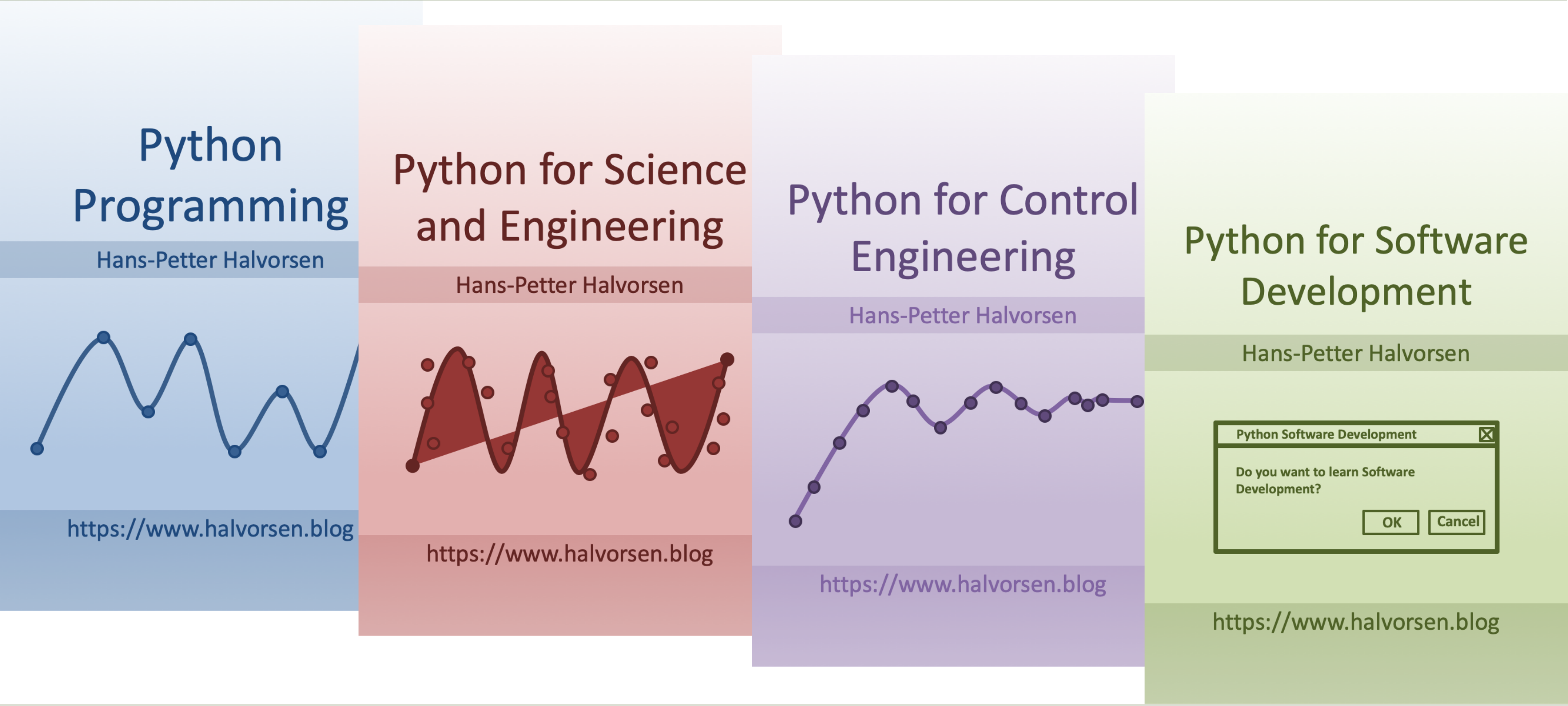
micro:bit and Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

https://www.halvorsen.blog/documents/programming/python/

Additional Python Resources

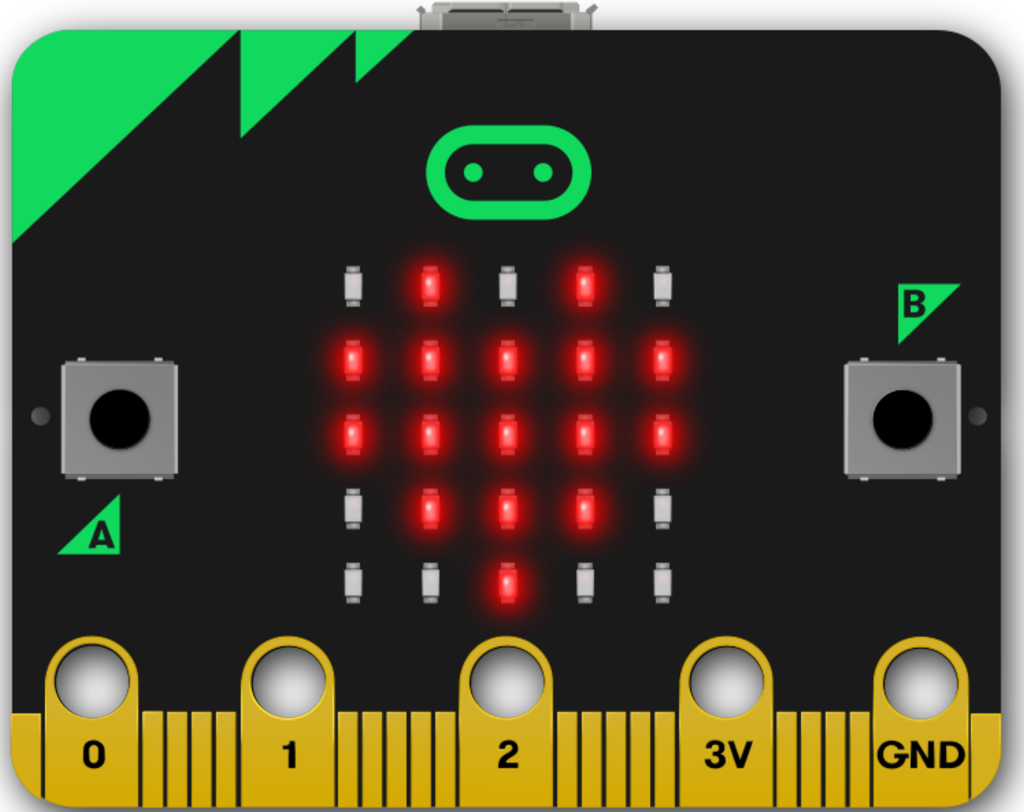


https://www.halvorsen.blog/documents/programming/python/

Contents

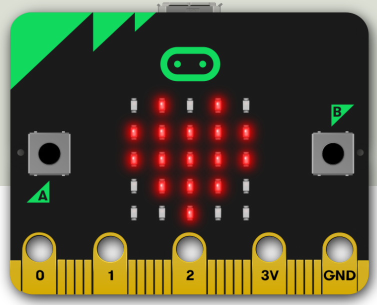
- Introduction to micro:bit
- Python, MicroPython
- Programming the micro:bit using Python

micro:bit



https://microbit.org

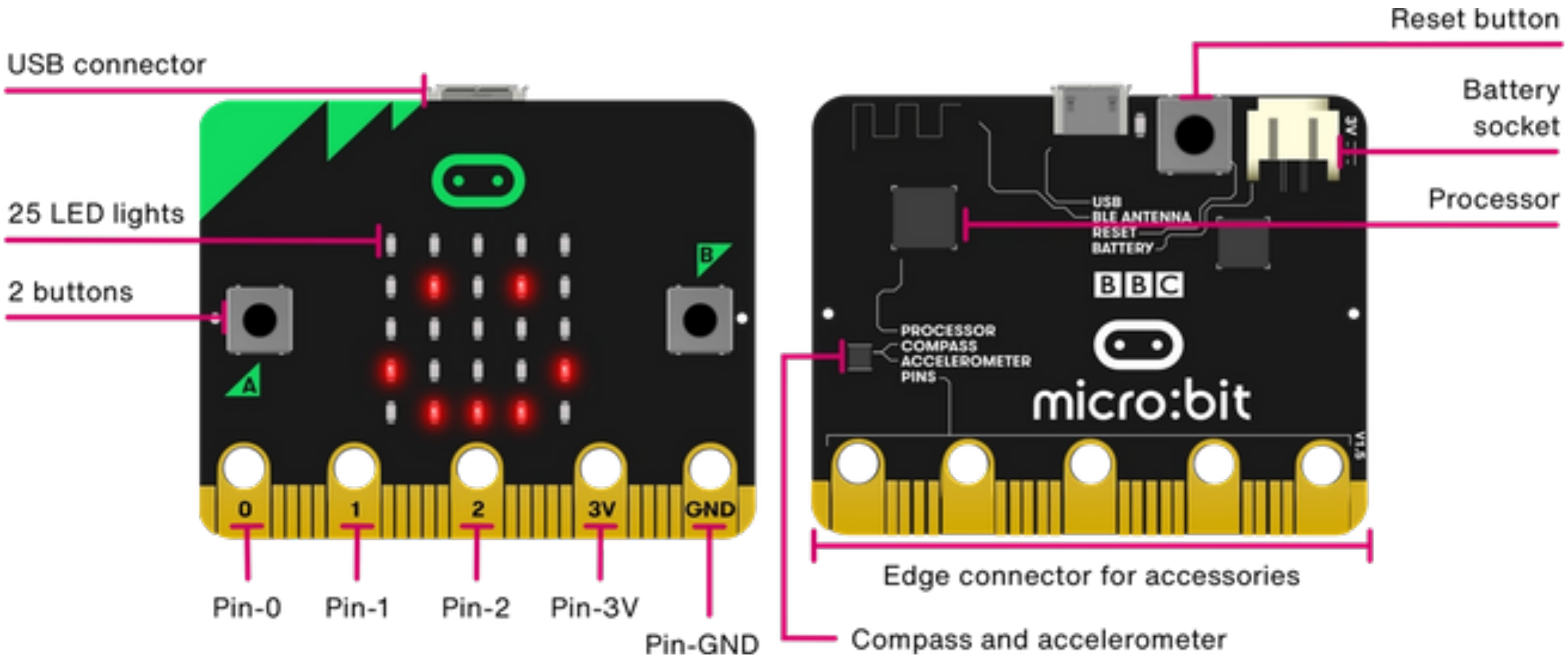
micro:bit



- micro:bit is a small microcontroller
- micro:bit runs a special version of MicroPython
- MicroPython is a down-scaled version of Python
- micro:bit is smaller than a credit card
- Price is about 300-400NOK (15-$30)

