Sensors and Actuators in Python

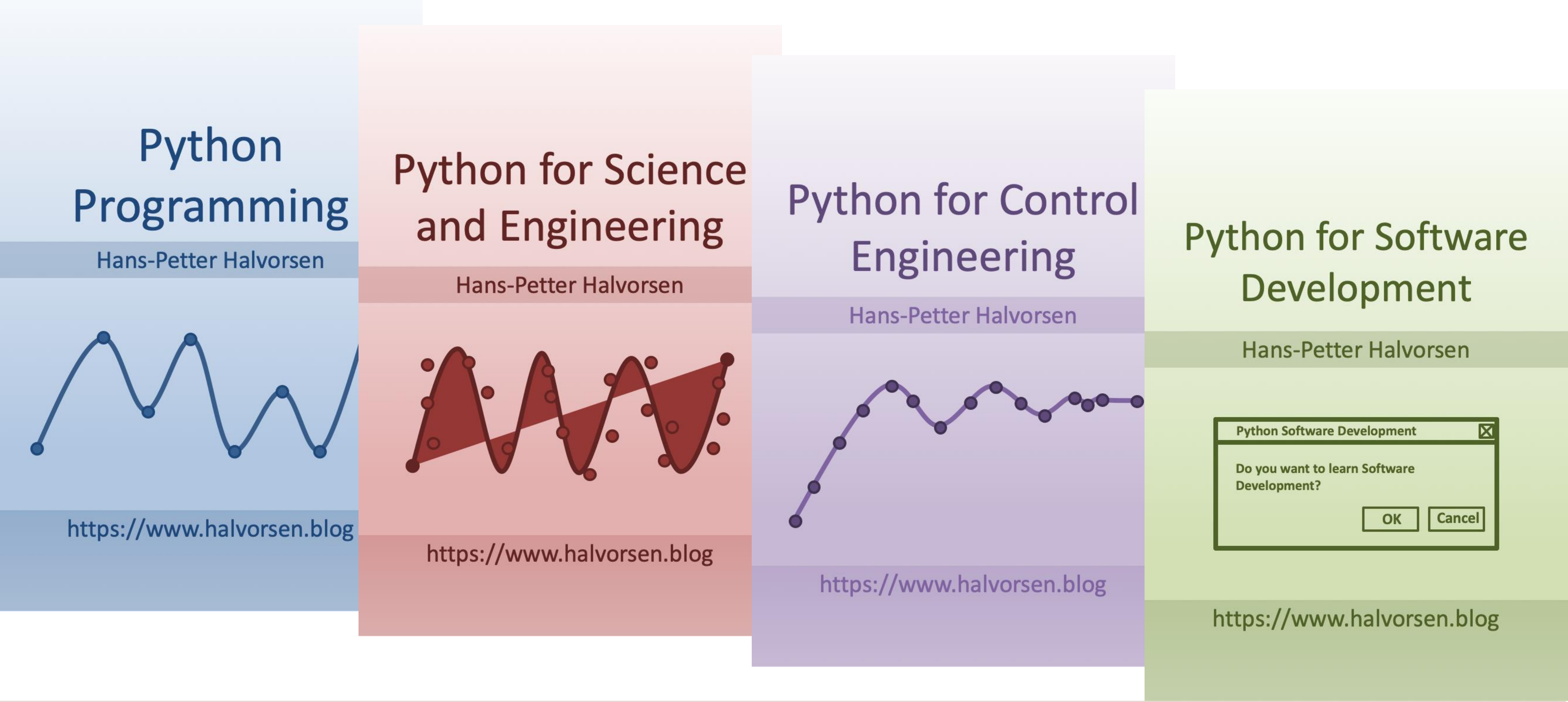
Exemplified by using NI USB-6008 I/O Module

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/

Contents

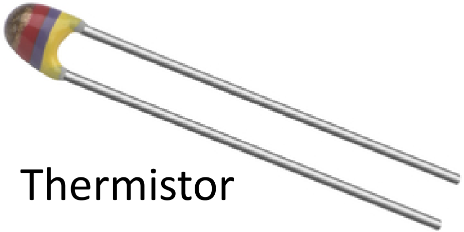
- DAQ and I/O Modules
- NI-DAQ
- Sensors and Actuators
- Python Examples
  - LED, TMP36 Temperature, Thermistor, Push Button/Switch, Light Sensor

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)

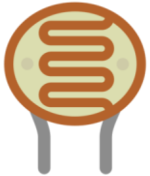
Push Button



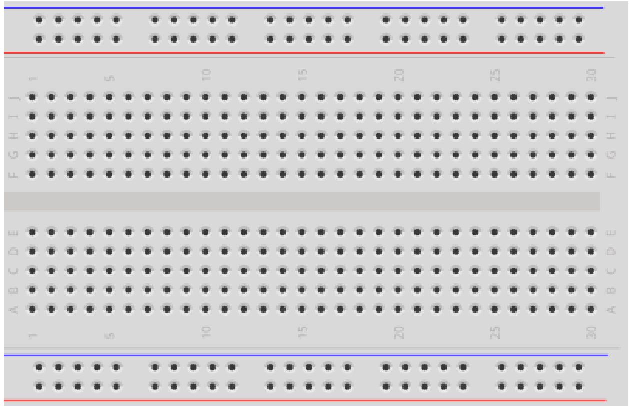
Thermistor

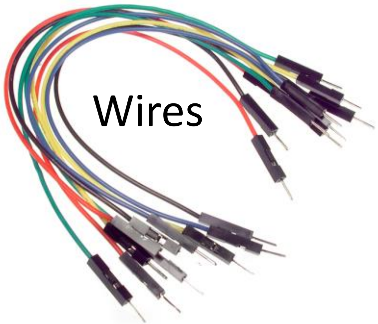
TMP36

Light Sensor

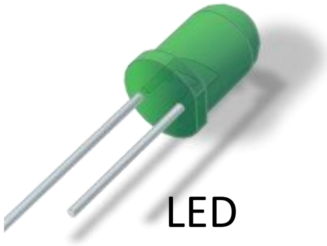


Breadboard

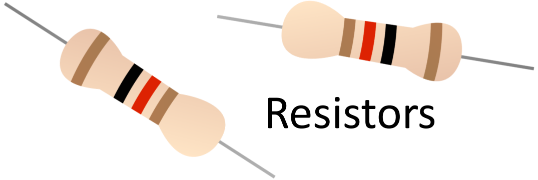




Wires



LED

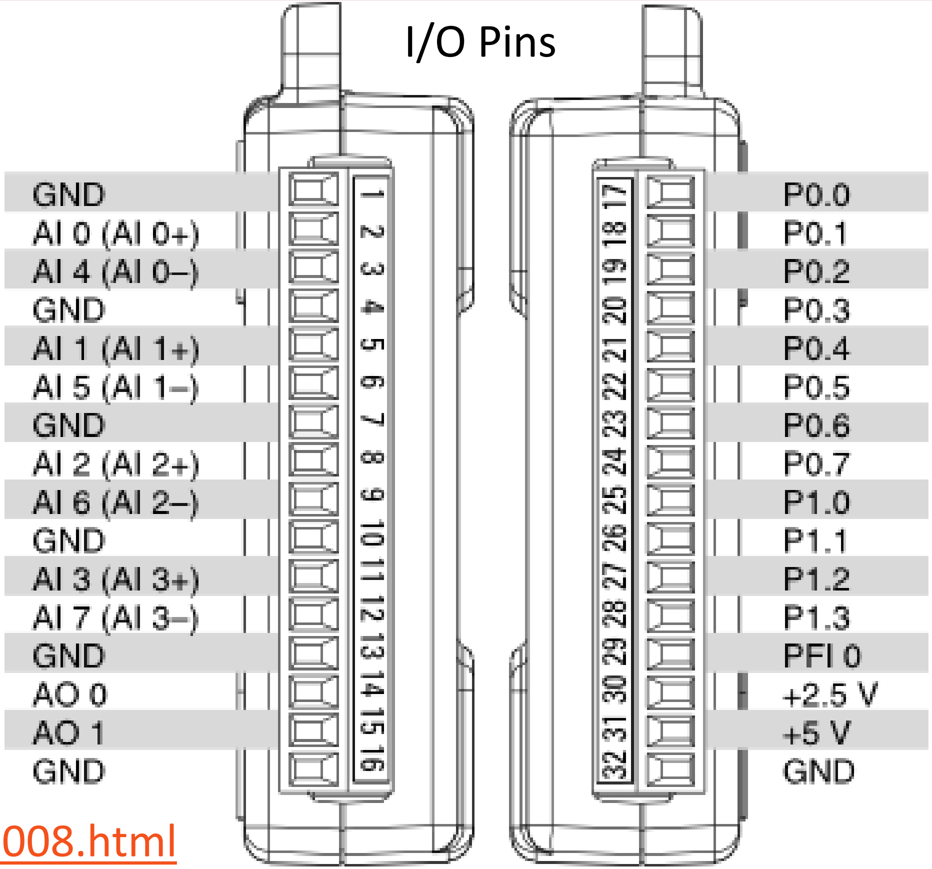


Resistors

NI USB-6008

We will use NI USB-6008 in our examples

I/O Pins



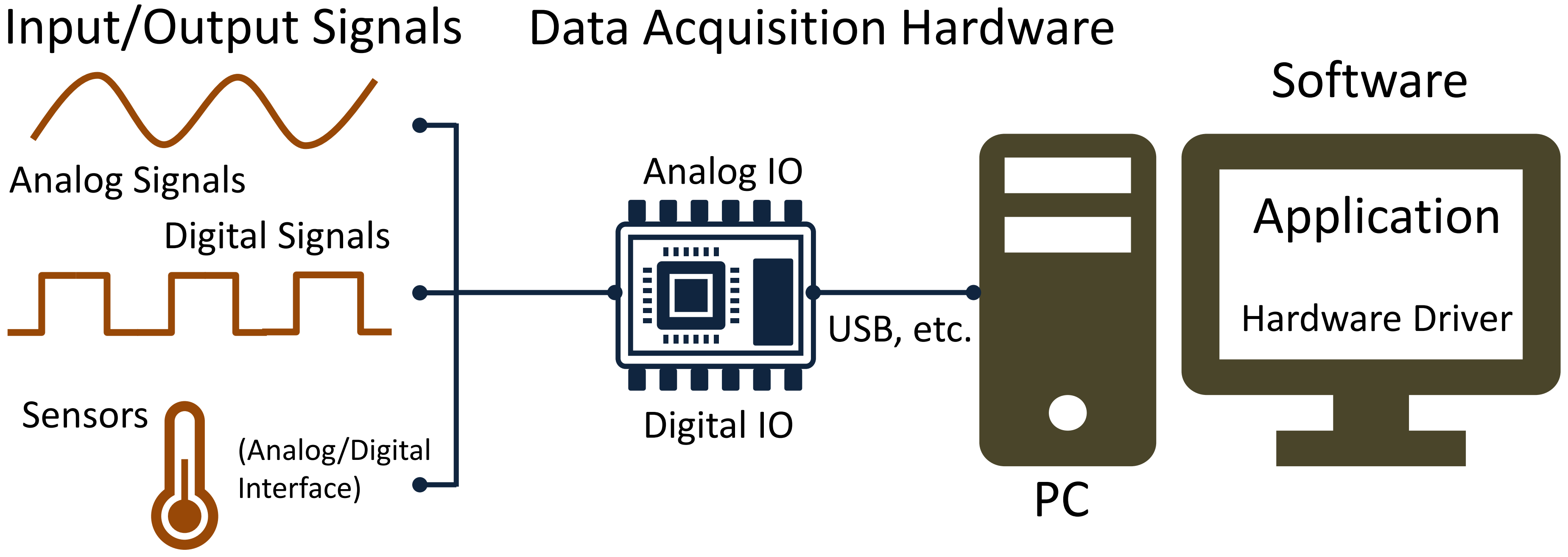
http://www.ni.com/en-no/support/model.usb-6008.html

NI DAQ Device with Python

How to use a NI DAQ Device with Python

Python Application	Your Python Program
nidaqmx Python Package (Free)	Python Library/API for Communication with NI DAQmx Driver
Python (Free)	Python Programming Language
NI DAQmx (Free)	Hardware Driver Software
NI DAQ Hardware	In this Tutorial we will use NI TC-01 Thermocouple

DAQ System

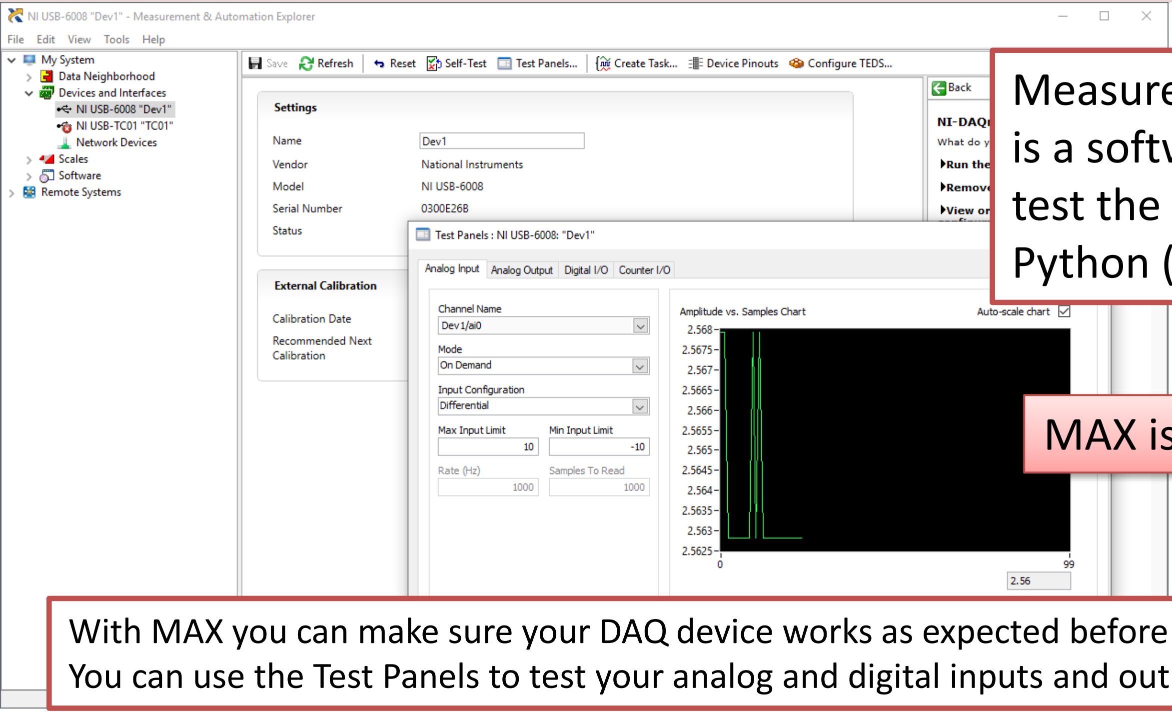


Input/Output Signals
Data Acquisition Hardware
Software
Analog Signals
Digital Signals
Sensors
(Analog/Digital
Interface)
Analog IO
Digital IO
USB, etc.
Application
Hardware Driver
PC

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)



