Technology understanding

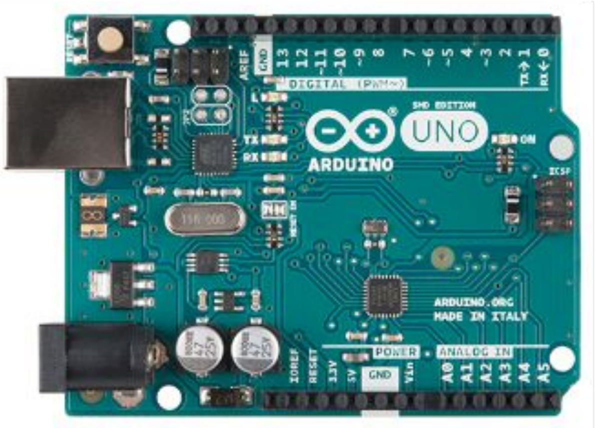
MODULE 2

Agenda ~4t

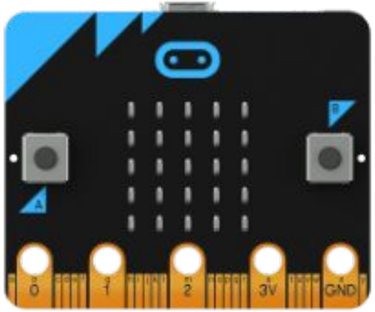
- 09.00 - 11.00 Workshop
- 11.30 - 12.00 Tour of EUC Syd
- 12.00 - 12.45 Lunch
- 12.45 - 14.15 Workshop
- 14.15 - 14.30 Coffee and refreshments
- 14.30 - 14.50 Follow up, take-home messages, feedback
- 14.50 - 15.00 Goodby and farewell

Similarities and differences between the platforms

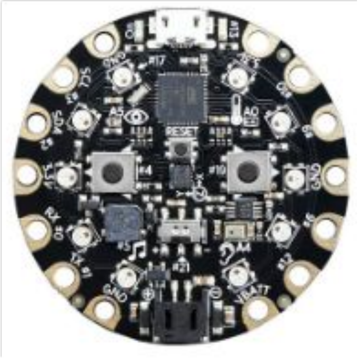
Arduino UNO



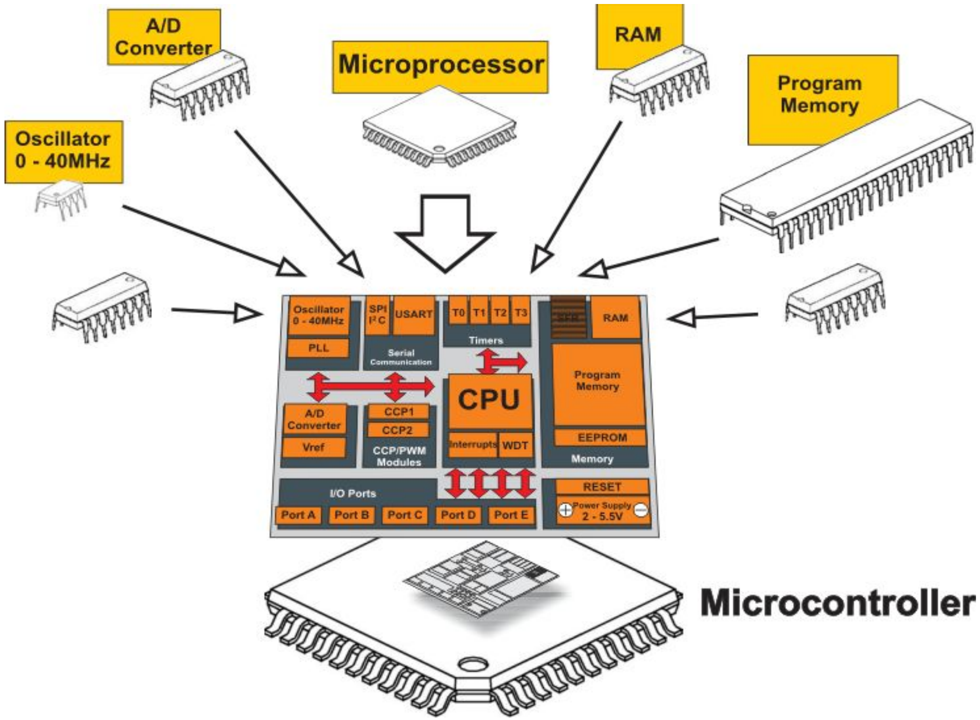
Micro:Bit



CPX



The Microcontroller

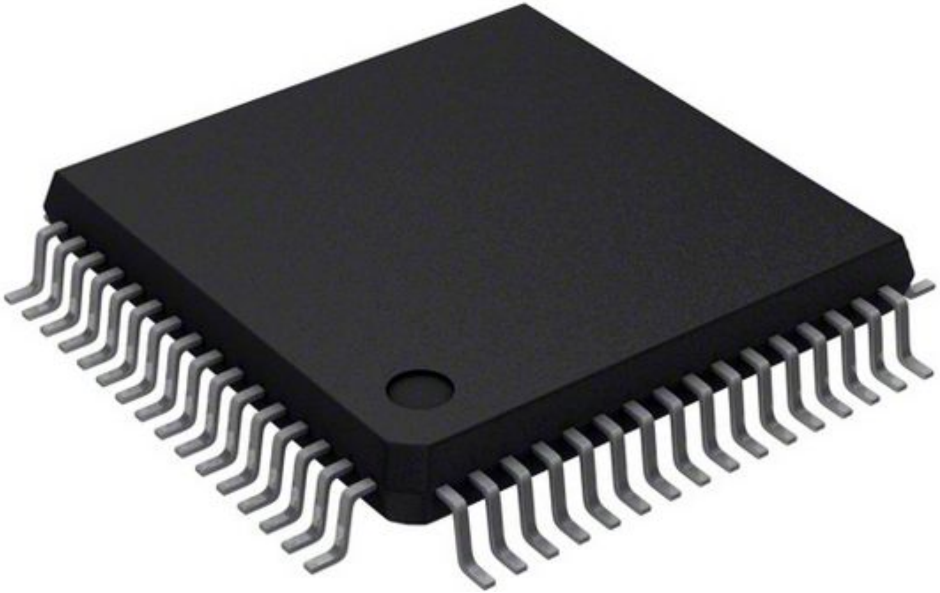


The Microcontroller

- An entire computer in an integrated circuit
- 8bit, 16bit, 32bit?
- Various devices to communicate with the outside world
    - Communication (i2c, art, usb, radio, etc.)
    - A / D converter
    - audio circuitry
    - Touch circuits
    - Display circuit

The Microcontroller

- Clock
- Digital Ports
- Analog Ports (A/D)
- Timers
- PWM
- Communikation
- Storage space

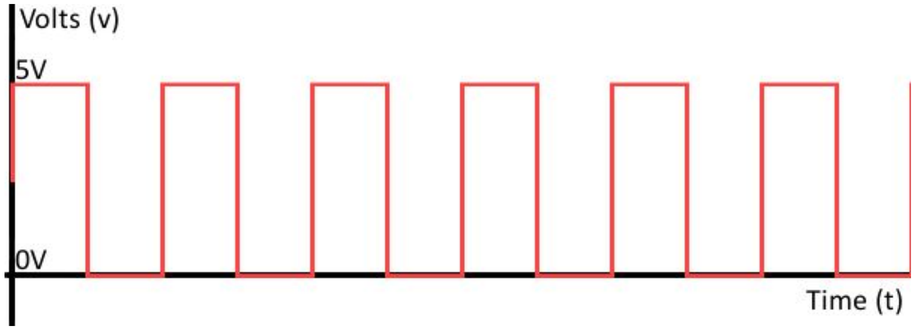


Clock

- The sinus rhythm of the microcontroll
- Controls how fast operations are performed
- Timer Interrupts
- Pulse Width Modulation
- Communication

Digital Port

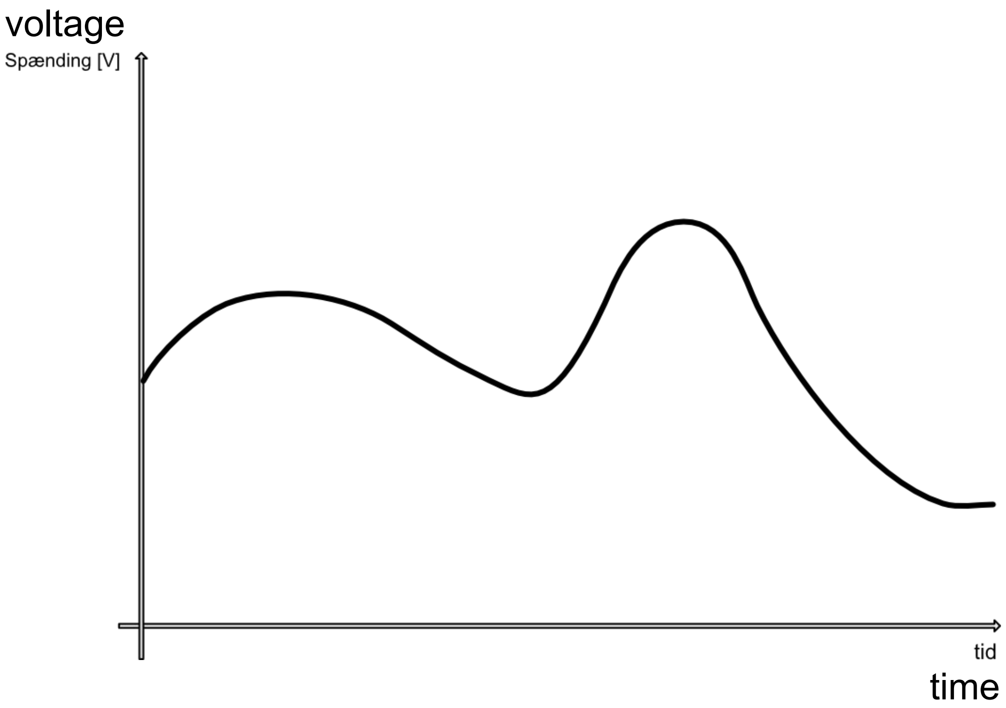
- 2 Levels – HIGH/LOW or 1/0
  - Depending on the Microcontroller's supply voltage



- Set as INPUT or OUTPUT
  - A limited current can flow from or to the port
- Internal Pull-up resistors

Analog Ports

- Samples analog values via the built-in A / D Converter
  - Blackboard example
  - Number of bits
  - Sample rate
- Can often act as digital I / O ports as well



A/D - D/A Conversion

- A/D (Analog til Digital)- Only the analog ports
  - In programming languages, typically called analogRead(), or something similar
  - Reads an analog voltage from a sensor and converts it to a digital value
  - The value depends the number of bits on the converter - typically 10 bits: 0 - 1023

A/D - D/A Conversion

- D/A - Output of analog values
  - The vast majority of microcontrollers can only emulate this via a digital signal
  - PWM (Pulse Width Modulation) - Blackboard example
  - Most often only a few selected digital ports
  - In programming languages, typically called analogWrite(), or something similar
  - Typically converts an 8-bit value (0 - 255) to a digital PWM signal that switches between 0 and 1, with interval lengths depending on the desired analog voltage value

Debugging

Debugging

- Probably the biggest challenge as it is very nonspecific
- The error can be in many places
- Often requires routine knowing where to look
    - Is it in the code?
    - Is it in the circuit?
    - Is it in the physical set-up?
- At the same time, this is where you often stop (it can take a long time)
    - But here, too, you often learn the most if you manage to solve the problem

