https://www.halvorsen.blog

DAQ I/O Modules with Python

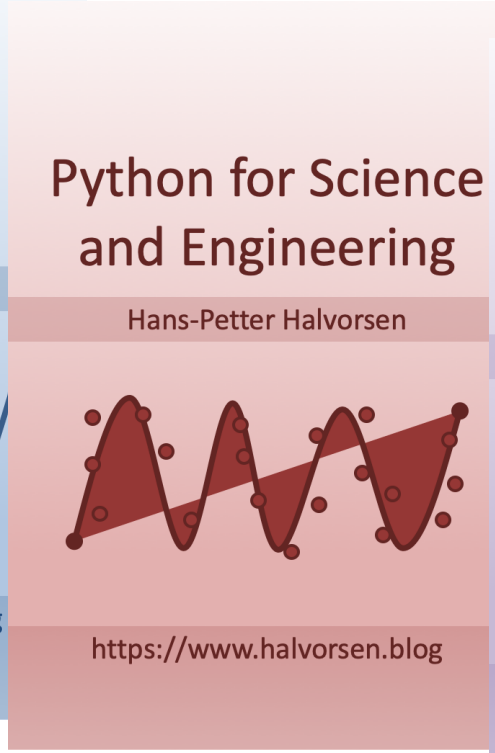
Exemplified by using NI USB-6008 I/O Module

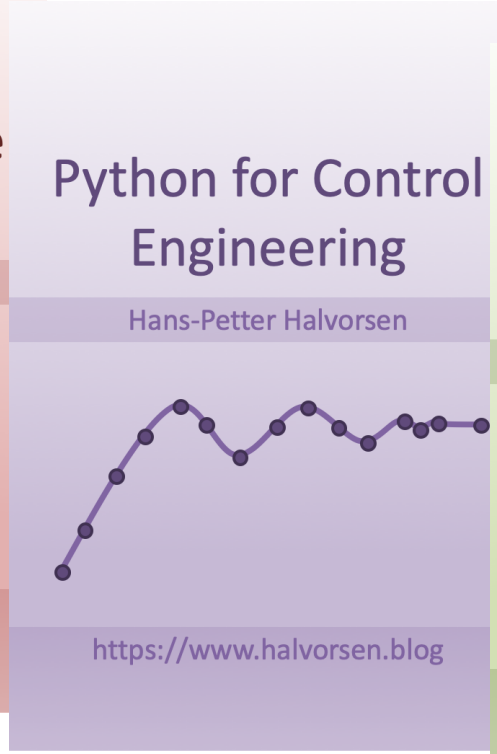
Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

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

Additional Python Resources





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

Contents

- How can we use NI Hardware with Python?
- What is DAQ and I/O Modules?
- NI-DAQmx
- nidaqmx Python API
- Python Examples
  - Analog Out (Write) - AO
  - Analog In (Read) - AI
  - Digital Out (Write) - DO
  - Digital In (Read) - DI

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 I/O Module or DAQ Device NI USB-6008 as an Example

How can we use NI Hardware with Python?

- NI is a company that manufacture and sell both Hardware and Software
- The are most famous for their LabVIEW software
- LabVIEW is popular Graphical Programming Language
- Typically you use LabVIEW in combination with NI DAQ Hardware, but the NI-DAQmx can also be used from C, C#, Python, etc.
- Control NI DAQ Device with Python and NI DAQmx
  - https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P8o0SAC

LabVIEW

- In this Tutorial we will use Python and not LabVIEW
- But if you want to learn more about LabVIEW, you may take a look at my LabVIEW resources:
- https://halvorsen.blog/documents/programming/labview/labview.php

NI DAQ Hardware

Some Examples

TC-01 Thermocouple

myDAQ

NI-DAQmx
Hardware Driver

USB-6001

USB-6008

cDAQ

Note! The Python Examples provided will work for all NI-DAQ Devices using the NI-DAQmx Driver, which is several hundreds different types

USB-600x

- NI has many DAQ devices (or I/O Modules) that can be used together with NI-DAQmx Python API
- Examples of low-cost USB DAQ Devices from NI: USB-6001, .. ,USB-6008, USB-6009

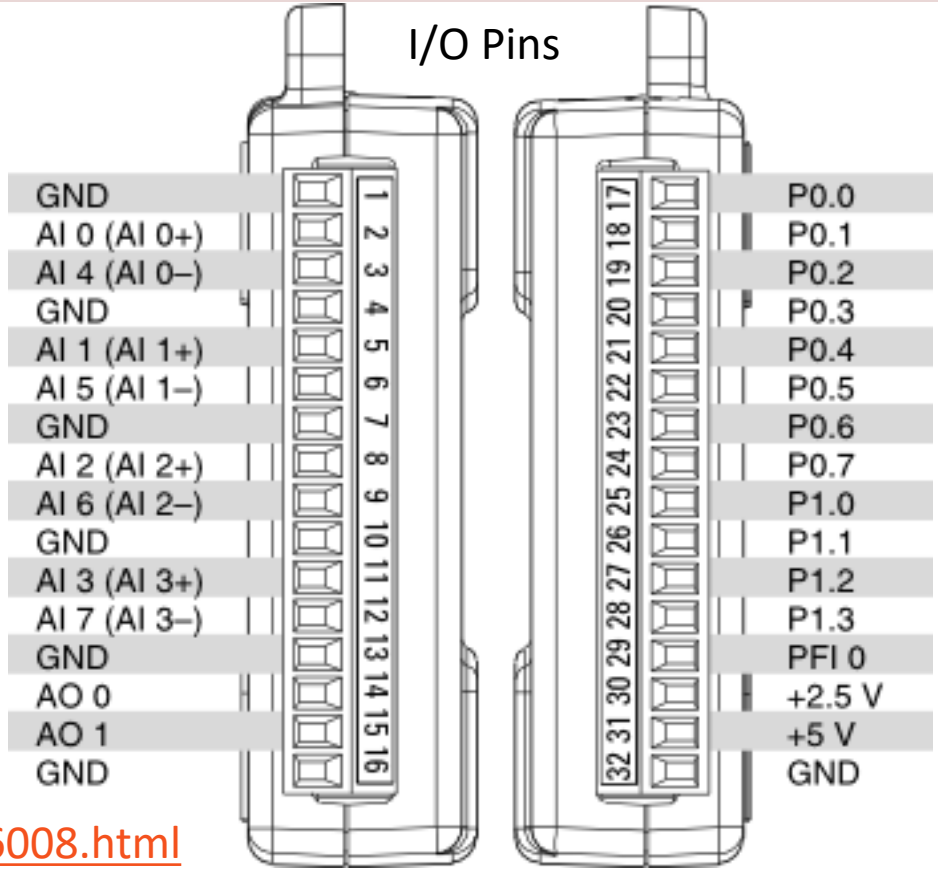
USB-6001

... ...

USB-6008

NI USB-6008

We will use NI USB-6008 in our examples



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

NI USB-6008

- The USB-6008 is a low-cost, multifunction DAQ device.
- It offers analog I/O, digital I/O, and a 32-bit counter.
- The USB-6008 provides basic functionality for applications such as simple data logging, portable measurements, and academic lab experiments.
- You can easily connect sensors and signals to the USB-6008 with screw-terminal connectivity.
- 8 AI Single-ended or 4 AI Differential (12-Bit, 10 kS/s)
- 2 AO (150 Hz)
- 12 DIO (you can choose DI or DO)

http://www.ni.com/pdf/manuals/375295c.pdf

NI DAQ Device with Python

How to use a NI DAQ Device with Python

Data Acquisition (DAQ)

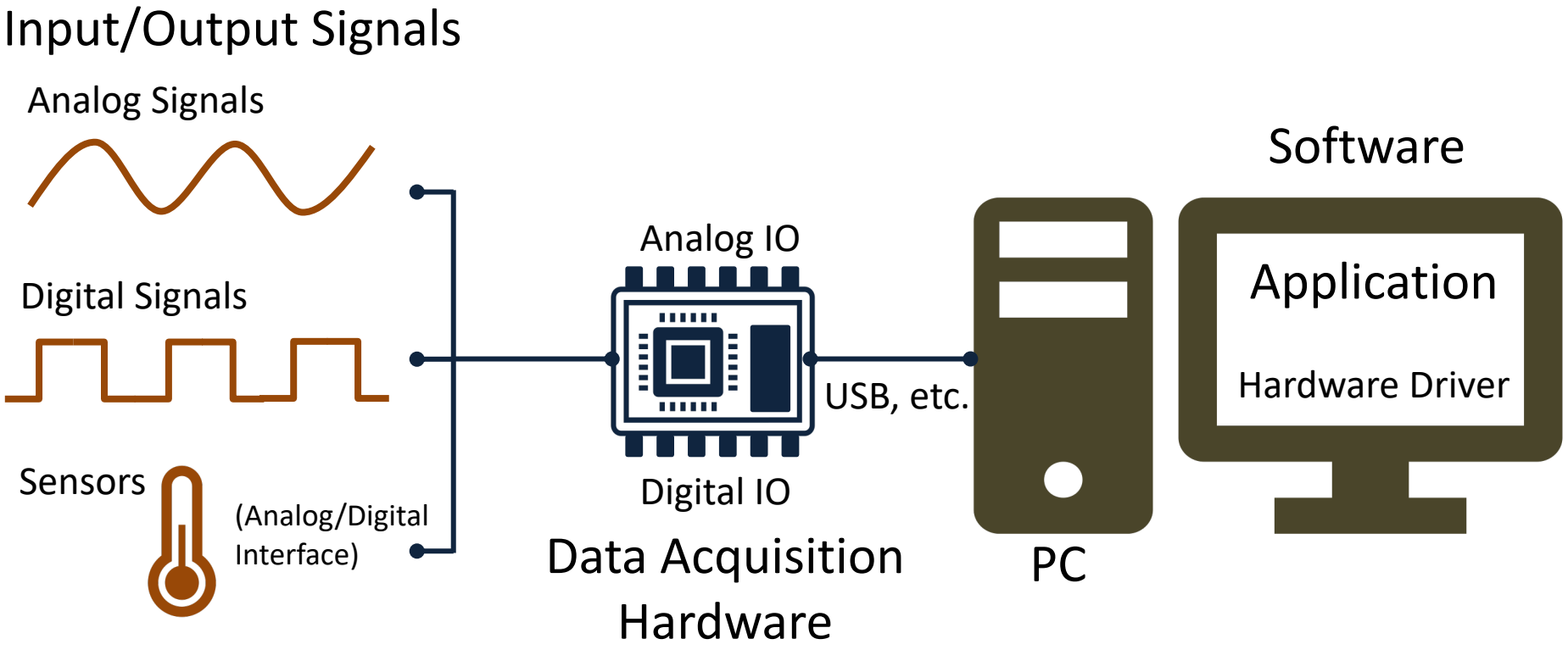
- To read sensor data you typically need a DAQ (Data Acquisition) device connected to you PC
- You can alos use devices like Arduino , Raspberry Pi, etc.
- In all cases you will typically need to install a driver from the vendor of the DAQ device or the sensor you are using

DAQ System

A DAQ System consists of 4 parts:

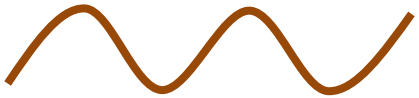
- Physical input/output signals, sensors
- DAQ device/hardware
- Driver software
- Your software application (Application Software) - in this case your Python application

DAQ System


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

I/O Module

Analog Signals



$0 - 5V$ or $0 - 10V$

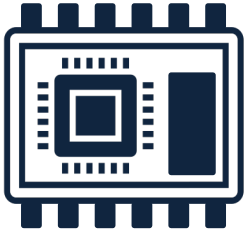
Analog Sensors

Analog IO

Analog Input (**AI**)

Analog Output (**AO**)

I/O Module



Digital IO

Digital Signals

True

False

Digital Input (**DI**)

Digital Output (**DO**)

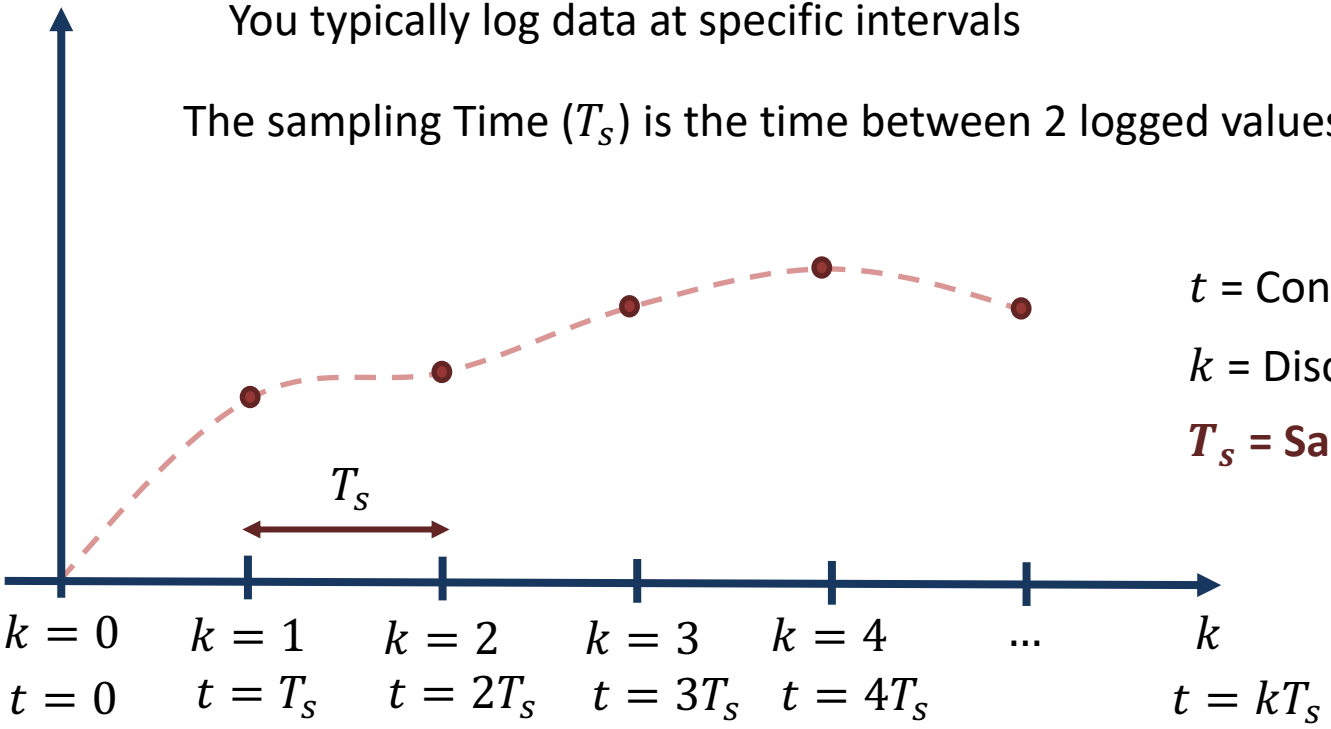
Sensors with Digital Interface (e.g., SPI, I2C)

Digital Signals

A computer can only deal with discrete signals

You typically log data at specific intervals

The sampling Time (T_s) is the time between 2 logged values



t = Continuous Time

k = Discrete Time

T_s = Sampling Time

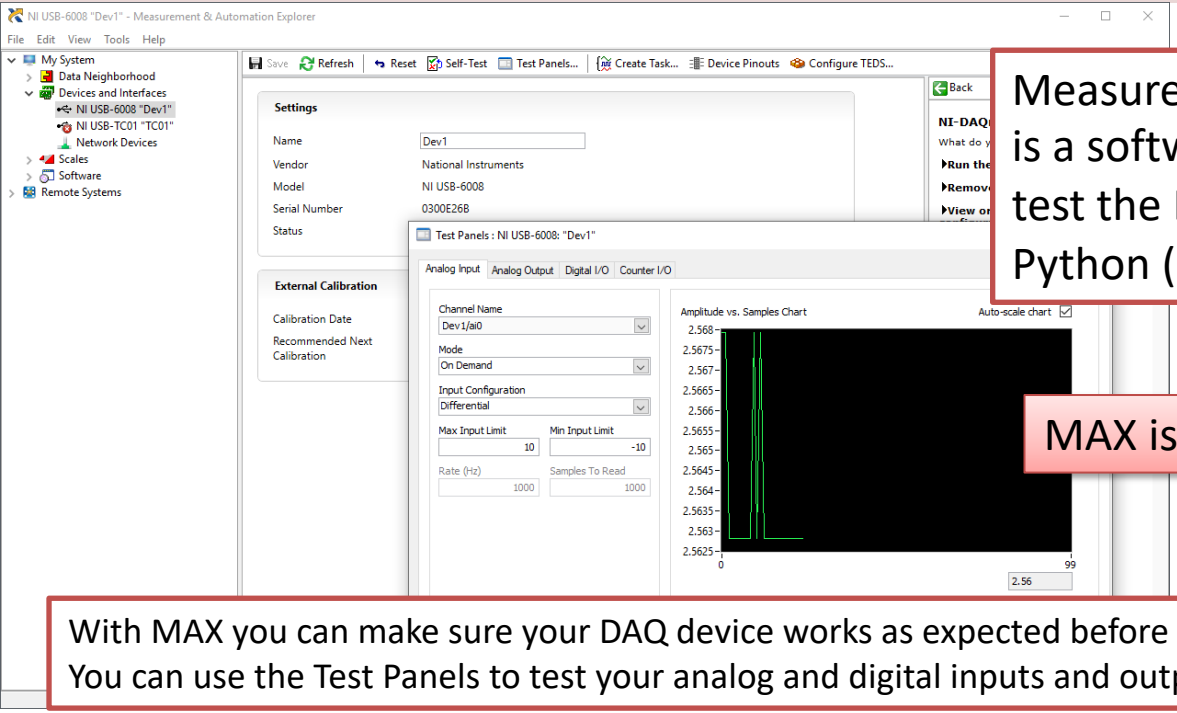
DAQ

- Here you find more information, resources, videos and examples regarding DAQ:
- https://www.halvorsen.blog/documents/technology/daq

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)



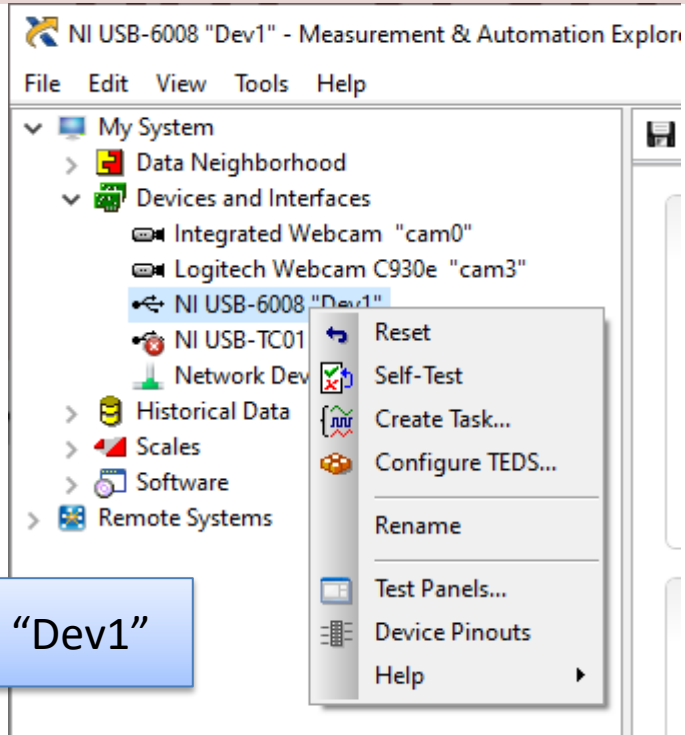
Measurement & Automation Explorer (MAX) is a software you can use to configure and test the DAQ device before you use it in Python (or other programming languages).

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.

Measurement & Automation Explorer (MAX)

Rename Device Name:



You can change the Name of the Device by right-clicking and select "Rename"

The default Name is "Dev1"

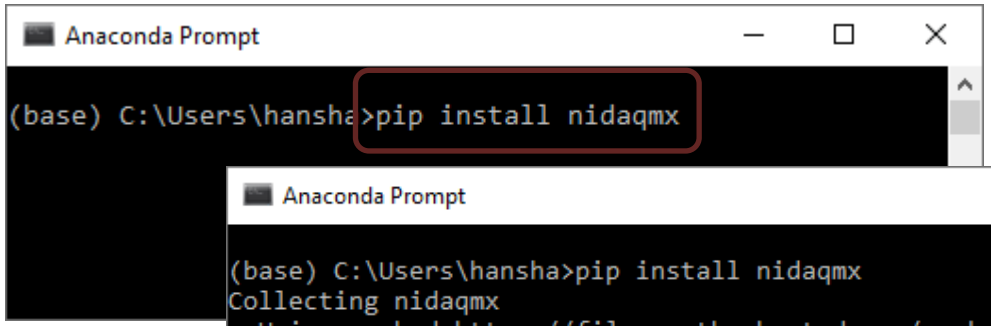
You will use this Device Name inside your Python code

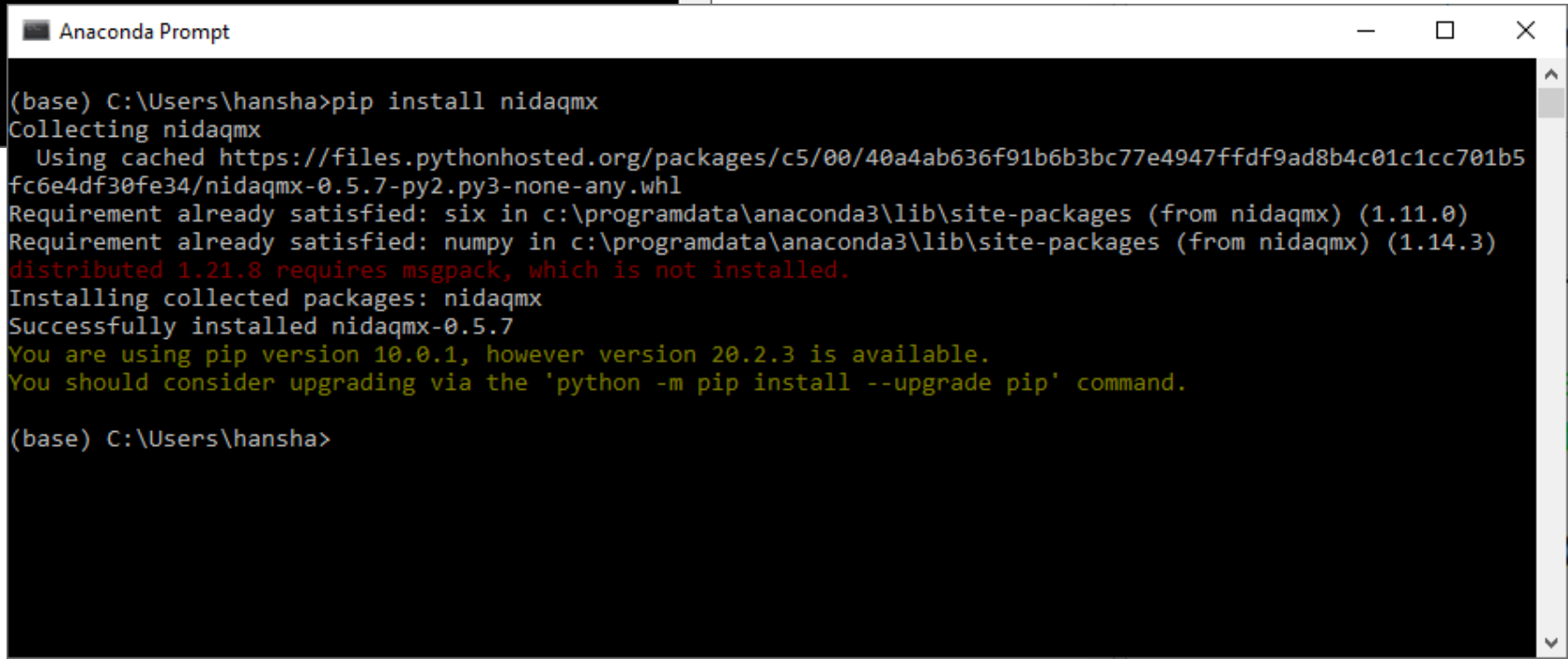
nidaqmx Python API

- 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

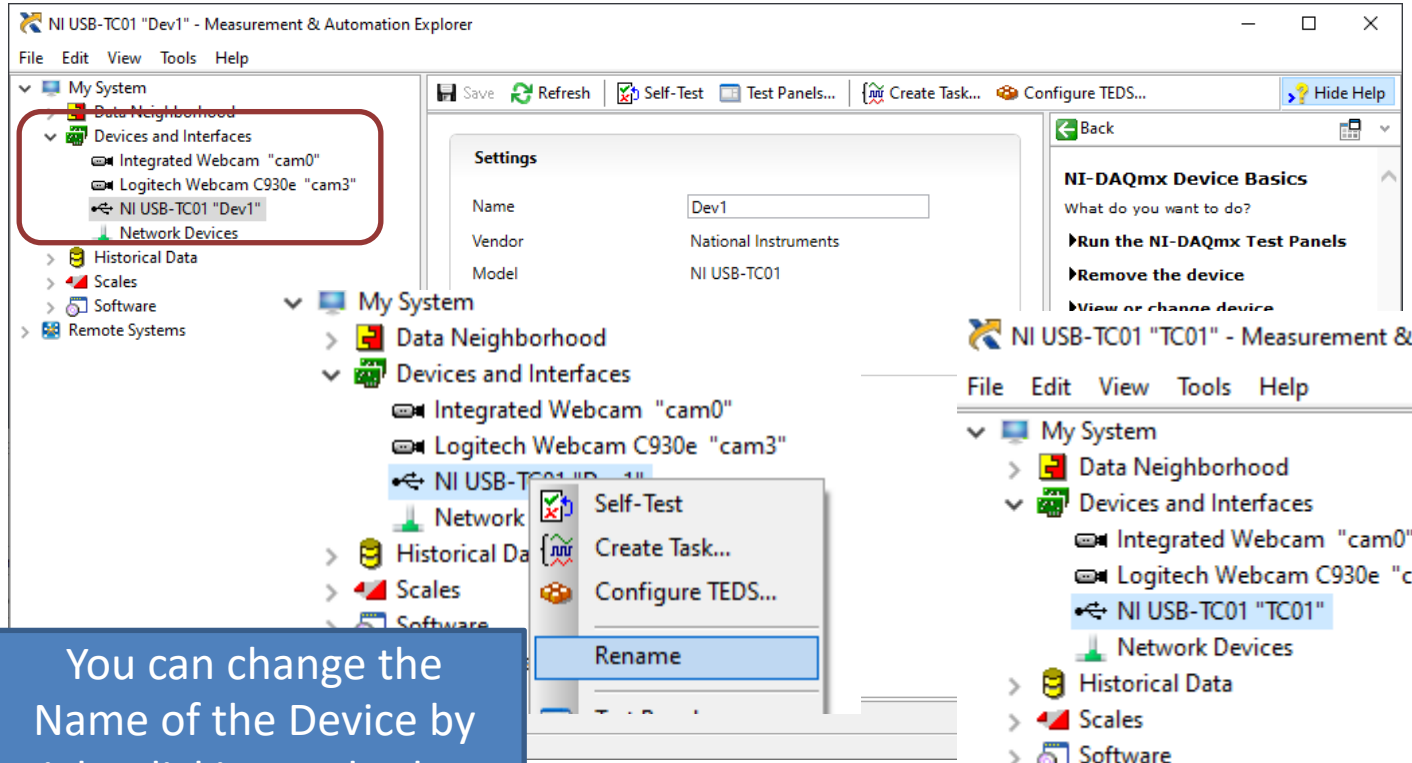
nidaqmx Python Package

Installation

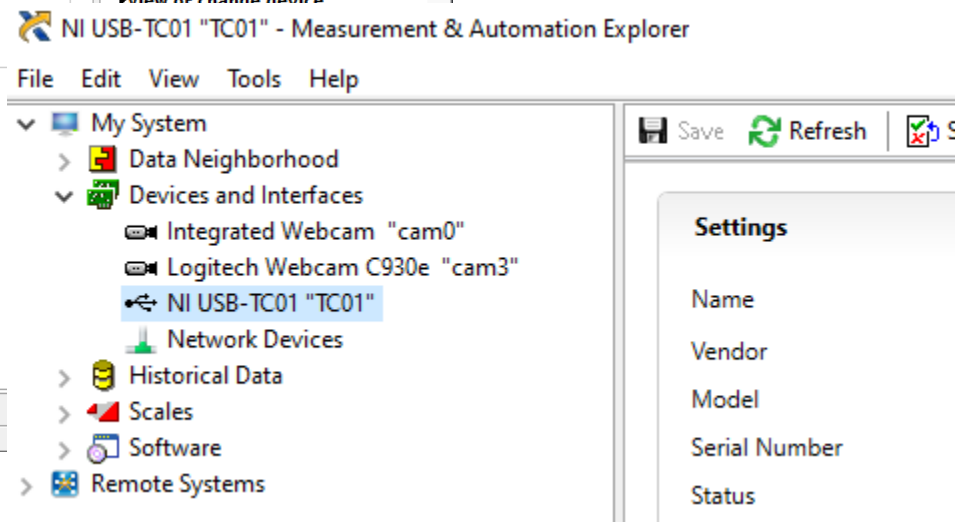




nidaqmx Python Package



MAX



You can change the Name of the Device by right-clicking and select “Rename”

Python Examples

Hans-Petter Halvorsen

Python Examples

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

Python Examples

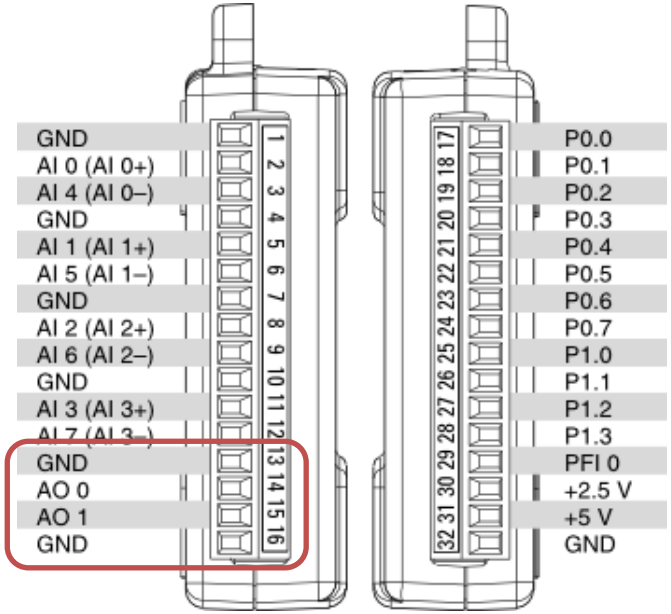
- You can easily extend this examples to make them suit your needs.
- Typically you need to include a while loop where you write and/or read from the DAQ device inside the loop,
- E.g. read values from one or more sensors, e.g., a Temperature sensor that are connected to the DAQ device
- You may want to create a control system reading the process value and then later write the calculated control signal (e.g. using a PID controller) back to the DAQ device and the process

Analog Out (Write)

Hans-Petter Halvorsen

Analog Out (Write)

- Note! The USB-6008 can only output a voltage signal between 0 and 5V
- The USB-6008 has 2 Analog Out Channels:
  - **AO0**
  - **AO1**



Analog Out (Write)

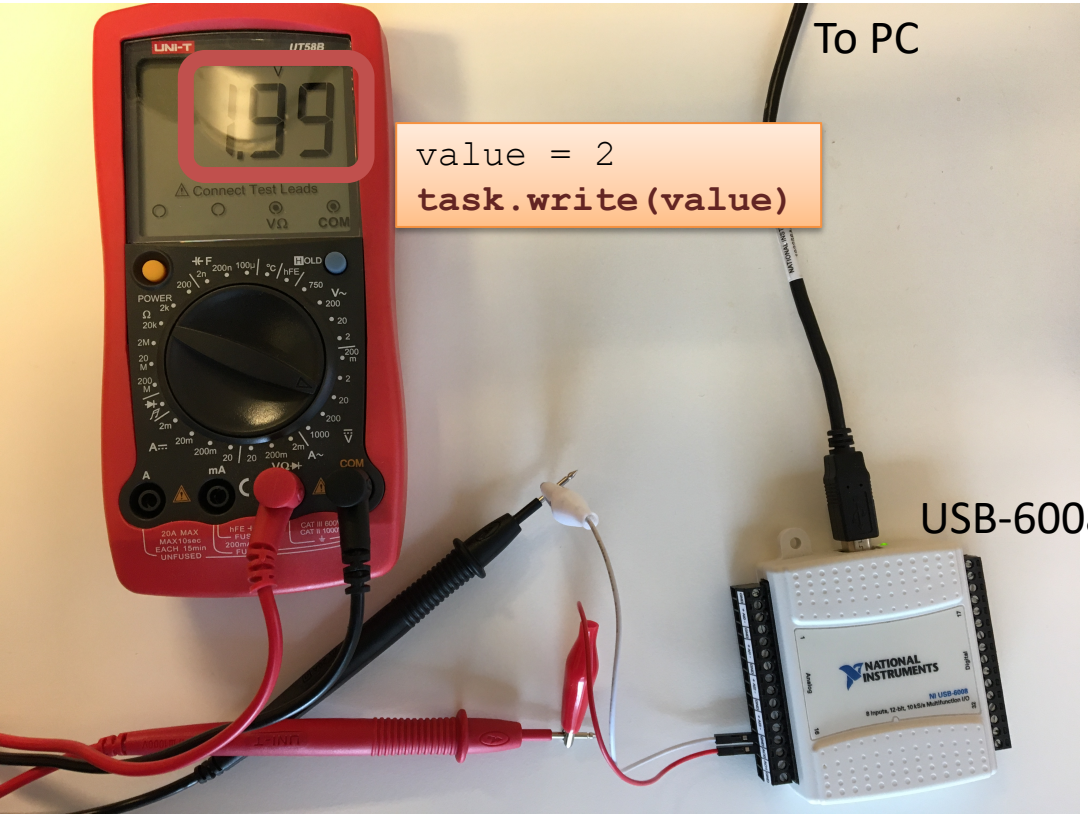
```
import nidaqmx

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()
```

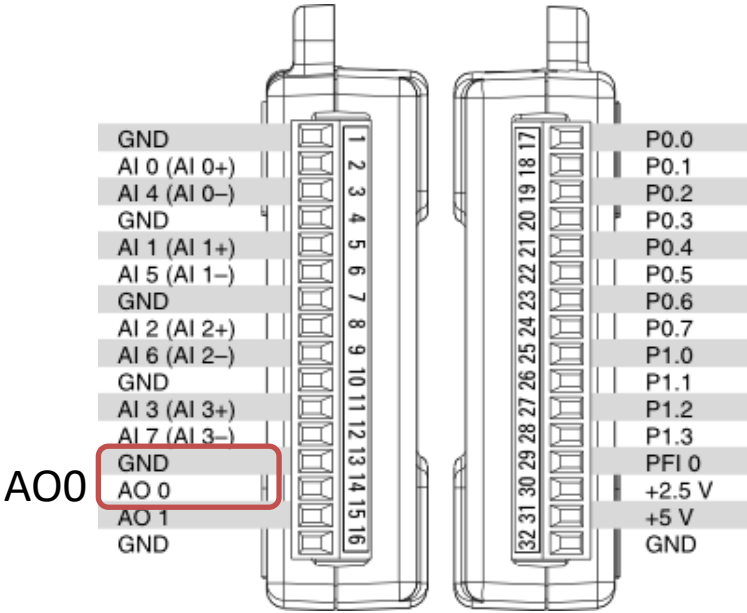
You can, e.g., use a **Multimeter** in order to check if the the program outputs the correct value

Hardware Setup and Testing

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

Multimeter





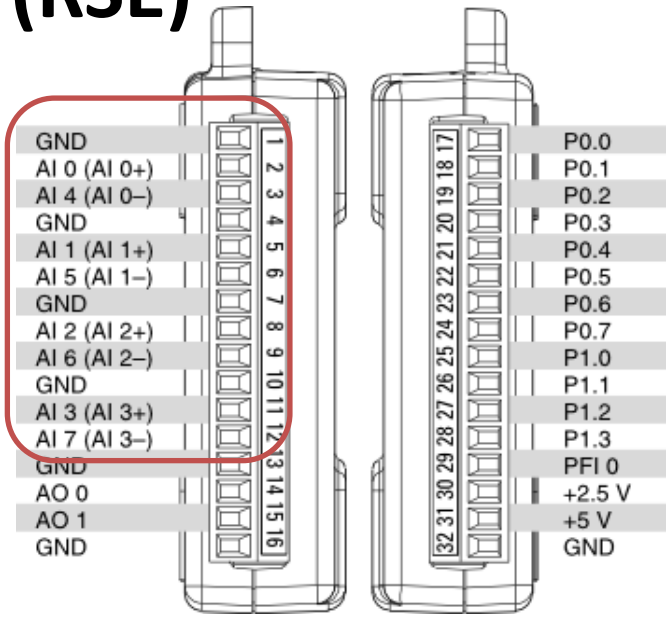
Analog In (Read)

Hans-Petter Halvorsen

Analog In (Read)

USB-6008 has

- 8 AI **Referenced Single Ended (RSE)** Analog Inputs Channels
- or 4 AI **Differential** Analog Inputs Channels Default



The Voltage Range is $-10V - 20V$

$0V - 5V$ is default

Analog In (Read)

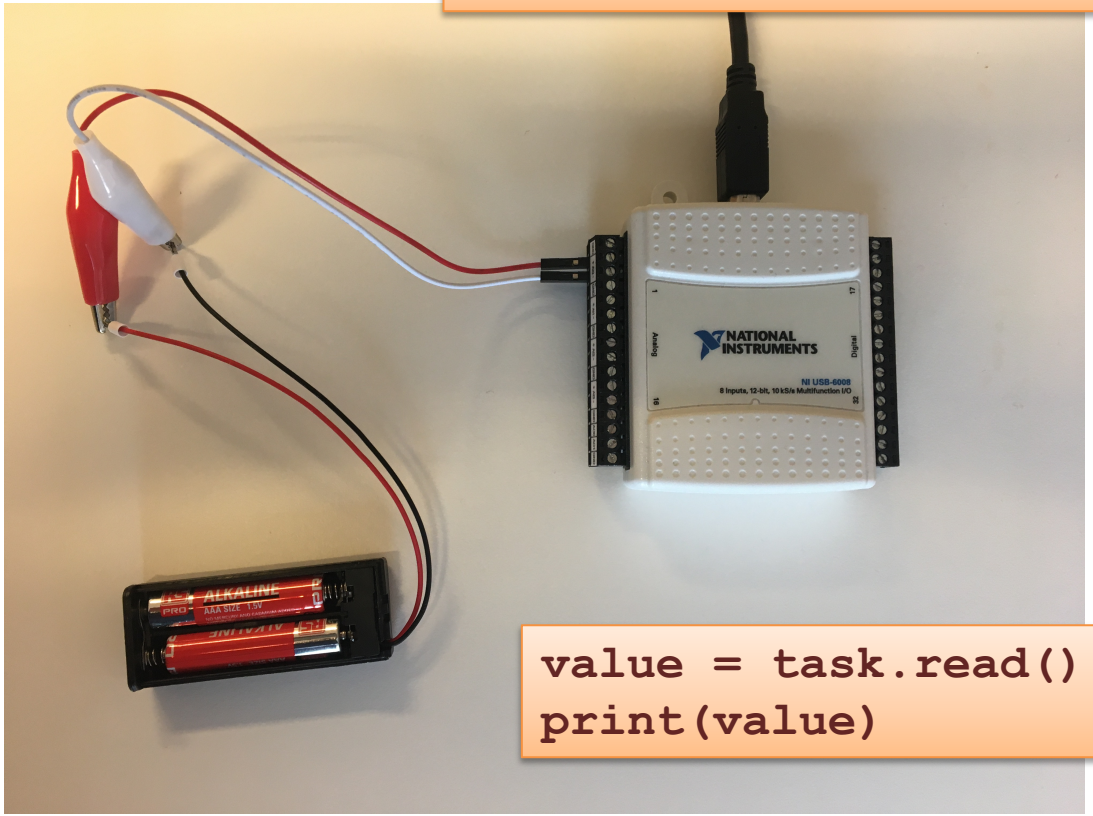
- In order to test your application you can connect, e.g., a battery to the Analog Input channel used in the program.
- Before you connect the battery to the DAQ device you can check the Voltage Level using a Multimeter.
- Another test is to combine the Analog Write and Analog Read examples and wire the Analog Write channel directly to the Analog Read channel used in the program, this is a so-called “**Loop-back Test**”.

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()
```

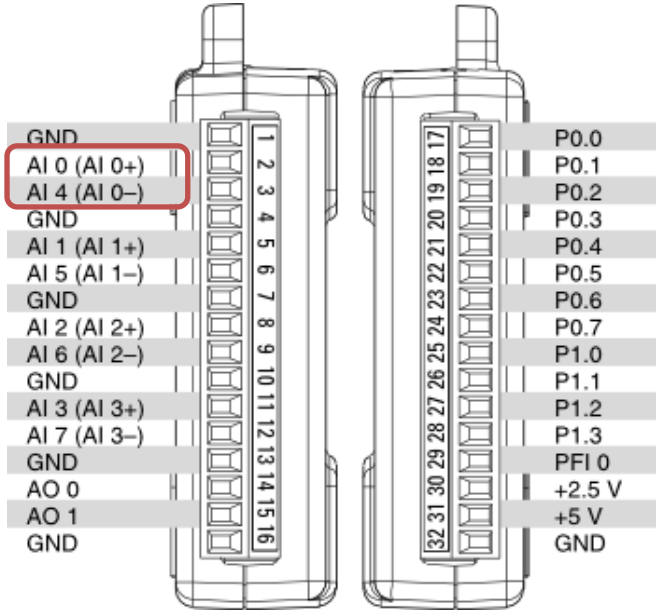
Hardware Setup and Testing

```
task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
```



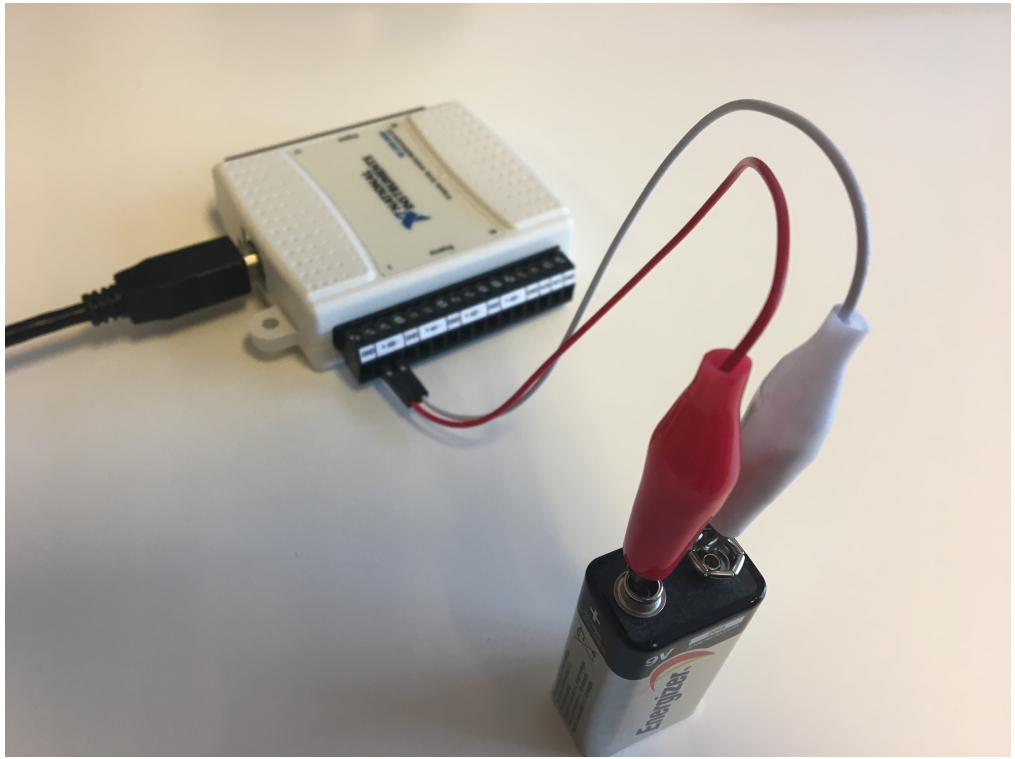

```
value = task.read()
print(value)
```

AI0

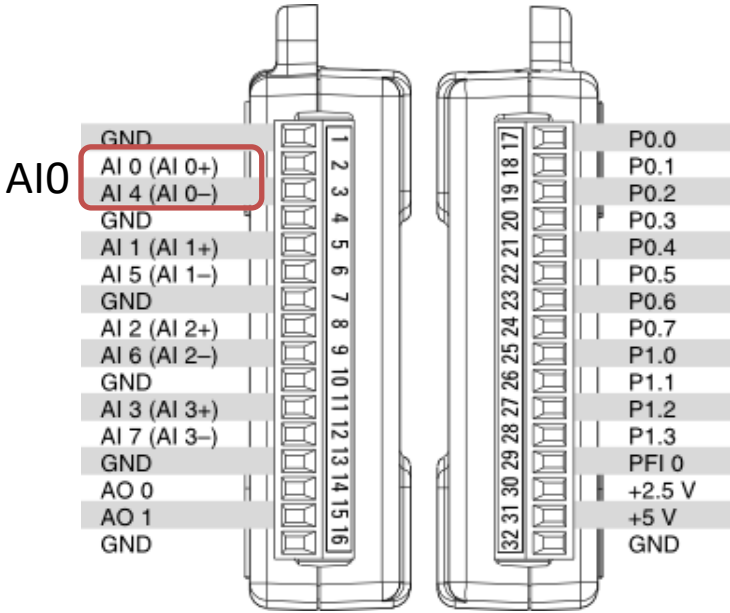


Hardware Setup and Testing

```
task.ai_channels.add_ai_voltage_chan("Dev1/ai0", min_val=0, max_val=10)
```



```
value = task.read()
print(value)
```



Analog In (Read)

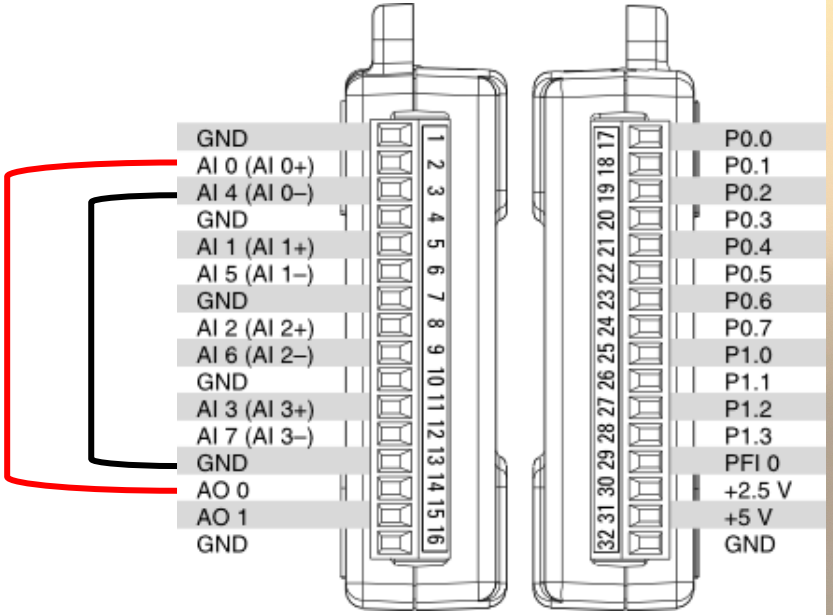
```
import nidaqmx

task = nidaqmx.Task()
task.ai_channels.add_ai_voltage_chan("Dev1/ai0", min_val=0, max_val=10)
task.start()

value = task.read()
print(value)

task.stop
task.close()
```

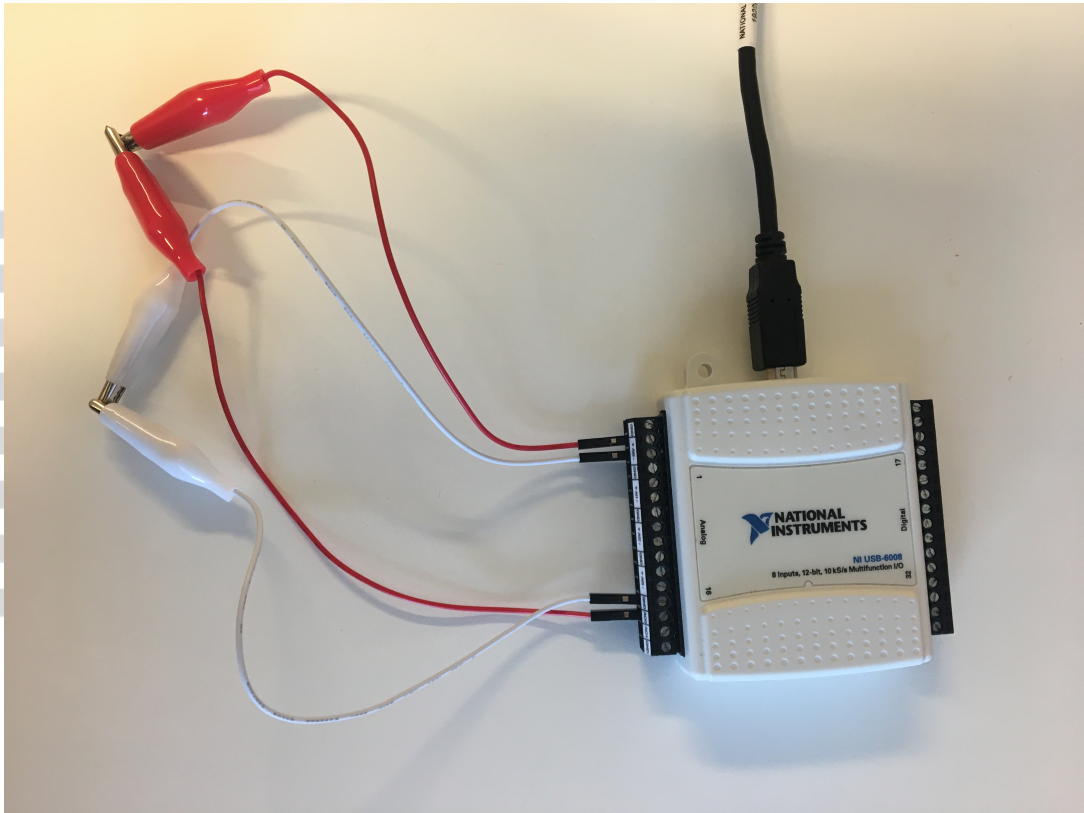
$0V - 5V$ is default

9V Battery

Loopback Test (Out + In)

Connect Analog Out connectors on DAQ device to the Analog In connectors:





Loopback Test - Example

1. First run this Python code:

Analog Out

```
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()
```

2. Then run this Python code:

Analog In

```
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()
```

Loopback Test – Example2

```
import nidaqmx

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

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

value = 4
task_write.write(value)

value = task_read.read()
print(value)

task_write.stop()
task_write.close()

task_read.stop()
task_read.close()
```

Analog Out + analog In
in the same program

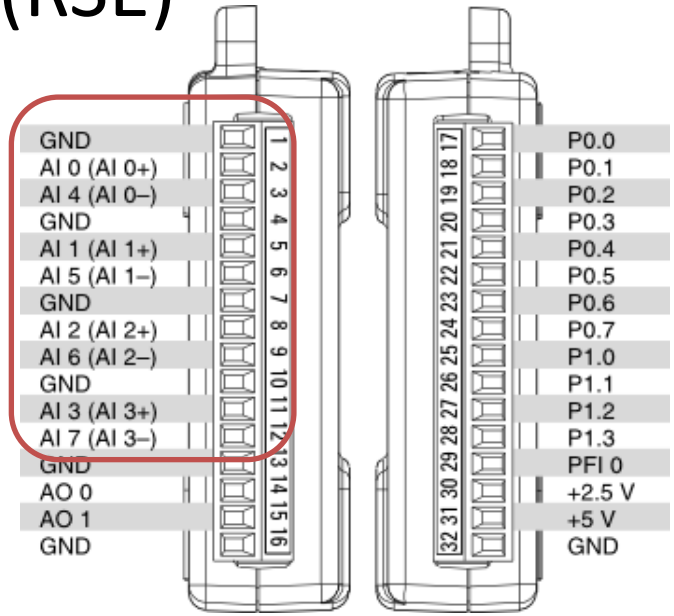
Loopback Test – Example3

```
import nidaqmx
import time

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

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


start=0; stop=6; increment=1

for k in range(start, stop, increment):
    value = k
    if value>5:
        value=5
    task_write.write(value)
    time.sleep(1)

    value = task_read.read()
    print(round(value,2))

task_write.stop()
task_write.close()

task_read.stop()
task_read.close()
```

Analog In – RSE vs Differential

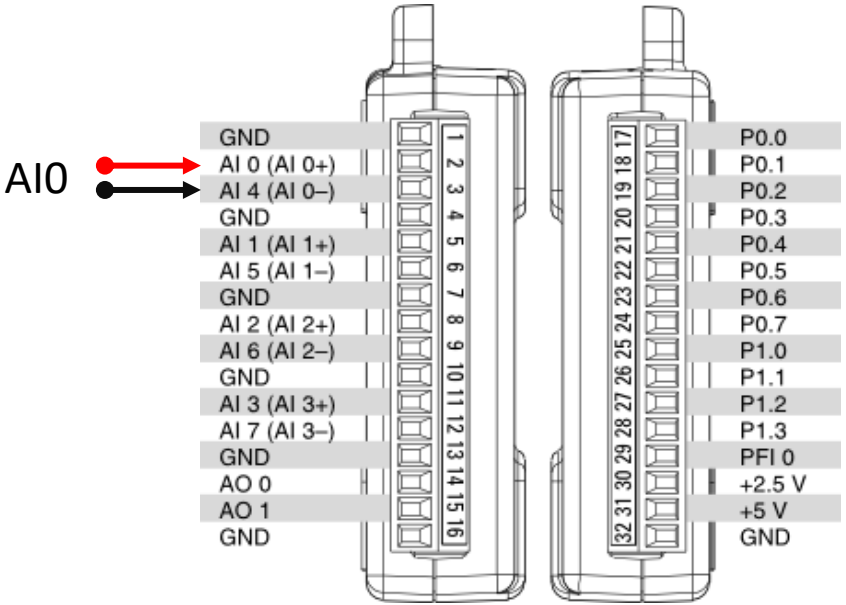
USB-6008 has

- 8 AI Referenced Single Ended (RSE) Analog Inputs Channels
- or 4 AI Differential Analog Inputs Channels



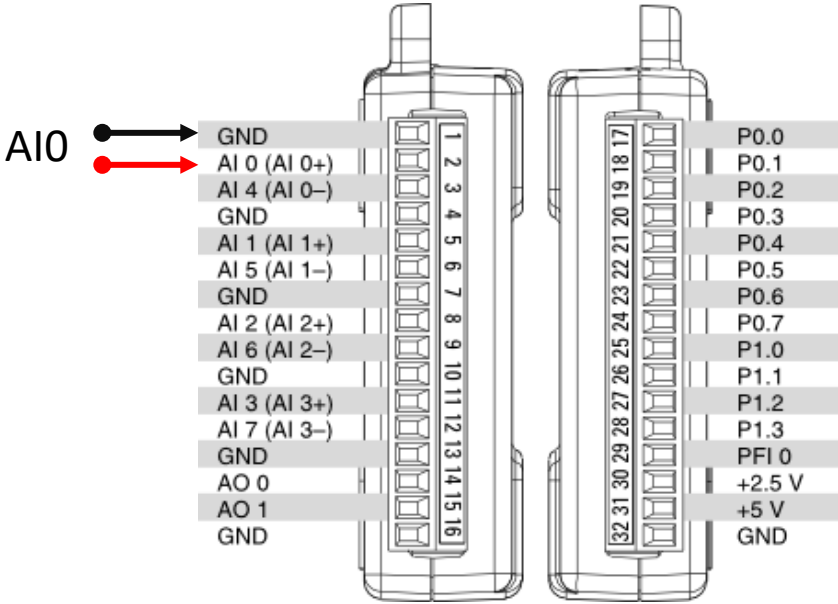
Analog In – RSE vs Differential

AI Differential Analog - 4 channels



AI Referenced Single Ended (RSE) - 8 channels

The Analog Channels have common ground



Analog In with RSE

```
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()
```

Analog In with Differential

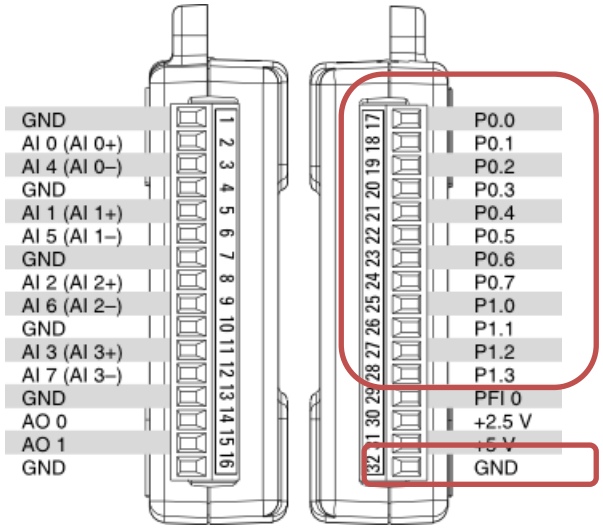
```
import nidaqmx

from nidaqmx.constants import (
    TerminalConfiguration)


task = nidaqmx.Task()

task.ai_channels.add_ai_voltage_chan("Dev1/ai0",
                                     terminal_config=TerminalConfiguration.DIFFERENTIAL)
task.start()

value = task.read()
print(value)

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

Digital I/O

Hans-Petter Halvorsen

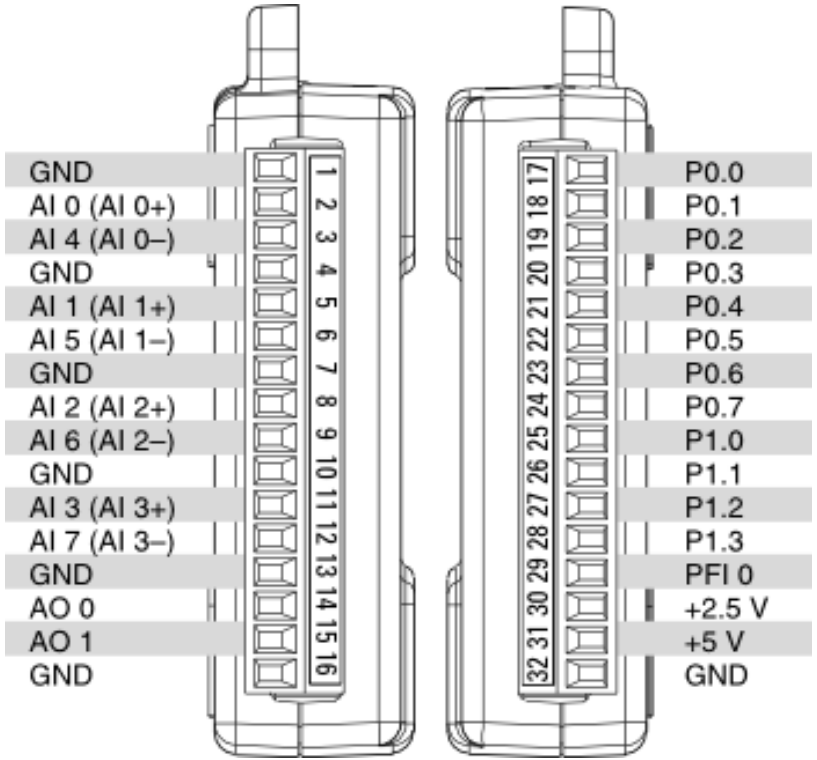
Digital I/O

- 12 Digital Channels
  - Port 0 Digital I/O Channels 0 to 7
  - Port 1 Digital I/O Channels 0 to 3



- You can individually configure each signal as an input or output.

Digital I/O



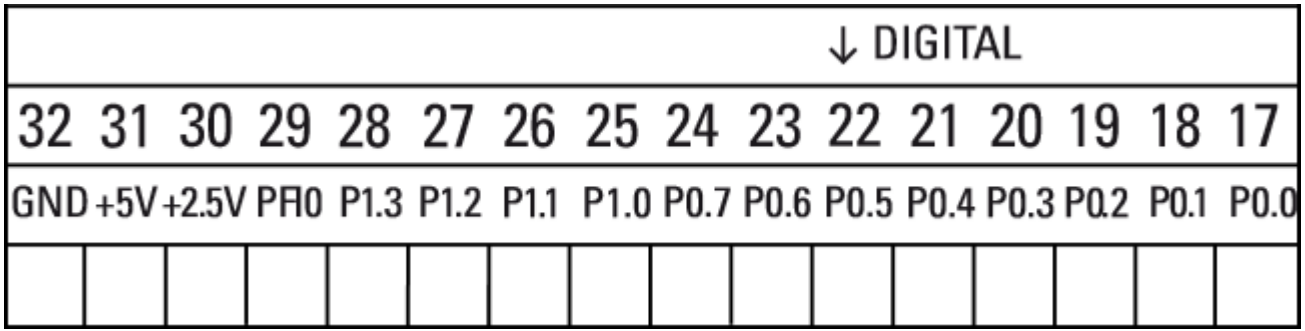
port0

port1

```
Dev1/port0/line0
Dev1/port0/line1
Dev1/port0/line2
Dev1/port0/line3
Dev1/port0/line4
Dev1/port0/line5
Dev1/port0/line6
Dev1/port0/line7
```

```
Dev1/port1/line0
Dev1/port1/line1
Dev1/port1/line2
Dev1/port1/line3
```

Digital I/O



Dev1/Port0/line0:7

P0.<0..7> Port 0 Digital I/O Channels 0 to 7 — You can individually configure each signal as an input or output.

Dev1/Port1/line0:3

P1.<0..3> Port 1 Digital I/O Channels 0 to 3 — You can individually configure each signal as an input or output

Digital Out (Write)

Hans-Petter Halvorsen

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()
```

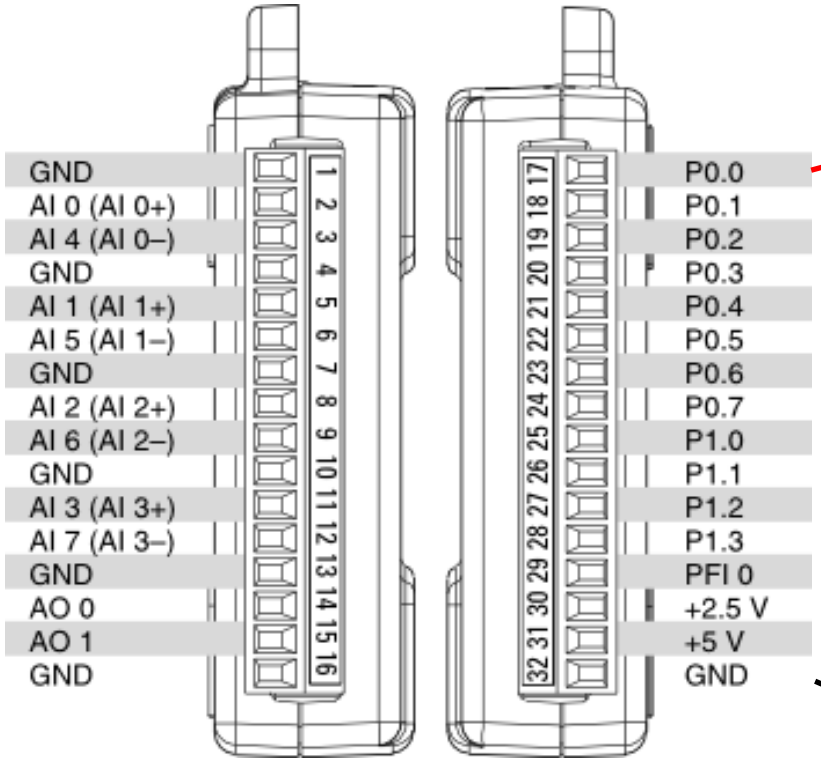
`value = True`

We measure $\sim 5V$ using a Multimeter

`value = False`

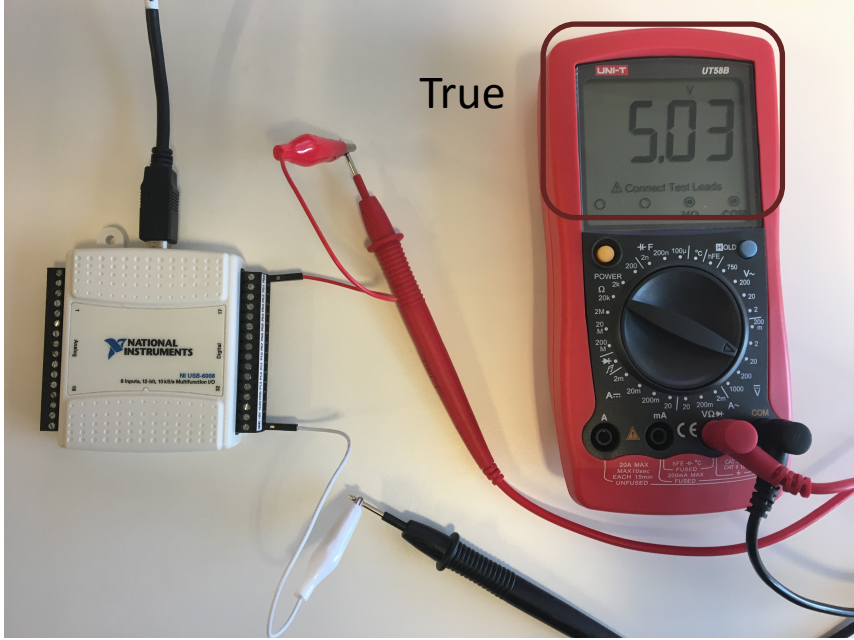
We measure $\sim 0V$ using a Multimeter

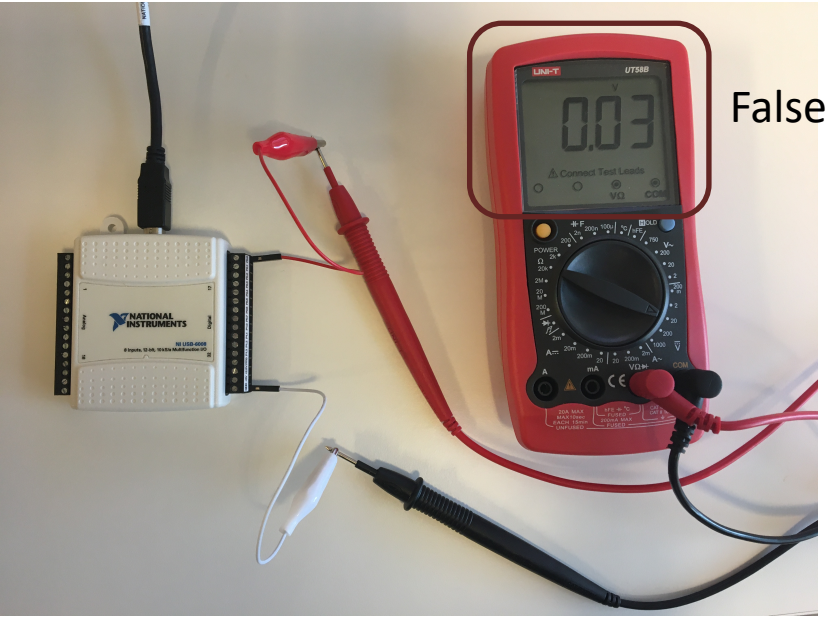
Digital Out (Write)



\+ We test with
\- a Multimeter

```
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()
```



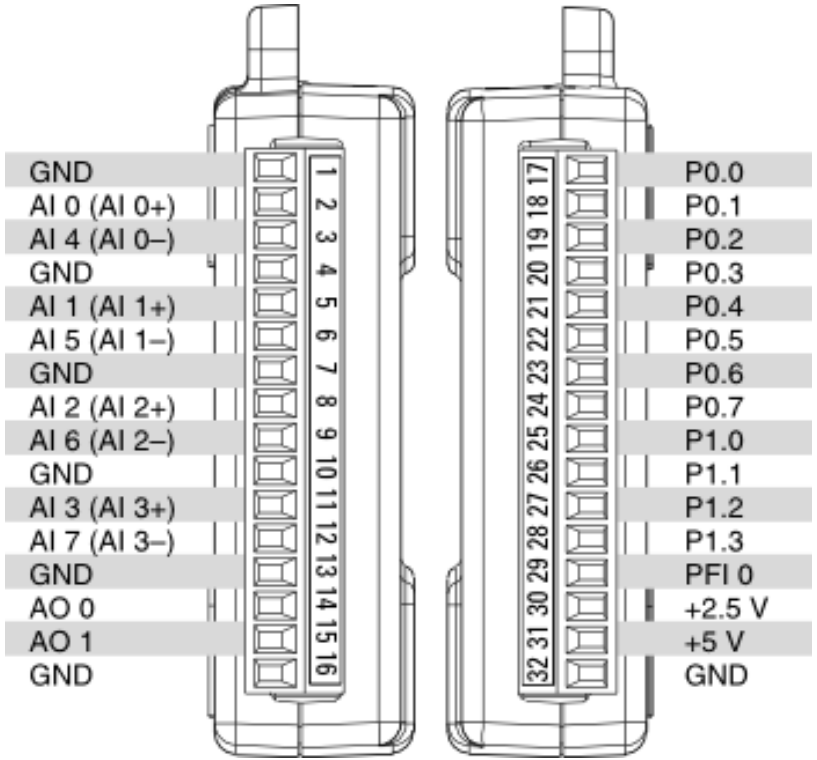



```
import nidaqmx

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

value = False
task.write(value)

task.stop
task.close()
```

Digital Out – For Loop

```
import nidaqmx
import time

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

value = True

N = 10
for k in range(N):
    print(value)
    task.write(value)
    value = not value
    time.sleep(5)

task.stop
task.close()
```

Digital I/O



port0

port1

```
Dev1/port0/line0
Dev1/port0/line1
Dev1/port0/line2
Dev1/port0/line3
Dev1/port0/line4
Dev1/port0/line5
Dev1/port0/line6
Dev1/port0/line7
```

```
Dev1/port1/line0
Dev1/port1/line1
Dev1/port1/line2
Dev1/port1/line3
```

Multiple Digital Out

```
import nidaqmx
import time

from nidaqmx.constants import (
    LineGrouping)

task = nidaqmx.Task()
task.do_channels.add_do_chan("Dev1/port0/line0:7","MyDOChannels",
                             line_grouping=LineGrouping.CHAN_PER_LINE)
task.start()

data = [True,False,True,True,False,True,False,True]
task.write(data)
time.sleep(5)

data[1] = True
task.write(data)

task.stop
task.close()
```

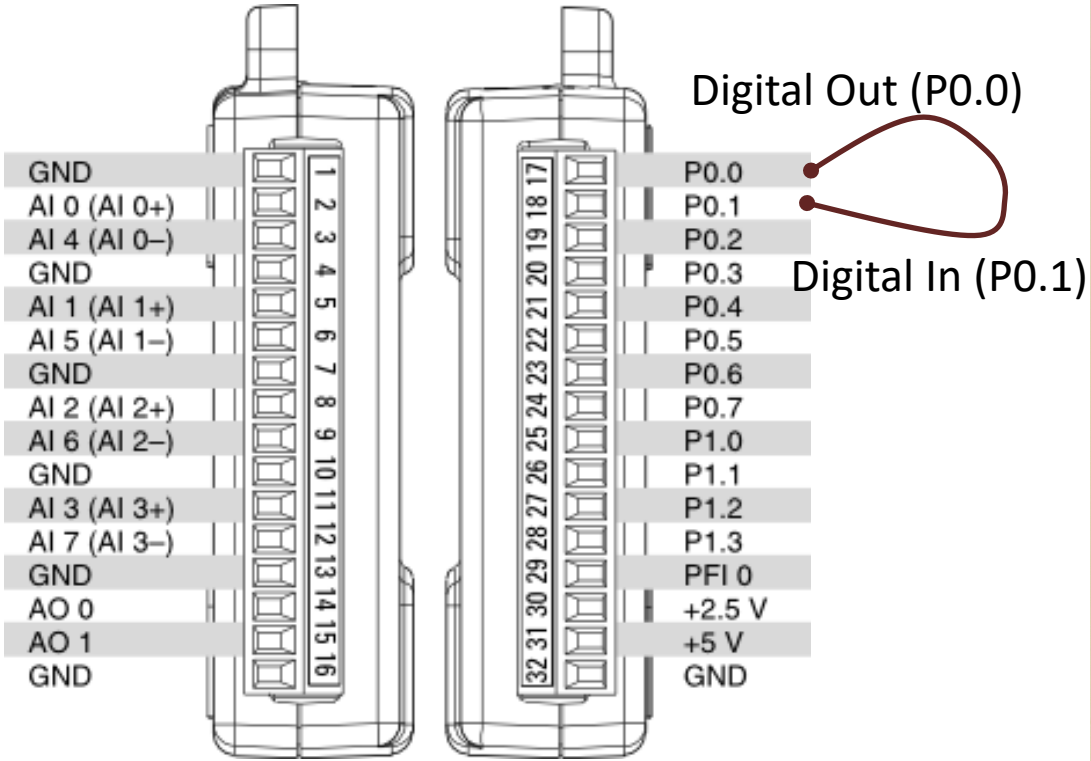
Digital In (Read)

Hans-Petter Halvorsen

Digital In (Read)

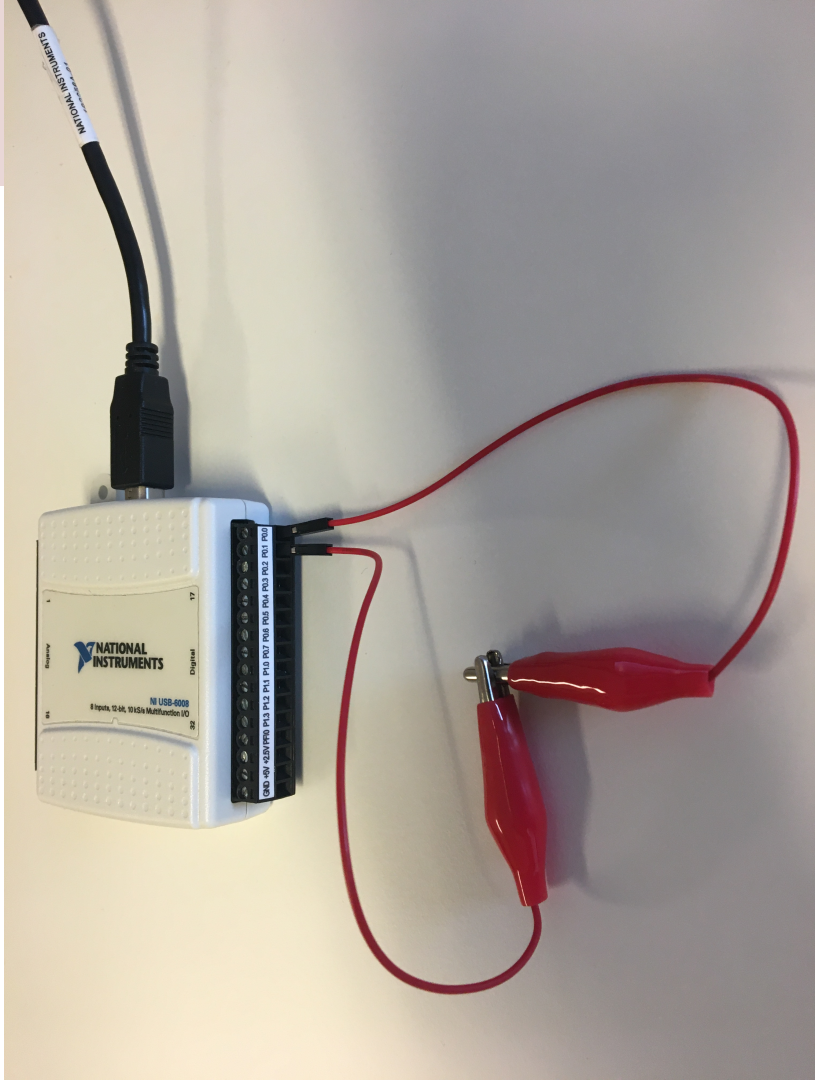
```
import nidaqmx

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

value = task.read()
print(value)

task.stop
task.close()
```

Loopback Testing



Digital Out (P0.0)

Digital In (P0.1)



Digital Out + In

DO

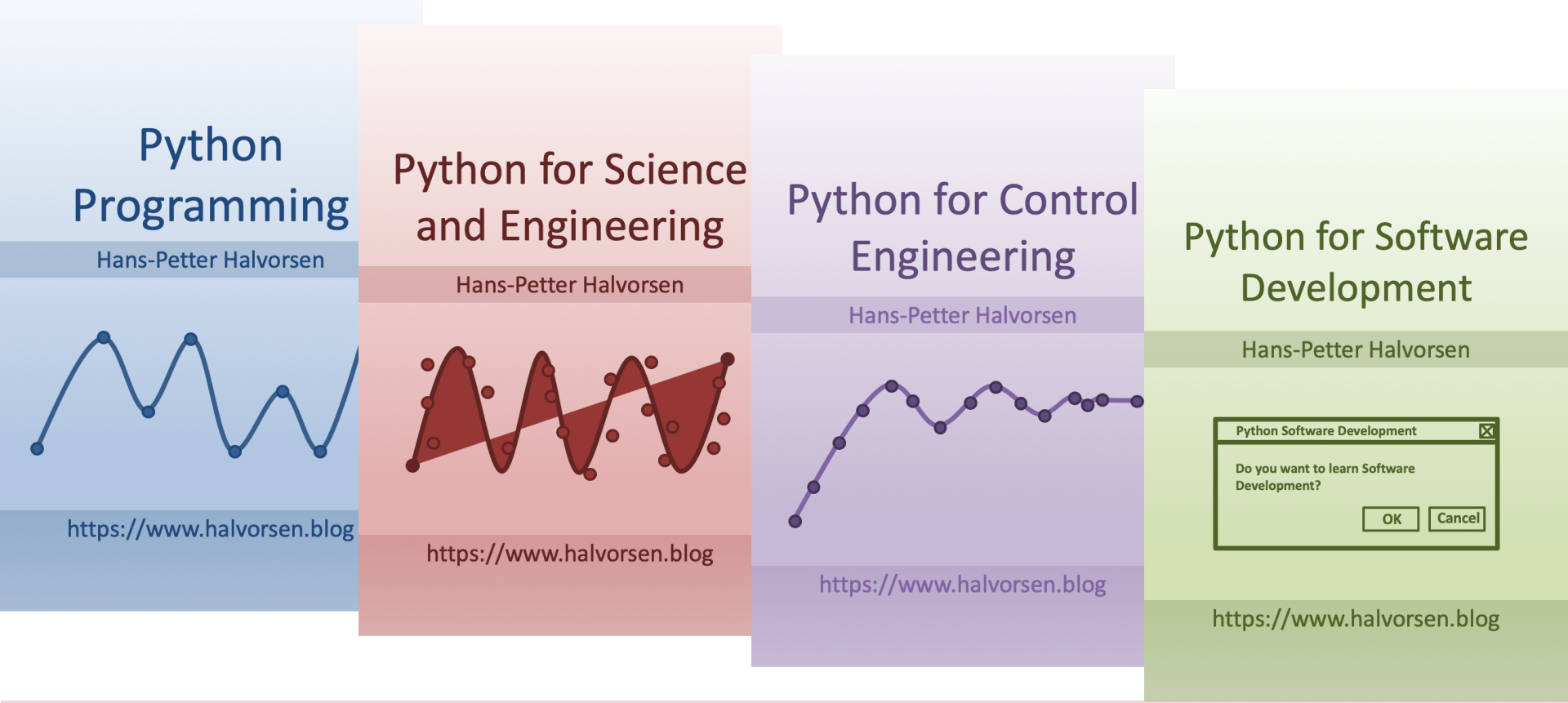
```
import nidaqmx

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

value = False #Change between True and False
task.write(value)

task.stop
task.close()
```

DI

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

value = task.read()
print(value)

task.stop
task.close()
```

In this example we connect DO0 and DI1

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