HUN-REN Hungarian Research Network
Research. Innovation. Impact.

Adapting Data Acquisition Strategies from Large-Scale Facilities – Or How to Dance in the Lab?

András Wacha

HUN REN | ttk

MTA Centre of Excellence

Excellent Research Infrastructure Hungary

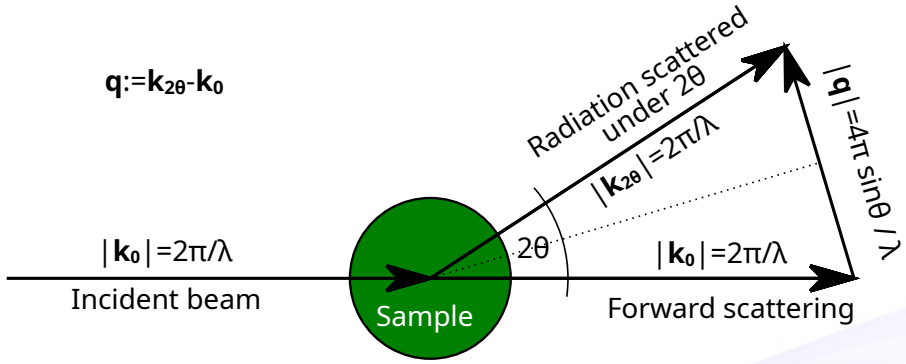
CREDO
CREATIVE RESEARCH EQUIPMENT FOR DIFFRACTION

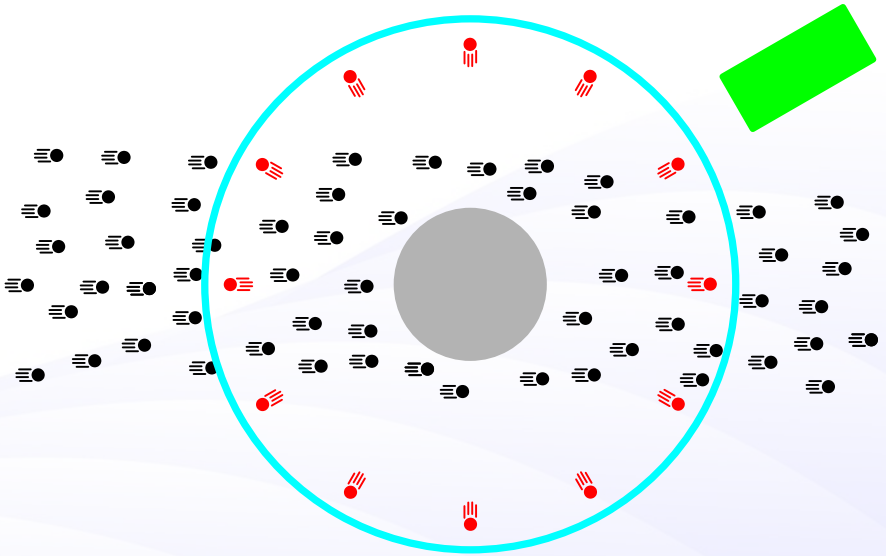
Motivation: Why do we need data acquisition strategies?

- Computerized measurements: possibilities and liabilities
  - Frequent recording of the measurand
  - Automation (sample, geometry or “environment” changes)
  - Easier storage, data is ready for reduction, interpretation…
- The requirement for reproducibility and correctness
  - Recording as many aspects of the experiment as possible (or necessary)
  - Eliminating uncontrolled parameters
  - Repeating experiments under the same conditions
- Data storage considerations
  - Safe (against data loss or corruption)
  - Secure (access control, privacy)
  - Short term / long term
  - Raw or processed data
  - Open data, FAIR principles
- “Do not reinvent the wheel!”
  - Build on already developed and proven utilities and techniques
  - Each experimental field has its own “best practices” and “publication guidelines”

Case study: small-angle X-ray scattering

Small-angle X-ray scattering – from the experimentalist's point of view

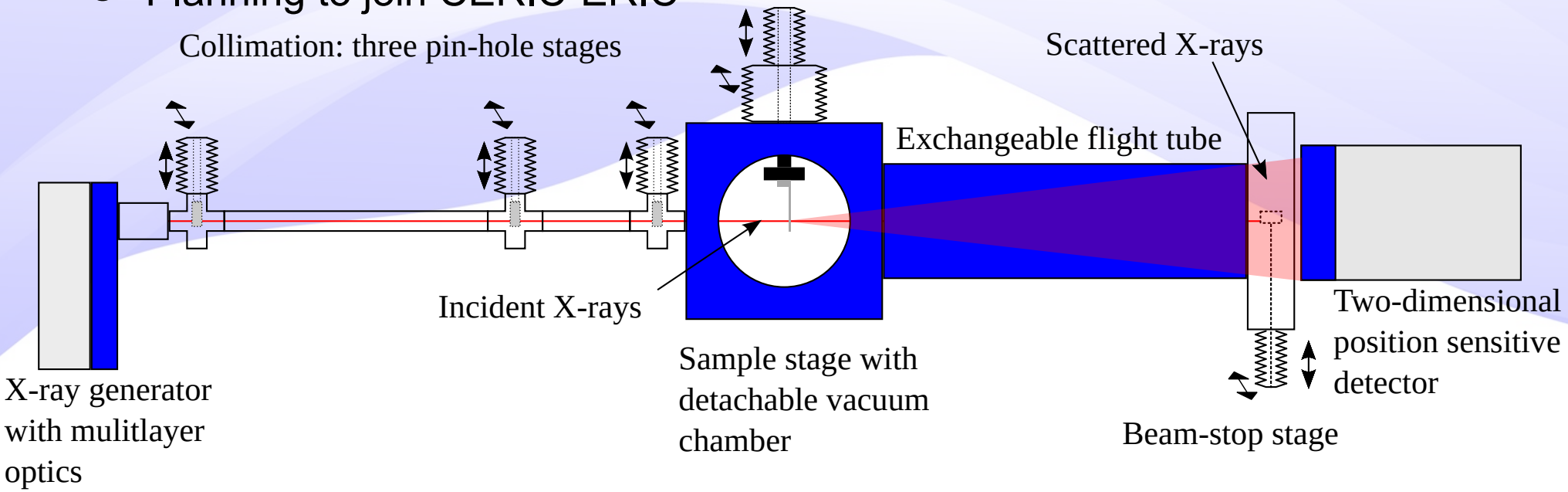




- Fixed wavelength, moderate monochromaticity ($\Delta\lambda/\lambda < 1\%$), highly parallel
- Physics: elastic scattering
- Primary data: "count rate" vs. "scattering angle"
- Instrument-independent units
  - Differential scattering cross-section ("$d\sigma/d\Omega$", $cm^2/cm^3 \times sr^{-1}$)
  - Momentum transfer ("q", nm^{-1})
- Typically two-dimensional detector, scattering pattern
- Stationary geometry, data acquisition for several minutes/hours (lab), few msecs (synchrotron)

The CREDO instrument (https://credo.ttk.hu)

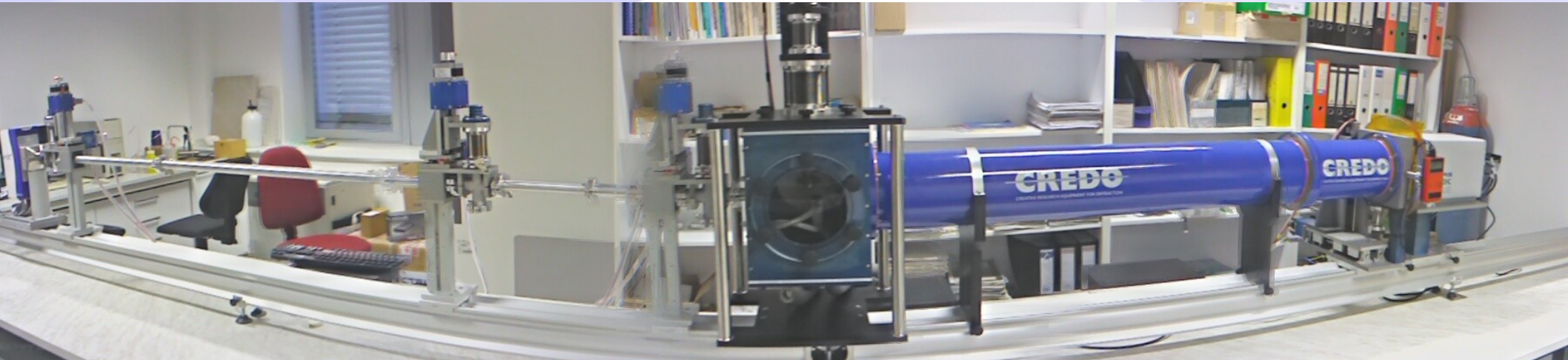
- In-house built laboratory SAXS camera (→ ask me for a tour in the lab)
  - Based on synchrotron experience
- Optimized three-pinhole collimation system
- Wide angular range from 0.02 to 30 nm^{-1} ($q=4\pi\sin(\theta)/\lambda$)
- Extensively motorized (XY for pin-holes, sample and beam stop)
- Unique in the region
  - Only SAXS instrument in Hungary
  - Serves all researchers in-house (including other groups)
  - Access for guest researchers via a beamtime proposal system
  - Planning to join CERIC-ERIC



