https://www.halvorsen.blog

TMP36 Temperature Sensor with Python

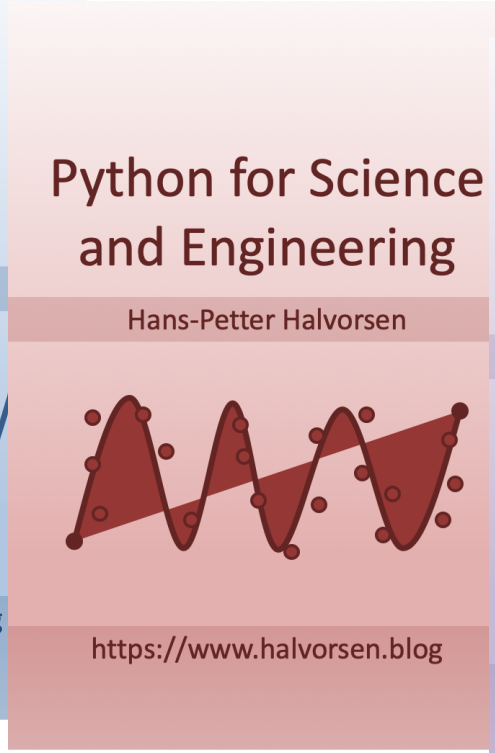
Exemplified by using NI USB-6008 I/O Module

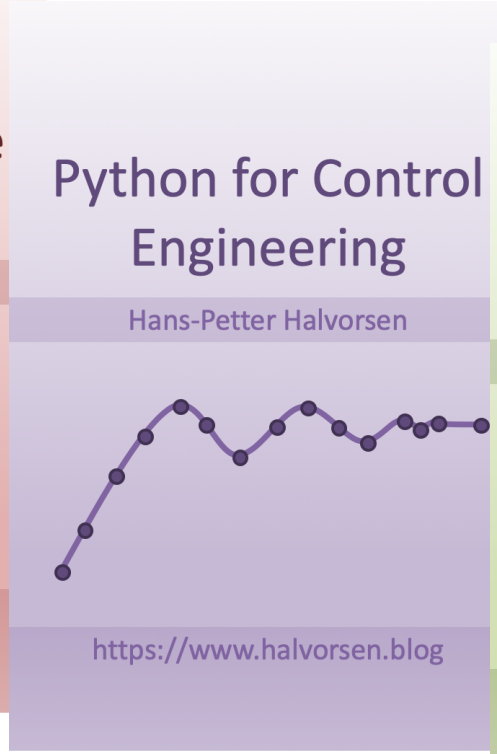
Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

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

Additional Python Resources



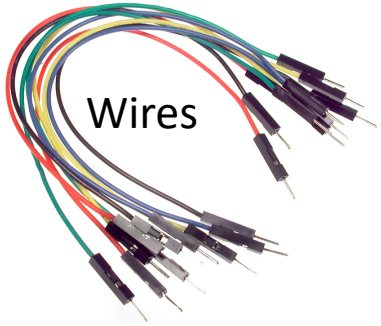


https://www.halvorsen.blog/documents/programming/python/

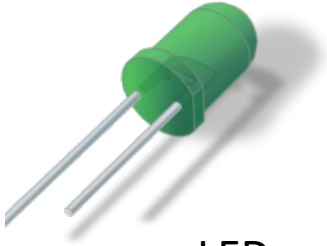
Equipment

USB-6008 (or similar DAQ Device)

