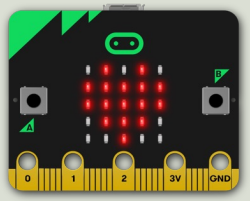
https://www.halvorsen.blog



micro:bit and LEDs

Hans-Petter Halvorsen

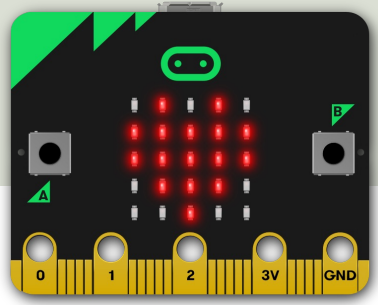
Contents

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

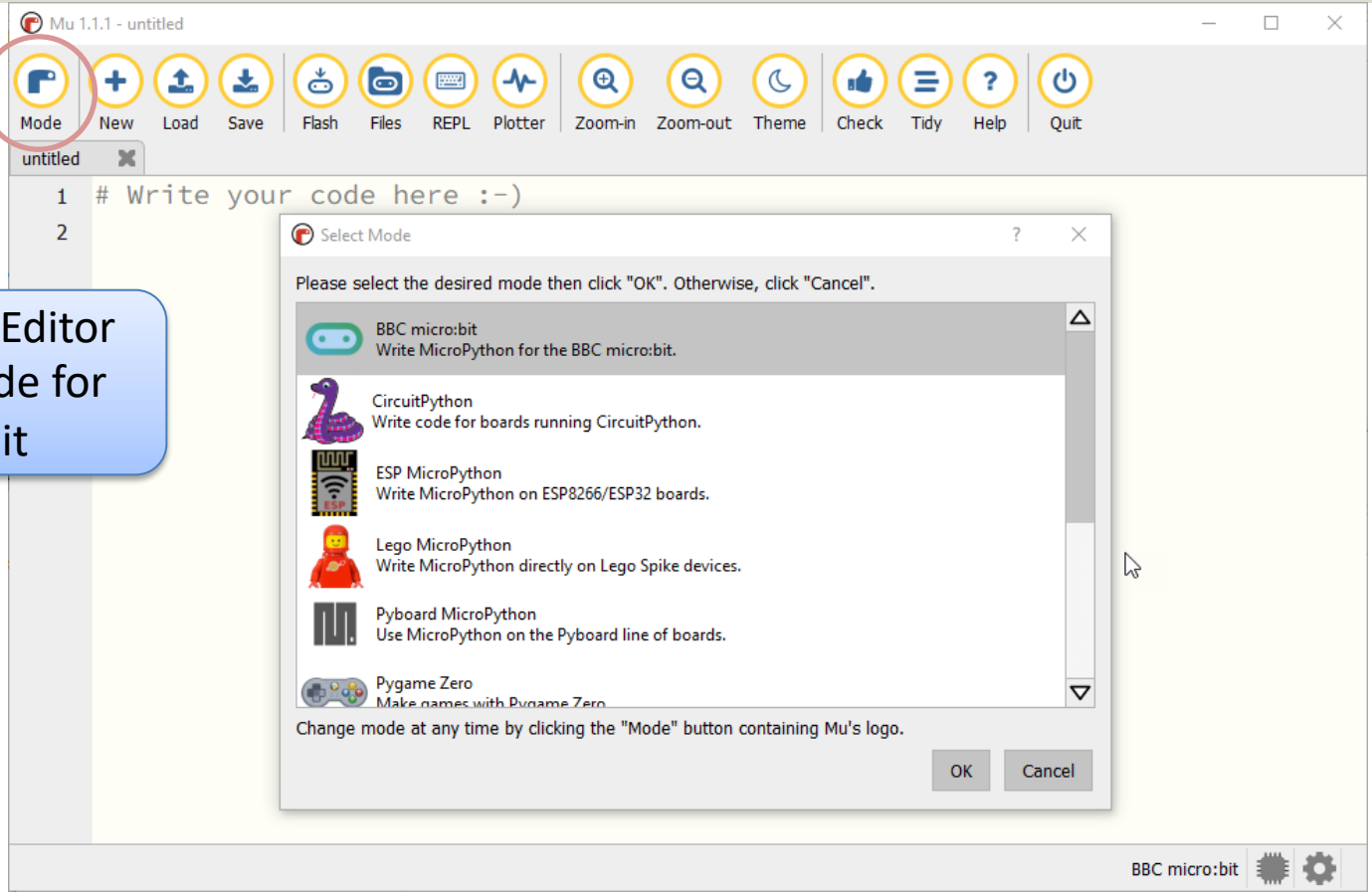
https://microbit.org

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

Hans-Petter Halvorsen

Table of Contents

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()
```

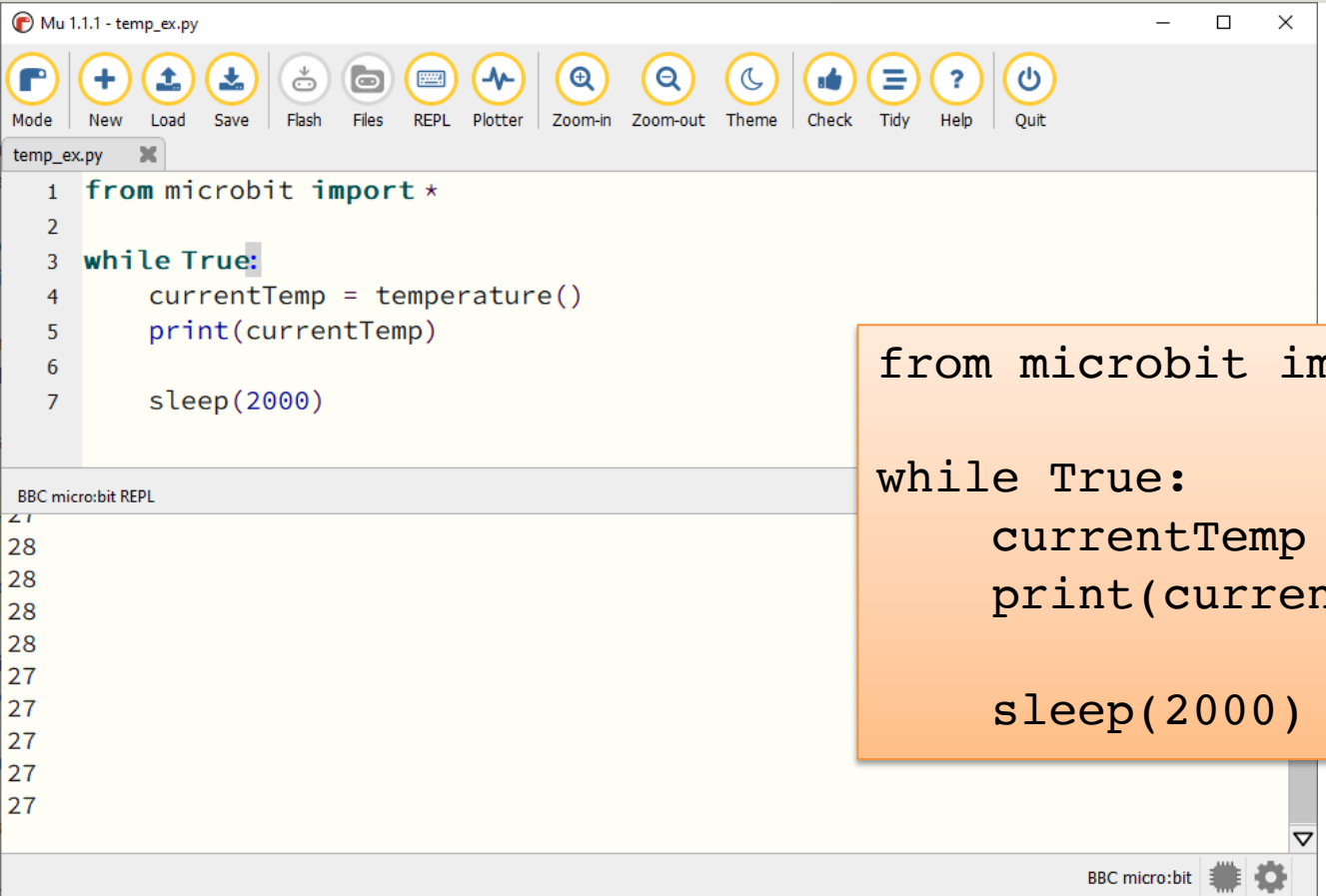
This examples displays the temperature on the LED matrix:

```
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

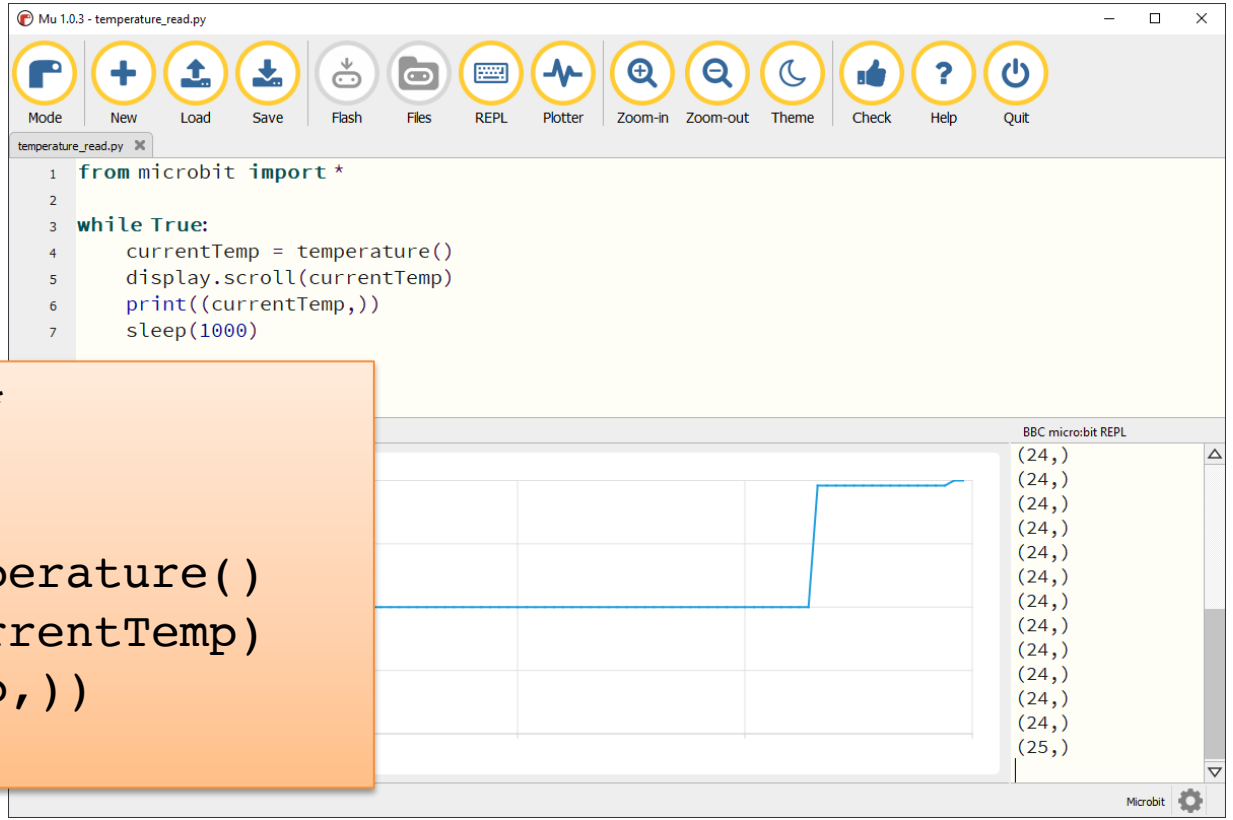



```
from microbit import *

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

    sleep(2000)
```

