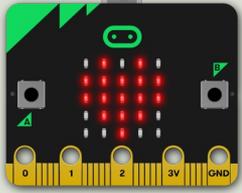


<https://www.halvorsen.blog>



# micro:bit and SPI using DAC MCP4911



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.
- TC74 Temperature Sensor with I2C Interface
- I2C and micro:bit
- TC74 and I2C Python Examples

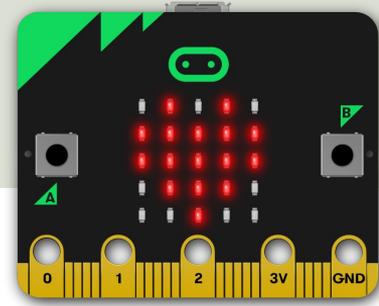


# Introduction to micro:bit

Hans-Petter Halvorsen

[Table of Contents](#)

# micro:bit

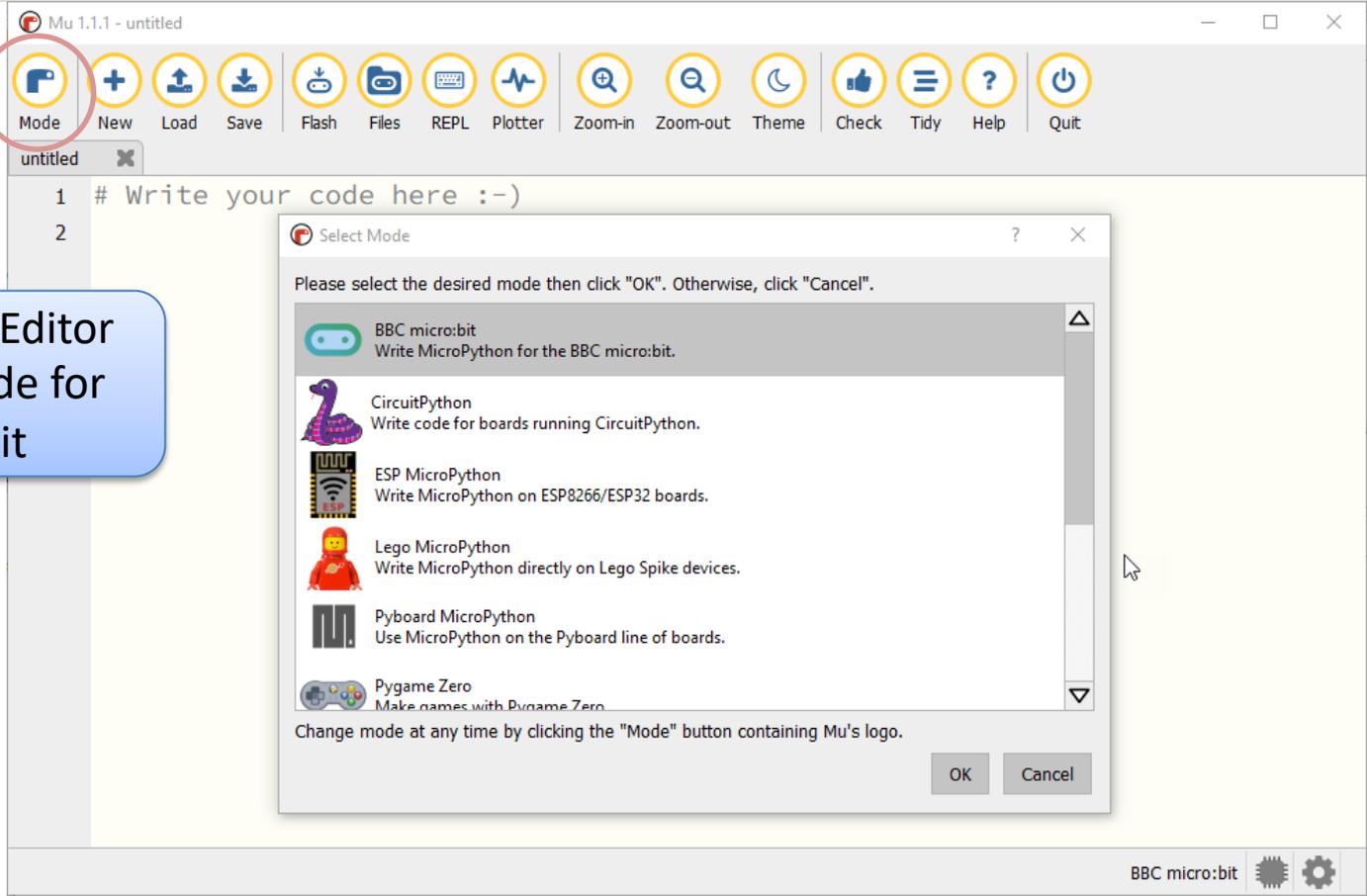


- micro:bit is a small microcontroller
- micro:bit is smaller than a credit card
- Price is about 150-400NOK (\$15-30)
- It can be used by kids and students to learn programming and technology
- micro:bit can run a special version of Python called MicroPython
- MicroPython is a down-scaled version of Python
- micro:bit Python User Guide  
<https://microbit.org/get-started/user-guide/python/>
- micro:bit MicroPython documentation  
<https://microbit-micropython.readthedocs.io>