TMP36 Temperature Sensor

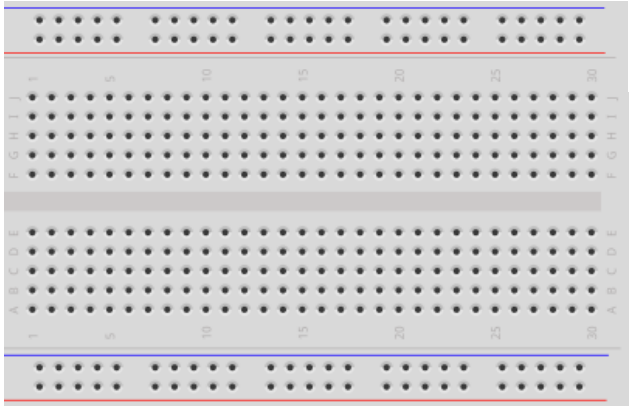


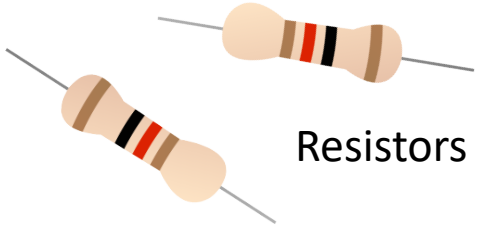
Wires



LED

Breadboard

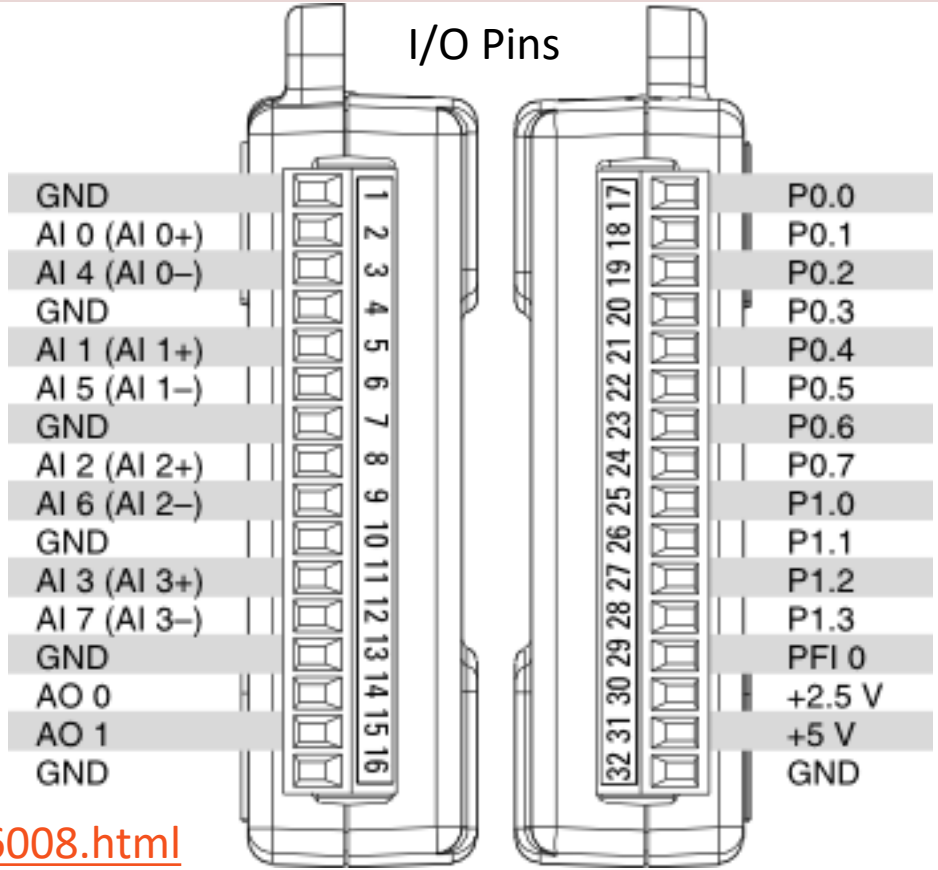




Resistors

NI USB-6008

We will use NI USB-6008 in our examples

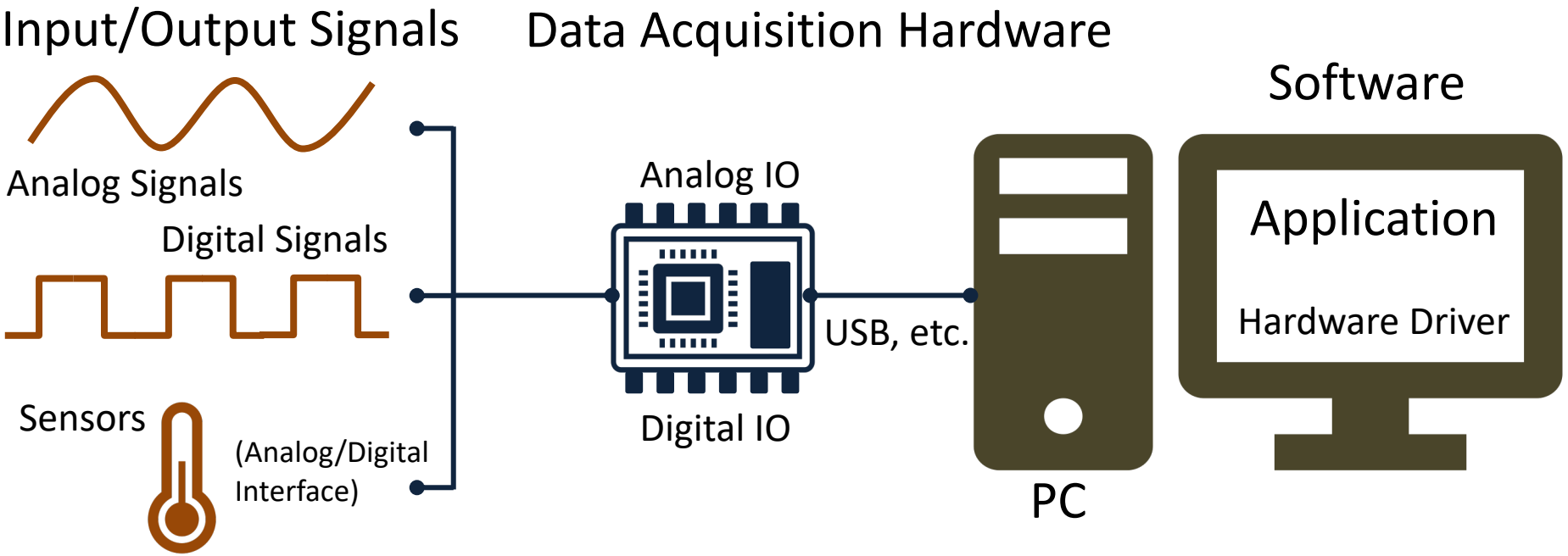


http://www.ni.com/en-no/support/model.usb-6008.html

NI DAQ Device with Python

How to use a NI DAQ Device with Python

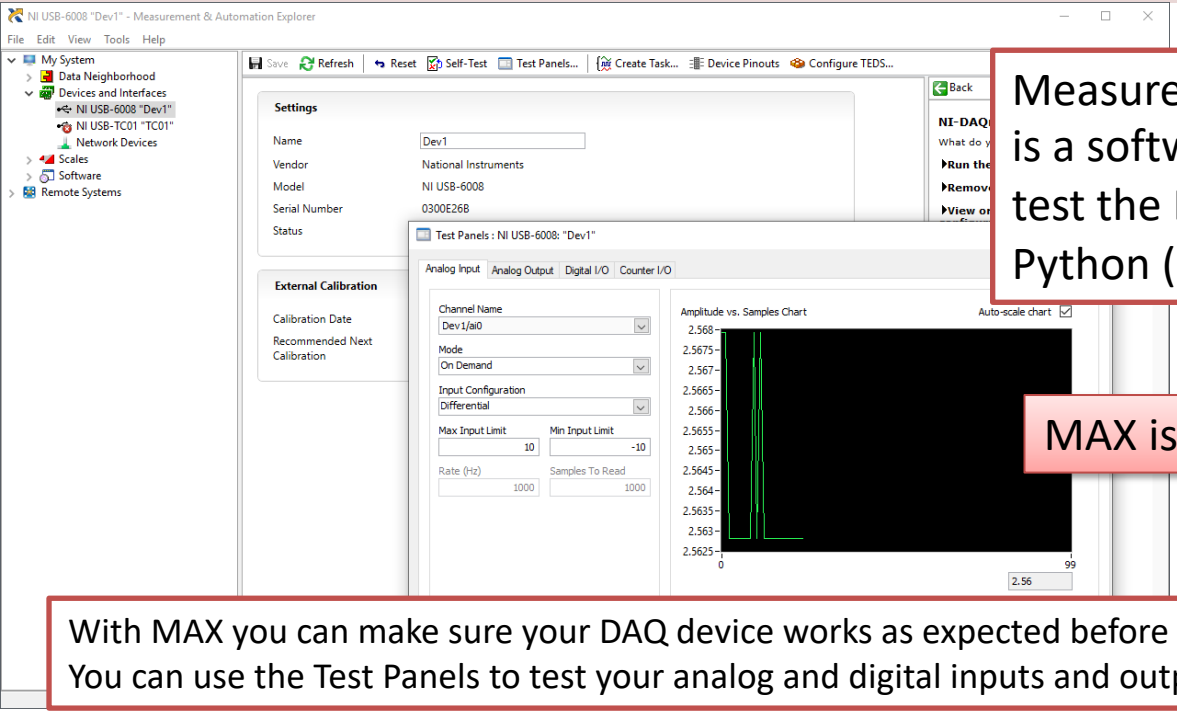
DAQ System


Input/Output Signals
Data Acquisition Hardware
Software
Analog Signals
Digital Signals
Sensors
(Analog/Digital Interface)
Analog IO
Digital IO
USB, etc.
PC
Application
Hardware Driver

NI-DAQmx

- NI-DAQmx is the software you use to communicate with and control your NI data acquisition (DAQ) device.
- NI-DAQmx supports only the Windows operating system.
- Typically you use LabVIEW in combination with NI DAQ Hardware, but the NI-DAQmx can also be used from C, C#, Python, etc.
- The NI-DAQmx Driver is Free!
- Visit the ni.com/downloads to download the latest version of NI-DAQmx

Measurement & Automation Explorer (MAX)



Measurement & Automation Explorer (MAX) is a software you can use to configure and test the DAQ device before you use it in Python (or other programming languages).