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

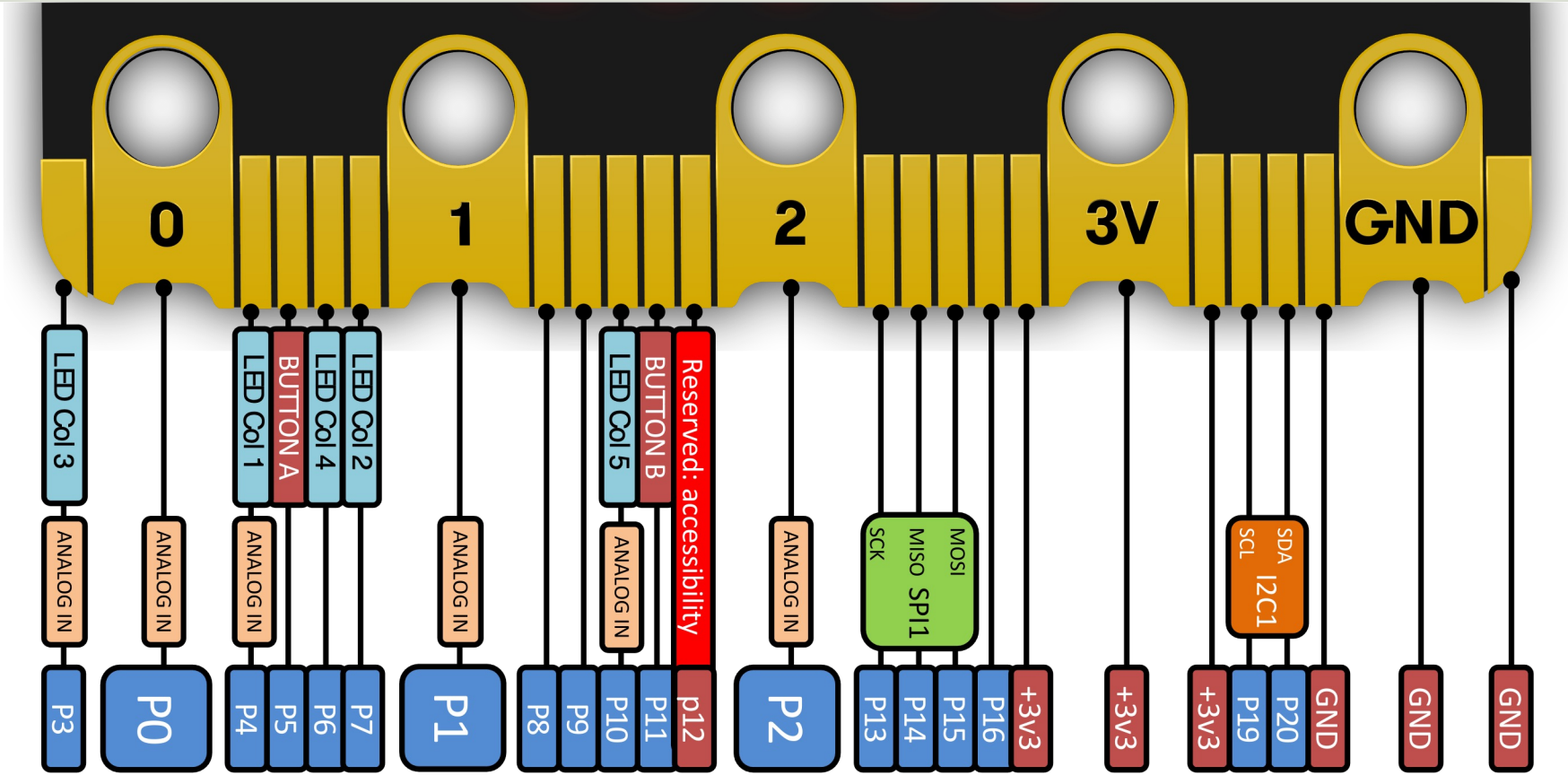
micro:bit I/O Pins

Hans-Petter Halvorsen

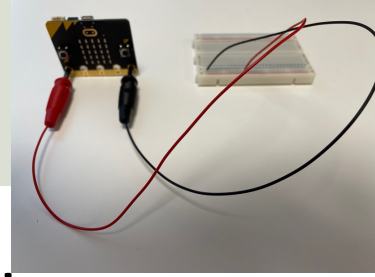
Table of Contents

https://microbit.pinout.xyz/

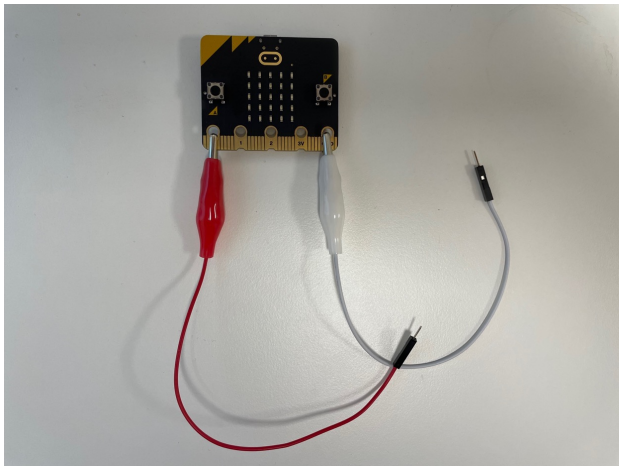
micro:bit I/O Pin Overview



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



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

https://makecode.microbit.org/device/crocodile-clips

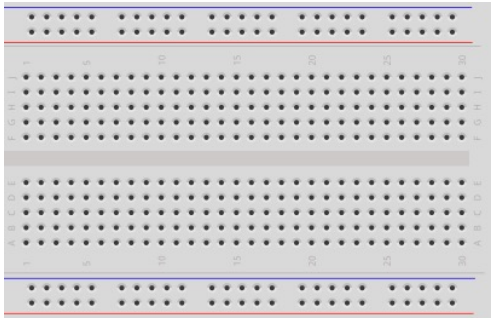
Types of I/O Pins

- **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

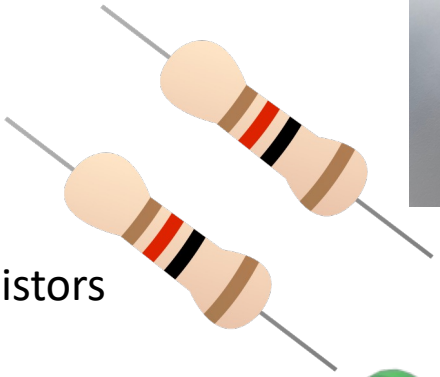
We will only use an Analog/Digital Input/Output pins in this Tutorial

