Sardana continuous scan next challenges

Jordi Aguilar
Jose Centeno
Roberto Homs
Zbigniew Reszela
Xavi Serra
Steven Wohl
Oriol Vallcorba

on behalf of
ALBA Synchrotron
Controls Section

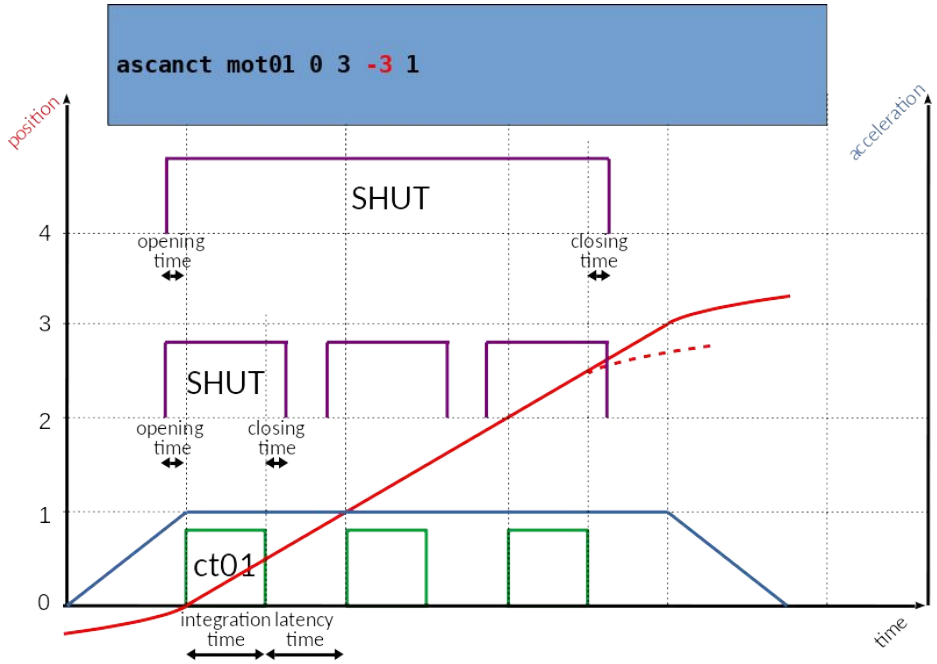
Vanessa Silva
Johan Forsberg

on behalf of
MAXIV Synchrotron
Software

Major issues

Allow different synchronization descriptions e.g. involving shutter control

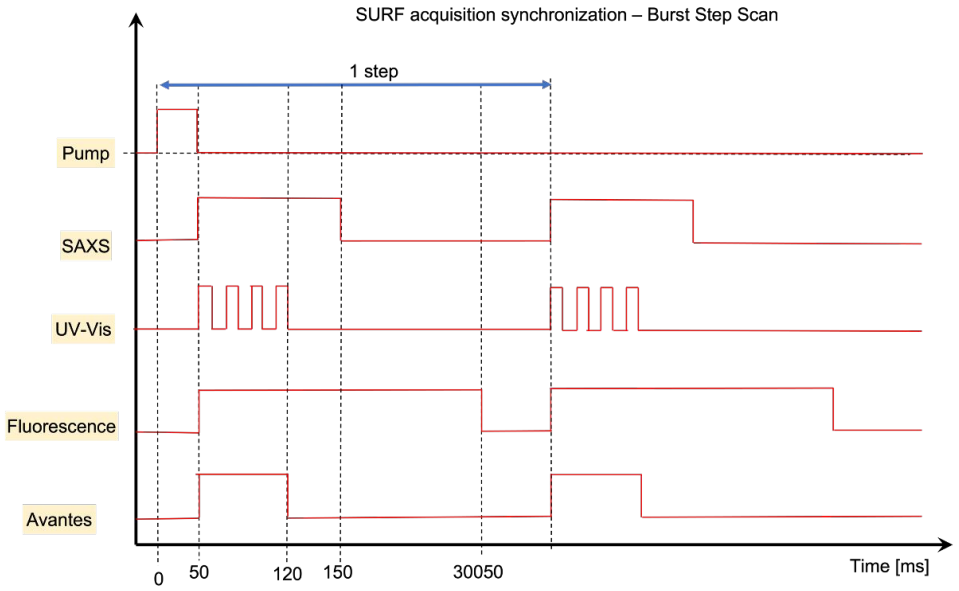
- Currently, all synchronizers receives the same synchronization description
- Usually shutter controlled acquisition require a different synchronization description for the detector and for the shutter.
- Workaround: develop specific TriggerGateController.