MAX is included with NI-DAQmx software

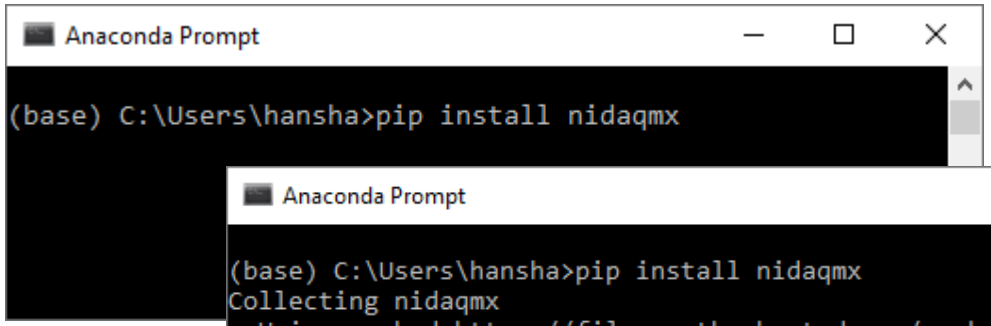
With MAX you can make sure your DAQ device works as expected before you start using it in your Python program. You can use the Test Panels to test your analog and digital inputs and outputs channels.

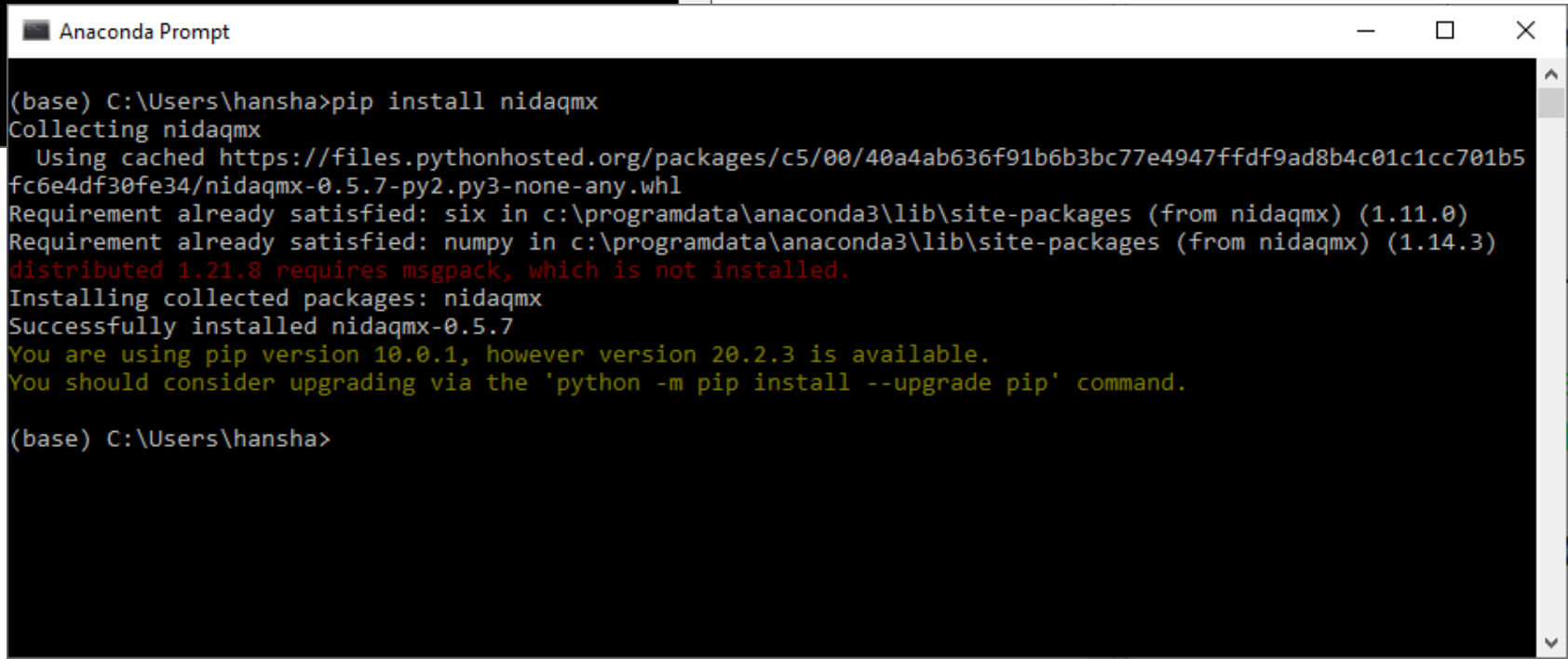
nidaqmx Python API

- Python Library/API for Communication with NI DAQmx Driver
- Running **nidaqmx** requires NI-DAQmx or NI-DAQmx Runtime
- Visit the ni.com/downloads to download the latest version of NI-DAQmx
- nidaqmx can be installed with **pip**:
  `pip install nidaqmx`
- https://github.com/ni/nidaqmx-python

nidaqmx Python Package

Installation



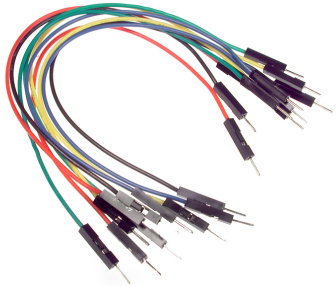


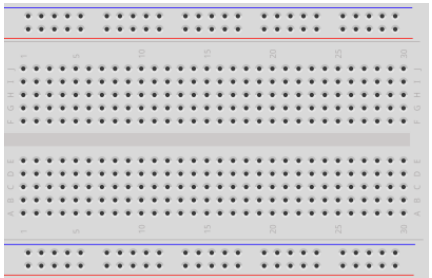
TMP36 Temperature with Python

Hans-Petter Halvorsen

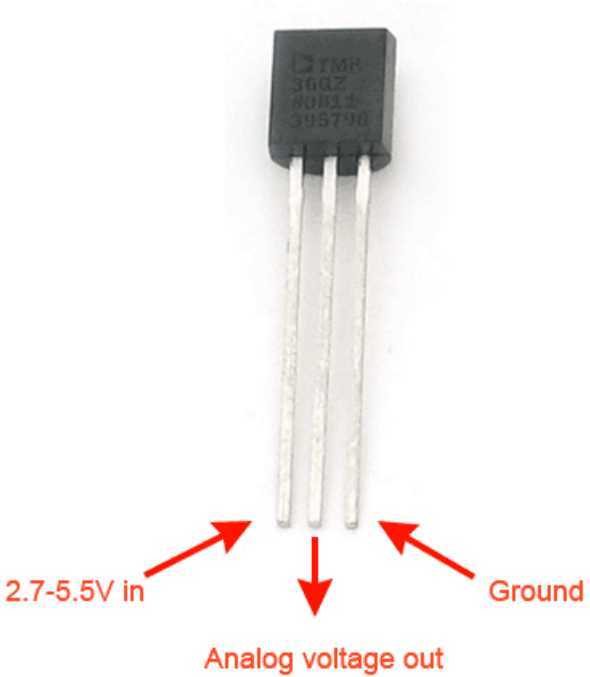
Necessary Equipment

- PC
- DAQ Module, e.g., USB-6008
- Breadboard
- TMP36
- Wires (Jumper Wires)





