https://www.halvorsen.blog

# Thermistor Temperature Sensors with Python

Exemplified by using NI USB-6008 I/O Module

Hans-Petter Halvorsen

# Free Textbook with lots of Practical Examples

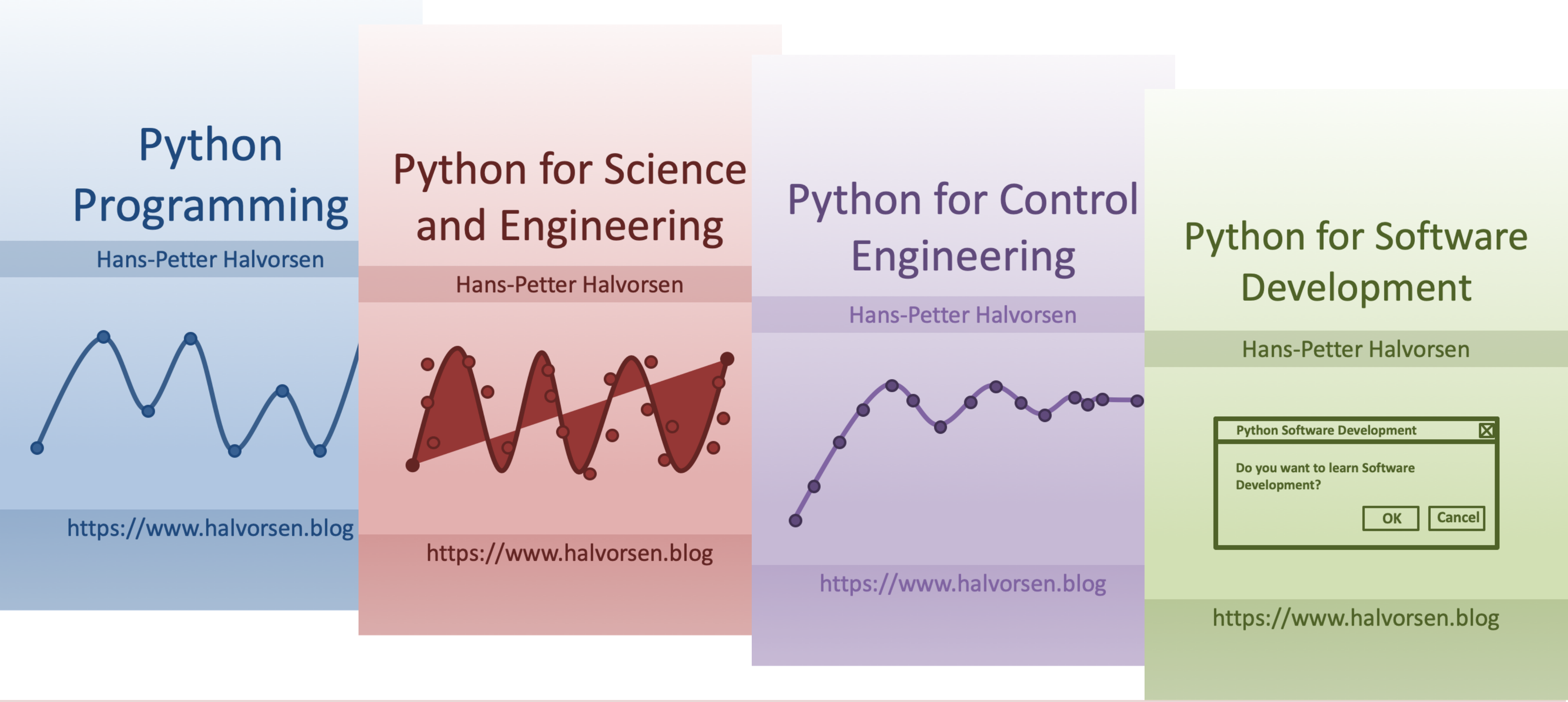
Python for Science and Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

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

# Additional Python Resources



Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Science and Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Control Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Software Development

Hans-Petter Halvorsen

Python Software Development

Do you want to learn Software Development?

OK Cancel

https://www.halvorsen.blog

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

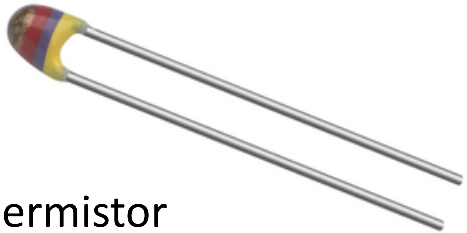
# Contents

- DAQ and I/O Modules
- NI-DAQ
- Thermistors
  - Resistance Temperature Detectors (RTD)
- Python Examples

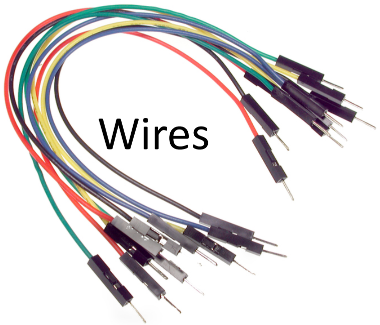
Note! The Python Examples provided will work for all NI-DAQ Devices using the NI-DAQmx Driver, which is several hundreds different types. We will use the NI USB-6008 DAQ Device or I/O Module as an Example

# Equipment

USB-6008 (or similar DAQ Device)