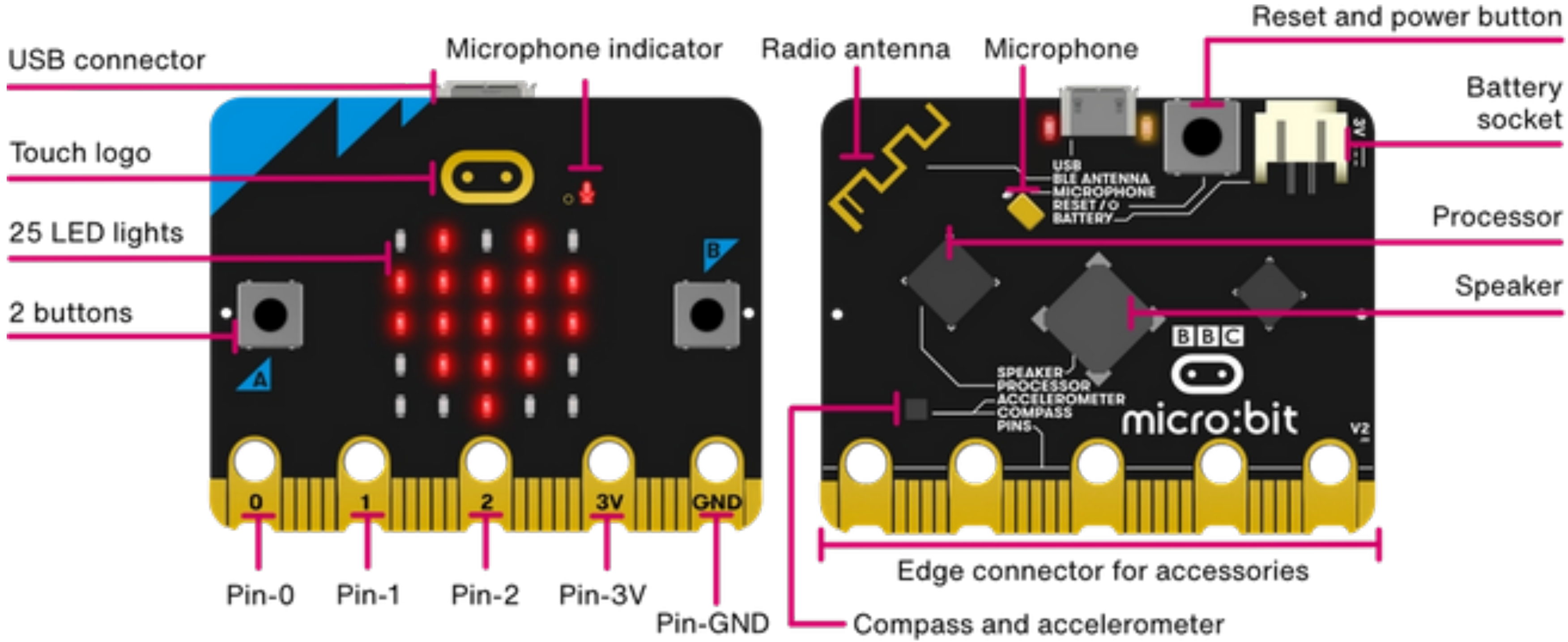
https://microbit.org

Original micro:bit



https://microbit.org

New micro:bit (micro:bit v2)



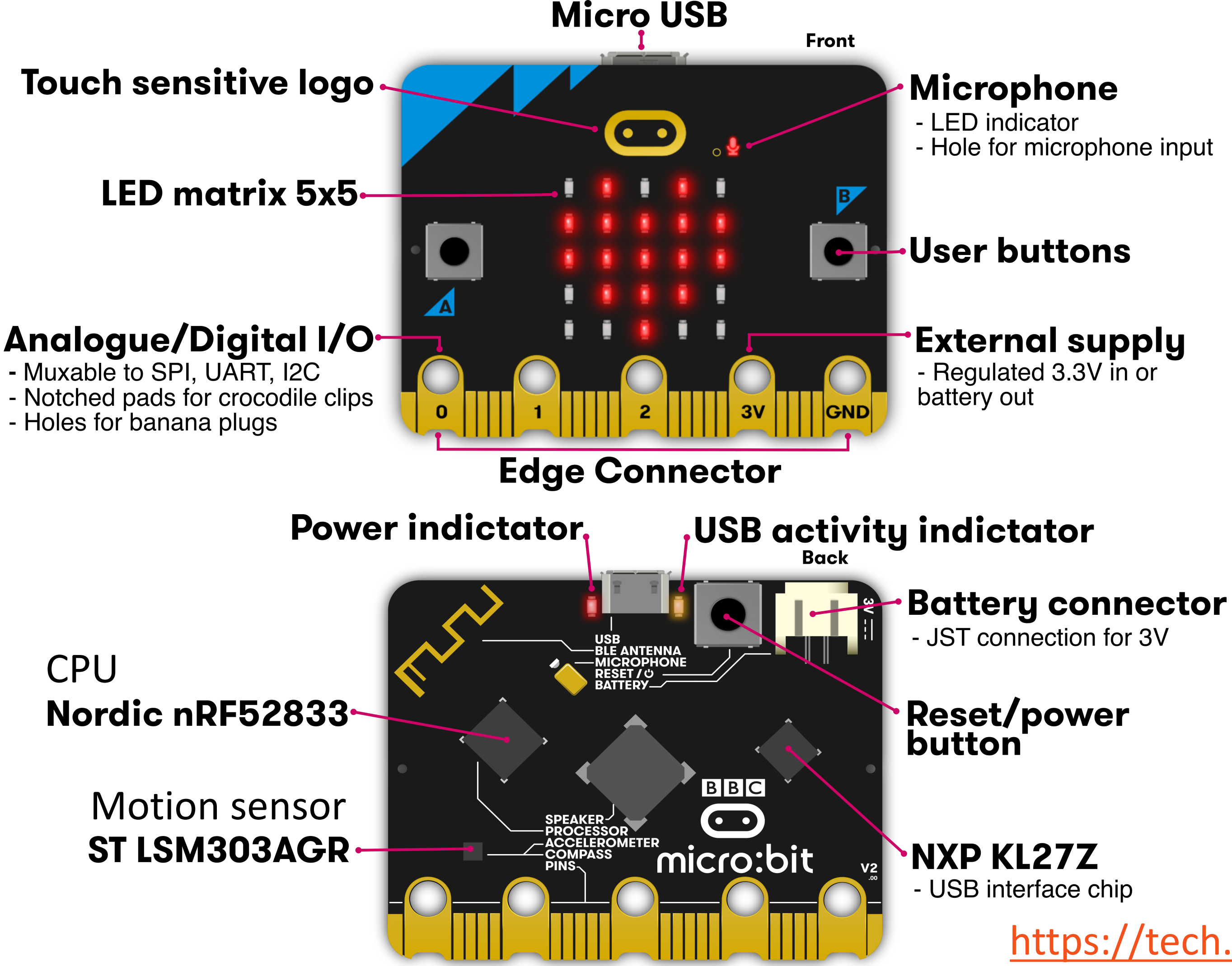
https://youtu.be/pIUJ4kvJ_QU

https://microbit.org

micro:bit Features

- USB Communication and Powered by micro-USB or JST Battery Connection
- 6 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)