TMP36 Temperature

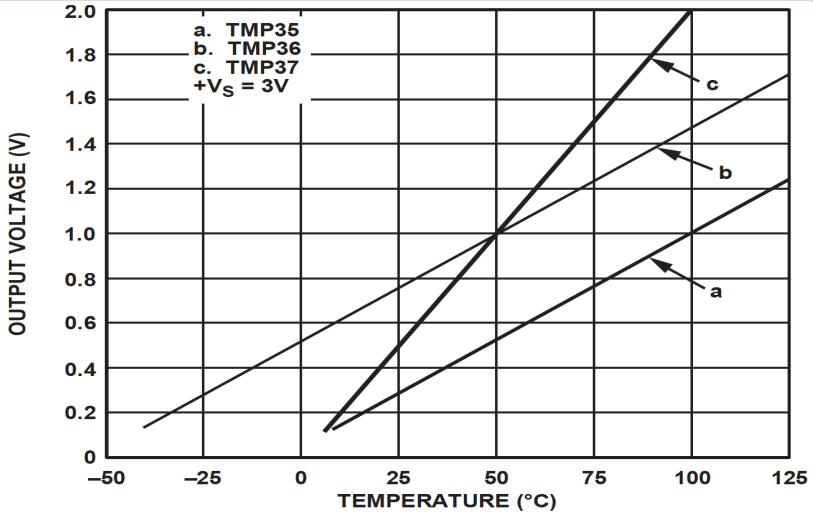


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

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

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

Scaling



This gives:

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

Then we get the following formula:

$$y = 100x - 50$$

Convert form Voltage (V) to degrees Celsius

From the Datasheet we have:

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

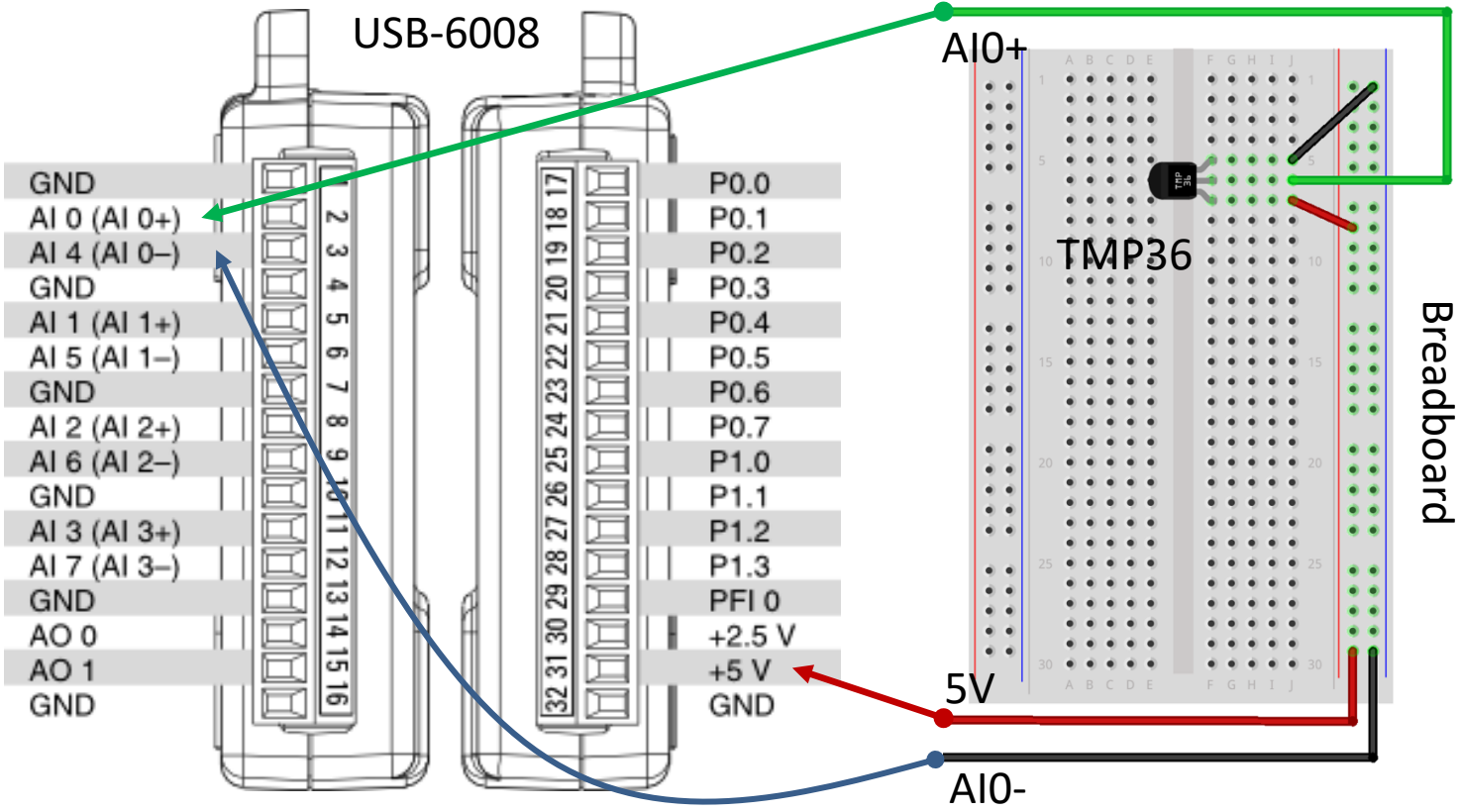
There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

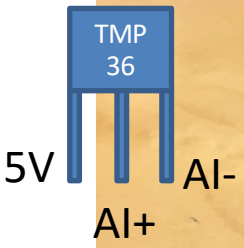
We can find a and b using the following known formula:

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

Wiring


USB-6008
GND
AI 0 (AI 0+)
AI 4 (AI 0–)
GND
AI 1 (AI 1+)
AI 5 (AI 1–)
GND
AI 2 (AI 2+)
AI 6 (AI 2–)
GND
AI 3 (AI 3+)
AI 7 (AI 3–)
GND
AO 0
AO 1
GND
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P1.0
P1.1
P1.2
P1.3
PFI 0
+2.5 V
+5 V
GND
AI0+
TMP36
5V
AI0-
Breadboard

Hardware Setup



We connect the TMP36 to LabVIEW using a USB DAQ Device from National Instruments, e.g., USB-6001, USB-6008 or similar. I have used a breadboard for the wiring.

Temperature Sensor - Python

In this Example we read one value from the sensor and convert from voltage to degrees Celsius.

```
import nidaqmx

task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
task.start()

voltage = task.read()
print(voltage)

degreesC = 100*voltage - 50

print(degreesC)

task.stop
task.close()
```

Formula converting from Voltage to Degrees Celsius:

$$y = 100x - 50$$

For Loop Example

In this Example we read data from the sensor within a For Loop.