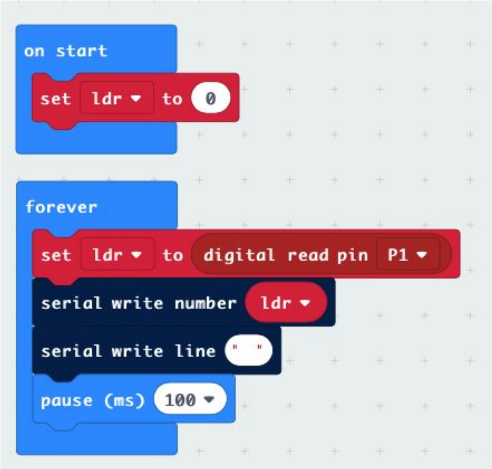
Experience

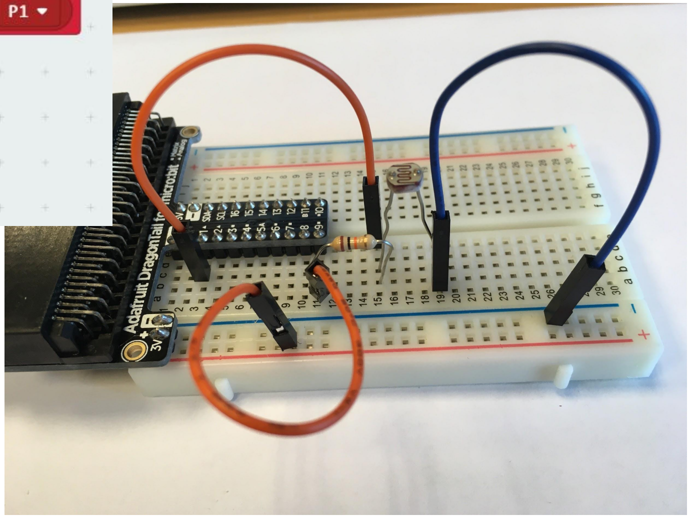
- What is your experience with debugging, while working with:
  - programming?
  - technology / robots?
- - and experiments in:
  - nature and science?
  - physics?
  - chemistry?
- Other subjects, where you are in contact with debugging
  - Mathematics?
  - Language (Dansk, English, Deutch)?
  - Craft work, Cooking, Visual Arts etc.?

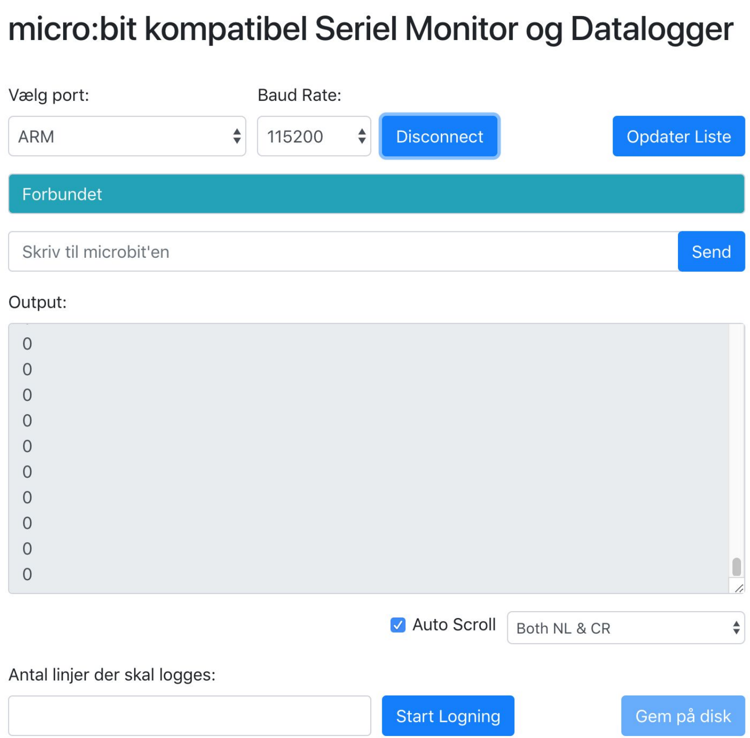
Debugging

- LED (Excellent for showing code flow)
- Multimeter (Good for DC voltage and current and continuity test)
    - Requires a little practice to get comfortable with it
- Oscilloscope (A class for itself. Can do anything, but its expensive too)
    - Can in most cases do far too much and it takes a lot of time to learn how to use it
- Display (micro:bit)
- The USB-connection to the PC
    - E.g. via data logging software or a serial monitor