Component Examples

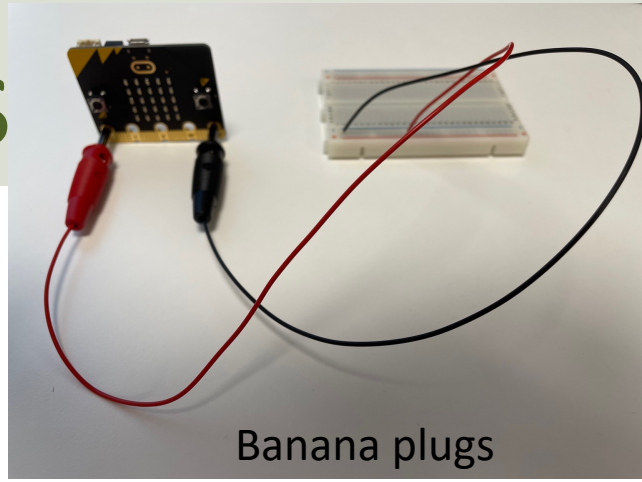


Breadboard

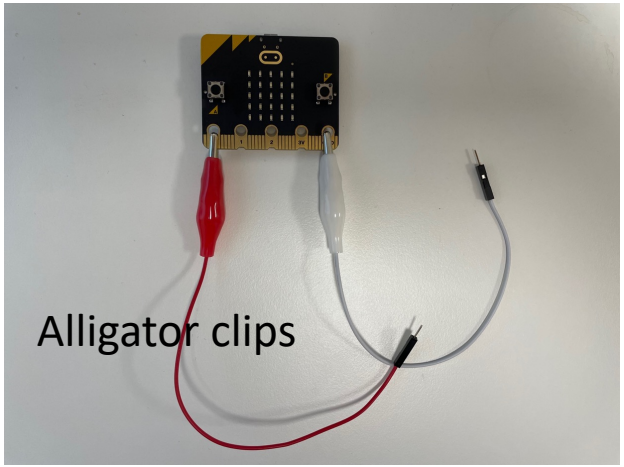
Temperature Sensor



Resistors

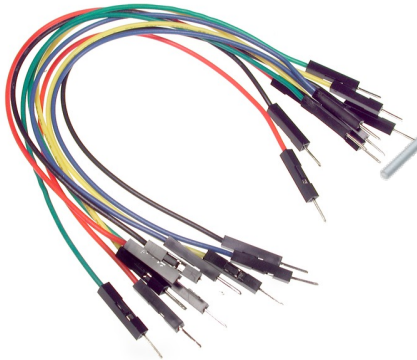


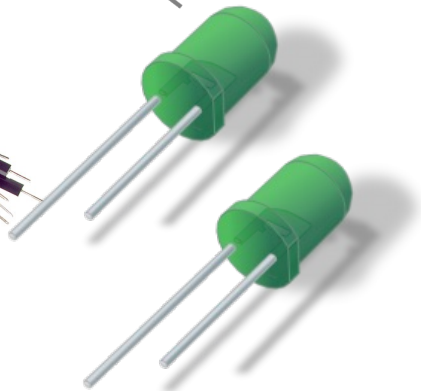
Banana plugs



Alligator clips

Wires

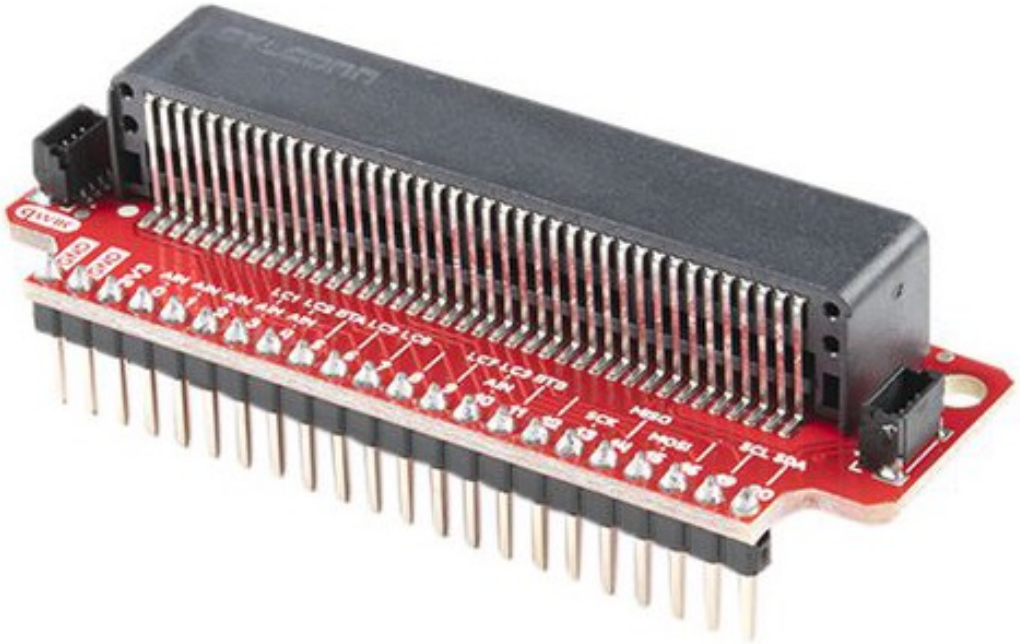




LEDs

Multimeter

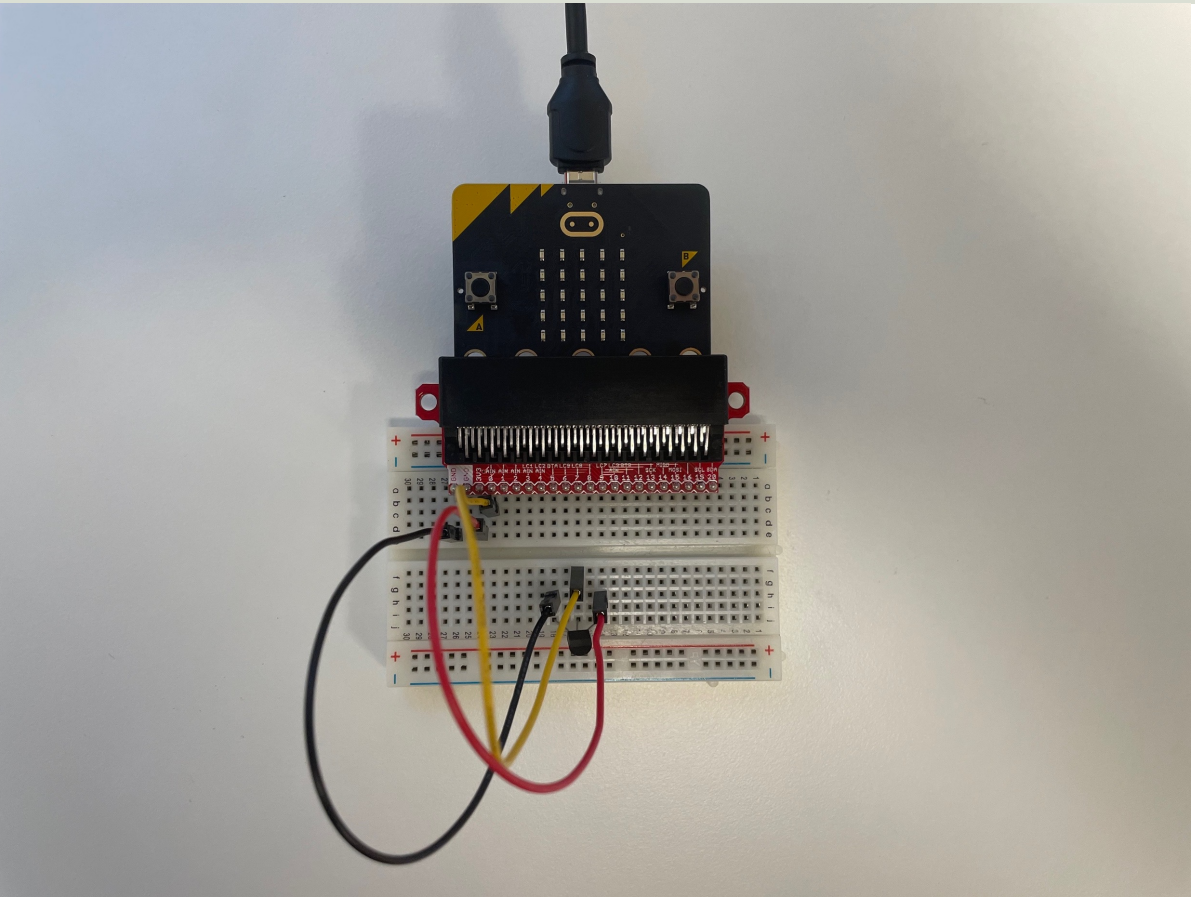
Adapter Breakout Board for micro:bit



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

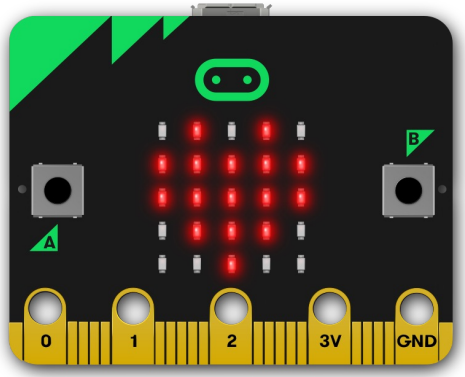
LEDs

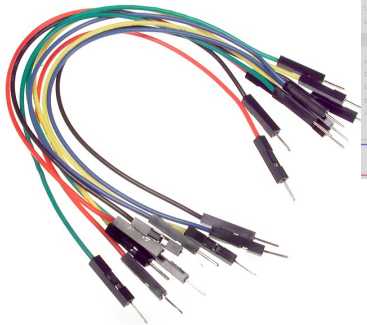
Hans-Petter Halvorsen

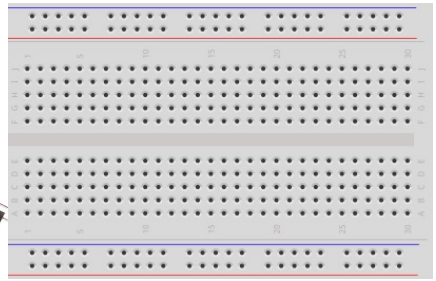
Table of Contents

Necessary Equipment

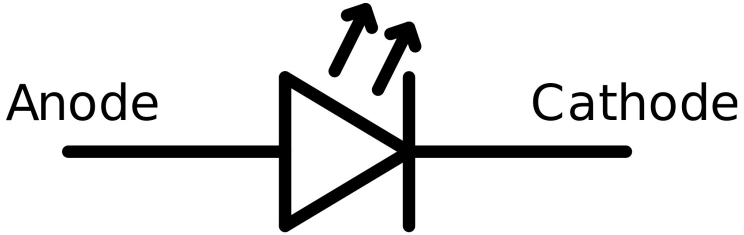
- micro:bit
- Breadboard
- LED
- Resistor, $R = 270\Omega$
- Wires (Jumper Wires)