The CREDO hardware

CREDO
CREATIVE RESEARCH EQUIPMENT FOR DIFFRACTION

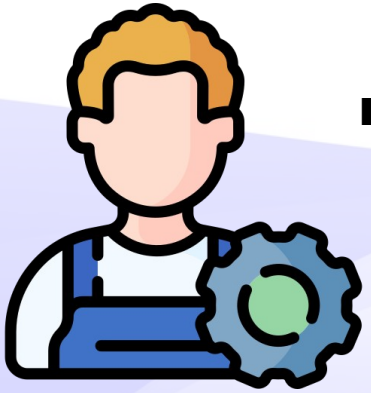
- Core components
  - Source: GeniX^{3D} Cu ULD (Xenocs SA, Sassenage, France)
  - Detector: Pilatus-300k (Dectris Ltd, Baden, Switzerland) → 0.3 Mpixel
  - Stepper motor controllers (Trinamic GmbH, now Analog Devices)
  - Vacuum gauge: TPG-201 (Pfeiffer Vacuum)
  - Thermometer (SE521)
- In situ measurements
  - Thermostating water circulator (Haake Phoenix P25C)
  - Peristaltic pump (LeadFluid BT100s)
  - Sample illumination (Schott KL2500)
  - Magnetic stirrer (IKA RET control-visc)
- Other
  - Uninterruptible Power Supply (Tecnoware Evo DSP Plus)
  - Detector gas supply sensor (In-house built, Raspberry-Pi-based)
  - …
- RS-232, RS-485, Modbus-TCP, USB HID, TMCL, and other domain-specific solutions



Designing a data acquisition system

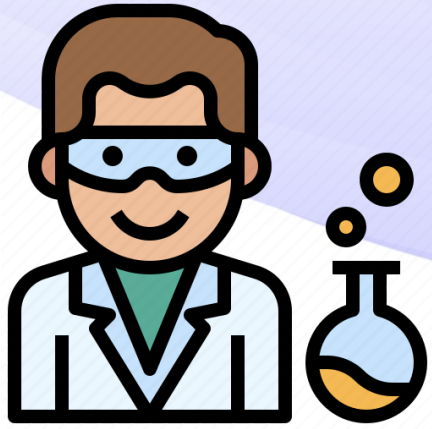
Requirements towards the control and data acquisition system

- For the developer: easy to extend
  - flexible enough to accommodate heterogeneous equipment
  - Specification always change
  - Modular: devices can be inserted and removed
  - Developing drivers for new devices should be easy
  - Build on already available solutions
  - Free software vs. paid solutions?



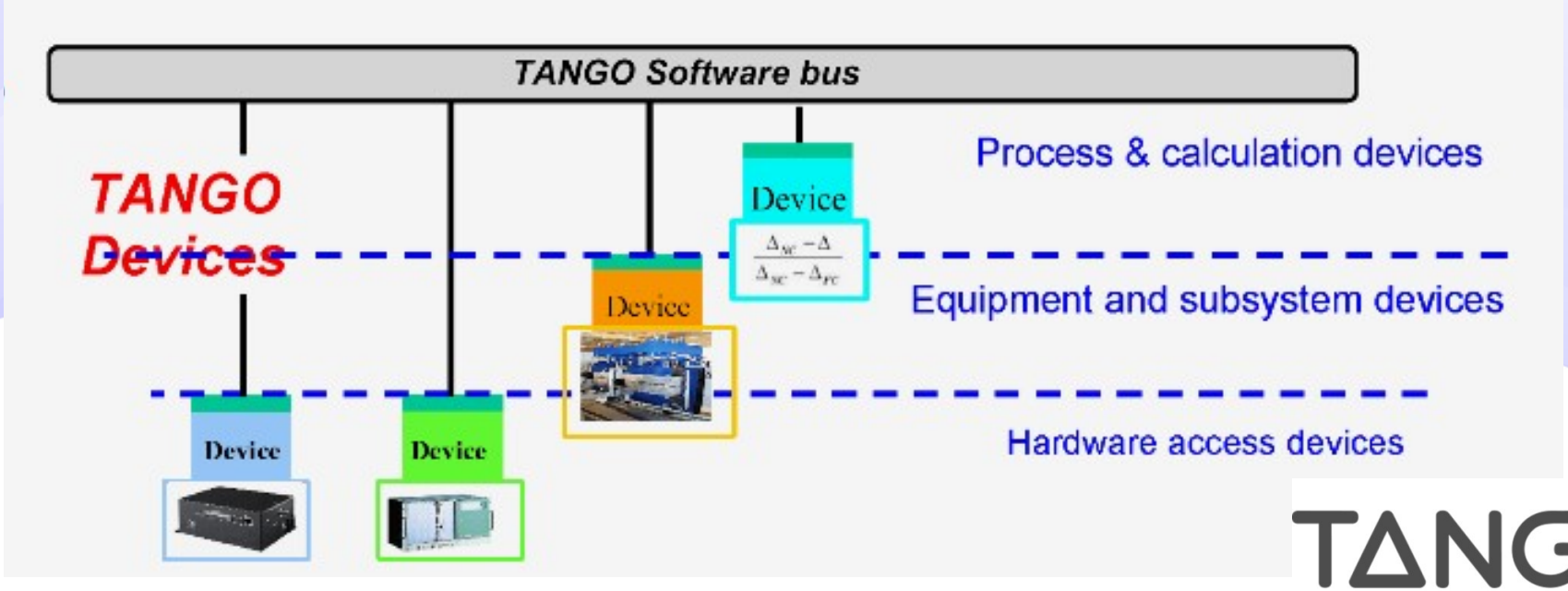
- For the operator / beamline scientist: highly automated, ease of use
  - Perform required background and calibrant measurements
  - Measure multiple samples in a sequence
  - Standardized procedures (data acquisition, reduction)

- For the casual user / guest researcher: reliable, correct
  - Control *in situ* parameters (temperature, shear, illumination, flow rate…)
  - In-process visualization and assessment of the quality of the results
  - User-friendly
  - Standards-compatible data storage
  - High throughput



The TANGO Control System (https://www.tango-controls.org)