# 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 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:

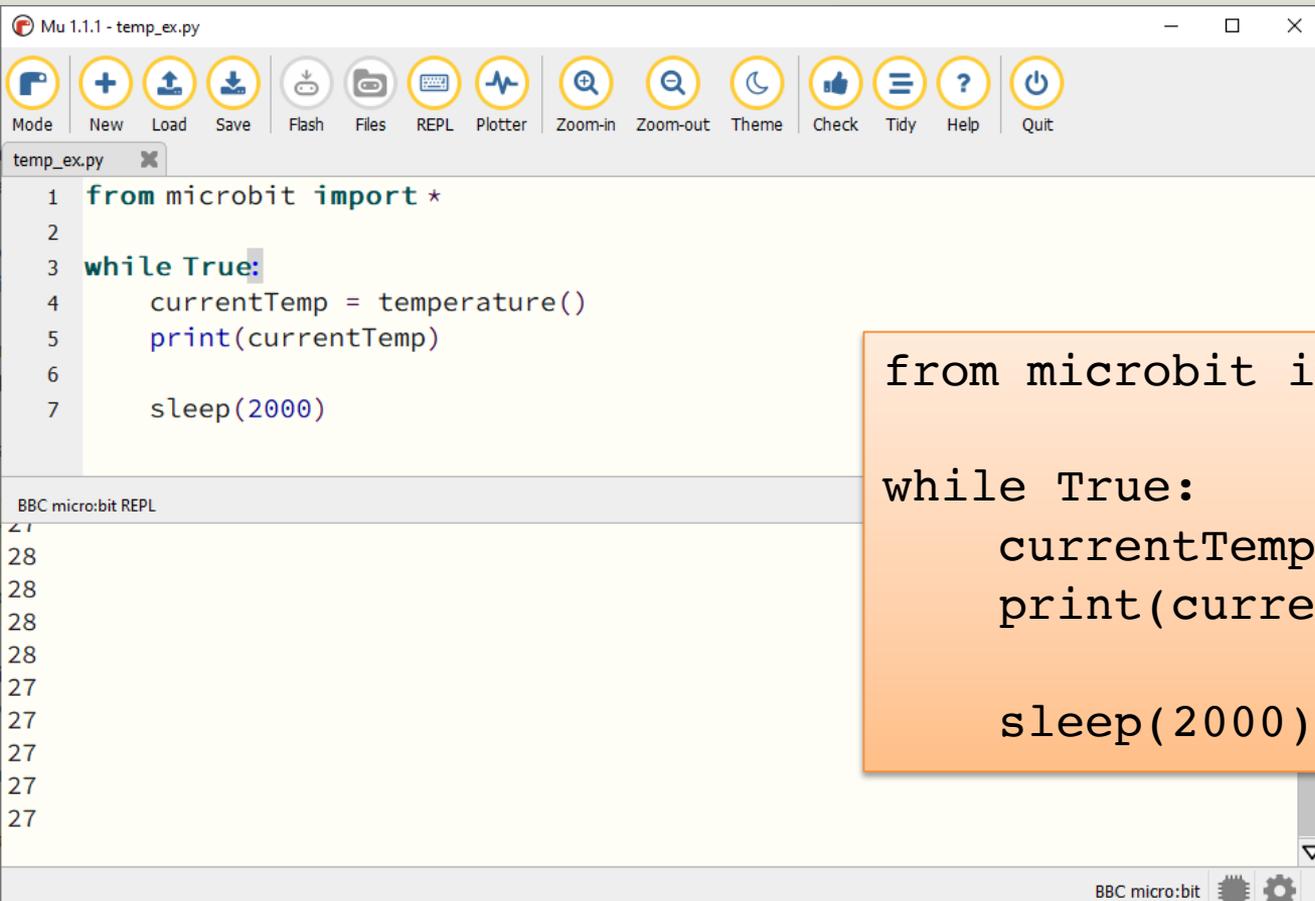
```
from microbit import *  
  
currentTemp = temperature()
```

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 that the program is running and printing the temperature value every 2000 milliseconds.

```
from microbit import *

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

    sleep(2000)
```