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

Anaconda Prompt

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

Anaconda Prompt

```
(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>
```

I/O Signals

Hans-Petter Halvorsen

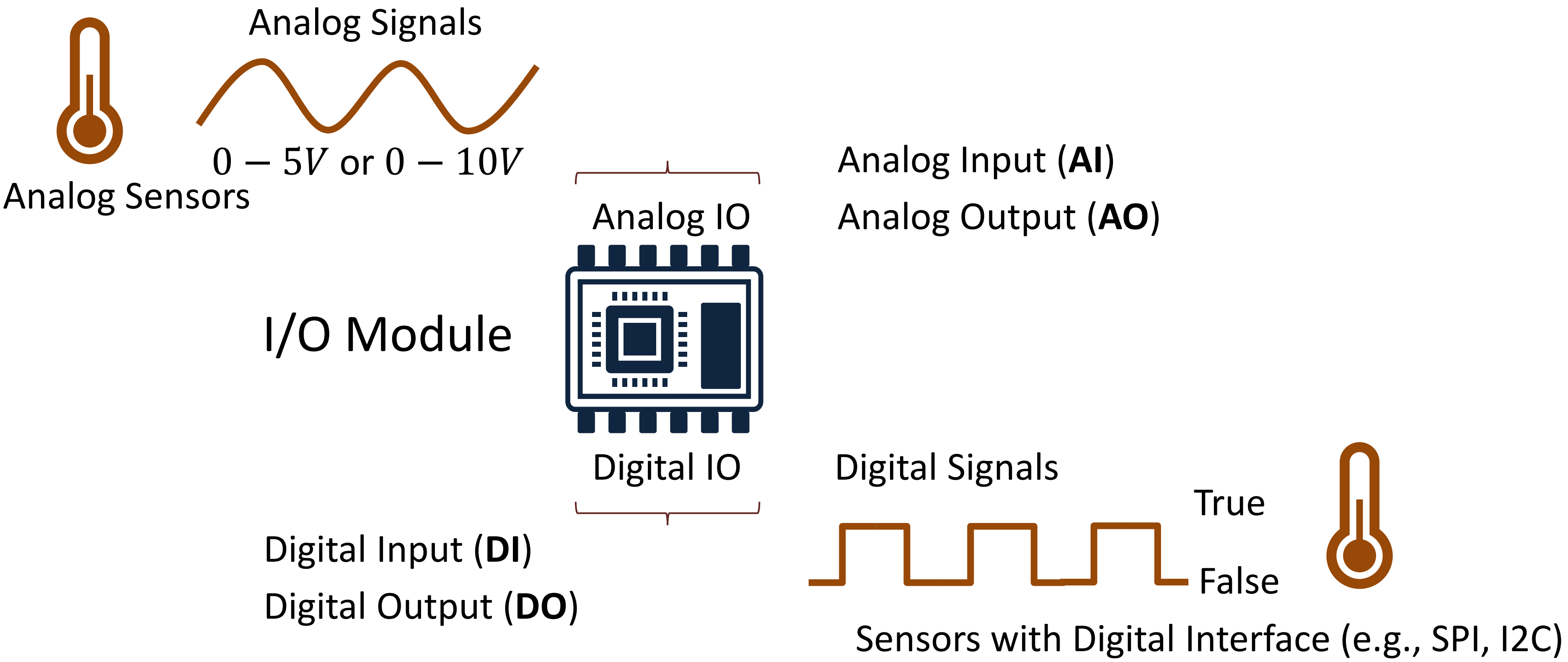
I/O Signals

Using a DAQ device we have 4 options

- **Analog Out** (Write) - AO
- **Analog In** (Read) - AI
- **Digital Out** (Write) - DO
- **Digital In** (Read) - DI

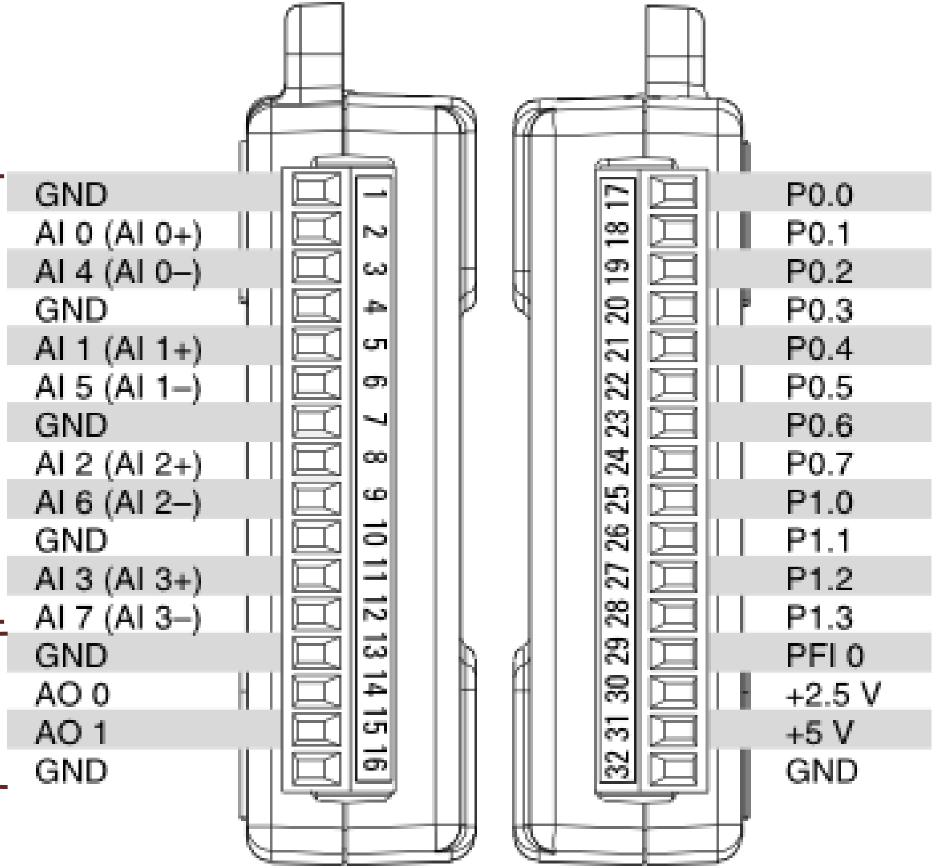
We will show some basic examples in each of these categories

I/O Module



Analog Signals
$0 - 5V$ or $0 - 10V$
Analog Sensors
Analog IO
Analog Input (**AI**)
Analog Output (**AO**)
I/O Module
Digital IO
Digital Signals
True
False
Digital Input (**DI**)
Digital Output (**DO**)
Sensors with Digital Interface (e.g., SPI, I2C)

NI USB-6008



Analog In - AI

Analog Out - AO

Digital Out - DO

Digital In - DI

All Digital Channels can be configured either to be Digital Out or Digital In

Analog Out (Write)

```
import nidaqmx

task = nidaqmx.Task()
task.ao_channels.add_ao_voltage_chan('Dev1/ao0','mychannel',0,5)
task.start()

value = 2
task.write(value)

task.stop()
task.close()
```

You can, e.g., use a **Multimeter** in order to check if the the program outputs the correct value

Analog In (Read)

```
import nidaqmx

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

value = task.read()
print(value)

task.stop
task.close()
```

Digital Out (Write)

```
import nidaqmx

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

value = True
task.write(value)

task.stop
task.close()
```

`value = True`

We measure $\sim 5V$ using a Multimeter

`value = False`

We measure $\sim 0V$ using a Multimeter

Digital In (Read)

```
import nidaqmx

task = nidaqmx.Task()
task.di_channels.add_di_chan("Dev1/port0/line1")
task.start()

value = task.read()
print(value)

task.stop
task.close()
```

Sensors and Actuators

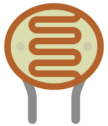
Hans-Petter Halvorsen

Sensors and Actuators

Temperature Sensor

Push Button

Light Sensor

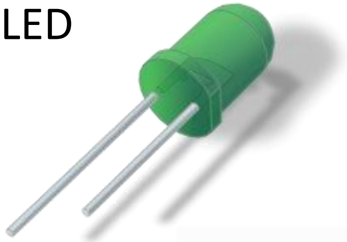


Potentiometer

Analog/Digital Inputs

Signals from the surroundings

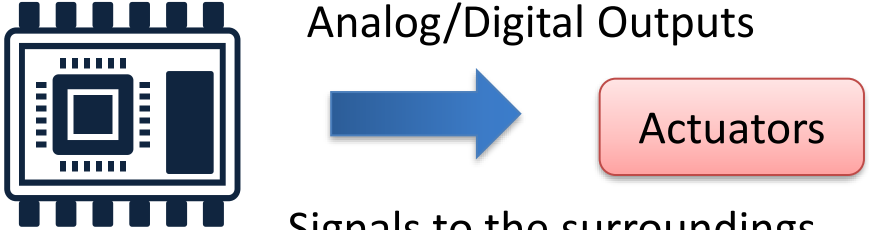
LED



Motor

Buzzer

Analog/Digital Outputs



Signals to the surroundings

Sensors and Actuators

- A **Sensor** is a converter that measures a physical size and converts it to a signal that can be read by an instrument, data acquisition device, or an Arduino.
  Examples: temperature sensor, pressure sensor, etc.
- An **Actuator** is a kind of motor that moves or controls a mechanism or system. It is powered by an energy source, typical electric current, hydraulic fluid pressure, or air pressure, and converts this energy into motion.
  Examples: Engine, Pump, Valve, etc.

Sensors and Actuators

Actuator

LED

Sensor

TMP36
Temperature Sensor

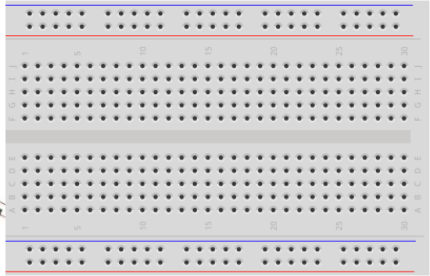
LED with Python

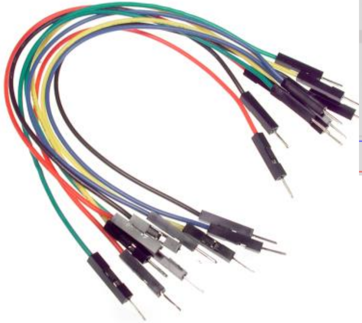
Light-Emitting Diode (LED)

Hans-Petter Halvorsen

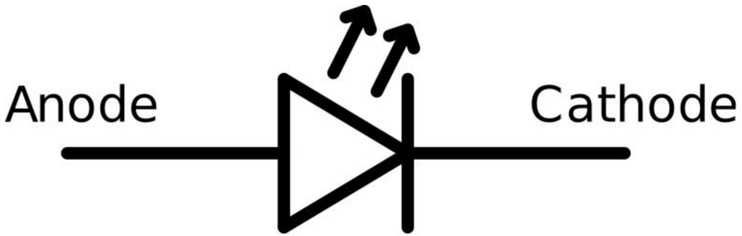
Necessary Equipment

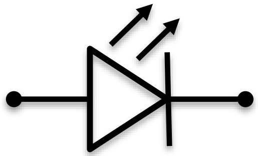
- DAQ Device (e.g., USB-6008)
- Breadboard
- LED
- Resistor, $R = 270\Omega$
- Wires (Jumper Wires)

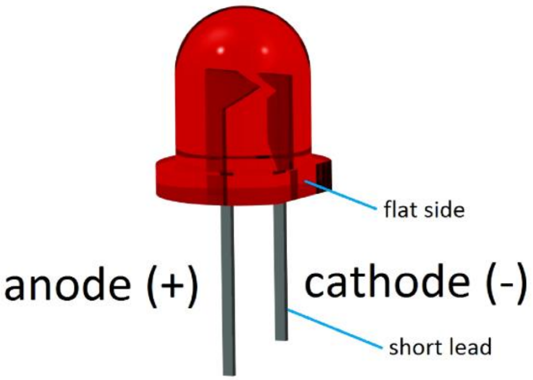


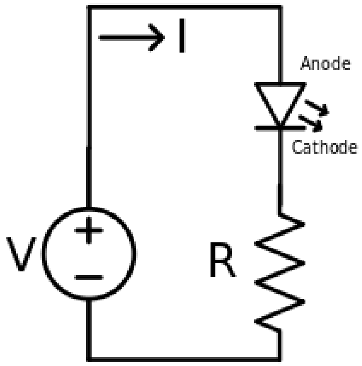


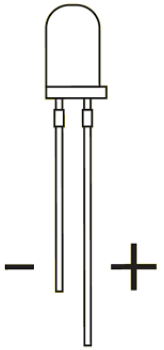
LED





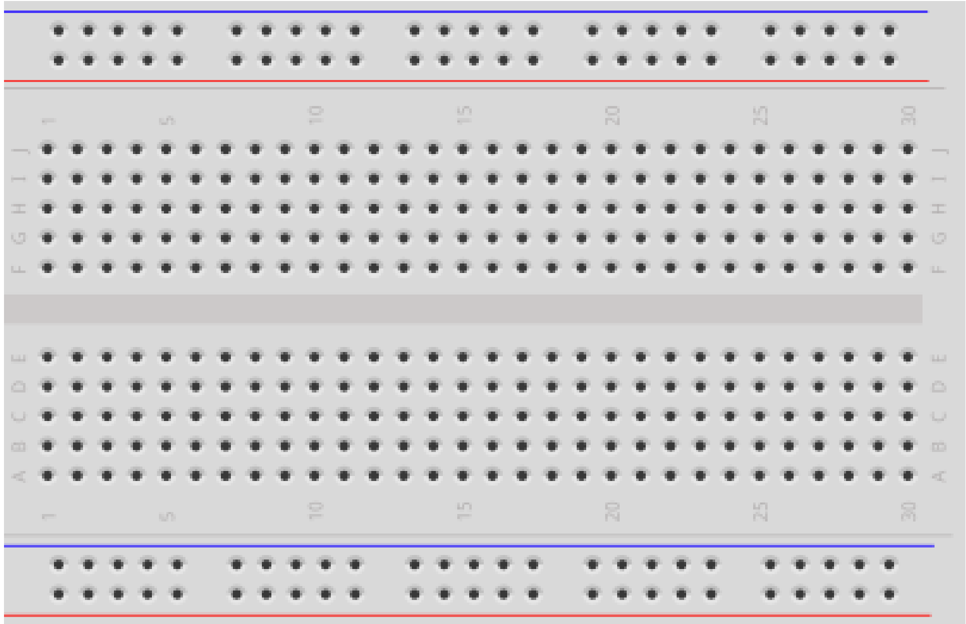




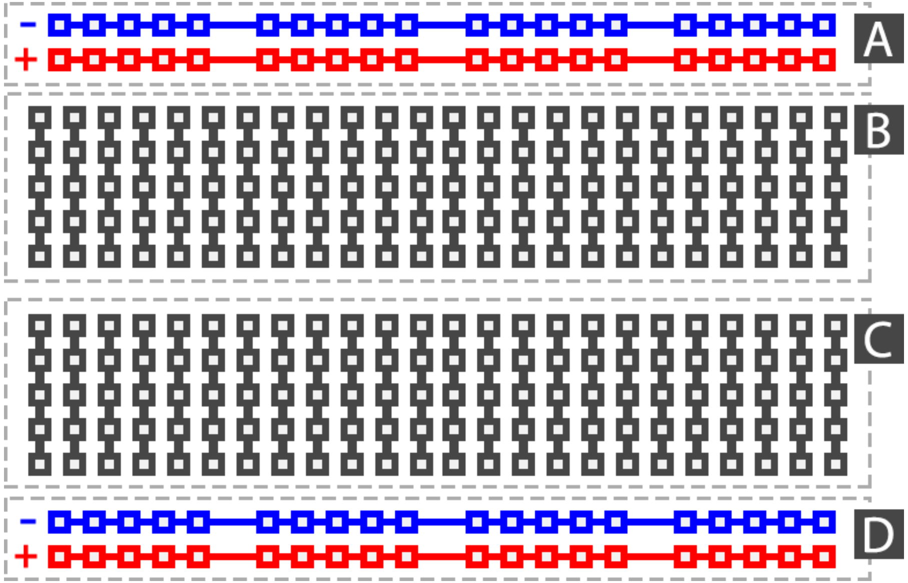


[Wikipedia]

