SENSIRION

THE SENSOR COMPANY

Python in the Hardware Industry

Raphael Nestler (@rnestler on)

February 17, 2017

Sensirion AG

SENSIRION
THE SENSOR COMPANY

Outline

1. How Sensirion Uses Python

2. Growing Pains

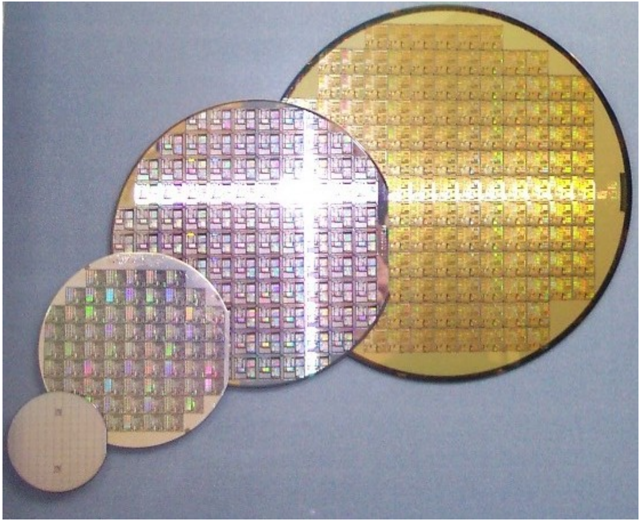
3. Our Solution

How Sensirion Uses Python

An Embarrassingly Short Introduction To Sensirion

An Embarrassingly Short Introduction To Sensirion

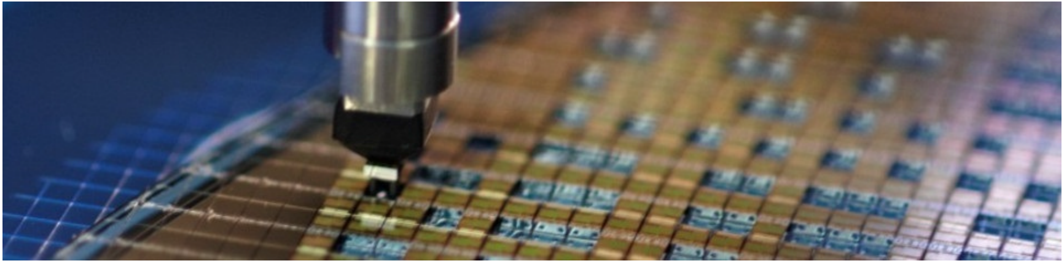
We turn these:



- Custom ASIC
- Produced with a standard CMOS process
- Delivered to us as wafers

An Embarrassingly Short Introduction To Sensirion

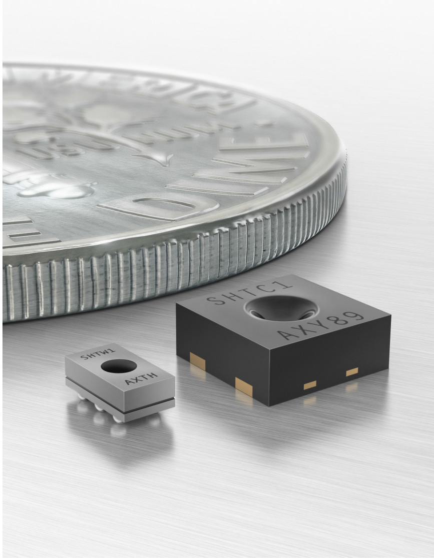
With lots of magic:



- Testing the ASIC
- Cutting the wafer
- Adding out magic sauce (the sensor)
- Calibrate

An Embarrassingly Short Introduction To Sensirion

Into those:

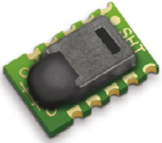


- The final sensor
- Integrated on one chip
- Fully calibrated
- Digital interface to measure

And Make Them Tinier And Tinier…

2001

First digital RH/T sensor



5x7.5x2.5mm 2.4-5.5V

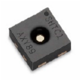
2010

First DFN package RH/T sensor

3x3x1.1mm 2.1-3.6V

2012

World's smallest RH/T for Consumer Electronics



2x2x0.8mm 1.8V

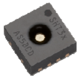
2014

First Chips Scale Package & World's smallest RH/T Sensor

1.3x0.7x0.6mm 1.8V

2015

Most versatile and smallest Automotive Grade RH/T Sensor



2.4x2.4x0.9mm 2.4-5.5V

We Are a Hardware Company

- We produce Hardware not Software

We Are a Hardware Company

- We produce Hardware not Software
- But we use in house developed Software everywhere
  - Production critical Software written in C#
  - Python used in automation, data-analysis, R&D purpose, laboratory measurements

We Are a Hardware Company

- We produce Hardware not Software
- But we use in house developed Software everywhere
  - Production critical Software written in C#
  - Python used in automation, data-analysis, R&D purpose, laboratory measurements
    → Written by non Software Engineers

Life Cycle of a Sensor

During research and development a new sensor goes roughly through these (horribly simplified) stages:

1. Early experimentation
2. First prototype
3. First Silicon
4. Qualification
5. 0-Series
6. Final Product

Life Cycle of a Sensor

During research and development a new sensor goes roughly through these (horribly simplified) stages:

1. Early experimentation
2. First prototype
3. First Silicon
4. Qualification
5. 0-Series
6. Final Product

During steps 1-4 lots of software work is done in the lab with Python.

How Sensirion Uses Python

Some Examples

Example: Data Analysis

- Pandas[1] is very powerful for data processing
- jupyter notebooks are awesome for interactive work
- PyQt (PySide[2]) can be used to create GUIs for recurring analysis
- Two Types of Data
  - Wafer (Sensor) data
  - Experiment data

[1] Python Data Analysis Library: `http://pandas.pydata.org/`

[2] Python binding of the cross-platform GUI toolkit Qt: `https://wiki.qt.io/PySide`

Example: Data Analysis - Wafer Data

- Data comes from many sources in many formats
  - Supplier delivered data (CSV, Excel, JSON, ...)
  - Sensirion Internal Data (SQL, CSV)
- Formats change over time! (Even from the same supplier)