# 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.

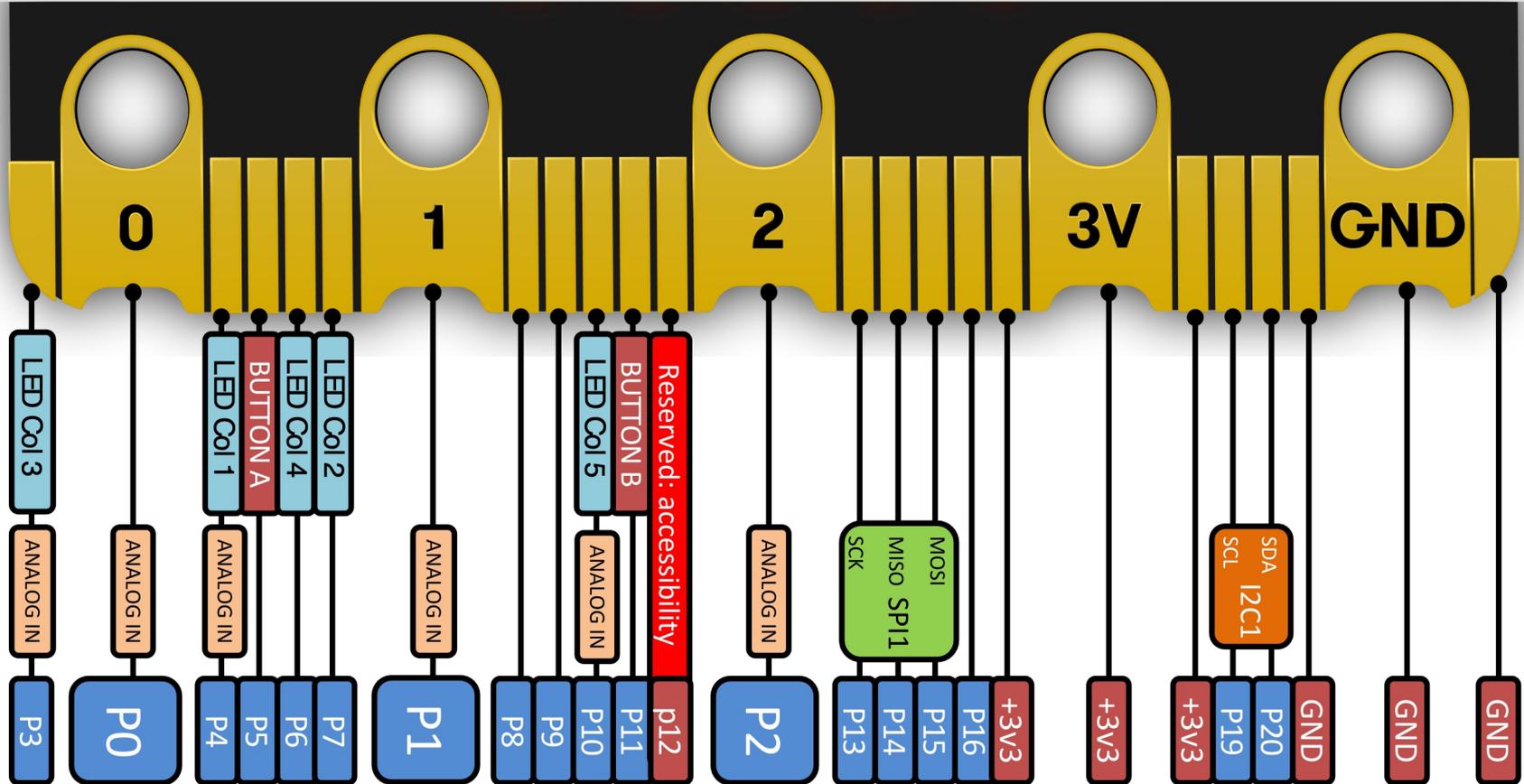
If you click A Button, the Minimum Temperature for the period (since you started the program/turned on the Micro:bit) is shown on the LED matrix