https://microbit.org

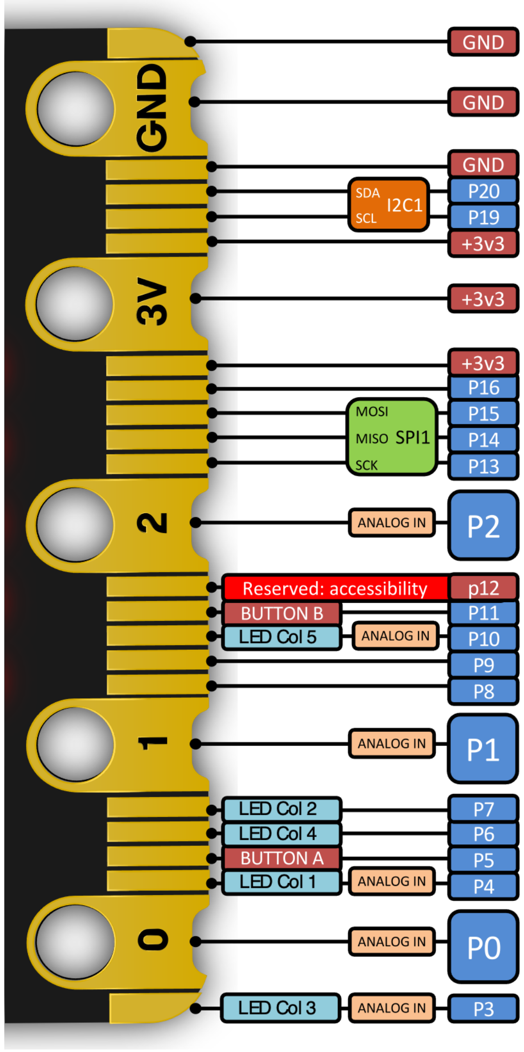


https://tech.microbit.org/hardware/

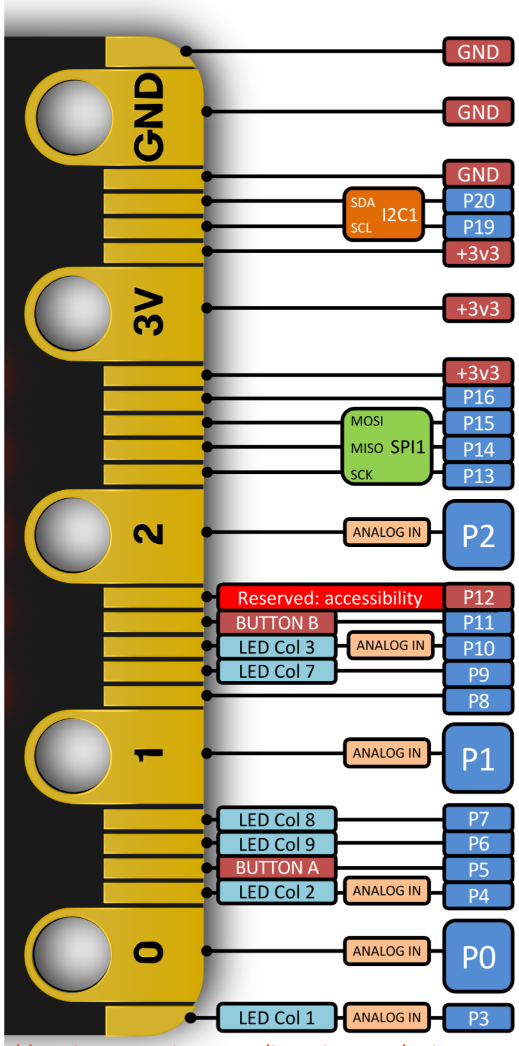
I/O Pin Overview

https://microbit.pinout.xyz/

New micro:bit (micro:bit v2)



Original micro:bit



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

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 great!
- micro:bit runs a special version of MicroPython
- MicroPython is a down-scaled version of Python
- You can use different Python Editors; the Mu Python Editor is recommended

MicroPython

- MicroPython is a lean 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://micropython.org

micro:bit Python Editors

- The combination of the micro:bit Hardware and the Python Programming Language is great!
- Online Editor (used in your Browser) https://python.microbit.org
- Mu Python Editor https://codewith.mu

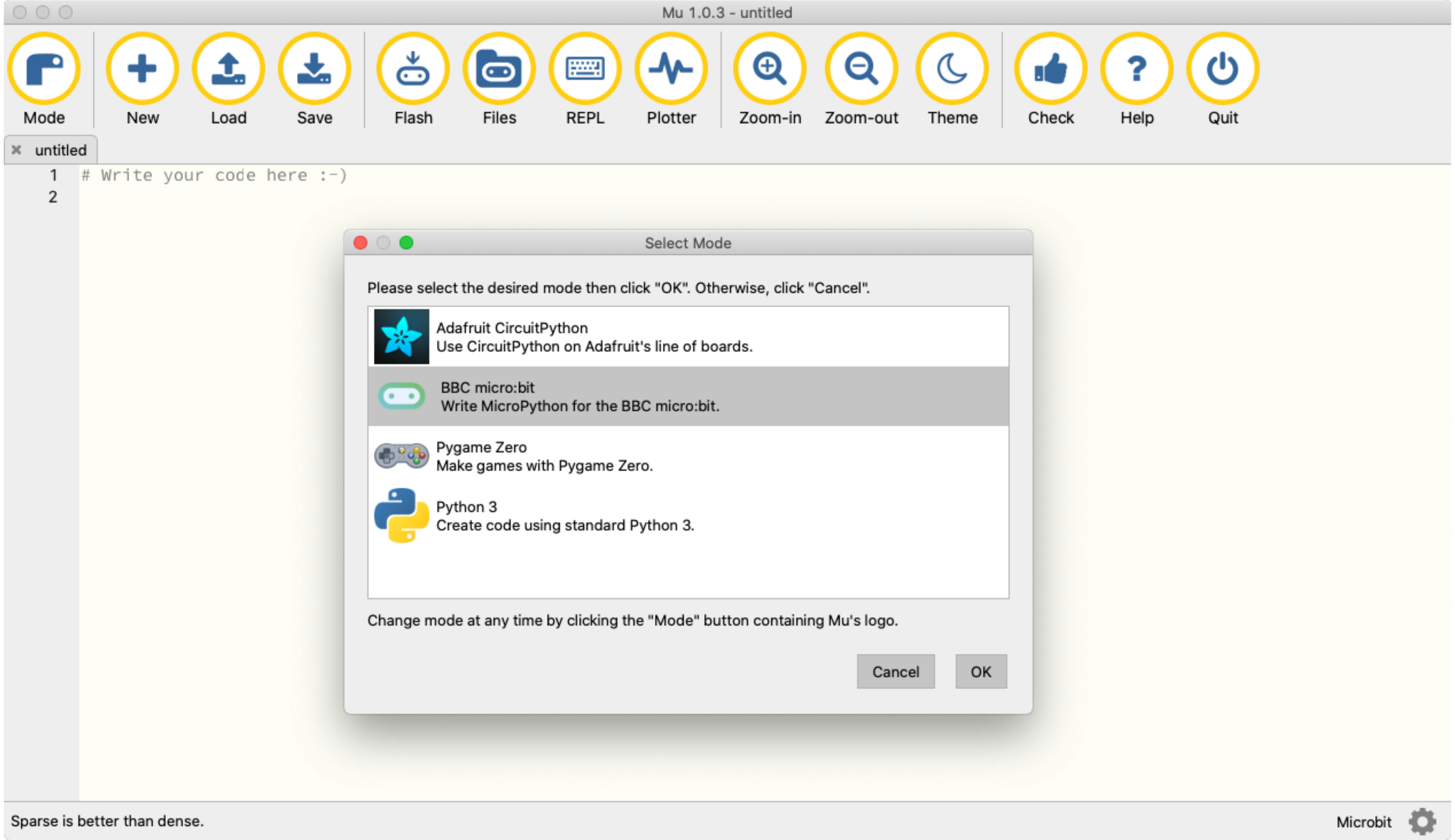
micro:bit Python Documentation

- micro:bit Python User Guide
  https://microbit.org/get-started/user-guide/python/
- micro:bit MicroPython documentation
  https://microbit-micropython.readthedocs.io

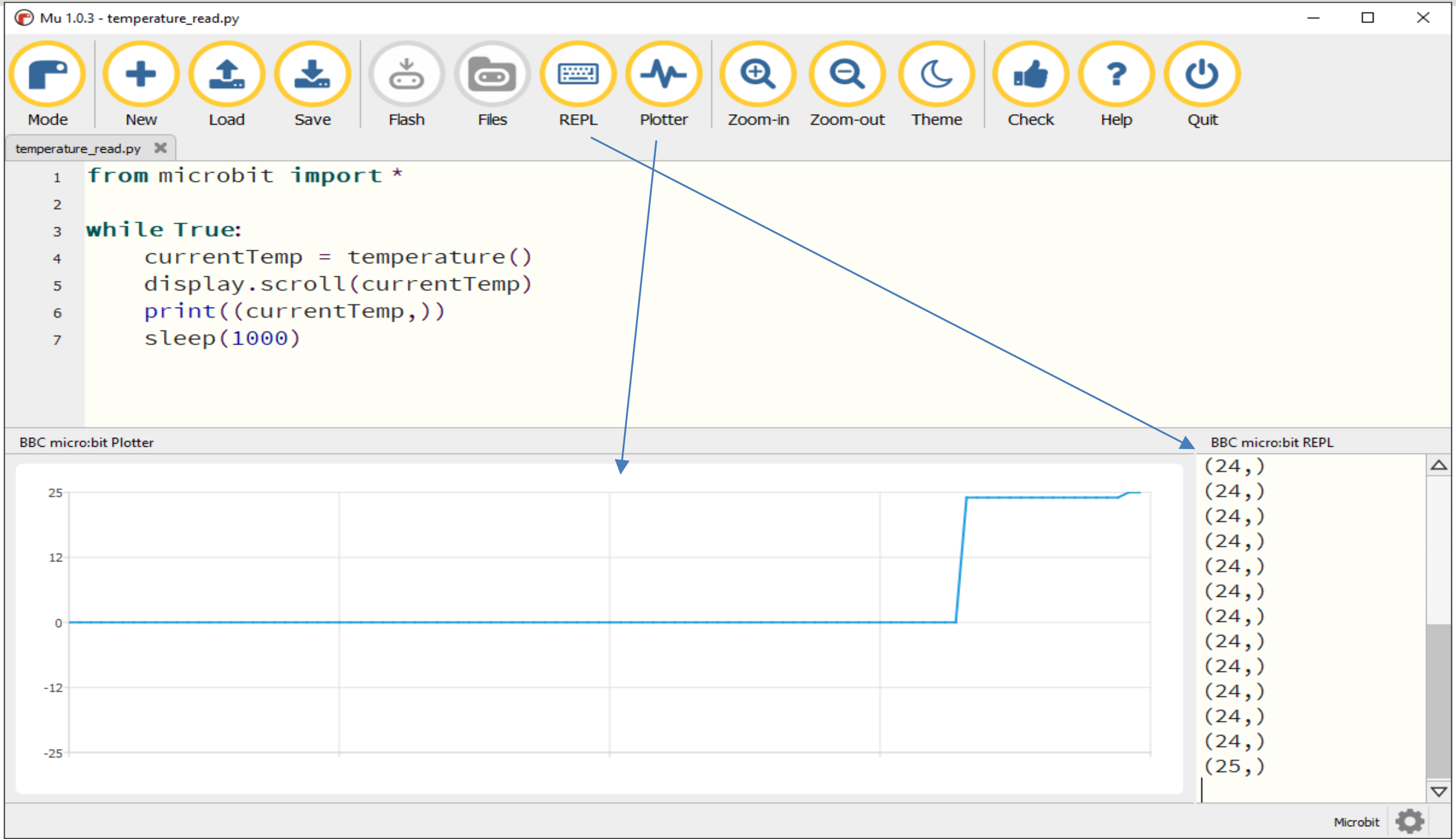
Mu Python Editor

- Mu is a Python code editor for beginner programmers
- It is among others 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