Element	Synchronizer	Synchronization
ct01	tg01	Trigger
shut	tg02	Gate

Allow different synchronization descriptions e.g. multiple techniques experiment

- Multiple techniques running in parallel
  - different equipments measuring different types of signals require specific exposure time and number of triggers
- Current solution
  - personalized macro to set all parameters for each technique
  - macro prepare all channels with respective configuration
  - parameters on PrepareOne are ignored
  - each equipment writes its own file
  - fake references to match expected shape by OutputRecorder



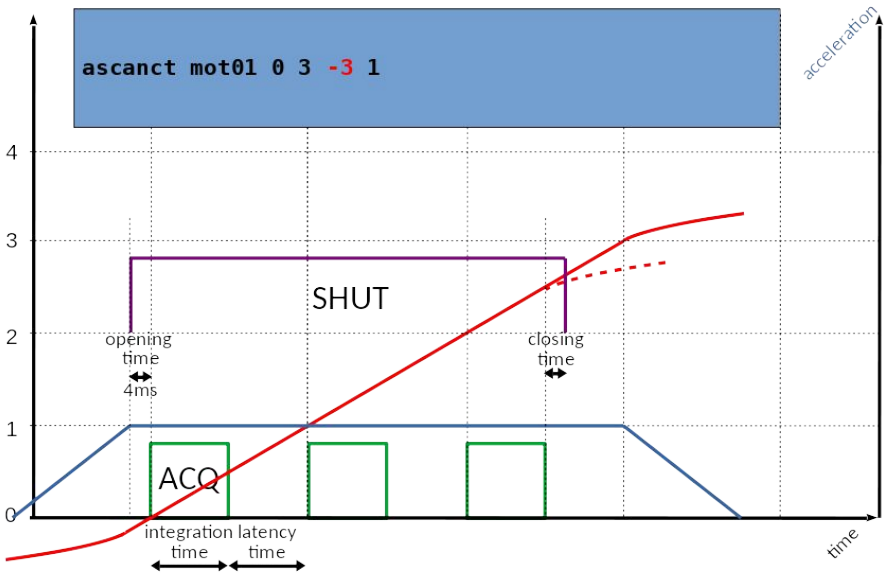
Diffractometer Sample-scan in BL06 - XAIRA: Detector & Fast-shutter

Requirements:
- Trigger detector by position
- Trigger fast-shutter at least 4ms before the detector
- Similar requirements at BL31-FAXTOR

Aerotech A3200 Constraints:
- Single-channel incremental-encoder counter in Aerotech A3200
  - Must trigger fast-shutter by software: 1KHz clock in Aerobasic operations

Suggestion:
- Trigger the fast-shutter immediately after acceleration, when already at desired scan velocity
  - Must adapt the motion start position further back than just for the acceleration profile

Sardana Solution:
- Dedicated A3200 TriggerGateController (+aerobasic for configuration) to trigger detector by position
- Added feature in A3200 MotorController (+aerobasic for monitoring accel. profile) to trigger fast-shutter
  - General scan-hooks to enable this feature for an *ascanct* macro



How to deal with the extra pre-start displacement range?

Event driven measurement

Requirements:

- Acquire data by an external event

Proposal

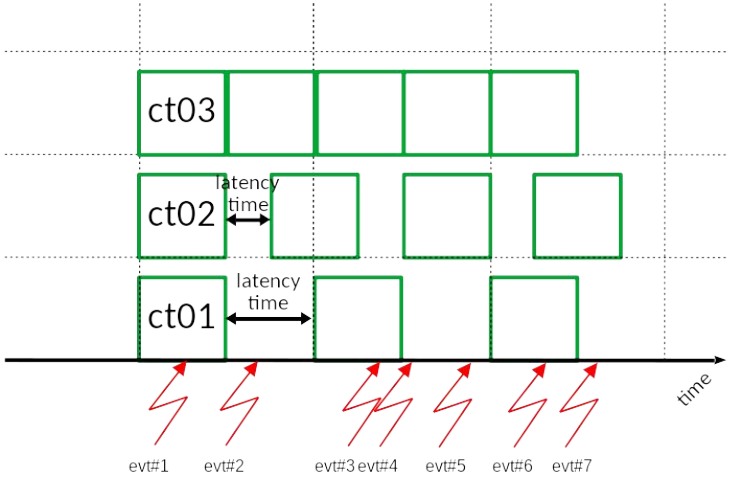
- Timestamp events and values

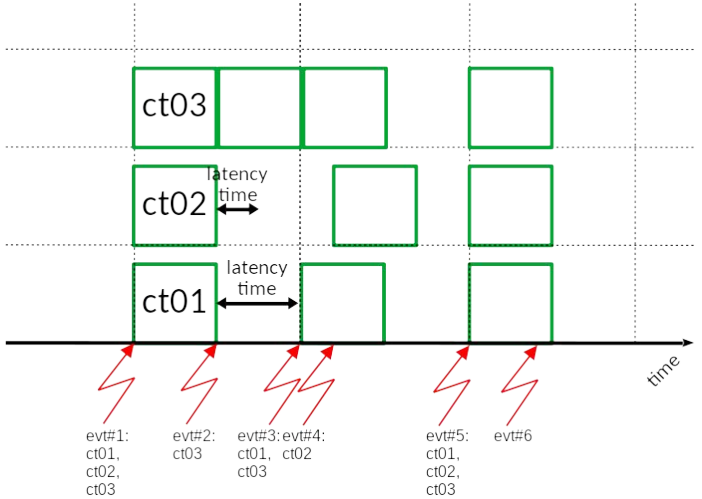
#1:

- Continuous acquisition (as fast as you can and independently for each channel)
- (After acquisition: discard values not corresponding to events and interpolate values)

#2:

- Trigger channels by an external event

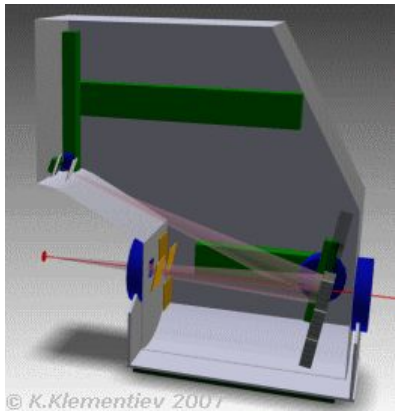


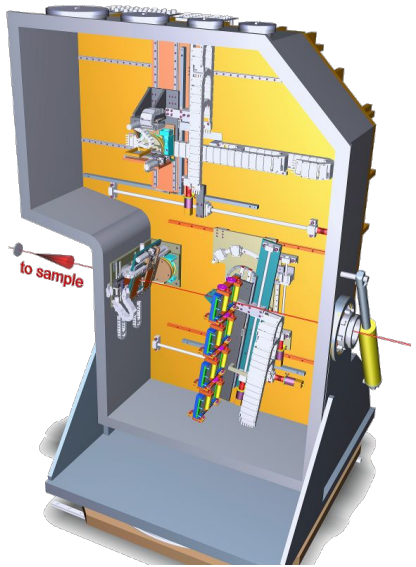


Motivations to implement trajectory in BL22-Claess

- **Avoid collisions**: detector and analyser between them and the wall
- **Increase scan velocity**: use continuous scan instead of step scan (from 45min to ~ 3min)

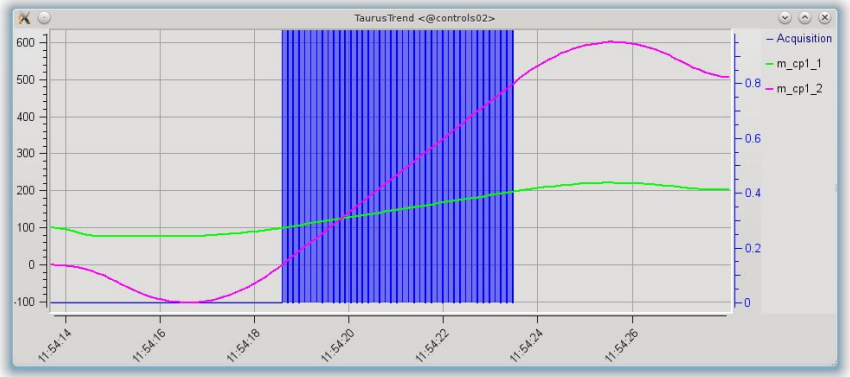
CLEAR Spectrometer
Core Level Emission
Analyzer and
Reflectometer





Pseudo-motors limitations

- Do not implement velocity and acceleration
- Sardana continuous scan framework implements constant velocity for all physical motors of one pseudo-motor, does not follow the trajectory.
- Pseudo-motors do not implement movement logic, only calculation.



Trajectory Motor Controller

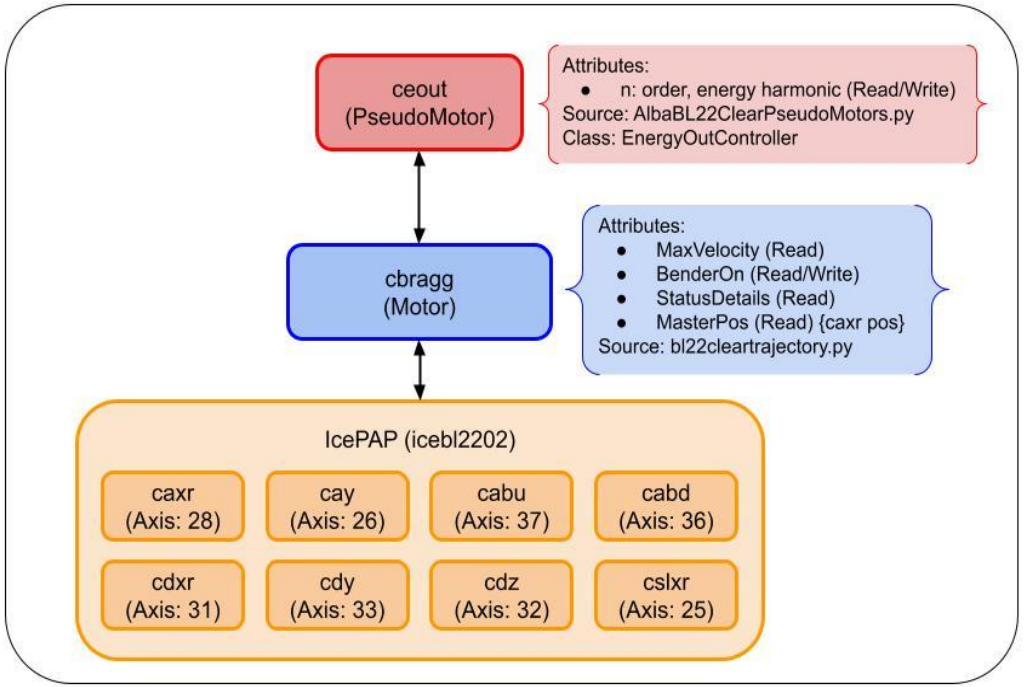
Pros:

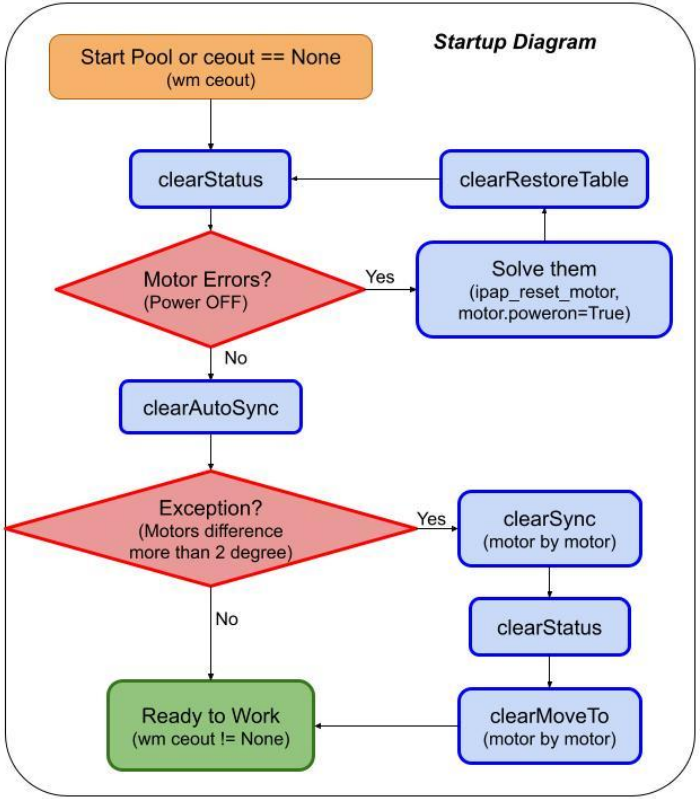
- Movement logic, velocity and acceleration control.
- Compatible with continuous scan framework.

Cons:

- **Complex**
- Need macros to configure/load/calculate trajectory tables
- Leading to inconsistencies in element states: there are two controller for the same icepap axes, trajectory and normal motors

Trajectory Motor Controller





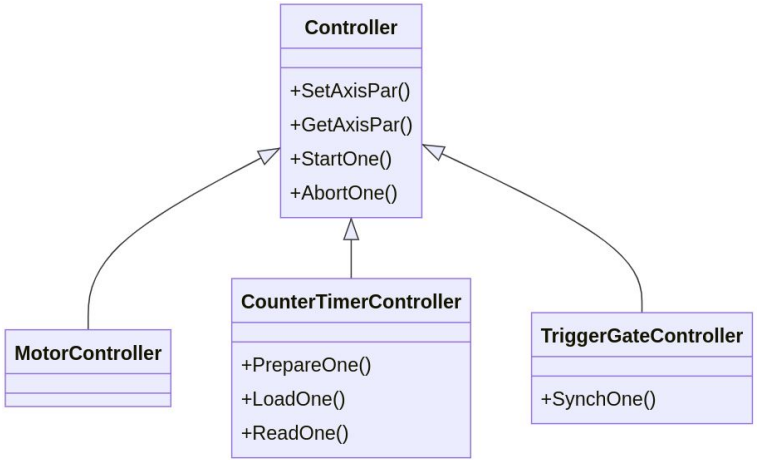
Other trajectory cases

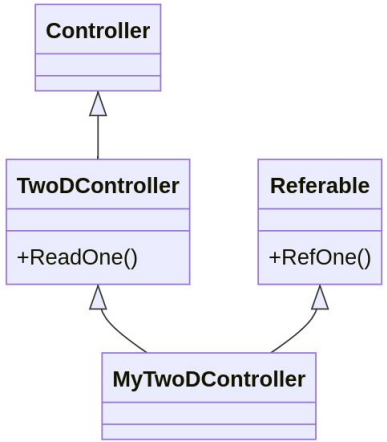
- Scan the Energy on:
  - Double Crystal Monochromator
  - Insertions Devices

- Helical scan on crystallography beamlines

Current controller plugin design for multipurpose hardware

- Sardana original design was focused on one-purpose controller plugins.
- Standard interface attributes are implemented with Set/GetAxisPar() methods.
  - Non available features (attributes) could be not exposed by removing them with the GetAxisAttributes().
- Optimized integration of 1D/2D detectors was implemented using *mixin* with the Referable class.