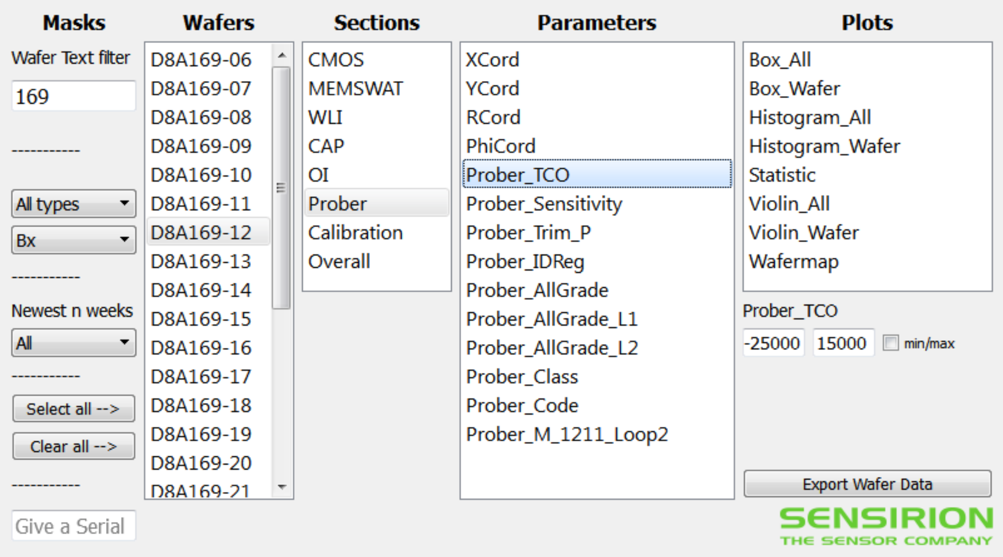
Example: Data Analysis - Wafer Data

- Data comes from many sources in many formats
  - Supplier delivered data (CSV, Excel, JSON, ...)
  - Sensirion Internal Data (SQL, CSV)
- Formats change over time! (Even from the same supplier)

  → Reformat to standard csv format

  → Store it systematically
- Python Scripts with quick iterations (New data → new workarounds for conversion)

Example: Data Analysis - Wafer Visualization



Wafer Text filter: 169

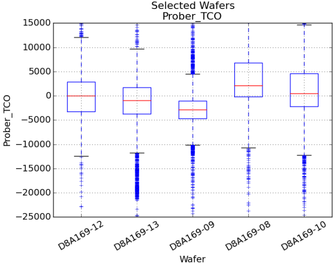
All types

Bx

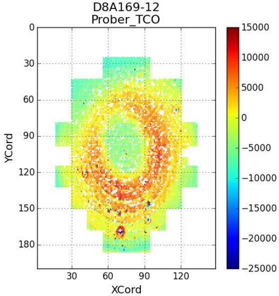
Newest n weeks: All

Prober_TCO: -25000 15000 □ min/max

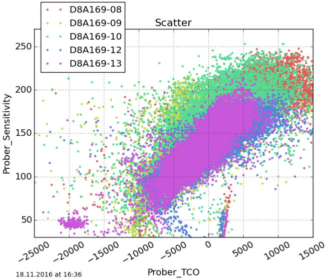
Example: Data Analysis - Wafer Visualization



18.11.2016 at 16:34

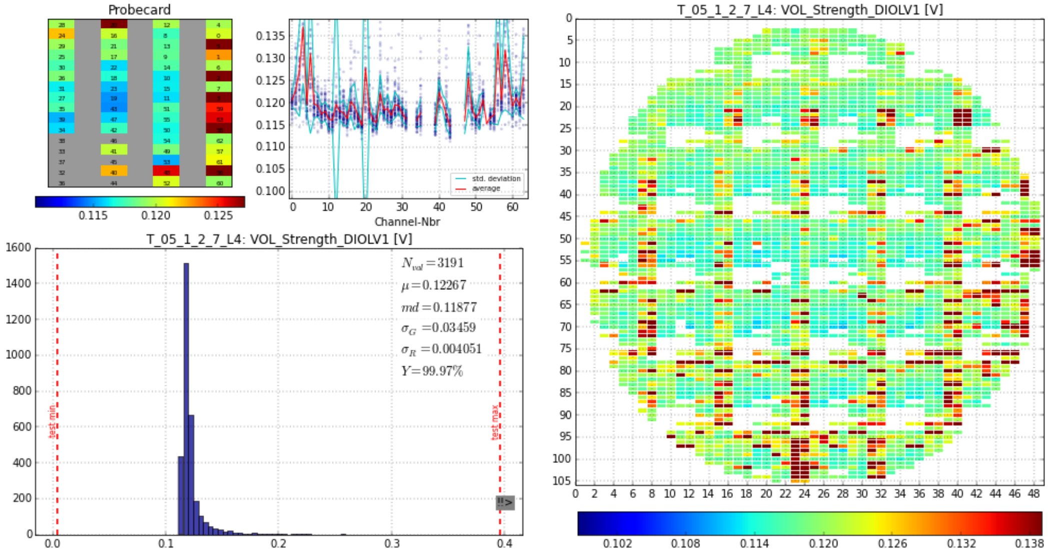


18.11.2016 at 16:33



18.11.2016 at 16:36

Example: Data Analysis - Wafer Visualization



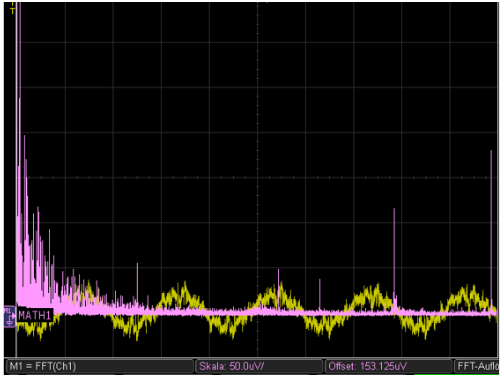
Example: Data Analysis - Conclusions

- Pandas and PySide are very powerful tools for data analysis
- Standardize the input data format (and convert if necessary) and data storage
  → Consistent evaluation, always find your data
- Standardize the presentation of data
  → Everybody understands the plots

Example: Noise Analysis on Electronics

We had a problem with noise on certain hardware:

Guter Kanal (BW 60KHz)



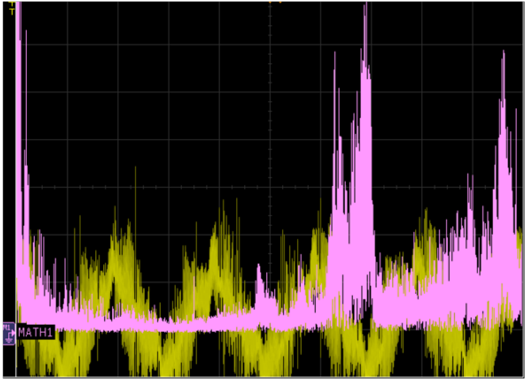
M1 = FFT(Ch1)