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

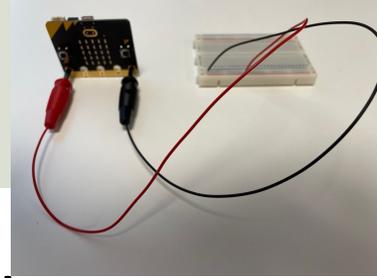


# micro:bit I/O Pins

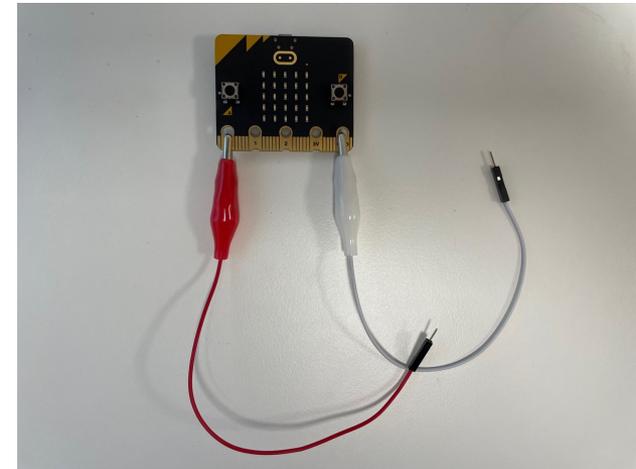
# micro:bit I/O Pin Overview



# 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



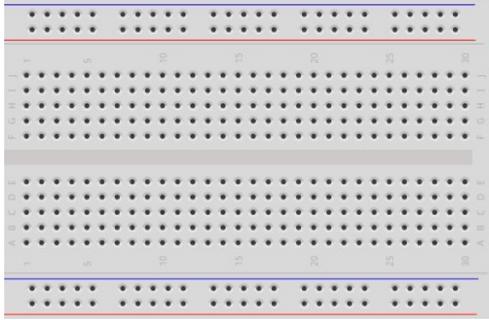
# Types of I/O Pins

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

<https://microbit-micropython.readthedocs.io/en/latest/pin.html>

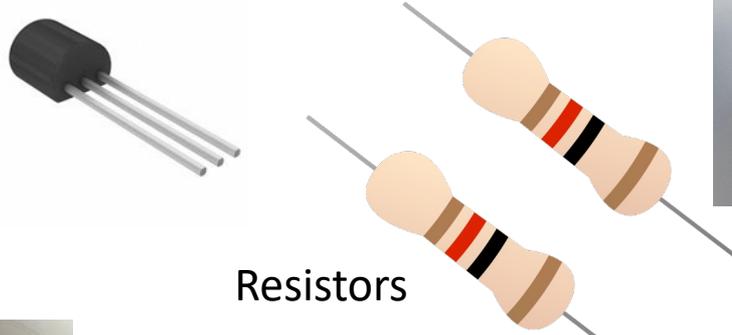
We will only use an Analog/Digital Input/Output pins in this Tutorial

