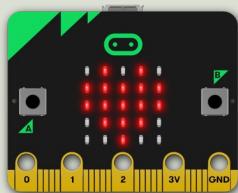


<https://www.halvorsen.blog>



# micro:bit and Thermistor 10K Temperature Sensor



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 a Thermistor 10K Temperature Sensor

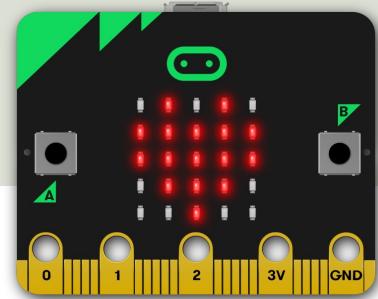


# Introduction to micro:bit

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

Mu 1.1.1 - untitled

Mode New Load Save Flash Files REPL Plotter Zoom-in Zoom-out Theme Check Tidy Help Quit

untitled

```
1 # Write your code here :-)
2
```

Select Mode

Please select the desired mode then click "OK". Otherwise, click "Cancel".

- BBC micro:bit  
Write MicroPython for the BBC micro:bit.
- CircuitPython  
Write code for boards running CircuitPython.
- ESP MicroPython  
Write MicroPython on ESP8266/ESP32 boards.
- Lego MicroPython  
Write MicroPython directly on Lego Spike devices.
- Pyboard MicroPython  
Use MicroPython on the Pyboard line of boards.
- Pygame Zero  
Make games with Pygame\_Zero

Change mode at any time by clicking the "Mode" button containing Mu's logo.

OK Cancel

BBC micro:bit

<https://www.halvorsen.blog>



# 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 *\n\ncurrentTemp = temperature()
```

This examples displays the temperature on the LED matrix:

```
from microbit import *\n\nwhile True:\n    if button_a.was_pressed():\n        display.scroll(temperature())
```

# Temperature Sensor

Mu 1.1.1 - temp\_ex.py

The screenshot shows the Mu code editor interface. The title bar says "Mu 1.1.1 - temp\_ex.py". The menu bar includes "Mode", "New", "Load", "Save", "Flash", "Files", "REPL", "Plotter", "Zoom-in", "Zoom-out", "Theme", "Check", "Tidy", "Help", and "Quit". The main workspace contains the following Python code:

```
from microbit import *
while True:
    currentTemp = temperature()
    print(currentTemp)
    sleep(2000)
```

Below the code is the BBC micro:bit REPL window, which displays the number 28 multiple times.

```
from microbit import *
while True:
    currentTemp = temperature()
    print(currentTemp)
    sleep(2000)
```

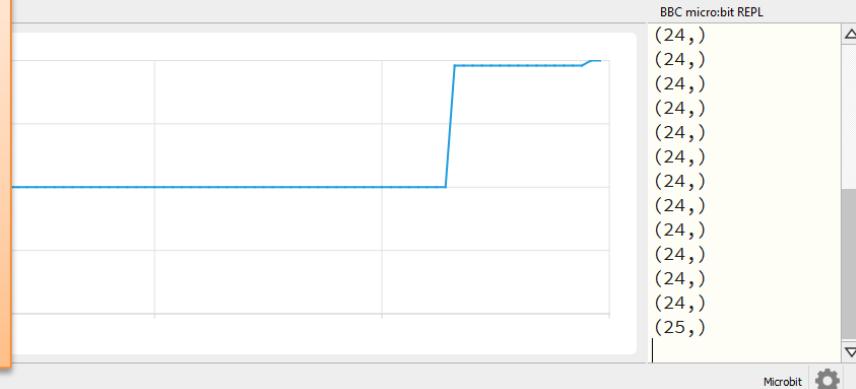
# Temperature Sensor

The screenshot shows the Mu 1.0.3 Python editor interface. The menu bar at the top includes 'File', 'Edit', 'Run', 'Terminal', 'Help', and 'About'. Below the menu is a toolbar with icons for Mode (micro:bit), New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Help, and Quit. The main workspace shows a script named 'temperature\_read.py' with the following code:

```
from microbit import *
while True:
    currentTemp = temperature()
    display.scroll(currentTemp)
    print((currentTemp,))
    sleep(1000)
```