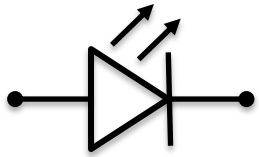


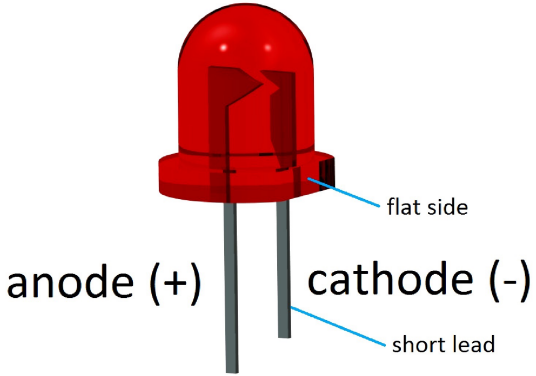


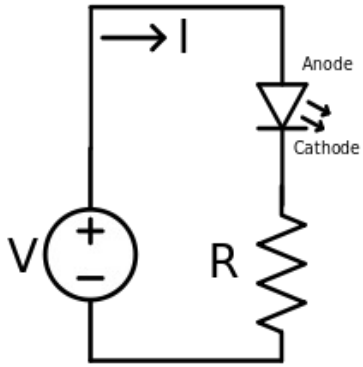


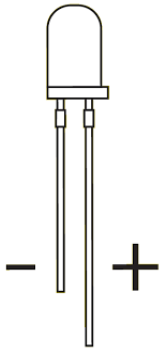
LED





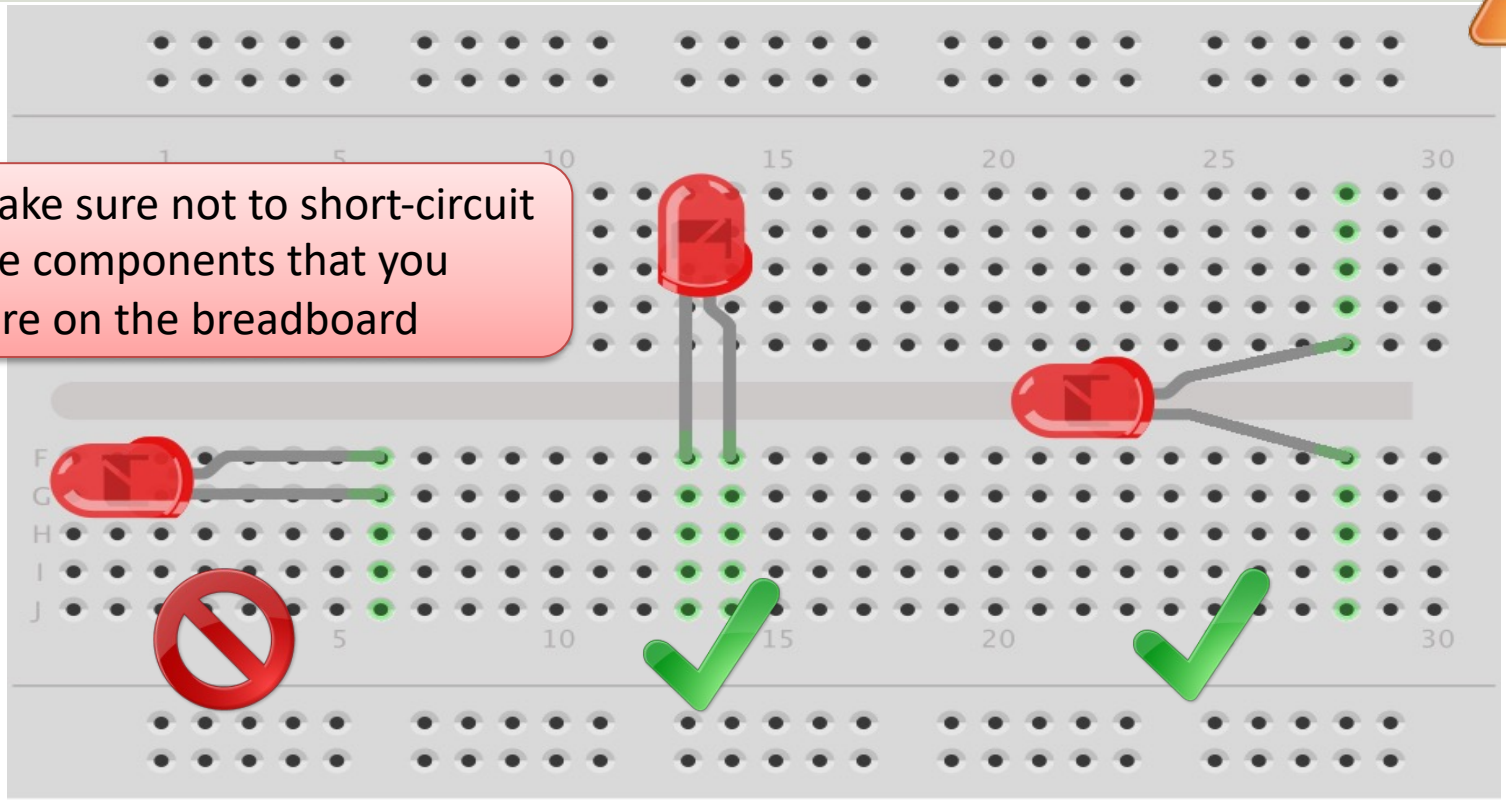






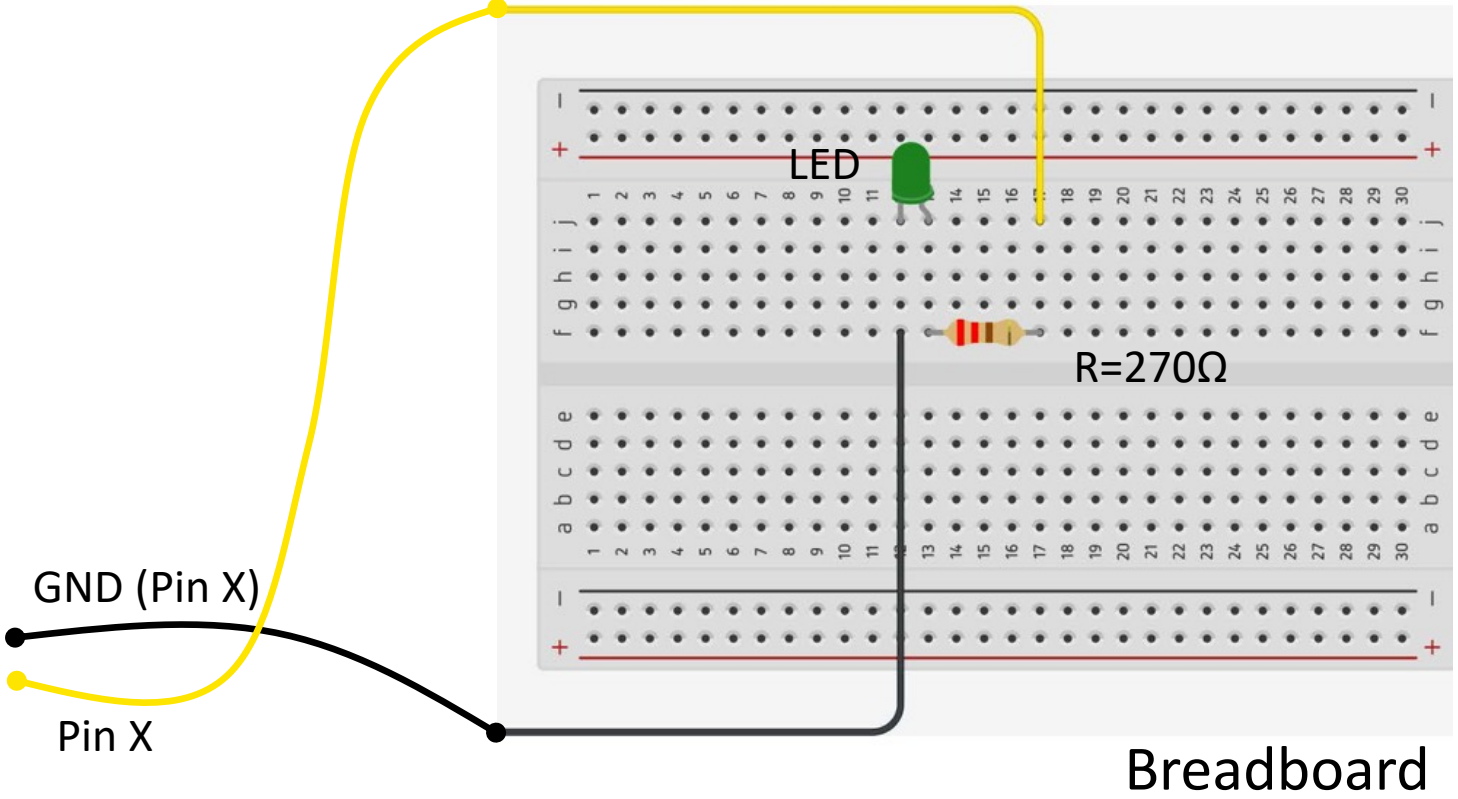
[Wikipedia]

Breadboard Wiring



LED Example

micro:bit GPIO Pins



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:

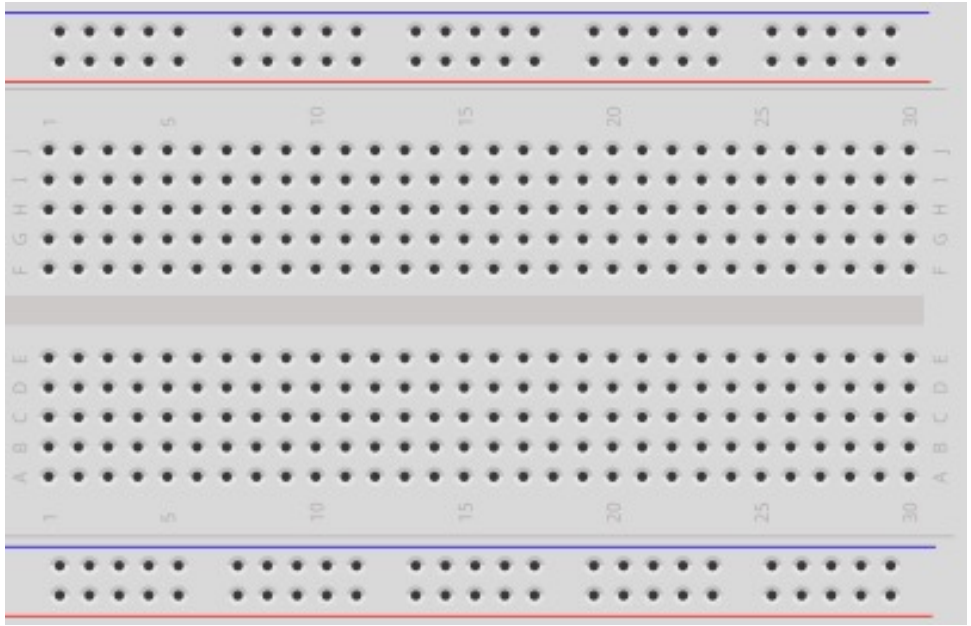
$$U = RI$$

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

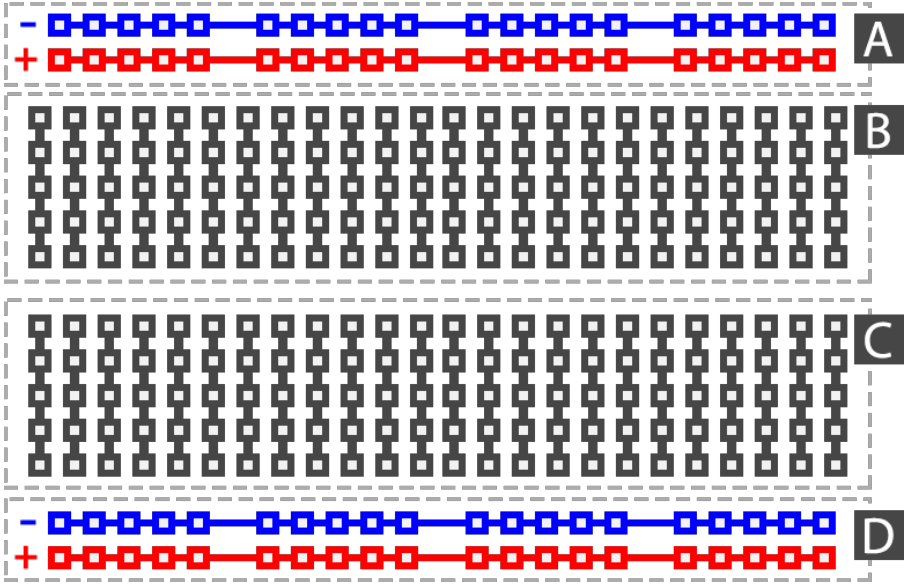
$$R = \frac{U}{I}$$

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

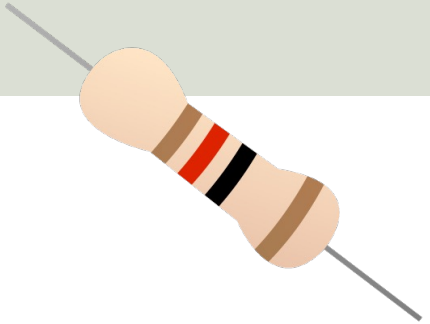
Breadboard



A breadboard is used to wire electric components together

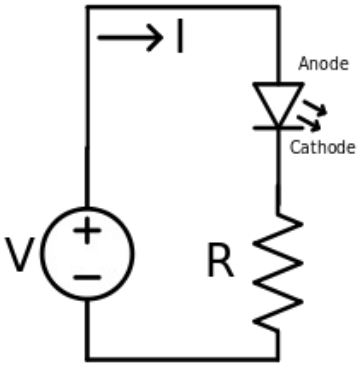


Resistors



Resistance is measured in Ohm (Ω)

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



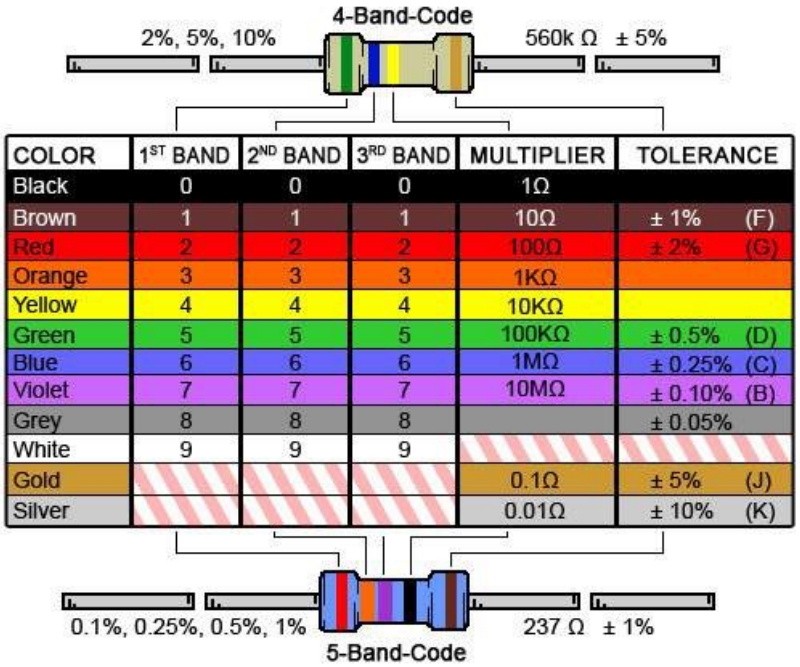
The resistance can be found using **Ohms Law**

$$U = RI$$

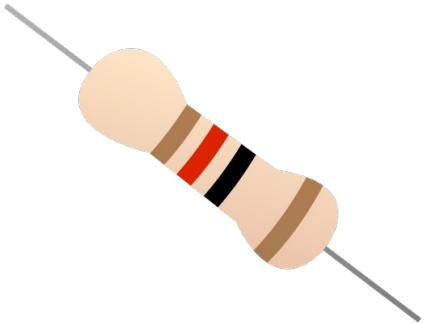
https://en.wikipedia.org/wiki/Resistor

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/

Examples

LED Examples

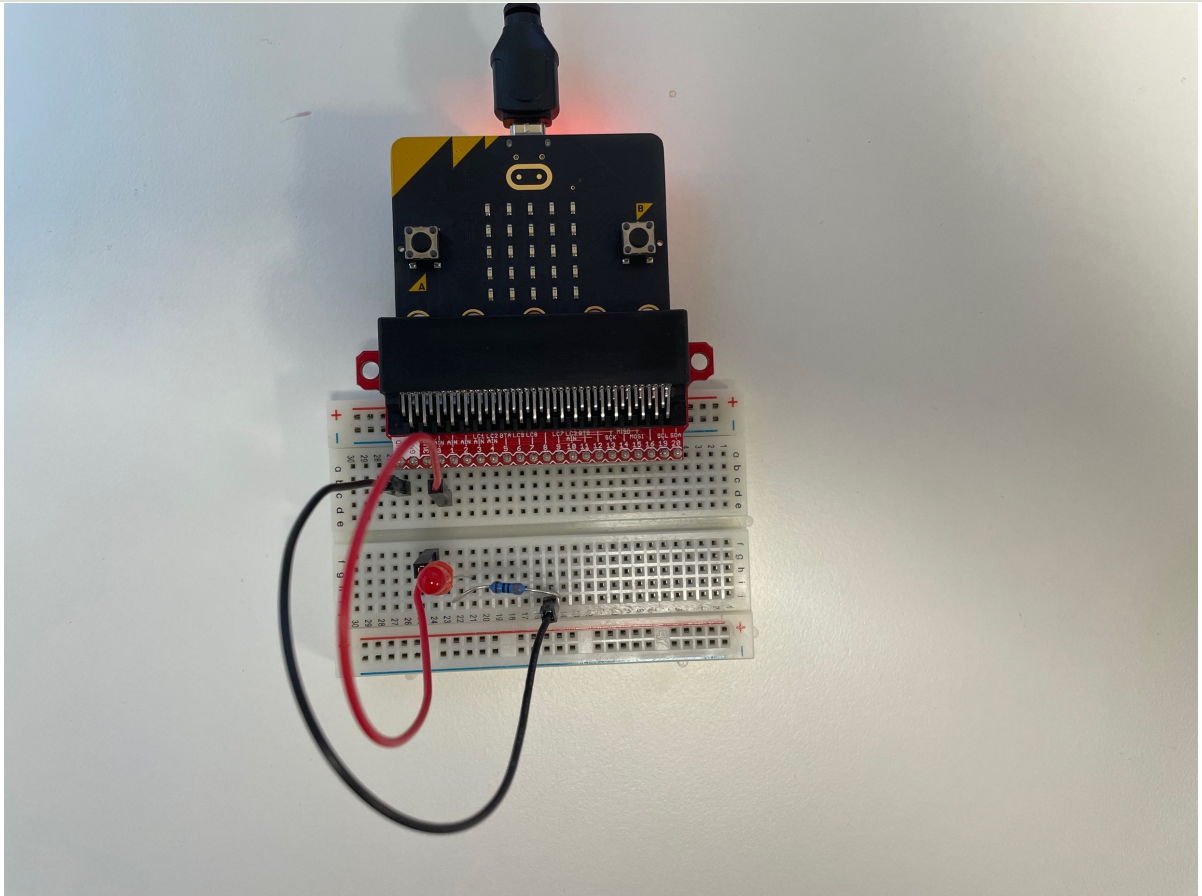
- Blinking LED
- Controlling the Brightness of a LED using PWM
- Turn LED on and off using one of the built in Buttons (A or B)
- Turn LED on and off using the built in Touch button