# Component Examples

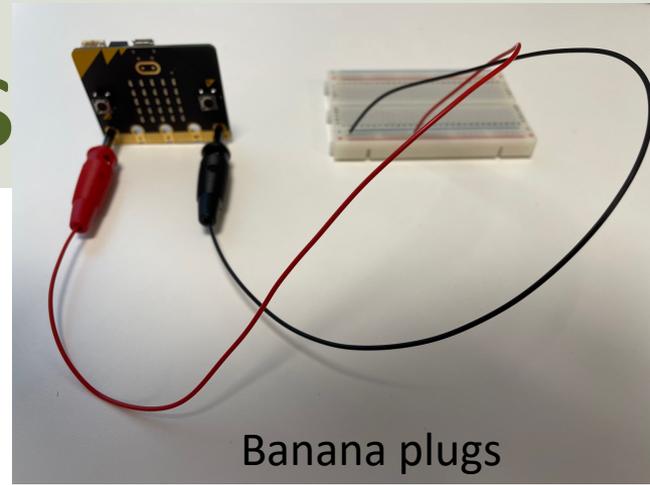


Breadboard

Temperature Sensor



Resistors



Banana plugs

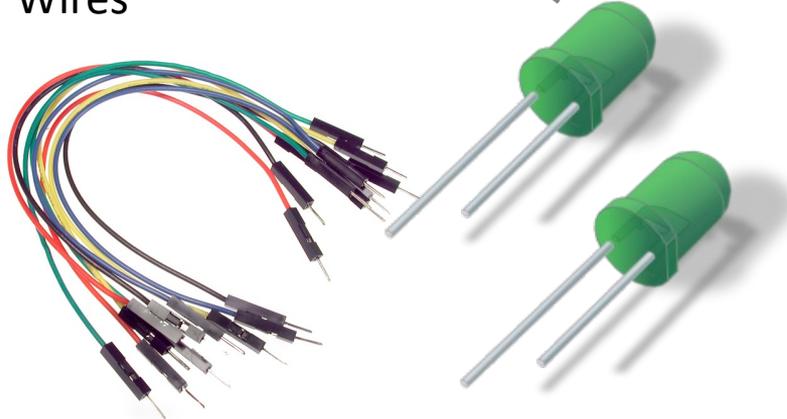
LEDs

Multimeter

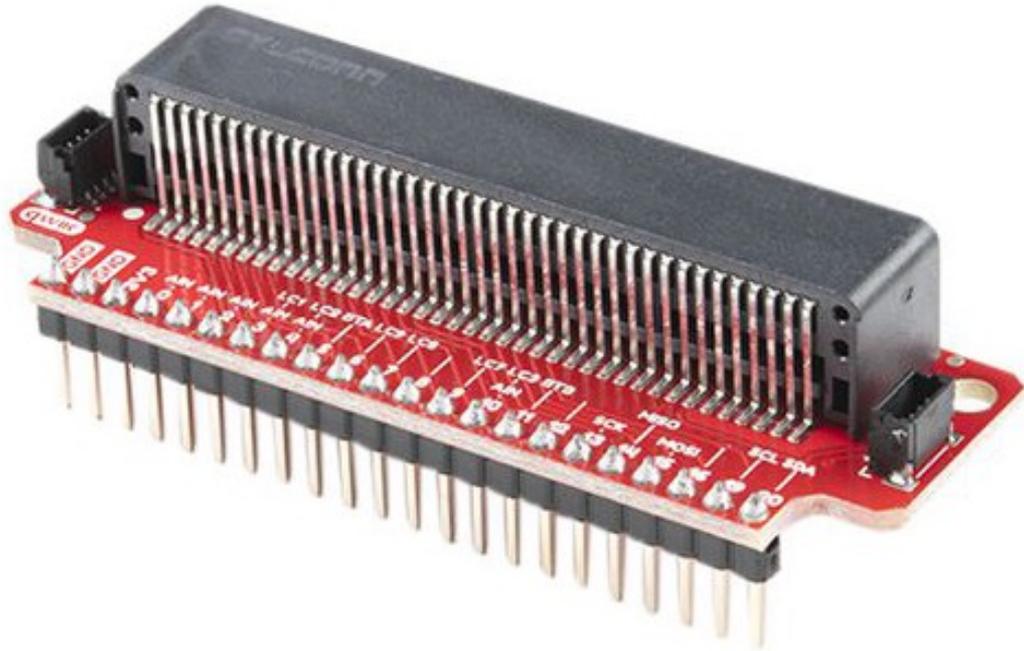


Alligator clips

Wires



# 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





# SPI

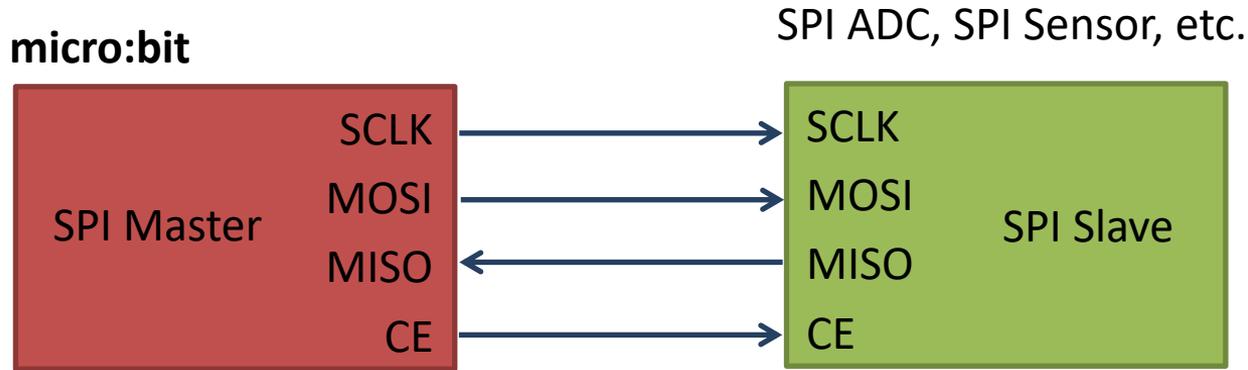
XXX

# SPI

- Serial Peripheral Interface (SPI)
- 4–Wire Protocol (SCLK, CE, MOSI, MISO)
- SPI is an interface to communicate with different types of electronic components like Sensors, Analog to Digital Converts (ADC), etc. that supports the SPI interface
- Thousands of different Components and Sensors supports the SPI interface

# SPI

SPI devices communicate in full duplex mode using a master-slave architecture with a single master



The SPI bus specifies four logic signals:

- **SCLK**: Serial **C**lock (output from master)
- **MOSI**: Master Out Slave In (**data output from master**)
- **MISO**: Master In Slave Out (**data output from slave**)
- **CE** (often also called SS - Slave Select): Chip Select (often active low, output from master)

# SPI with micro:bit

## Initialize I2C Communication:

```
i2c.init(freq=100000, sda=pin20, scl=pin19)
```

(No need to change anything here)

## Read Data from the connected I2C device:

```
i2c.read(addr, n, repeat=False)
```

Read n bytes from the device with 7-bit address addr.

If repeat is True, no stop bit will be sent.