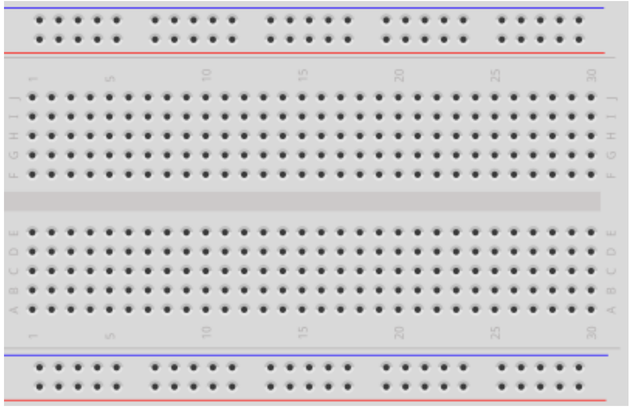


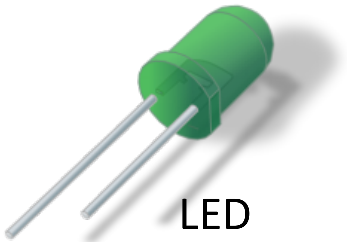
Thermistor



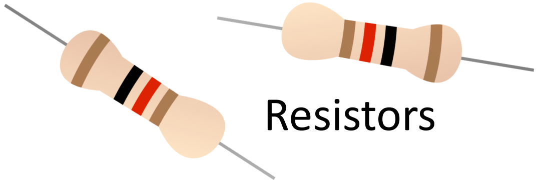
Wires

Breadboard





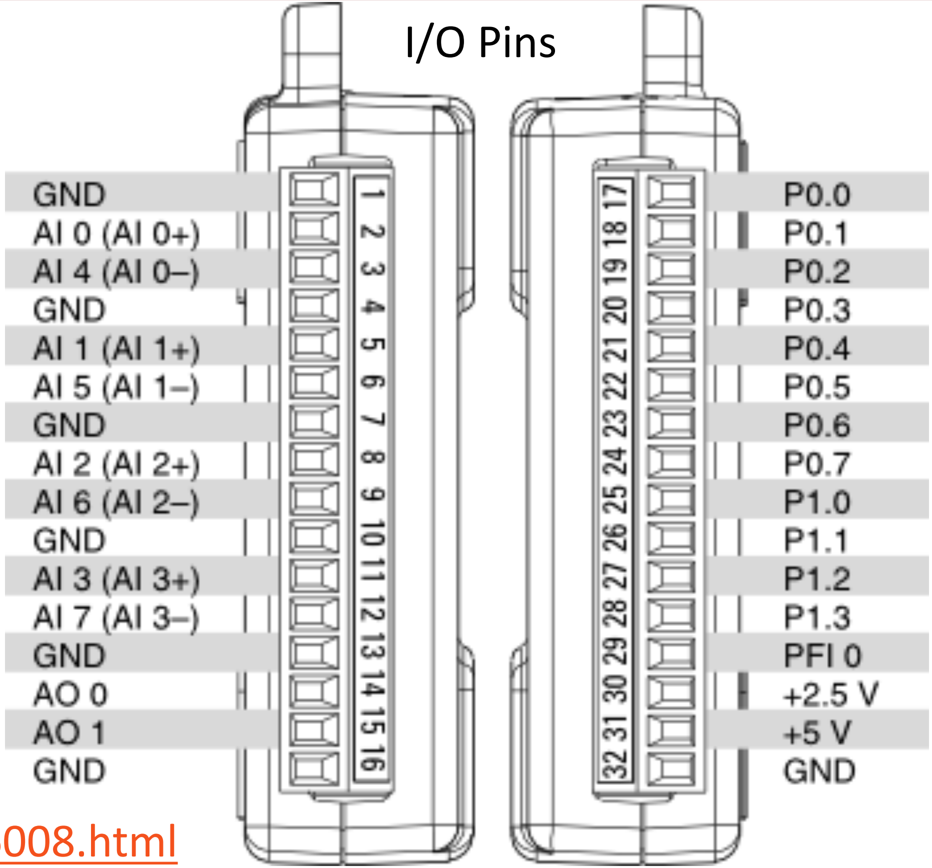
LED



Resistors

# NI USB-6008

We will use NI USB-6008 in our examples



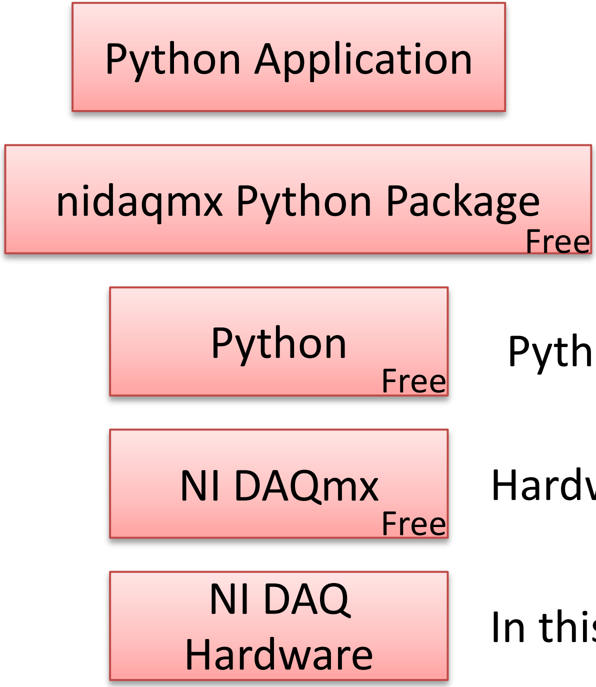
I/O Pins

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

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

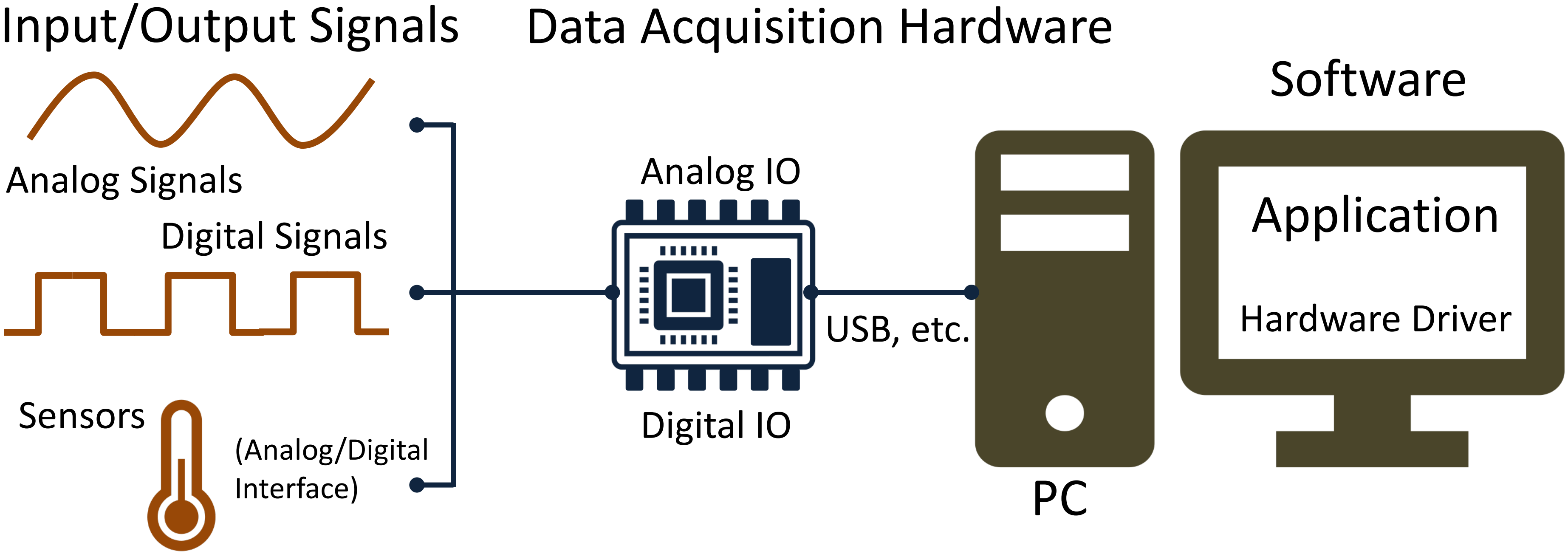
# NI DAQ Device with Python

How to use a NI DAQ Device with Python



| | |
|---|---|
| Python Application | Your Python Program |
| nidaqmx Python Package<br>Free | Python Library/API for Communication with NI DAQmx Driver |
| Python<br>Free | Python Programming Language |
| NI DAQmx<br>Free | Hardware Driver Software |
| NI DAQ Hardware | In this Tutorial we will use USB-6008 DAQ Device or I/O Module |

