*adc  
    degC = 100*volt - 50  
  
    degC = round(degC, 1)  
  
    return degC  
  
def c2f(Tc):  
    Tf = (Tc*9/5) + 32  
    Tf = round(Tf, 1)  
    return Tf  
  
def saveData(temp):  
    fahr = c2f(temp)  
    log.add({  
        'k': k,  
        'temp_c': temp,  
        'temp_f': fahr  
    })
```

```
from microbit import *  
import log  
  
log.set_labels('k', 'temp_c', 'temp_f')
```

```
while True:  
    degC = tempTmp36()  
    saveData(degC)  
  
    display.scroll(round(degC))  
    print(round(degC, 1))  
    k = k + 1  
    sleep(5000)
```

Data

The Data are stored in a File **MY_DATA.HTM**



The screenshot shows a web browser window with the address bar displaying 'D:/MY_DATA.HTM'. The browser's navigation bar includes icons for back, refresh, and home, along with search engines like Google and Google Translate. Below the browser, there is a dark header with the 'micro:bit' logo. The main content area features the title 'micro:bit data log' and five interactive buttons: 'Download', 'Copy', 'Update data...', 'Clear log...', and 'Visual preview'. Below these buttons is a paragraph of text explaining the data and providing a link to learn more. At the bottom, there is a table with four columns: 'Time (seconds)', 'k', 'temp_c', and 'temp_f', containing eight rows of data.

micro:bit data log

Download

Copy

Update data...

Clear log...

Visual preview

This is the data on your micro:bit. To analyse it and create your own graphs, transfer it to your computer. You can copy and paste your data, or download it as a CSV file which you can import into a spreadsheet or graphing tool. [Learn more about micro:bit data logging.](#)

Time (seconds)	k	temp_c	temp_f
0.26	0	25.8	78.4
7.51	1	25.8	78.4
14.77	2	25.8	78.4
22.02	3	27.4	81.3
29.27	4	29.7	85.5
36.38	5	29.4	84.90001
43.63	6	28.1	82.6
1.20	0	28.1	82.6

CSV

The screenshot shows a Microsoft Excel spreadsheet with the following data:

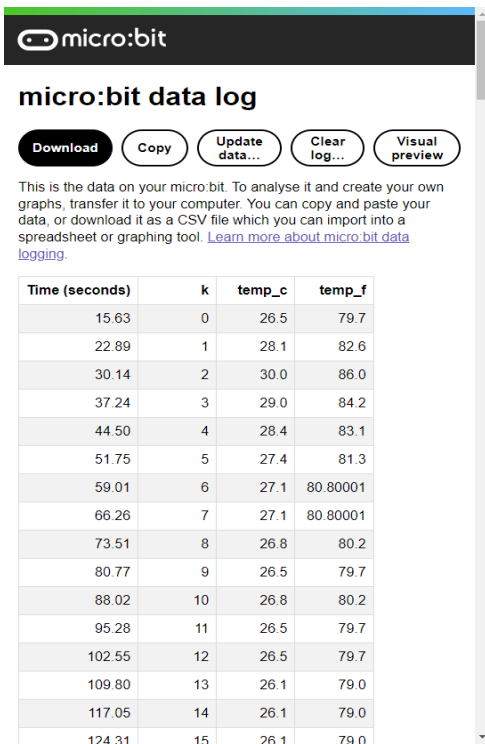
Time (seconds)	temp_c	temp_f
0.26	0	25.8
7.51	1	25.8
14.77	2	25.8
22.02	3	27.4
29.27	4	29.7
36.38	5	29.4
43.63	6	28.1
1.2	0	28.1

The Windows File Explorer window shows the file 'microbit.csv' in the Downloads folder, with a date modified of 2022-09-07 13:40.

You can download as a CSV File and open it in, e.g., Excel, etc.

Improved Ex.

Here, the Logging starts when pushing the “Button A” on the micro:bit



The screenshot shows the 'micro:bit data log' interface. At the top, there's a 'micro:bit' logo. Below it, the title 'micro:bit data log' is displayed. There are four buttons: 'Download', 'Copy', 'Update data...', and 'Clear log...'. To the right of these buttons is a 'Visual preview' button. Below the buttons, there is a paragraph of text explaining that the data is from the micro:bit and can be used for analysis or downloaded as a CSV file. A link 'Learn more about micro:bit data logging.' is provided. At the bottom, there is a table with four columns: 'Time (seconds)', 'k', 'temp_c', and 'temp_f'. The table contains 15 rows of data.

Time (seconds)	k	temp_c	temp_f
15.63	0	26.5	79.7
22.89	1	28.1	82.6
30.14	2	30.0	86.0
37.24	3	29.0	84.2
44.50	4	28.4	83.1
51.75	5	27.4	81.3
59.01	6	27.1	80.80001
66.26	7	27.1	80.80001
73.51	8	26.8	80.2
80.77	9	26.5	79.7
88.02	10	26.8	80.2
95.28	11	26.5	79.7
102.55	12	26.5	79.7
109.80	13	26.1	79.0
117.05	14	26.1	79.0
124.31	15	26.1	79.0

```
from microbit import *
import log

def tempTmp36():
    adc = pin0.read_analog()
    volt = (3.3/1023)*adc
    degC = 100*volt - 50

    degC = round(degC, 1)

    return degC

def c2f(Tc):
    Tf = (Tc*9/5) + 32
    Tf = round(Tf, 1)
    return Tf

def saveData(temp):
    fahr = c2f(temp)
    log.add({
        'k': k,
        'temp_c': temp,
        'temp_f': round(fahr,1)
    })

logging = False
display.scroll("Push A")
while True:
    if button_a.was_pressed(): # Start Logging when pressing Button A
        logging = True
        k = 0
        log.delete()
        log.set_labels('k', 'temp_c', 'temp_f')

    if logging:
        degC = tempTmp36()
        saveData(degC)

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

        k = k + 1
    else:
        display.scroll("Push A")

    sleep(5000)
```

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>
- MicroPython: <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>
- Micro:bit Datalogging: <https://microbit.org/get-started/user-guide/data-logging/>

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

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