<https://microbit-micropython.readthedocs.io/en/v2-docs/spi.html>



# DAC MCP4911

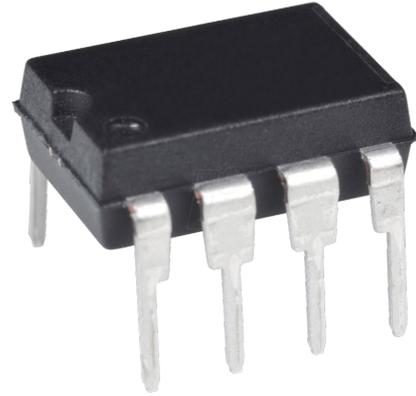
# DAC – MCP4911

- DAC – Digital to Analog Converter
- Arduino UNO has no real Analog Out Channel – only Digital PWM channels
- We can use an external DAC in order to provide a real Analog Out
- MCP4911 is a single channel, 10-bit DAC with an external voltage reference and SPI interface

# MCP49xx

MCP49xx is a family of DAC ICs:

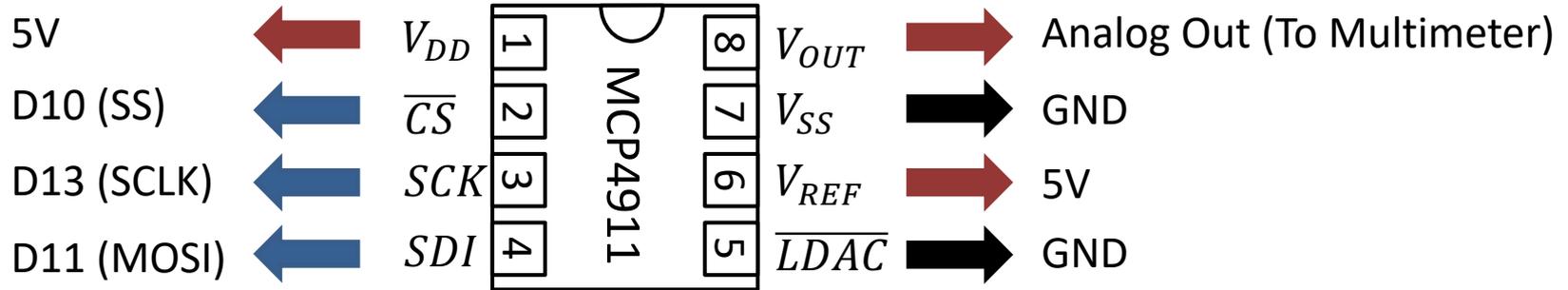
- MCP4901: 8-Bit Voltage Output DAC
- **MCP4911: 10-Bit Voltage Output DAC**
- MCP4921: 12-Bit Voltage Output DAC

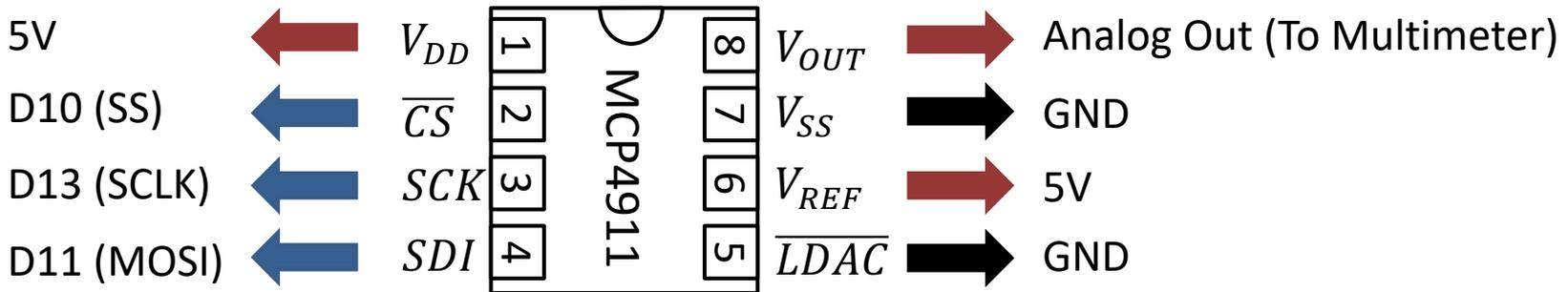
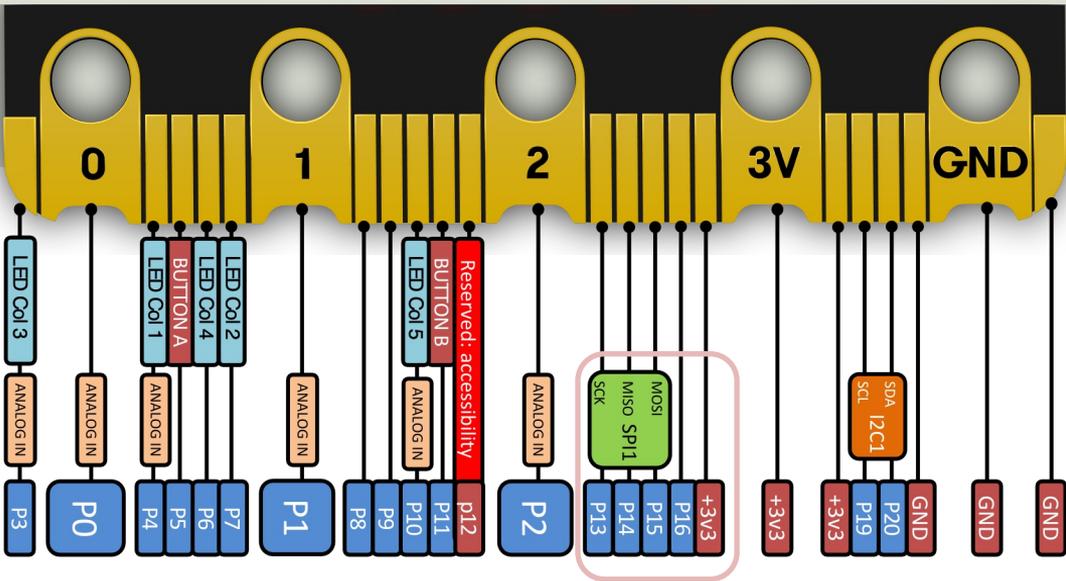