```
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

<https://www.halvorsen.blog>

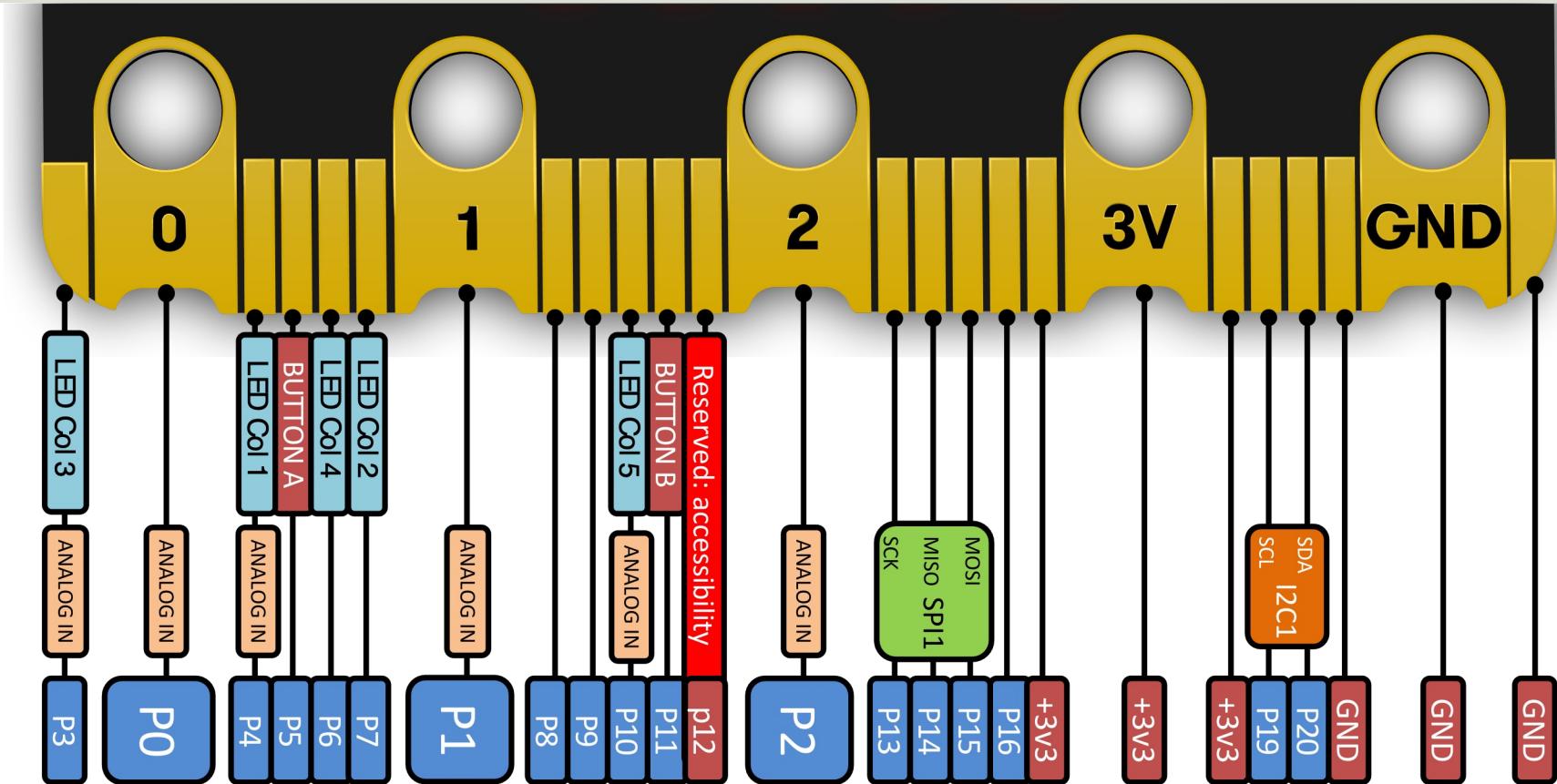


# micro:bit I/O Pins

Hans-Petter Halvorsen

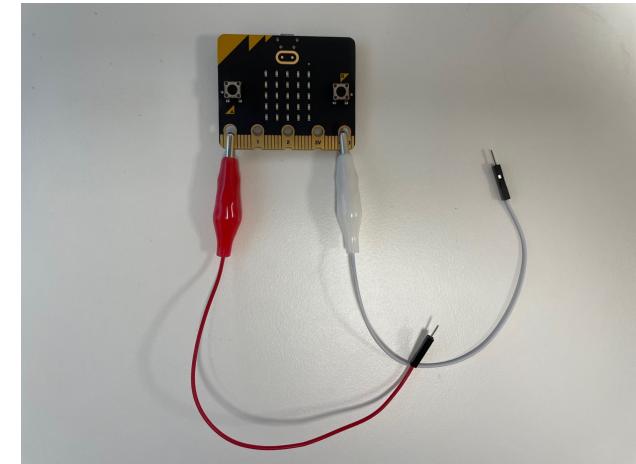
[Table of Contents](#)

