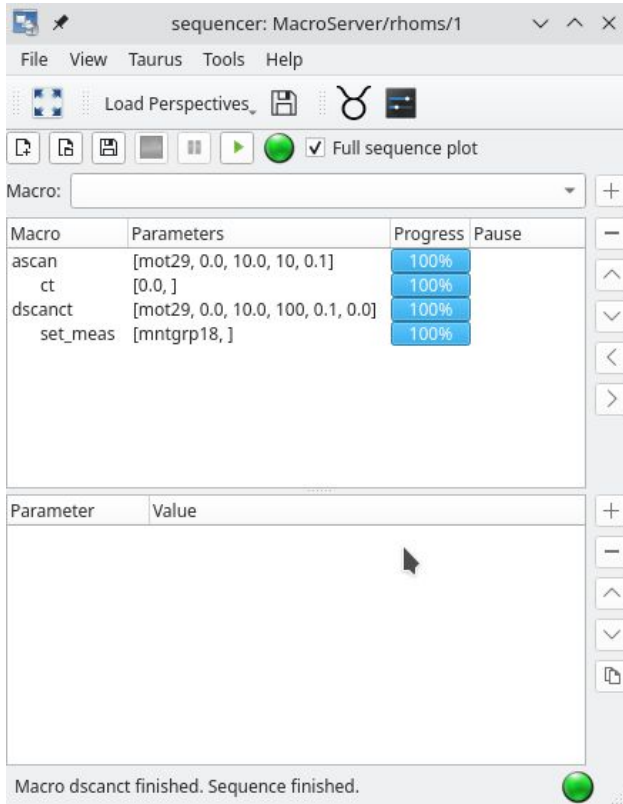# Complex experiment applications at ALBA

Jordi Aguilar
Roberto Homs
Albert Ollé
Zbigniew Reszela
Oriol Vallcorba
Steven Wohl

on behalf of
ALBA Synchrotron
Controls Section

Sardana sequencer

- Easy to use hooks
- Easy to Start/Pause/Restart/Stop
- Easy to save
- Easy to check progress status
- Load from text file (.txt), does not work hooks and params repeats, XML file support them but more complex to write.
- Do not possible to do loops, macro repeat is not adapted (feature/bug)
- Do not possible to do logic blocks (if/else)
- Do not possible to validate parameters (motor limits, valid integration time, …)
- Do not possible to catch exceptions and recover from fails

BL22 CLAESS Experiment UI

Initial requirements:
Easy for external user
Robust
   Protections
   Recover from fails
Repetitions
Automatic configuration
Temperature Control
Time estimation
Log files
Simulation mode
Work for 90% of proposals
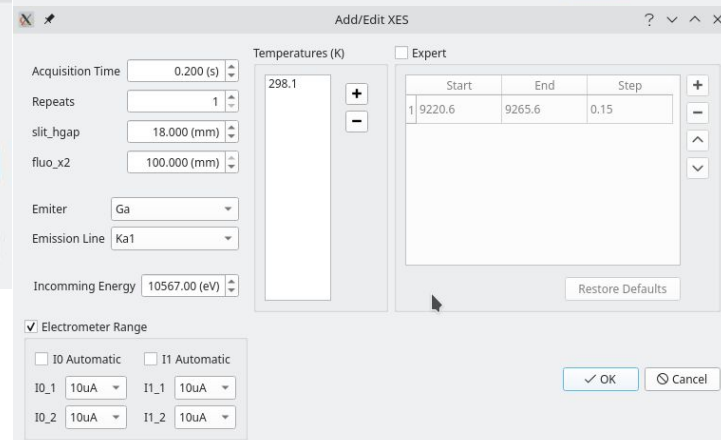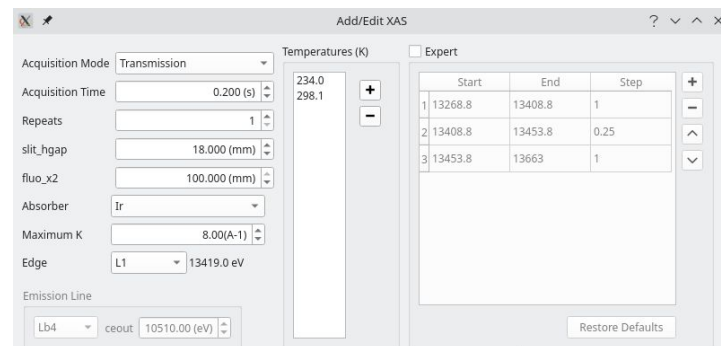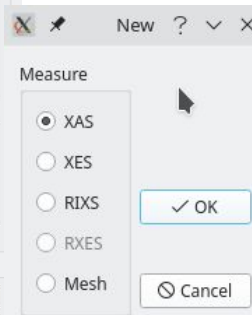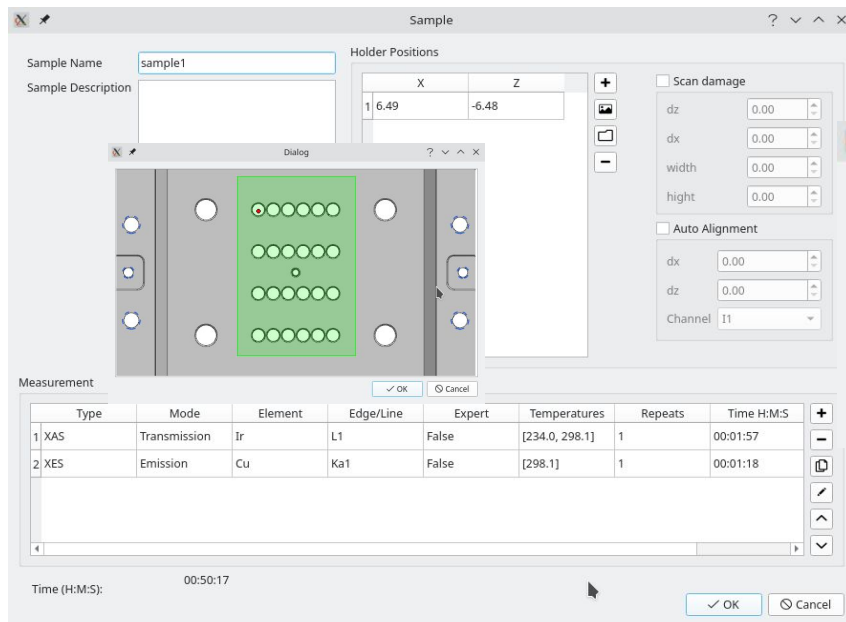Sardana python 2.7
GUI python 3.9 pyqt=5.15