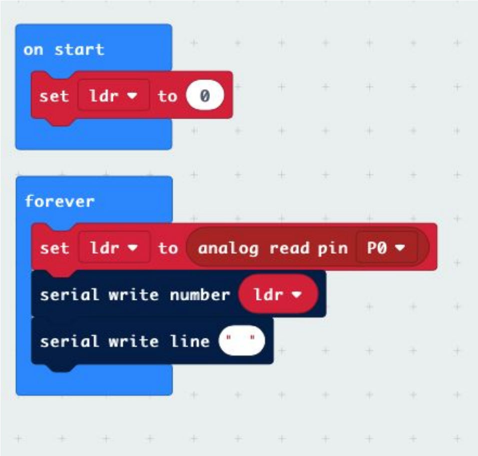
Find the errors #1

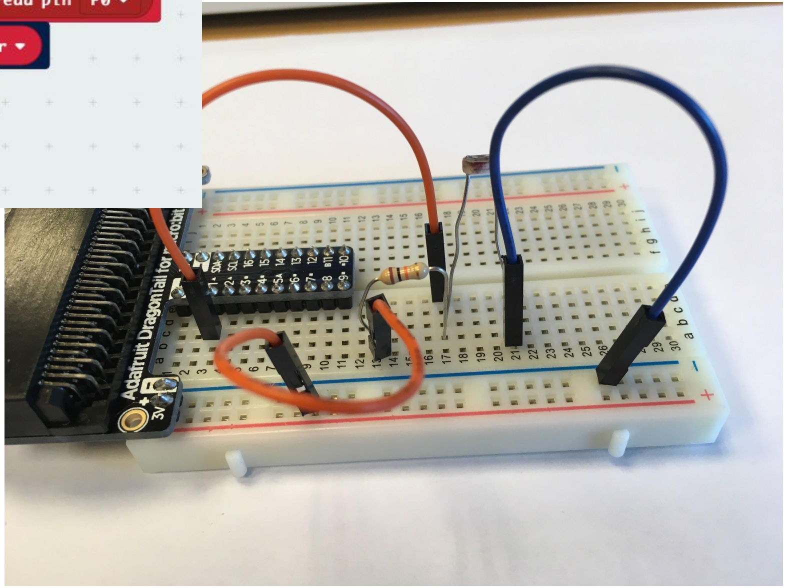


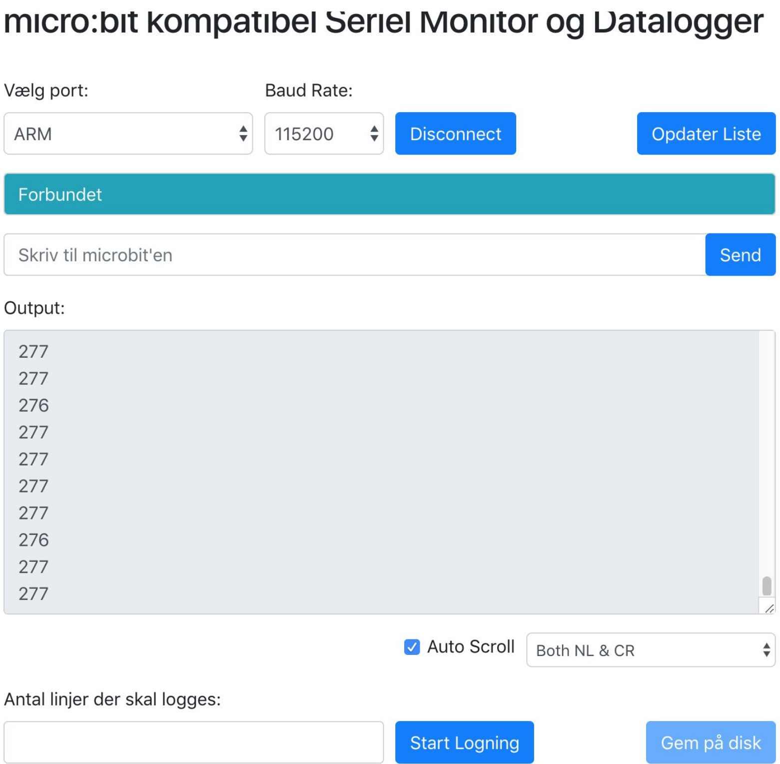




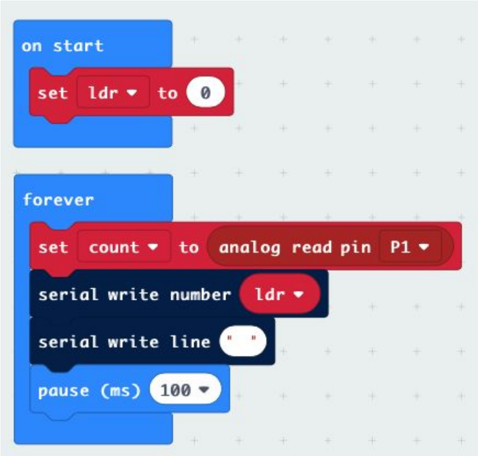
Find the errors #2

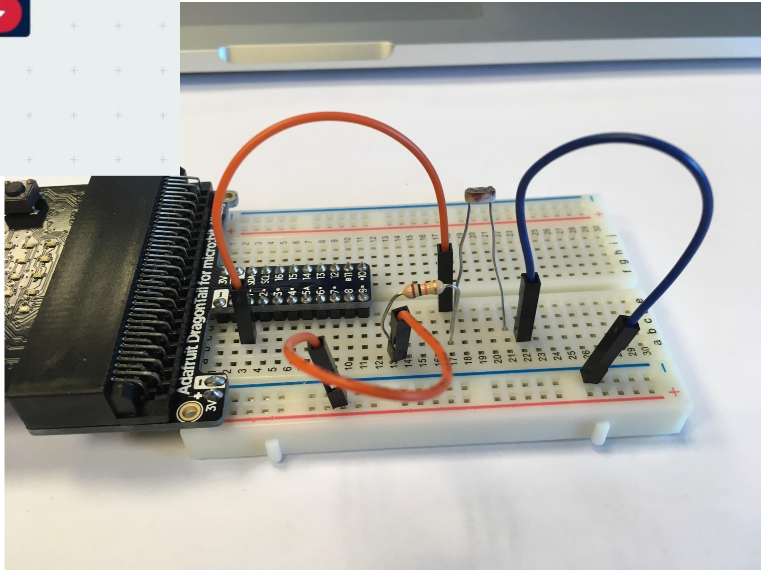


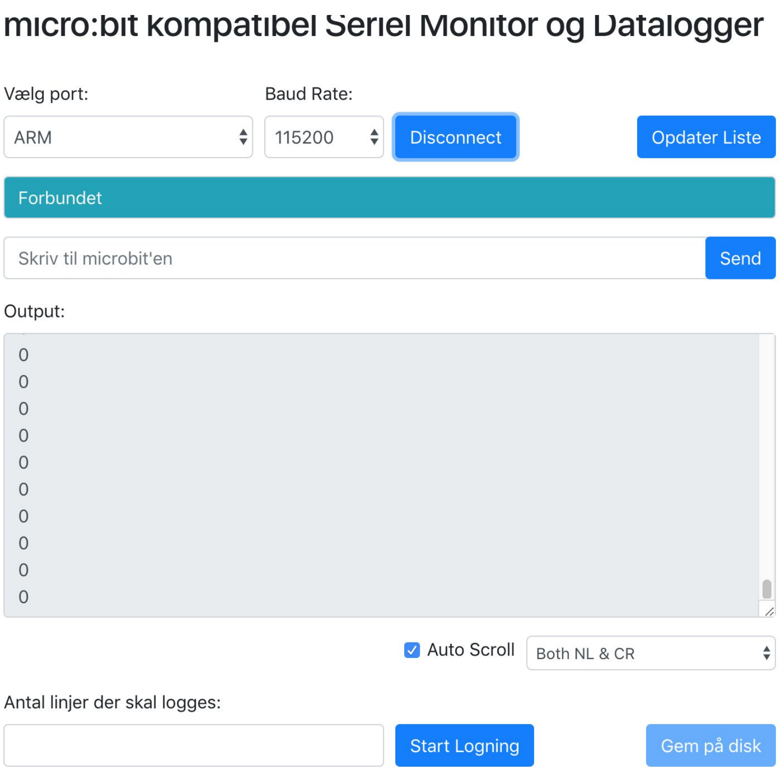




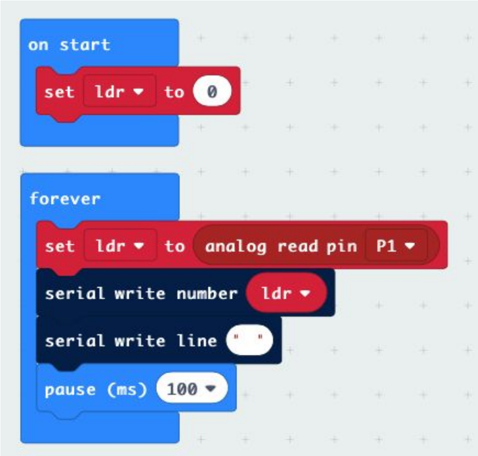
Find the errors #3

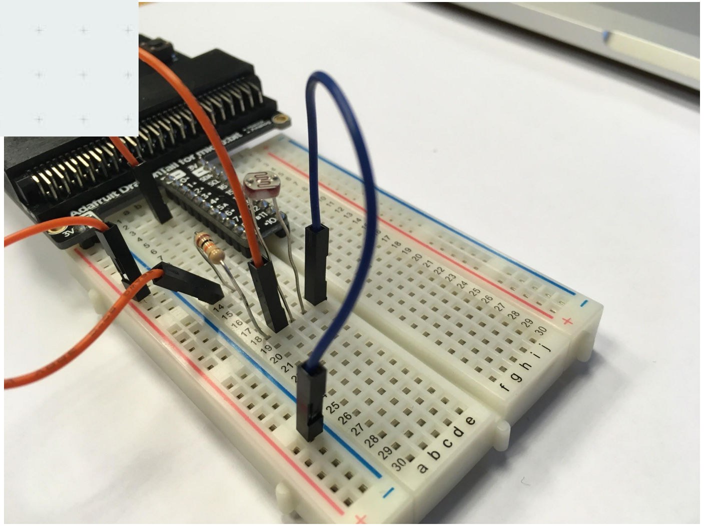


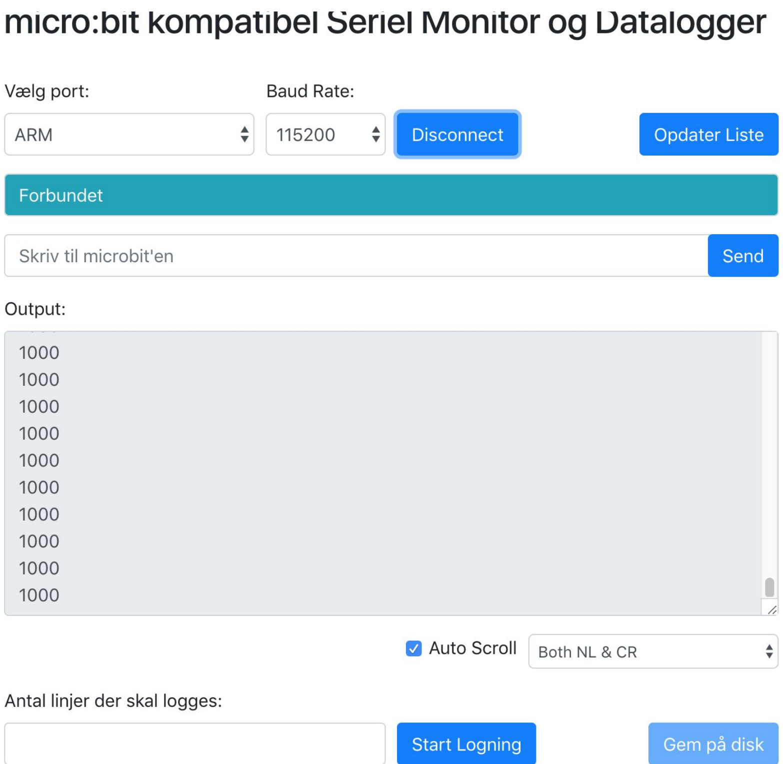




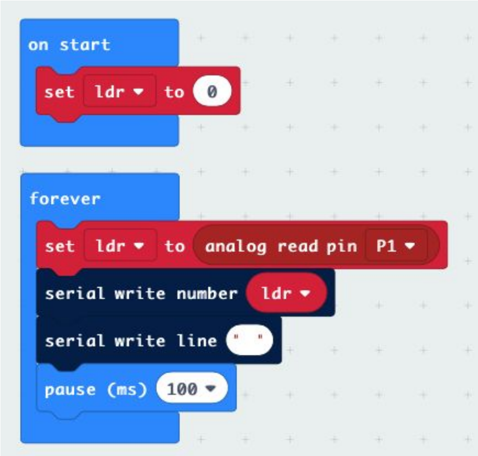
Find the errors #4

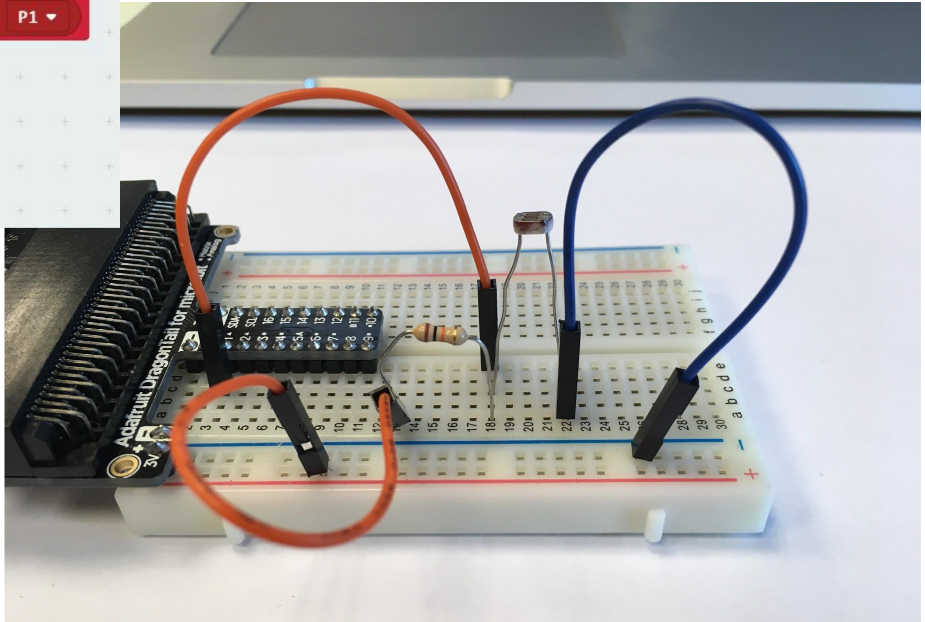


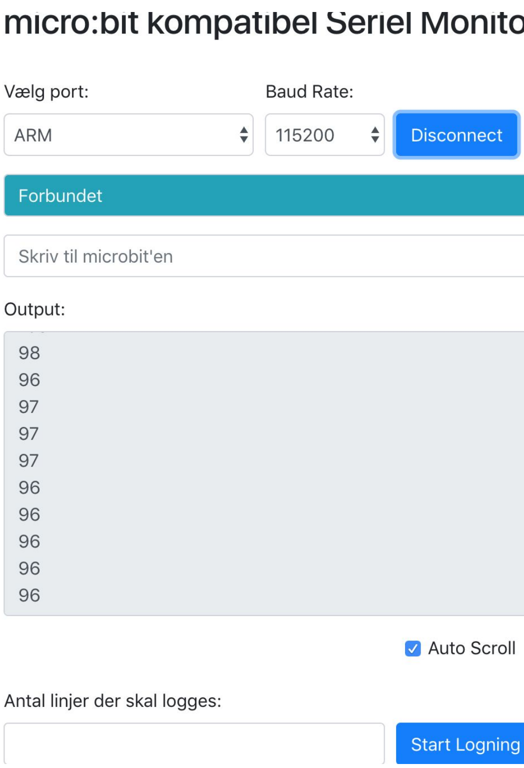




Find the errors #5

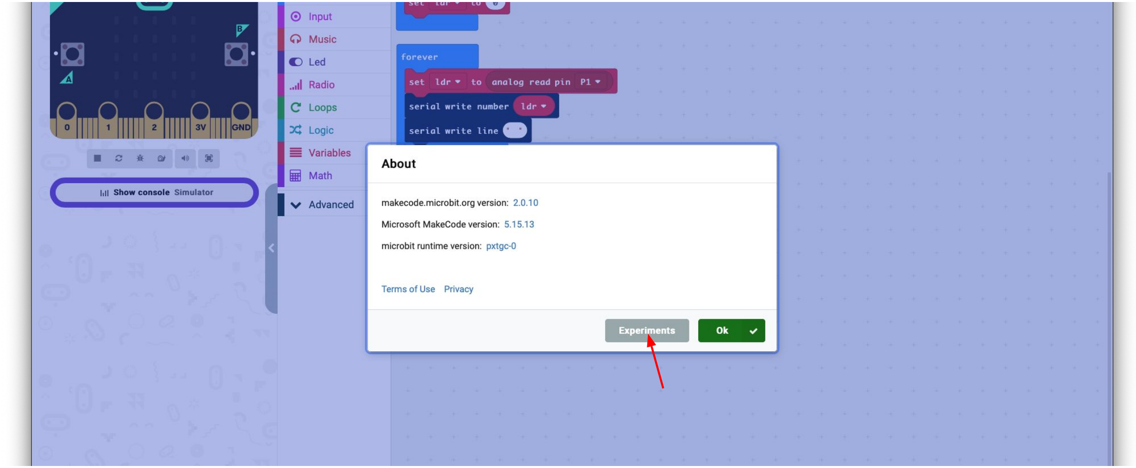




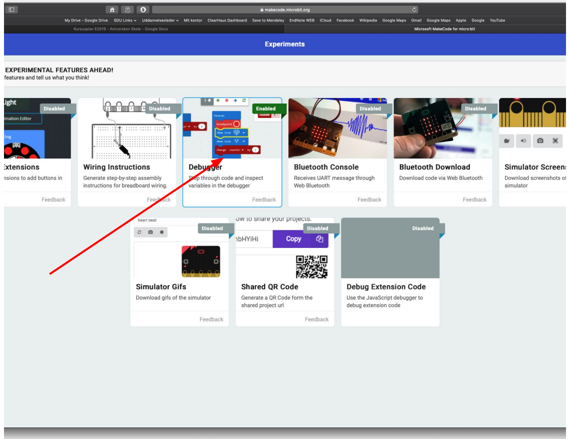


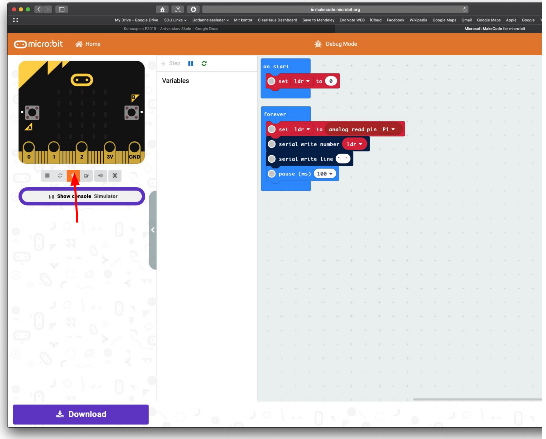
Methods for Debugging