Skala: 50.0uV/

Offset: 153.125uV

FFT-Auflö

Schlechter Kanal (BW 60KHz)



Erhöhte Rauschenergie ab ca. 18KHz

Example: Noise Analysis on Electronics

So we recorded the noise and analysed it:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
...
# some math...
def AggregateSpectralEnergy(x):
    fft = np.fft.fft(x.values)
    fs = 1.0/T
    N = len(x.values)
    dF = (fs/N)
    return np.sum(np.abs(fft[np.floor(lowPass/dF):np.floor(N/2)])*2.0/N)
```

Example: Noise Analysis on Electronics

```
# some data...
for i in range(8):
    fine.append(pd.read_csv(fineFiles+str(i)+'.csv'))
    fine[i].drop('Sample', 1, inplace = True)
    fine[i].columns = fine[i].columns.astype(int)
    crappy.append(pd.read_csv(crappyFiles+str(i)+'.csv'))
    crappy[i].drop('Sample', 1, inplace = True)
    crappy[i].columns = crappy[i].columns.astype(int)
```

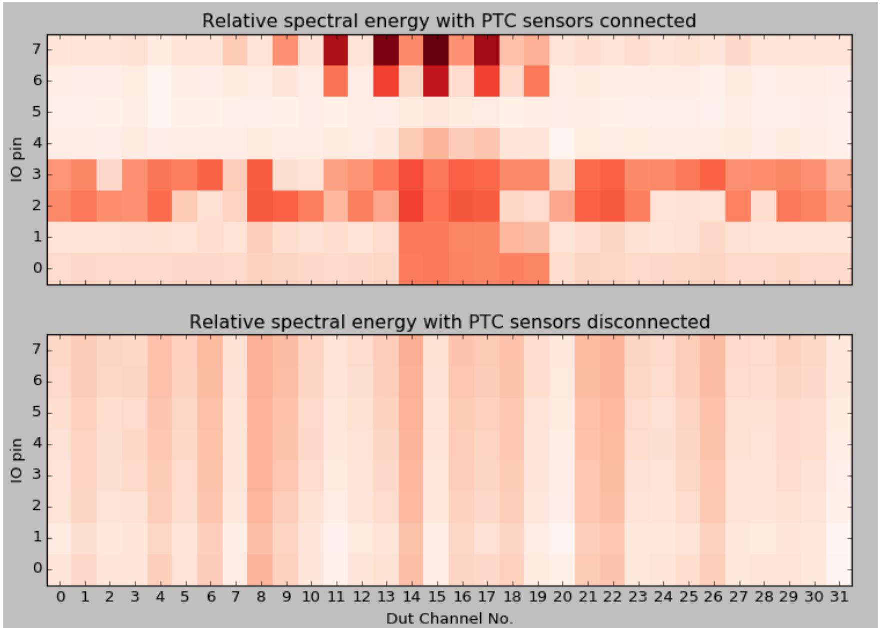
In between more magic and ad hoc code ;)

Example: Noise Analysis on Electronics

```
# some plotting...
axG.pcolor(np.log(goodFrame.values), cmap=plt.cm.Reds, vmin=np.log(1.0), v
axG.set_xlim([0, 32])
axG.set_ylim([0, 8])
axG.set_ylabel('IO pin')
axG.set_yticks(np.arange(0.5, len(goodFrame.index), 1))
axG.set_yticklabels([str(s) for s in goodFrame.index])
axG.set_xticks(np.arange(0.5, len(goodFrame.columns), 1))
axG.set_xticklabels([str(s) for s in goodFrame.columns])
axG.set_xlabel('Dut Channel No.')
axG.set_title('Relative spectral energy  Pilatus South')
plt.show()
```

And finally...

Example: Noise Analysis on Electronics



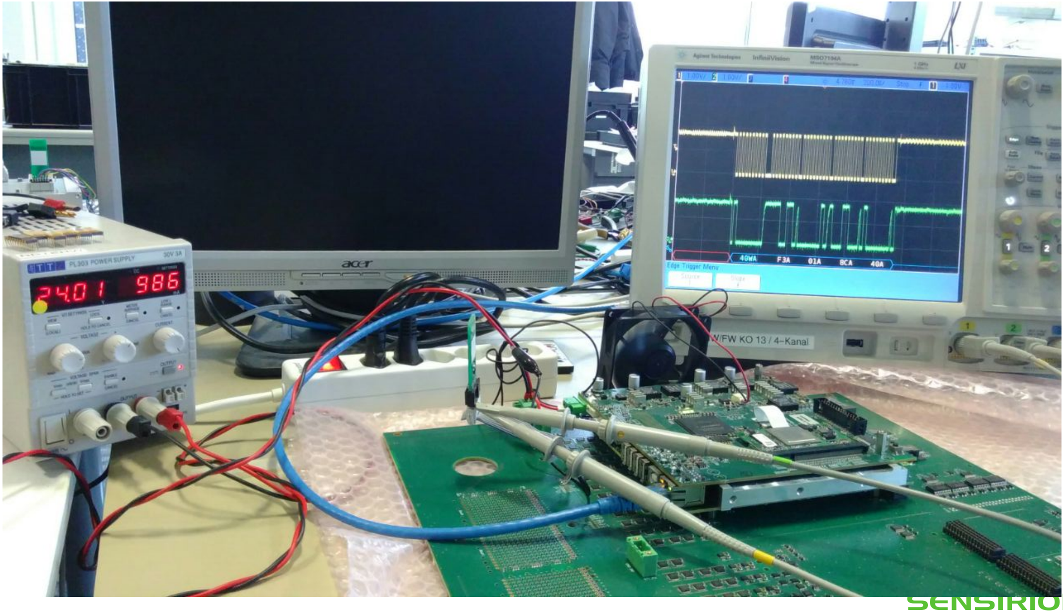
Example: Noise Analysis on Electronics

- We found the noise was specific to some nearby channels
- An external PTC sensor was coupling noise into these channels
- A layout change fixed the issue
- The ”measure and analyze offline” approach saved time!

Example: Automated Hardware Testing

- A lot of time one needs to qualify a small number of prototypes (Sensors, some electronics board, ...)
- Most of the times this involves ad-hoc measurement setups

Example: Automated Hardware Testing

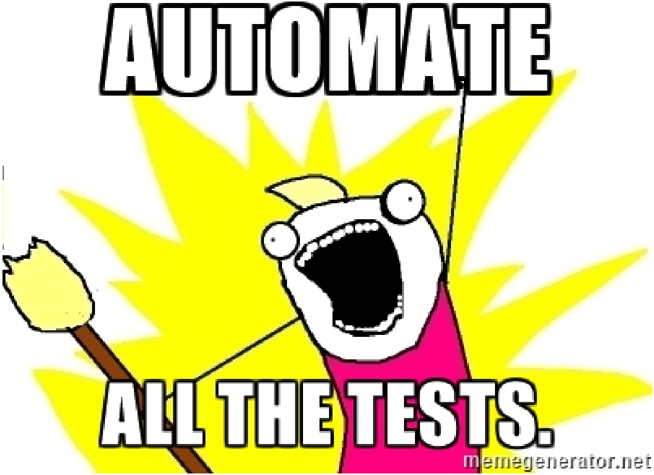


Example: Automated Hardware Testing

- Its tempting to do these tests manually
  - I only have to do it for 5 boards, automating it doesn't scale
- You as software engineers should know the benefits of automated tests ;)

Example: Automated Hardware Testing

- Its tempting to do these tests manually
  - I only have to do it for 5 boards, automating it doesn't scale
- You as software engineers should know the benefits of automated tests ;)



Example: Automated Hardware Testing

- Lots of electronic lab equipment supports either
  - RS232 (if it is old)
  - USB
  - LXI[3] over Ethernet (if it is less old)
  - If you are lucky it supports the IVI[4] API
  - If you are really lucky your device is even supported by python-ivi[5] (If your device is not listed, just try one with a similar name!)

[3] https://en.wikipedia.org/wiki/LAN_eXtensions_for_Instrumentation

[4] http://www.ivifoundation.org/

[5] https://github.com/python-ivi/python-ivi

Example: Automated Hardware Testing

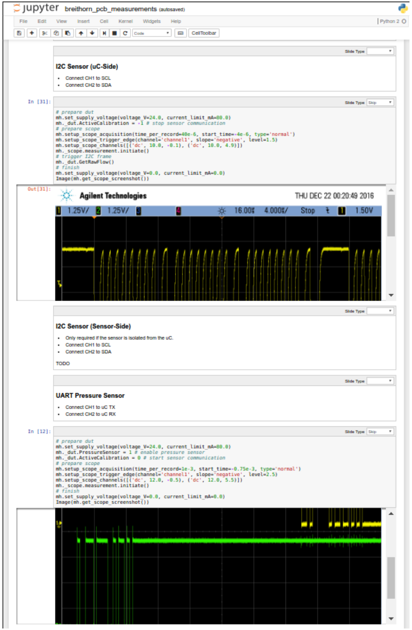
- Lots of electronic lab equipment supports either
  - RS232 (if it is old)
  - USB
  - LXI[3] over Ethernet (if it is less old)
  - If you are lucky it supports the IVI[4] API
  - If you are really lucky your device is even supported by python-ivi[5] (If your device is not listed, just try one with a similar name!)
- So lets automate it and put everything in a jupyter notebook!

[3] https://en.wikipedia.org/wiki/LAN_eXtensions_for_Instrumentation

[4] http://www.ivifoundation.org/

[5] https://github.com/python-ivi/python-ivi

Example: Automated Hardware Testing



I2C Sensor (uC-Side)

- Connect CH1 to SCL
- Connect CH2 to SDA

In [31]:

```
# prepare dut
mh.set_supply_voltage(voltage_V=24.0, current_limit_mA=80.0)
mh._dut.ActiveCalibration = -1 # stop sensor communication
# prepare scope
mh.setup_scope_acquisition(time_per_record=40e-6, start_time=-4e-6, type='normal')
mh.setup_scope_trigger_edge(channel='channel1', slope='negative', level=1.5)
mh.setup_scope_channels([('dc', 10.0, -0.1), ('dc', 10.0, 4.9)])
mh._scope.measurement.initiate()
# trigger I2C frame
mh._dut.GetRawFlow()
# finish
mh.set_supply_voltage(voltage_V=0.0, current_limit_mA=0.0)
Image(mh.get_scope_screenshot())
```

Out[31]:

I2C Sensor (Sensor-Side)

- Only required if the sensor is isolated from the uC.
- Connect CH1 to SCL
- Connect CH2 to SDA

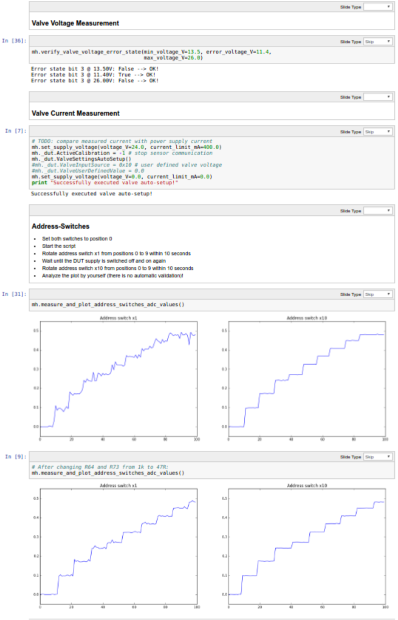
TODO

UART Pressure Sensor

- Connect CH1 to uC TX
- Connect CH2 to uC RX

In [12]:

```
# prepare dut
mh.set_supply_voltage(voltage_V=24.0, current_limit_mA=80.0)
mh._dut.PressureSensor = 1 # enable pressure sensor
mh._dut.ActiveCalibration = 0 # start sensor communication
# prepare scope
mh.setup_scope_acquisition(time_per_record=1e-3, start_time=-0.75e-3, type='normal')
mh.setup_scope_trigger_edge(channel='channel1', slope='negative', level=2.5)
mh.setup_scope_channels([('dc', 12.0, -0.5), ('dc', 12.0, 5.5)])
mh._scope.measurement.initiate()
# finish
mh.set_supply_voltage(voltage_V=0.0, current_limit_mA=0.0)
Image(mh.get_scope_screenshot())
```



Valve Voltage Measurement

In [36]:

```
mh.verify_valve_voltage_error_state(min_voltage_V=13.5, error_voltage_V=11.4,
                                    max_voltage_V=26.0)
```

```
Error state bit 3 @ 13.50V: False --> OK!
Error state bit 3 @ 11.40V: True --> OK!
Error state bit 3 @ 26.00V: False --> OK!
```

Valve Current Measurement

In [7]:

```
# TODO: compare measured current with power supply current
mh.set_supply_voltage(voltage_V=24.0, current_limit_mA=400.0)
mh._dut.ActiveCalibration = -1 # stop sensor communication
mh._dut.ValveSettingsAutoSetup()
#mh._dut.ValveInputSource = 0x10 # user defined valve voltage
#mh._dut.ValveUserDefinedValue = 0.0
mh.set_supply_voltage(voltage_V=0.0, current_limit_mA=0.0)
print "Successfully executed valve auto-setup!"
```