# micro:bit I/O Pin Overview



# I/O Pins

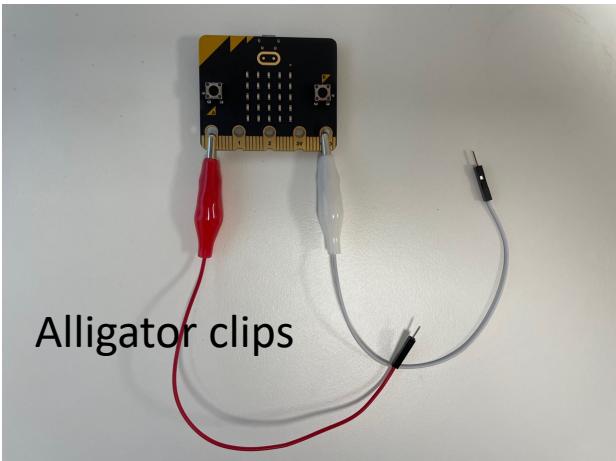
- We use the I/O pins to connect external components like LEDs, different types of Sensors, etc.
- You can use 4mm Banana plugs or Alligator/Crocodile clips
- Typically, you also want to use a Breadboard



# Component Examples



Breadboard

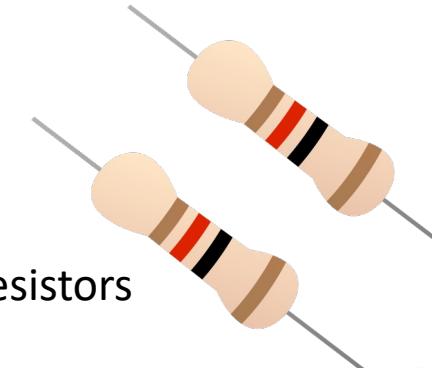


Alligator clips

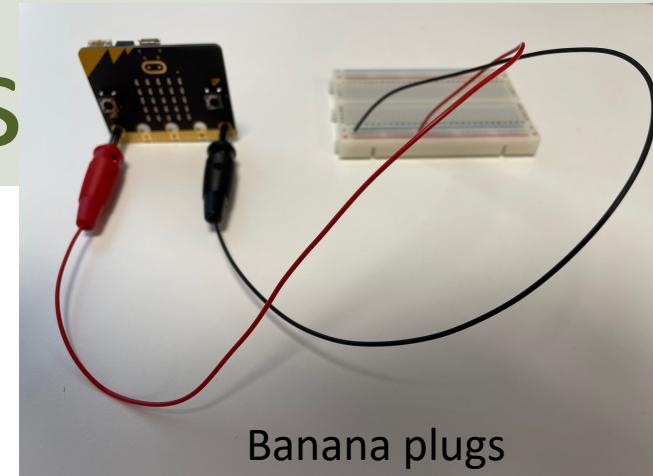
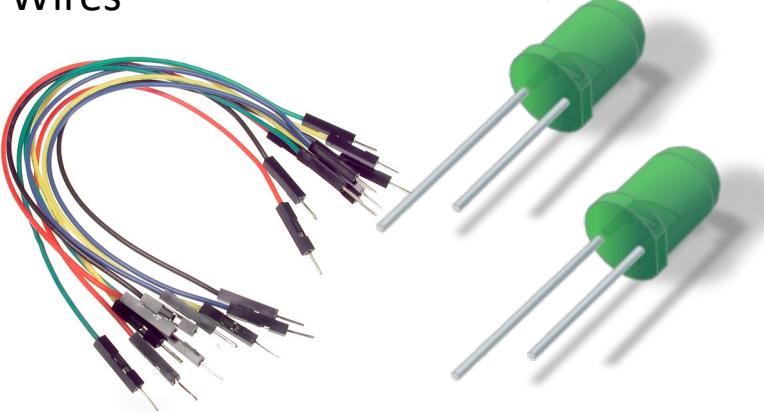
Temperature Sensor



Resistors



Wires



Banana plugs

LEDs



Multimeter



# Types of I/O Pins

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

We will only use an Analog Input pin in this Tutorial

<https://www.halvorsen.blog>



# Thermistor 10K Temperature Sensor

Hans-Petter Halvorsen

[Table of Contents](#)

# Thermistor



A thermistor is an electronic component that changes resistance to temperature - so-called Resistance Temperature Detectors (RTD). It is often used as a temperature sensor.