BL22 CLAESS Experiment UI

Sample position protection (green part)
Measure type selection
Automatic specific configuration, allow Expert mode
Electrometer automatic configuration

BL22 CLAESS Experiment UI

BL22 CLAESS Experiment UI

```python
def run(self, filename, run, wait_fe, force_qExafs):
    self.demo = not run
    self._exp_data = None
    self._force_qExfas = force_qExafs
    self._interrupt = False
    self.log_file = None
    try:
        if self.demo:
            self.execMacro = self.info
        self._exp_data = None
        self.execMacro('blreconfig')
        if wait_fe:
            self.execMacro('blshopen')
        self._run(filename, wait_fe)
    except InterruptException:
        self._interrupt = True
        raise
    except Exception as e:
        self.error( msg: 'Exception %s', *args: e, exc_info=True)
    finally:
        if self._exp_data is not None:
            self._save_env()
        if self._interrupt:
            self.error('Macro aborted. Run blreconfig!')
        else:
            self.execMacro('blreconfig')
        if self.log_file is not None:
            for handler in self.log_file.handlers:
                handler.flush()
                self.log_file.removeHandler(handler)
                del handler
```

```
10ke27ke
├── 20250623_173057_log.txt
├── Mo
│   └── Mo
│       ├── Mo_Mo.dat
│       └── Mo_XAS_Mo_K_1_0_000.dat
├── Nb
│   └── Nb
│       ├── Nb_Nb.dat
│       └── Nb_XAS_Nb_K_1_0_000.dat
├── Pd
│   └── Pd
│       ├── Pd_Pd.dat
│       └── Pd_XAS_Pd_K_1_0_000.dat
├── Ru
│   └── Ru
│       ├── Ru_Ru.dat
│       ├── Ru_XAS_Ru_K_1_0_000.dat
│       └── Xspress3_Images
│           └── Ru_Ru_129061
│               ├── x_images0000.edf
│               ├── x_images0001.edf
│               └── ......
├── Sr
│   └── Sr
│       ├── Sr_Sr.dat
│       ├── Sr_XAS_Sr_K_1_0_000.dat
│       └── Xspress3_Images
│           └── Sr_Sr_129064
│               ├── x_images0000.edf
│               ├── x_images0001.edf
│               └── ....
├── Cu
│   └── Cu
│       ├── Cu_Cu.dat
│       └── Cu_XAS_Cu_K_1_0_000.dat
├── Fe
│   └── Fe
│       ├── Fe_Fe.dat
│       └── Fe_XAS_Fe_K_1_0_000.dat
├── Ni
│   └── Ni
│       ├── Ni_Ni.dat
│       └── Ni_XAS_Ni_K_1_0_000.dat
├── Se
│   └── Se
│       ├── Se_Se.dat
│       └── Se_XAS_Se_K_1_0_000.dat
├── Zn
│   └── Zn
│       ├── Zn_XAS_Zn_K_1_0_000.dat
│       └── Zn_Zn.dat
```

Sardana TODOs / DONE
Use environment for state saving (use deepcopy)
Implement abort/stop propagation on gscan

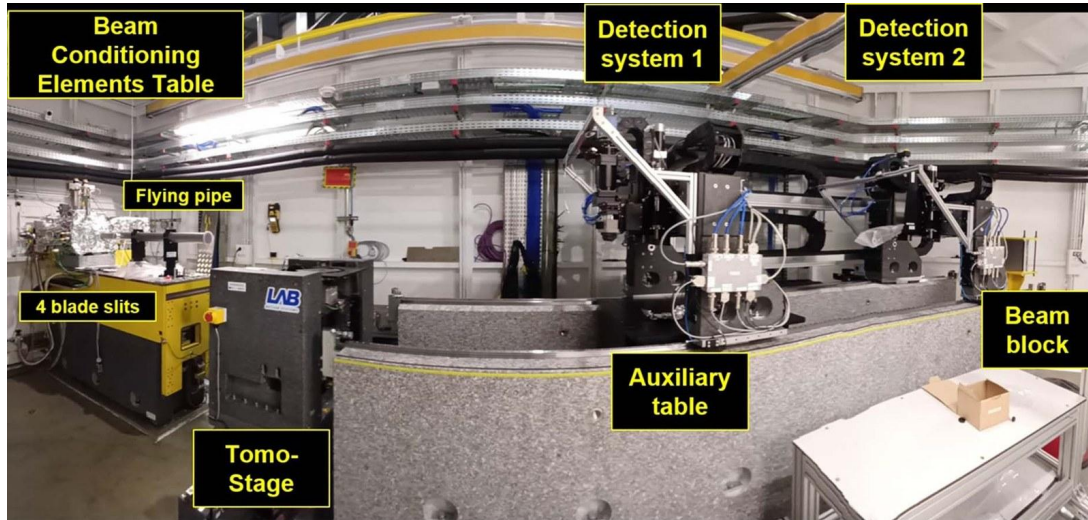# BL22 CLAESS Experiment UI - what's next?

Connect GUI and MacroServer now both are running in the same python version:

- Allow to Start/Pause/Restart/Stop
- Show real progress
- Start from specific measure

Reuse it on BL16 Notos similar experiment measures.

BL31 Faxtor

Camera	FPS
PCO.edge 4.2	100
PCO DiMaX CS4	1102
C-Blue	85
Phantom S710	5700

GONIO: ACS

BL31 Faxtor *Tomo Scan*

Sequence of scans:
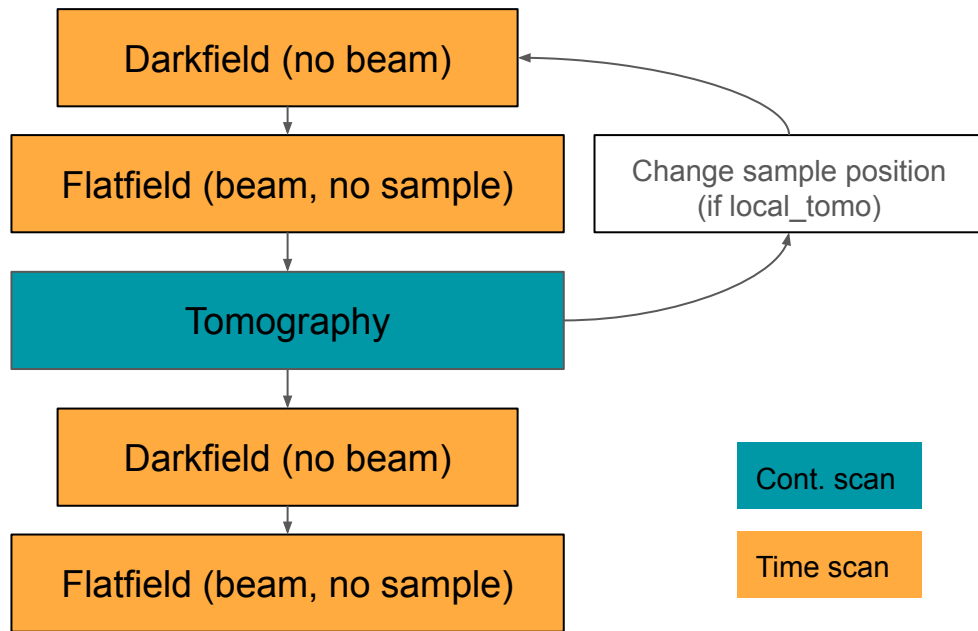
Requirements:
- Handle fast shutter
- Asynchronous motor movements between scans
- Custom file storage
- Restart sequence from a given point
- Custom gonio behaviour:
  - Custom acceleration
  - Uninterrupted rotation

BL31 Faxtor *Tomo Scan*

- Scan configuration is serializable a Pydantic model

```python
class TomoModel(BaseModel):
    tomo_steps: list[Step]
    current_slice_id: int =


class Step(BaseModel):
    sample_position: SamplePosition = SamplePosition()
    projections: int = Field(..., description="Number of images to take")
    exposure_time: Union[float, int] = Field(..., description="Time of the exposure")
    latency_time: Union[float, int] = Field(..., description="Time of the exposure")
    mode: Mode
    state: ScanState
    hooks: Dict = Field(default_factory=dict,exclude=True)
    point_id: int
    channel_refs:Dict[str, str] = Field(
        default_factory=dict,
        description="Key-value pairs for 2d channels and respective computed ref pattern"
    )
    gonio_scan: bool = False
    gonio_rotation: float = 180
```

BL31 Faxtor *Tomo Scan*

- Scan configuration is serializable a Pydantic model
- Scan can be stored and loaded from environment

Store

```
tomo_model = self.tomo_model.dict()
self.setEnv(bl31.ENV_KEYS.TOMO_MODEL, tomo_model)
```

Load

```
model = TomoModel(**self.getEnv(bl31.ENV_KEYS.TOMO_MODEL))
```
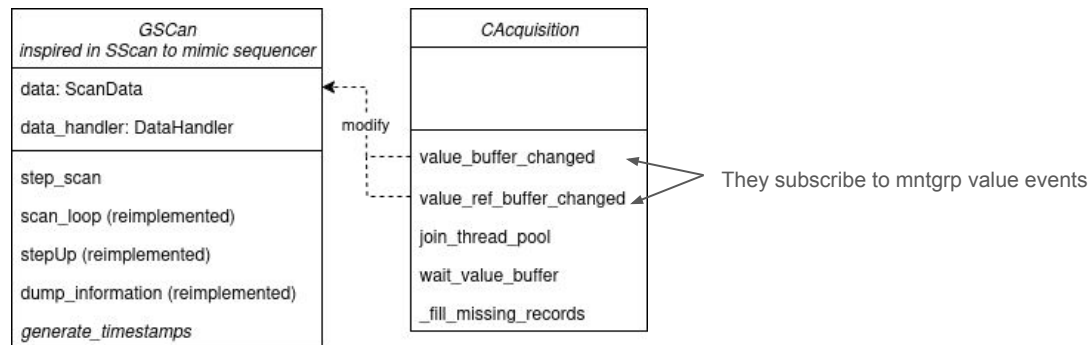
BL31 Faxtor *Tomo Scan*

- Scan configuration is serializable a Pydantic model
- Scan configuration can be stored and loaded from environment
- Scan configuration is updated at every step

```
def generator():
    for step in tomo_model:
        step.update(state=ScanState.Started)
        save tomo_model to env
        yield step
        step.update(state=ScanState.Finished)
        save tomo_model to env
```
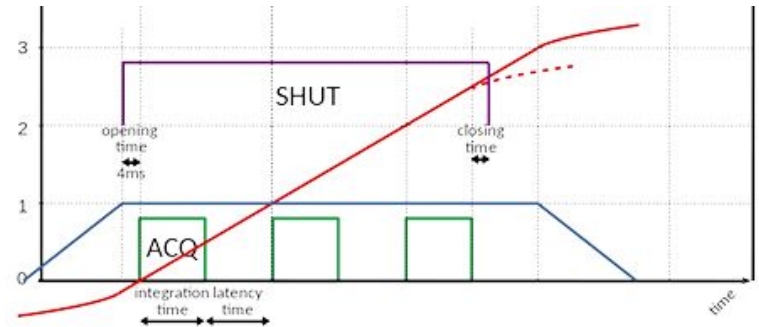
BL31 Faxtor *Tomo Scan*

- Scan configuration is serializable a Pydantic model
- Scan configuration can be stored and loaded from environment
- Scan configuration is updated at every step
- Scan Framework API used:
  - Inherit from GScan and CAcquisition
  - We could reuse more code

GSCan *inspired in SScan to mimic sequencer*
data: ScanData
data_handler: DataHandler
step_scan
scan_loop (reimplemented)
stepUp (reimplemented)
dump_information (reimplemented)
generate_timestamps

modify

CAcquisition
value_buffer_changed
value_ref_buffer_changed
join_thread_pool
wait_value_buffer
_fill_missing_records

They subscribe to mntgrp value events

Attributes and commands used or reimplemented from Scan API

BL31 Faxtor *Tomo Scan:* shutter

- Time to open and time to close is considered in order to avoid sample damage
- Uses **MultipleSynchDescription** and **Shutter** object
- Shutter in measurement group
- According to step, its TriggerGate Axis gets configured
  - Darkfield: disabled
  - Flatfield: master of camera
  - Tomo: slave of gonio, master of camera

BL31 Faxtor Tomography UI