The different MCP49xx DACs work in the same manner, the only difference is the resolution (8, 10, or 12 resolution)

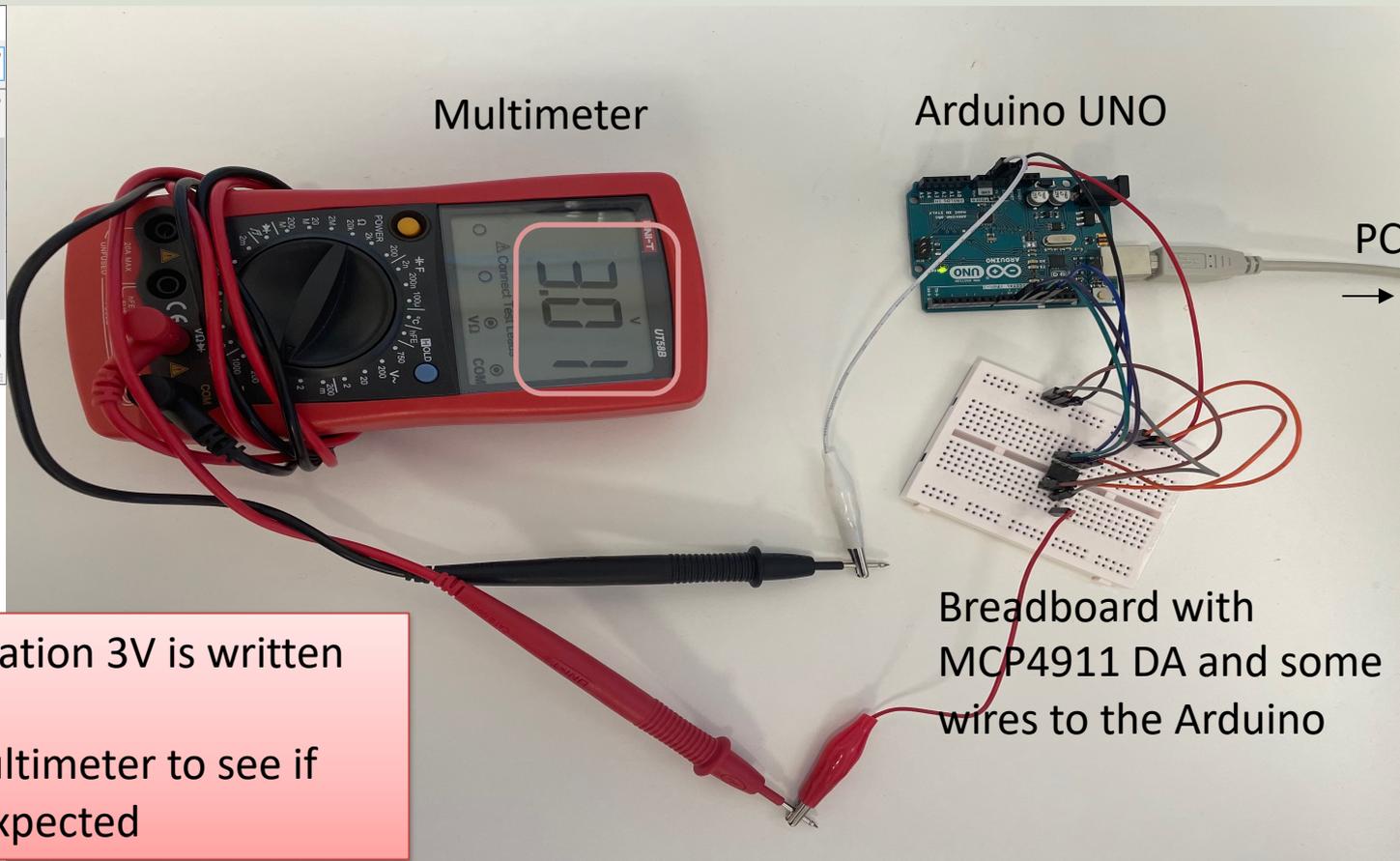
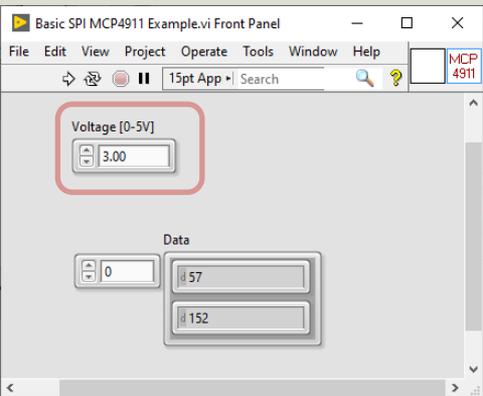
Datasheet: <https://www.microchip.com/en-us/product/MCP4911>

