HELMHOLTZ ZENTRUM
DRESDEN ROSSENDORF

PyMeasure

First impressions of the PyMeasure module for experiment control

https://github.com/pymeasure/pymeasure/
https://pymeasure.readthedocs.io/en/stable/

Institute of Fluid Dynamics · Department of Transport Processes at Interfaces · Dr. Till Zürner · t.zuerner@hzdr.de · www.hzdr.de

DRESDEN concept

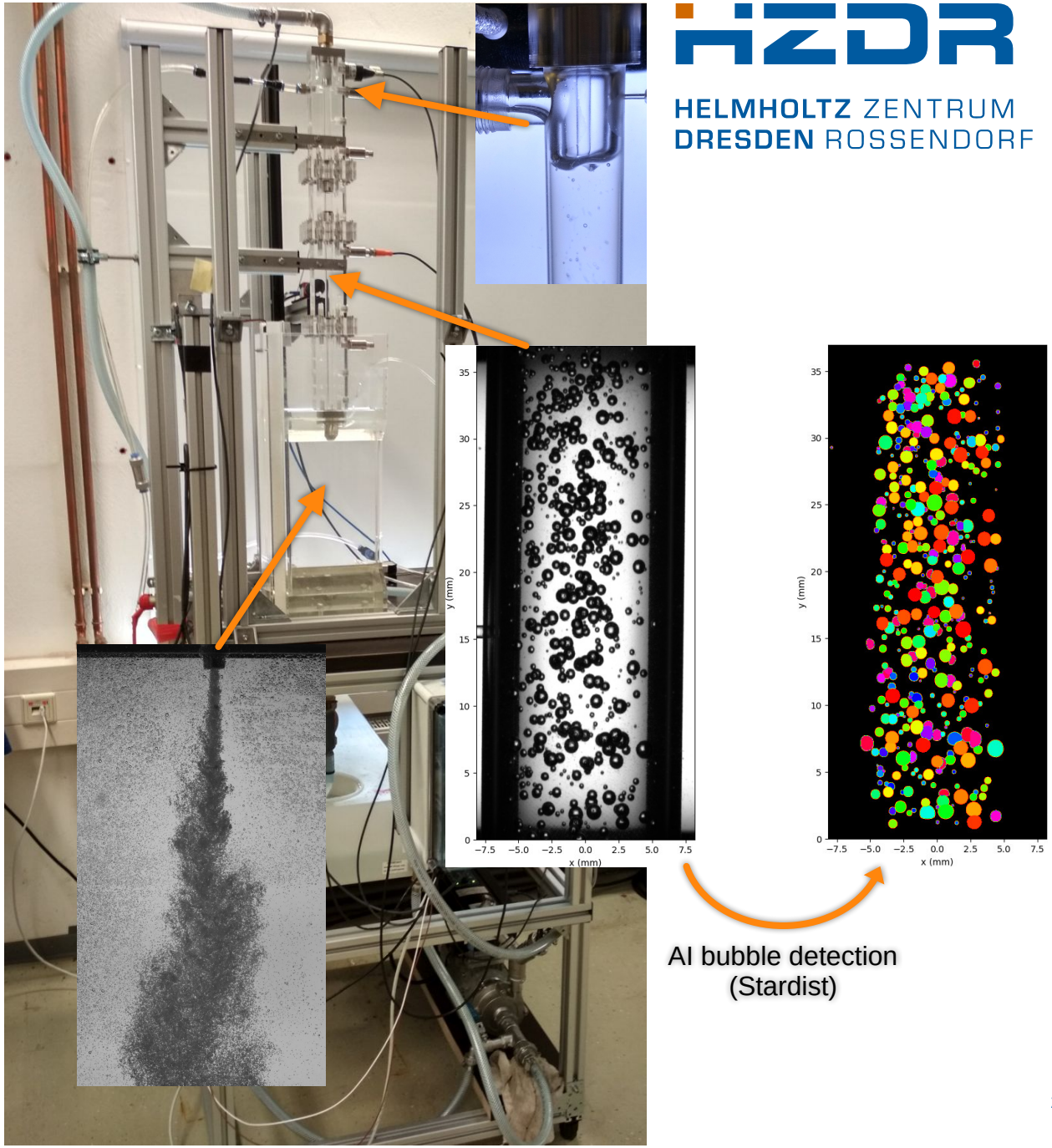
Who am I?

Till Zürner (FWDT)

- Postdoc in experimental fluid mechanics
- Research in multiphase flows (bubbles & particles)
- RSE interests:
  - Data analysis
  - Experiment control & measurement

Current goal:

Migrate measurement program from LabVIEW to Python



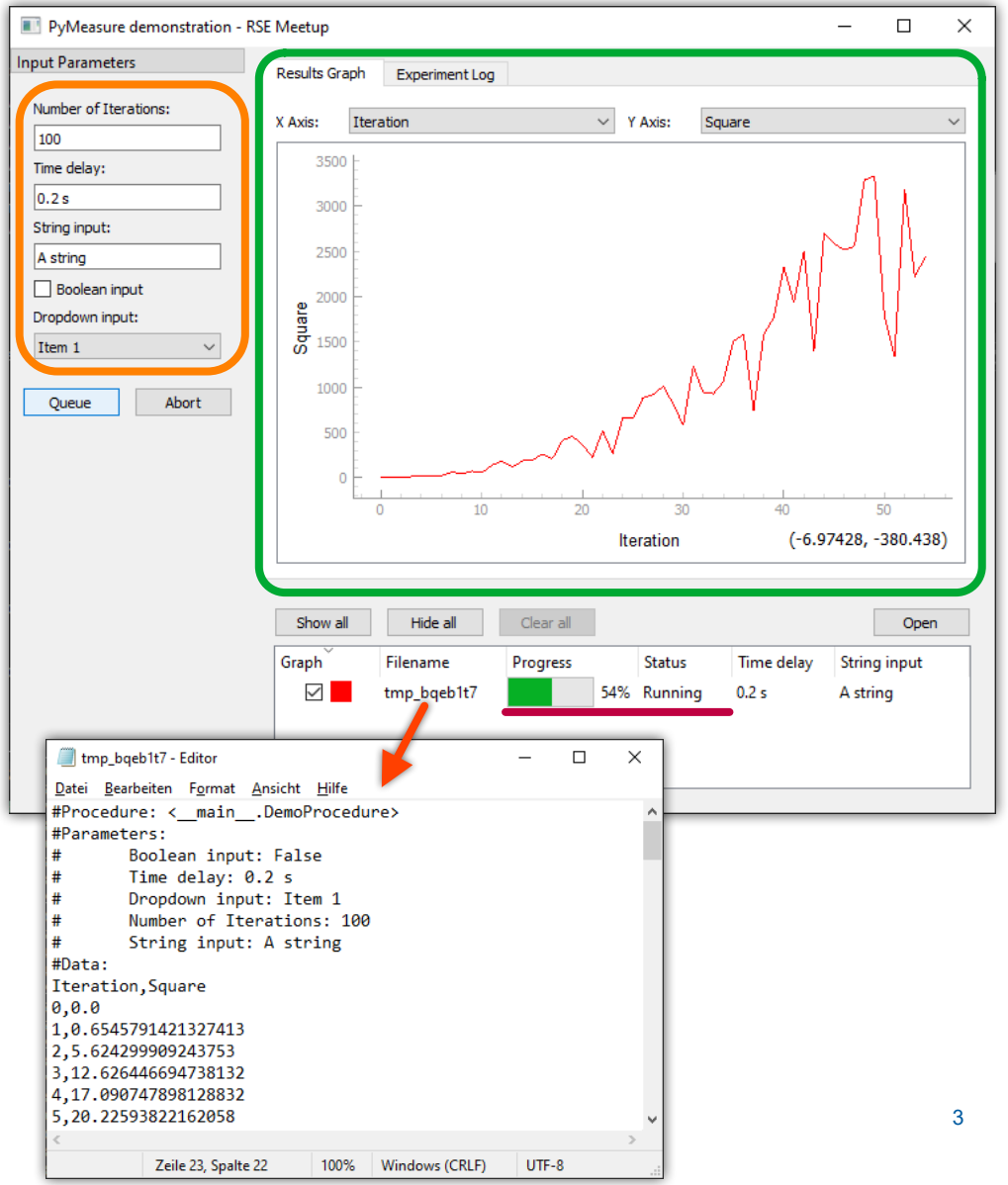
The PyMeasure framework

```python
class DemoProcedure(Procedure):
    # input data
    iterations = IntegerParameter('Number of Iterations', default=100)
    delay = FloatParameter('Time delay', units='s', default=0.2,
                           minimum=0, maximum=10)
    string = Parameter('String input', default='A string')
    boolean = BooleanParameter('Boolean input', default=False)
    dropdown = ListParameter('Dropdown input', default='Item 1',
                             choices=['Item 1', 'Item 2', 'Item 3'])

    # output data
    DATA_COLUMNS = ['Iteration', 'Square']

    def startup(self):
        # prepare experment (connect to devices etc.)
        pass

    def execute(self):
        # define the measurement sequence
        for i in range(self.iterations):

            noise = 0.5 + random.random()
            self.emit('results', {'Iteration': i,
                                  'Square': i**2 * noise})
            self.emit('progress', 100 * i / self.iterations)  # in %

            sleep(self.delay)

            if self.should_stop():
                break

    def shotdown(self):
        # safely disconnect from devices etc.
        pass
```

PyMeasure is based on Procedures defined by

- **Input parameters**
- **Output quantities**
- **Measurement sequence**
  - `startup()`
  - `execute()`
  - `Shutdown()`

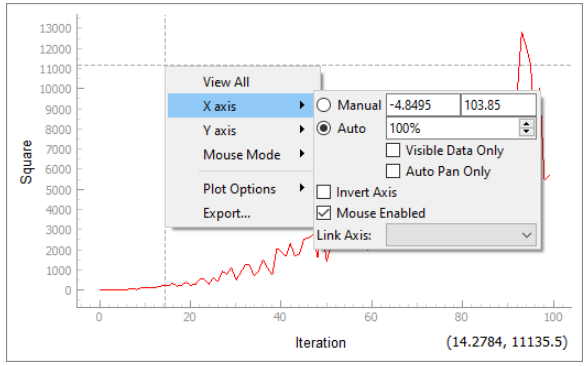
Pre-made graphical interface and writing results to file (csv)



Pros & Cons

Pros

- You can write Procedures however you like
- Pre-made GUI & data recording for quick and simple measurements
- Live display of data with interactive diagram (log-scale, FFT, ...)
- Input GUI easily extendable (Qt-based)
- Driver-based device communication with a sizeable community-created list of drivers



Instruments by manufacturer:

- Active Technologies
  - Active Technologies AWG-401x 1.2GS/s Arbitrary Waveform Generator
- Advantest
  - Advantest R3767CG Vector Network Analyzer
  - Advantest R6245/R6246 DC Voltage/Current Sources/Monitors
- Agilent
  - Agilent 8257D Signal Generator
  - Agilent 8722ES Vector Network Analyzer
  - HP/Agilent/Keysight 34450A Digital Multimeter
  - Agilent 4155/4156 Semiconductor Parameter Analyzer
  - Agilent 33220A Arbitrary Waveform Generator
  - Agilent 33500 Function/Arbitrary Waveform Generator Family
  - Agilent 33521A Function/Arbitrary Waveform Generator
  - Agilent B1500 Semiconductor Parameter Analyzer
- AJA International
  - AJA DCXS-750 or 1500 DC magnetron sputtering power supply
- Ametek
  - Ametek 7270 DSP Lockin Amplifier

Cons

- You *have to* implement Procedures yourself
- Made for one sequential Procedure (no help with parallel or asynchronous processes)
- Heavily relies on PyVISA (reliant on NI license, though serial adapter are available)
- Still in development (latest stable release 0.13.1)

Conclusion

Great for simple quick implementations.

More complicated projects:
Probably easier to write a program yourself than to fit into the PyMeasure mould.

Questions

HELMHOLTZ ZENTRUM
DRESDEN ROSSENDORF

What other experiences and recommendations do you have for experiment control/measurement (in Python)?

Can we contribute code written at HZDR to the development on GitHub?

Appendix: Demo source code

HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

```python
# -*- coding: utf-8 -*-
""" PyMeasure demonstration script for RSE Meetup 2024-03-28

By Till Zürner (t.zuerner@hzdr.de)

using PyMeasure:
https://pymeasure.readthedocs.io/en/stable/index.html

Installation with conda:
------------------------
Create environment `pymeasure-demo`

```
conda create -n pymeasure-demo -c conda-forge pymeasure=0.13.1 pyqt
```

Instead of `pyqt` (PyQt5), you can also install PyQt6, PySide2 or
PySide6 for the graphical display.
"""

import sys
from time import sleep
import tempfile
import random

from pymeasure.display.Qt import QtWidgets, QtCore
from pymeasure.display.windows import ManagedWindow
from pymeasure.experiment import Procedure, Results
from pymeasure.experiment import (
    IntegerParameter,
    FloatParameter,
    Parameter,
    BooleanParameter,
    ListParameter
)
```

```python
class DemoProcedure(Procedure):
    # input data
    iterations = IntegerParameter('Number of Iterations',
                                  default=100)
    delay = FloatParameter('Time delay', units='s', default=0.2,
                           minimum=0, maximum=10)
    string = Parameter('String input', default='A string')
    boolean = BooleanParameter('Boolean input', default=False)
    dropdown = ListParameter('Dropdown input', default='Item 1',
                             choices=['Item 1', 'Item 2', 'Item 3'])

    # output data
    DATA_COLUMNS = ['Iteration', 'Square']

    def startup(self):
        # prepare experment (connect to devices etc.)
        pass

    def execute(self):
        # define the measurement sequence
        for i in range(self.iterations):

            noise = 0.5 + random.random()
            self.emit('results', {'Iteration': i,
                                  'Square': i**2 * noise})
            self.emit('progress', 100 * i / self.iterations)  # in %

            sleep(self.delay)

            if self.should_stop():
                break

    def shotdown(self):
        # safely disconnect from devices etc.
        pass
```

```python
class MainWindow(ManagedWindow):
    def __init__(self):
        super().__init__(
            procedure_class=DemoProcedure,
            inputs=['iterations', 'delay', 'string',
                    'boolean', 'dropdown'],
            displays=['delay', 'string'],
            x_axis='Iteration',
            y_axis='Square'
        )
        self.setWindowTitle('PyMeasure demonstration - RSE Meetup')

    def queue(self):
        filename = tempfile.mktemp()

        procedure = self.make_procedure()
        results = Results(procedure, filename)

        self.manager.queue(self.new_experiment(results))


if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)

    # current version pymeasure 0.13.1 has issues with German locale
    # see https://github.com/pymeasure/pymeasure/issues/602
    QtCore.QLocale.setDefault(
        QtCore.QLocale(QtCore.QLocale.English,
                       QtCore.QLocale.UnitedKingdom))

    window = MainWindow()
    window.show()
    sys.exit(app.exec())
```

You may have to activate Python's UTF-8 mode for consistent encoding of the results file.

Appendix: pyCon² – My program example