Blinking LED

Hans-Petter Halvorsen

Table of Contents

Wiring

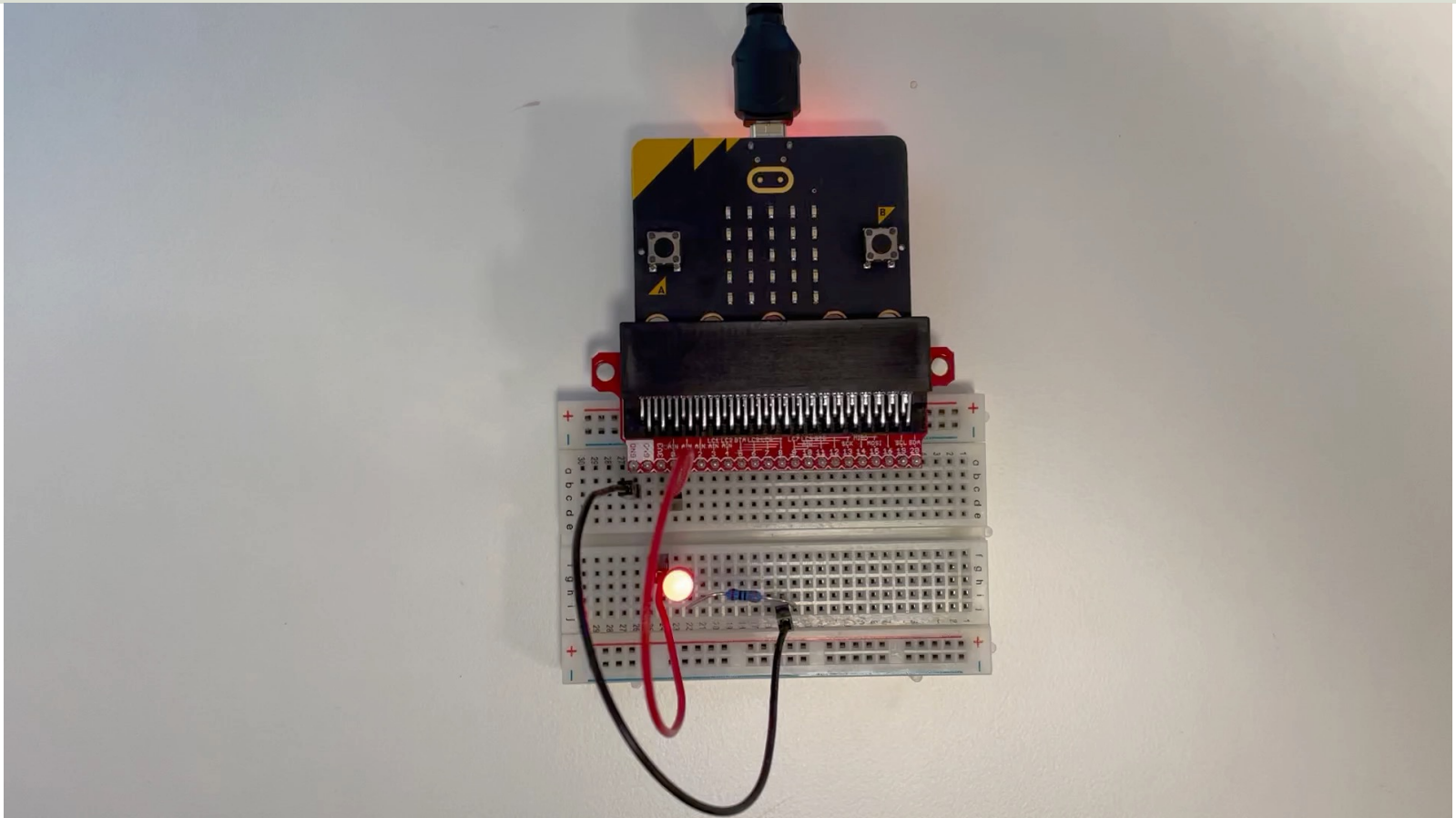


Python

```
from microbit import *

while True:
    pin0.write_digital(1)
    sleep(1000)
    pin0.write_digital(0)
    sleep(1000)
```

Results



Controlling the Brightness of a LED using PWM

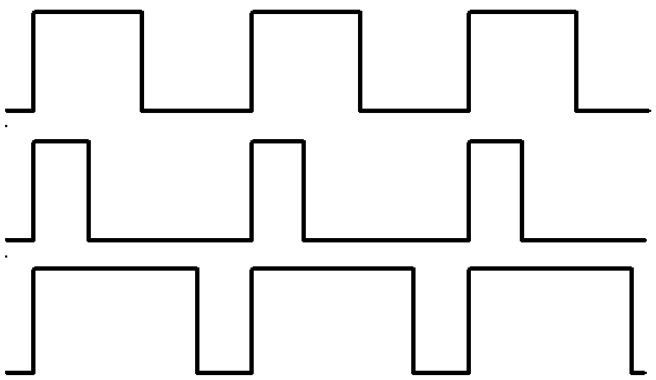
Hans-Petter Halvorsen

Table of Contents

micro:bit and PWM

Pulse-Width Modulation

The pins of your board cannot output analog signal the way an audio amplifier can do it – by modulating the voltage on the pin. Those pins can only either enable the full 3.3V output or pull it down to 0V. However, it is still possible to control the brightness of LEDs or speed of an electric motor, by switching that voltage on and off very fast, and controlling how long it is on and how long it is off. This technique is called Pulse-Width Modulation (PWM), and that's what the write_analog() method does.



The first one would be generated by `write_analog(511)`, as it has exactly 50% duty – the power is on half of the time, and off half of the time. The result of that is that the total energy of this signal is the same, as if it was 1.65V instead of 3.3V.

The second signal has 25% duty cycle, and could be generated with `write_analog(255)`. It has similar effect as if 0.825V was being output on that pin.

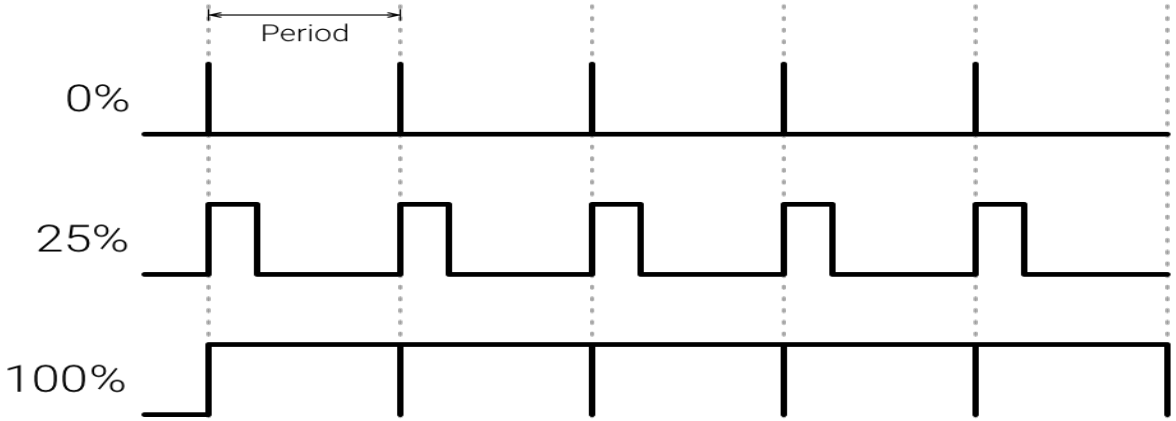
The third signal has 75% duty cycle, and can be generated with `write_analog(767)`. It has three times as much energy, as the second signal, and is equivalent to outputting 2.475V on th pin.

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

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:



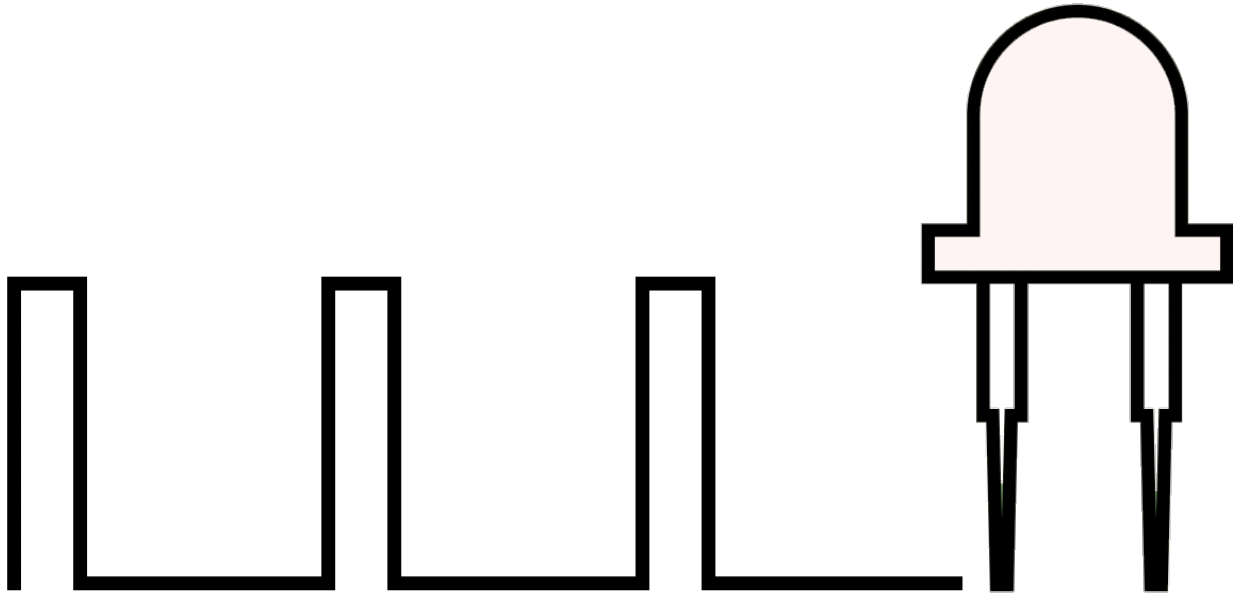
Controlling LED Brightness using PWM

- 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

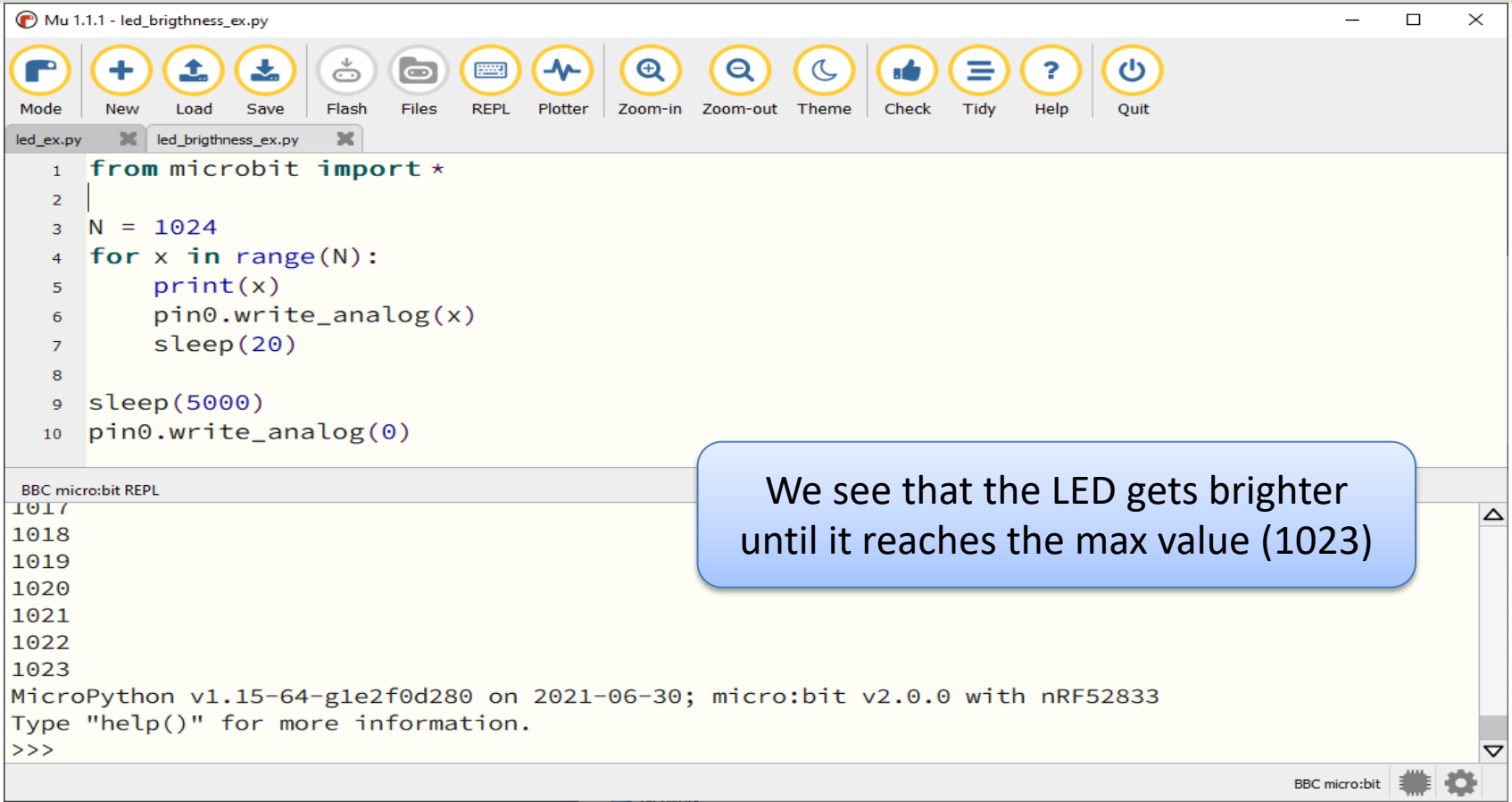
Controlling LED Brightness using PWM

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

Python



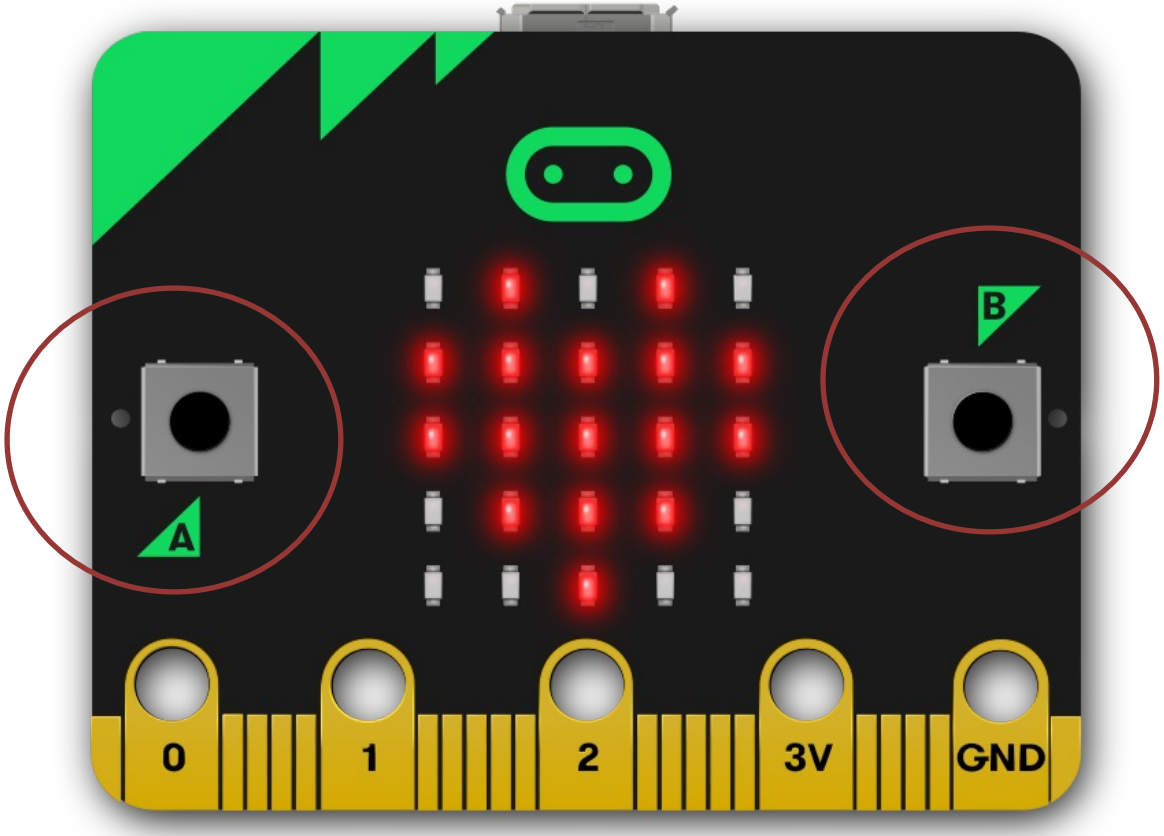
We see that the LED gets brighter until it reaches the max value (1023)

Turn LED on and off using one of the built in Buttons (A or B)

Hans-Petter Halvorsen

Table of Contents

Buttons (A and B)



Buttons (A and B) Example

```
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)
```

LED + Button A + B

```
from microbit import *

while True:
    if button_a.was_pressed():
        pin0.write_digital(1)
    elif button_b.was_pressed():
        pin0.write_digital(0)
    sleep(1000)
```

Turn LED on and off using the built in Touch button

Hans-Petter Halvorsen

Table of Contents

Python

```
from microbit import *

touch = 0

while True:
    if pin_logo.is_touched():
        display.show(Image.HAPPY)
        touch = 1
    else:
        display.show(Image.ASLEEP)
        touch = 0

    pin0.write_digital(touch)
    sleep(1000)
```

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog