

The logo for MAX IV, featuring the letters 'MAX IV' in a stylized, white, sans-serif font. The letters are interconnected, with a white swoosh or underline that loops around the 'M', 'A', and 'X' and extends under the 'I' and 'V'. The background is dark with vibrant, multi-colored streaks of light in shades of blue, purple, and orange, creating a dynamic, high-tech atmosphere.

MAX IV

A Taurus GUI for Automated BioSAXS Experiments at the CoSAXS Beamline

Lukas Wittenbecher¹, Hanno Perrey², Lin Zhu¹, Miaoxin Gong³, Benjamin Folsom¹

(1) MAX IV Laboratory, Lund, Sweden

(2) European Spallation Source, Lund, Sweden

(3) Prevas Test & Measurement, Lund

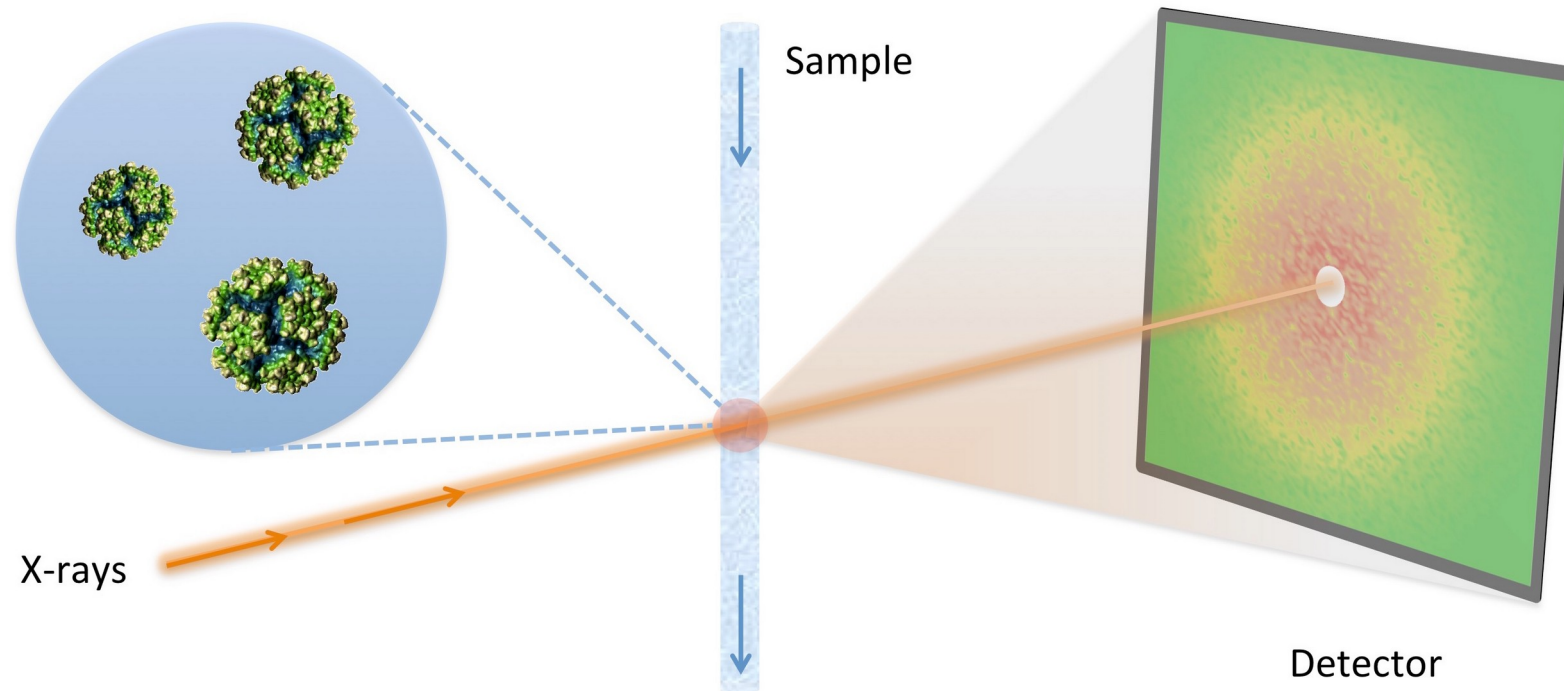
Outline

- What is BioSAXS?
- GUI Motivation
- Demo
- Control system integration
- How we use Taurus



BioSAXS

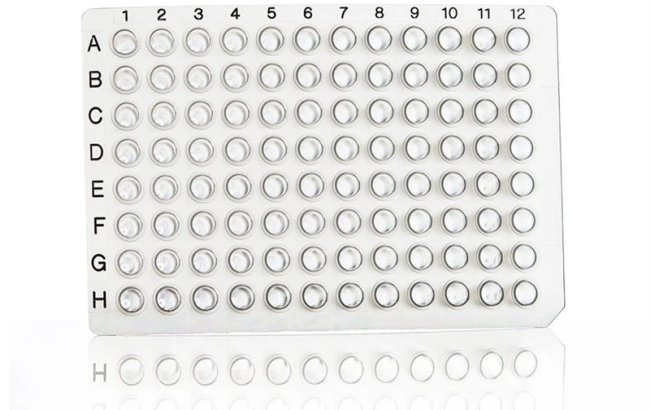
- Small-angle X-ray scattering (SAXS) from Biological samples



https://en.wikipedia.org/wiki/Fluctuation_X-ray_scattering

BioSAXS

- ~20 seconds per measurement
- Dozens to hundreds of samples
- **Automation**

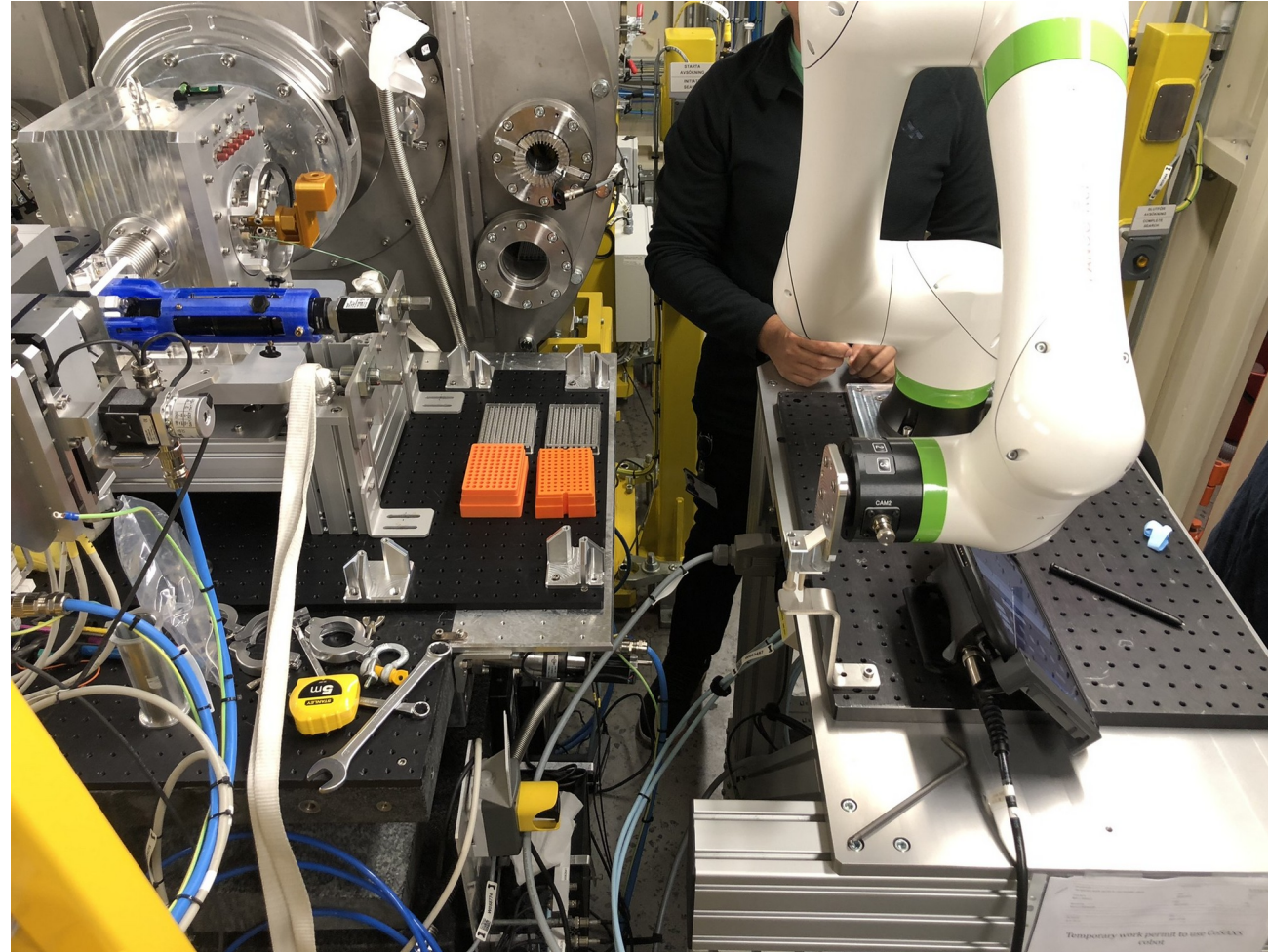


<https://www.azenta.com/products/96-well-non-skirted-pcr-plate>



<https://crx.fanuc.eu/bg/fanuc-cobot-crx-5ia/>

BioSAXS @ CoSAXS



BioSAXS GUI – basic requirements

- **Sample management**
 - Create a pool of samples available for measurements
- **Scheduling**
 - Compile an experimental schedule that will be run automatically (“Autorun”)
- **Configuration**
 - Set parameters such as sample volume, exposure time, ...
- **Feedback**
 - Status updates from an ongoing experiment

GUI demo

BioSAXS Experimental Control Interface

File View Taurus Tools Help

Load Perspectives_ Save Perspective

Tray 1 Tray 2

1 1 2 3 4 5 6 7 8 9 10 11 12

A A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12

B B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12

C C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12

D D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12

E E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12

F F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12

G G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12

H H1 H2 H3 H4 H5 H6 H7 H8 H9 H10 H11 H12

PROCESS selected delete selected save JSON clear tray

MIX selected edit selected load JSON manual

A1	My sample #1	Buffer:	<input type="checkbox"/>	
A2	My sample #2	Buffer:	<input type="checkbox"/>	
A3	My sample #3	Buffer:	<input type="checkbox"/>	
B1	My buffer #1	Buffer:	<input checked="" type="checkbox"/>	
B2	My buffer #2	Buffer:	<input checked="" type="checkbox"/>	
B3	My buffer #3	Buffer:	<input checked="" type="checkbox"/>	

Autorun Manual

move up clear schedule start

move down delete selected stop

1 Process [T1-A1] DONE

Description My sample #1

Advanced settings

2 Process [T1-B1][B] RUNNING

Description My buffer #1

Advanced settings

Comment

Buffer

Viscous

Well ID T1-B1

Volume [ul] 20,00

Load flow rate [ul/s] 10,00

Pipetting flow rate [ul/s] 10,00

Loading timeout [s] 60

Use camera feedback

Loading duration [s] 10

Exposure time [ms] 20

Exposure flow rate [ul/s] 10,00

Number of frames 200

Exposure timeout [s] 10

3 Process [T1-A2] PENDING

Description My sample #2

Advanced settings

4 Process [T1-B2][B] PENDING

Description My buffer #2

Advanced settings

5 Mix [T1-A2 into T1-A3] PENDING

Advanced settings

6 Process [T1-A3] PENDING

Description My sample #3

Advanced settings

Autorun status

RUNNING

Step Process [T1-B1][B]

Substep Dispense sample

Last error

Capillary live view

Subdevice status

camera	None
cleampump	RUNNING
robot	RUNNING
door	None
pandabox	RUNNING
samplepump	RUNNING
pipettepump	RUNNING
metadata	RUNNING
airvalve	RUNNING
gatevalve	RUNNING

Tip status

Next tip: 3 of 192

Change next tip

GUI demo

"Autorun schedule" – a list of items to be processed. An item could be either processing a sample (loading, measuring, cleaning), or mixing the contents of two sample wells.

Status widget, provides information about the ongoing experiment

Tray editor – allows the user to specify the wells in which sample has been loaded and to build a pool of samples. For each sample, the user may enter a description and set the 'buffer' flag.

The screenshot shows the BioSAXS Experimental Control Interface. The interface is divided into several panels:

- Tray Editor (left, green border):** A grid of 96 wells (A1-H12) with a list below for sample descriptions and buffer flags.
- Autorun Schedule (center, orange border):** A list of 6 processes with their descriptions and status (DONE, RUNNING, PENDING).
- Status Widget (right, blue border):** Shows the current process (Process [T1-B1][B]) and its substep (Dispense sample).
- Capillary Live View (bottom right):** A plot showing the capillary live view.
- Subdevice Status (bottom right):** A list of subdevices (camera, cleampump, robot, door, pandabox, samplepump, pipettepump, metadata, airvalve, gatevalve) with their current status (None or RUNNING).

Sample live view

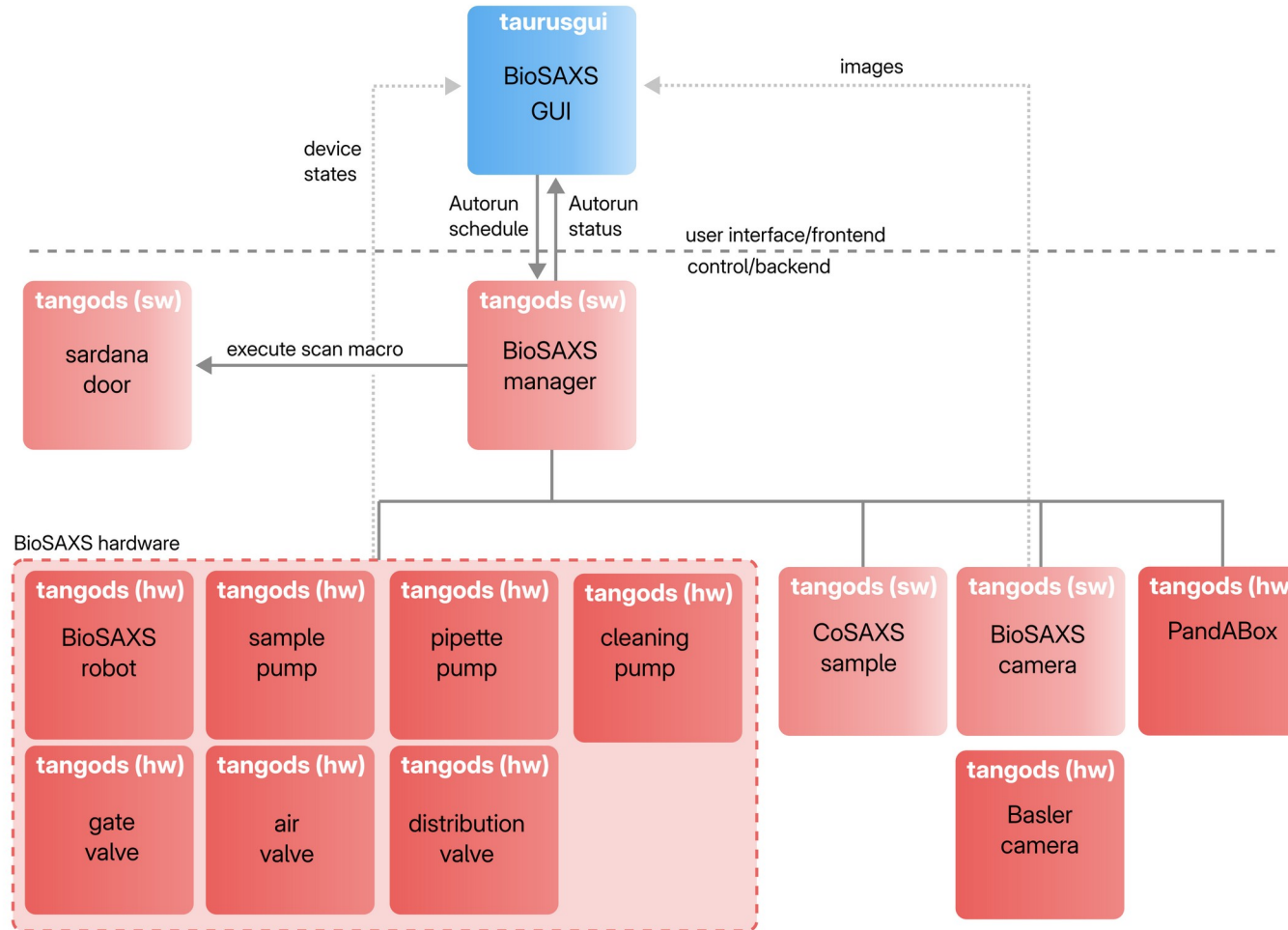
States of sub-devices (valves, pumps, robot, ...)

Control system integration

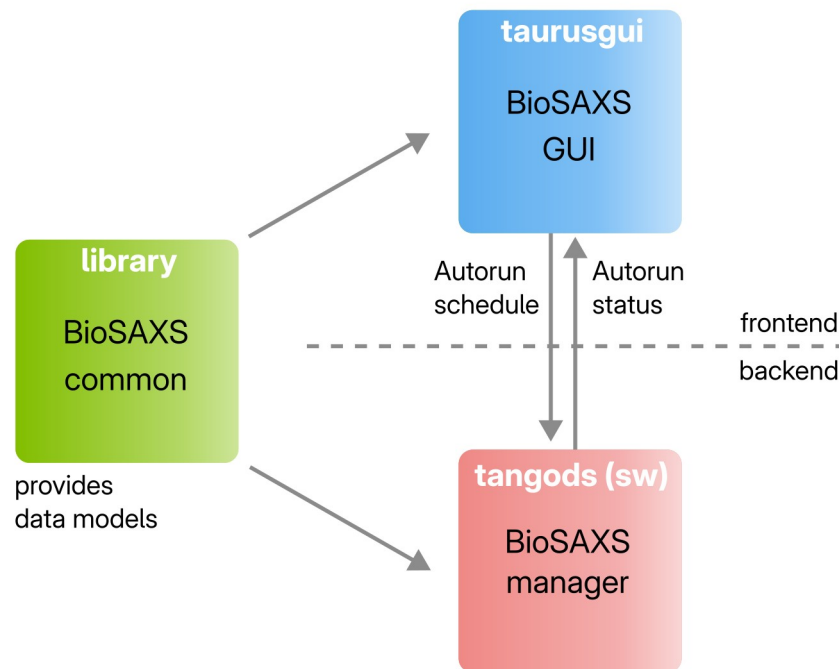
User interface

Orchestration
(sardana/Tango)

Tango devices
(components)



GUI and Manager Device



- GUI-manager communication primarily via
 - `AautorunSchedule` – the list of items to process and their configuration
 - `AautorunStatus` – information about current step/item, sub-step and exception

```
from pydantic import BaseModel
```

```
class AautorunStatus(BaseModel):
```

```
    """
```

```
    Provides information about the current status of an autorun (ongoing or finished).
```

```
    Provides the data structure for the `AautorunStatus` attribute of the  
    BioSAXS manager tango device.
```

```
    """
```

```
    status: Status | None = None
```

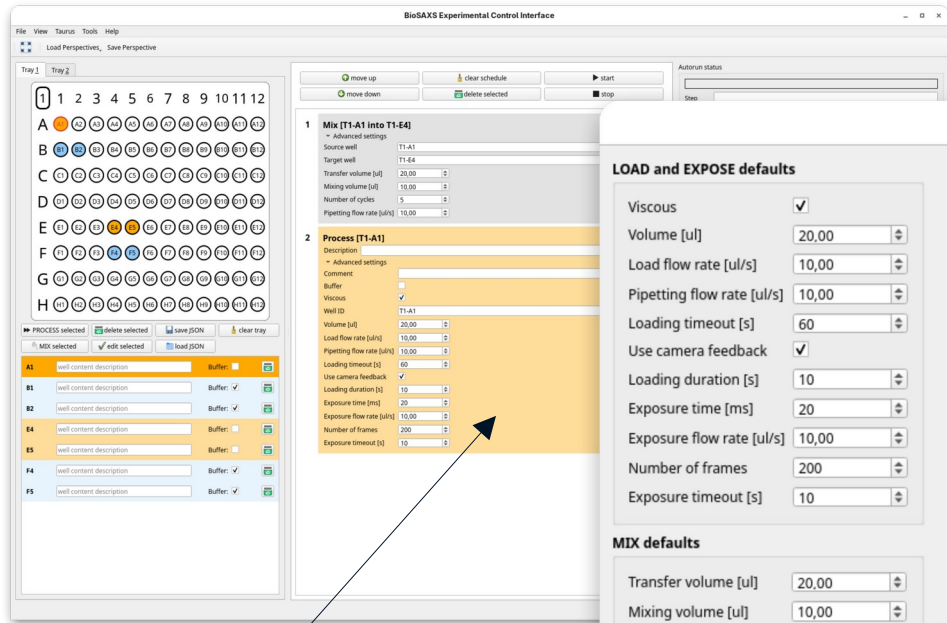
```
    step: AautorunStep | None = None
```

```
    substep: SubStepInfo | None = None
```

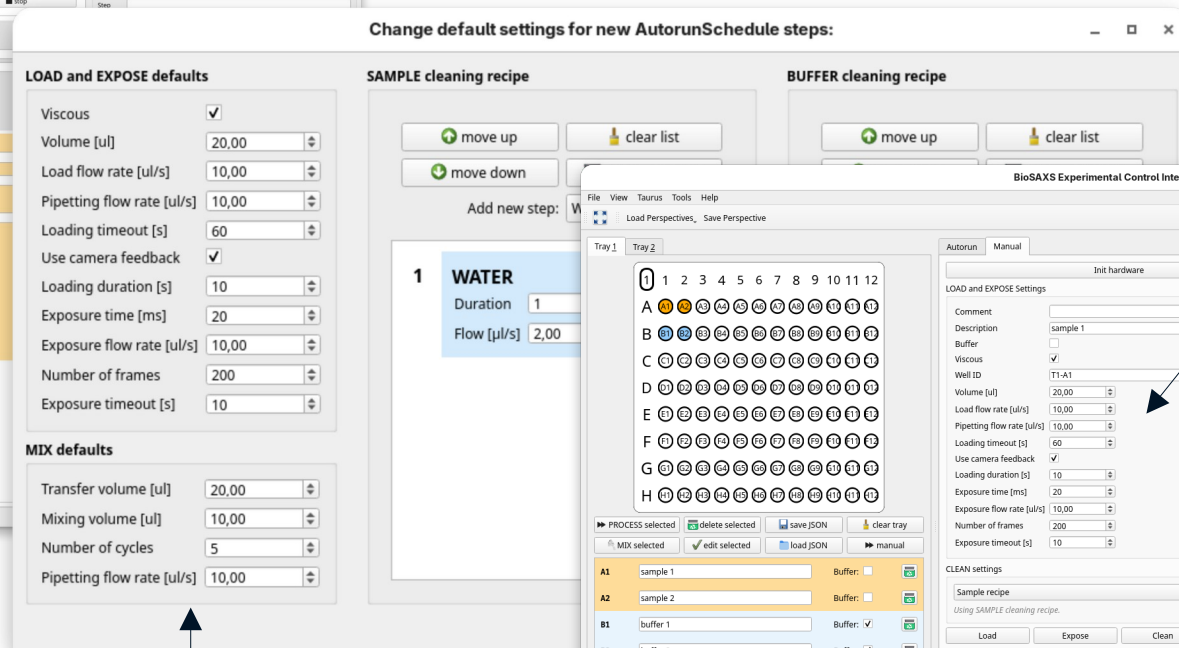
```
    error: AautorunExcInfo | None = None
```

- Definition of configuration parameters in shared lib
 - Configuration: used in manager device to configure hardware
 - GUI: Various widgets for editing configuration parameters