Debugging in MakeCode



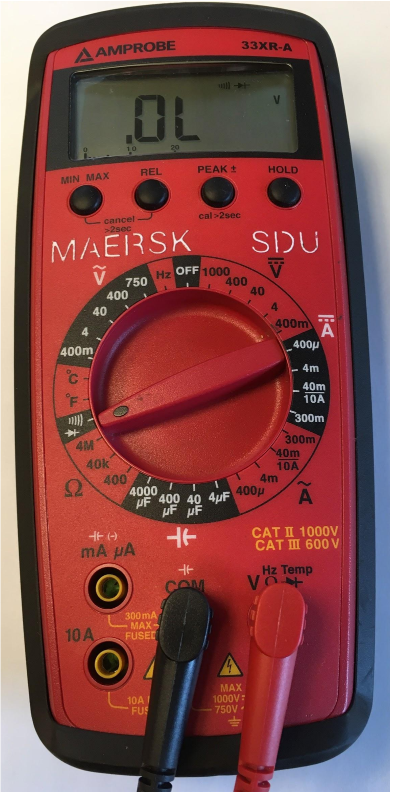
- Can be used to step through the code - one step at a time - and see how the variables change
- **Activated under settings -> about**
- Works only with the simulator!



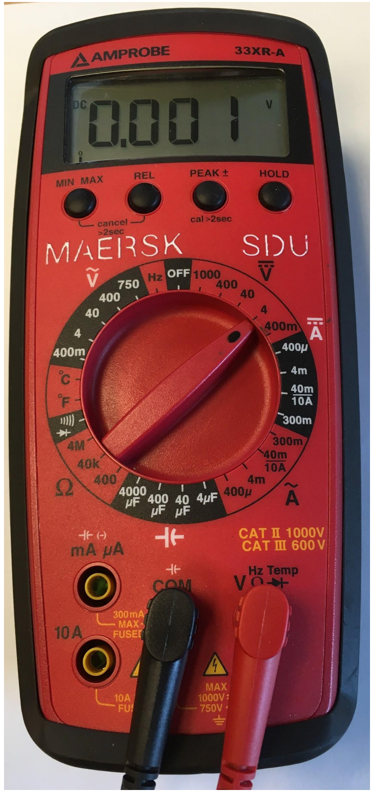


Multimeter options

Continuity Test

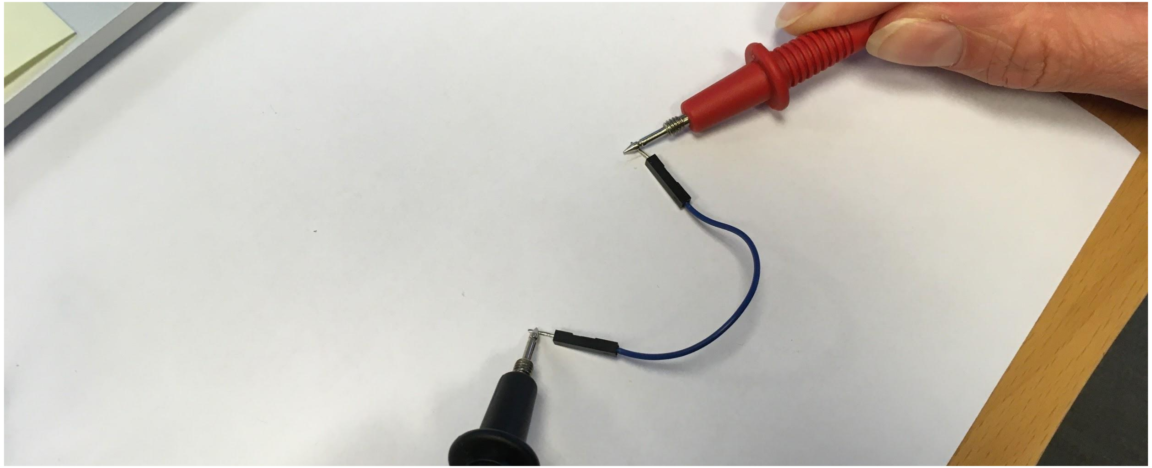


Voltage Measurement

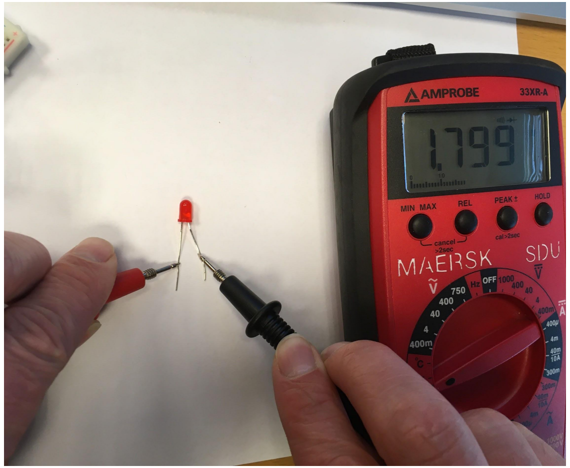


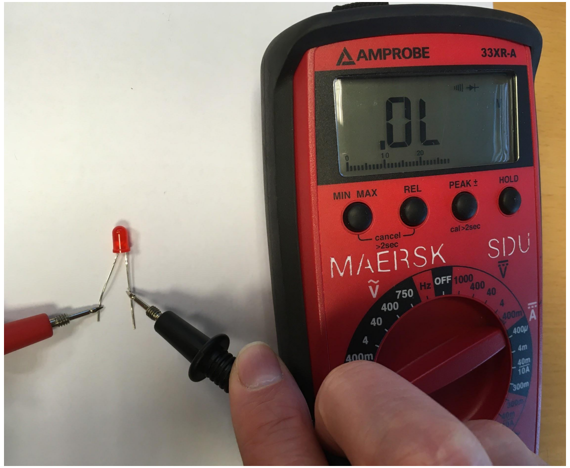
Current measurement

How to measure on a circuit - Continuity / Diode

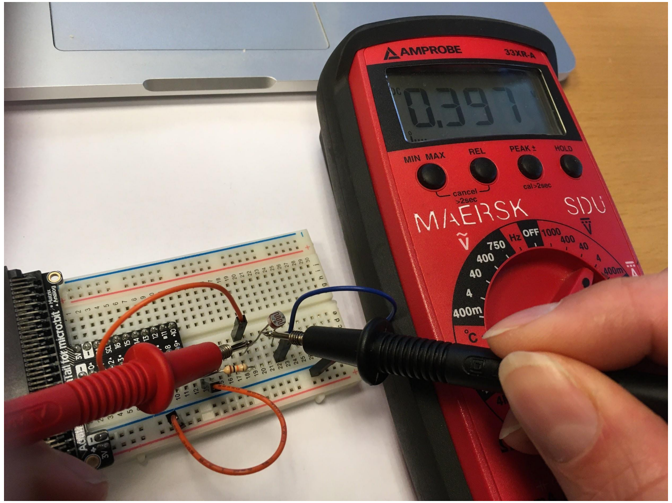


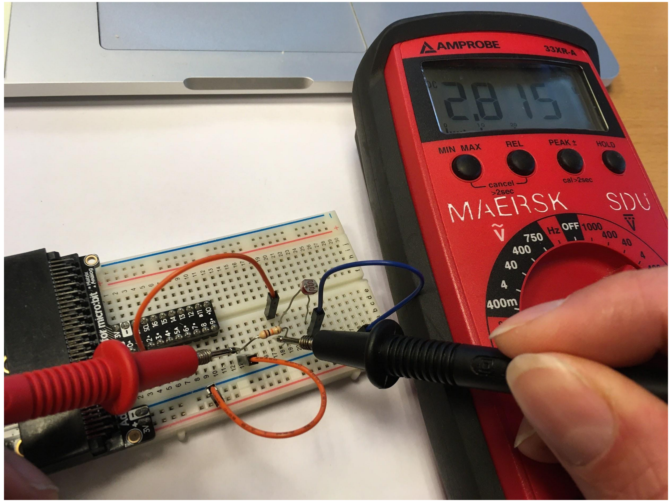
- Measures electrical connection
- You can also measure it directly at the circuit, if it is not connection to a voltage supply