```
Successfully executed valve auto-setup!
```

Address-Switches

- Set both switches to position 0
- Start the script
- Rotate address switch x1 from positions 0 to 9 within 10 seconds
- Wait until the DUT supply is switched off and on again
- Rotate address switch x10 from positions 0 to 9 within 10 seconds
- Analyze the plot by yourself (there is no automatic validation)!

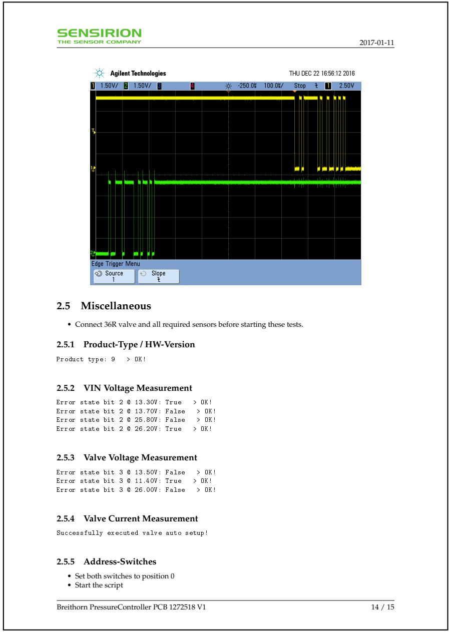
In [31]:

```
mh.measure_and_plot_address_switches_adc_values()
```

In [9]:

```
# After changing R64 and R73 from 1k to 47R:
mh.measure_and_plot_address_switches_adc_values()
```

Example: Automated Hardware Testing - PDF export



2.5 Miscellaneous

- Connect 36R valve and all required sensors before starting these tests.

2.5.1 Product-Type / HW-Version

```
Product type: 9   > OK!
```

2.5.2 VIN Voltage Measurement

```
Error state bit 2 @ 13.30V: True   > OK!
Error state bit 2 @ 13.70V: False   > OK!
Error state bit 2 @ 25.80V: False   > OK!
Error state bit 2 @ 26.20V: True   > OK!
```

2.5.3 Valve Voltage Measurement

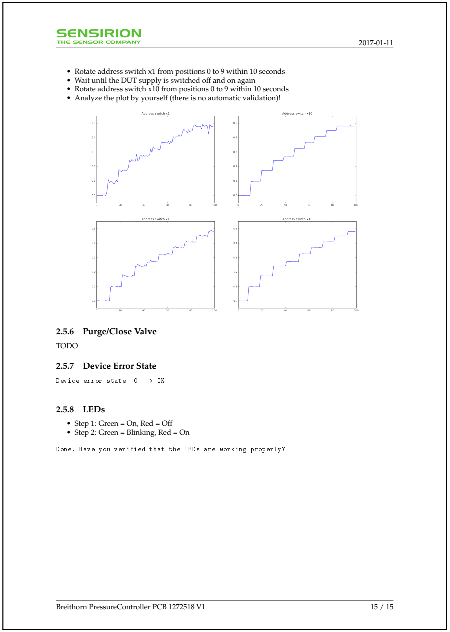
```
Error state bit 3 @ 13.50V: False   > OK!
Error state bit 3 @ 11.40V: True   > OK!
Error state bit 3 @ 26.00V: False   > OK!
```

2.5.4 Valve Current Measurement

```
Successfully executed valve auto setup!
```

2.5.5 Address-Switches

- Set both switches to position 0
- Start the script



- Rotate address switch x1 from positions 0 to 9 within 10 seconds
- Wait until the DUT supply is switched off and on again
- Rotate address switch x10 from positions 0 to 9 within 10 seconds
- Analyze the plot by yourself (there is no automatic validation)!

2.5.6 Purge/Close Valve

TODO

2.5.7 Device Error State

```
Device error state: 0   > OK!
```

2.5.8 LEDs

- Step 1: Green = On, Red = Off
- Step 2: Green = Blinking, Red = On

```
Done. Have you verified that the LEDs are working properly?
```

Example: Automated Hardware Testing

- Reproducible measurements
- Scales for the next 10 prototype you have to test
- Test description / instructions stored together with code
- No fiddling with oscilloscope settings
- You can hand it off to a non-engineer

Example: Verifying Embedded Algorithms

Smart Gadget Development Kit[6]

- Modules consisting of
  - Low Power μC
  - Sensor
  - Some Peripheral
- Used for
  - Compensation
  - Additonal communication protocols
  - Demonstrators
  - ..

[6] `https://www.sensirion.com/products/humidity-sensors/development-kit/`

Example: Verifying Embedded Algorithms

- Reference compensation implemented in Python
- Port to embedded system (C / C++)
  - No floating point
  - Constrained resources
- How do we make sure it still works the same?

[7] SPS-2016 Armin Rigo – CFFI: Call C from Python

Example: Verifying Embedded Algorithms

- Reference compensation implemented in Python
- Port to embedded system (C / C++)
  - No floating point
  - Constrained resources
- How do we make sure it still works the same?
- Use CFFI[7] to call the C-code!

[7] SPS-2016 Armin Rigo – CFFI: Call C from Python

Example: Verifying Embedded Algorithms - A CFFI hack

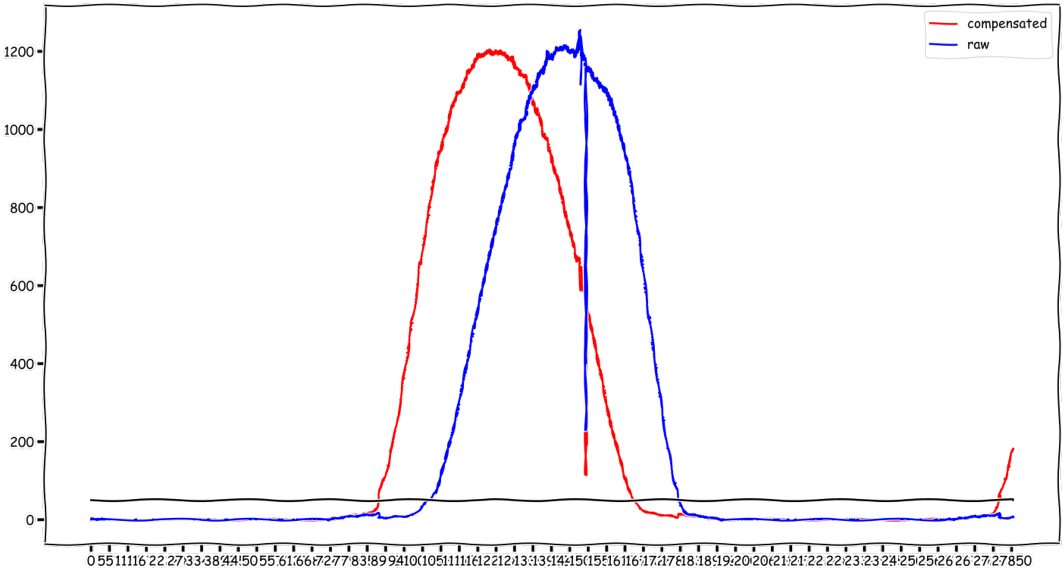
- Plug all your includes together into `AllIncludes.h`
- Preprocess them with `gcc -E`