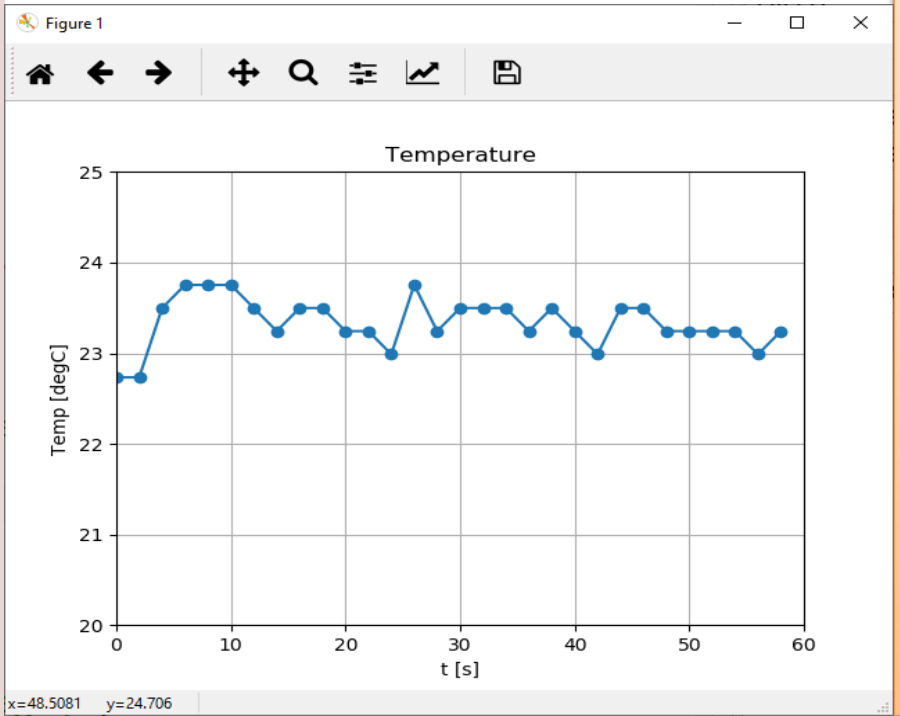
```
import nidaqmx
import time

task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
task.start()

Ts = 2
N = 10
for k in range(N):
    voltage = task.read()
    degreesC = 100*voltage - 50
    print(round(degreesC,1))
    time.sleep(Ts)

task.stop
task.close()
```

Plotting Temperature Data

Hans-Petter Halvorsen

Plotting Temperature Data

In this Example we read data from the sensor within a For Loop and Plot the Data using **matplotlib**



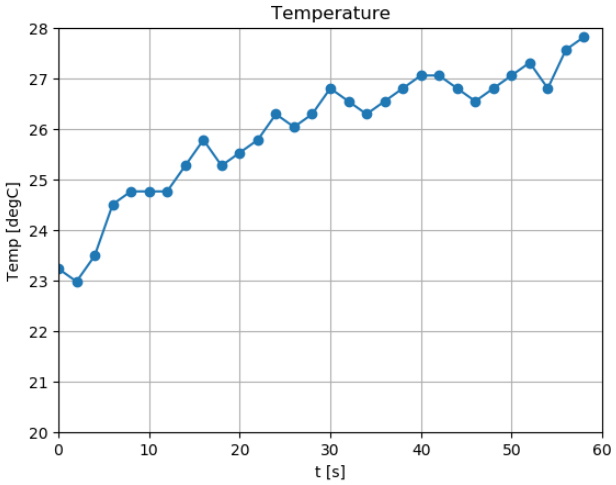

```
import numpy as np
import time
import matplotlib.pyplot as plt
import nidaqmx

# Initialize Logging
Tstop = 60 # Logging Time [seconds]
Ts = 2 # Sampling Time [seconds]
N = int(Tstop/Ts)
data = [] # Create Array for storing Temperature Data

# Initialize DAQ Device
task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
task.start()

# Start Logging
for k in range(N):
    voltage = task.read()
    degreesC = 100*voltage - 50
    print("T =", round(degreesC,1), "[degC]")
    data.append(degreesC)
    time.sleep(Ts)

# Terminate DAQ Device
task.stop
task.close()

# Plotting
t = np.arange(0,Tstop,Ts)
plt.plot(t,data, "-o")
plt.title('Temperature');plt.xlabel('t [s]')
plt.ylabel('Temp [degC]')
plt.grid()
Tmin = 20; Tmax = 25
plt.axis([0, Tstop, Tmin, Tmax])
plt.show()
```

Logging Temperature Data to File

Hans-Petter Halvorsen

Logging Data to File

In this Example we read data from the sensor within a For Loop and Plot the Data using matplotlib and Save the Temperature values to a File as well.



tempdata.txt - Notepad

File Edit Format View Help

0	23.2
2	23.0
4	23.5
6	24.5
8	24.8
10	24.8
12	24.8
14	25.3
16	25.8
18	25.3
20	25.5
22	25.8
24	26.3
26	26.0
28	26.3
30	26.8
32	26.6
34	26.3
36	26.6
38	26.8
40	27.1
42	27.1
44	26.8
46	26.6
48	26.8
50	27.1
52	27.3
54	26.8
56	27.6
58	27.8

100% Windows (CRLF) UTF-8

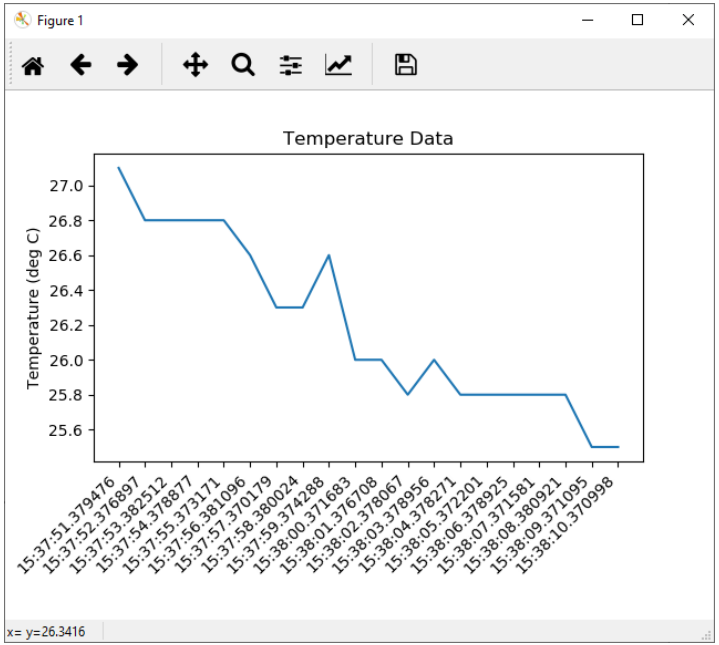
```
import numpy as np
import time
import matplotlib.pyplot as plt
import nidaqmx

# Initialize Logging
Tstop = 60 # Logging Time [seconds]
Ts = 2 # Sampling Time [seconds]
N = int(Tstop/Ts)
data = [] # Create Array for storing Temperature Data

# Open File
file = open("tempdata.txt", "w")

# Initialize DAQ Device
task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
task.start()

# Write Data to File Function
def writefiledata(t, x):
    time = str(t)
    value = str(round(x, 2))
    file.write(time + "\t" + value)
    file.write("\n")

# Start Logging
for k in range(N):
    voltage = task.read()
    degreesC = 100*voltage - 50
    print("T =", round(degreesC,1), "[degC]")
    data.append(degreesC)
    writefiledata(k*Ts, round(degreesC,1))
    time.sleep(Ts)

# Terminate DAQ Device
task.stop
task.close()

# Close File
file.close()

# Plotting
t = np.arange(0,Tstop,Ts)
plt.plot(t,data, "-o")
plt.title('Temperature')
plt.xlabel('t [s]')
plt.ylabel('Temp [degC]')
plt.grid()
Tmin = 20; Tmax = 28
plt.axis([0, Tstop, Tmin, Tmax])
plt.show()
```