Mu Python Editor



micro:bit Interfaces

Hans-Petter Halvorsen

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)
- Analog/Digital Input/Output Pins

LED Matrix (5x5)

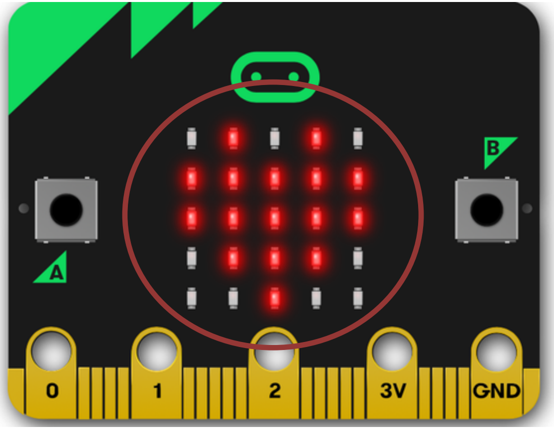
- An LED, or light-emitting diode is an output device that gives off light.
- The Micro:bit has a display of 25 (5x5) LEDs for you to program.
- You can use the LED matrix to show images or show text or numbers

LED Matrix - 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.

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"

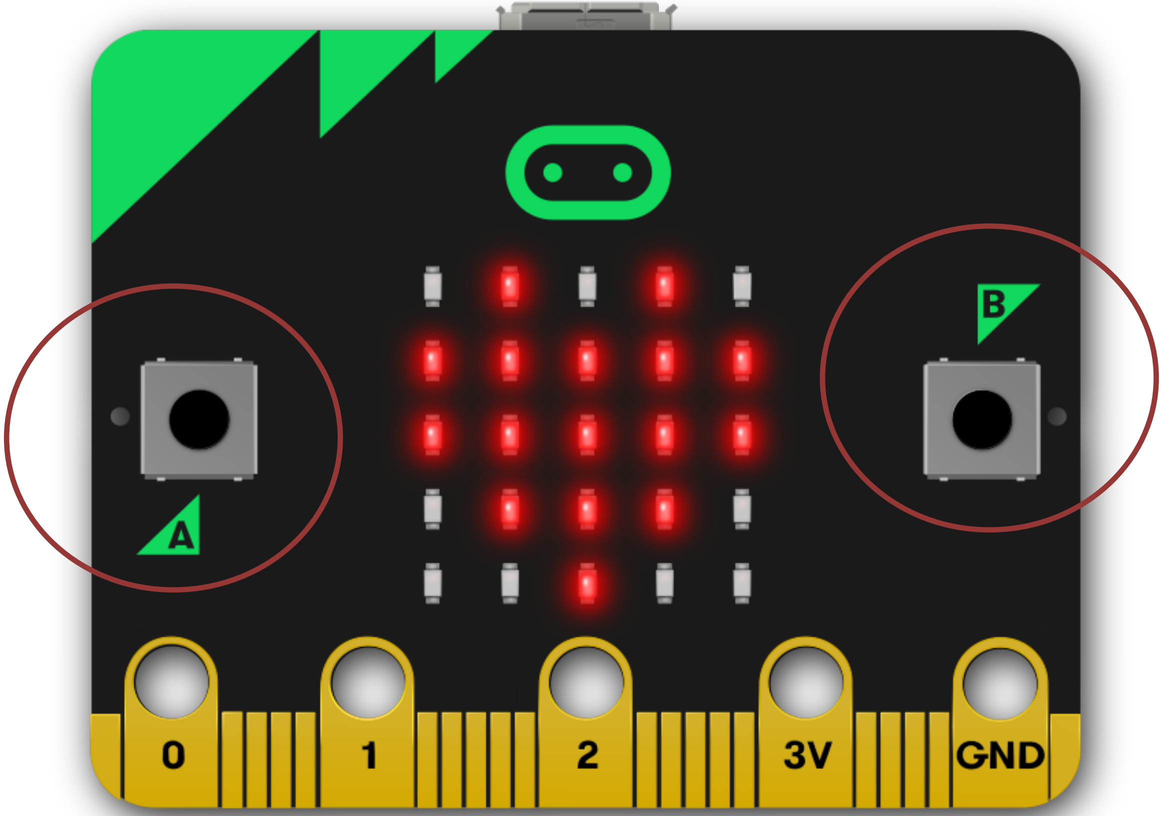
```
from microbit import *

display.scroll("WELCOME")
```

The word "WELCOME" will scroll over the LED matrix

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

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

- 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()
```

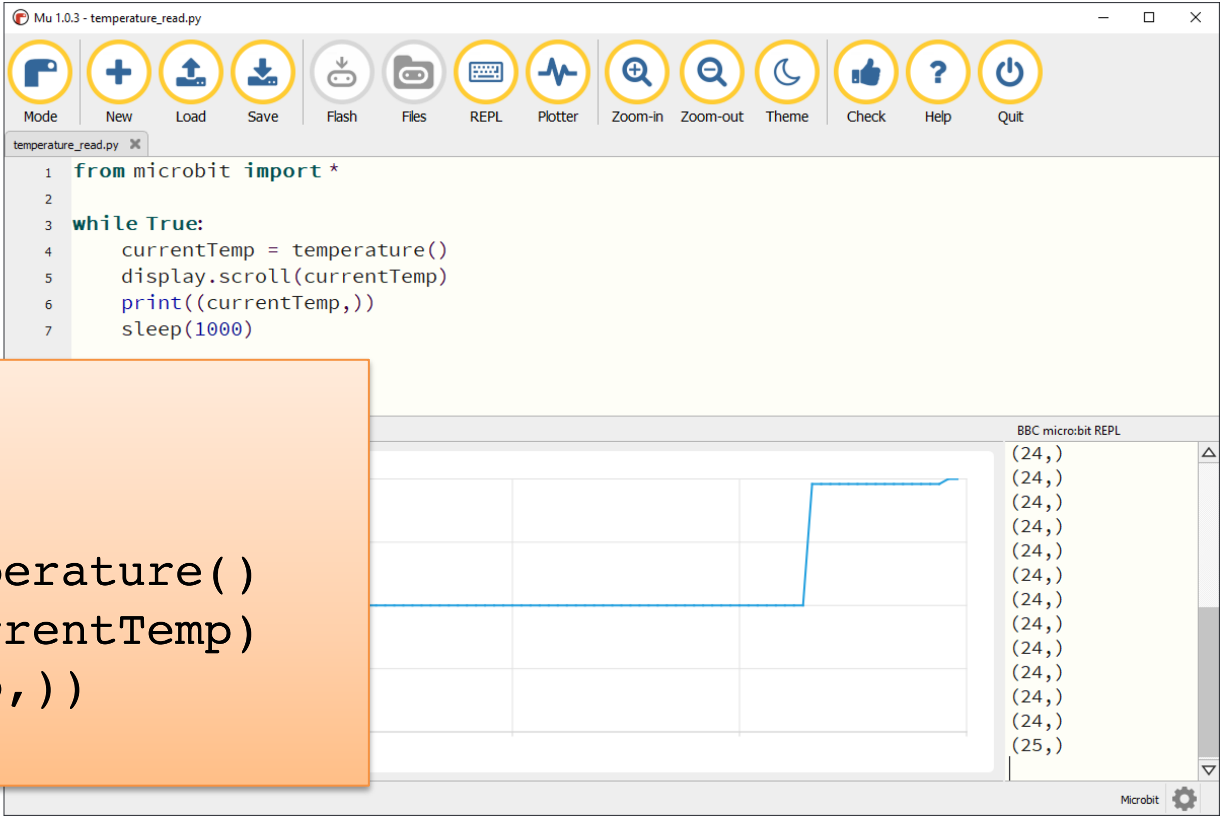
This examples displays the temperature on the LED matrix:

```
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




```
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

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

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