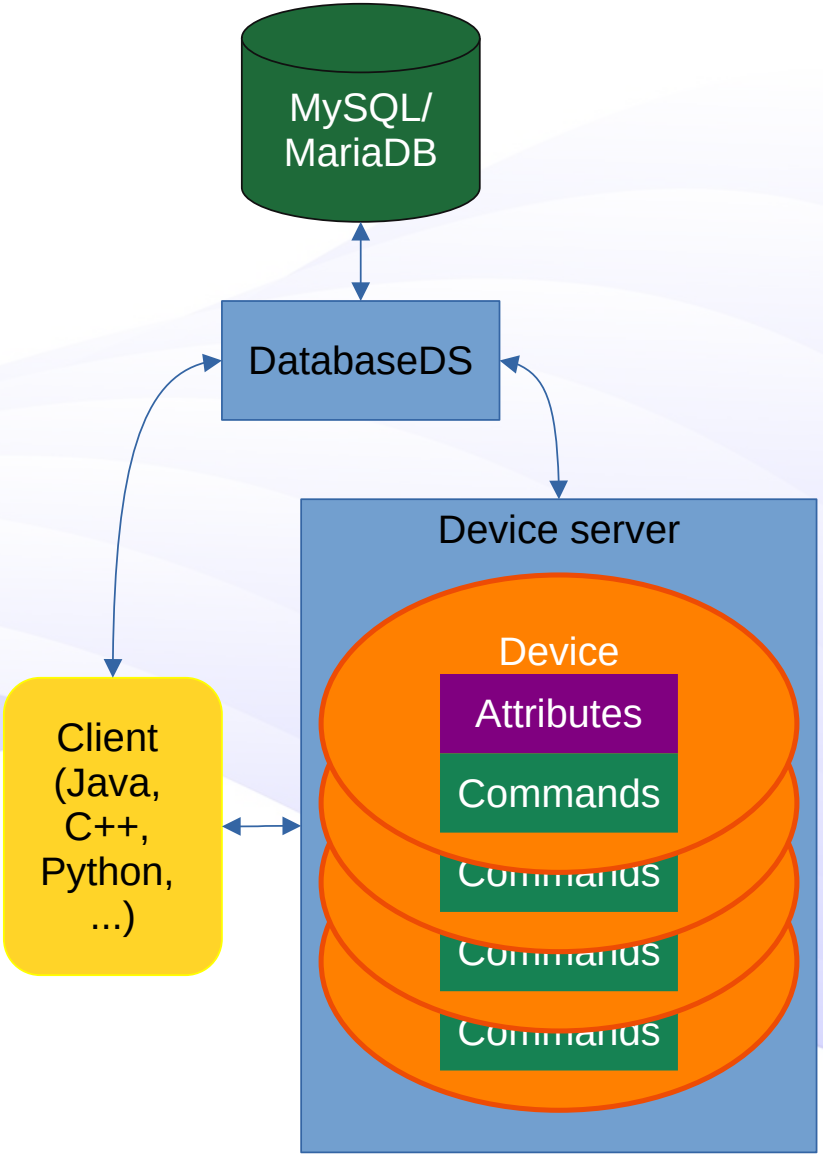
- Developed initially at ESRF (Grenoble), then by the Tango Controls Consortium: ESRF, DESY, ALBA, SOLEIL, ELETTRA, SOLARIS, ELI BEAMS, MAX-IV, FRM-II, ...
- A “software bus”: a standardized way of communication between various parts of the instrument
- Unified interface: hiding how the equipments are connected (USB, TCP/IP, RS-232…) and where (individual computers on the network)
- Distributed: devices can be attached anywhere, even relocated
- Object-oriented approach (CORBA: Common Object Request Broker Architecture™)



TANGO
Connecting things together

The Tango Architecture

- Basic unit: device
  - Device server: a program managing various devices of the same type (“device class”)
  - Not just hardware devices:
    - Algorithms (data reduction, file writer, loggers, software subsystems)
    - Meta-devices (e.g. stepper motors belonging to the same controller)
    - Interfaces (RS232, Modbus, TCP socket…): other device servers may rely on these
- Server code: C++, Java, Python
- Client code: C++, Java, Python, Matlab, Igor Pro, Labview…



The TANGO device: living inside the device server

- Name: `<domain>/<family>/<member>`
  - e.g., `<beamline>/<equipment type>/<equipment name>`
    - `credo/detector/pilatus300k`
  - e.g., `<interface type>/<computer>/<equipment name>`
    - `serial/credo-pi/thermostat`, `motor/beamstopcontroller/bsx`
- Object-oriented approach
  - Class: the type of equipment it controls (e.g. a Haake Phoenix P25C thermostat via RS-232)
  - Attributes: state variables
    - Read-only (current temperature, current motor speed)
    - Read/write (temperature set point, target motor position)
    - 0-1-2 dimensional
  - Commands (“methods” in the C++ terminology): operations that can be performed
    - e.g. open the beam shutter, start water circulation…
  - Properties: parameters and settings not expected to change
    - Hardware address of the device
  - Pipes: transferring a large amount of data

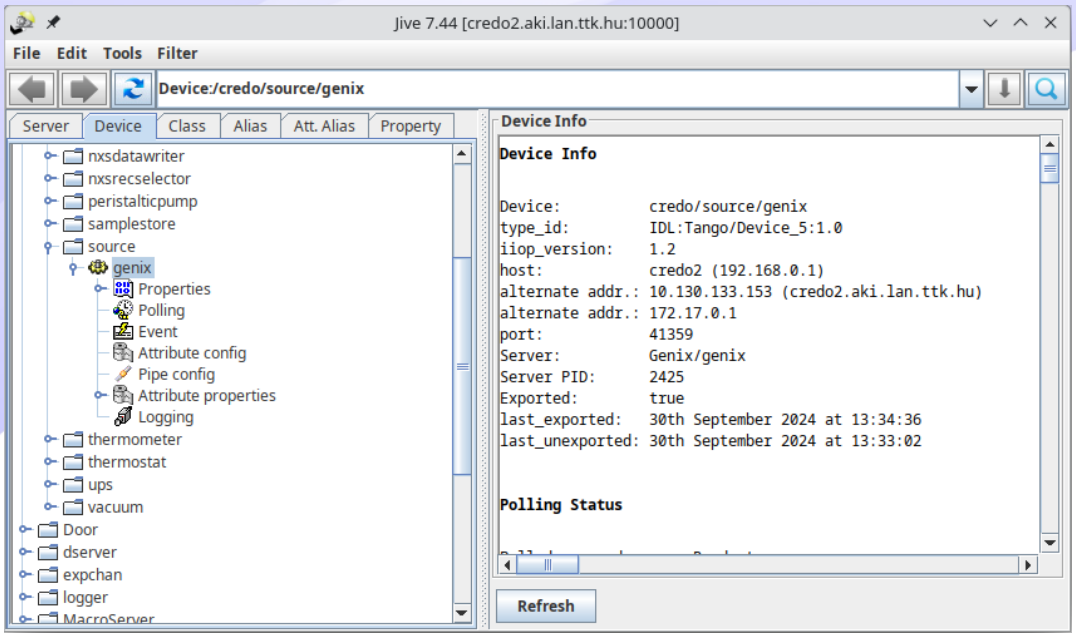
Anatomy of a TANGO device: the X-ray source of CREDO

- `credo/source/genix` (the X-ray source)
  - Properties:
    - *modbus*: `modbus/credo2/genix` (another TANGO device!)
  - Attributes:
    - *State* (common to all TANGO devices: one of ON, OFF, CLOSE, OPEN, INSERT, EXTRACT, MOVING, STANDBY, FAULT, INIT, RUNNING ALARM, DISABLE or UNKNOWN)
    - *Status* (common to all TANGO devices: a textual representation of the current state)
    - *current*: the actual current in the X-ray tube (mA)
    - *ht*: high tension of the X-ray tube (kV)
    - *vacuum_fault*…
    - …
  - Commands:
    - *StartWarmUp*, *StopWarmUp*, *FullPower*, *GoStandby*, *PowerOff*, *ResetFaults*, …

Special devices: DatabaseDS and Starter

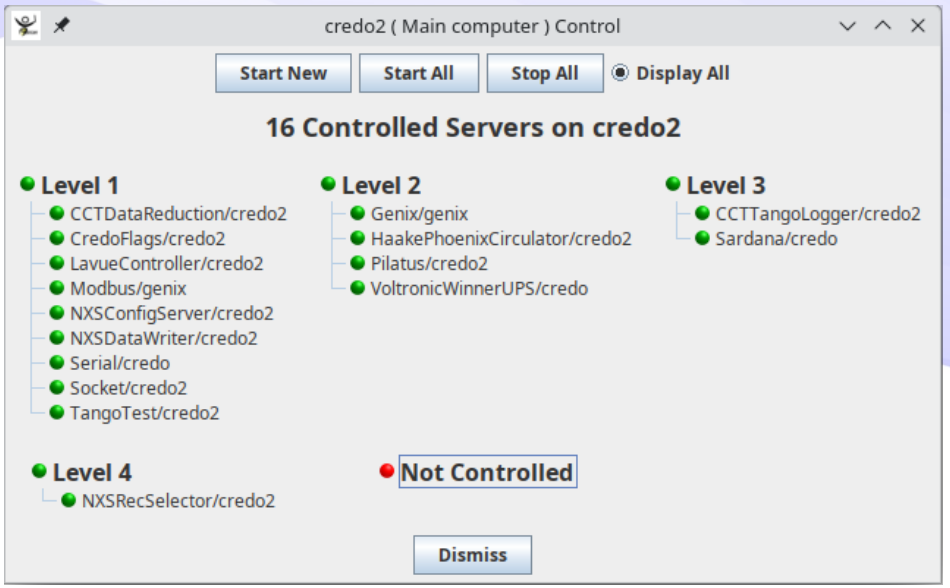
DatabaseDS

- One is required for each Tango system (on a dedicated server host)
- Interface to the SQL database
- Provides name resolution: where can the device `x/y/z` be found?
- Storing information about devices: classes, on which computer they are running, which instances, which device servers, name aliases…
- GUI: Jive