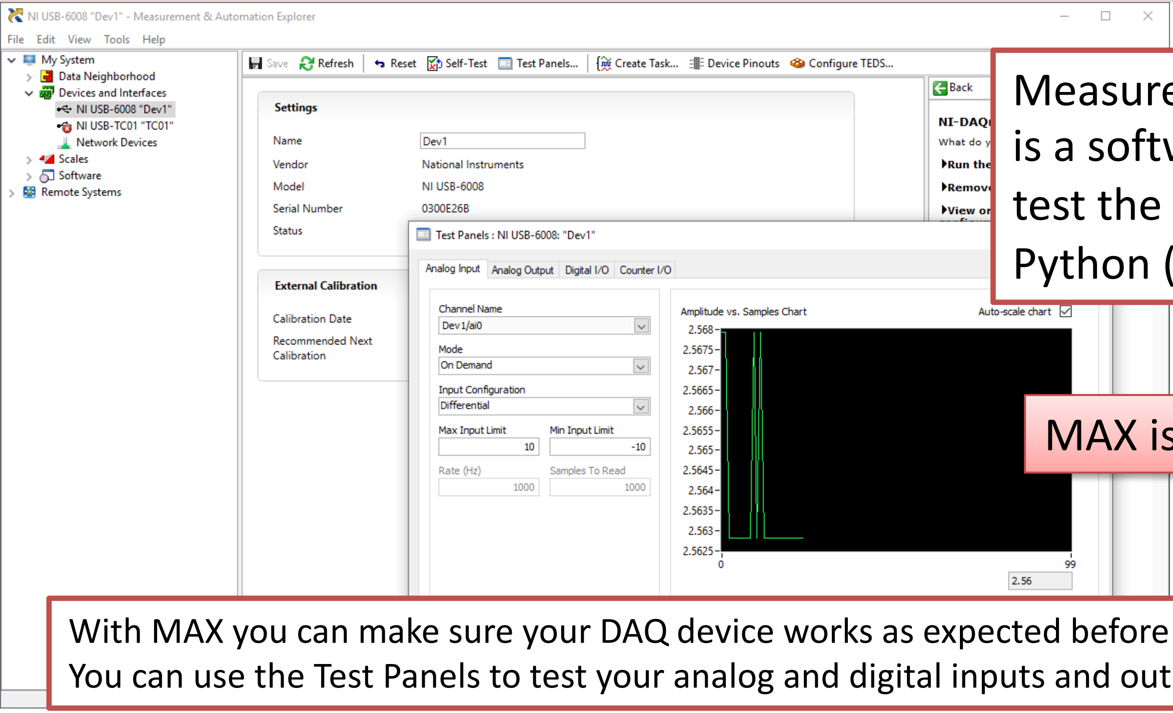
# DAQ System


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

# 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)



Settings

Name: Dev1
Vendor: National Instruments
Model: NI USB-6008
Serial Number: 0300E26B
Status

External Calibration

Calibration Date
Recommended Next Calibration

Test Panels : NI USB-6008: "Dev1"

Analog Input | Analog Output | Digital I/O | Counter I/O

Channel Name: Dev1/ai0
Mode: On Demand
Input Configuration: Differential
Max Input Limit: 10
Min Input Limit: -10
Rate (Hz): 1000
Samples To Read: 1000
Auto-scale chart ☑
2.56

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

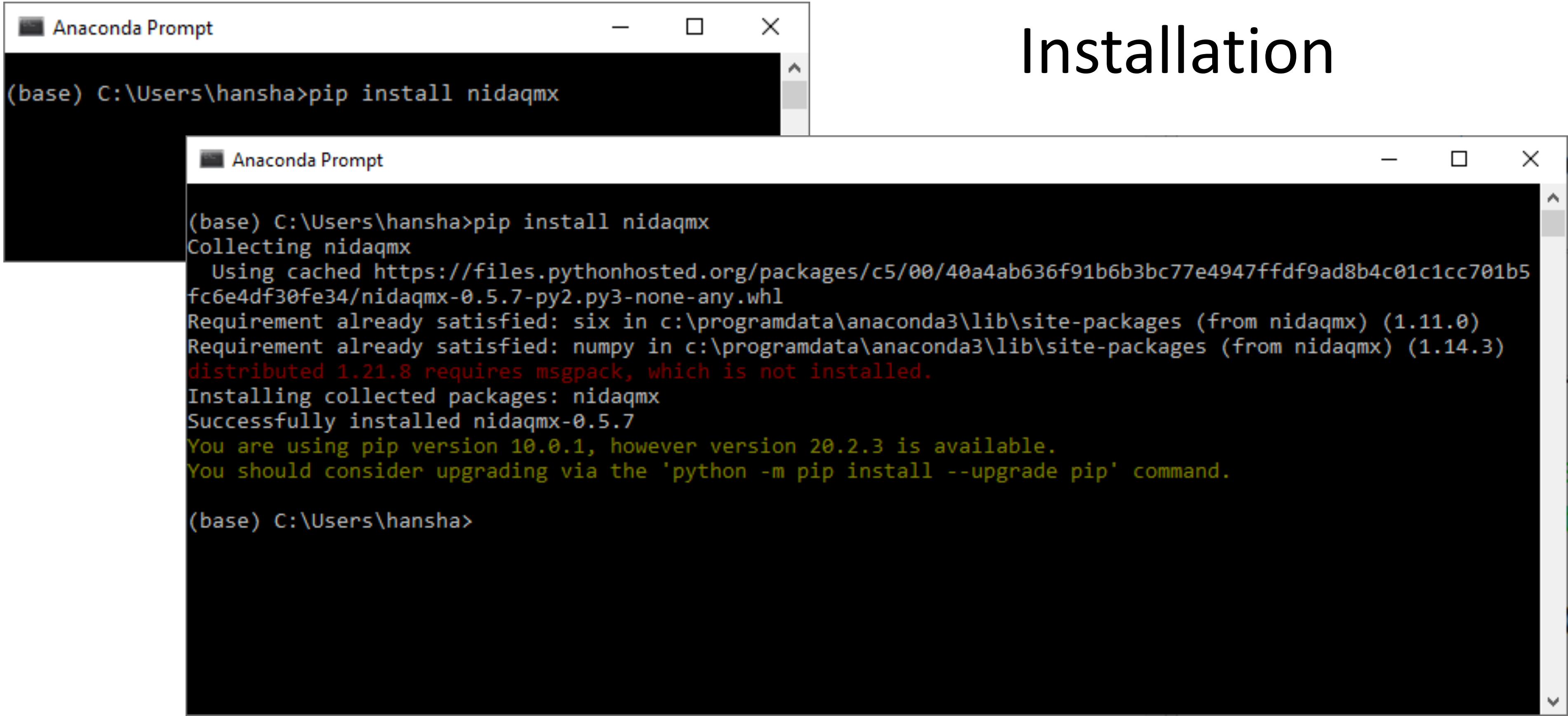
With MAX you can make sure your DAQ device works as expected before you start using it in your Python program. You can use the Test Panels to test your analog and digital inputs and outputs channels.

# 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



```
(base) C:\Users\hansha>pip install nidaqmx
```

```
(base) C:\Users\hansha>pip install nidaqmx
Collecting nidaqmx
  Using cached https://files.pythonhosted.org/packages/c5/00/40a4ab636f91b6b3bc77e4947ffdf9ad8b4c01c1cc701b5
fc6e4df30fe34/nidaqmx-0.5.7-py2.py3-none-any.whl
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from nidaqmx) (1.11.0)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from nidaqmx) (1.14.3)
distributed 1.21.8 requires msgpack, which is not installed.
Installing collected packages: nidaqmx
Successfully installed nidaqmx-0.5.7
You are using pip version 10.0.1, however version 20.2.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(base) C:\Users\hansha>
```

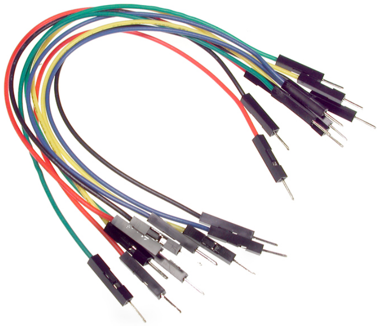
https://www.halvorsen.blog

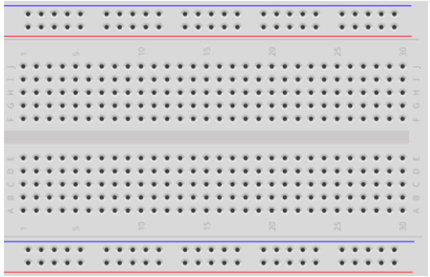
# Thermistor with Python

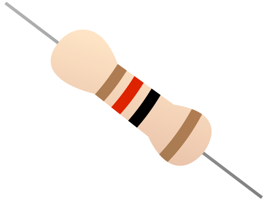
Hans-Petter Halvorsen

# Necessary Equipment

- PC
- DAQ Module, e.g., USB-6008
- Breadboard
- Thermistor
- Resistor 10 kΩ
- Wires (Jumper Wires)







# 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**):

[Wikipedia]

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

where $A, B, C$ are constants given below

$A = 0.001129148, B = 0.000234125 \; and \; C = 8.76741E-08$

# 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

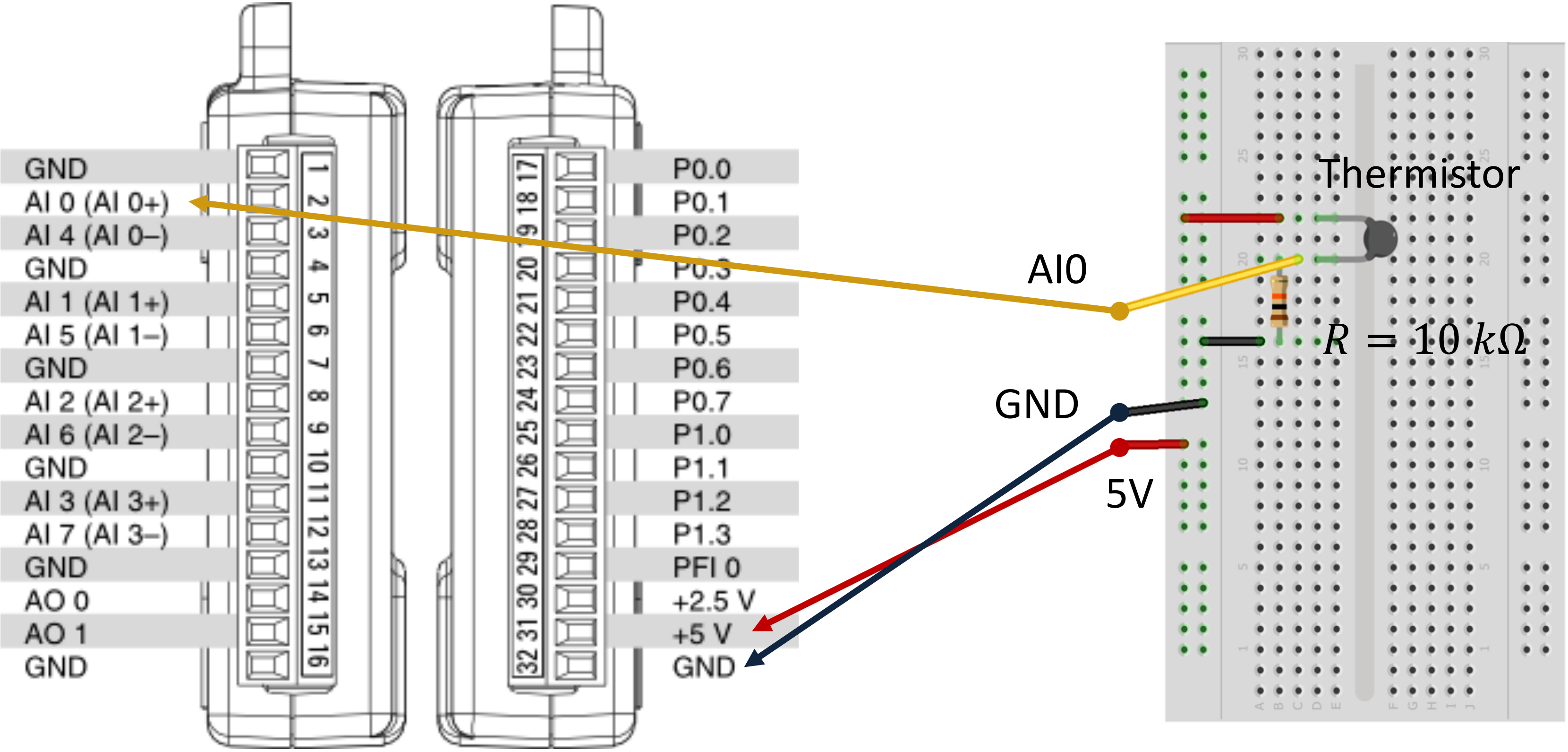
$A, B\ and\ C$ are constants

$$A = 0.001129148,$$
$$B = 0.000234125$$
$$C = 0.0000000876741$$

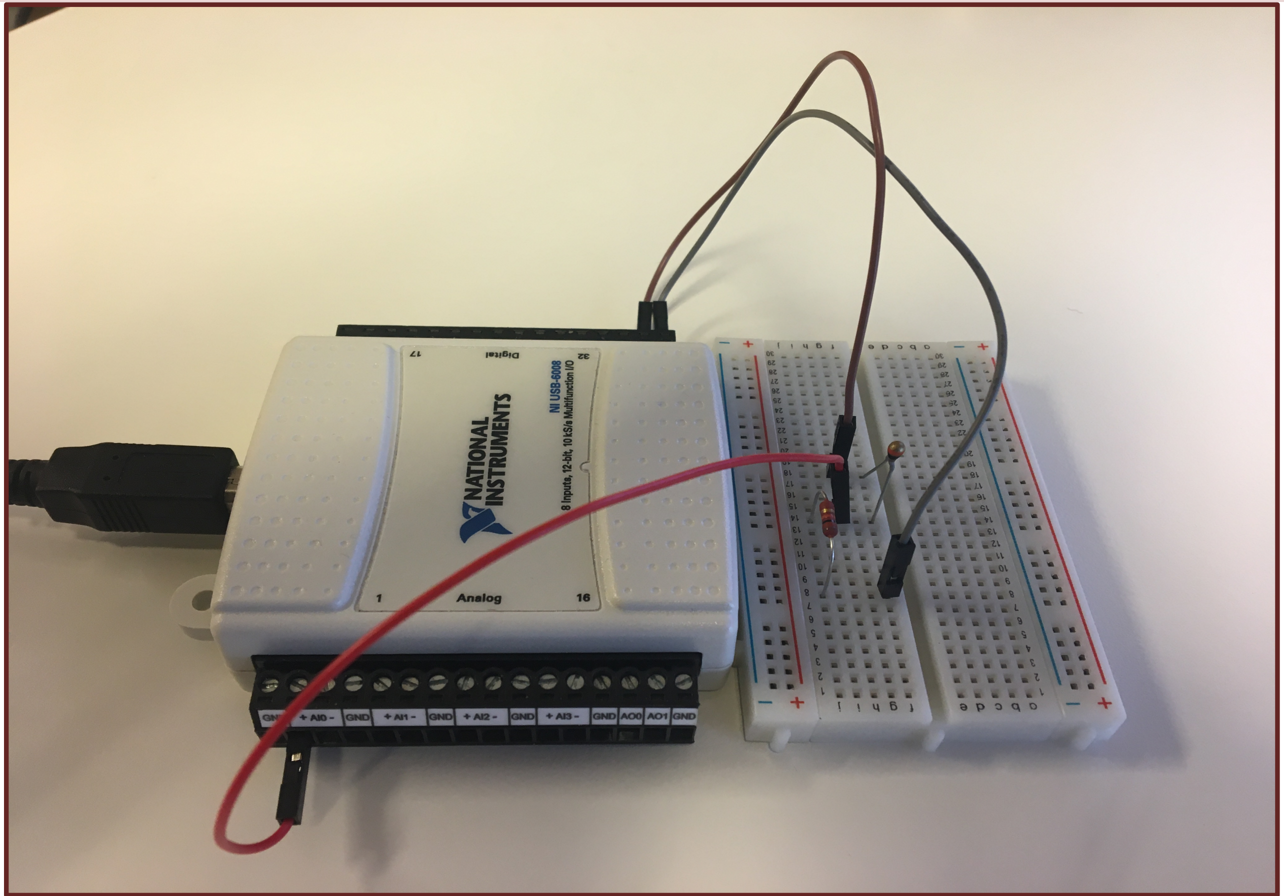
The Temperature in degrees Celsius will then be:

$$T_C = T_K - 273.15$$

# Wiring

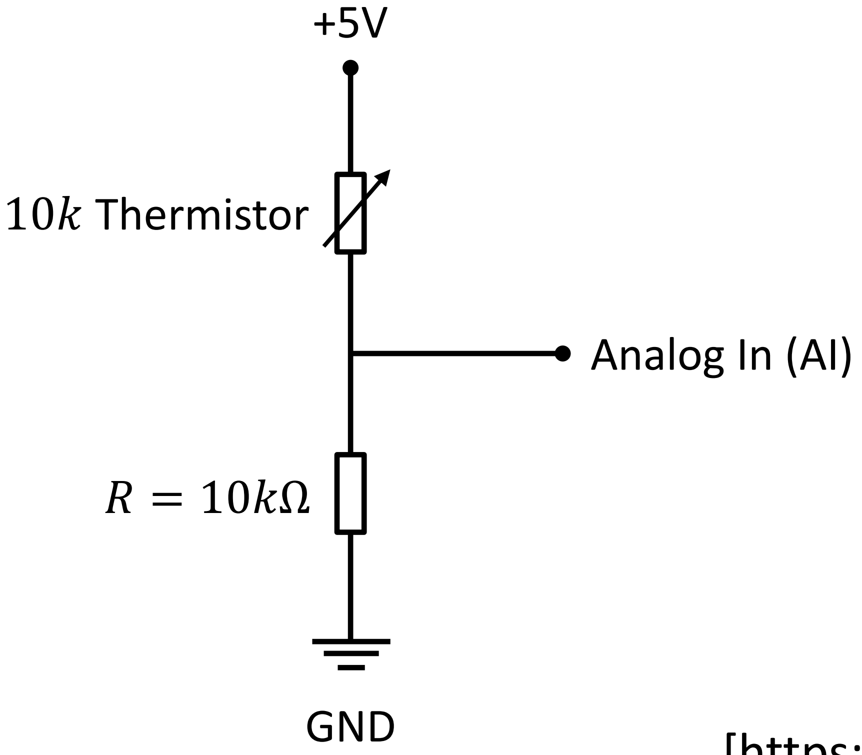

GND
AI 0 (AI 0+)
AI 4 (AI 0–)
GND
AI 1 (AI 1+)
AI 5 (AI 1–)
GND
AI 2 (AI 2+)
AI 6 (AI 2–)
GND
AI 3 (AI 3+)
AI 7 (AI 3–)
GND
AO 0
AO 1
GND
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P1.0
P1.1
P1.2
P1.3
PFI 0
+2.5 V
+5 V
GND
Thermistor
AI0
$R = 10\,k\Omega$
GND
5V

# Hardware Setup


NATIONAL INSTRUMENTS
NI USB-6008
8 Inputs, 12-bit, 10 kS/s Multifunction I/O
Digital
17
32
Analog
1
16
GND + AI0 - GND + AI1 - GND + AI2 - GND + AI3 - GND AO0 AO1 GND

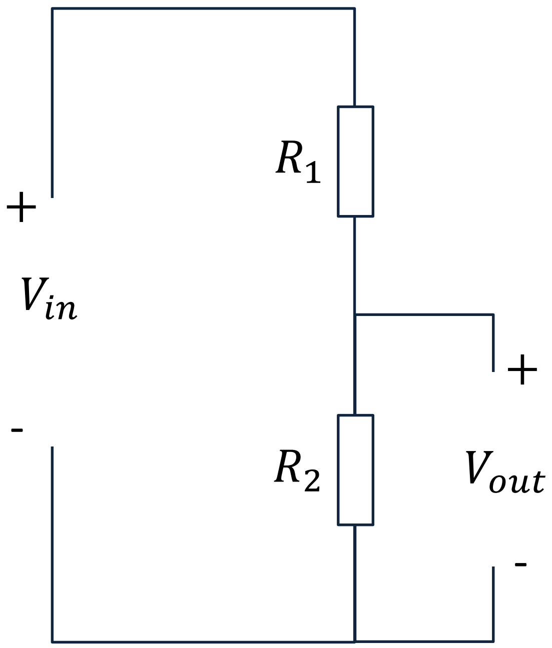
# Voltage Divider

The wiring is called a “Voltage Divider”