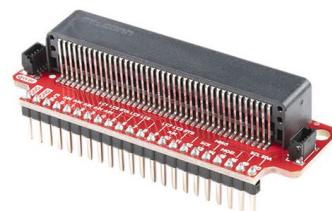
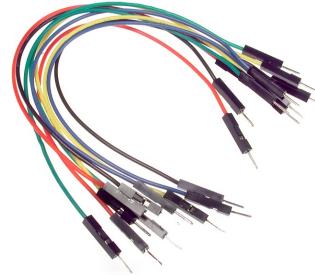
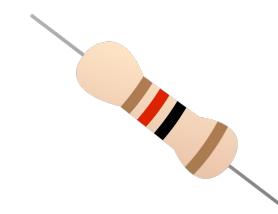
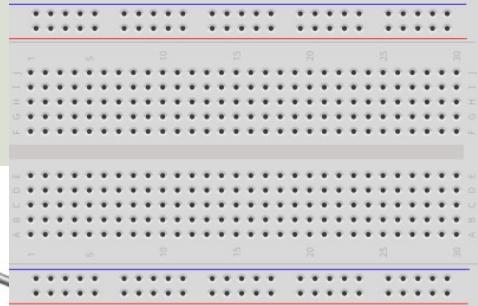
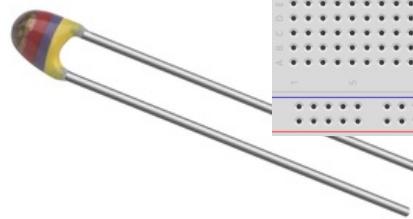
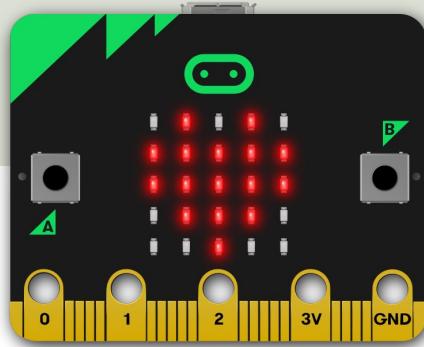
Our Thermistor is a so-called NTC (Negative Temperature Coefficient). In a NTC Thermistor, resistance decreases as the temperature rises.