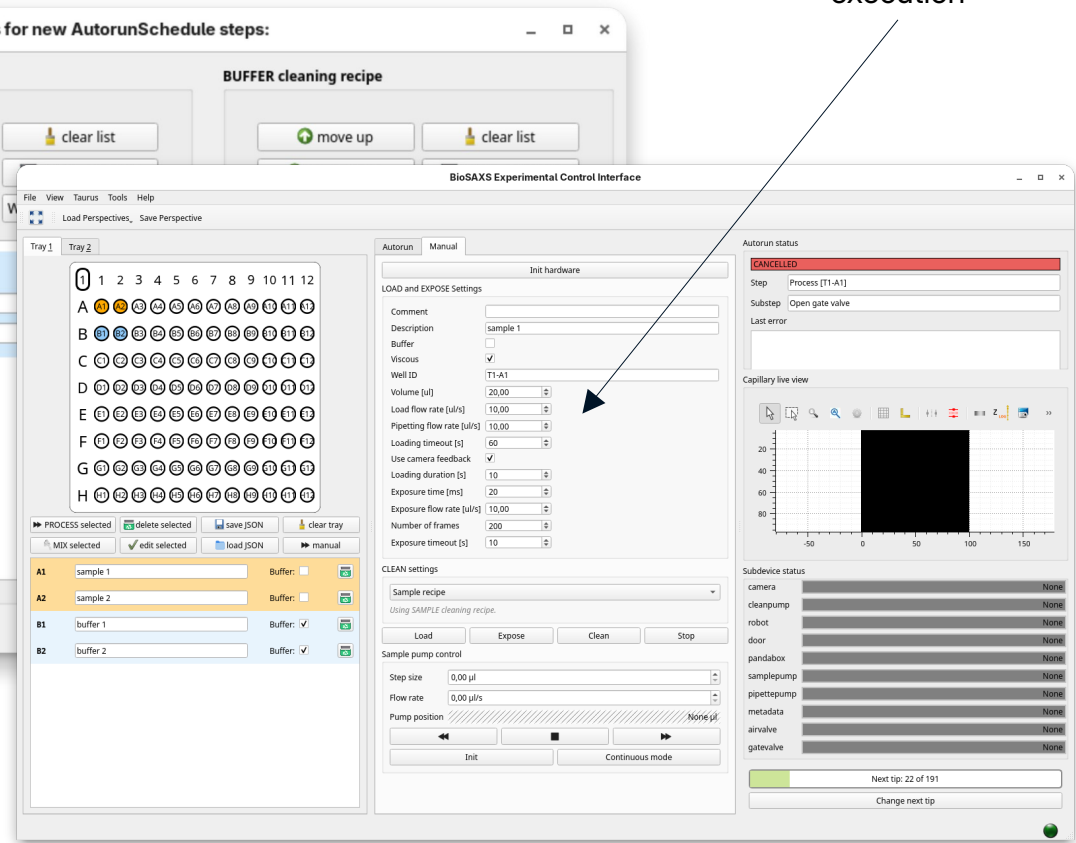
Configuration Parameters



Autorun items can be configured individually



Default settings editor widget



Configuration for manual step execution

Configuration Parameters

common library

Configuration
for manual step
execution

GUI

```
from pydantic import BaseModel, Field
```

```
class MixSettings(BaseModel):
```

```
    well_id_from: str = ""
    well_id_to: str = ""
    transfer_volume: float = 20.0 # in µl
    mixing_volume: float = 10.0 # in µl
    num_cycles: int = 5 # how often to aspirate and dispense
    pipetting_flow: float = 10.0 # in µl/s
```

- Initial approach: settings models in common library define parameter **name**, **type** and **default value**
- Parameter **meta-information** (min, max, step size, read only, ...) in a separate list on the GUI side
- Disadvantages:
 - Spreads information about config parameters across two repositories
 - Creates hidden dependency between lib and GUI

```
class MixSamplesItem(AutorunBaseItem):
```

```
    """
    Widget for representing a 'MIX samples' step in the autorun schedule.
    """
```

```
    configParams: list[ConfigParameter] = [
```

```
        ConfigParameter(
            name="well_id_from",
            category="mix",
            label="Source well",
            dataType=str,
            defaultEditable=False,
            expert=True,
            readOnly=True,
```

```
        ),
        ConfigParameter(
            name="well_id_to",
            category="mix",
            label="Target well",
            dataType=str,
            defaultEditable=False,
            expert=True,
            readOnly=True,
```

```
    ),
```

```
    ...
```

```
]
```

Configuration Parameters

common library

GUI

```
from pydantic import BaseModel, Field
```

```
class MixSettings(BaseModel):
```

```
...
```

```
transfer_volume: float = Field(
    default=20.0,
    ge=0.0,
    le=200.0,
    title="Transfer volume",
    description="Volume to transfer from source to target well.",
    json_schema_extra={
        "category": "mix",
        "unit": "ul",
        "read_only": False,
        "default_editable": True,
        "step_size": 0.1,
        "expert": True,
    },
)
```

```
...
```

```
class MixSamplesItem(AutorunBaseItem):
```

```
"""
```

```
Widget for representing a 'MIX samples' step in the autorun schedule.
```

```
"""
```

```
configParams: list[ConfigParameter] = paramsFromModel(MixSettings)
```

1 **Mix [T1-A1 into T1-E4]** PENDING

▼ Advanced settings

Source well

Target well

Transfer volume [ul]