```
from microbit import *
import random

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

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))
```

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 have to calibrate it – a little game appears on the screen where you have to 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

Examples using the I/O Pins

Examples

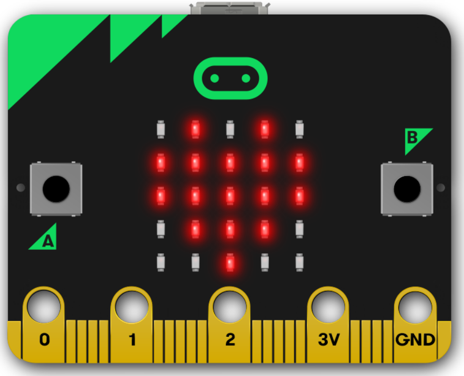
Let's connect some external components to the micro:bit GPIO pins. Examples:

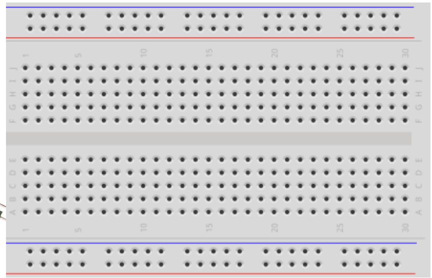
- LEDs
- TMP36 Temperature Sensor
- ..

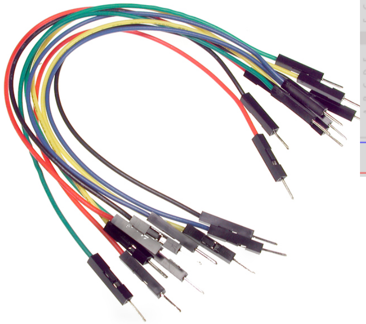
LEDs

Necessary Equipment

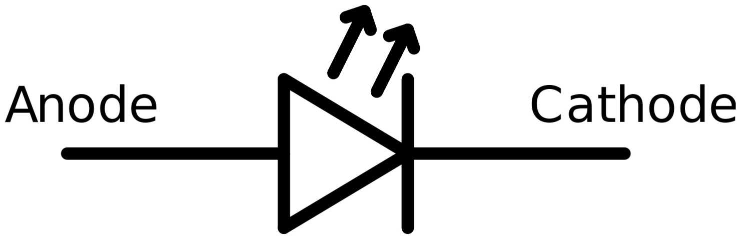
- micro:bit
- Breadboard
- LED
- Resistor, $R = 270\Omega$
- Wires (Jumper Wires)

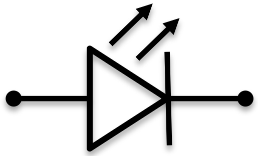


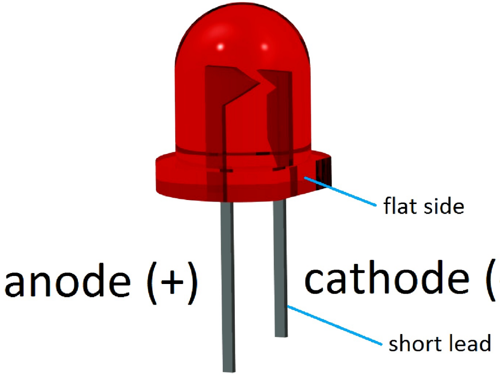


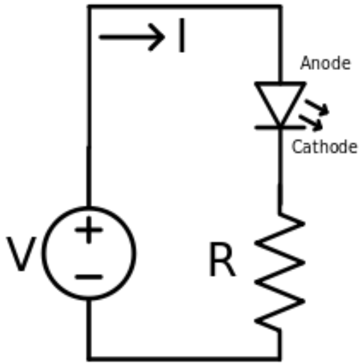


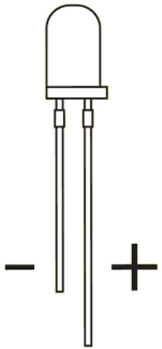
LED







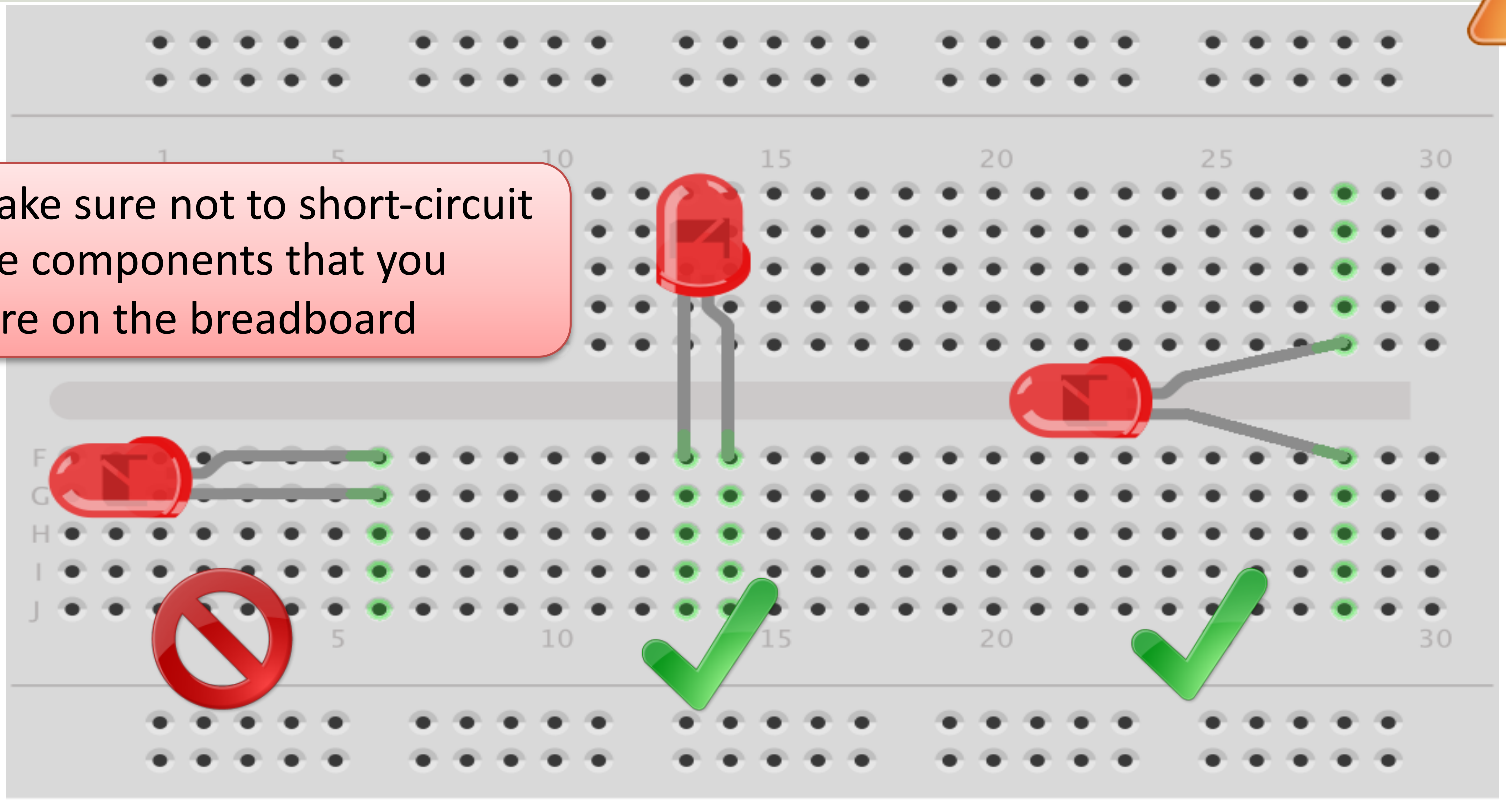




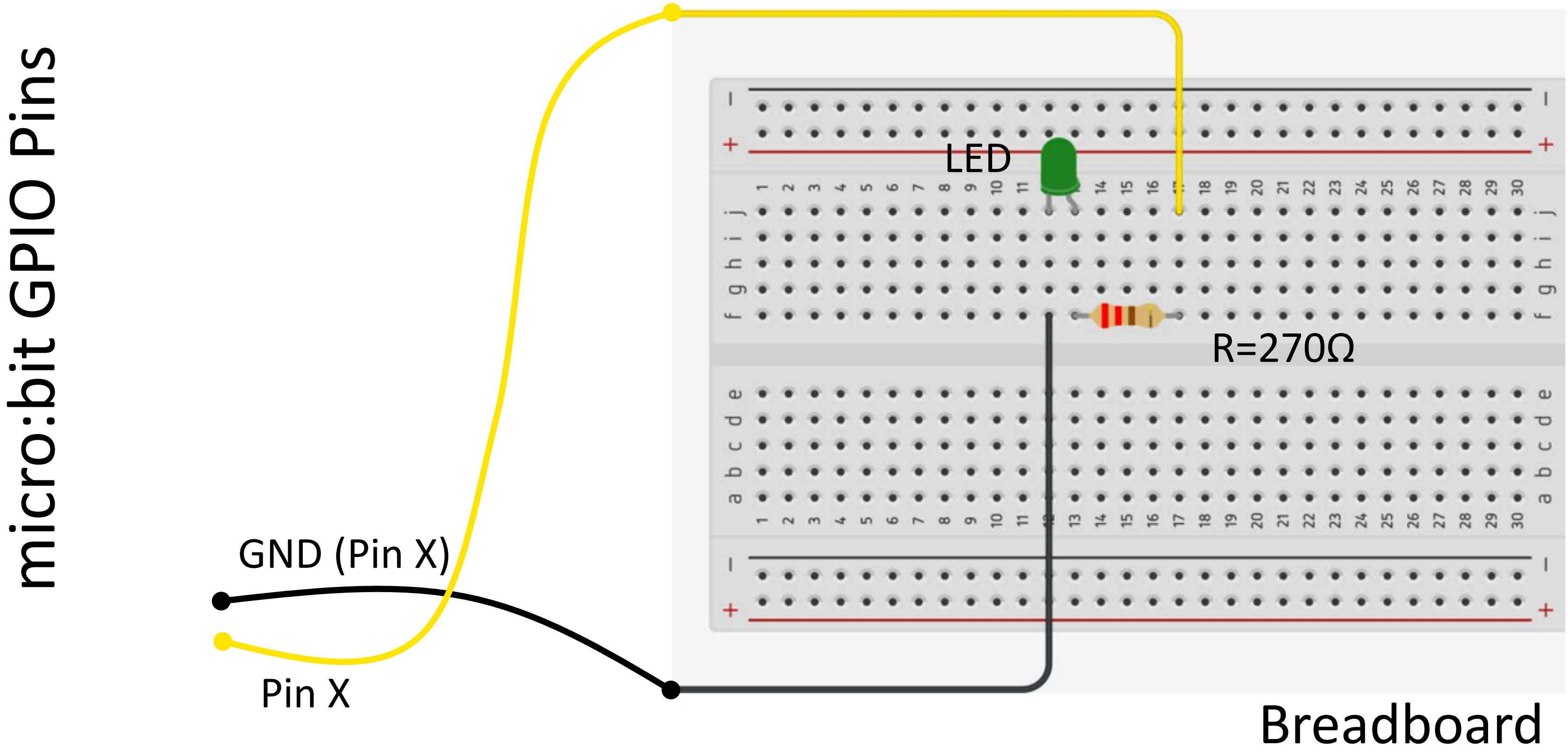
[Wikipedia]

Breadboard Wiring

Make sure not to short-circuit the components that you wire on the breadboard



LED Example



Why do you need a Resistor?

If the current becomes too large, the LED will be destroyed. To prevent this to happen, we will use a Resistor to limit the amount of current in the circuit.

What should be the size of the Resistor?

A LED typically need a current like 20mA (can be found in the LED Datasheet).