There is a **non-linear relationship** between resistance and excitement. To find the temperature we can use the following equation (**Steinhart-Hart equation**):

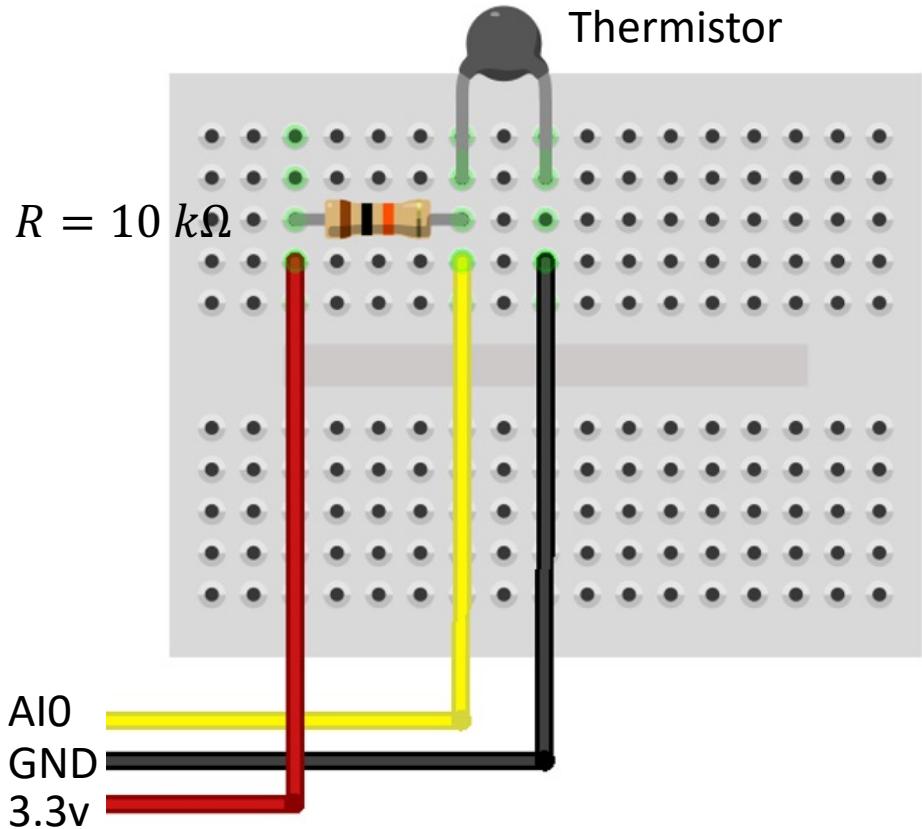
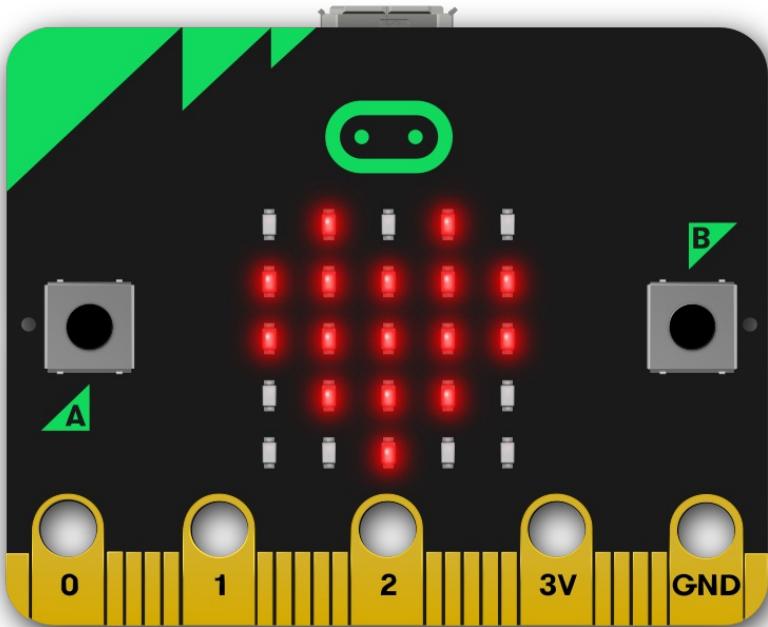
$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3$$
 where  $A, B, C$  are constants given below  
[Wikipedia]  
 $A = 0.001129148, B = 0.000234125$  and  $C = 8.76741E - 08$