[https://en.wikipedia.org/wiki/Voltage_divider]

# General Voltage Divider



We want to find $V_{out}$

Formula:

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

https://learn.sparkfun.com/tutorials/voltage-dividers/all

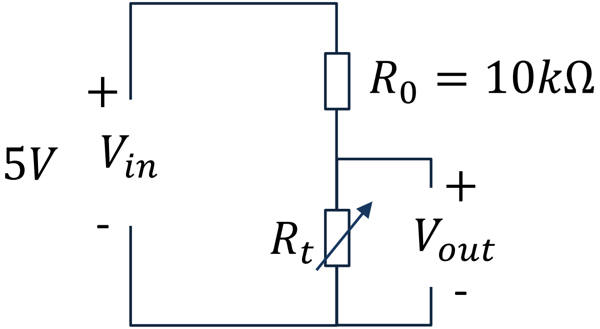
# Voltage Divider for our System

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°C

<u>Steps:</u>

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

# Python Code

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

```
import math as mt
import nidaqmx

# Initialization

from nidaqmx.constants import (
    TerminalConfiguration)

# Voltage Divider
Vin = 5;
Ro = 10000 # 10k Resistor

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

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

# Read from DAQ Device
Vout = task.read()
print(Vout)

# 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 * mt.log(Rt)) + C * mt.pow(mt.log(Rt),3))

# Convert from Kelvin to Celsius
TempC = TempK - 273.15

print(TempC)

task.stop
task.close()
```

# Python Code

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

## thermistor.py

```
import math as mt

# Function for finding Temperature in degrees Celsius
def thermistorTemp(Vout):
    # Voltage Divider
    Vin = 5;
    Ro = 10000 # 10k Resistor

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

    # 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 * mt.log(Rt)) + C * mt.pow(mt.log(Rt),3))

    # Convert from Kelvin to Celsius
    TempC = TempK - 273.15

    return TempC
```

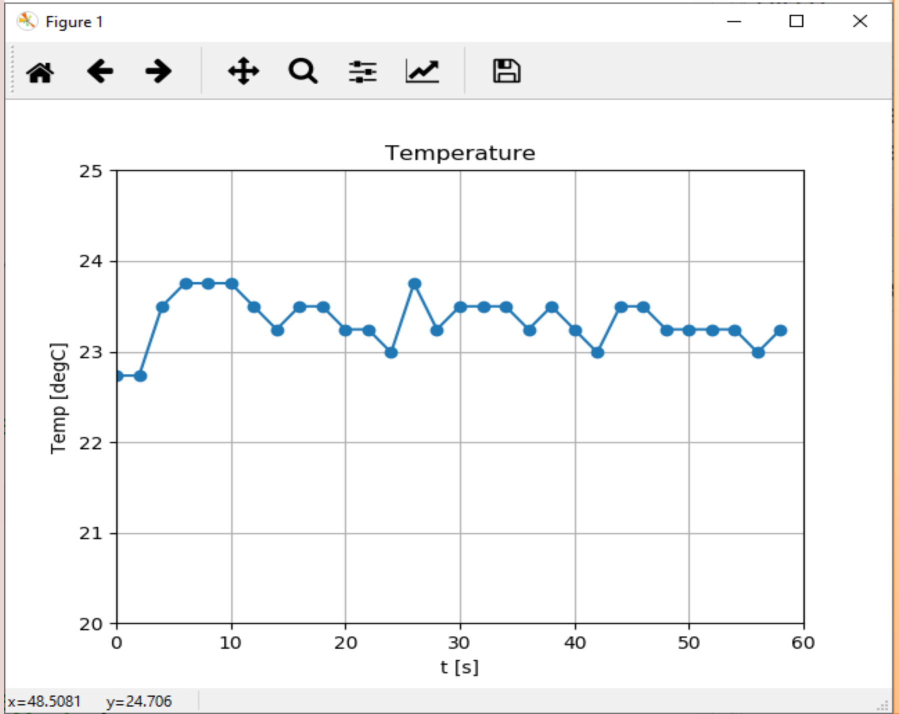
## Thermistor Application:

```
import time
import nidaqmx
import thermistor

# Initialize DAQ Device
from nidaqmx.constants import (
    TerminalConfiguration)

task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0",
            terminal_config=TerminalConfiguration.RSE)
task.start()

# Initialization
Tstop = 60 # Logging Time [seconds]
Ts = 2 # Sampling Time [seconds]
N = int(Tstop/Ts)

for k in range(N):
    # Read from DAQ Device
    Vout = task.read()

    TempC = thermistor.thermistorTemp(Vout)
    print(round(TempC,1))

    time.sleep(Ts)

task.stop
task.close()
```

# Plotting

# Plotting Temperature Data

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

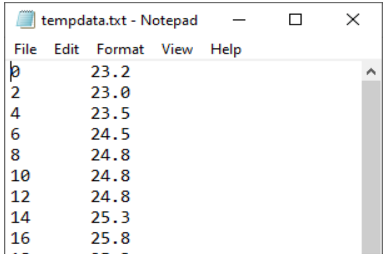


```
import numpy as np
import time
import matplotlib.pyplot as plt
import nidaqmx
import thermistor

# Initialize DAQ Device
from nidaqmx.constants import (
    TerminalConfiguration)

# Initialize Logging
Tstop = 20 # 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",
                 terminal_config=TerminalConfiguration.RSE)
task.start()

# Start Logging
for k in range(N):
    Vout = task.read()
    TempC = thermistor.thermistorTemp(Vout)
    print(round(TempC,1))
    data.append(TempC)
    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 = 30
plt.axis([0, Tstop, Tmin, Tmax])
plt.show()
```

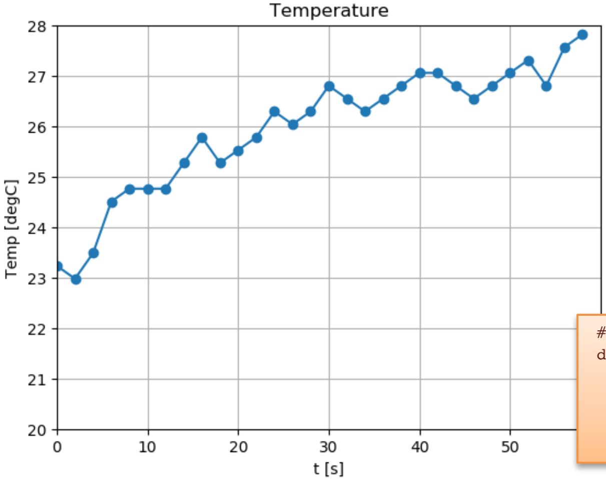
# Logging to File

# 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.



```
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
```