We use Ohm's Law:

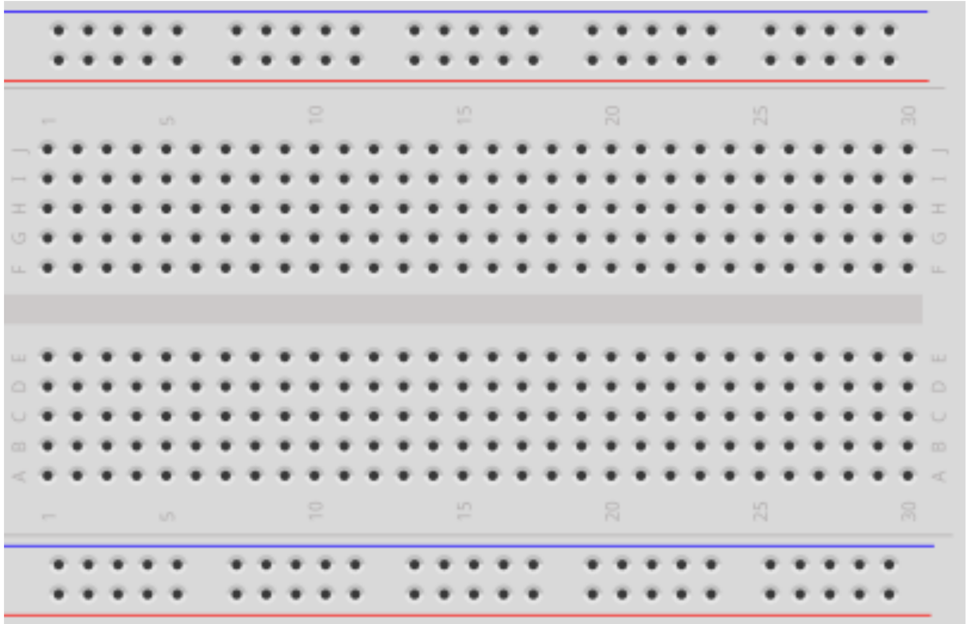
$$U = RI$$

Arduino gives U=5V and I=20mA. We then get:

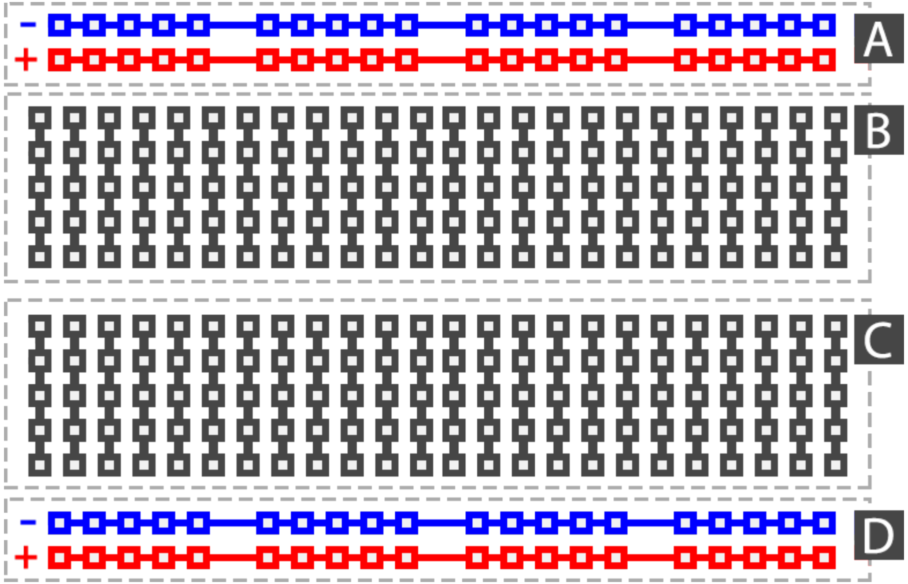
$$R = \frac{U}{I}$$

The Resistor needed will be $R = \frac{5V}{0.02A} = 250\Omega$. Resistors with R=250Ω is not so common, so we can use the closest Resistors we have, e.g., 270Ω

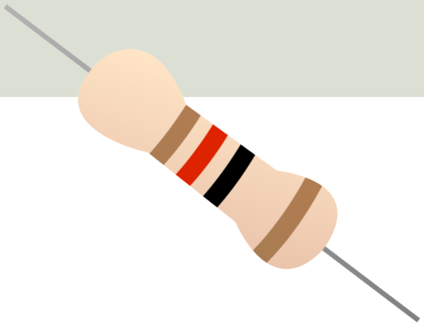
Breadboard



A breadboard is used to wire electric components together

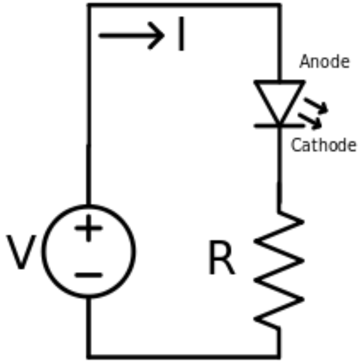


Resistors



Resistance is measured in Ohm (Ω)

Resistors comes in many sizes, e.g., 220Ω , 270Ω, 330Ω, 1kΩm 10kΩ, ...



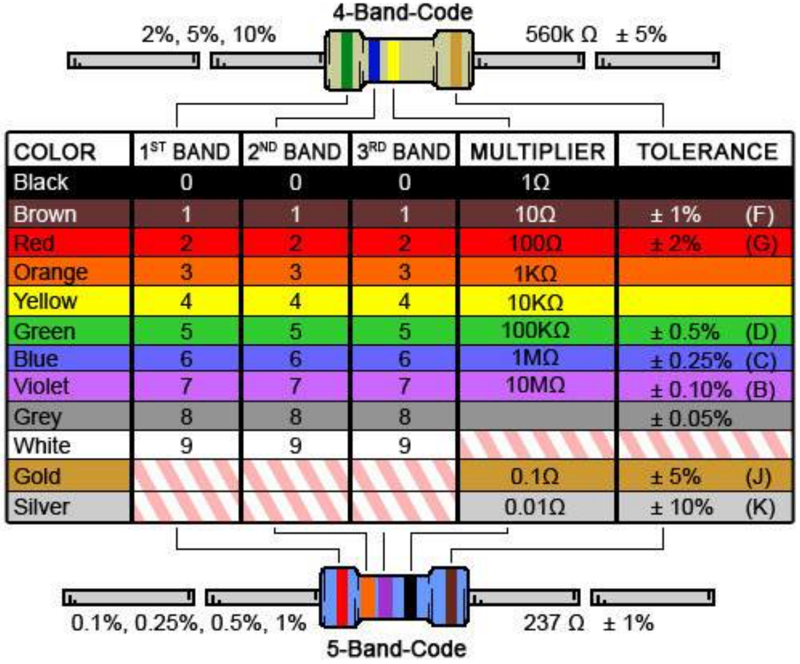
The resistance can be found using **Ohms Law**

$$U = RI$$

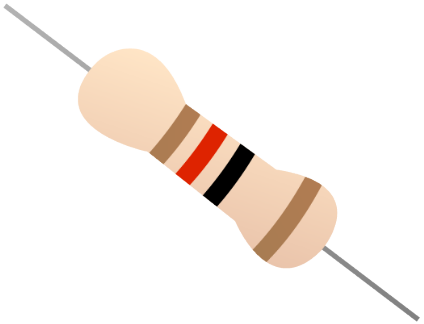
https://en.wikipedia.org/wiki/Resistor

Electrical symbol: 

Resistor Colors



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)



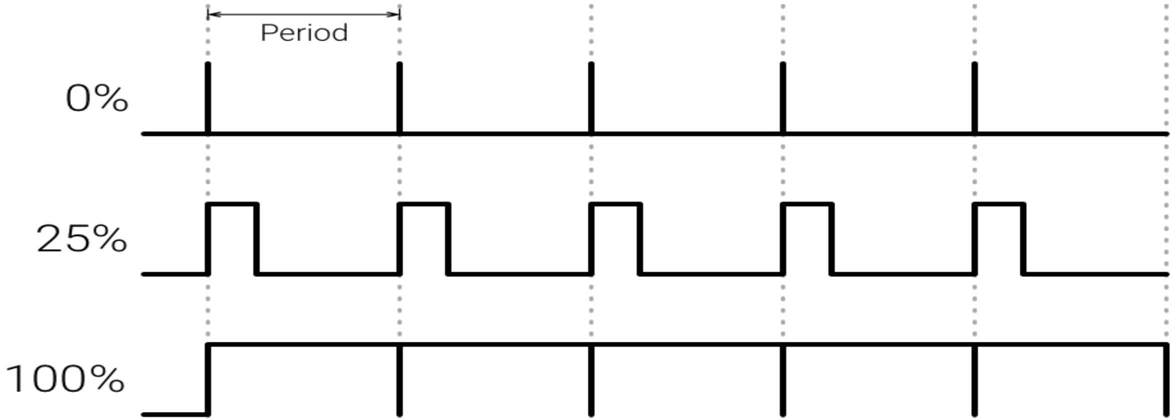
You can also use a **Multimeter**

Resistor Calculator: http://www.allaboutcircuits.com/tools/resistor-color-code-calculator/

PWM

PWM is a digital (i.e., square wave) signal that oscillates according to a given *frequency* and *duty cycle*.
The frequency (expressed in Hz) describes how often the output pulse repeats.
The period is the time each cycle takes and is the inverse of frequency.
The duty cycle (expressed as a percentage) describes the width of the pulse within that frequency window.