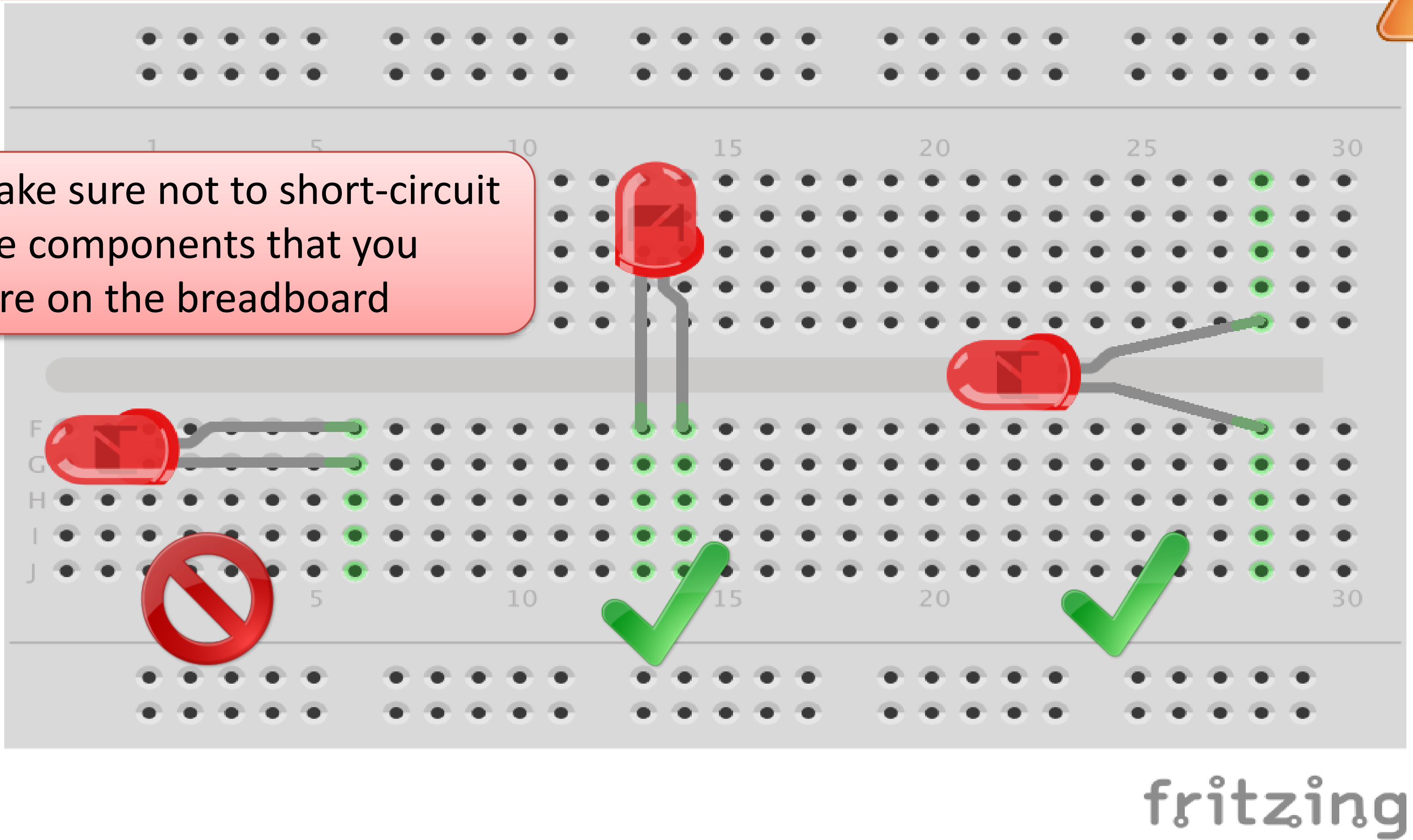
Breadboard



A breadboard is used to wire electric components together

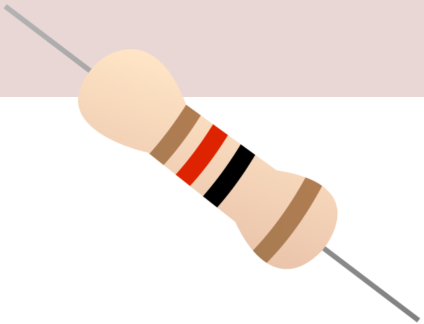


Breadboard Wiring



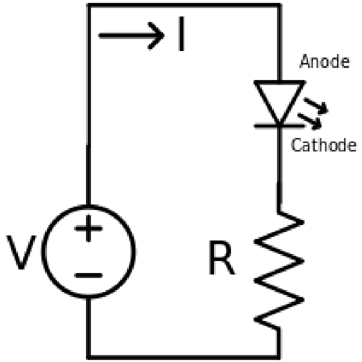
Make sure not to short-circuit the components that you wire on the breadboard

Resistors



Resistance is measured in Ohm (Ω)

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



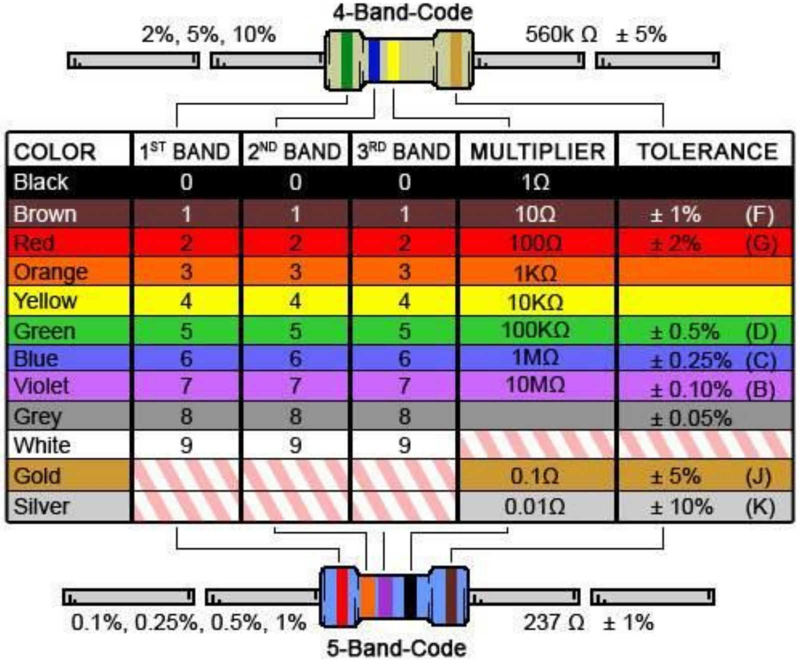
The resistance can be found using **Ohms Law**

$$U = RI$$

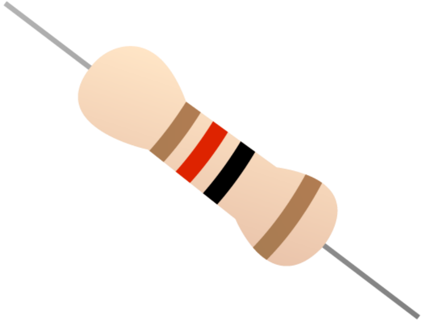
https://en.wikipedia.org/wiki/Resistor

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/

Why do you need a Resistor?

If the current becomes to 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:

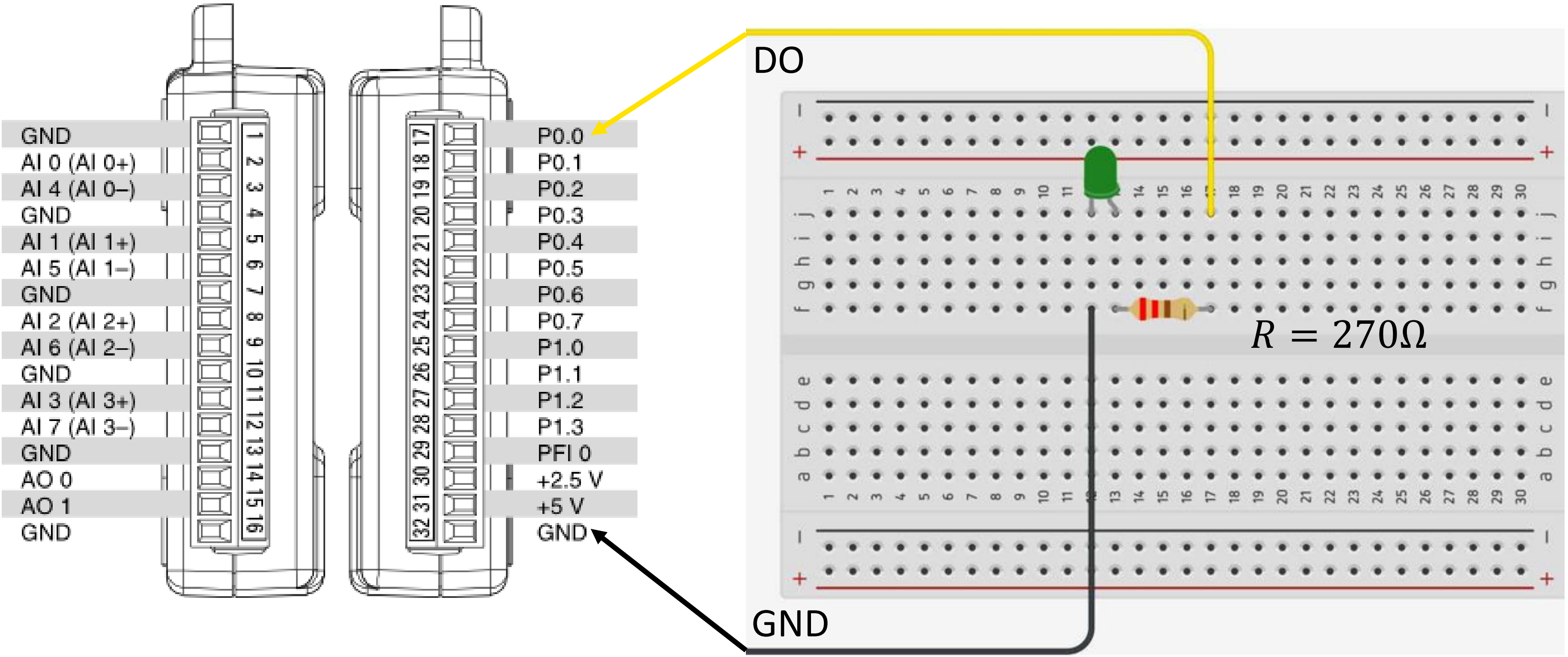
$$U = RI$$

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

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

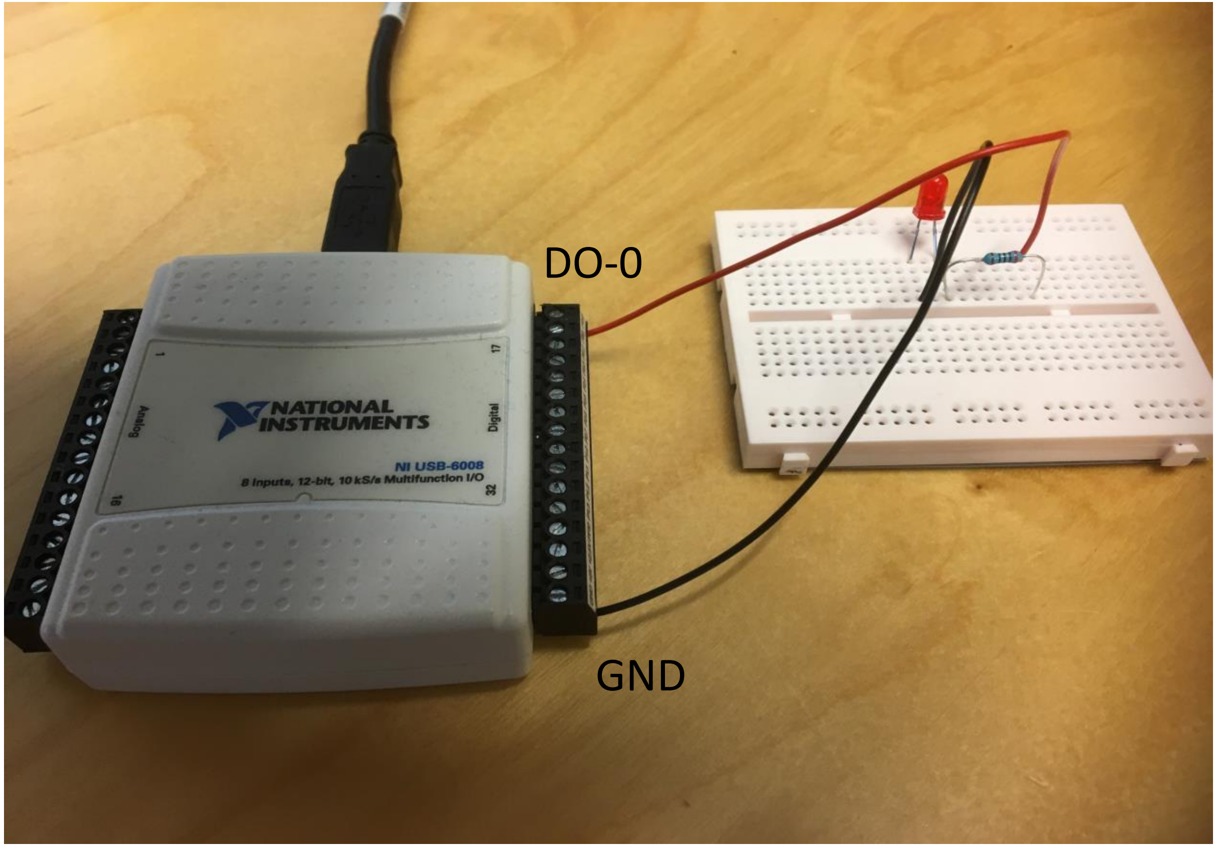
The Resistor needed will be $R = \frac{5V}{0.02A} = 250\Omega$. Resistors with R=250Ω is not so common, so we can use the closest Resistors we have, e.g., 270Ω

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
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
GND
$R = 270\Omega$

Hardware Setup



$R = 270\Omega$

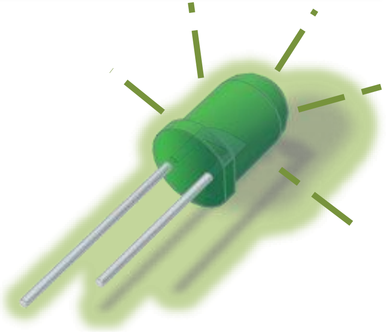
Python Example

Digital Out

In this basic Example we turn on a LED

`value = False`

`value = True`



```
import nidaqmx

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

value = True
task.write(value)

task.stop
task.close()
```

Blinking LED

Digital Out

```
import nidaqmx
import time

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

value = True
N = 10
blinktime = 1 #seconds

for k in range(N):
    task.write(value)
    time.sleep(blinktime)
    value = not value

task.stop; task.close()
```

Brightness

The Digital Out (DO) channels are either False (0V) or True (5V).

To control the Brightness of the LED we need to use an **Analog Out (AO) channel**

In this Example we increase the Brightness of the LED step by step from 0V, 0.1V, 0.2V, …5V

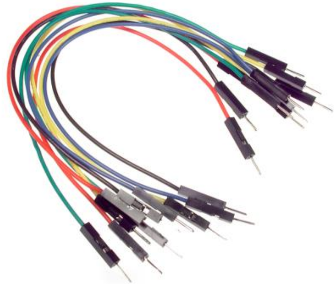
```
import numpy as np
import nidaqmx
import time

task = nidaqmx.Task()

task.ao_channels.add_ao_voltage_chan('Dev1/ao0',
                                     'mychannel',0,5)
task.start()

start=0; stop=5.1; step=0.1
brightness = np.arange(start, stop, step)

for brightlevel in brightness:
    task.write(brightlevel)
    print("brightlevel =", brightlevel, "V")
    time.sleep(0.2)

task.write(0)
task.stop; task.close()
```

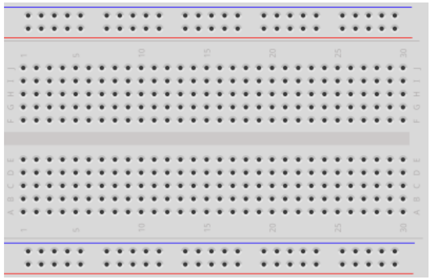
TMP36 Temperature with Python

Hans-Petter Halvorsen

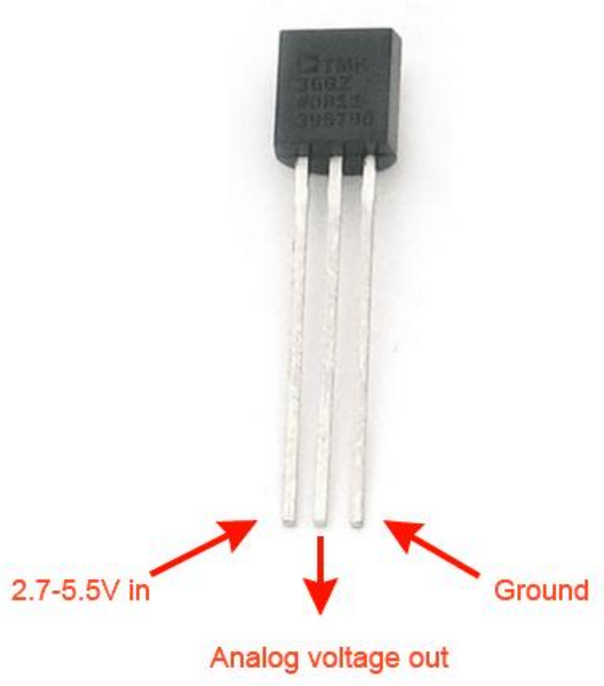
Necessary Equipment

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





TMP36 Temperature

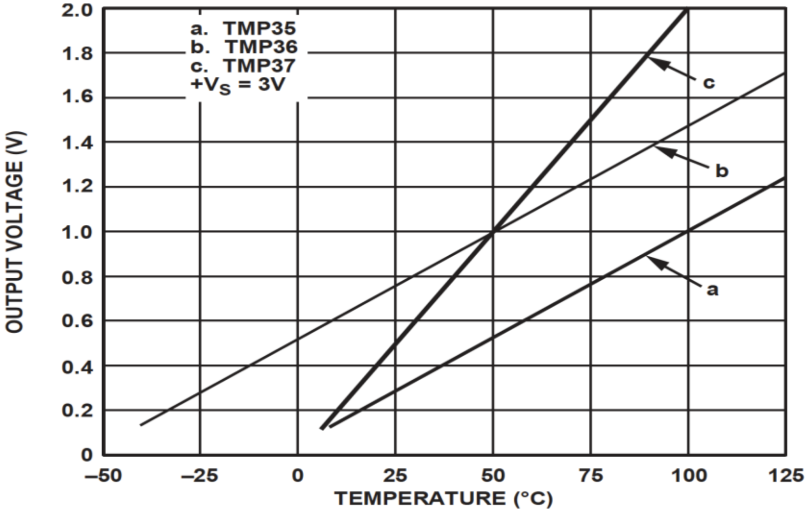


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

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



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:

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

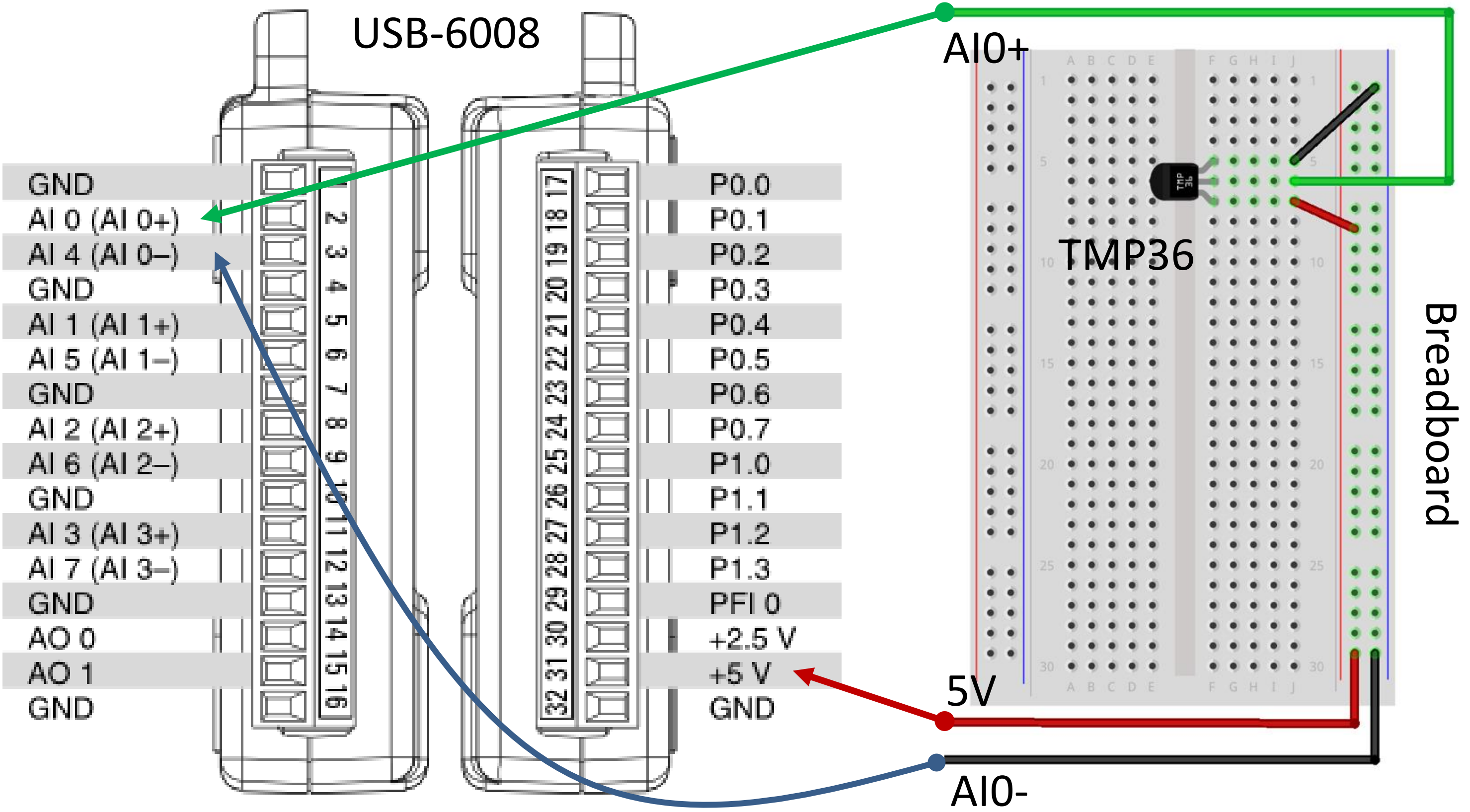
This gives:

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

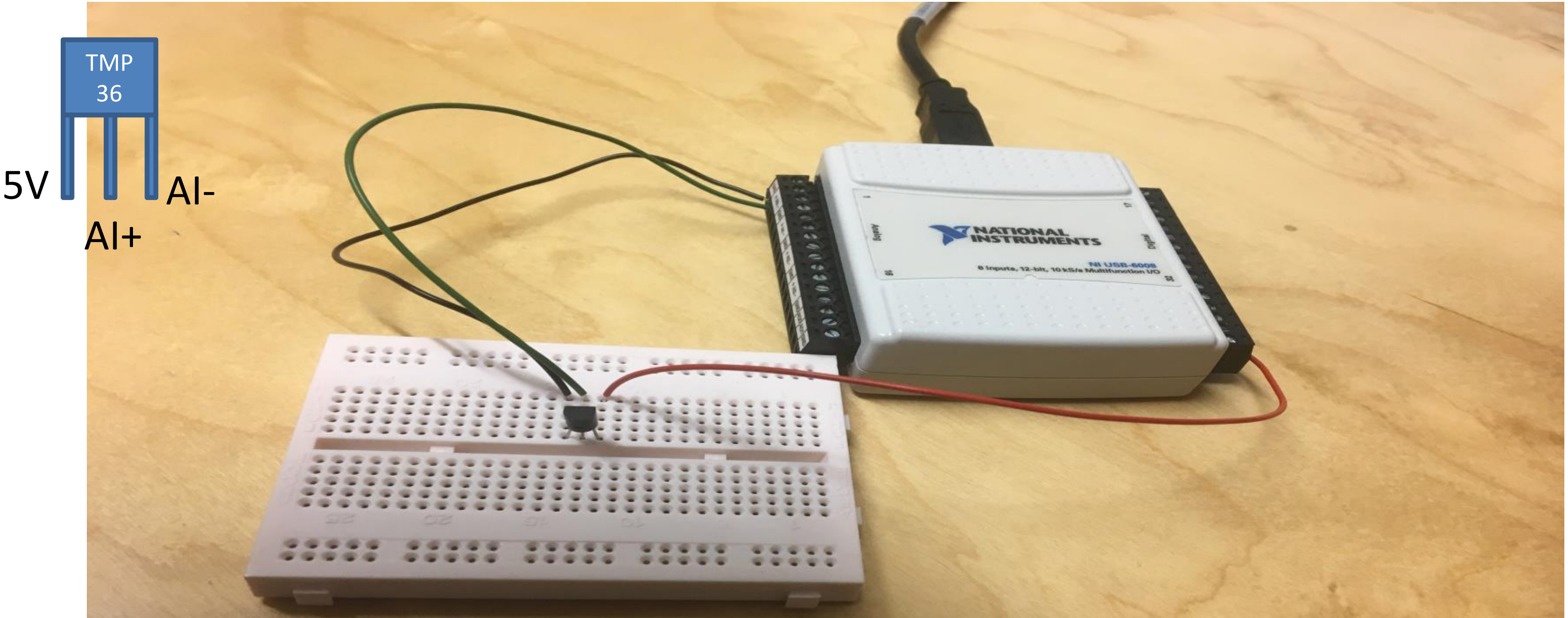
Then we get the following formula:

$$y = 100x - 50$$

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
Breadboard
5V
AI0-

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

Analog In

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

Formula converting from Voltage to Degrees Celsius:

$$y = 100x - 50$$

```
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()
```

For Loop Example

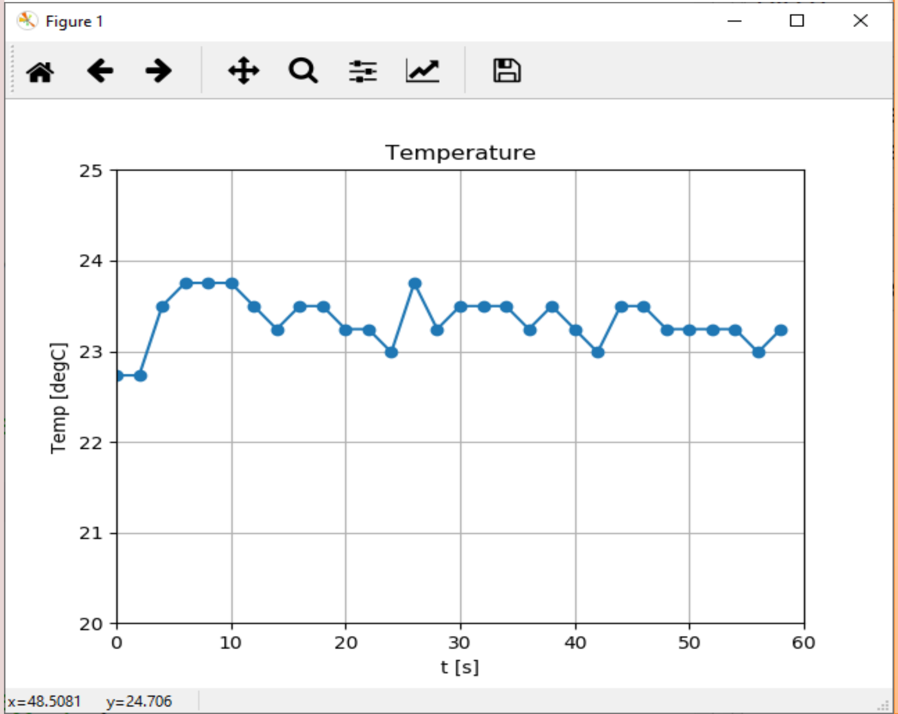
Analog In

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

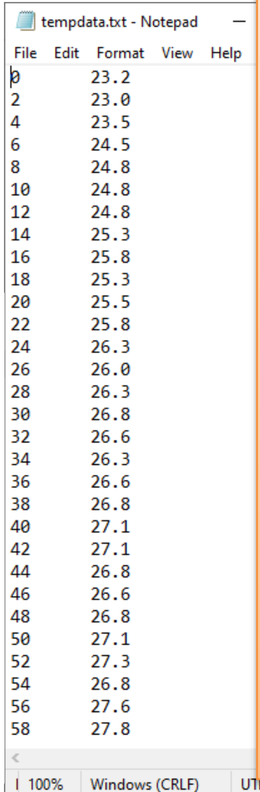
In this Example we read data from the sensor within a For Loop and Plot the Data using matplotlib



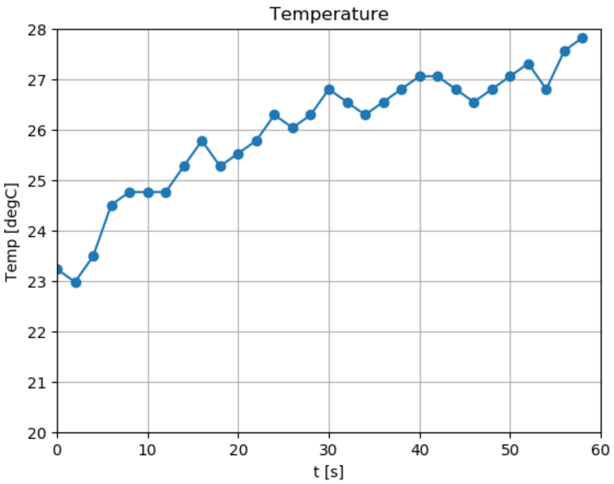

```
import numpy as np
import time
import matplotlib.pyplot as plt
import nidaqmx

# 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 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
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




```
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()
```

Sensors and Actuators with Python

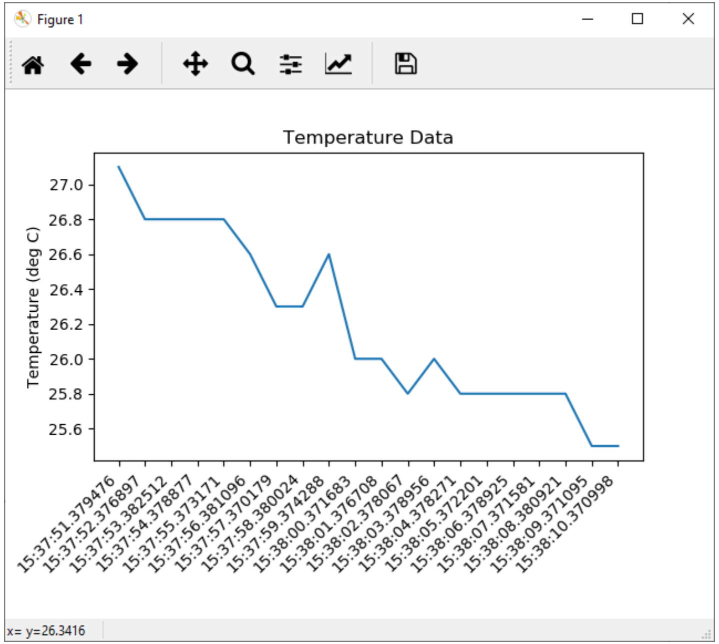
Exemplified by using NI USB-6008 I/O Module

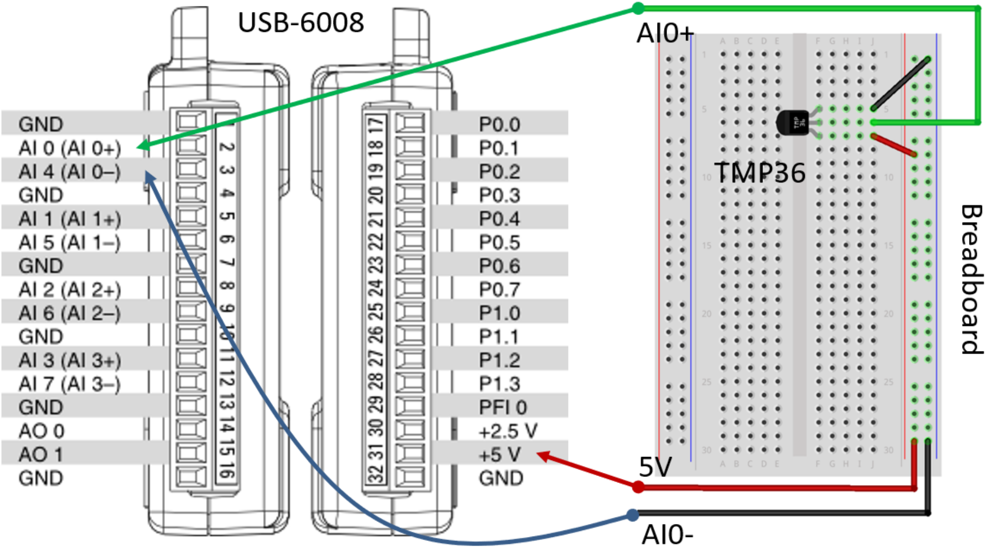
Hans-Petter Halvorsen

Real-Time Plotting of Data

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
```

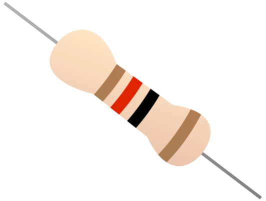
```
# This function is called periodically from FuncAnimation
def readdaq(i, xs, ys):

    #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
```