# Hardware

- micro:bit
- Breadboard
- Thermistor 10K (Temperature Sensor)
- Wires (Jumper Wires)
- Resistor 10 kΩ
- We can also use an **Adapter Breakout Board for micro:bit** instead of Alligator/Crocodile clips

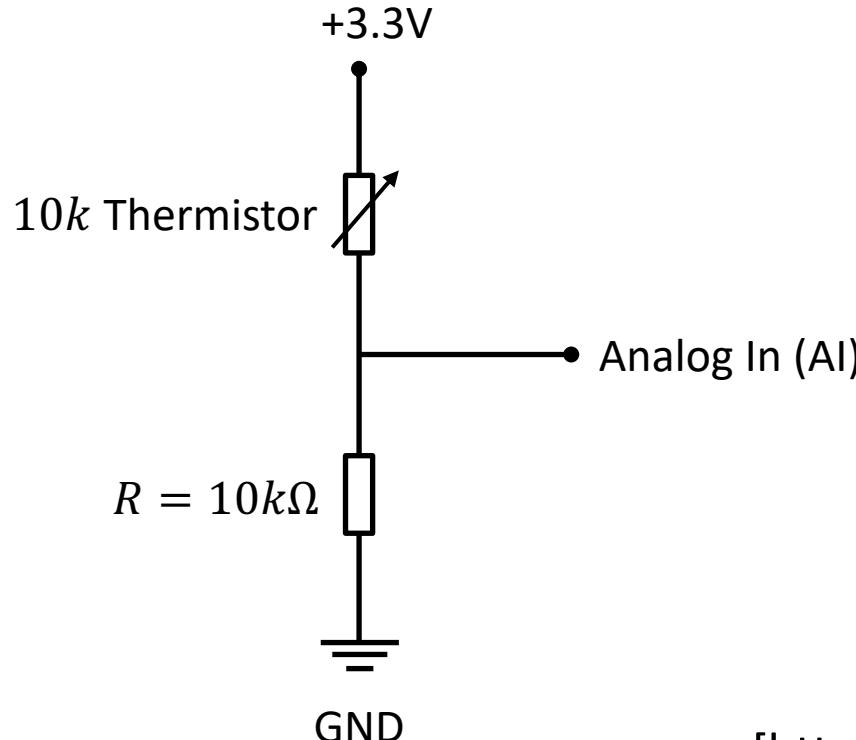


# Wiring

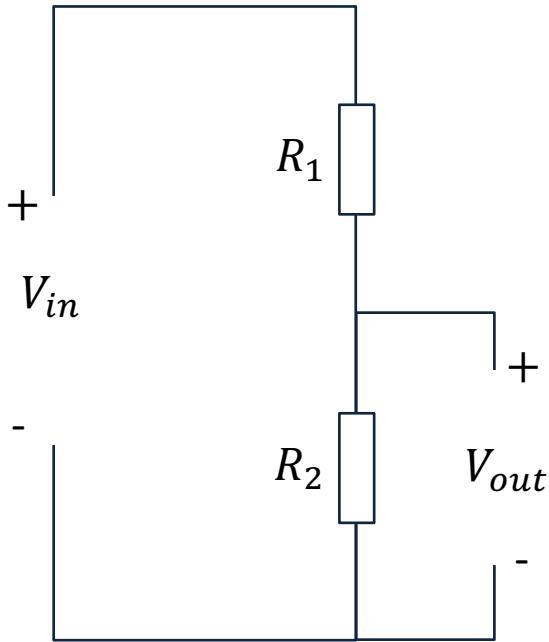


# Voltage Divider

The wiring is called a “Voltage divider”:



# General Voltage Divider



We want to find  $V_{out}$

Formula:

$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2}$$

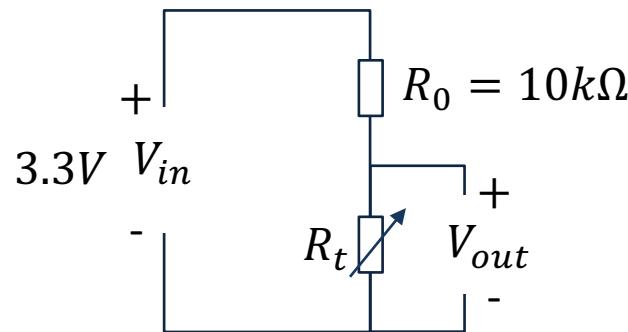
# Voltage Divider for our System

Voltage Divider Equation:

$$V_{out} = V_{in} \frac{R_t}{R_0 + R_t}$$

We want to find  $R_t$ :

$$R_t = \frac{V_{out}R_0}{V_{in} - V_{out}}$$



$R_t$  - 10k Thermistor. This varies with temperature. From Datasheet we know that  $R_t = 10k\Omega @ 25^\circ\text{C}$

Steps:

1. We wire the circuit on the Breadboard and connect it to the DAQ device
2. We measure  $V_{out}$  using the DAQ device
3. We calculate  $R_t$  using the Voltage Divider equation
4. Finally, we use Steinhart-Hart equation for finding the Temperature

# Steinhart-Hart Equation

To find the Temperature we can use Steinhart-Hart Equation:

$$\frac{1}{T_K} = A + B \ln(R) + C(\ln(R))^3$$

This gives:

$$T_K = \frac{1}{A + B \ln(R) + C(\ln(R))^3}$$

Where the Temperature  $T_K$  is in Kelvin

$A, B$  and  $C$  are constants

$$\begin{aligned} A &= 0.001129148 \\ B &= 0.000234125 \\ C &= 0.0000000876741 \end{aligned}$$

The Temperature in degrees Celsius will then be:

$$T_C = T_K - 273.15$$

# Pseudo Code

1. Get  $V_{out}$  from the DAQ device (Arduino UNO)
2. Calculate  $R_t = \frac{V_{out}R_0}{V_{in}-V_{out}}$
3. Calculate  $T_K = \frac{1}{A+B \ln(R_t)+C(\ln(R_t))^3}$
4. Calculate  $T_C = T_K - 273.15$
5. Present  $T_C$  in the User Interface

# Pseudo Code

```
float Vin = 3.3;  
float Ro=10000;  
float Rt = (Vout*Ro)/(Vin-Vout);
```

```
//Steinhart constants  
float A = 0.001129148;  
float B = 0.000234125;  
float C = 0.0000000876741;
```

```
//Steinhart-Hart Equation  
float TempK = 1 / (A + (B * ln(Rt)) + (C * ln(Rt)**3));
```

```
//Convert from Kelvin to Celsius  
float TempC = TempK - 273.15;
```