Starter: orchestration

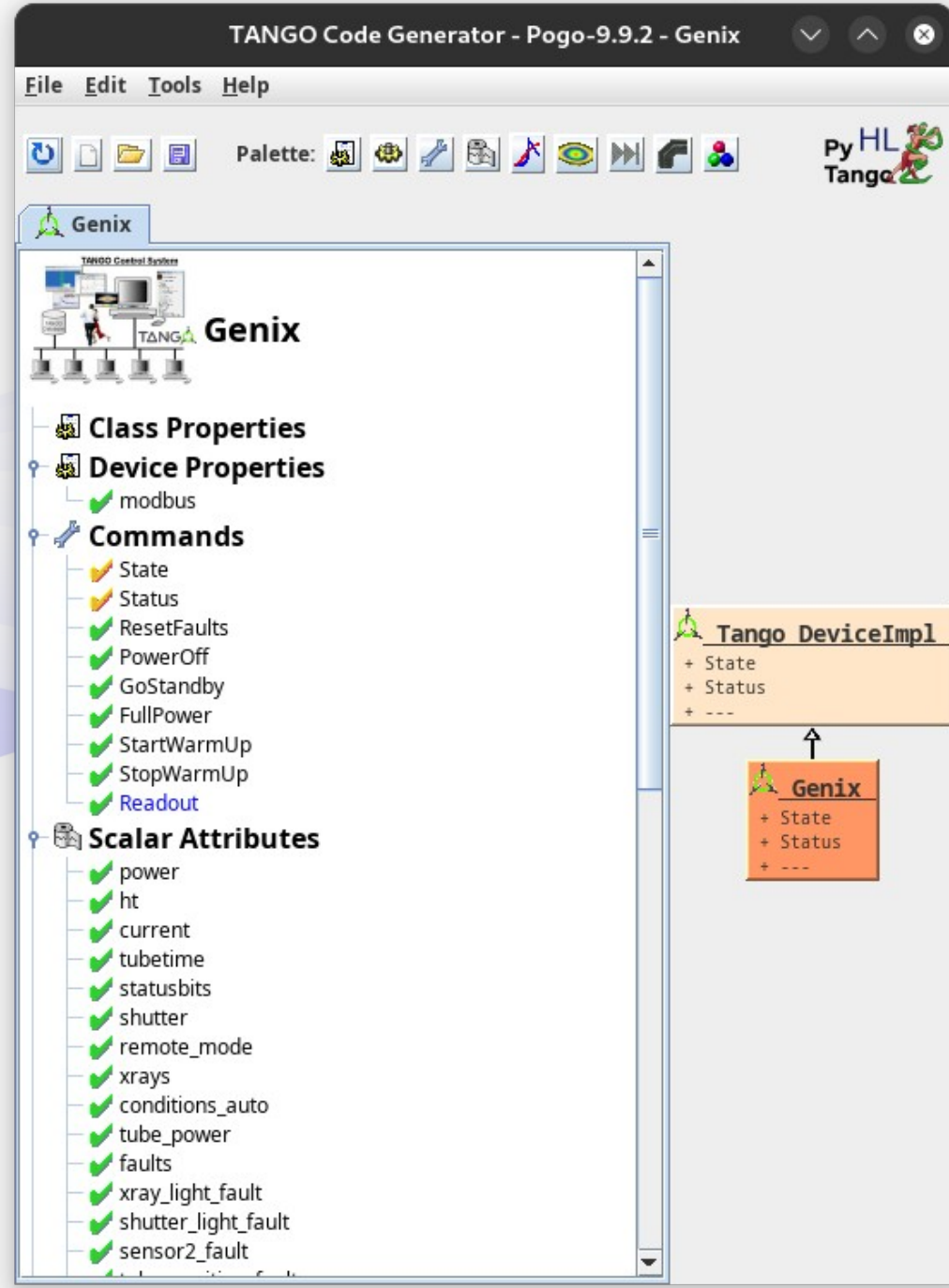
- Not required but *very* helpful
- Typically on each computer in the Tango system where device servers are running
- Started by the OS
- Responsible for starting, stopping, restarting device servers on this computer (as defined in the database)
- GUI: Astor (*cf* ~ Piazzola)



Developing Device Servers – Pogo

- **P**rogram **O**bviously used to **G**enerate **O**bjects
- Graphical user interface to define the devices
- Store the data in an XML file
- Writes skeleton code (C++, Java, Python), which can be fleshed out
- Auto-generation of documentation
- XML file can be uploaded to the Tango Classes Catalogue (https://www.tango-controls.org/developers/dsc/)

```
def read_shutter(self):
    # PROTECTED REGION ID(Genix.shutter_read) ENABLED START #
    """Return the shutter attribute."""
    # ...
    # Actual code to read the variable comes here
    # ...
    return self._shutter
    # PROTECTED REGION END # // Genix.shutter_read

def write_shutter(self, value):
    # PROTECTED REGION ID(Genix.shutter_write) ENABLED START #
    """Set the shutter attribute."""
    # ...
    # Actual code to write the variable comes here
    # ...
    # PROTECTED REGION END #    //  Genix.shutter_write
```



Python Language Bindings

- PyTango (https://pytango.readthedocs.io/en/latest/)
- Object-oriented approach
  - Tango attributes → data members
  - Tango commands → member functions
- Both for servers and clients
- Introspection
  - get_attribute_list()
  - get_command_list()
- Interaction with the Tango database
- ITango: improved, Tango-aware variant of Ipython

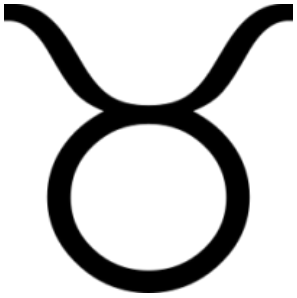

```
In [1]: from tango import DeviceProxy

In [2]: vac = DeviceProxy('credo/vacuum/tpg201')

In [3]: vac.get_attribute_list()
Out[3]: ['pressure', 'version', 'units', 'State', 'Status']

In [4]: vac.pressure
Out[4]: 0.034

In [5]: vac.Status()
Out[5]: '0.034 mbar'

In [6]: genix = DeviceProxy('credo/source/genix')

In [7]: genix.shutter
Out[7]: False

In [8]: genix.shutter = True

In [9]: genix.get_command_list()
Out[9]: ['FullPower', 'GoStandby', 'Init', 'PowerOff', 'Readout',
'ResetFaults', 'StartWarmUp', 'State', 'Status', 'StopWarmUp']

In [10]: genix.PowerOff()
```

Other Uses and Features of TANGO

- Logging facility: messages from the device servers (errors, warnings…)
  - Java GUI: LogViewer
- Polling:
  - automatic, periodic query of attributes (state variables)
  - … or execution of commands
- Event system
  - Client programs can subscribe to event notifications by the device servers
  - React to changes instantaneously
  - Warning and alarm events
    - Lower and upper thresholds for attribute values
- Archival of attribute values
  - either periodically,
  - … or upon a large enough change
- Automatic generation of (very basic) overview GUIs for devices (ATK, Synoptic views with JDraw)
- Access control: restrict user access on devices

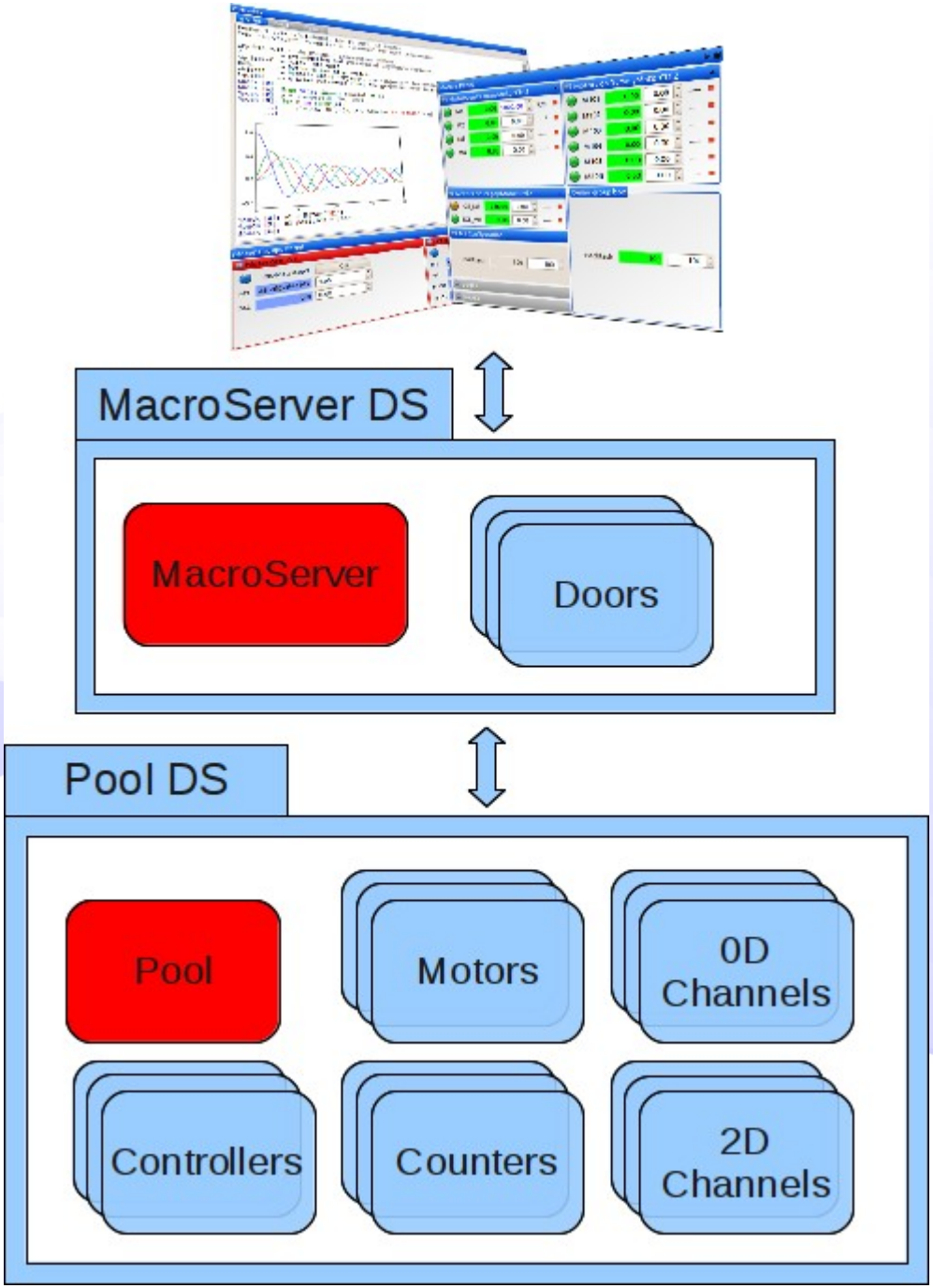
The Sardana Framework (https://www.sardana-controls.org)

- Developed principally at ALBA (Barcelona, Spain)
- Aims of the project
  - A further abstraction layer on Tango: controllers and devices
  - Highly flexible and powerful framework for scan measurements
  - Macro server
  - Interactive command line
  - Data recorders: HDF5, SPEC, NeXus
- Sister project: Taurus (https://taurus-scada.org)
  - Creation of full-featured GUIs (forms, plots, controls etc) for data acquisition
  - “Configure instead of coding”



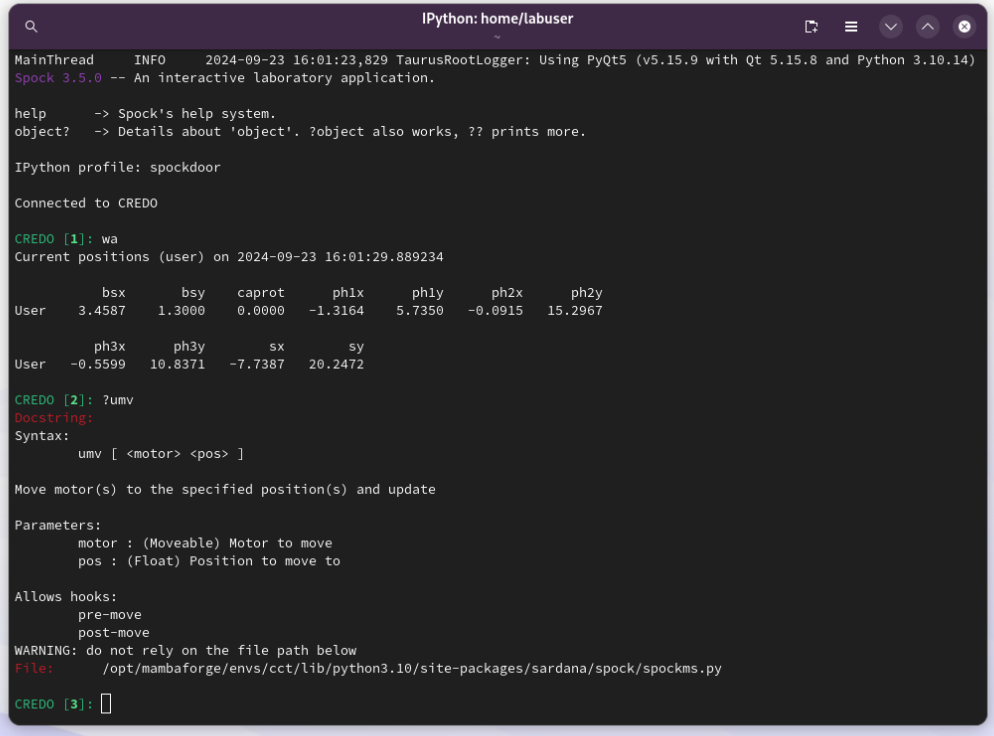
The Sardana Architecture

- Two main device servers, each running several device classes
  - **Pool:** list of physical and virtual devices known to Sardana
    - Elements: controllers and their controlled devices
  - **MacroServer:** list of macros (procedures) that can be used
    - Door: a macro running context. Sardana clients (see next slide) connect to these. Each door can run one macro at a time.
- Communication between pool and macroserver objects, as well as clients: over Tango