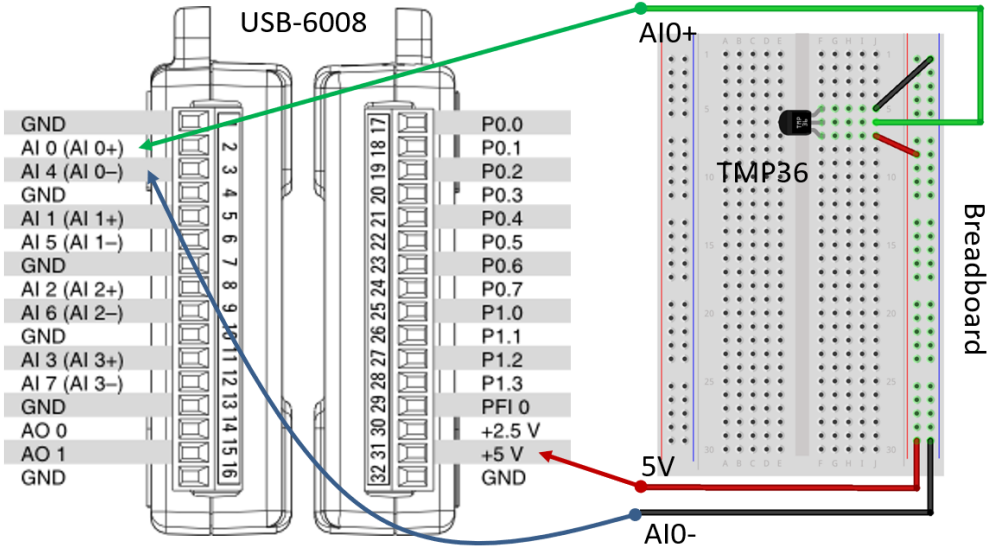
Real-Time Plotting of Data

Hans-Petter Halvorsen

Real-Time Plotting

Here in this Example we will read the value from the TMP36 Sensor and Plot the Data in Real-Time





Python Code

```
import nidaqmx
import datetime as dt
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# Create figure for plotting
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
xs = []
ys = []

# Initialize DAQ device
task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
task.start
```

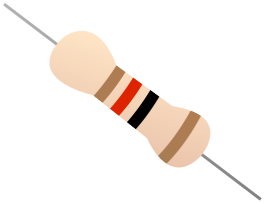
```
# This function is called periodically from FuncAnimation
def readdaq(i, xs, ys):

    #Read Value from DAQ device
    voltage = task.read()

    #Convert Voltage to degrees Celsius
    degreesC = 100*voltage - 50
    temp_c = round(degreesC, 1)
    print("Celsius Value:", temp_c)

    # Add x and y to lists
    xs.append(dt.datetime.now().strftime('%H:%M:%S.%f'))
    ys.append(temp_c)

    # Limit x and y lists to 20 items
    xs = xs[-20:]; ys = ys[-20:]

    # Draw x and y lists
    ax.clear()
    ax.plot(xs, ys)

    # Format plot
    plt.xticks(rotation=45, ha='right')
    plt.subplots_adjust(bottom=0.30)
    plt.title('Temperature Data')
    plt.ylabel('Temperature (deg C)')

# Set up plot to call readdaq() function periodically
ani = animation.FuncAnimation(fig, readdaq, fargs=(xs, ys),
interval=1000)
plt.show()
task.stop
```

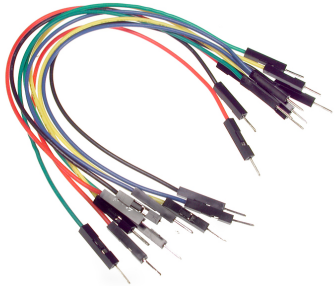
Temperature with Alarm

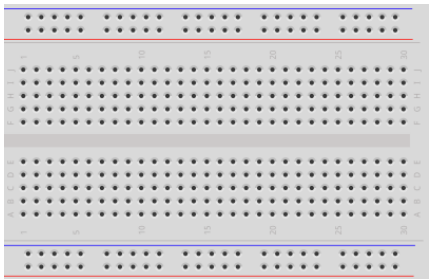
Hans-Petter Halvorsen

Necessary Equipment

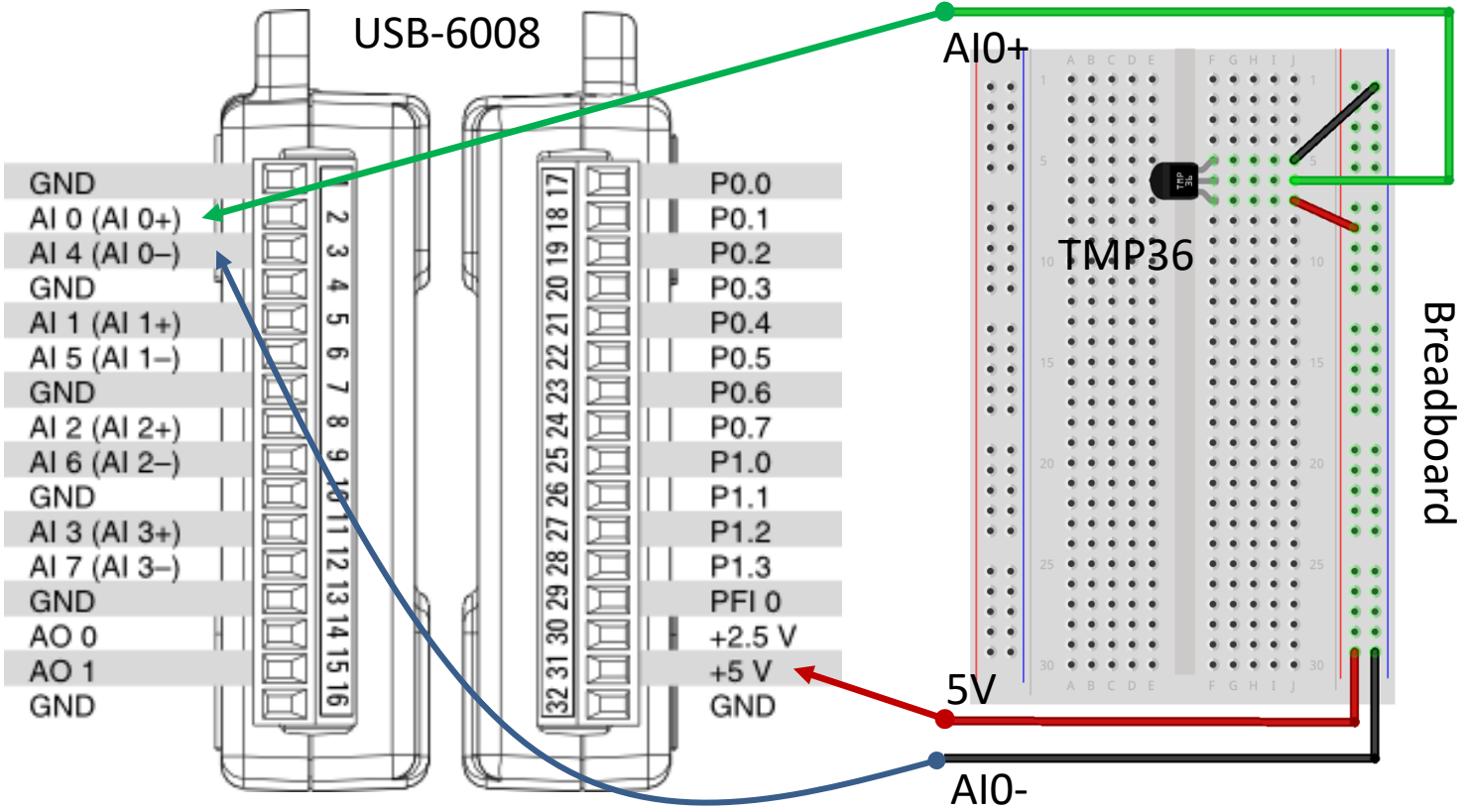
- PC
- DAQ Module, e.g., USB-6008
- Breadboard
- TMP36
- **LED**
- **Resistor, $R = 270\Omega$**
- Wires (Jumper Wires)