Mu 1.1.1 - thermistor\_ex.py

The Mu IDE interface shows a script window titled "thermistor\_ex.py". The window contains Python code for reading analog data from a micro:bit's pin 0, calculating the resistance of a thermistor using the Steinhart-Hart equation, and then converting that resistance to Celsius temperature. The code also prints the temperature to the BBC micro:bit REPL and scrolls it onto the micro:bit's display. The REPL window at the bottom shows the output "24.6" repeated three times.

```
from microbit import *
import math

# Voltage Divider
Vin = 3.3
Ro = 10000 # 10k Resistor

# Steinhart Constants
A = 0.001129148
B = 0.000234125
C = 0.0000000876741

while True:
    adc = pin0.read_analog()
    Vout = (3.3/1023)*adc

    # Calculate Resistance
    Rt = (Vout * Ro) / (Vin - Vout)

    # Steinhart - Hart Equation
    TempK = 1 / (A + (B * math.log(Rt)) + C * math.pow(math.log(Rt), 3))

    # Convert from Kelvin to Celsius
    TempC = TempK - 273.15

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

    sleep(5000)
```

BBC micro:bit REPL

```
24.6
24.6
24.6
```

# Python

The Code works as follows:

1. Get  $V_{out}$  from the DAQ device
2. Calculate  $R_t = \frac{V_{out}R_0}{V_{in}-V_{out}}$
3. Calculate  $T_K = \frac{1}{A+B \ln(R_t)+C(\ln(R_t))^3}$
4. Calculate  $T_C = T_K - 273.15$
5. Present  $T_C$  in the User Interface

```
from microbit import *
import math

# Voltage Divider
Vin = 3.3
Ro = 10000 # 10k Resistor

# Steinhart Constants
A = 0.001129148
B = 0.000234125
C = 0.0000000876741

while True:
    adc = pin0.read_analog()
    Vout = (3.3/1023)*adc

    # Calculate Resistance
    Rt = (Vout * Ro) / (Vin - Vout)
    # Rt = 10000 # Used for Testing. Setting Rt=10k should give TempC=25

    # Steinhart - Hart Equation
    TempK = 1 / (A + (B * math.log(Rt)) + C * math.pow(math.log(Rt), 3))

    # Convert from Kelvin to Celsius
    TempC = TempK - 273.15

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

    sleep(5000)
```

# Python

Here, I have made a separate Python function for the thermistor logic. This makes it easy to use this part in several Applications.

```
import math

def thermistorTemp(Vout):
    # Voltage Divider
    Vin = 3.3
    Ro = 10000 # 10k Resistor

    # Steinhart Constants
    A = 0.001129148
    B = 0.000234125
    C = 0.0000000876741

    # Calculate Resistance
    Rt = (Vout * Ro) / (Vin - Vout)

    # Steinhart - Hart Equation
    TempK = 1 / (A + (B * math.log(Rt)) + C * math.pow(math.log(Rt), 3))

    # Convert from Kelvin to Celsius
    TempC = TempK - 273.15

return TempC
```

thermistor.py

## Thermistor Application:

```
from microbit import *
import thermistor

while True:
    adc = pin0.read_analog()
    Vout = (3.3/1023)*adc

    TempC = thermistor.thermistorTemp(Vout)

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

    sleep(5000)
```