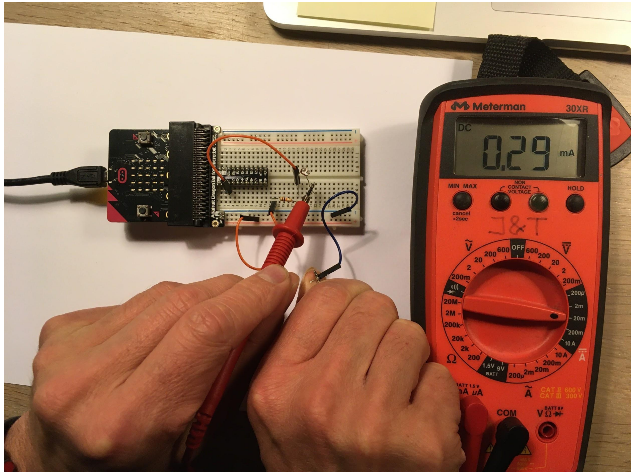
How to measure on a circuit - Voltage

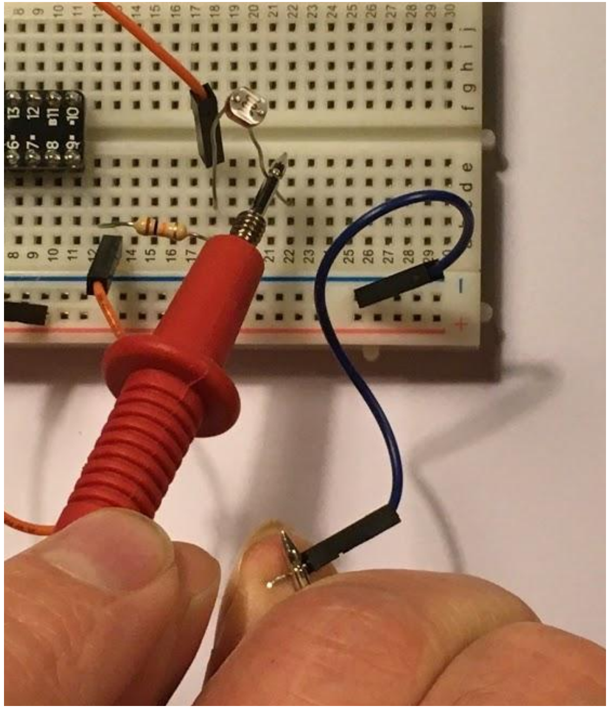




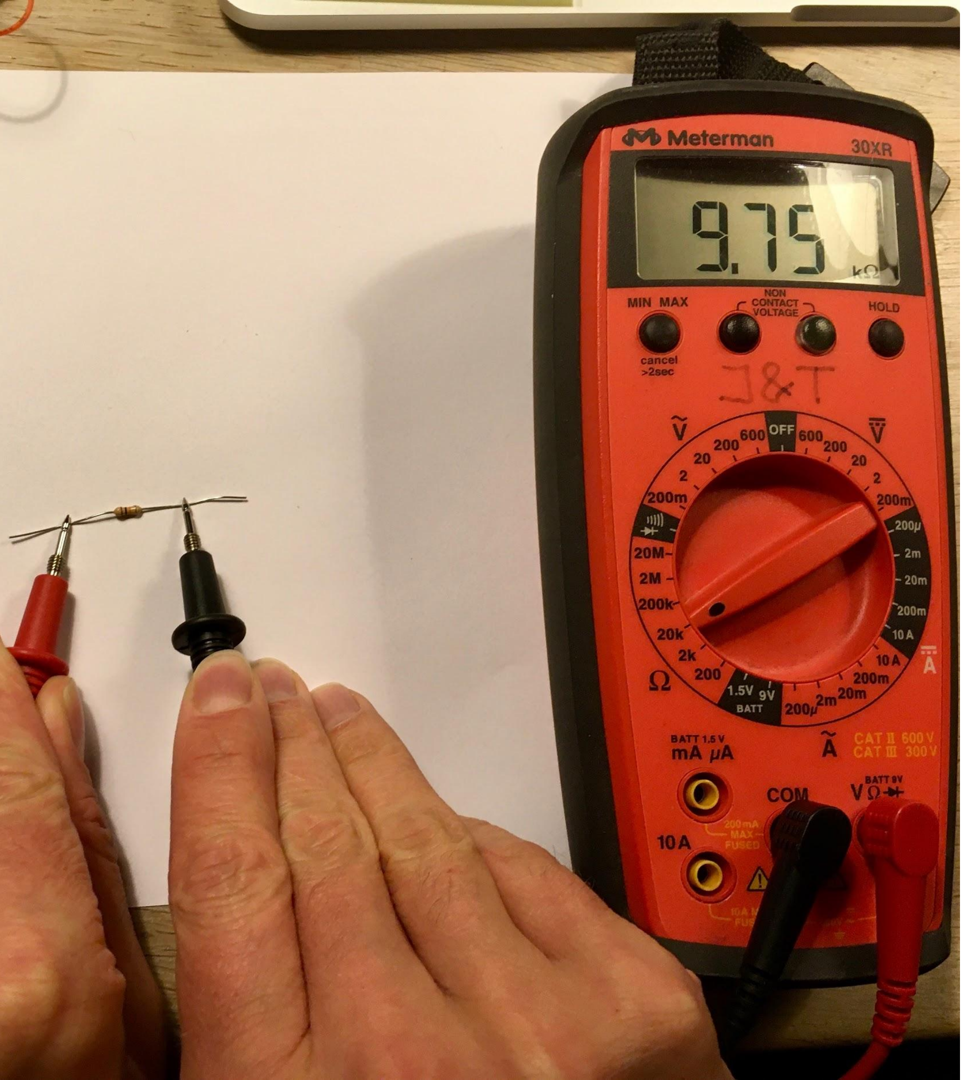
- Measured in parallel with components
- The circuit must, of course, be connected to a voltage supply
- Remember to set it to the correct voltage range

How to measure on a circuit - Current





- Measured in the circuit
    - You have to lift components
- Separate connection on multimeter - see previous slide.
- Remember to set to the correct current range - often between 2mA and 200mA



How to check resistance values

- Set the multimeter in the **Ω** range to the value that is within the range the resistance should be.
- Ex. If you expect the resistance to be at 5kOhm then put the multimeter in the range of 20k.
- Remember that the resistor must be removed from the circuit when it is being measured - otherwise you may be measuring other components as well.

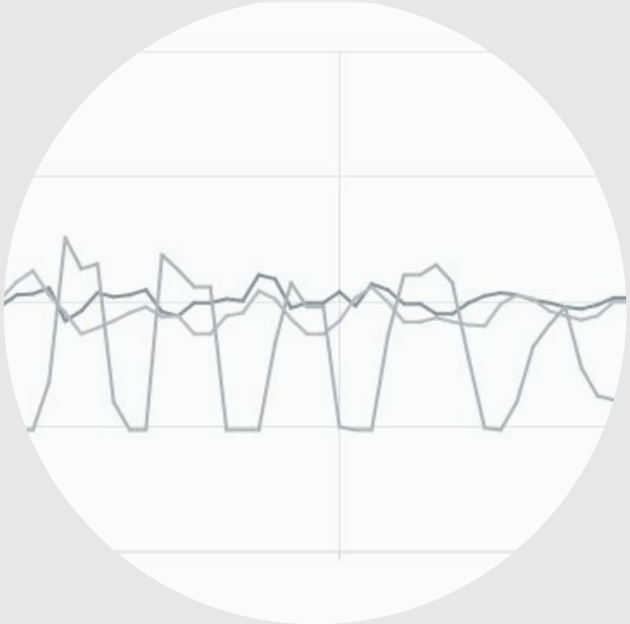
Debugging in pairs (30 min)

- You each make conscious mistakes in a setup, that you also know how to build a working model of
  - code, breadboard or both
- You briefly describe to each other what it was intended to do
  - What should the code do?
  - What should the circuit do?
  - If it is a circuit, then make a circuit diagram showing the circuit
- The you debug each others circuits
  - Try to find all the mistakes you can and write them down on paper so you can evaluate later
  - Try if necessary, to reconstruct the circuit diagram from what is built on the breadboard
- You may change / make new mistakes several times

Measuring Exercises

- Build the following circuit and check and record the values for current, voltage and resistance
- Draws on whiteboard / smartboard

Use of the micro:bit for data logging

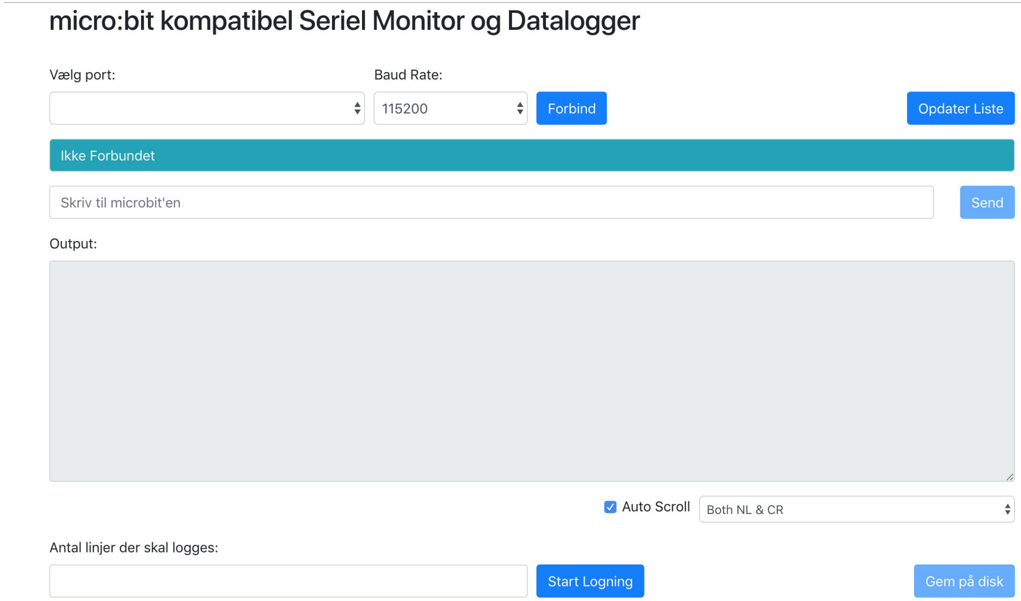


Data logging software

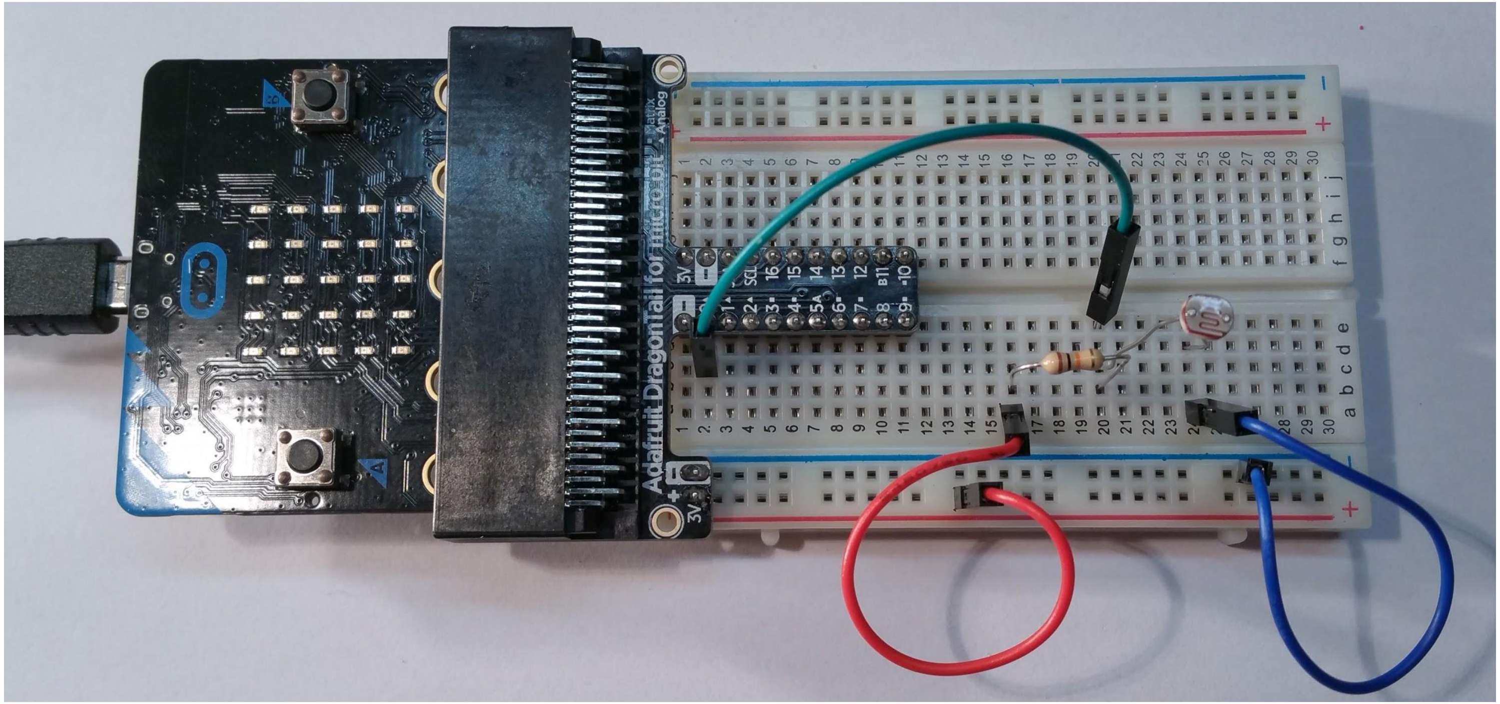
Link to software for Mac and Windows:

https://teknologihuset.dk/serial-link-til-microbit/

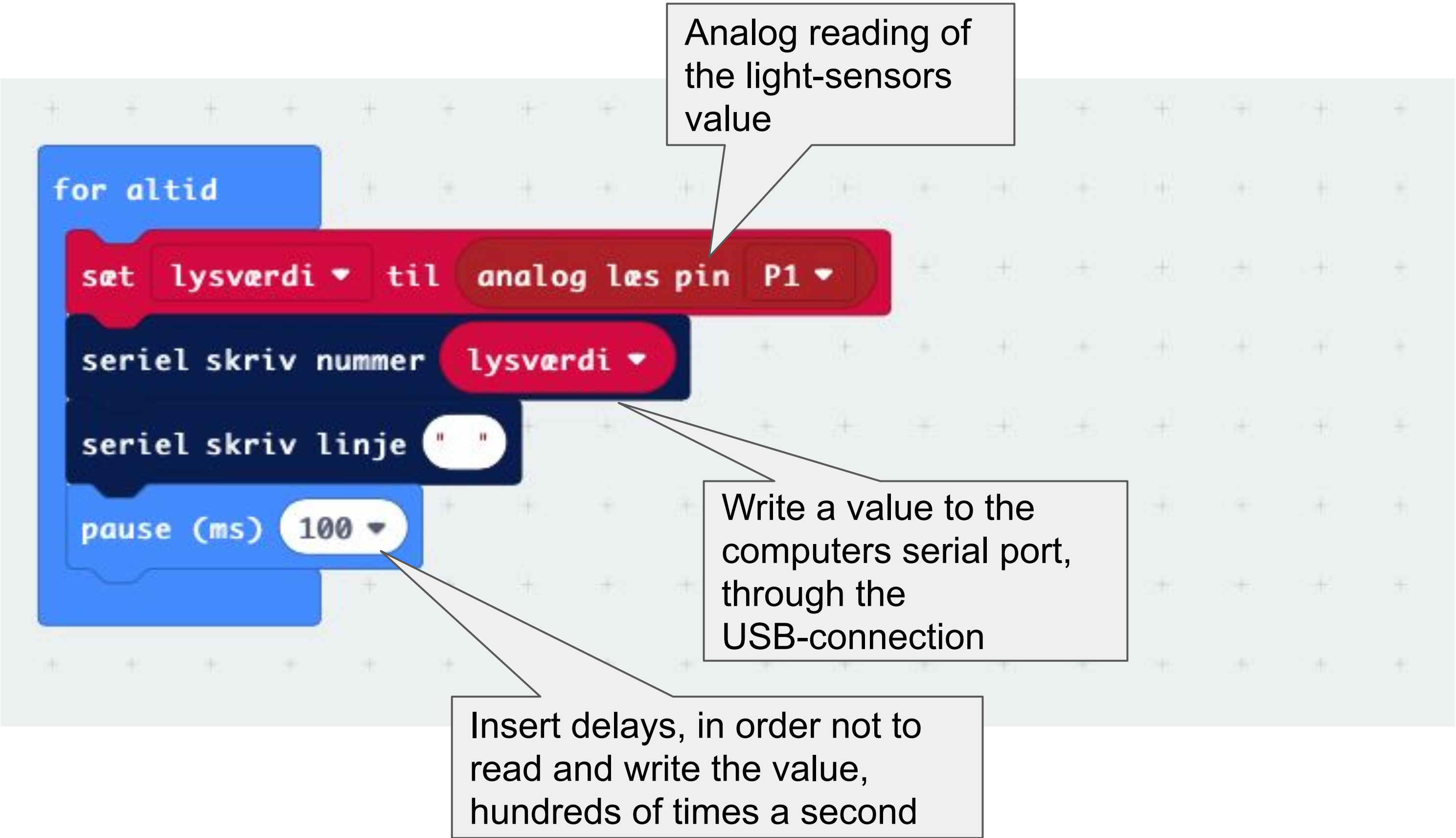
Program demo ...



Light-sensor



micro:bit program



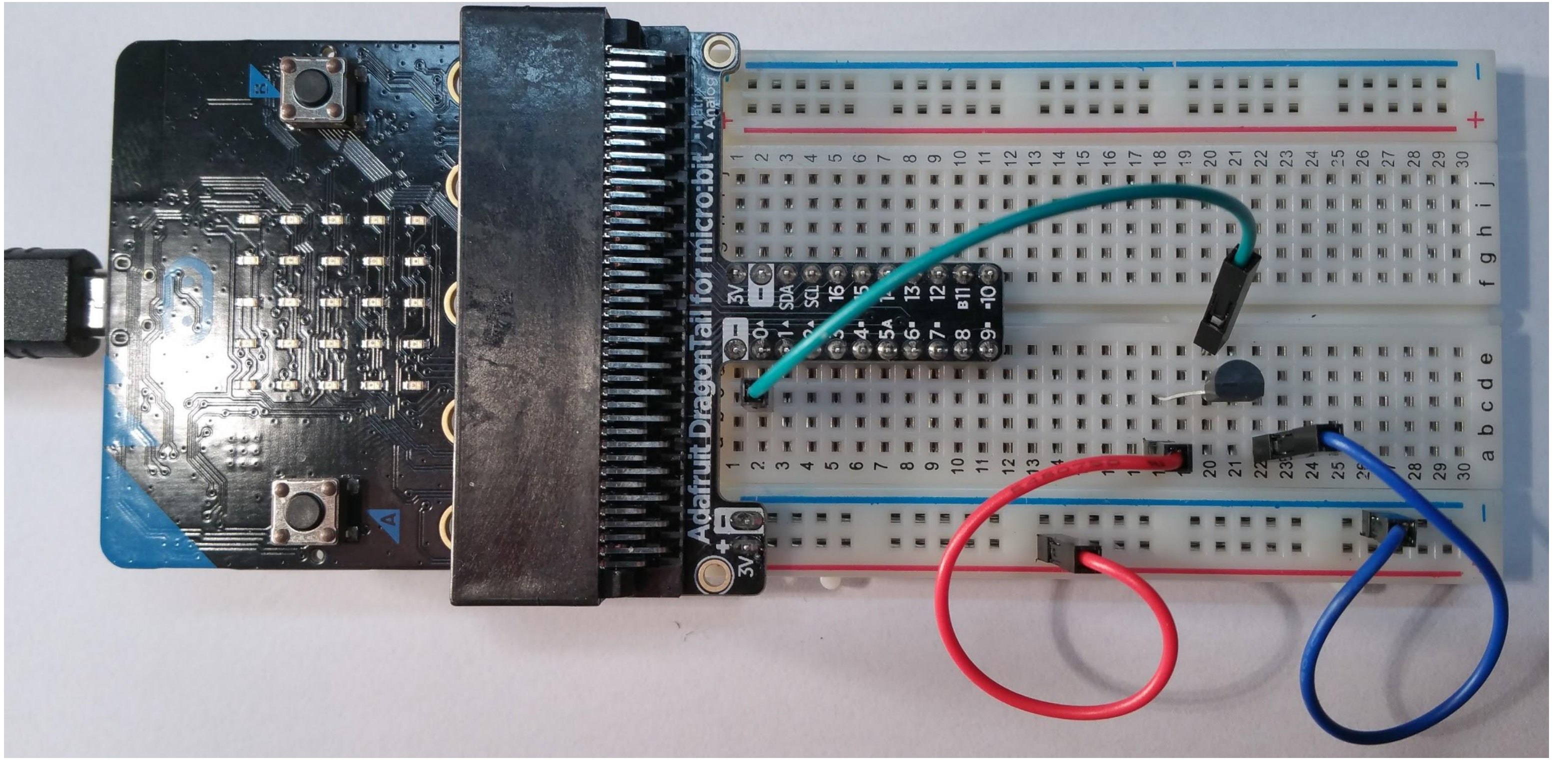
Evaluation

Is the assignment too easy/hard?

Would your pupils be able to use the diagram to build the circuit?

Is the explanation of the theory understandable?

Temperature - MCP9700



Temperature sensor

Conversion from Analog to Digital

$$V_{in} = \left(\frac{3.2V}{1024} \cdot ADC\right) \cdot 1000\,[mV]$$

$$T = \frac{V_{in} - 500}{10}\,[^{\circ}C]$$

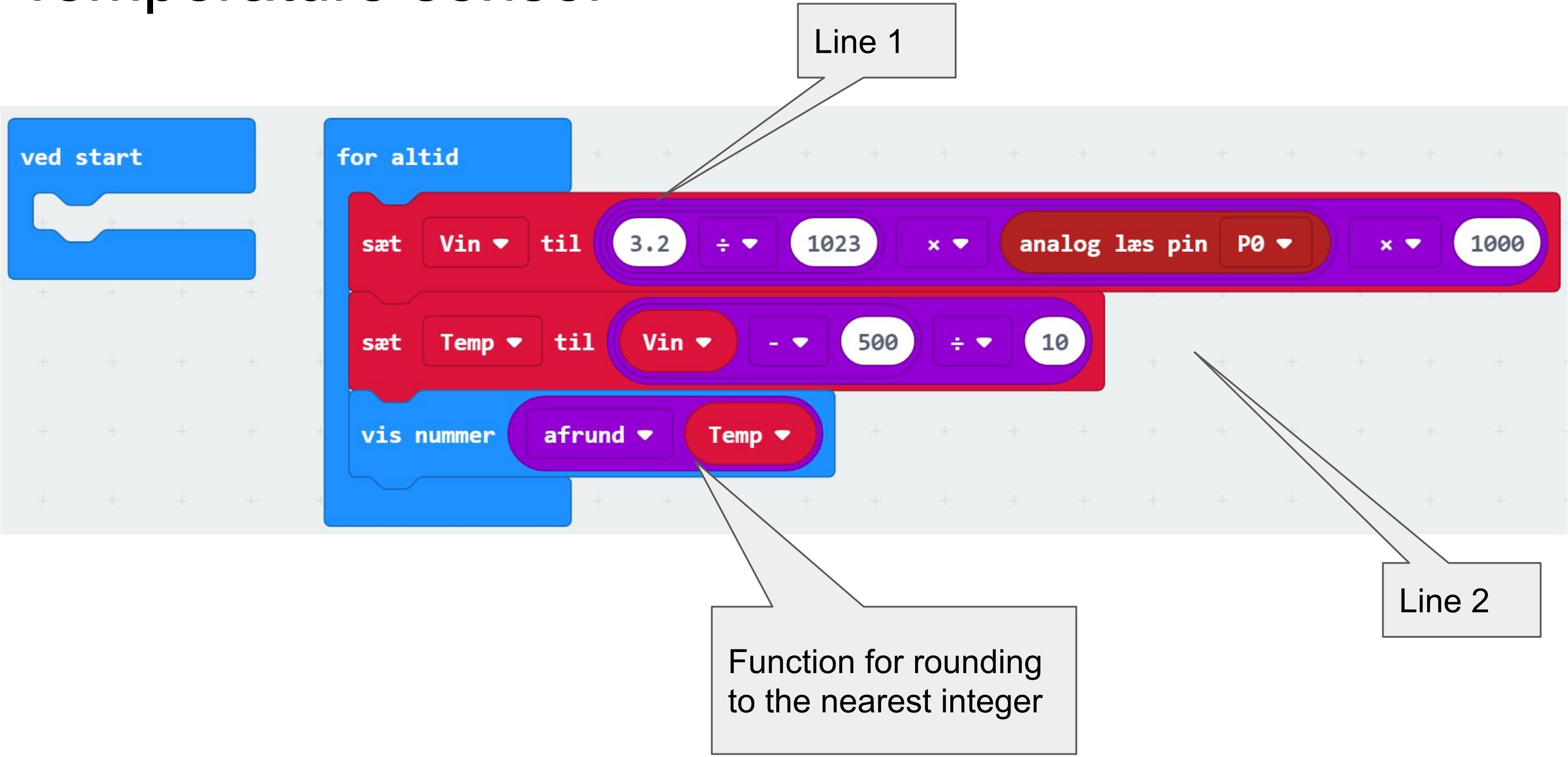
When using the external temperature sensor, we must do the math behind the conversion from a binary value to a decimal that gives the temperature.

The two formulas do this.
Line 1) is the conversion from binary to a voltage in [mV] (millivolts).
Line 2) is the conversion from voltage to temperature.