```
AllIncludes.txt: AllIncludes.h
	gcc -E -P -I${INCLUDE_DIR} AllIncludes.h > AllIncludes.txt
```

- Call it easily with CFFI

```python
from cffi import FFI
ffi = FFI()
lib = ffi.dlopen("./your_library.so")
with open('AllIncludes.txt') as f:
    ffi.cdef(f.read())
lib.lib_call()
```

Example: Verifying Embedded Algorithms - Plotting from Python



Growing Pains

In the beginning everything was easy...

- It was decided we use the Python(x,y) distribution
- Python(x,y) 2.6 was installed by everyone

In the beginning everything was easy...

- It was decided we use the Python(x,y) distribution
- Python(x,y) 2.6 was installed by everyone
  → Every script run on every machine
  → Nobody had to care about dependencies, everything was there

Until Time Passed

- Python(x,y) ships with lots of libraries for the same purpose
  → Sharing code gets difficult
- Python(x,y) 2.6 started to getting outdated
  - Individuals required newer pandas version
  - Some special packages only provided wheels for python 2.7 and upwards
  - ...

Until Time Passed

- Python(x,y) ships with lots of libraries for the same purpose
  → Sharing code gets difficult
- Python(x,y) 2.6 started to getting outdated
  - Individuals required newer pandas version
  - Some special packages only provided wheels for python 2.7 and upwards
  - ...

  → Parts of Sensirion upgraded to Python(x,y) 2.7

Until Time Passed

- Python(x,y) ships with lots of libraries for the same purpose
  → Sharing code gets difficult
- Python(x,y) 2.6 started to getting outdated
  - Individuals required newer pandas version
  - Some special packages only provided wheels for python 2.7 and upwards
  - ...

    → Parts of Sensirion upgraded to Python(x,y) 2.7
- Suddenly code was running only inside the individual groups

Custom Python Installation per Group

Soon every group had their own Python Setup instructions:

- Check that the directories

  ```
  C:\work\SVN\Pressure\Libraries
  C:\work\SVN\Pressure\Tools
  C:\work\SVN\DevelopmentPythonToolbox
  ```

  are checked out from their respective directories.
- copy the folder `C:/work/SVN/PythonDevices` and set PYTHONPATH to it.
- Copy .NET DLLs and enter the path to them in some config
- ...
- Piles and piles of hacks

Subversion as Package Management

People even started inventing their own SVN based packaging and distribution system:

```
logger
  __init__.py
  tags
      __init__.py
      v1_0_0
          __init__.py
           logger.py
    ...
      v1_0_4
         __init__.py
           logger.py
  trunk
  __init__.py
  logger.py
```

Subversion as Package Management

People even started inventing their own SVN based packaging and distribution system:

```
logger
 __init__.py
 tags
   __init__.py
   v1_0_0
     __init__.py
     logger.py
  ...
   v1_0_4
     __init__.py
     logger.py
 trunk
 __init__.py
 logger.py
```

```python
import sr830_driver.tags.v0_1_2.sr830 as sr830
import nidaqmx_driver.tags.v0_1_1.nidaqmx as nidaqmx
```

- This worked surprisingly good!
- But is a maintenance hell!
- In tags only import from other tags
- From trunk import from wherever you like

Some Pain Points

- pythonnet[8] is awesome! Allows to call into existing .NET code

[8] http://pythonnet.github.io/

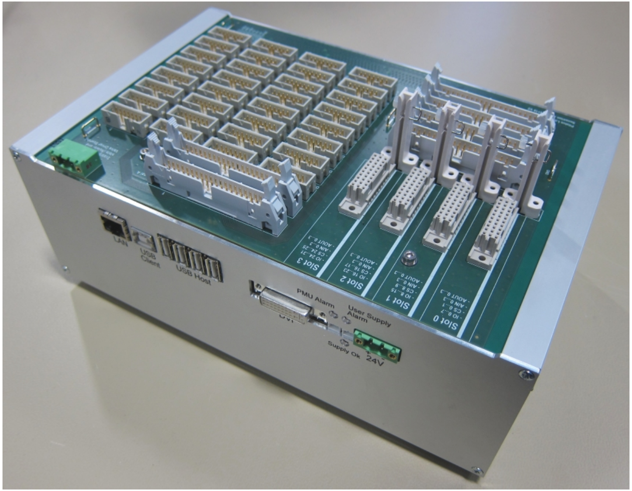
Some Pain Points

- pythonnet[8] is awesome! Allows to call into existing .NET code
- Not so awesome with dependencies between .NET libraries
- Classic diamond dependency hell
- Sometimes random runtime issues with .NET libraries

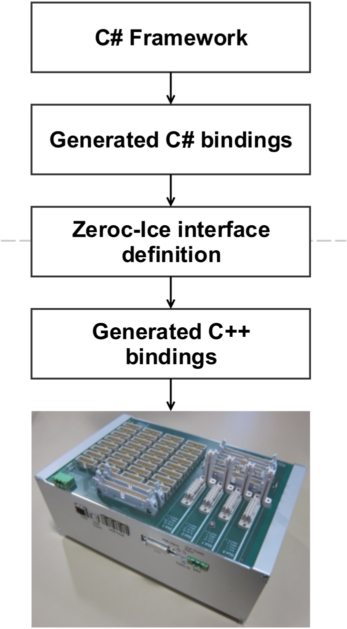
[8] http://pythonnet.github.io/

Some Examples of Over-engineering - Pilatus

- We have an in-house developed test platform called Pilatus
- Used both in production and development



Some Examples of Over-engineering - Pilatus



C# Framework

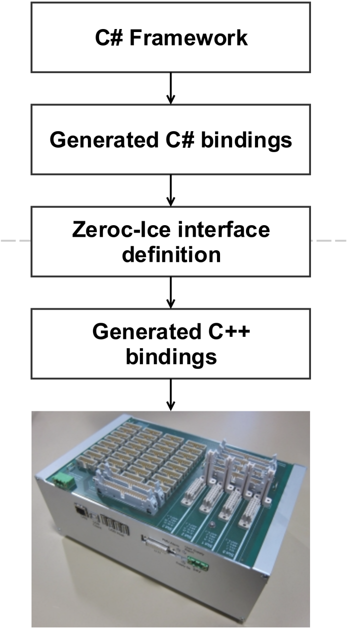
Generated C# bindings

Zeroc-Ice interface definition

Generated C++ bindings

- We use a RPC framework (`https://zeroc.com`) to communicate with it via TCP/IP
- One defines interfaces and can generate code for C#, C++, ...
- Lots of C# code for production

Some Examples of Over-engineering - Pilatus



C# Framework

Generated C# bindings

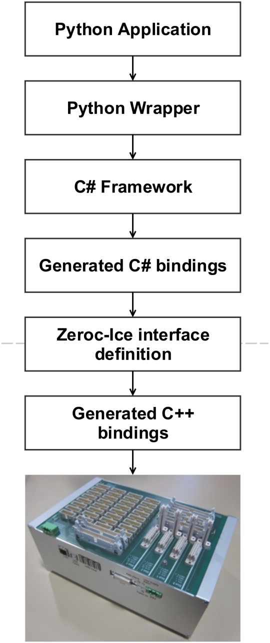
Zeroc-Ice interface definition

Generated C++ bindings

- We use a RPC framework (`https://zeroc.com`) to communicate with it via TCP/IP
- One defines interfaces and can generate code for C#, C++, ...
- Lots of C# code for production
- Lets reuse all this awesome production code in the lab!

Some Examples of Over-engineering - Pilatus

Lets add some Python to it!



Python Application

Python Wrapper

C# Framework

Generated C# bindings

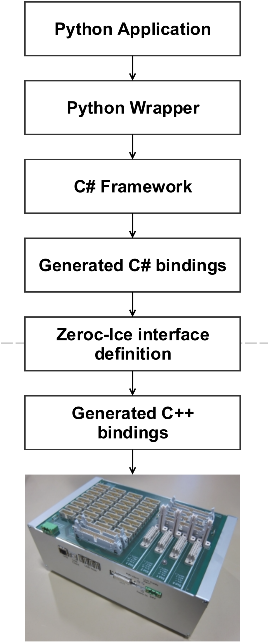
Zeroc-Ice interface definition

Generated C++ bindings

Some Examples of Over-engineering - Pilatus

Lets add some Python to it!

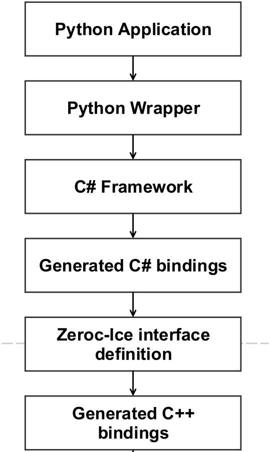
- A change in the Firmware needed to propagate to the top
- Interference with other .NET code (dependency problem)
- In the lab you actually need *low-level* access



Some Examples of Over-engineering - Pilatus

Lets add some Python to it!

- A change in the Firmware needed to propagate to the top
- Interference with other .NET code (dependency problem)
- In the lab you actually need *low-level* access
- I call this Lasagne-code (Too many layers)



Python Application

Python Wrapper

C# Framework

Generated C# bindings

Zeroc-Ice interface definition

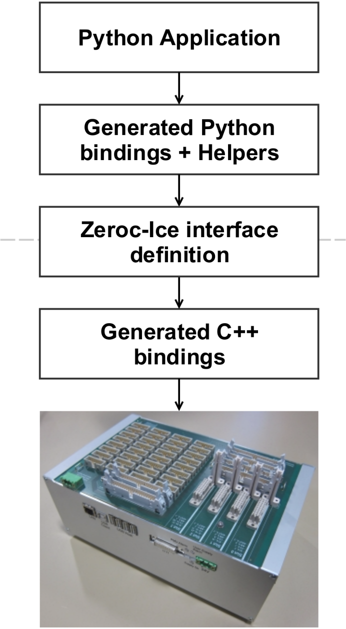
Generated C++ bindings

Some Examples of Over-engineering - Pilatus

Some Examples of Over-engineering - Pilatus

The solution: Generate Python bindings and use them

- No interference with other .NET using libraries
- Immediate access to new functionality
- As low-level as you want


Python Application
Generated Python bindings + Helpers
Zeroc-Ice interface definition
Generated C++ bindings

Lesson Learned

- Don't use a big Python distribution which ships piles and piles of libraries.
- Standardize your base install, but keep it up to date!
- If it is simple to implement in pure python, do it!
- Build proper Python packages for reusable libraries!

Our Solution

Our own Python User Group

- Python User Group (PUG) with experienced Python user from every group → Gather and distribute Python knowledge inside Sensirion

Our own Python User Group

- Python User Group (PUG) with experienced Python user from every group
  → Gather and distribute Python knowledge inside Sensirion

Sensiron PUG mascot

Our own Python User Group

- Python User Group (PUG) with experienced Python user from every group
  → Gather and distribute Python knowledge inside Sensirion
- Provide infrastructure
- Coordinate Sensiron wide updates of the Python base installation
- Collect common requirements and implement reusable packages

Sensiron PUG mascot

Our Solution

Packaging infrastructure

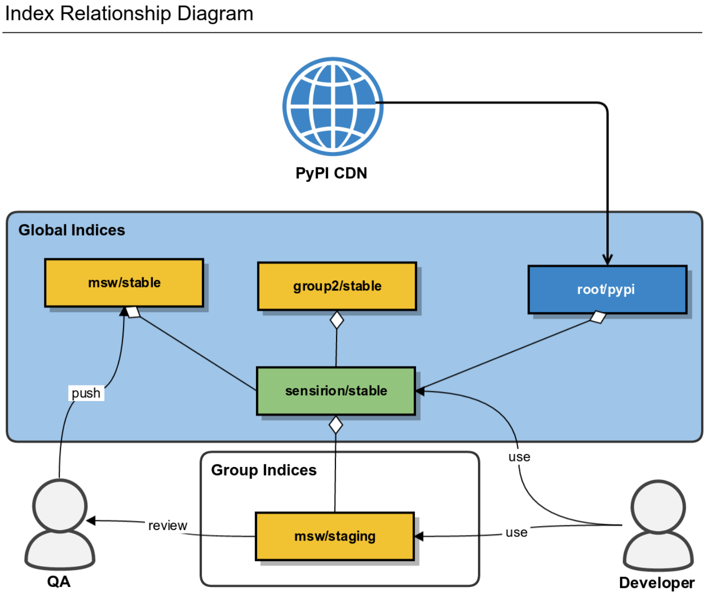
Packaging infrastructure - devpi

We provide a devpi[9] server instance

- *PyPI server and packaging / testing / release tool*
- Mirrors `pypi.org` (performance)
- One staging / stable index per group
- Provide our own wheels for hard to compile packages (numpy, scipy, ...)

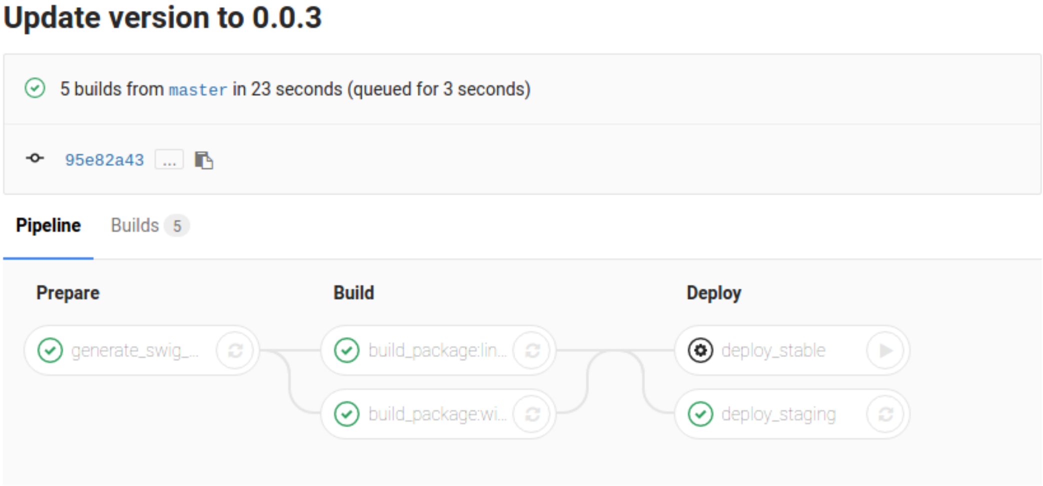
[9] `http://doc.devpi.net/latest/`

Packaging infrastructure - devpi



Packaging infrastructure - Jenkins / GitLab

We use Jenkins and GitLab CI to upload nightly builds to devpi/staging



Update version to 0.0.3

✓ 5 builds from `master` in 23 seconds (queued for 3 seconds)

95e82a43 ...

Our Solution

Standardization

Standardize File Formats

- A lot of Engineers used some kind of CSV formats for data storage

Standardize File Formats

- A lot of Engineers used some kind of CSV formats for data storage
  → Created the Experiment Data Format (EDF). Our internal standard for storing measurements from experiments.
  → Basically CSV with standardized meta data.
- USP of EDF: Can be opened with Excel!

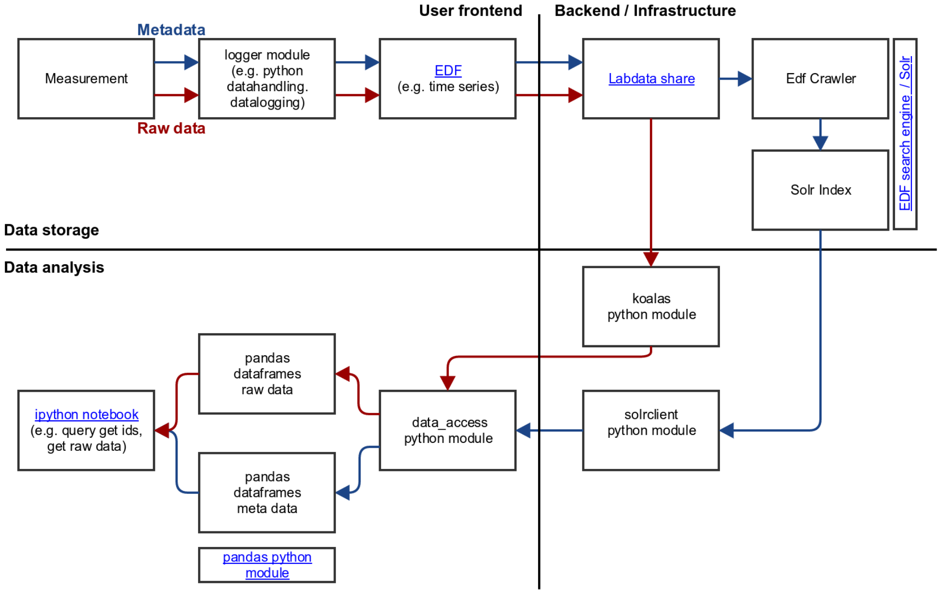
```
# EdfVersion=4.0
# Date=2015-04-23T13:07:10.520000+02:00
# Type=float, Format=.3f    Type=int
Epoch_UTC   Some_Value
1429787230.005  1
```

Standardize File Storage

- Storing the EDF files with standardized metadata and storage place
- Index them with solr[10]

[10] `https://lucene.apache.org/solr/`

Standardize File Storage



Standardize File Storage

```
In [1]: from data_access import solr, load_edf
In [2]: solr.get_fns_by_keywords({'DummyFileType': 'Training'})
Out[2]:
[u'/media/Labdata/DummyForTraining/20160330T161152Z_Example.edf',
 u'/media/Labdata/DummyForTraining/201603291633_ExampleEDF.edf',
 u'/media/Labdata/DummyForTraining/201603291620_ExampleEDF.edf']
```

Standardize File Storage

```
In [1]: from data_access import solr, load_edf
In [2]: solr.get_fns_by_keywords({'DummyFileType': 'Training'})
Out[2]:
[u'/media/Labdata/DummyForTraining/20160330T161152Z_Example.edf',
 u'/media/Labdata/DummyForTraining/201603291633_ExampleEDF.edf',
 u'/media/Labdata/DummyForTraining/201603291620_ExampleEDF.edf']

In [3]: load_edf.get_sensordfs_from_sensor_ids("TrainingDummy01",
    start_date=datetime(2016, 3, 28)).head(3)
Out[14]:
                         SomeValue
Epoch_UTC
2016-03-29 14:09:44.560          0
2016-03-29 14:09:44.661          1
2016-03-29 14:09:44.761          2
```

Summary

- Python is awesome for
  - automated testing in the lab
  - data analysis
  - creating beautiful plots ;)
- Try to establish a common base of packages, but keep it up to date
- Use proper python packages for reusable code
- Standardize your data formats

Thank you!

www.sensirion.com

SENSIRION
THE SENSOR COMPANY