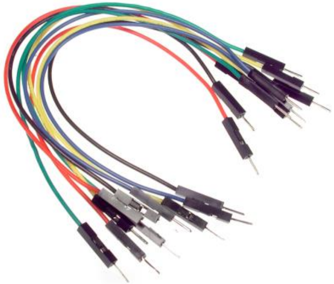
Temperature with Alarm

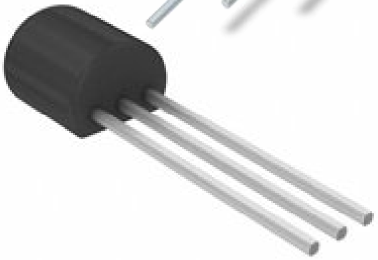
Hans-Petter Halvorsen

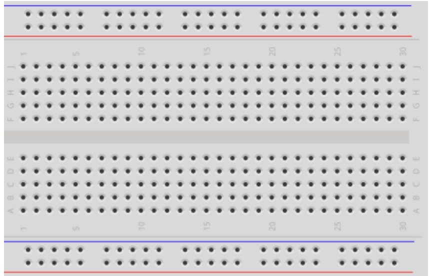
Necessary Equipment

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

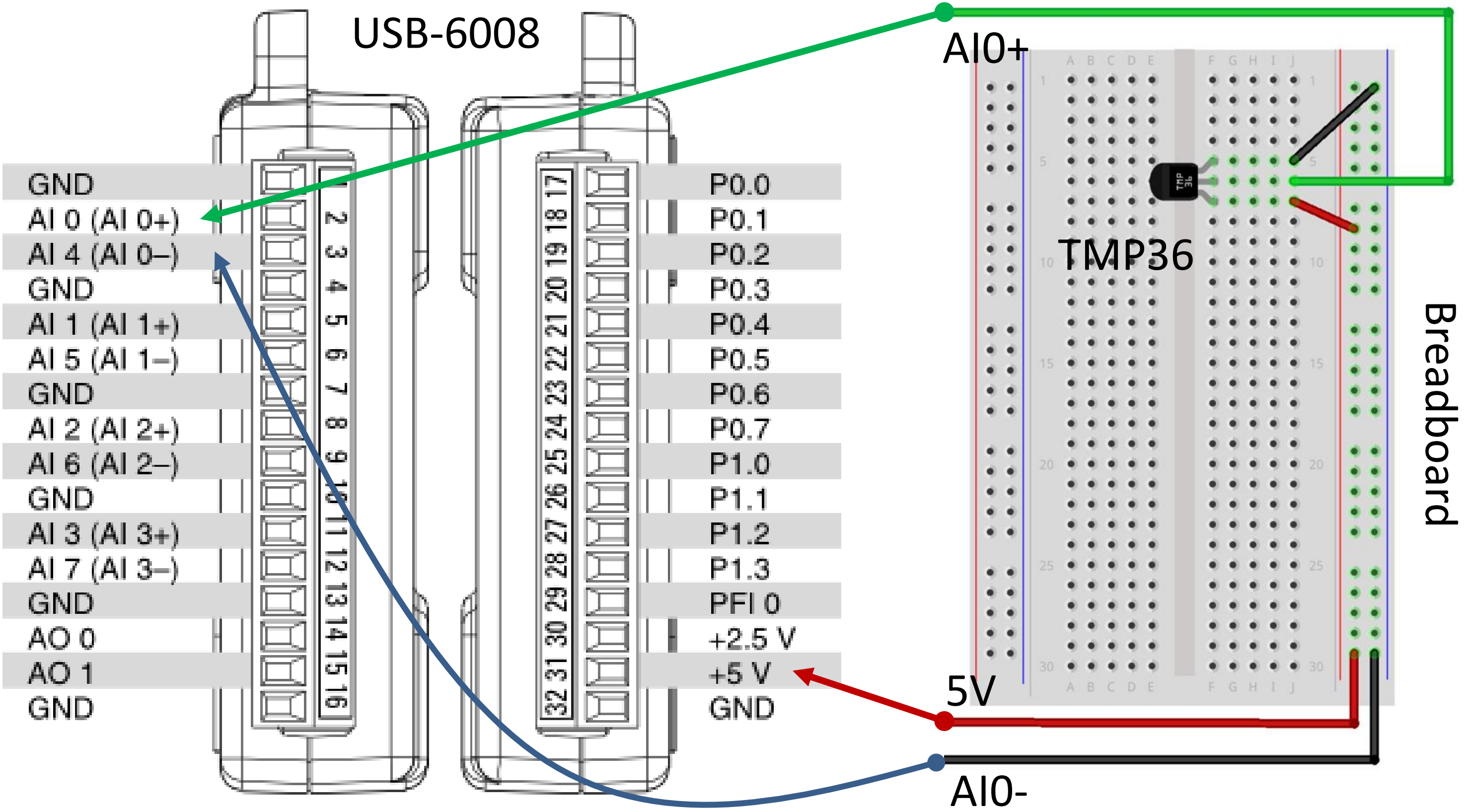




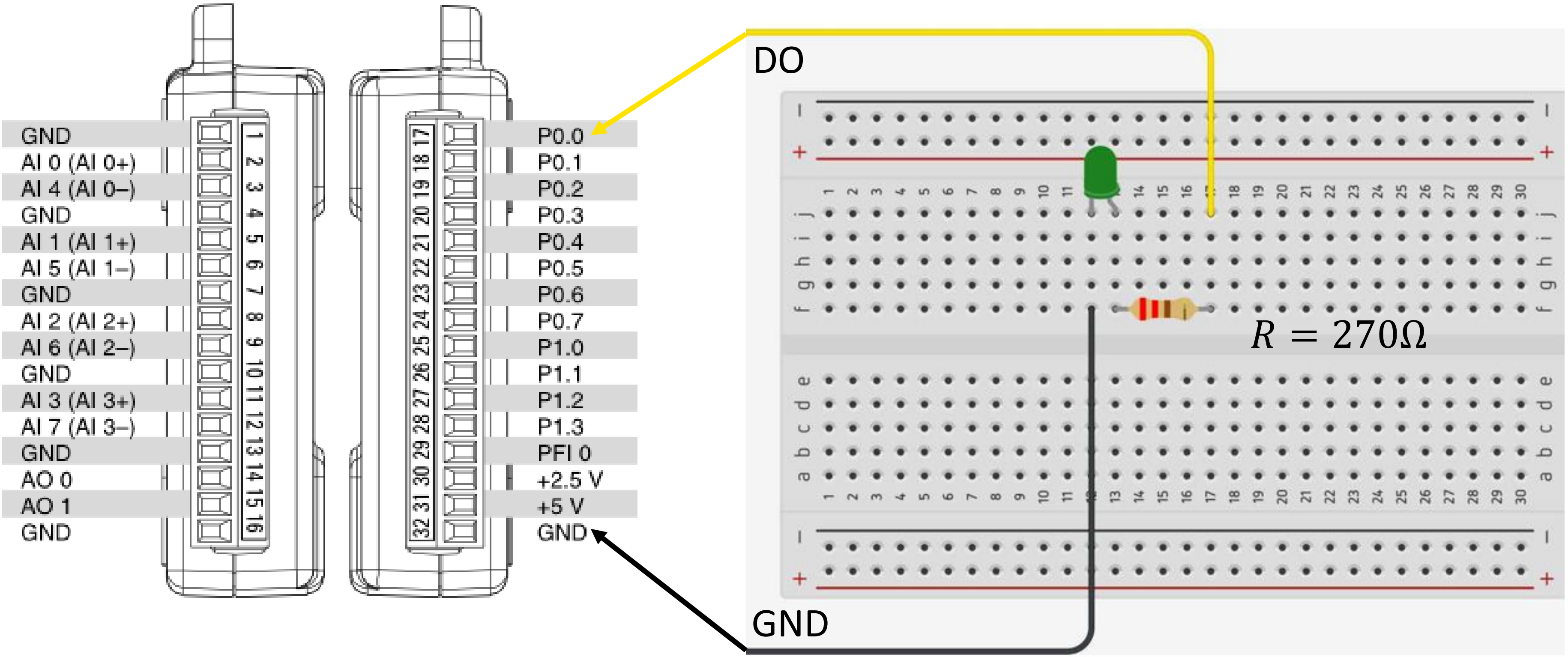




TMP36 Wiring

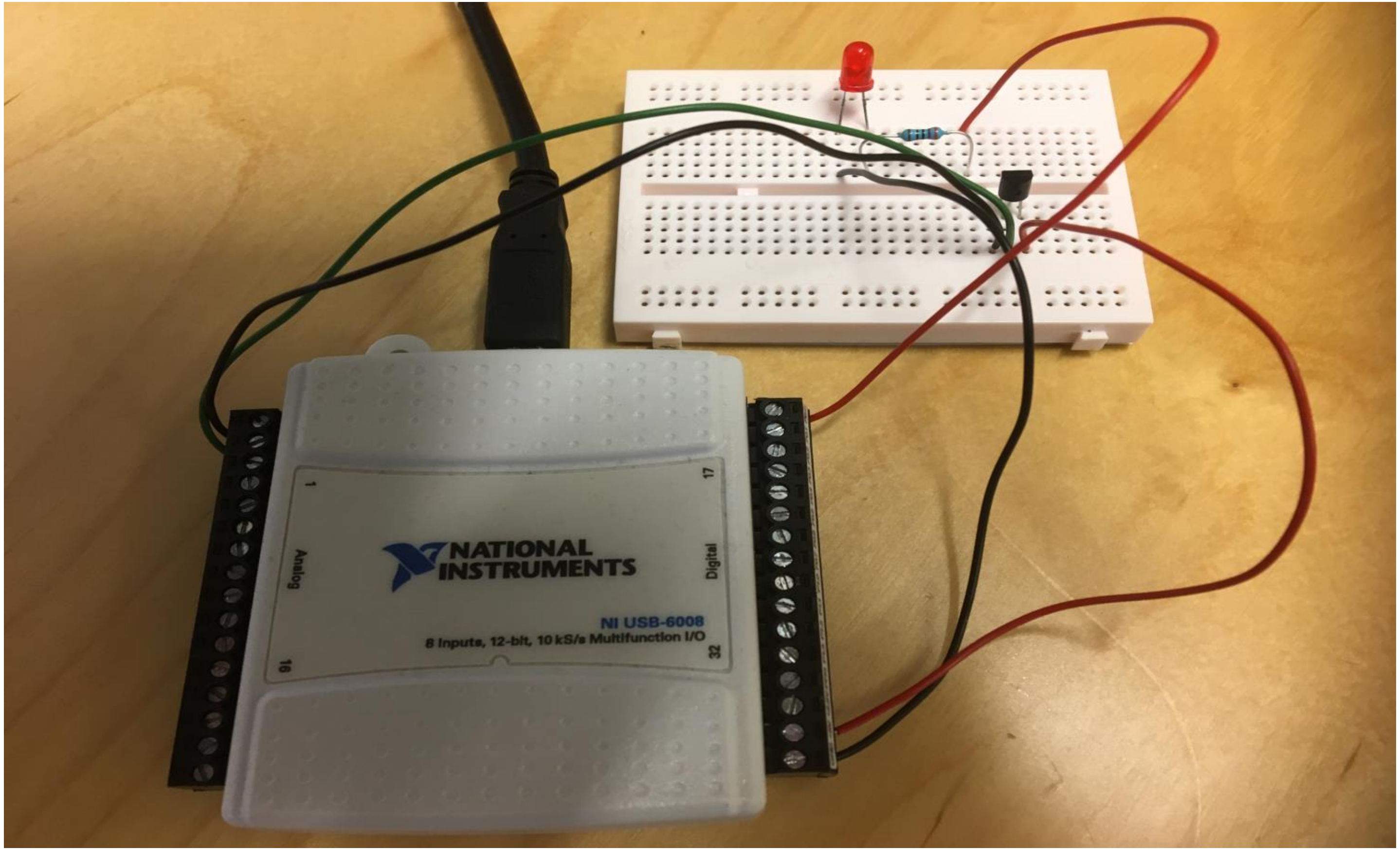


LED Wiring



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
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
GND
$R = 270\Omega$

Hardware Setup

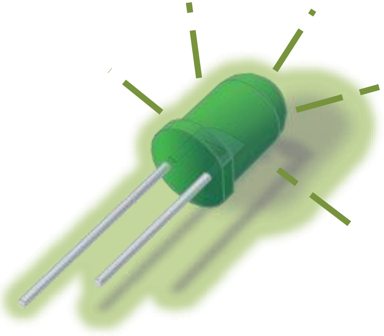


Python Code

Temperature > Limit?

No

Yes



LED OFF

LED ON

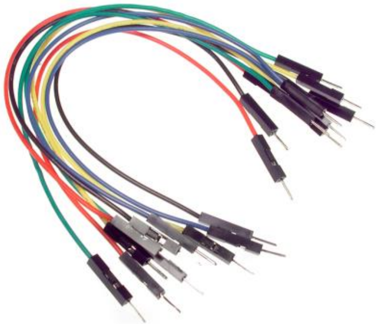
```
import nidaqmx
import time

# 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()
```

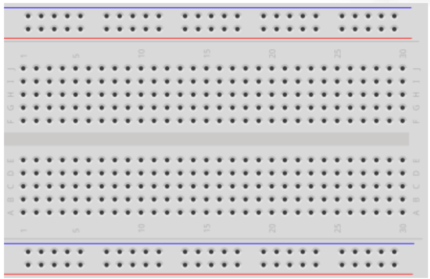
Thermistor with Python

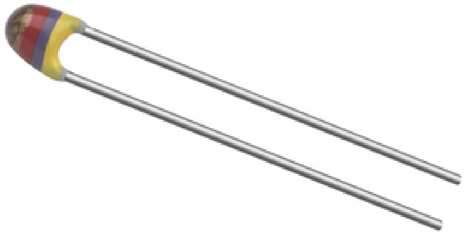
Hans-Petter Halvorsen

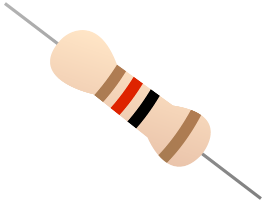
Necessary Equipment

- 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

$A = 0.001129148, B = 0.000234125 \text{ and } C = 8.76741E-08$

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:

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

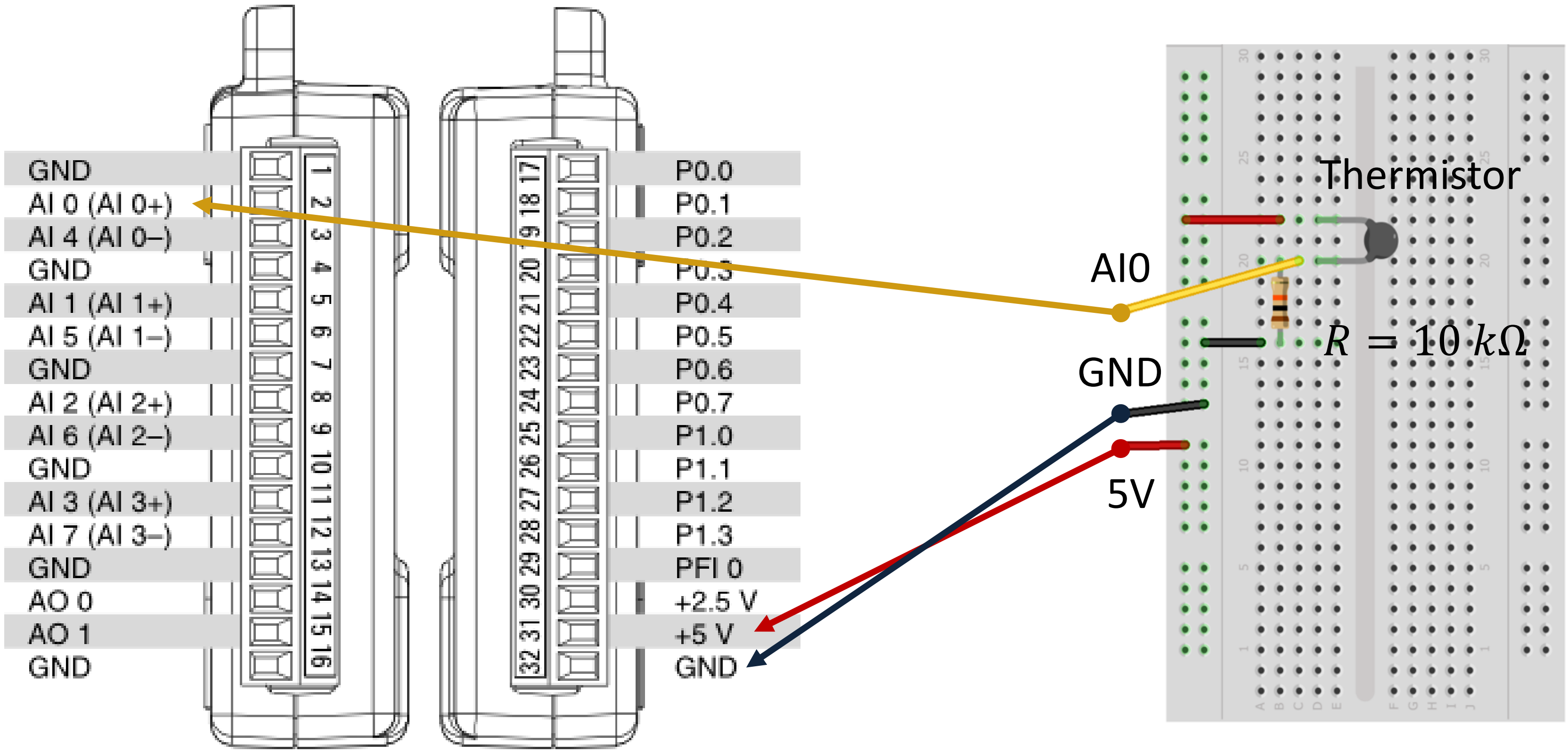
Where the Temperature T_K is in Kelvin
$A, B\ and\ C$ are constants

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

The Temperature in degrees Celsius will then be:

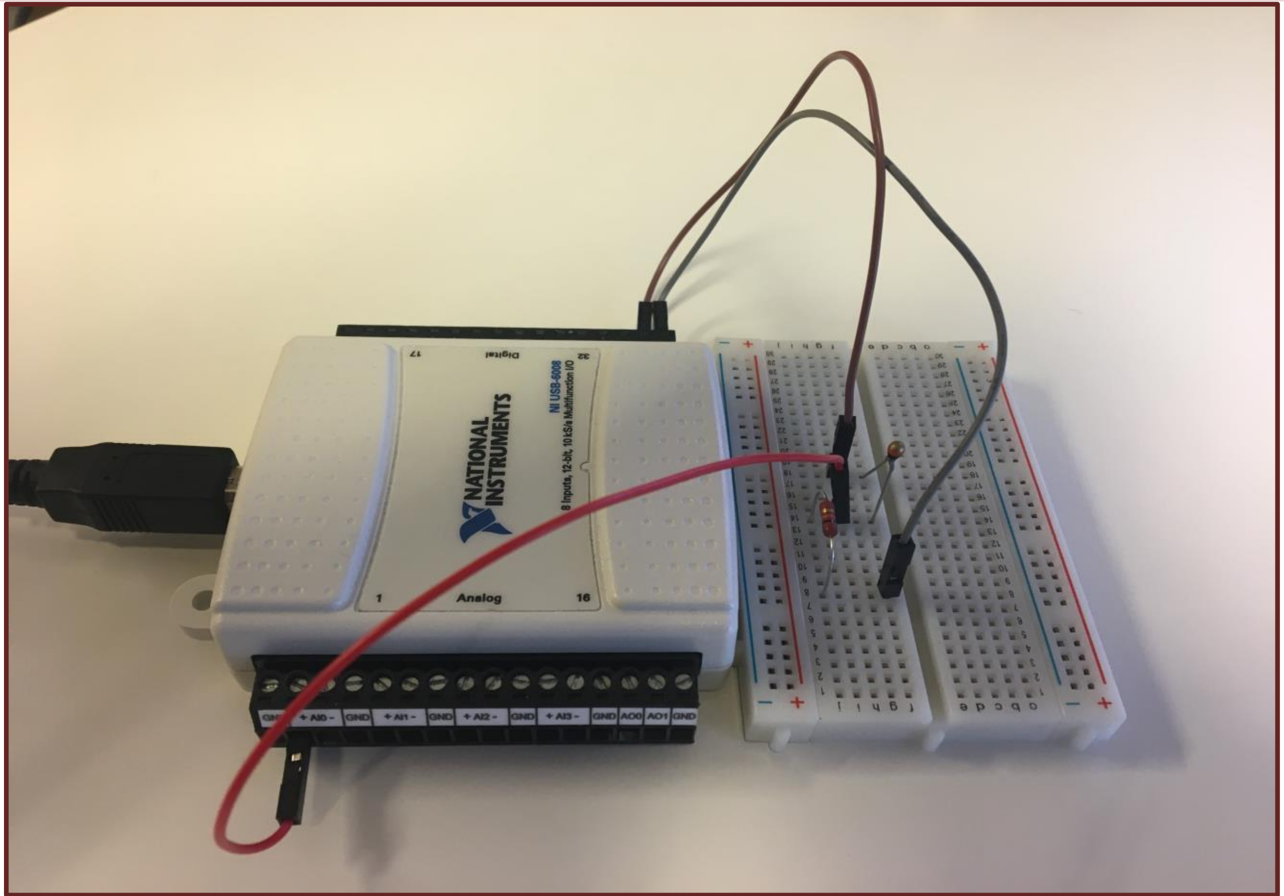
$$T_C = T_K - 273.15$$

Wiring



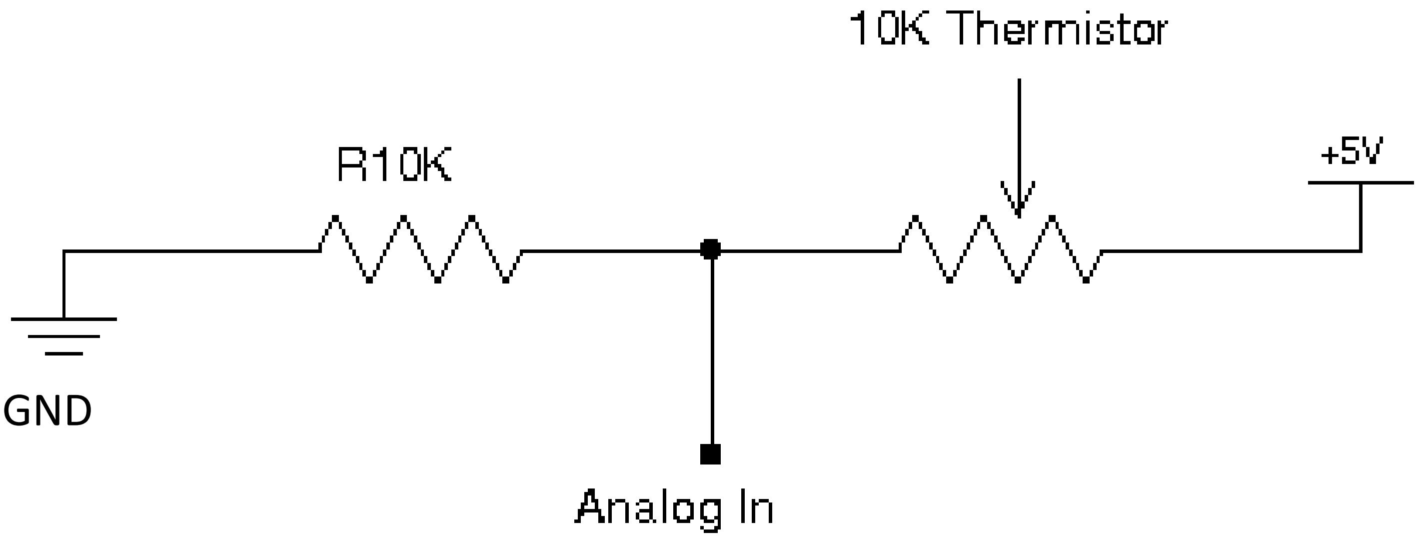
GND
AI 0 (AI 0+)
AI 4 (AI 0–)
GND
AI 1 (AI 1+)
AI 5 (AI 1–)
GND
AI 2 (AI 2+)
AI 6 (AI 2–)
GND
AI 3 (AI 3+)
AI 7 (AI 3–)
GND
AO 0
AO 1
GND
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P1.0
P1.1
P1.2
P1.3
PFI 0
+2.5 V
+5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

Hardware Setup



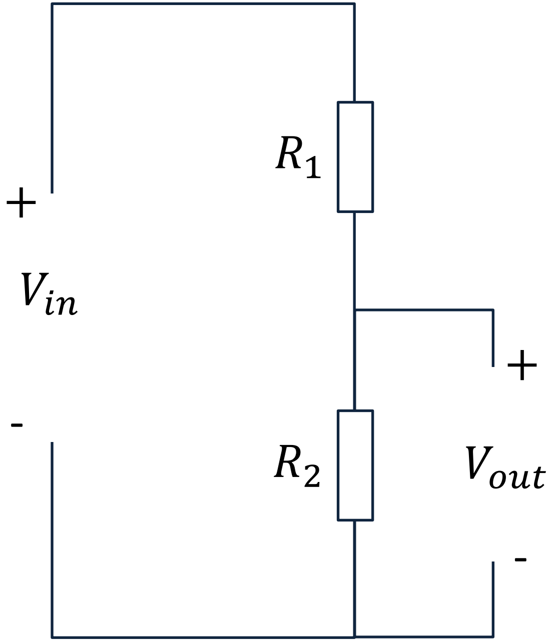
Voltage Divider

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

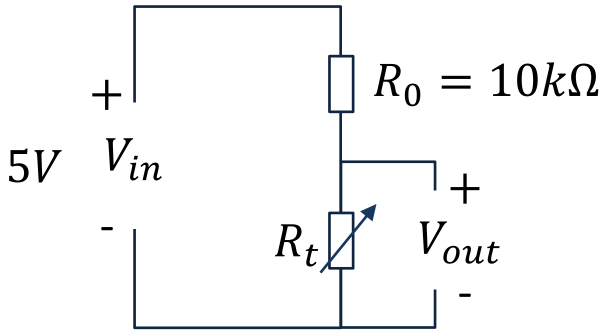
Voltage Divider for our System

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℃

<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
```

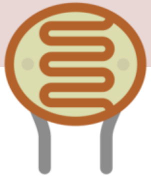
Thermistor Application:

```
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()
```

Light Sensor with Python

Hans-Petter Halvorsen

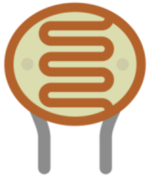
Light Sensor

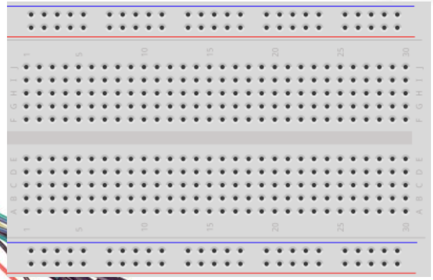


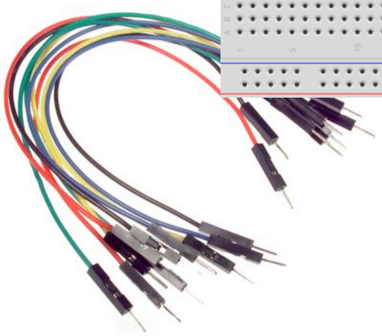
- Light sensor, Photocell (Photo resistor), LDR (light-dependent resistor)
- A light sensor / photocell is a sensor used to detect light.
- The resistance changes with the change in light intensity

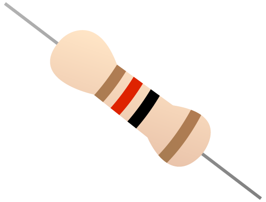
Necessary Equipment

- PC
- DAQ Module, e.g., USB-6008
- Breadboard
- Light Sensor
- Wires (Jumper Wires)
- Resistors, $R = 10\ k\Omega$

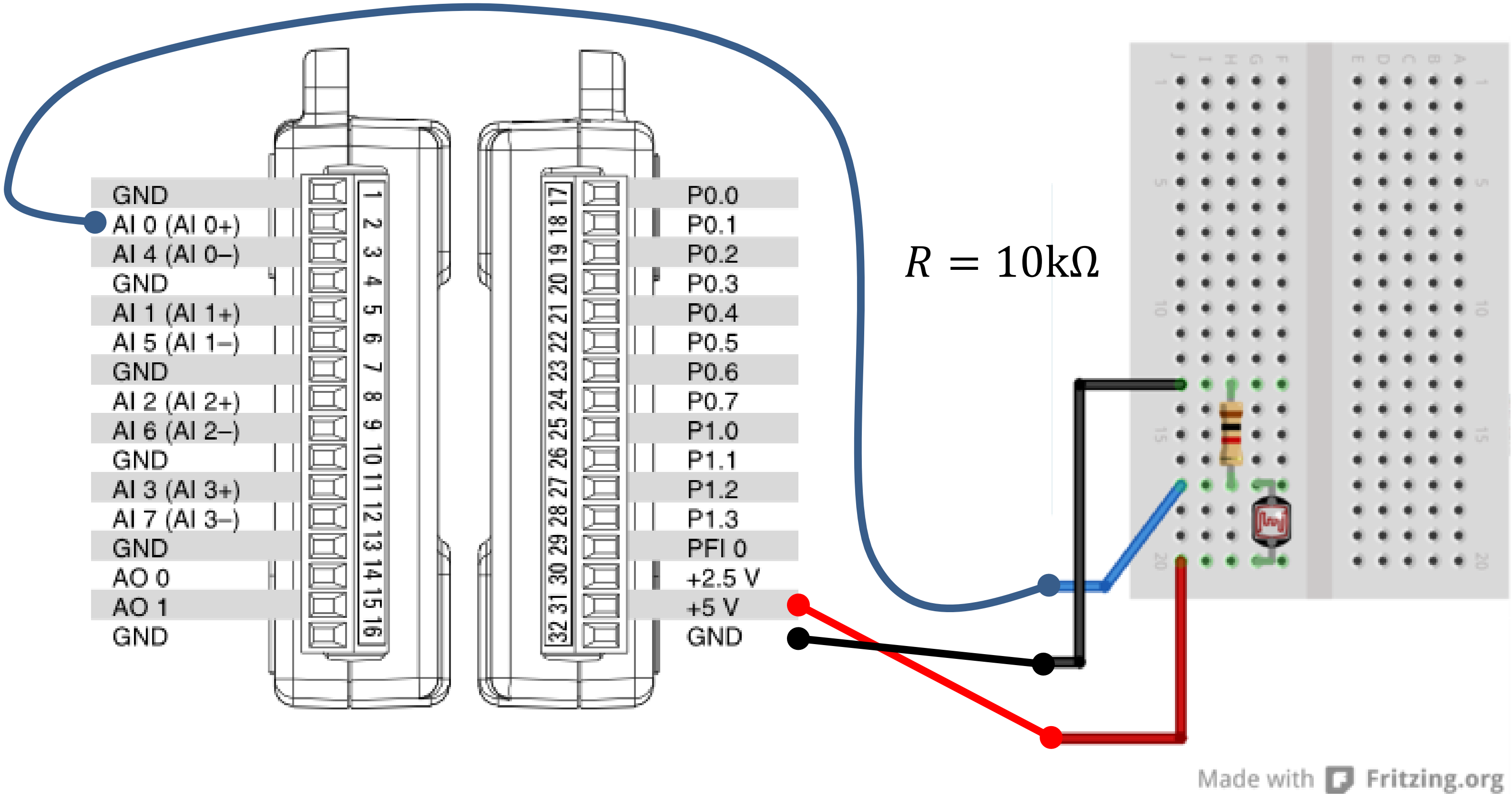








Hardware Setup



Python Code

```
import nidaqmx

from nidaqmx.constants import (
    TerminalConfiguration)

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

value = task.read()
print(value)

task.stop
task.close()
```

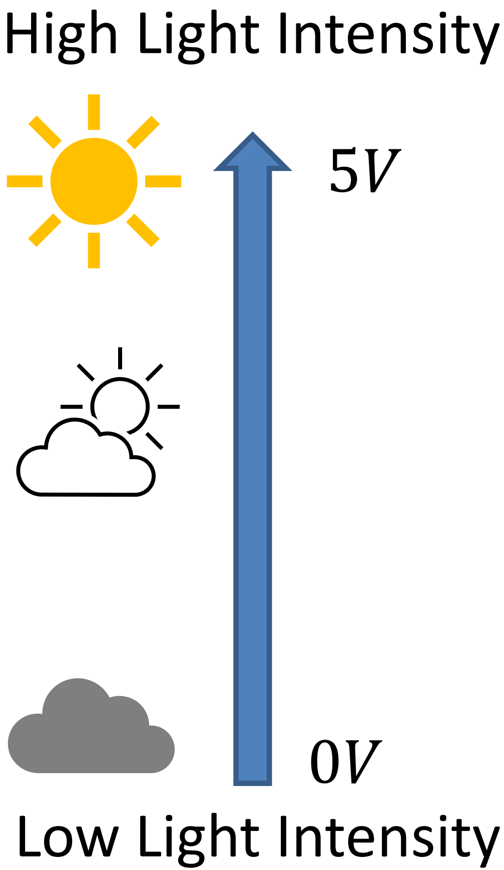
Python Code – For Loop

```
import nidaqmx
import time

from nidaqmx.constants import (
    TerminalConfiguration)

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

N = 60
for k in range(N):
    Vout = task.read()
    print(Vout)
    time.sleep(1)

task.stop
task.close()
```

Light Sensor Results

- The resistance changes with the change in light intensity.
- We measure the the voltage (using a Voltage Divider)
- When the Light Intensity gets Higher, the Voltage Level gets Higher

The Light Sensor has not very high accuracy, but you can typically use it to automatically turn on a light when it get dark outside (or inside)



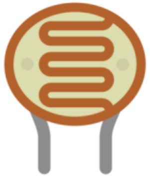
Light Sensor Example

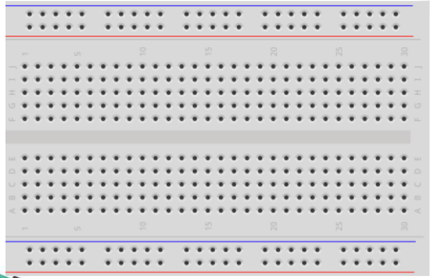
- The Light Sensor has not very high accuracy, but you can typically use it to automatically turn on a light when it get dark outside (or inside)
- In this example we will use a light sensor to measure the light intensity of the room.
  - If it's dark, we will turn on the light (LED)
  - If it's bright, we'll turn off the light (LED)

Necessary Equipment

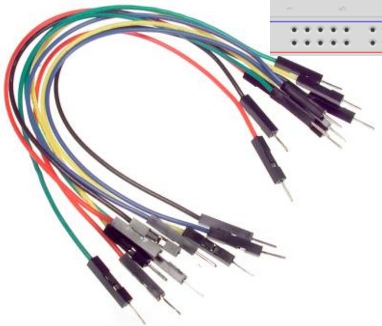
- PC
- DAQ Module, e.g., USB-6008
- Breadboard
- Light Sensor
- Wires (Jumper Wires)
- Resistors

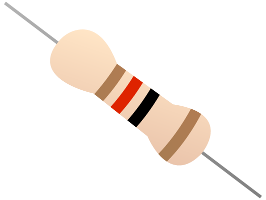
$$R = 270\Omega$$

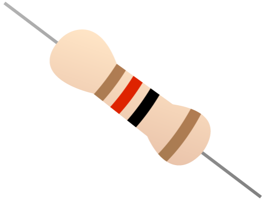




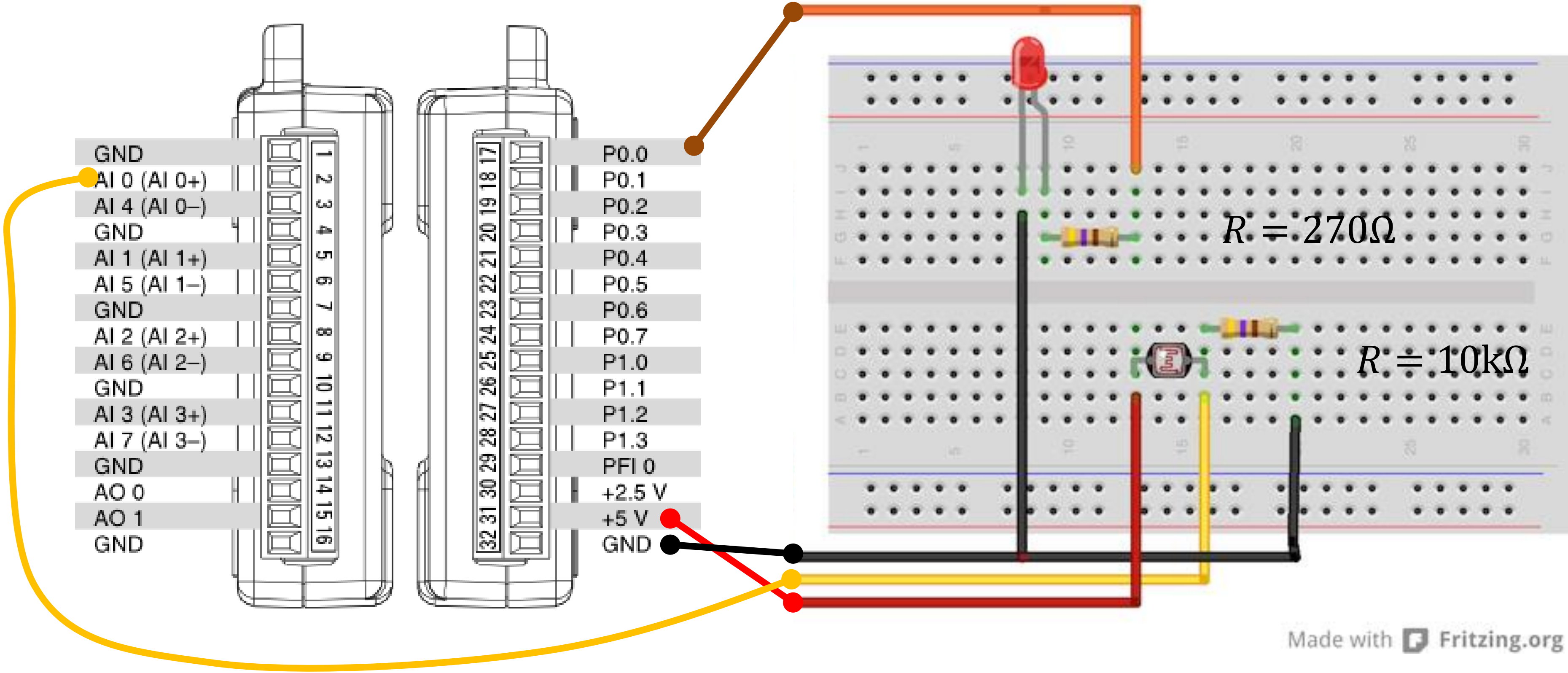
R







Hardware Setup



Python Code

If it's dark, we will turn on the light (LED)
If it's bright, we'll turn off the light (LED)

In the Example a the “Bright Level” is set to 0.2V

This value needs to be adjusted (“trial and error“) depending on the use of the application.

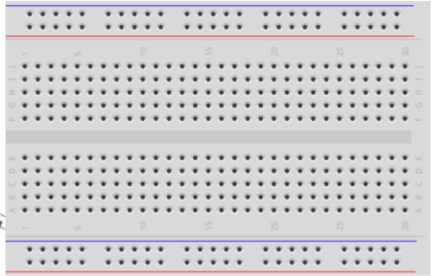
```
import nidaqmx
import time

from nidaqmx.constants import (
    TerminalConfiguration)

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()

brightlevel = 0.2
N = 60
for k in range(N):
    Vout = task_ai.read()
    print(round(Vout,2))

    task_do.write(True)

    if Vout < brightlevel:
        task_do.write(True)
    else:
        task_do.write(False)
    time.sleep(1)

task_do.write(False)

task_ai.stop; task_ai.close()
task_do.stop; task_do.close()
```

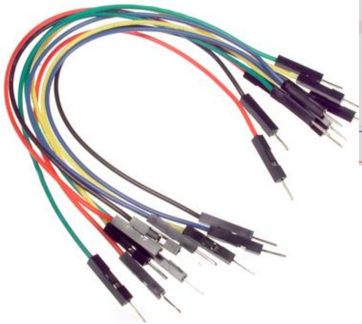
Push Button with Python

Hans-Petter Halvorsen

Necessary Equipment

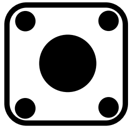
- DAQ Device (e.g., USB-6008)
- Breadboard
- Push Button
- Wires (Jumper Wires)

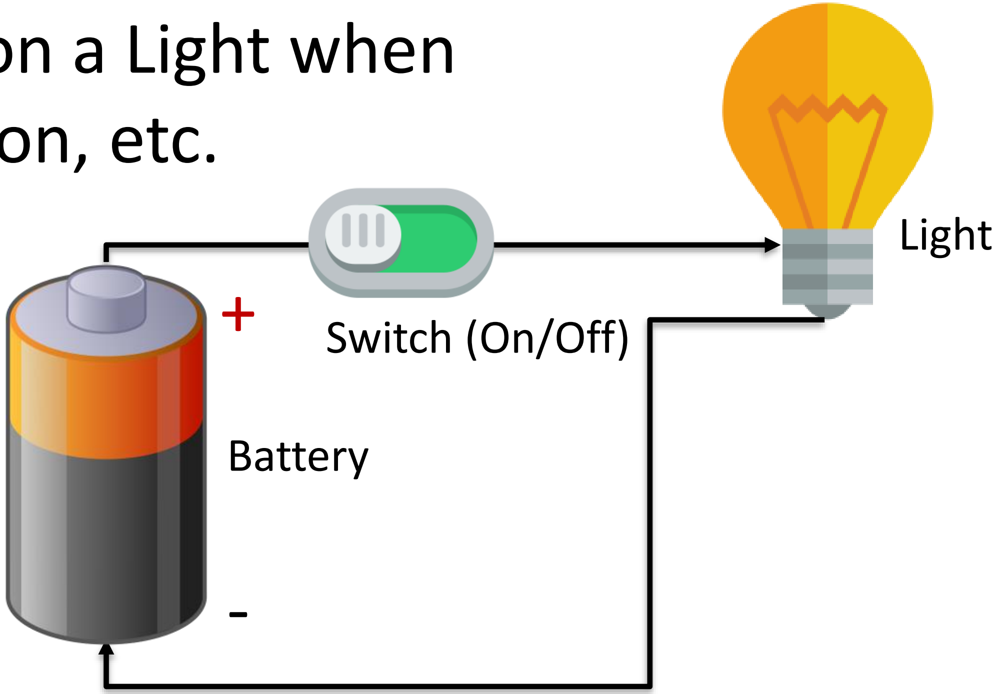




Push Button/Switch

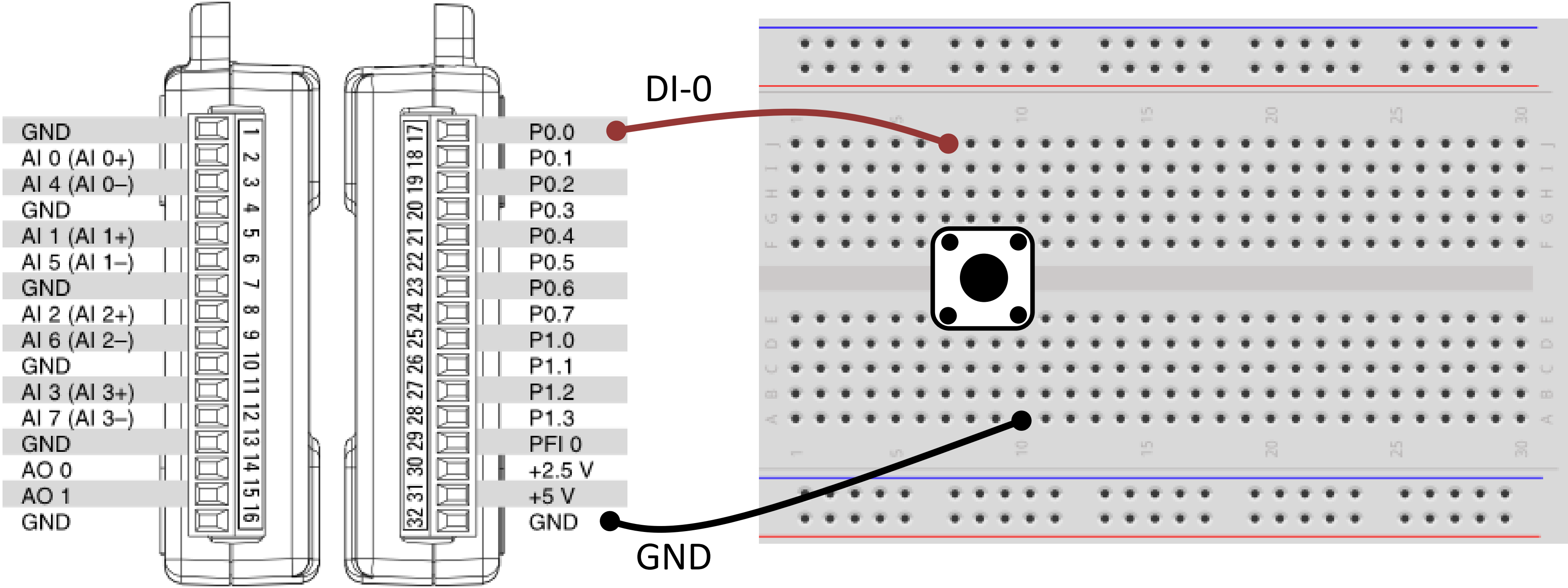
- Pushbuttons or switches connect two points in a circuit when you press them.
- You can use it to turn on a Light when holding down the button, etc.





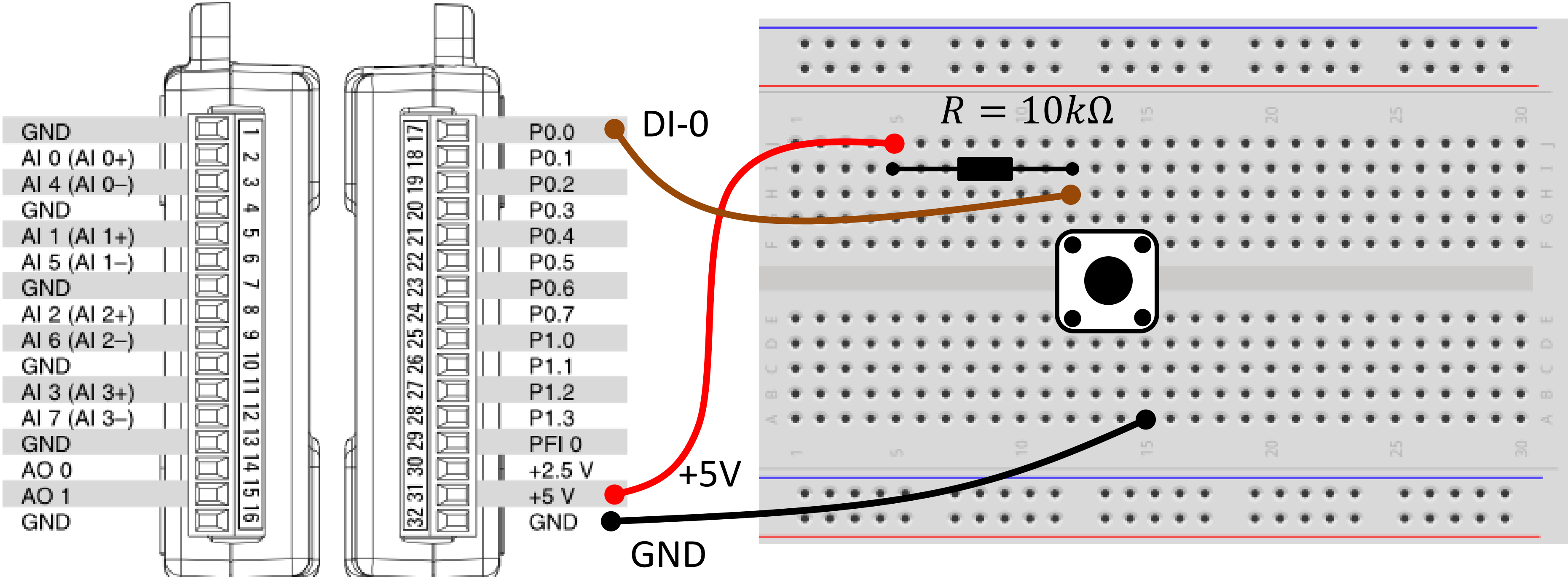
Hardware Setup

Using built-in 4.7 kΩ **Pull-up Resistor**



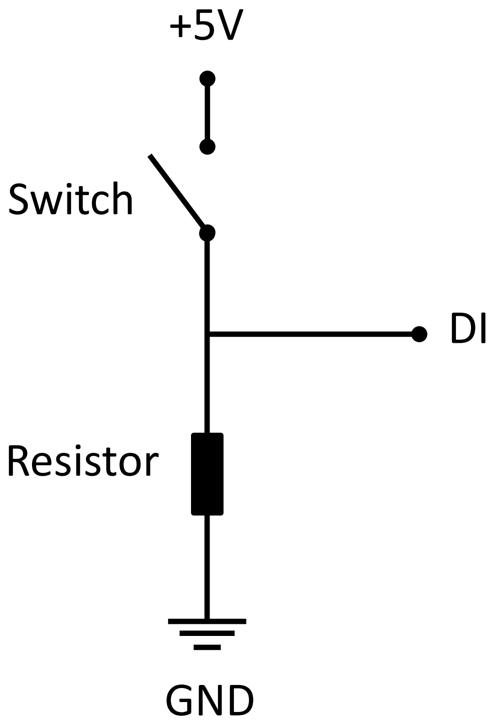
Hardware Setup

Using external Pull-up Resistor

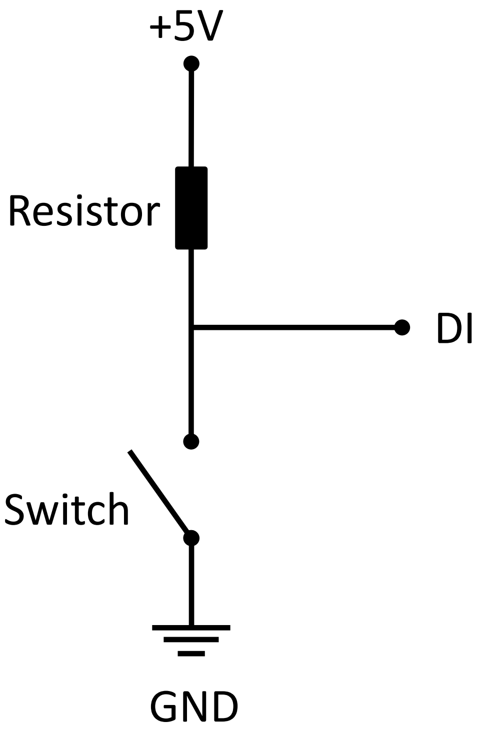


Pull-down/Pull-up Resistor

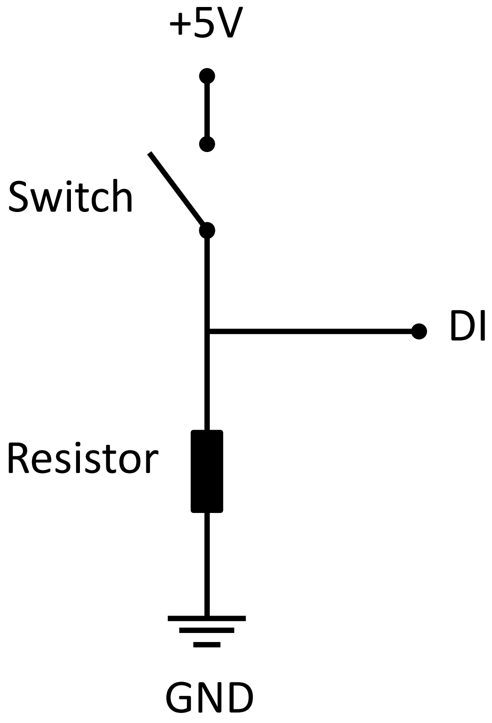
Pull-down Resistor



Pull-up Resistor

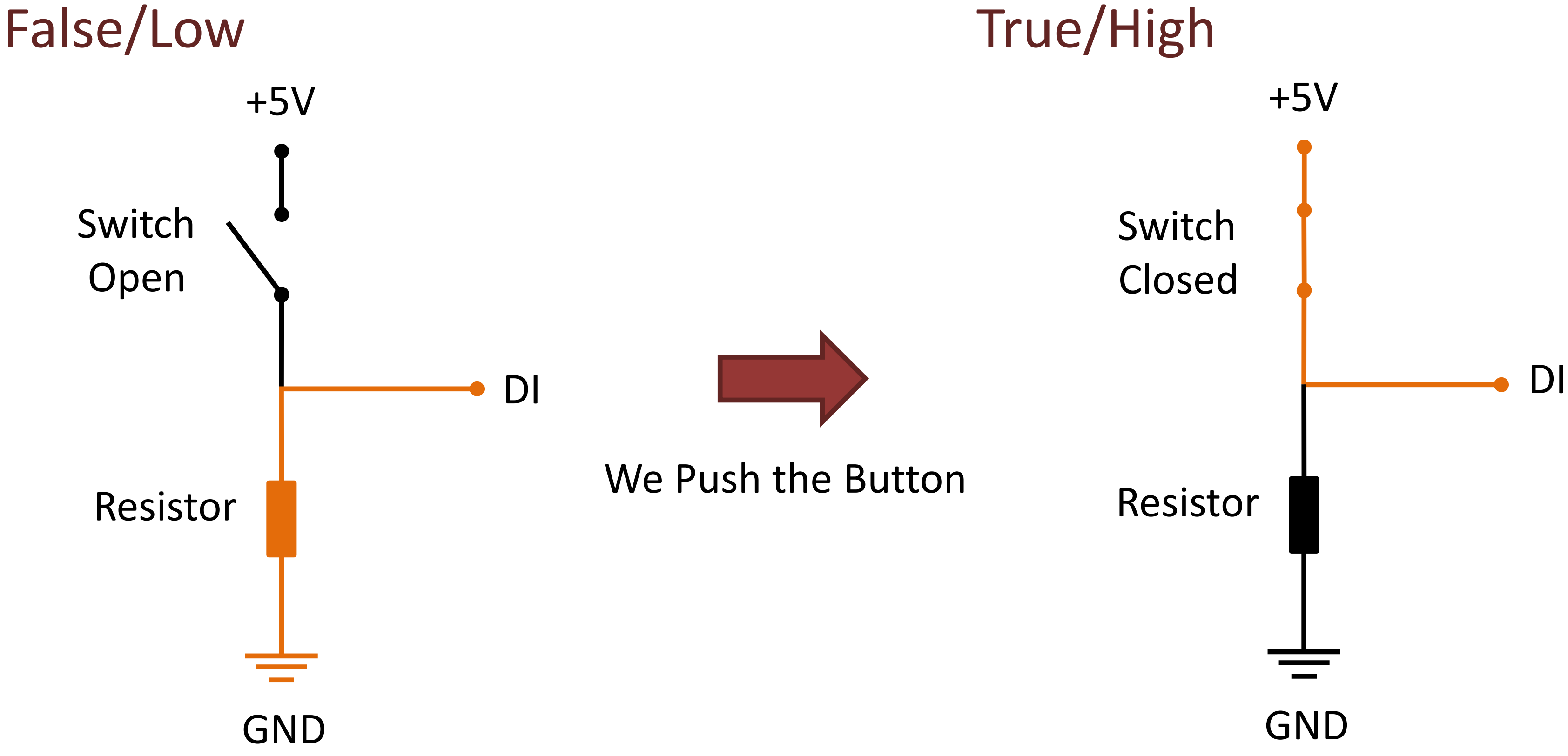


Pull-down Resistor

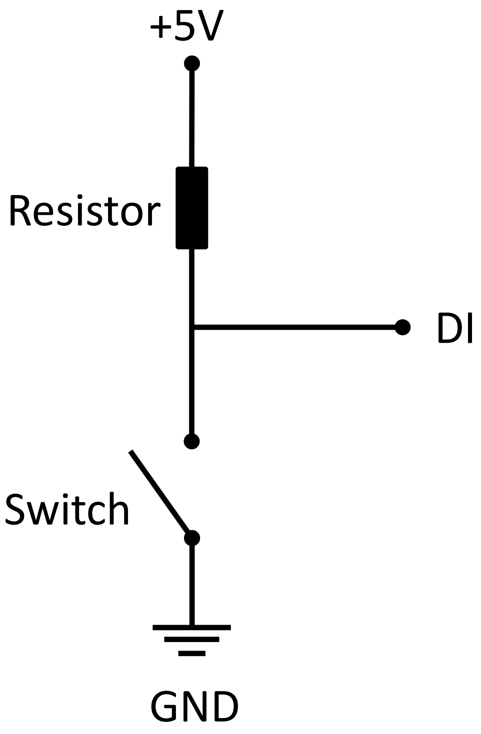


- When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton
- This means the DI pin is connected to ground through the pull-down resistor and we read a **False** (Low).
- When the button is closed (pressed), it makes a connection between its two legs
- This means the DI pin is connected to +5V, so then we read **True** (High).

Pull-down Resistor

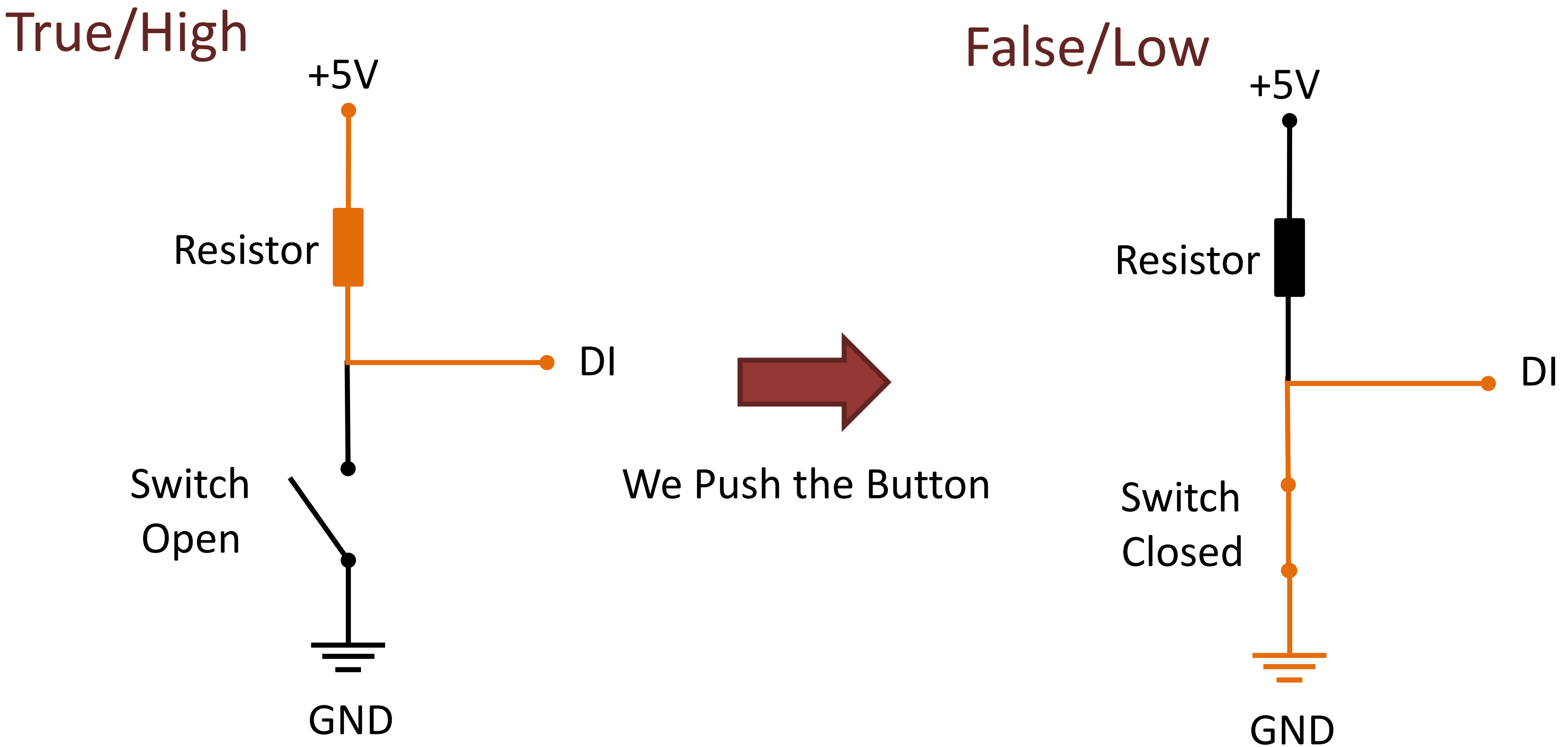


Pull-up Resistor



- When the pushbutton is open (unpressed) there is a connection between 5V and the DI pin.
- This means the default state is **True** (High).
- When the button is closed (pressed), the state goes to **False** (Low).

Pull-up Resistor



Pull-down/Pull-up Resistor

Why do we need a pull-up or pull-down resistor in the circuit?

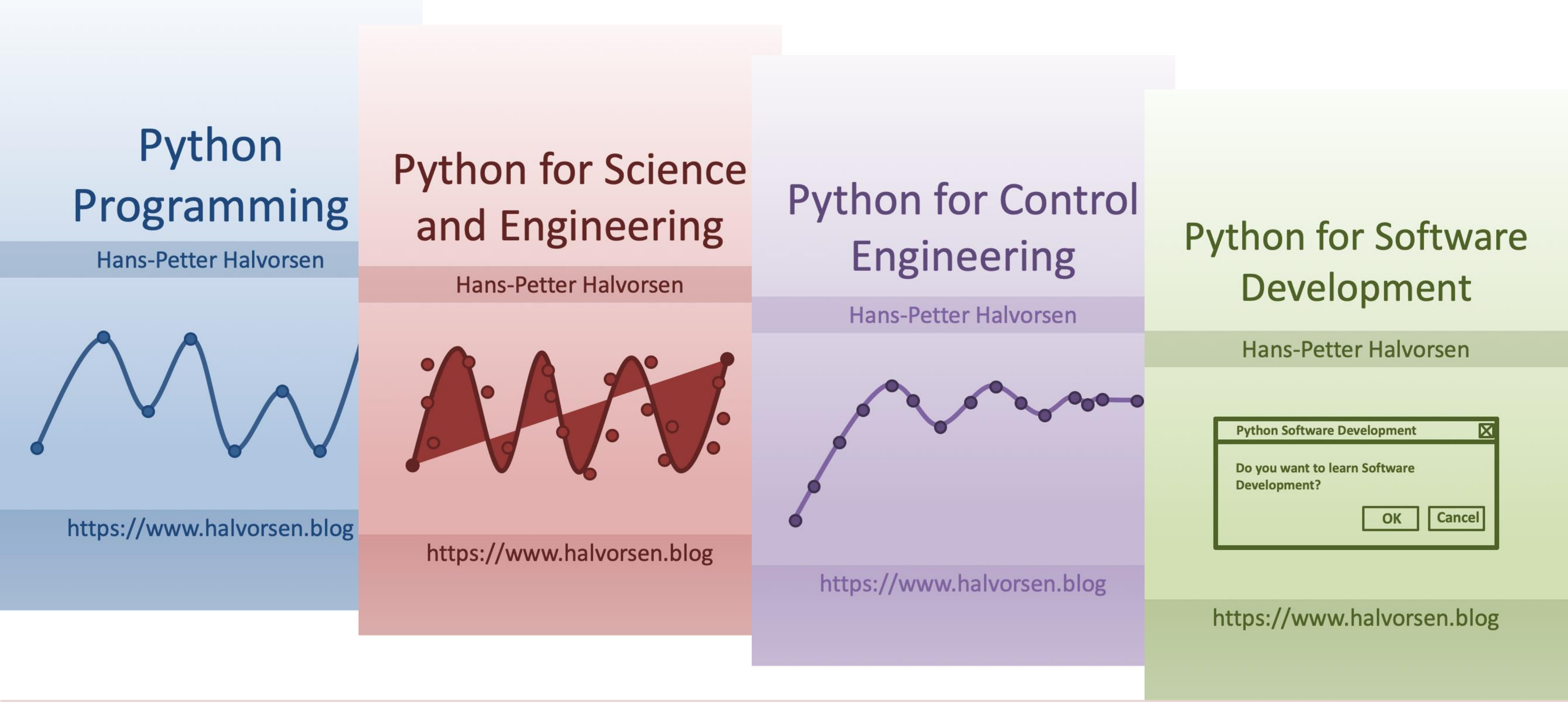
- If you disconnect the digital I/O pin from everything, the LED may blink in an irregular way.
- This is because the input is "floating" - that is, it will randomly return either HIGH or LOW.
- That's why you need a pull-up or pull-down resistor in the circuit.

Python

Here you can do the magic, e.g., turn on a light, an engine or what ever. In this basic example I just print a message to the user.

```
import nidaqmx
import time

task_di = nidaqmx.Task()
task_di.di_channels.add_di_chan("Dev1/port0/line0")
task_di.start()

N = 10
for k in range(N):
    buttonstate = task_di.read()

    if buttonstate != True:
        print("The Button is Pushed")
    else:
        print("Nothing")

    time.sleep(1)

task_di.stop
task_di.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