datalogging.py

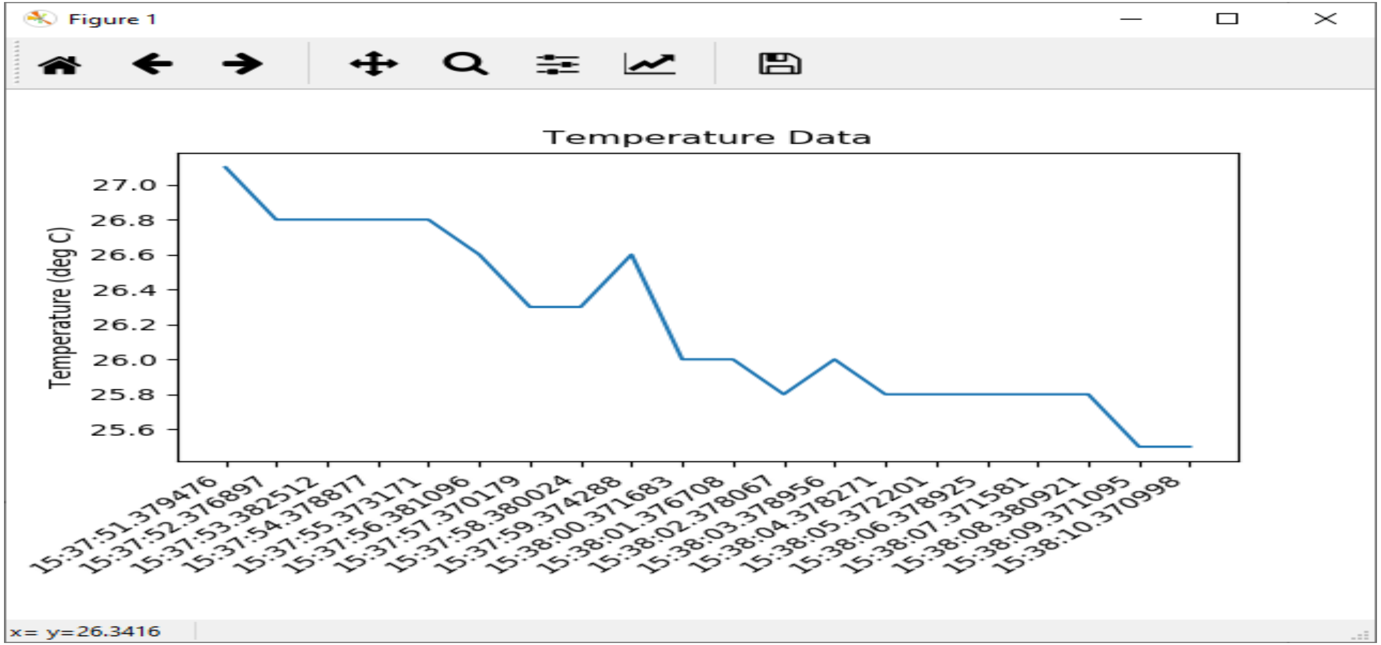
```
# Write Data to File Function
def writefiledata(file, t, x):
    time = str(t)
    value = str(round(x, 2))
    file.write(time + "\t" + value)
    file.write("\n")
```

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

# Initialize DAQ Device
from nidaqmx.constants import (
    TerminalConfiguration)

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

# Initialize Logging
Tstop = 20 # 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",
                    terminal_config=TerminalConfiguration.RSE)
task.start()

# Start Logging
for k in range(N):
    Vout = task.read()
    TempC = thermistor.thermistorTemp(Vout)
    print(round(TempC,1))
    datalogging.writefiledata(file, k*Ts, round(TempC,1))
    data.append(TempC)
    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 = 30
plt.axis([0, Tstop, Tmin, Tmax])
plt.show()
```

# Real-Time Plotting of Data

# Real-Time Plotting

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



# Python Code

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

Ts = 2 #Sampling Time in Seconds

# Initialize DAQ Device
from nidaqmx.constants import (
    TerminalConfiguration)

# 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",
            terminal_config=TerminalConfiguration.RSE)
task.start
```

```
# This function is called periodically from FuncAnimation
def readdaq(i, xs, ys):
    Vout = task.read()
    TempC = thermistor.thermistorTemp(Vout)

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

    # 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,"-o")

    # 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=Ts*1000)
plt.show()
task.stop
```