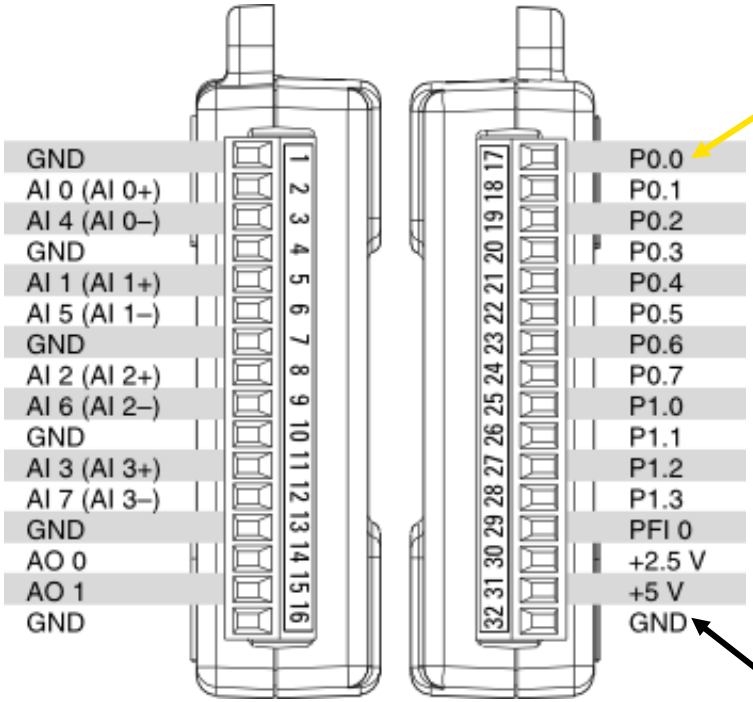


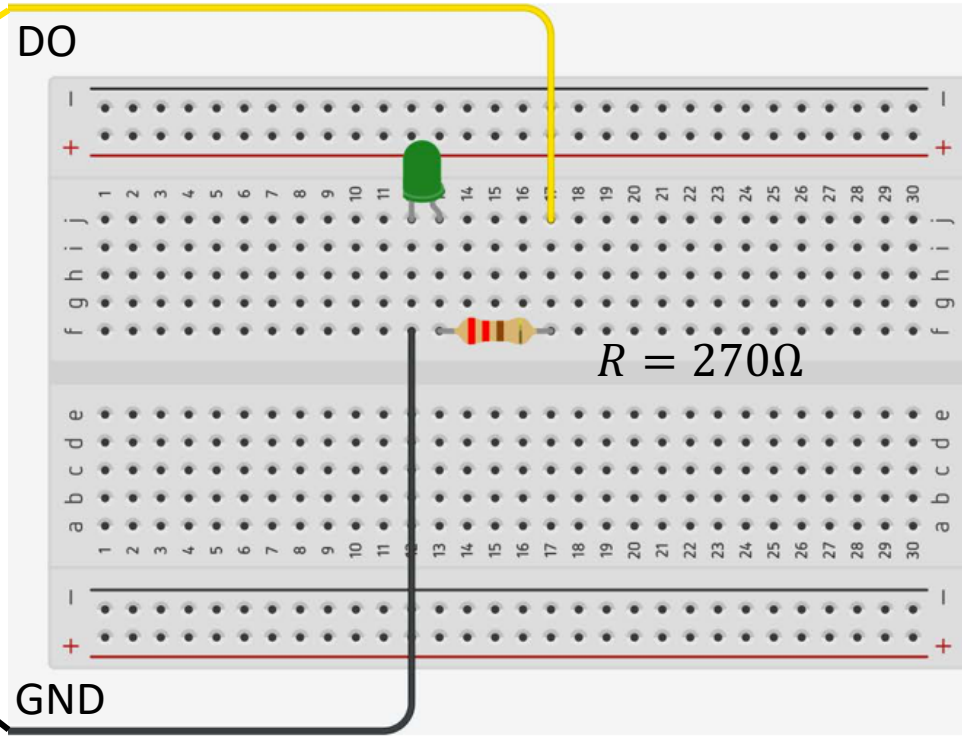


TMP36 Wiring

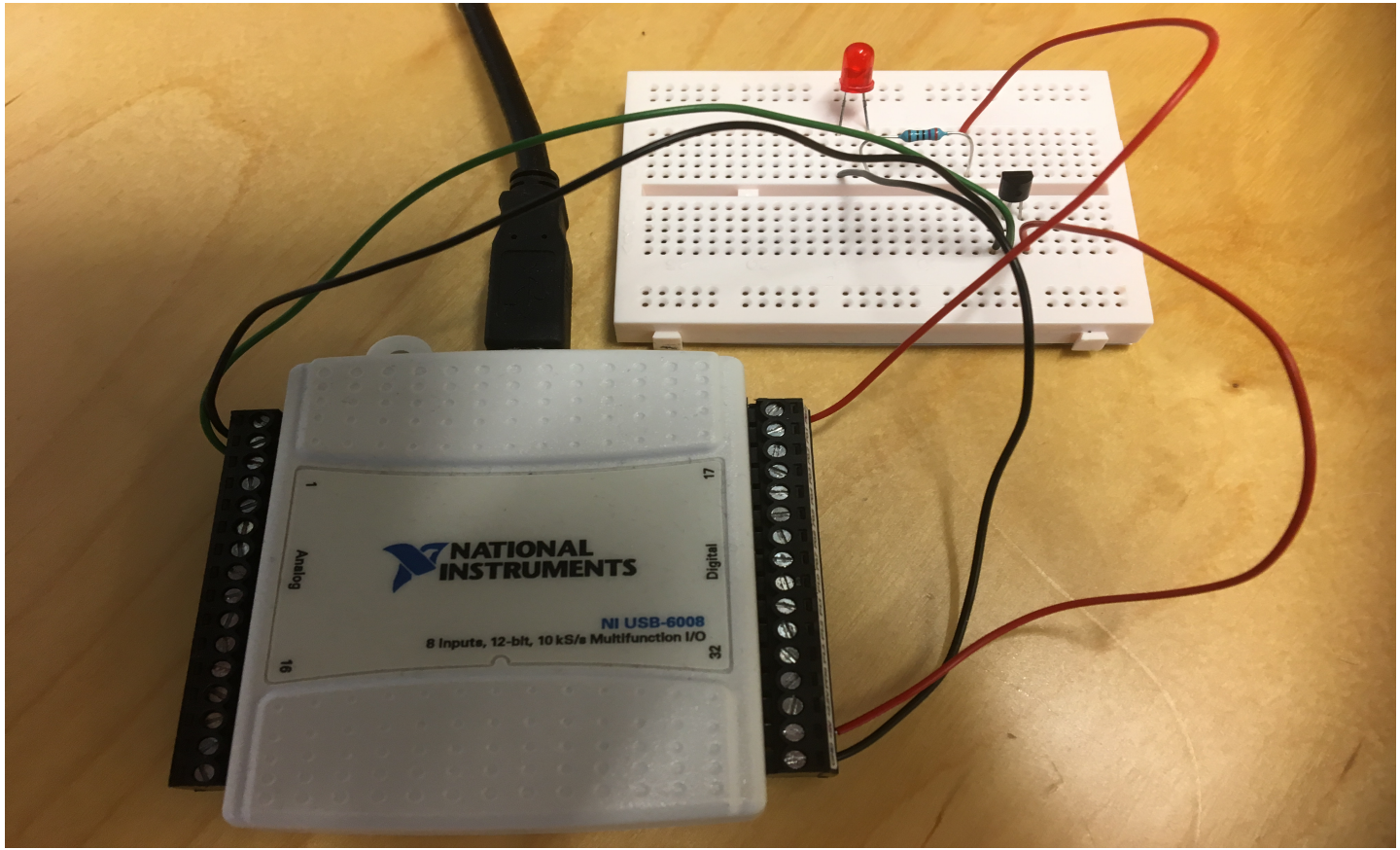


LED Wiring


GND
AI 0 (AI 0+)
AI 4 (AI 0–)
GND
AI 1 (AI 1+)
AI 5 (AI 1–)
GND
AI 2 (AI 2+)
AI 6 (AI 2–)
GND
AI 3 (AI 3+)
AI 7 (AI 3–)
GND
AO 0
AO 1
GND
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P1.0
P1.1
P1.2
P1.3
PFI 0
+2.5 V
+5 V
GND


DO
$R = 270\Omega$
GND

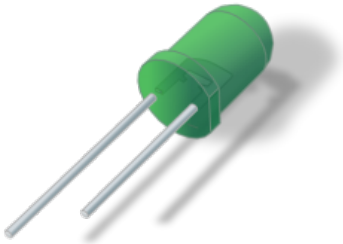
Hardware Setup



Python Code

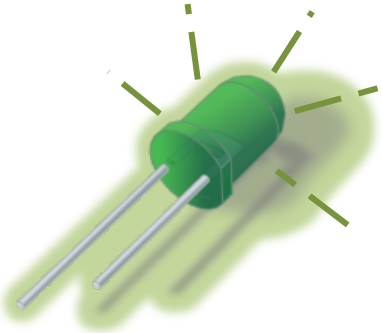
Temperature > Limit?

No



LED OFF

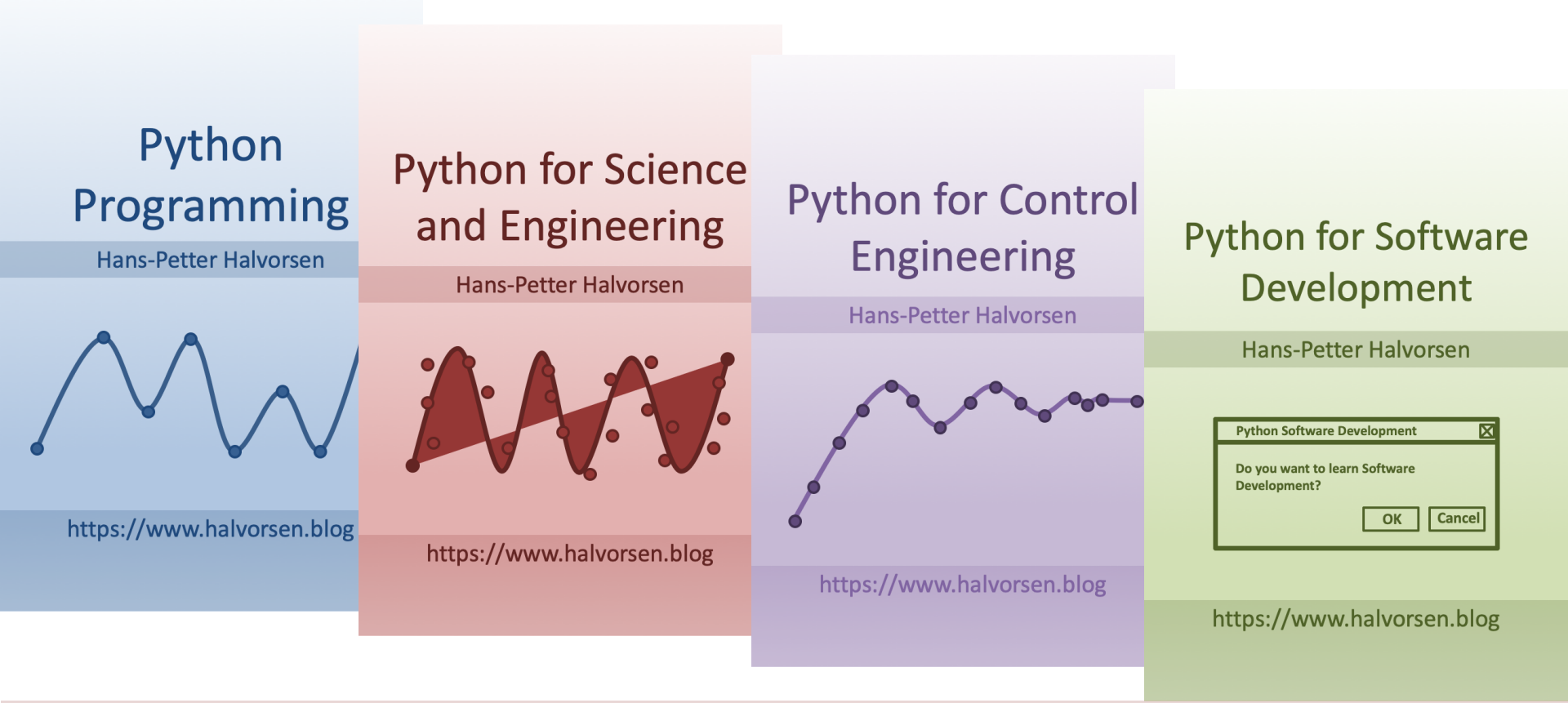
Yes



LED ON

```
import nidaqmx
import time

# Initialize DAQ Device
task_ai = nidaqmx.Task()
task_ai.ai_channels.add_ai_voltage_chan("Dev1/ai0")
task_ai.start()

task_do = nidaqmx.Task()
task_do.do_channels.add_do_chan("Dev1/port0/line0")
task_do.start()

alarmlimit = 24 #degrees Celsius

Ts = 2
N = 10
# Start Logging
for k in range(N):
    voltage = task_ai.read()
    degreesC = 100*voltage - 50
    print(round(degreesC,1))

    if degreesC >= alarmlimit:
        task_do.write(True)
    else:
        task_do.write(False)

    time.sleep(Ts)

# Terminate DAQ Device
task_ai.stop; task_ai.close()
task_do.stop; task_do.close()
```

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