When an analog voltage from e.g. a temperature sensor must be converted to a digital (binary) value, an A / D converter is used. This could, as with a microbit, be a 10-bit converter. This means that it will give a binary value between 0 and 1023 for a voltage between 0V and 3.2V on pin A0 (microbit). The "ADC" in line 1 is the binary value from our A / D converter in the microbit. In order to convert the binary value to a voltage, we must first find out how many volts there are per stage, of which (3.2V / 1024). Then we multiply with ADC (number of steps we have). Hereby we get the voltage in [V] and then we multiply by 1000 to get it in [mV]

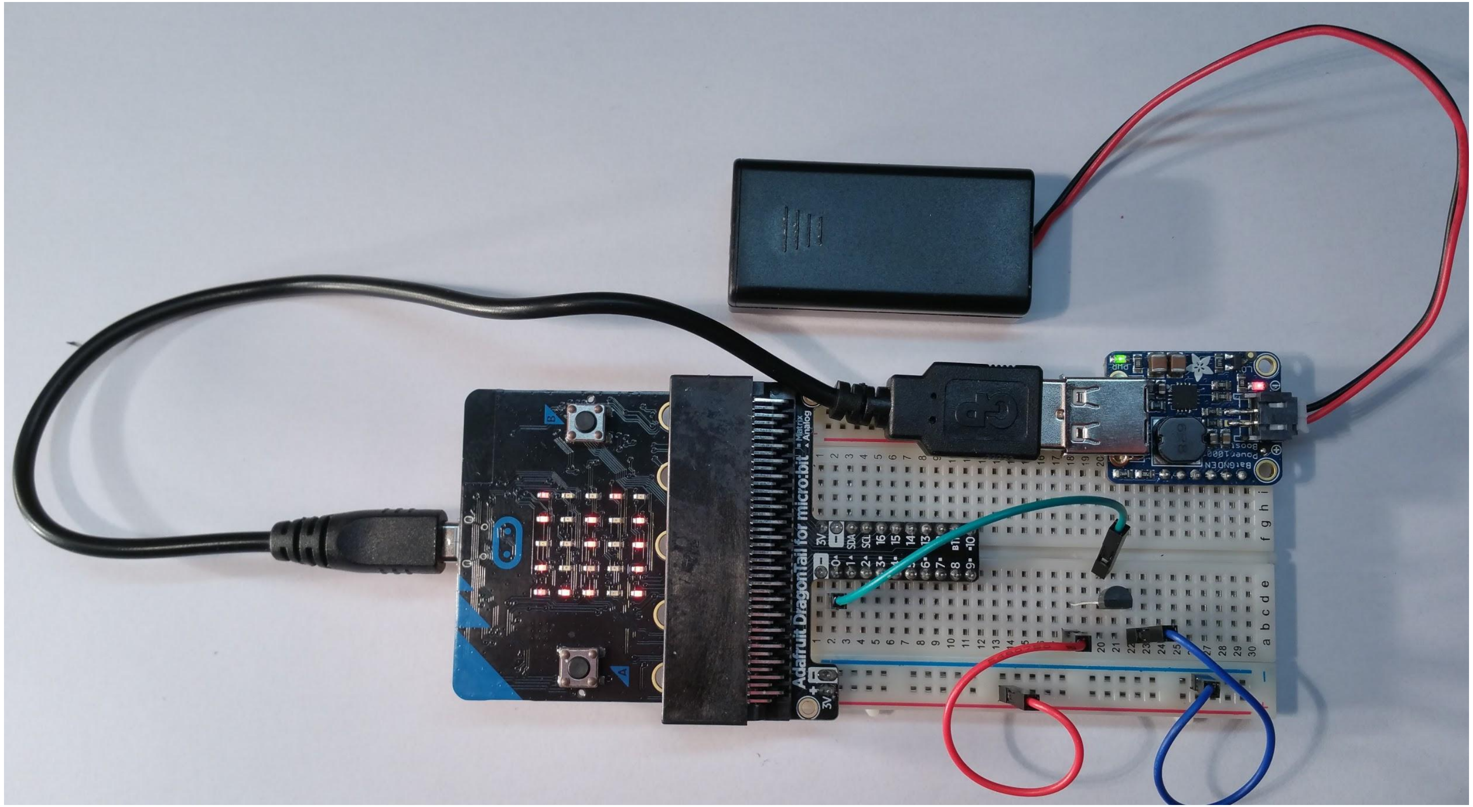
In line 2, we need to convert from a voltage to a temperature. This formula comes directly from the temperature sensor datasheet. When Vin is inserted into [mV] you get the temperature in Celcius.

Temperature sensor



Was there anything which left you wondering?

Use the power booster



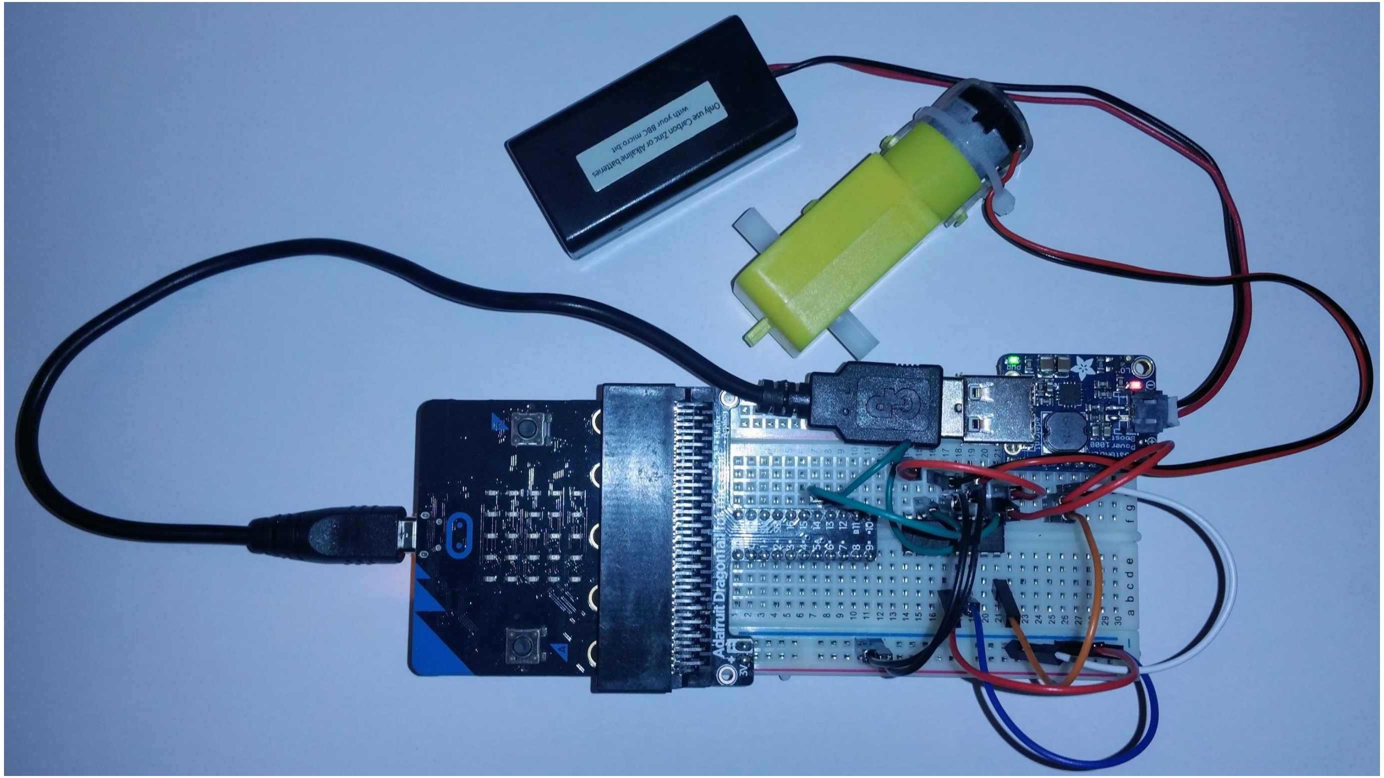
Evaluation

Is the assignment too easy/hard?

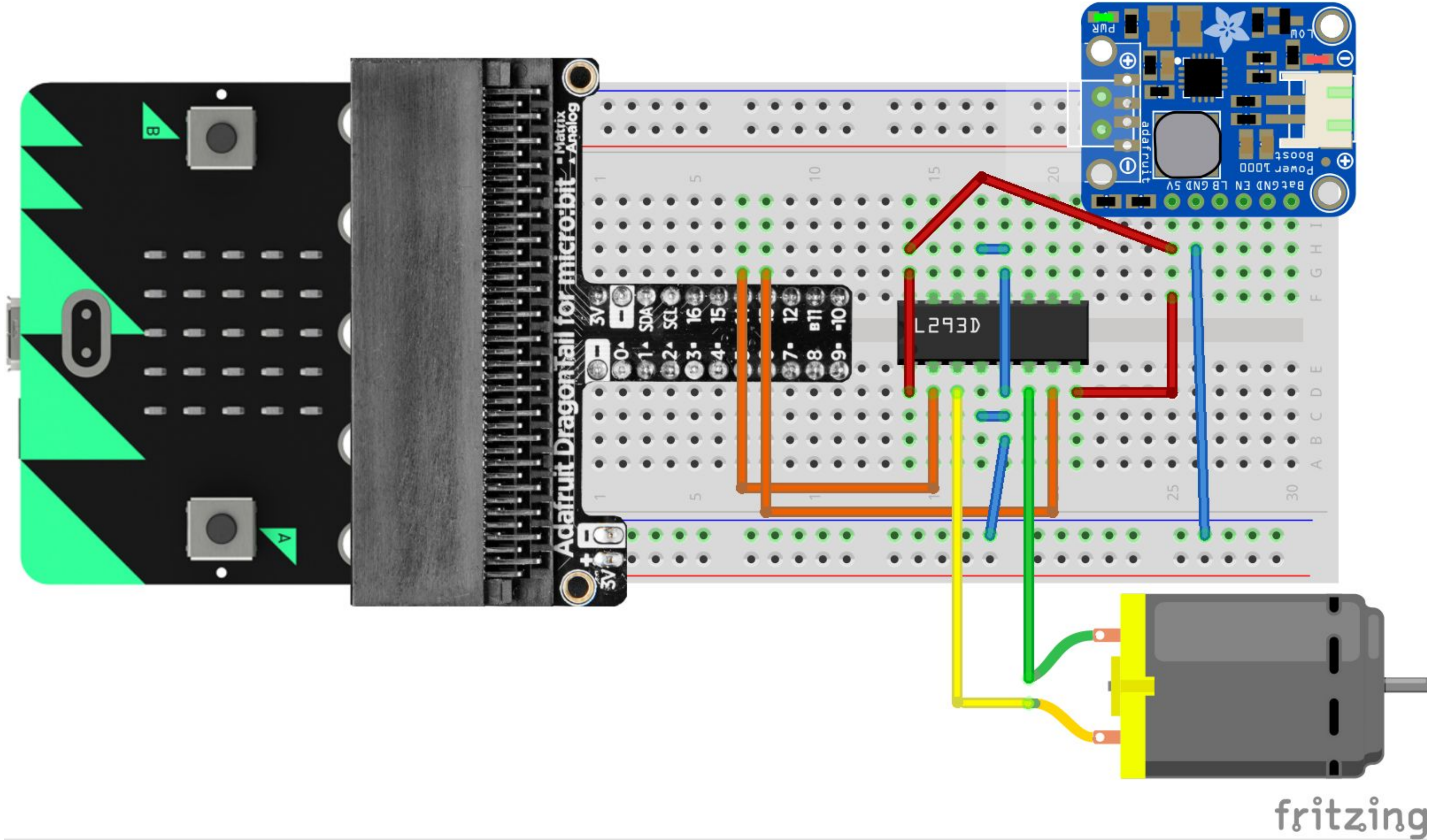
Would your pupils be able to use the diagram to build the circuit?