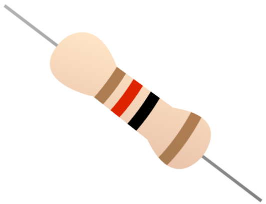
https://www.halvorsen.blog

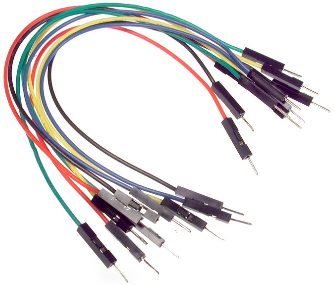
# Temperature with Alarm

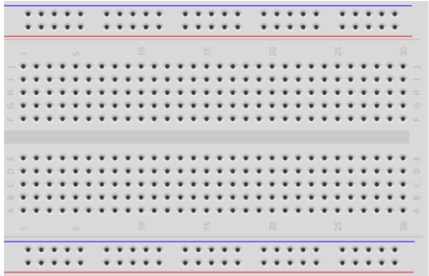
Hans-Petter Halvorsen

# Necessary Equipment

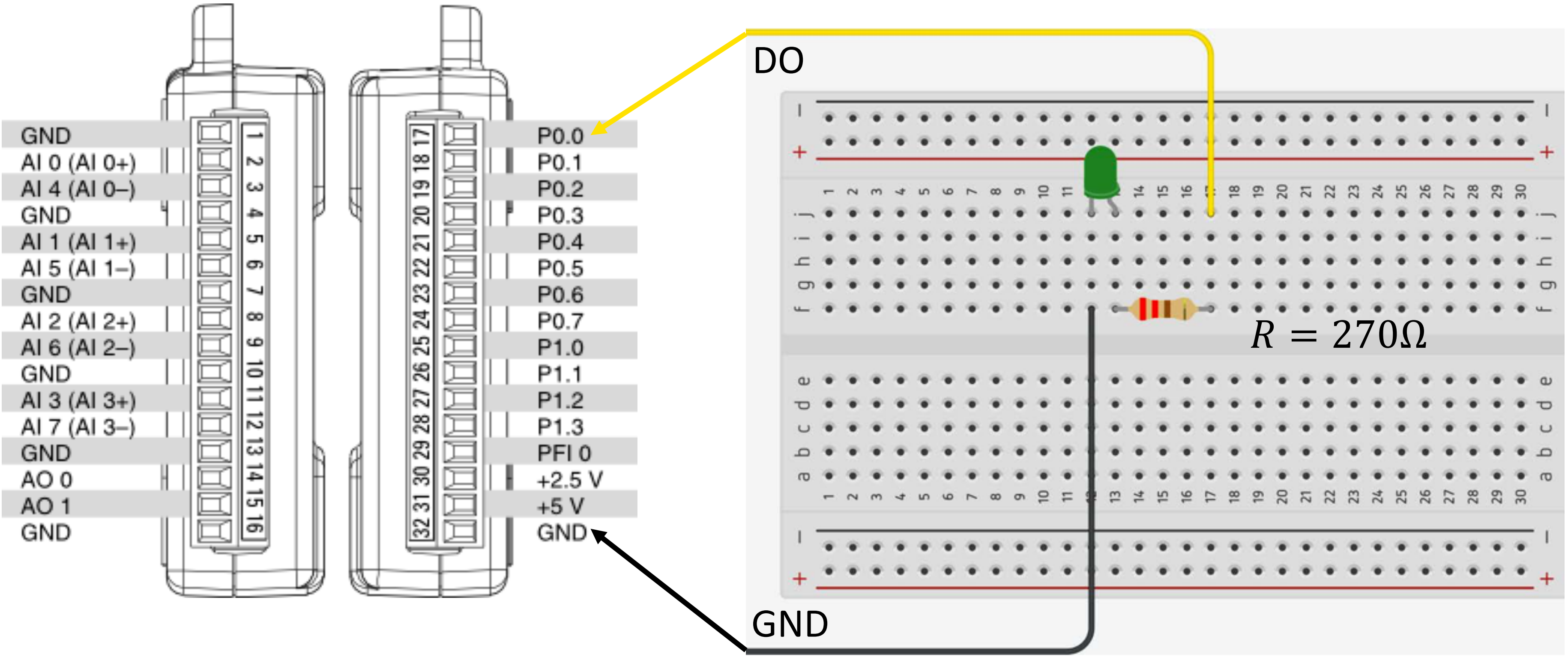
- PC
- DAQ Module, e.g., USB-6008
- Breadboard
- Thermistor
- LED
- Resistor, $R = 270\Omega$
- Wires (Jumper Wires)







# LED Wiring

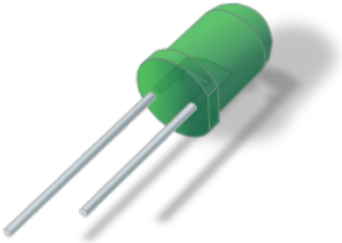


$R = 270\Omega$

# Python Code

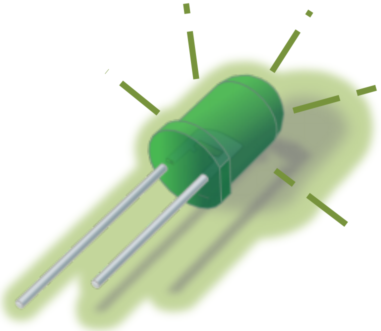
Temperature > Limit?

No



LED OFF

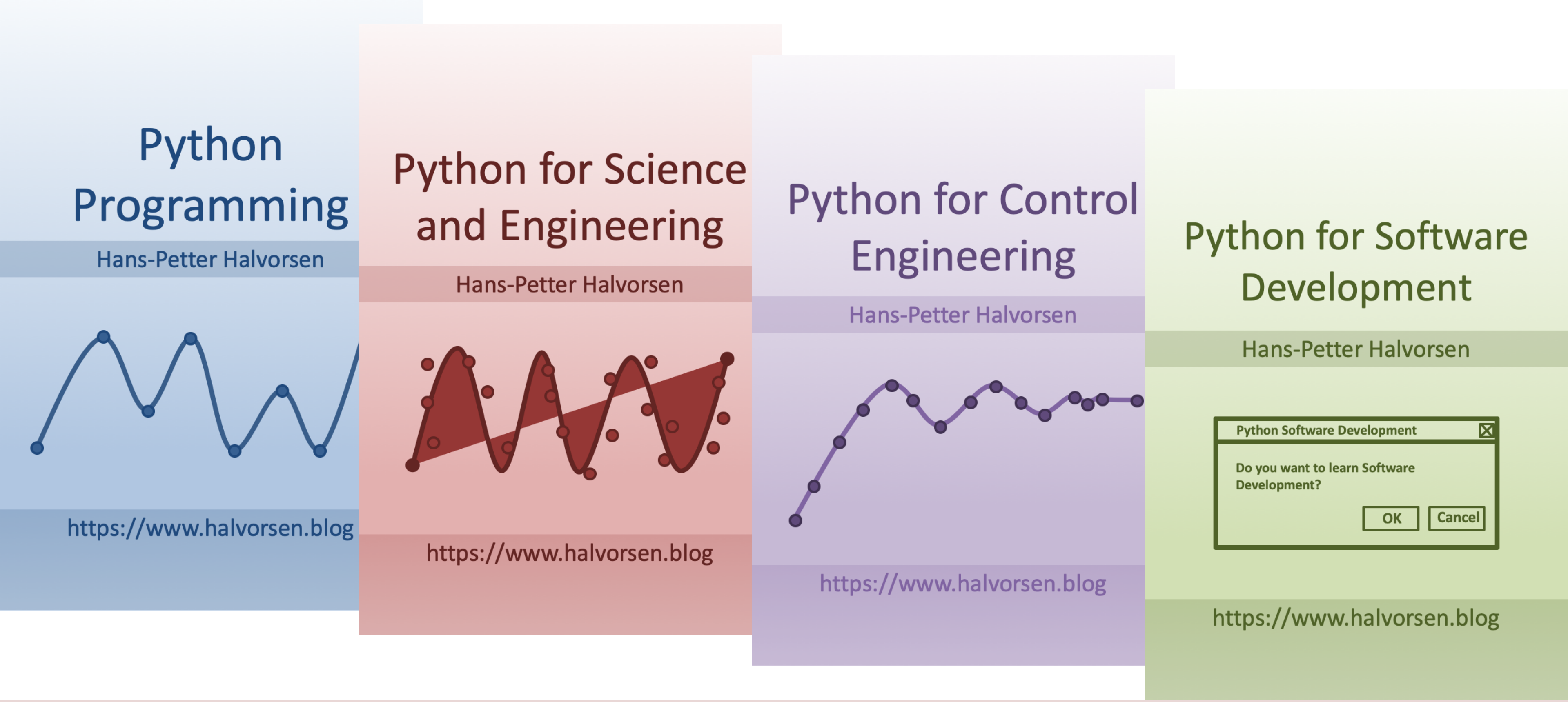
Yes



LED ON

```
import nidaqmx
import time
import thermistor

# Initialize DAQ Device
from nidaqmx.constants import (
    TerminalConfiguration)

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

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

alarmlimit = 28 #degrees Celsius

Ts = 2
N = 10
# Start Logging
for k in range(N):
    Vout = task_ai.read()
    TempC = thermistor.thermistorTemp(Vout)
    print(round(TempC,1))

    if TempC >= alarmlimit:
        task_do.write(True)
        print("Alarm!")
    else:
        task_do.write(False)
        print("OK")

    time.sleep(Ts)

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

# Additional Python Resources



Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Science and Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Control Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Software Development

Hans-Petter Halvorsen

Python Software Development

Do you want to learn Software Development?

https://www.halvorsen.blog

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