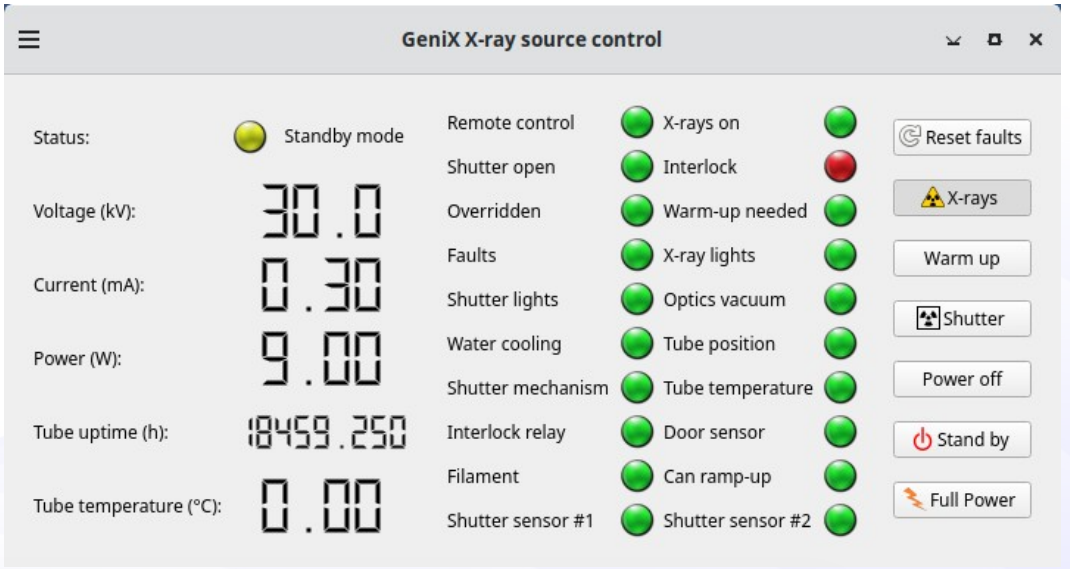
Spock: the Primary Interface to Sardana

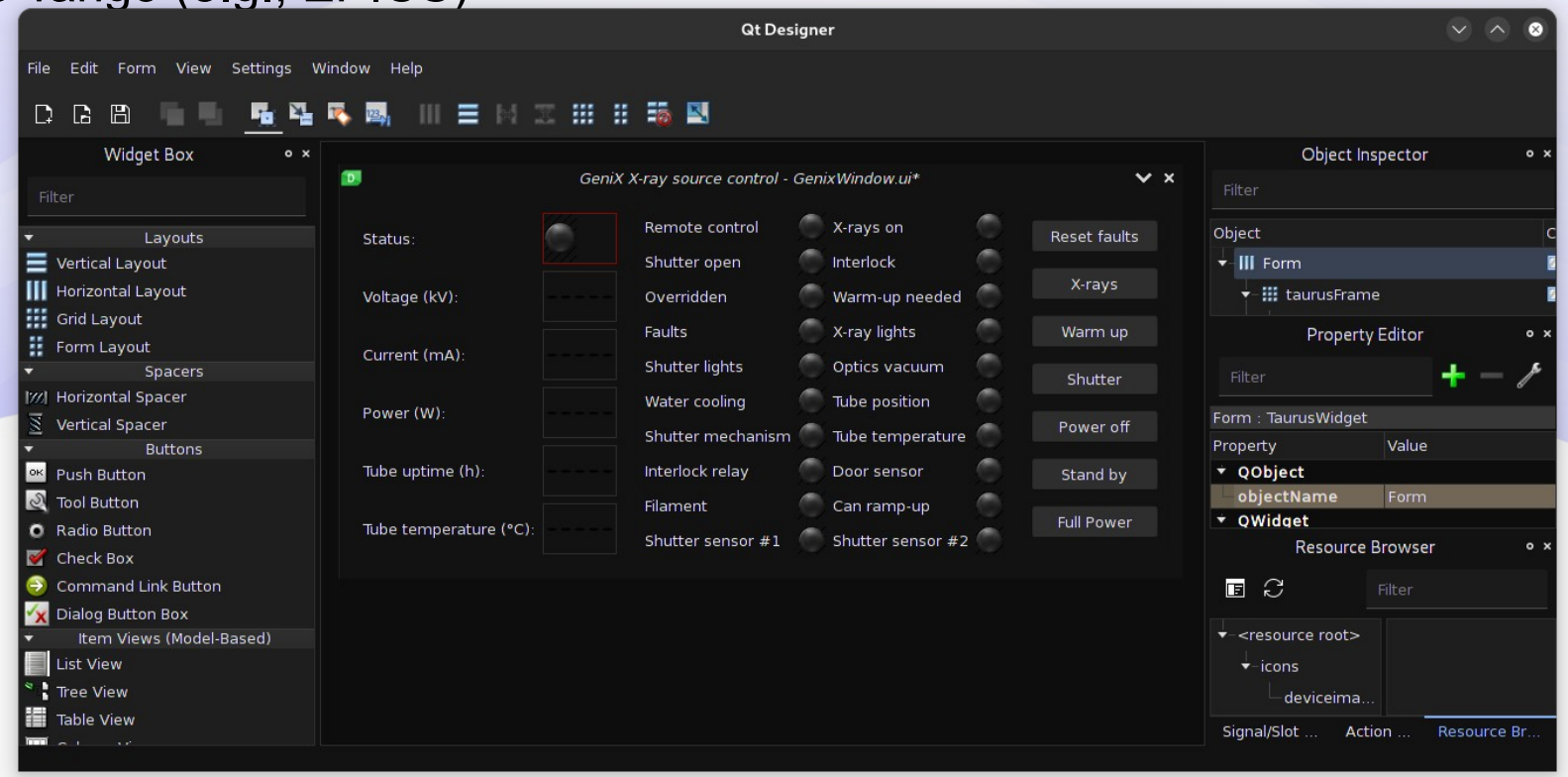
- (Not just) a clone of SPEC
- An extension of IPython
- Macros:
  - IPython “magic commands”
  - Some are familiar from SPEC:
    - mv, umv, mvr, umvr: move motors
    - ascan, dscan: scans (step-wise or continuous, multidimensional also)
    - ct: count for a given time
    - wa, wm, …: query motor positions
- Extensible in Python:
  - Macros (relatively easy to develop them)
  - Controllers (timers&gates, 0-1-2D counters, multi-axis motor controllers, I/O registers, pseudo counters, pseudo motors)
  - Recorders (usually the out-of-the-box ones are fine: .spec, .hdf5, .nxs)
- Not the only viable method: MacroExecutor GUI (or build your own)



Taurus: graphical user interface

- Based on Qt v5
- Widgets associated to Tango device attributes
  - Automatic polling for changes
  - Instantaneous visualization of device state
  - Control of devices
- Taurus Designer: Qt Designer extended with Taurus widgets
- Might be applicable to others systems beside Tango (e.g., EPICS)

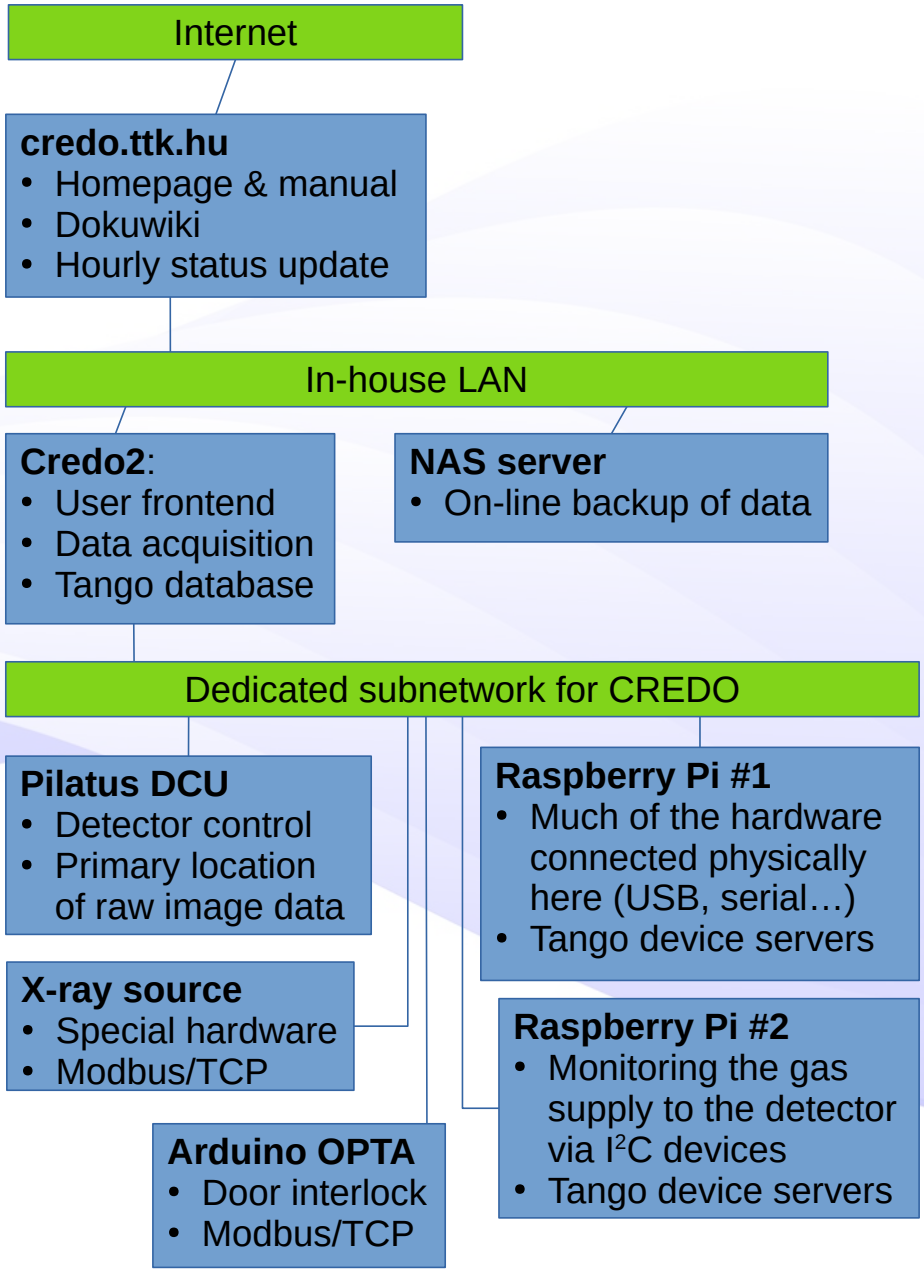




How it is done in ~~reality~~ our lab?

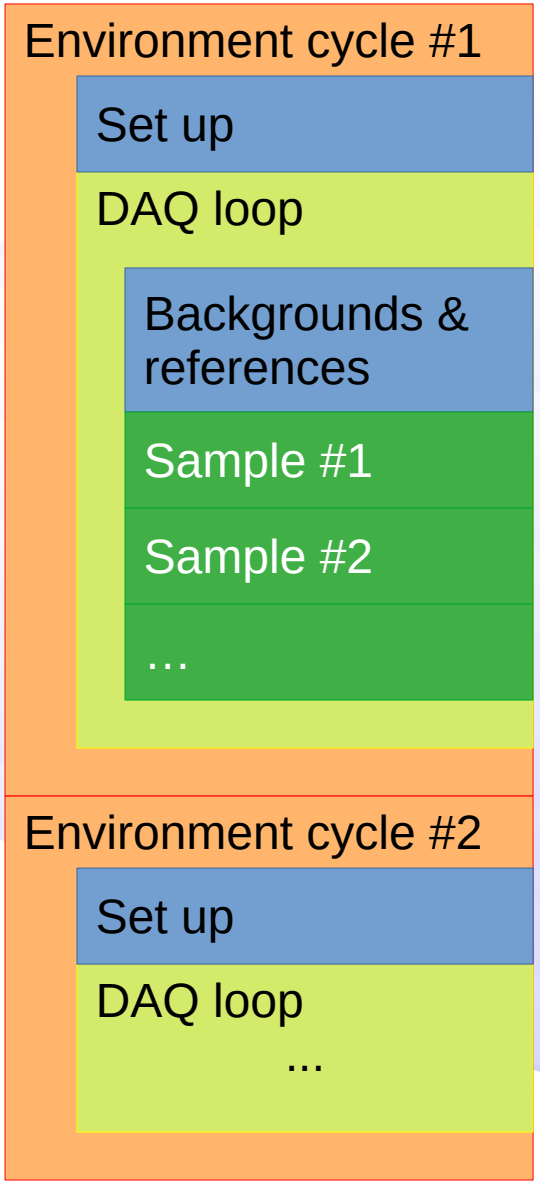
Tango and Sardana in CREDO

- Computers on a dedicated subnet
  - **Credo2:** instrument control, user interface, SQL database, DatabaseDS
  - **Raspberry Pi #1:**
    - RS-232 and USB device connections
  - **Raspberry Pi #2:**
    - I²C devices for detector gas supply monitoring
  - **X-ray source** (Modbus over TCP)
  - **Pilatus-300k DCU:**
    - Dedicated computer, runs `camserver`
  - **Arduino OPTA**
    - door interlock, to be implemented)
- Device servers:
  - Interfaces (SerialLine, Modbus) from the Tango Class Catalogue, mostly on RPi #1
  - Hardware-specific: skeleton made by Pogo, coded with PyTango
  - Software “devices”:
    - Sardana MacroServer and Pool
    - …



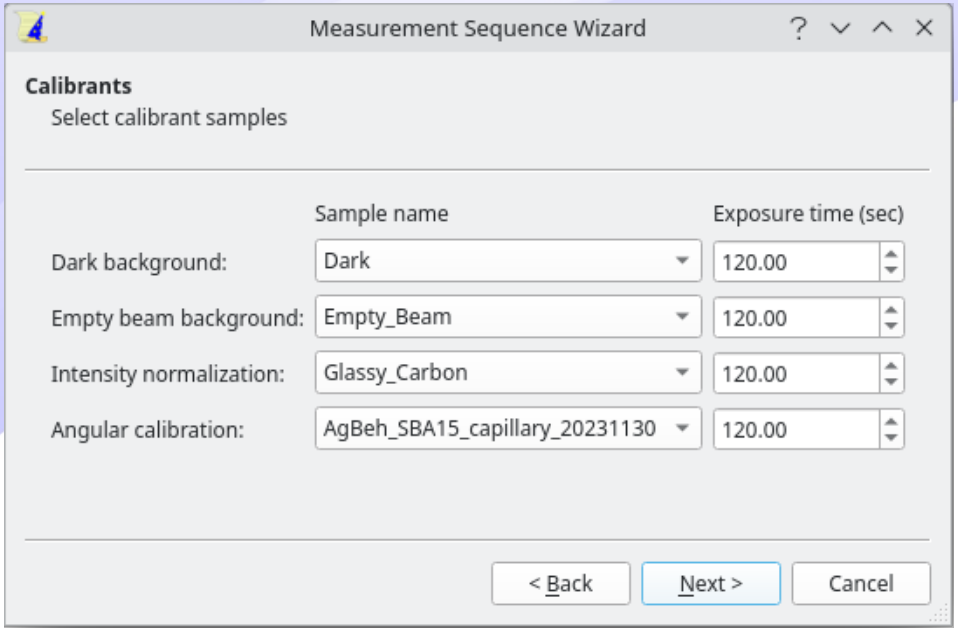
Data Acquisition Routine in CREDO using Tango and Sardana

- **Geometry set-up:** Aperture alignment using *scans*: maximizing beam intensity, minimizing “parasitic scattering”
  - Virtual counters: full detector area sum, …
- **Finding samples:** motor positions corresponding to individual samples on the motorized sample stage: *scans* again
  - Updating the sample database
- **Transmission measurement:** only once, before data acquisition
- **Data acquisition sequence**
  - Environment cycle: constant *in situ* parameters
  - Data acquisition loop: recording images
    - First measure correction and calibration images
    - Then the samples sequentially
    - “Rinse, repeat”
- At every step: Sardana macros



DAQ Sequence in TOML files

- TOML: Tom's Obvious Minimal Language (https://toml.io)
- Human-readable (and editable) syntax
- Less syntactic cruft than JSON
- Can be validated against a schema (semantic check)
  - Required fields
  - Sanity of values (type, range)
- Can be created by a step-by-step GUI wizard or edited by hand

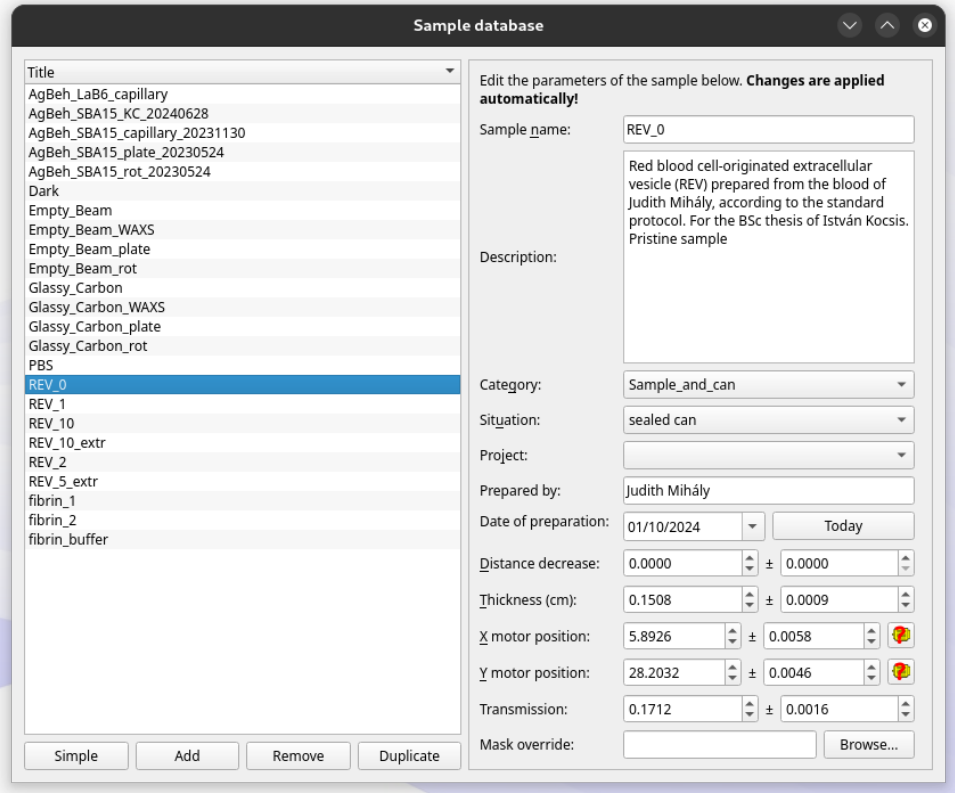



```
# CREDO experiment sequence generated on 2024-07-18 14:36:58.354822
# Run it in Spock with the command
#    saxsseq <absolute path to script file>

[config]
iterations = 1
# … omitted for clarity

[init]
beamstopin = true
xraypower = "full"
closeshutter = true
openshutter = false

[trim]
energy = 4024  # eV
gain = "highG"

[finish]
xraypower = "off"

[references]
dark.name = "Dark"
dark.time = 120.000  # sec
empty.name = "Empty_Beam"
empty.time = 120.000  # sec
absint.name = "Glassy_Carbon"
# … omitted for clarity

[[sample]]
name = "DSPC_chol_PEG_1x"
time = 300.000  #sec
repeats = 6

[[sample]]
name = "DSPC_chol_PEG_2x"
time = 300.000  #sec
repeats = 6

[[cycle]]
iterations = inf
refs = true
samples = true
temperature.set = 25.000
temperature.stabilitycheckradius = 0.100  # °C
temperature.stabilitychecktime = 30.000  # sec
```

Additional Concepts – “Misusing” Tango

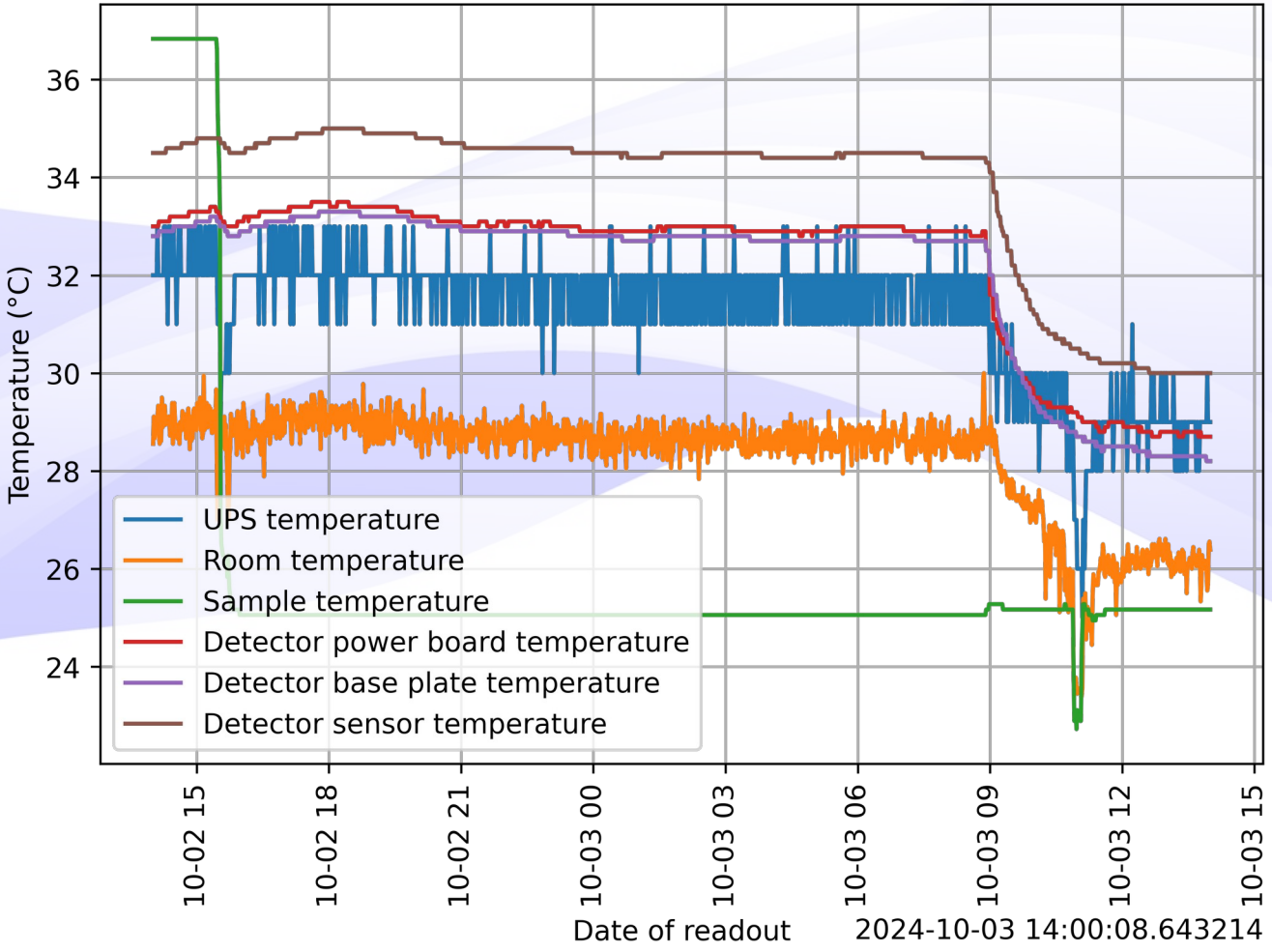
Additional Concepts – or How to “Misuse” Tango and Sardana

- **Sample database**
  - SQL table
    - Short, unique title + longer, free-format description
    - Thickness (for absolute intensity calibration)
    - Transmission (measured separately)
    - X and Y motor positions
    - …
  - Managed by: SampleStore Tango device
  - Editing and viewing through GUI and with custom Spock macros
  - NeXus-ready…



Additional Concepts – or How to “Misuse” Tango and Sardana

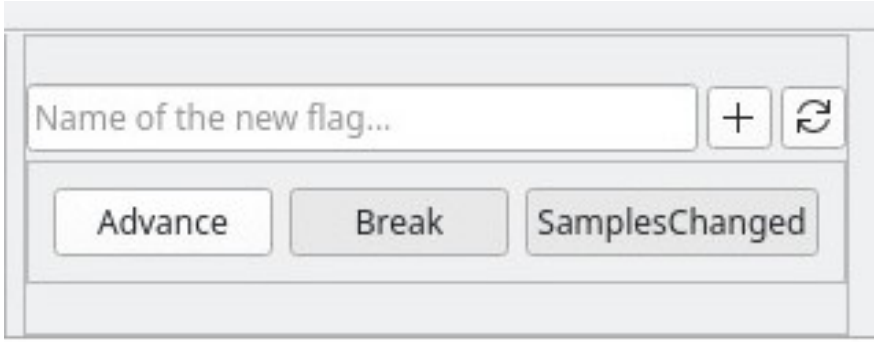
- **State logger: Tango Device**
  - Periodically saving selected attribute values into a SQL database
  - Script: hourly run, draw graphs
  - Periodically uploaded to the homepage (https://credo.ttk.hu/status:start)



Additional Concepts – or How to “Misuse” Tango and Sardana

- **On-line data reduction pipeline**
  - Required calibrations and corrections to be performed on each detector image as soon as they are ready
  - Implemented as a Tango device
  - Processing queue
  - Images submitted as soon as ready
  - Corrected data saved on disk

- **Flags**
  - On/off switches
  - User interaction into running macros
  - E.g., “Break”: stop data acquisition after the current image is done



What's past and what's next?

- Summary:
  - Frameworks developed with large-scale facilities in mind, but can be downscaled to lab-based instruments, too
  - Tango: software bus, distributed object model of real and virtual devices
  - Sardana: macros and procedures (and the needed devices), on Tango grounds
  - Taurus: GUI & abstraction layer (Tango, EPICS, …)
- Omitted (for now): data storage (See you tomorrow!)
  - How, where, what, why
  - Formats
- Guided tour in the SAXS lab for those interested, after this lecture
- Code available at:
  - https://gitlab.com/bionano/credo

TANGO
Connecting things together

CREDO
CREATIVE RESEARCH EQUIPMENT FOR DIFFRACTION

Thank you for your attention!

- Research Group for Biological Nanochemistry, HUN-REN Research Centre for Natural Sciences (https://bionano.ttk.hu/biological-nanochemistry)
- CREDO SAXS laboratory (https://credo.ttk.hu)