# MCP4911 - Wiring





# Test Setup



In the LabVIEW Application 3V is written to the MCP4911 DAC.  
Then we can use a Multimeter to see if everything works as expected

XXX



# Python Examples

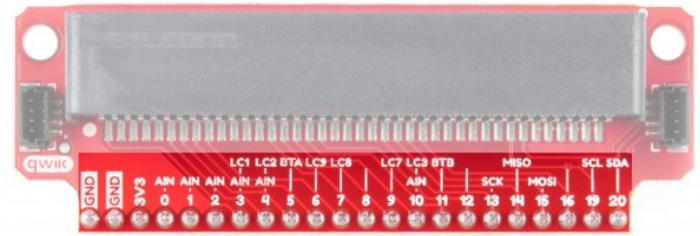
## MCP4911

# Breakout Board

For easy wiring using I2C, a Breakout board is recommended. Many different types do exist. In this tutorial “Sparkfun Microbit Breakout“ board will be used.

Sparkfun Microbit Breakout

<https://learn.sparkfun.com/tutorials/microbit-breakout-board-hookup-guide>



Pin	Function 1	Function 2	Description
GND			Ground
GND			Ground
3V3			3.3V
0	Analog In		Connected to large pin 0
1	Analog In		Connected to large pin 1
2	Analog In		Connected to large pin 2
3	Analog In	LED Column 1	Controls part of LED array
4	Analog In	LED Column 2	Controls part of LED array
5		Button A	Connected to Button A on micro:bit
6		LED Column 9	Controls part of LED array
7		LED Column 8	Controls part of LED array
8			Open GPIO pin
9		LED Column 7	Controls part of LED array
10	Analog In	LED Column 3	Controls part of LED array
11		Button B	Connected to Button B on micro:bit
12			Open GPIO pin
13	SCK		GPIO or SPI clock
14	MISO		GPIO or SPI MISO
15	MOSI		GPIO or SPI MOSI
16			Open GPIO pin
19	SCL		GPIO or I <sup>2</sup> clock
20	SDA		GPIO or I <sup>2</sup> data

# Python

```
from microbit import *

i2c.init(freq=100000, sda=pin20, scl=pin19)

address = 0x48

data = i2c.read(address, 1, repeat=False)
print(data) # Data received is a byte object

# Converting to int. Resolution for TC74 Sensor is +/-1°C
# byteorder is big where MSB is at start
temp = int.from_bytes(data, "big")
print(temp)
display.scroll(temp)
```

```
from microbit import *

i2c.init(freq=100000, sda=pin20, scl=pin19)

address = 0x48

while True:
    data = i2c.read(address, 1, repeat=False)
    # print(data) # Data received is a byte object

    # Converting to int. Resolution for TC74 Sensor is +/-1°C
    # byteorder is big where MSB is at start
    temp = int.from_bytes(data, "big")
    print(temp)
    display.scroll(temp)

    sleep(5000)
```



Mode



New



Load



Save



Flash



Files



REPL



Plotter



Zoom-in



Zoom-out



Theme



Check



Tidy



Help



Quit

tmp36\_led.py

tc74\_ex.py

tc74\_ex2.py

```
1 from microbit import *
2
3 i2c.init(freq=100000, sda=pin20, scl=pin19)
4
5 address = 0x48
6
7 while True:
8     data = i2c.read(address, 1, repeat=False)
9     # print(data) # Data received is a byte object
10
11     # Converting to int. Resolution for TC74 Sensor is +/-1°C
12     # bytearray is big where MSB is at start
13     temp = int.from_bytes(data, "big")
14     print(temp)
15     display.scroll(temp)
16
17     sleep(5000)
```

BBC micro:bit REPL

MicroPython v1.15-64-g1e2f0d280 on 2021-06-30; micro:bit v2.0.0 with nRF52833

Type "help()" for more information.

>>>

>>> 28

30

32

33

32

32



# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

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