You can adjust the duty cycle to increase or decrease the average "on" time of the signal. The following diagram shows pulse trains at 0%, 25%, and 100% duty:



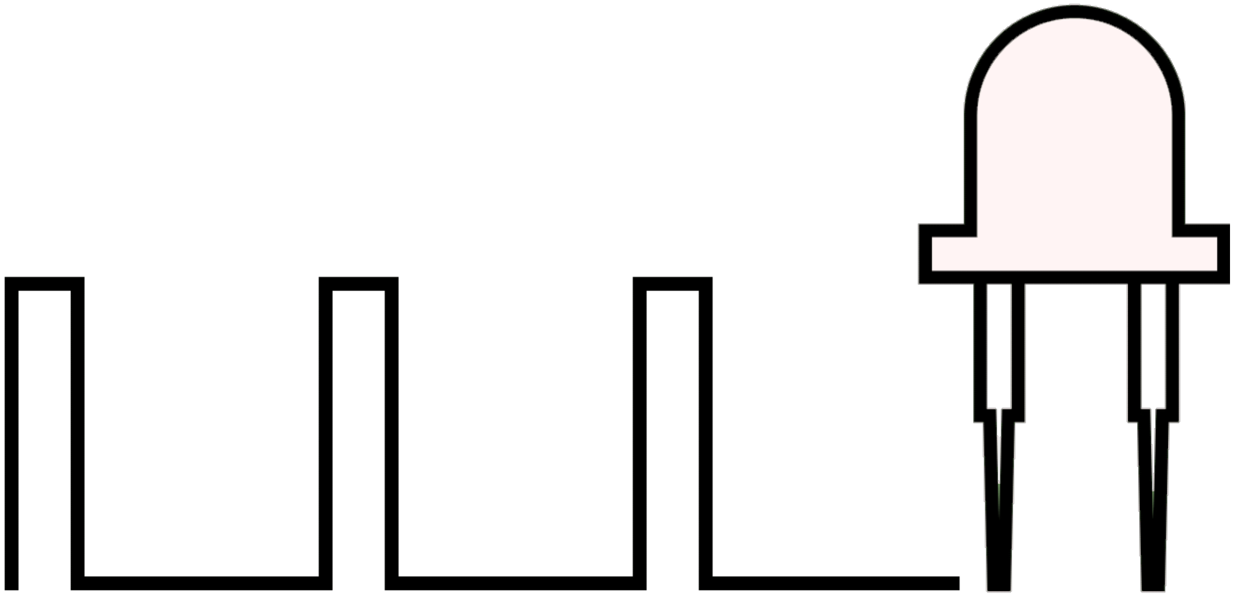
Controlling LED Brightness using PWM

- We've seen how to turn an LED on and off, but how do we control its brightness levels?
- An LED's brightness is determined by controlling the amount of current flowing through it, but that requires a lot more hardware components.
- A simple trick we can do is to flash the LED faster than the eye can see!
- By controlling the amount of time the LED is on versus off, we can change its perceived brightness.
- This is known as *Pulse Width Modulation* (PWM).

https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/experiment-1-digital-input-and-output

Controlling LED Brightness using PWM

Below we see how we can use PWM to control the brightness of a LED



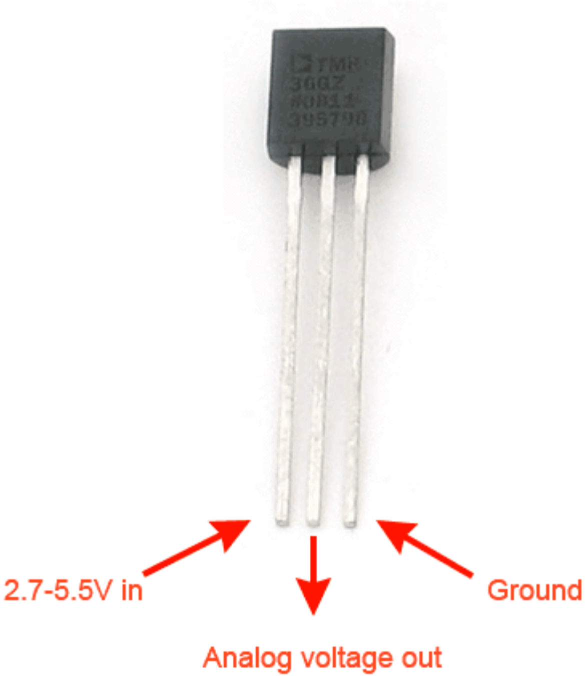
https://www.electronicwings.com/raspberry-pi/raspberry-pi-pwm-generation-using-python-and-c

TMP36

Temperature Sensor

Hans-Petter Halvorsen

TMP36 Temperature Sensor

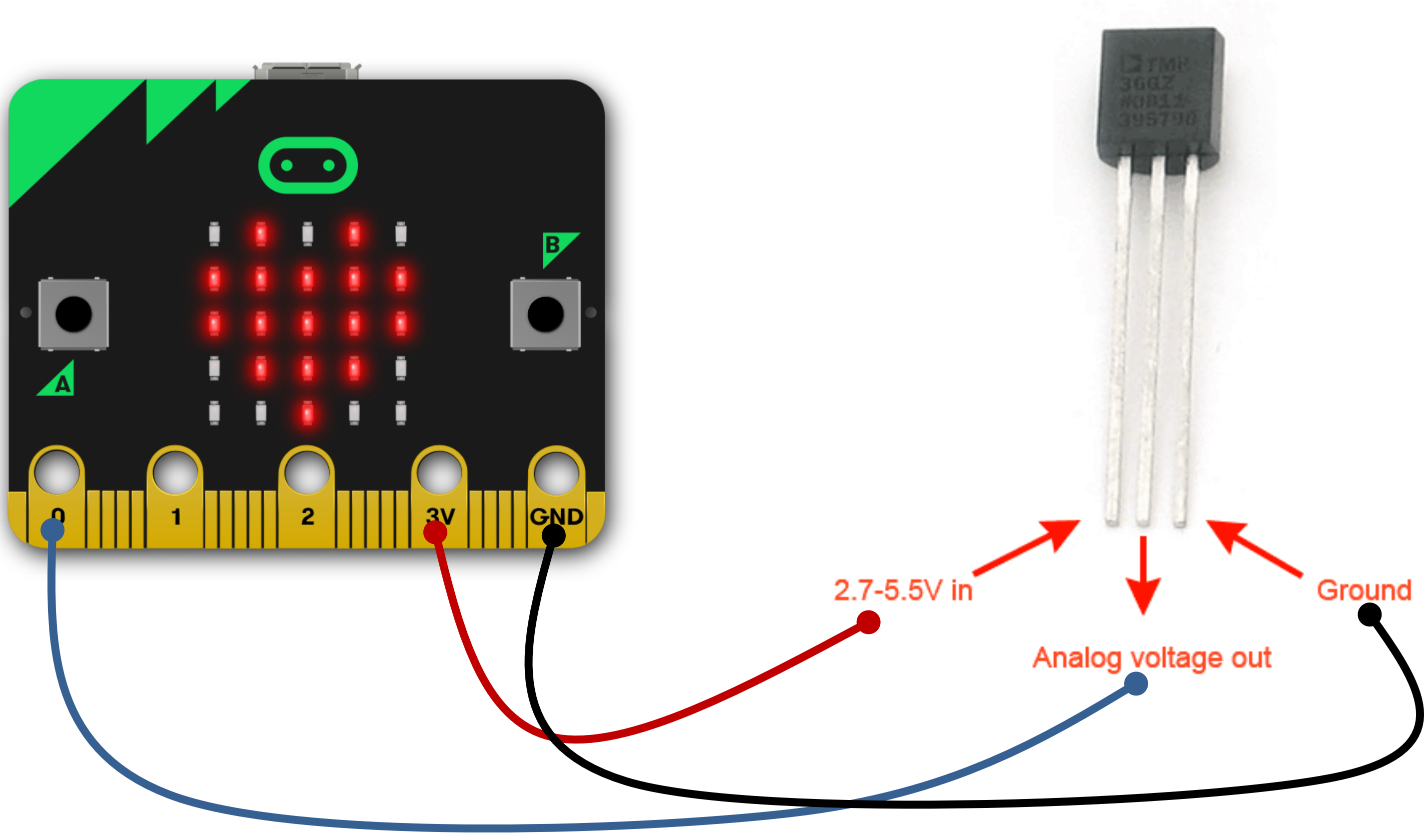


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

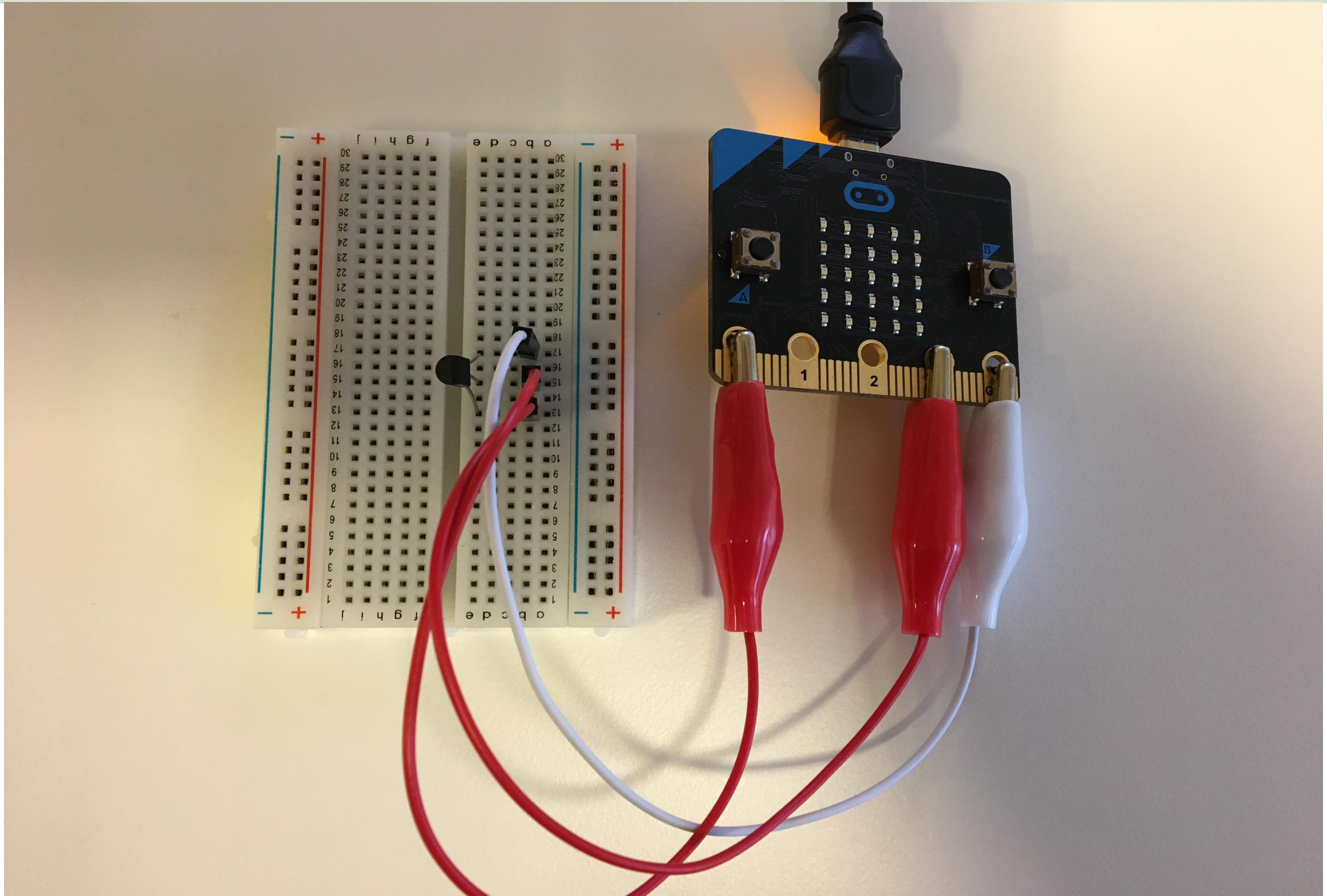
They use the fact as temperature increases, the voltage across a diode increases at a known rate.

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

Wiring



Breadboard and Crocodile Wires



Python