Mixing volume [ul] Volume to transfer from source to target w

Number of cycles

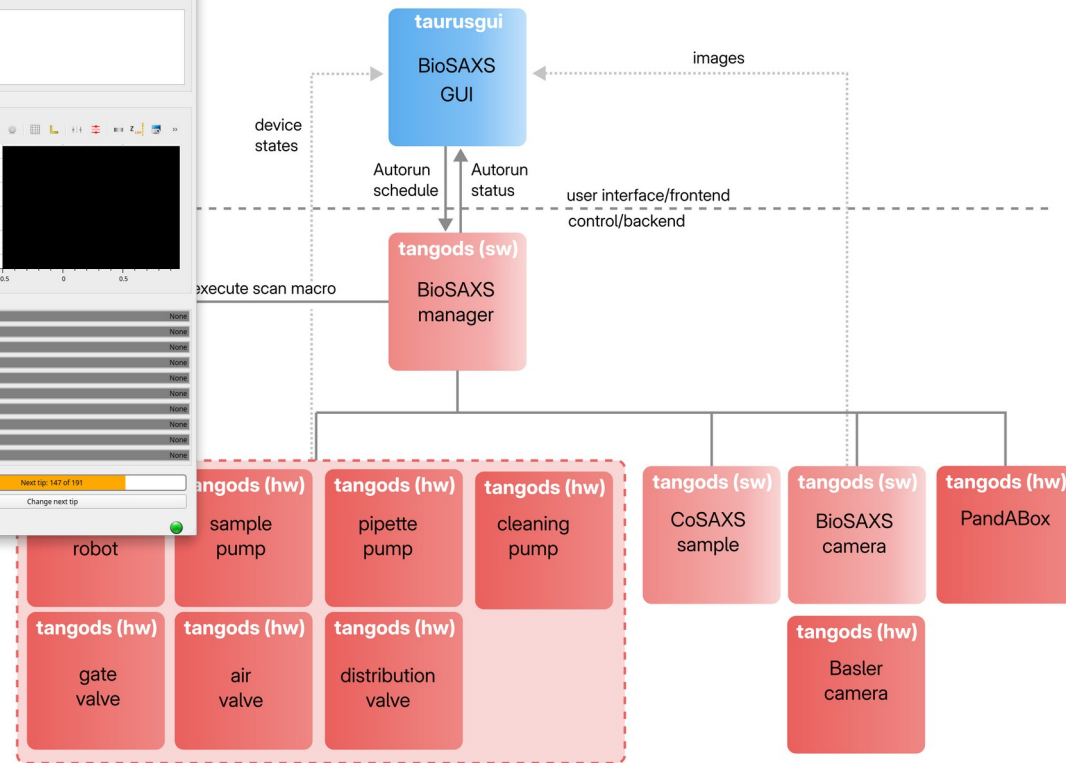
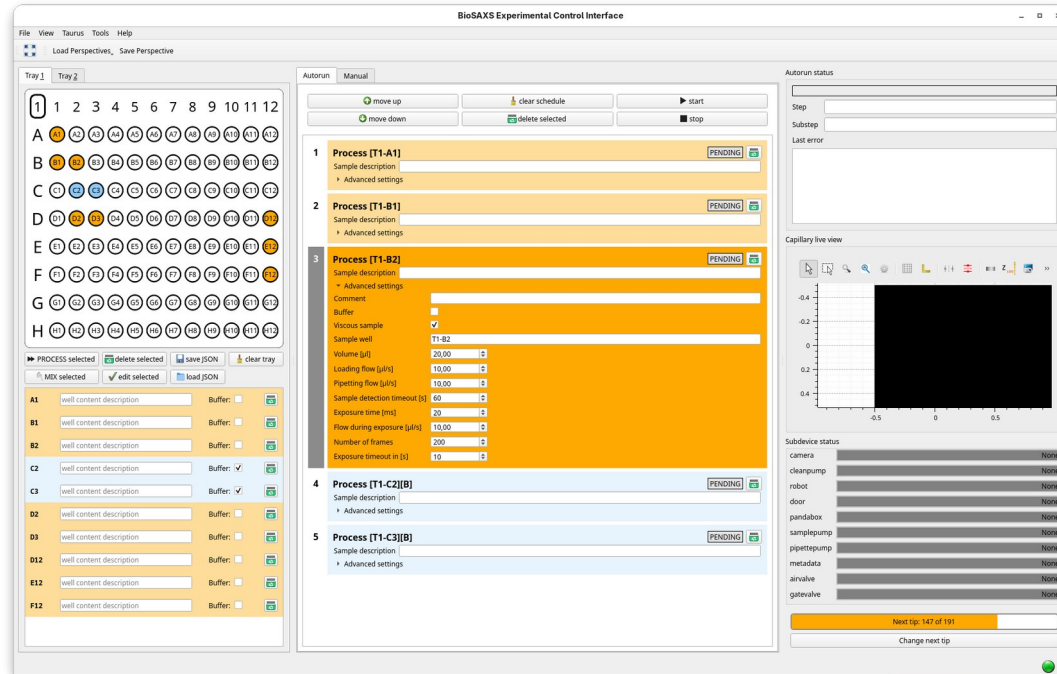
Pipetting flow rate [ul/s]

- Current approach: settings models in common library extended to contain parameter meta information
- GUI: widgets that expose configuration parameters to user are build from the model
- Single source of truth for configuration parameters
- Easy to add configuration parameters

Taurus

- `TaurusApplication` and `TaurusMainWindow` as base
- Display values from Tango control system
 - Live images of capillary
 - Device states
- Handle change events in custom widgets
 - Status updates during "Autorun"
 - Dynamic GUI configuration

Thank you for your attention!



The image features the word "MAXIV" in a stylized, white, sans-serif font. The letters are thick and have a slightly irregular, hand-drawn appearance. A thin, white, curved line arches over the letters, starting from the top of the 'M' and ending at the top of the 'V'. The background is a dark, almost black, space filled with numerous diagonal streaks of light. These streaks are primarily blue and purple, with some orange and red highlights, creating a sense of motion and energy. The streaks are most concentrated in the upper left and lower right corners, with some appearing as thin, sharp lines and others as thicker, more diffuse bands.

MAXIV