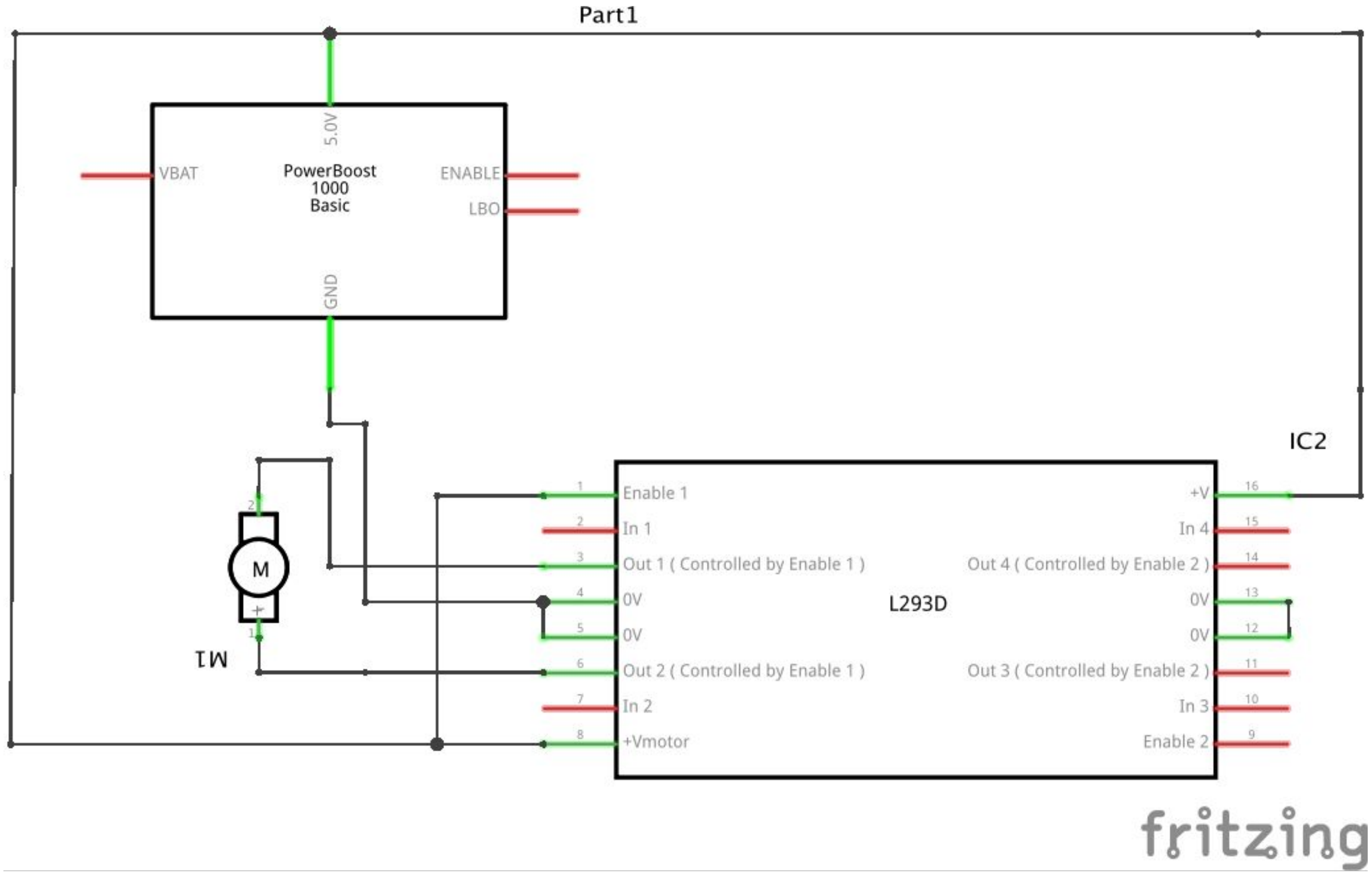
Is the explanation of the theory understandable?

Motor control



Diagram

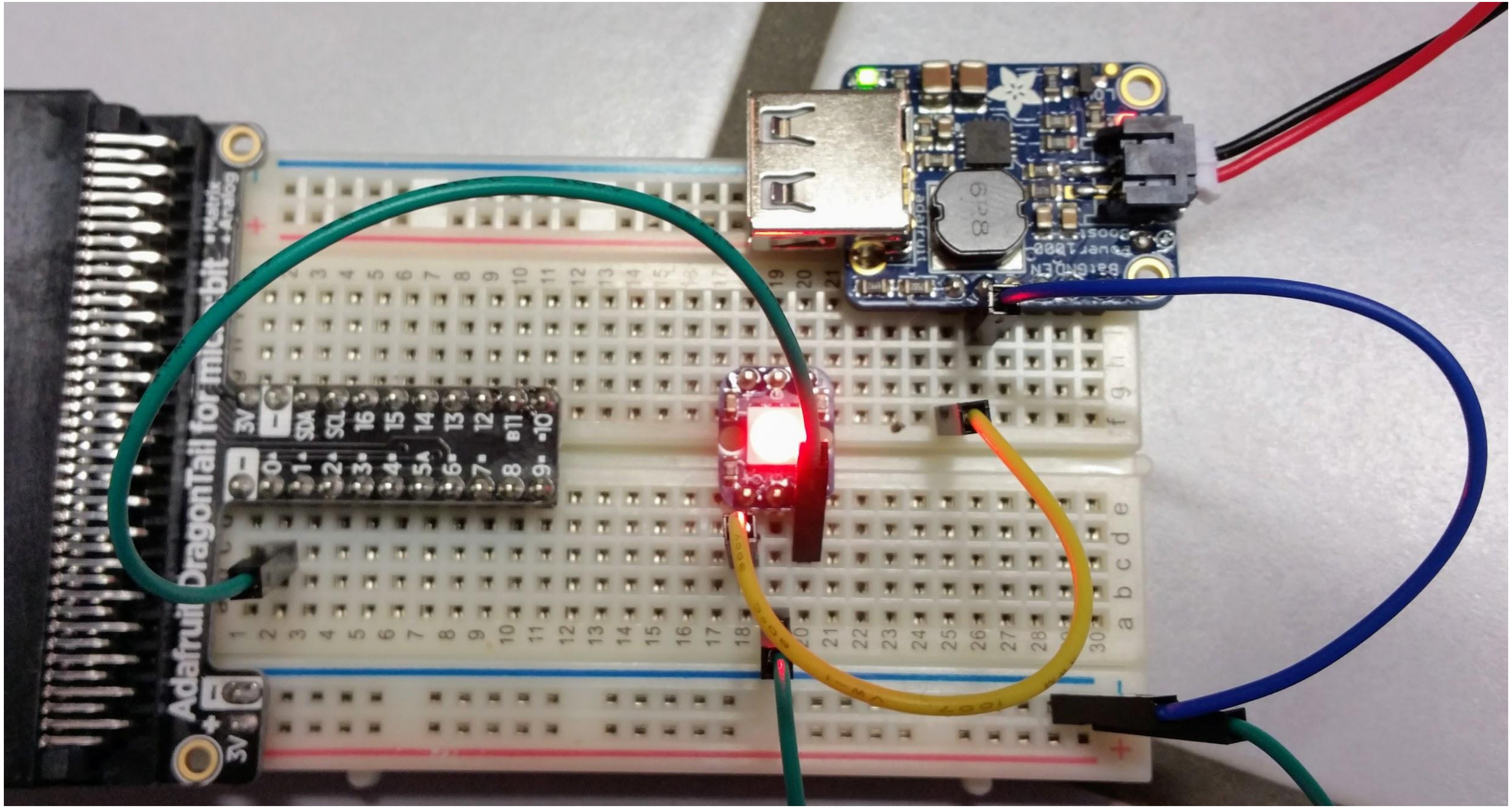

Adafruit DragonTail for micro:bit
L293D
PowerBoost 1000
fritzing


Part1
5.0V
VBAT
PowerBoost
1000
Basic
ENABLE
LBO
GND
IC2
M
M1
Enable 1
In 1
Out 1 (Controlled by Enable 1)
0V
0V
Out 2 (Controlled by Enable 1)
In 2
+Vmotor
L293D
+V
In 4
Out 4 (Controlled by Enable 2)
0V
0V
Out 3 (Controlled by Enable 2)
In 3
Enable 2
fritzing

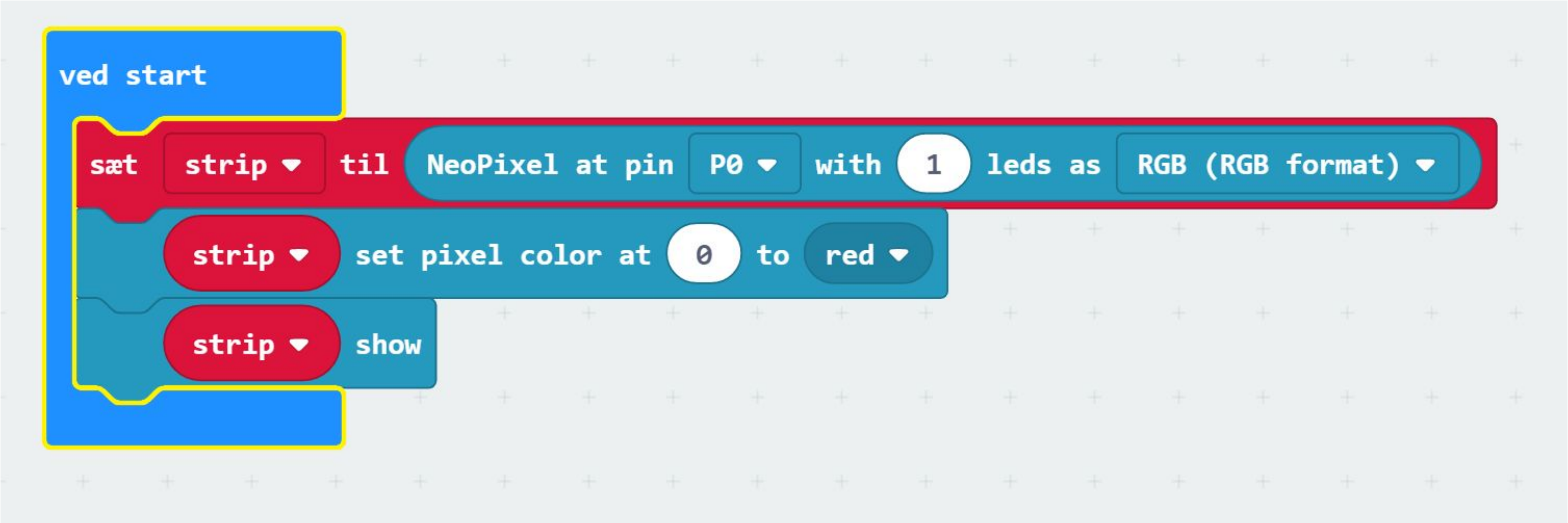
Code

ved start
digital skriv pin P13 til 0
digital skriv pin P14 til 0
når der trykkes på knappen A+B
digital skriv pin P13 til 0
digital skriv pin P14 til 0
når der trykkes på knappen B
digital skriv pin P13 til 1
digital skriv pin P14 til 0
når der trykkes på knappen A
digital skriv pin P13 til 0
digital skriv pin P14 til 1

NeoPixels



Code


ved start
sæt strip til NeoPixel at pin P0 with 1 leds as RGB (RGB format)
strip set pixel color at 0 to red
strip show