```
from microbit import *

while True:
    adc = pin0.read_analog()
    display.scroll(adc)
    sleep(5000)
```

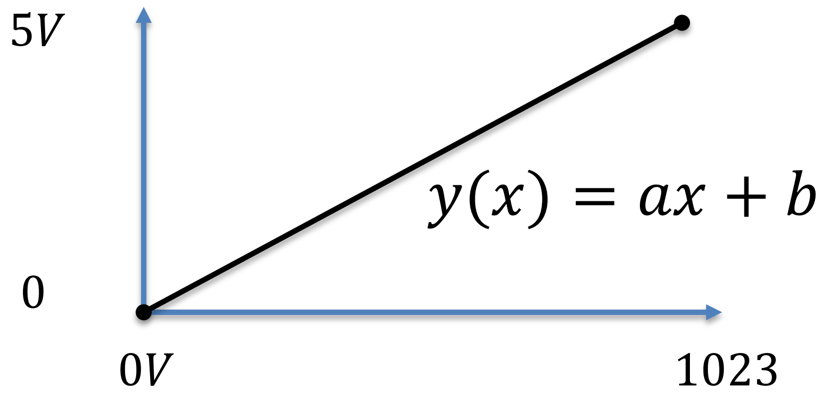
ADC Value to Voltage Value

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

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

ADC = 0 -> 0v

ADC = 1023 -> 3.3v



This gives the following formula:

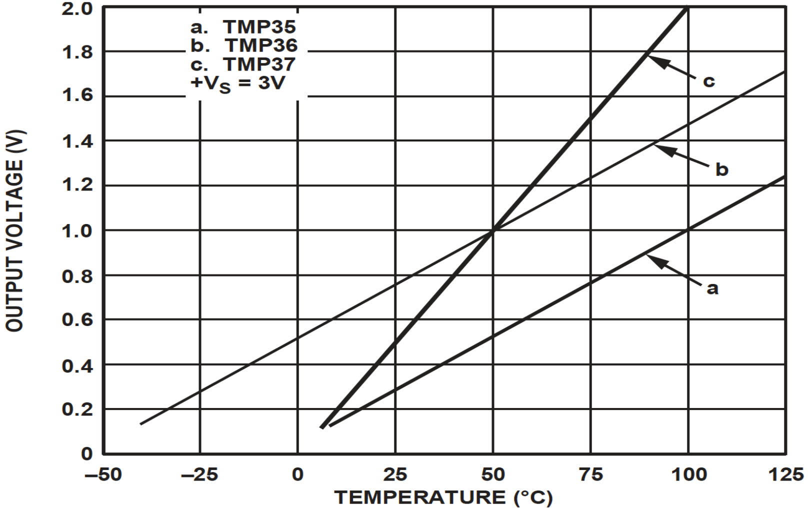
$$y(x) = \frac{3.3}{1023}x$$

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 form Voltage (V) to degrees Celsius

From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25°C)$$
$$(x_2, y_2) = (1V, 50°C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

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

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

This gives:

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

Then we get the following formula:

$$y = 100x - 50$$

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)
```

SPI and I2C

Hans-Petter Halvorsen

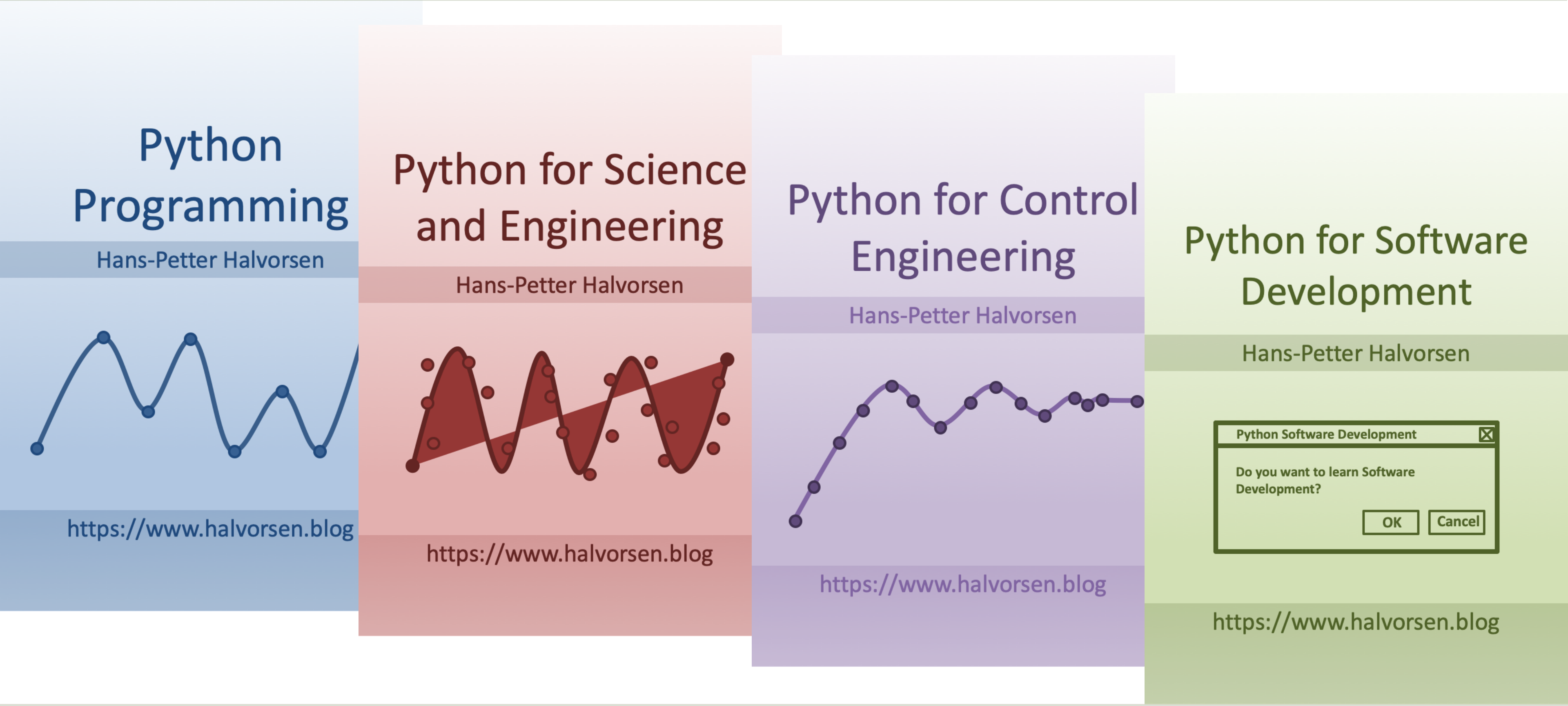
SPI

I2C

XXX

XXX

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog