Starting using PyMoDAQ: 1 year after

Aurore Finco

Laboratoire Charles Coulomb
Team Solid-State Quantum Technologies (S2QT)

CNRS and Université de Montpellier, Montpellier, France

Pymodays, October 16th 2023

slides available at https://magimag.eu

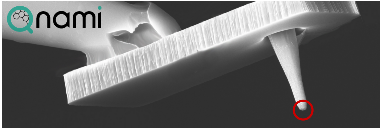
Outline

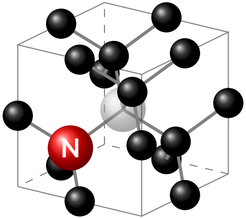
1. *The experiment:* Scanning NV center magnetometry
2. *The past:* Qudi
3. *The present:* PyMoDAQ, almost ready for routine scanning experiments
   - The microwave source as actuator
   - Controlling a stage for the microwave antenna
   - The NI acquisition card
   - Bringing them together as a 1D detector
4. *The future:* Advanced measurements

Outline

1. *The experiment:* Scanning NV center magnetometry
2. *The past:* Qudi
3. *The present:* PyMoDAQ, almost ready for routine scanning experiments
    - The microwave source as actuator
    - Controlling a stage for the microwave antenna
    - The NI acquisition card
    - Bringing them together as a 1D detector
4. *The future:* Advanced measurements

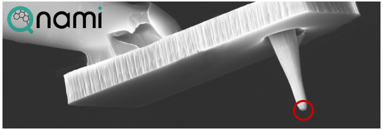
Probing magnetism at the nanoscale

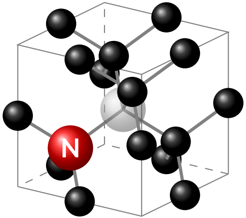




Nitrogen-Vacancy defect

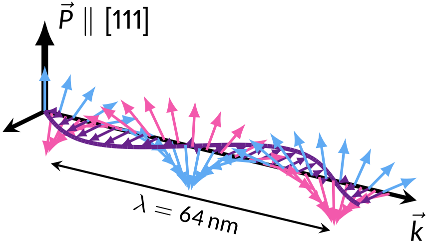
Probing magnetism at the nanoscale

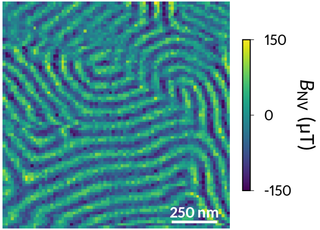




Nitrogen-Vacancy defect

Antiferromagnetic cycloid in $BiFeO_3$

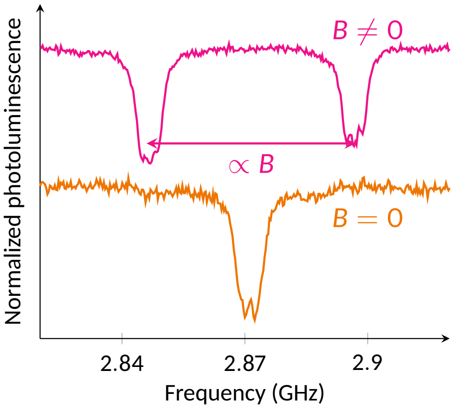




Principle of the magnetic field measurement

Measured with a
single photon counter
usually a few hundreds
of kcounts/s

Optically detected magnetic resonance



Applied with an antenna
and a microwave source
synchronized with the counting

Integration into an atomic force microscope

Attocube
piezo stack
to control the
tip position
↓
Controlled with the
NI card PCIe 6323

Attocube
piezo stack
to control the
sample position
↓
Controlled with the
NI card PCIe 6323

Performing the experiment

Steps:

1. Adjust the position of the tip to have the NV at the laser focus

   Actuators: Piezo scanners
   Detector: Photon counter (APD + NI card)

2. Move the sample to perform a scan

   Actuators: Piezo scanners

3. At every pixel in the scan, record a spectrum and extract the magnetic field value

   Actuator: Microwave source
   Detector: Photon counter

Performing the experiment

Steps:

1. Adjust the position of the tip to have the NV at the laser focus

   Actuators: Piezo scanners
   Detector: Photon counter (APD + NI card)

2. Move the sample to perform a scan

   Actuators: Piezo scanners

3. At every pixel in the scan, record a spectrum and extract the magnetic field value

   Actuator: Microwave source
   Detector: Photon counter

Performing the experiment

Steps:

1. Adjust the position of the tip to have the NV at the laser focus

   Actuators: Piezo scanners
   Detector: Photon counter (APD + NI card)

2. Move the sample to perform a scan

   Actuators: Piezo scanners

3. At every pixel in the scan, record a spectrum and extract the magnetic field value

   Actuator: Microwave source
   Detector: Photon counter

Outline

1. *The experiment:* Scanning NV center magnetometry
2. *The past:* Qudi
3. *The present:* PyMoDAQ, almost ready for routine scanning experiments
    - The microwave source as actuator
    - Controlling a stage for the microwave antenna
    - The NI acquisition card
    - Bringing them together as a 1D detector
4. *The future:* Advanced measurements

Our previous system: Qudi

"Qudi is a suite of tools for operating multi-instrument and multi-computer laboratory experiments. Originally built around a confocal fluorescence microscope experiments, it has grown to be a generally applicable framework for controlling experiments."

J. M. Binder *et al*. *SoftwareX* 6 (2017), 85

Developed at IQO in Ulm, Germany

Our previous system: Qudi

"Qudi is a suite of tools for operating multi-instrument and multi-computer laboratory experiments. Originally built around a confocal fluorescence microscope experiments, it has grown to be a generally applicable framework for controlling experiments."

J. M. Binder *et al.* *SoftwareX* 6 (2017), 85

Developed at IQO in Ulm, Germany

- Modules are specific to a measurement (confocal scan, ODMR spectrum, etc.)
- Specific GUI for each module, fully customized
- Optimized for confocal microscopy

- Interface system to describe the hardware, allowing an easy switch between different instruments with identical functionalities
- Data fitting included for NV center experiments

Why change?

Specific modules developed in our team: scanning NV center magnetometry, spectroscopy

➔ A whole specific GUI has to be created each time

➔ Not always easy to re-use pieces of another module

Why change?

Specific modules developed in our team: scanning NV center magnetometry, spectroscopy

➔ A whole specific GUI has to be created each time

➔ Not always easy to re-use pieces of another module

Situation last year:

- Need to create new modules/update the existing ones
- Running with many modules slightly modified from the standard qudi version, getting a little messy, updates complicated
- Qudi in the middle of a big change in the architecture
- Community not very active, main developers not at IQO anymore

Why change?

Specific modules developed in our team: scanning NV center magnetometry, spectroscopy

➔ A whole specific GUI has to be created each time

➔ Not always easy to re-use pieces of another module

Situation last year:

- Need to create new modules/update the existing ones
- Running with many modules slightly modified from the standard qudi version, getting a little messy, updates complicated
- Qudi in the middle of a big change in the architecture
- Community not very active, main developers not at IQO anymore

➔ Pymodaq!

Outline

1. *The experiment:* Scanning NV center magnetometry
2. *The past:* Qudi
3. *The present:* PyMoDAQ, almost ready for routine scanning experiments
    - The microwave source as actuator
    - Controlling a stage for the microwave antenna
    - The NI acquisition card
    - Bringing them together as a 1D detector
4. *The future:* Advanced measurements

The first easy step: the microwave source

`DAQ_move` plugin for SMA and SMB series from Rohde Schwarz
MW frequency as variable parameter, power selection in the settings

The first easy step: the microwave source

`DAQ_move` plugin for SMA and SMB series from Rohde Schwarz
MW frequency as variable parameter, power selection in the settings

- New PyMoDAQ plugin, created from the template
  https://github.com/Montpellier-S2QT/pymodaq_plugins_rohdeschwarz
- Hardware wrapper adapted from qudi
- Hardware wrapper with much more functions than setting/getting the frequency, for use in other modules combining several instruments
- Units? I used MHz because it is more convenient and there was a limit at 10^8 for the input position, but then I get displays in kMHz...

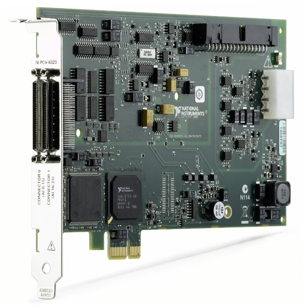
Newport piezo controller

New system for the microwave antenna
➔ Motorized 3D stage
➔ Picomotor 8742 controller

Elias Sfeir

- Writing of a new `DAQ_move` inside `pymodaq_plugins_newport`
- PR coming soon, after a few more tests in the lab

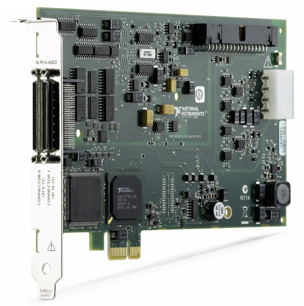
Much harder: the NI acquisition card



PCIe 6323 and 6321 used for:

- Counting the photons (TTL pulses from the detector)
- Controlling the piezo scanners (voltage output sent to an amplificator and then to the stacks)
- Triggering the MW frequency sweep

Much harder: the NI acquisition card

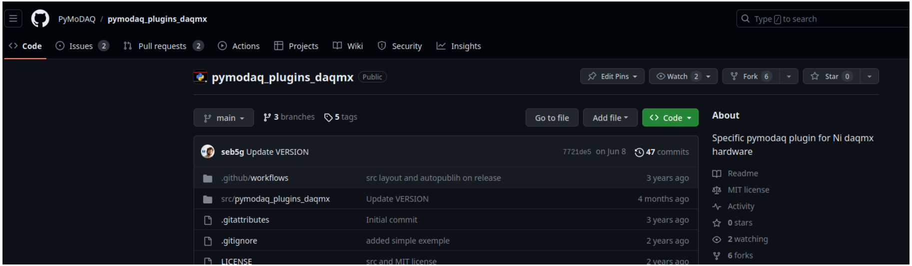


PCIe 6323 and 6321 used for:

- Counting the photons (TTL pulses from the detector)
- Controlling the piezo scanners (voltage output sent to an amplificator and then to the stacks)
- Triggering the MW frequency sweep

- No native control with Python, but an additional layer over C functions has been developed `https://pythonhosted.org/PyDAQmx/index.html`
- Careful with the installed NIDAQmx driver version!
- The documentation from NI is awful!

pymodaq_plugins_daqmx



Existing PyMoDAQ plugin with the hardware wrapper done.

Possibilities of use too broad, no specific `DAQ_move` or `DAQ_viewers`, you need to create your own.

The DAQmx object

Defined in `hardware/national_instruments/daqmx.py`

```
class DAQmx:
    """Wrapper around the PyDAQmx package giving an easy to use object to instantiate channels and tasks"""
    def __init__(self):
        self.devices = []
        self.channels = []
        self._device = None
        self._task = None
        self.update_NIDAQ_devices()
        self.update_NIDAQ_channels()
        self.c_callback = None
        self.callback_data = None
        self.is_scalar = True
        self.write_buffer = np.array([0.])

    @property
    def task(self):
        return self._task
```

Channel types:

- Analog Output
- Analog Input
- Digital Output
- Digital Input
- Counter / Clock

- Use it as your controller
- A single task per `DAQmx()`
- A task can handle only one type of channel → if you need a Clock and an Analog Output, you need 2 tasks and thus 2 `DAQmx()`
- Channel objects are also defined in the file, one for each type, in order to initialize the task.
- “Natural” methods to interact with the task are available in `DAQmx`: `writeAnalog()`, `readAnalog()`, etc.

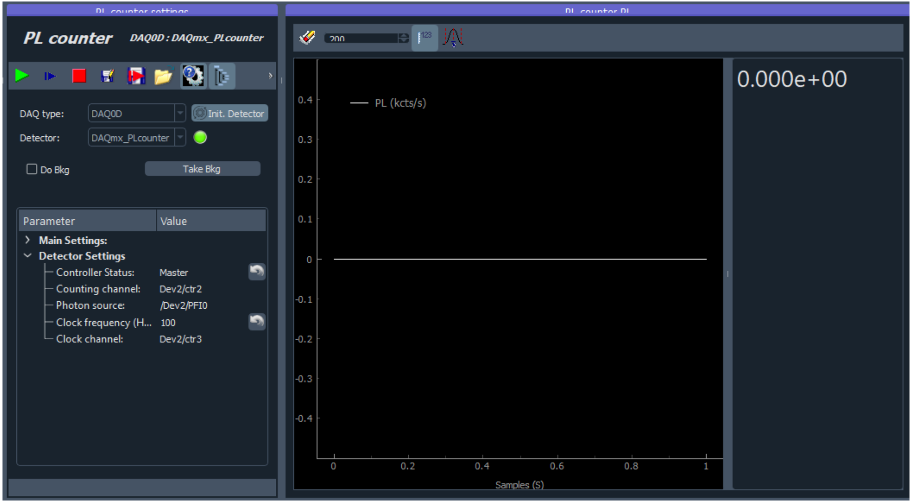
The PLcounter 0D detector

Experiment to perform: Count the number of TTL pulses coming from the photon detector, averaging at a given rate.

Implementation:

- 0D Viewer
- Use a Counter channel to get the number of counts
- Use a ClockCounter channel to handle the timing
- The controller is a `dict` containing 2 `DAQmx()`

The PLcounter plugin



Controlling the piezo scanners

Experiment to perform: Move slowly the scanners at a controlled speed, we need to avoid brutal movement because of the fragile diamond tip.

Controlling the piezo scanners

Experiment to perform: Move slowly the scanners at a controlled speed, we need to avoid brutal movement because of the fragile diamond tip.

First implementation tentative:

- DAQ Move
- Use an Analog Output to send the signal
- Control the speed:
  - Separate the movement in steps if the distance to cover is too large
  - Use a CounterClock channel to trigger each step of the Analog Output channel
- Read the position: from the clock index!
- The controller is a `dict` containing 2 `DAQmx()`
- *Parameters:* Analog output channel, Clock channel, Step size, Step time, Conversion factor distance/voltage, Bounds

Here come the troubles: using the scanning extension

We want to use 2 of these DAQ Move for scanners together with a detector in the Scan Extension.

Issue 1: There are only 4 Counter/clock channels on the device, so using the PL counter, 2 scanners for the tip or the sample, and later a clock to trigger the MW becomes tricky.

Issue 2: The scan extension sends the “move” commands to all the actuators at the same time, it really does not work if you use the same clock for both scanners.

Here come the troubles: using the scanning extension

We want to use 2 of these DAQ Move for scanners together with a detector in the Scan Extension.

Issue 1: There are only 4 Counter/clock channels on the device, so using the PL counter, 2 scanners for the tip or the sample, and later a clock to trigger the MW becomes tricky.

Issue 2: The scan extension sends the “move” commands to all the actuators at the same time, it really does not work if you use the same clock for both scanners.

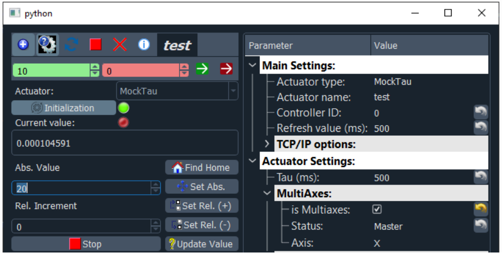
→ **We need to share a Clock channel between the scanners, and to coordinate their movements**

Solution based on a multiaxes controller

PyMoDAQ provides a way to share a controller between actuators: the Master/Slave mechanism for the Multiaxes controller.

3.4.2.1.5.2. Multiaxes controller

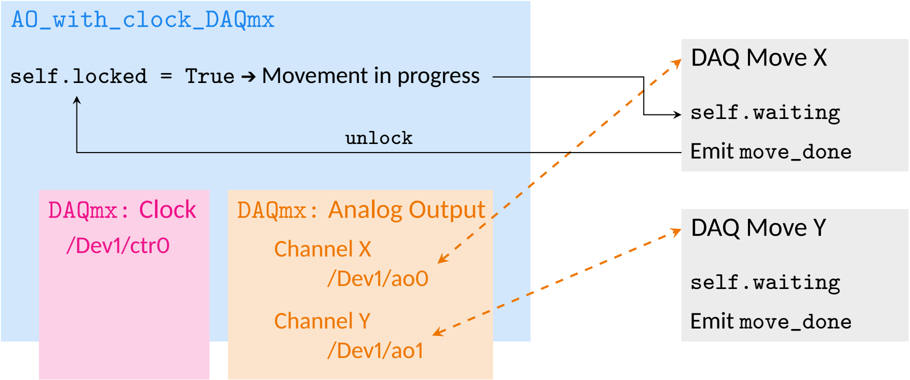
Sometimes one hardware controller can drive multiple actuators (for instance a XY translation stage). In the simplest use case, one should just initialize the instrument plugin and select (in the settings) which *axis* to use, see Fig. 3.13.



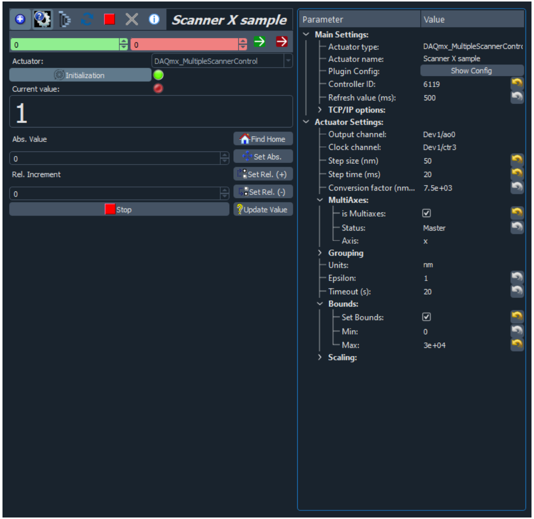
But we cannot use a simple `dict` of `DAQmx()` as before, since we need to perform the movements on each axis sequentially (shared Clock).

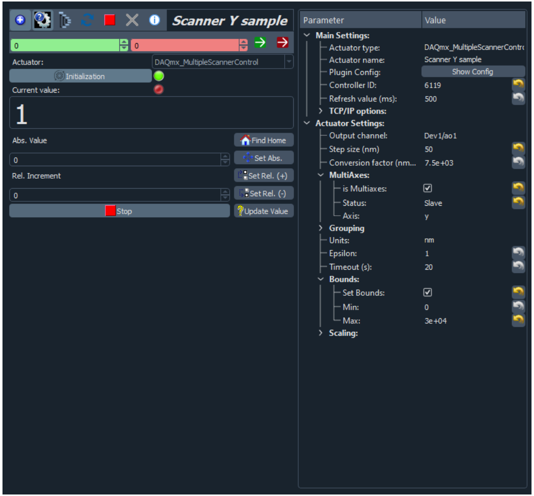
Building a new controller based on DAQmx objects

New object `AO_with_clock_DAQmx` in `daqmx_objects.py`, for use as a multiaxes controller.



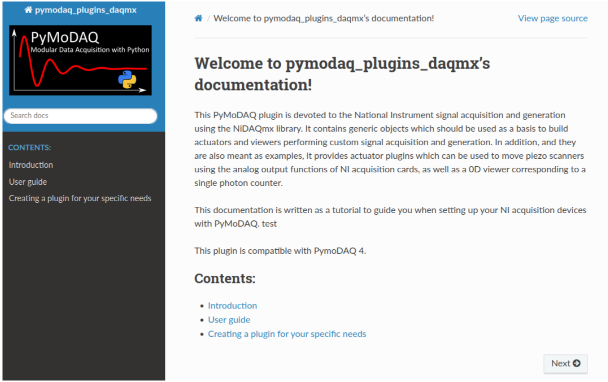
The MultipleScannerControl actuator





Documentation

Generated by Sphinx, deployed with Github pages.
On our repo for now, soon on the official one.



Optical detection of the magnetic resonance (ODMR)

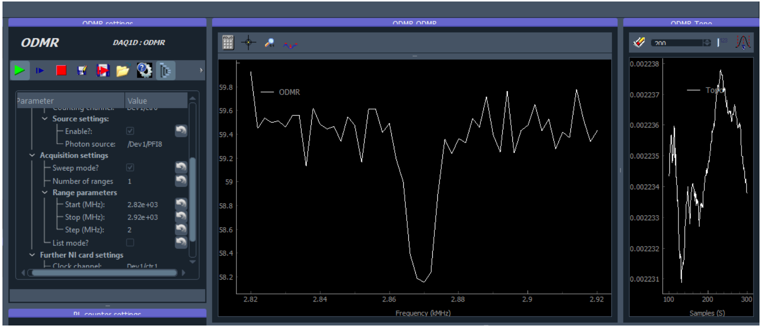
Experiment to perform: Measure the number of photons emitted as a function of the MW frequency

Implementation:

- 1D Viewer
- Use a Counter channel to get the number of counts
- Use a MWSource wrapper to control the MW generator
- Use a ClockCounter channel to trigger the change of the MW frequency
- The controller is an object containing 2 `DAQmx()` and a `MWSource()`

pymodaq_plugins_s2qt_odmr

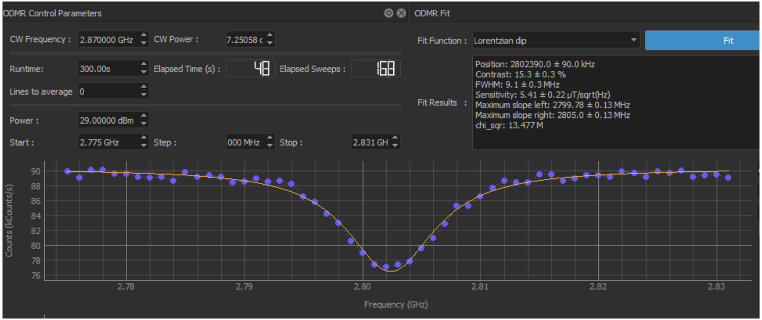
Repository: https://github.com/Montpellier-S2QT/pymodaq_plugins_s2qt_odmr



Additional options very recently added: measure only 2 frequency points and plot the difference (iso-B mode), select several frequency ranges (to track both resonances).

Fitting or not fitting?

Very convenient fitting options in Qudi, to optimize the signal



The analysis has to be done on the averaged signal, so **outside of the instrument plugin**…
Stick to "Do Math with ROI", at least for now.

Outline

1. *The experiment:* Scanning NV center magnetometry
2. *The past:* Qudi
3. *The present:* PyMoDAQ, almost ready for routine scanning experiments
    - The microwave source as actuator
    - Controlling a stage for the microwave antenna
    - The NI acquisition card
    - Bringing them together as a 1D detector
4. *The future:* Advanced measurements

Next steps

- Issues with the scanning extension
  - Avoid going back to init position when stopped
  - Visualisation of ND scans not working completely yet

- Set up the PID to the adjust the position of the NV at the laser focus

- Pulsed experiments
  - Generate sequences of laser and microwave pulses to manipulate the NV center
  - Record the PL trace during such a sequence
  - Extract the relevant information to display it during a scan

Work in progress...