<https://www.halvorsen.blog>

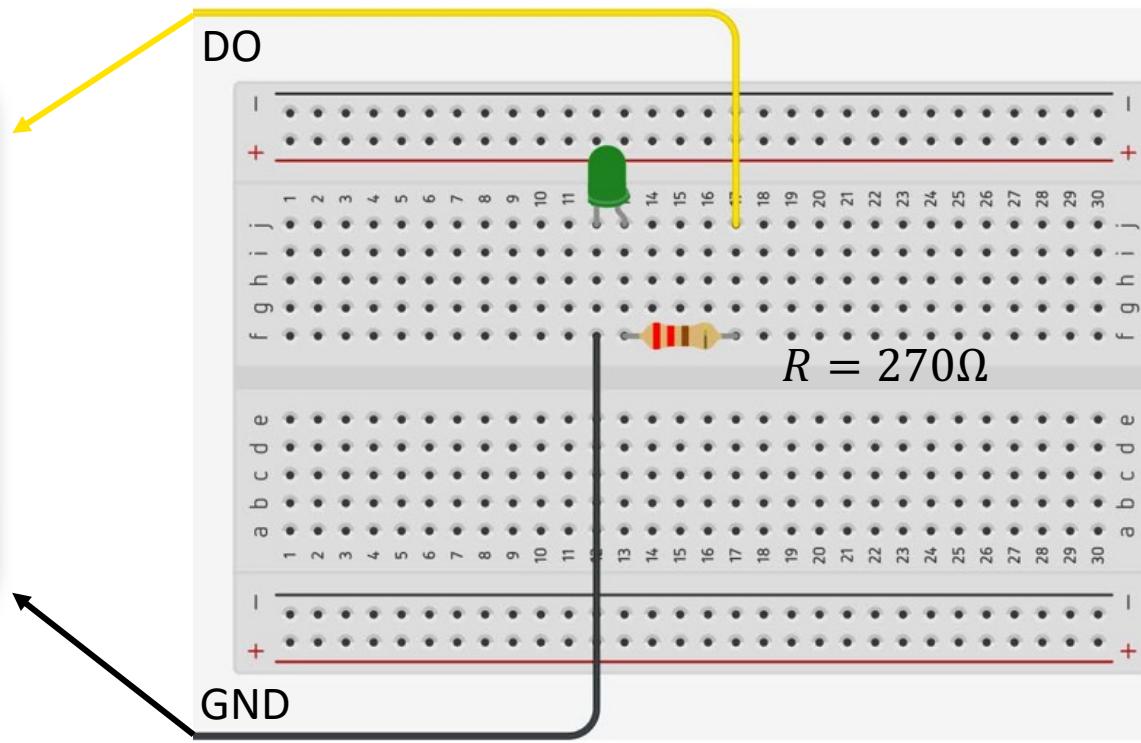
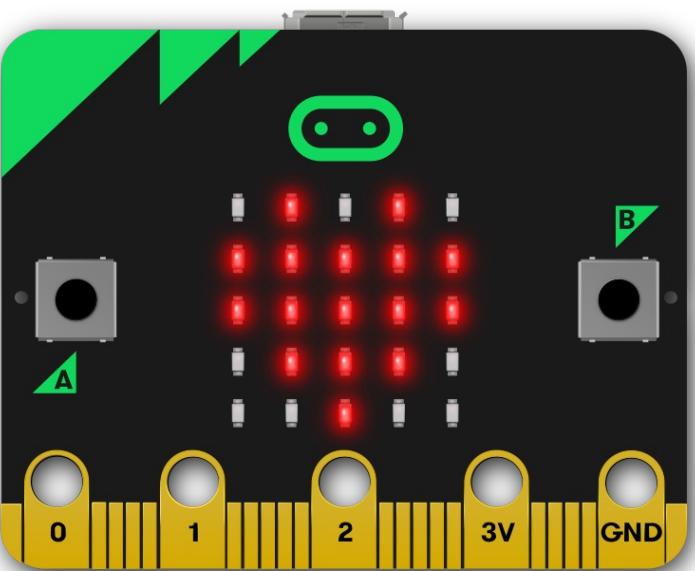


# Temperature with Alarm

Hans-Petter Halvorsen

[Table of Contents](#)

# LED Wiring



# Python



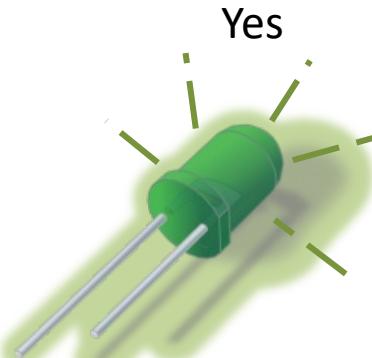
Temperature > Limit?

No



LED OFF

Yes



LED ON

```
from microbit import *
import thermistor

alarmLimit = 25

while True:
    adc = pin0.read_analog()
    Vout = (3.3/1023)*adc

    TempC = thermistor.thermistorTemp(Vout)

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

    if TempC > alarmLimit:
        print("Alarm")
        pin1.write_digital(1)
    else:
        pin1.write_digital(0)

    sleep(5000)
```

Mu 1.1.1 - thermistor\_led\_ex.py

The Mu IDE interface consists of several main sections: a top toolbar with icons for Mode, New, Load, Save, Flash, Files, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit; a tab bar with four tabs: thermistor.py, thermistor\_ex2.py, button\_led\_ex.py, and thermistor\_led\_ex.py; a code editor containing Python code for a BBC micro:bit; a BBC micro:bit REPL terminal at the bottom; and a footer with BBC micro:bit branding and settings icons.

thermistor.py    thermistor\_ex2.py    button\_led\_ex.py    thermistor\_led\_ex.py

```
1 from microbit import *
2 import thermistor
3
4 alarmLimit = 25
5
6 while True:
7     adc = pin0.read_analog()
8     Vout = (3.3/1023)*adc
9
10    TempC = thermistor.thermistorTemp(Vout)
11
12    print(round(TempC, 1))
13    display.scroll(round(TempC))
14
15    if TempC > alarmLimit:
16        print("Alarm")
17        pin1.write_digital(1)
18    else:
19        pin1.write_digital(0)
20
21    sleep(5000)
```

BBC micro:bit REPL

```
24.3
24.3
24.3
28.8
Alarm
```

BBC micro:bit

# 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>