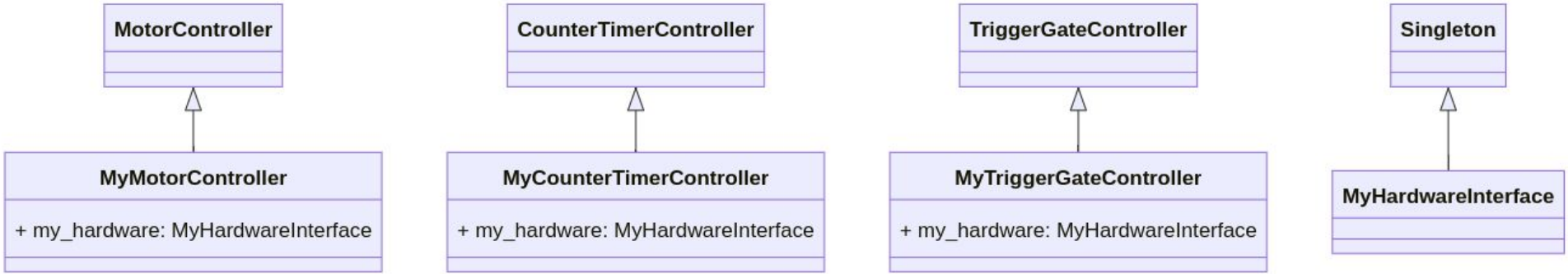




Current controller plugin design - limitations#1.

Versatile hardware controllers like powerful motor controllers may require 3 controller instances: Motor, CounterTimer and TriggerGate.

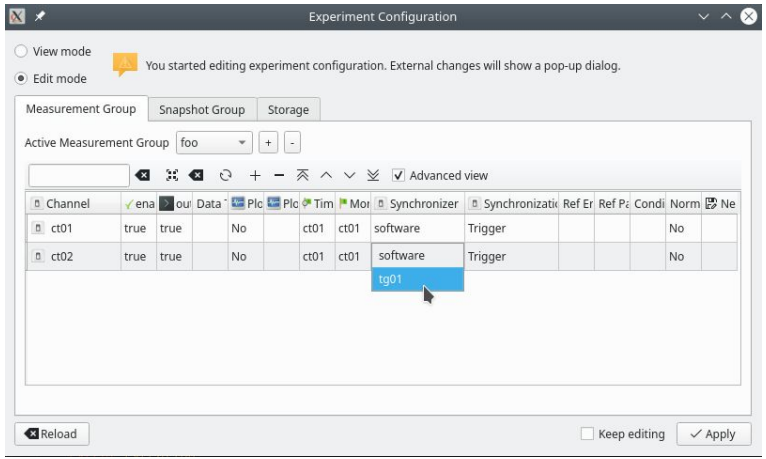
- Each controller instance will have its own element instance e.g mot01, mot01_ct, mot01_tg, some configuration e.g. step_per_unit will be available only to one controller instance.
- Each controller instance will have its own communication interface with the hardware.
- Both of these could be workaround by the use of the Singleton and Factory patterns.



Current controller plugin design - limitations#2.

Implementing of the hardware synchronization by the experimental channel controller is not *announced* to the system.

- In the expconf widget you could configure a channel to act on hardware synchronization even if the plugin does not implement that.
- Controller implementation requires a boilerplate “if…else” code depending on the selected synchronization.




```python
def ReadOne(self, axis):
    if self._synchronization == AcqSynch.SoftwareTrigger:
        if not self._new_data:
            raise Exception("Acquisition did not finish correctly. Adlink "
                            "State %r" % self._hw_state)
        return SardanaValue(self.dataBuff[axis][0])

    elif self._synchronization == AcqSynch.HardwareTrigger:
        if not self._new_data:
            return []
        else:
            return self.dataBuff[axis]
    else:
        raise Exception("Unknown synchronization mode.")
```

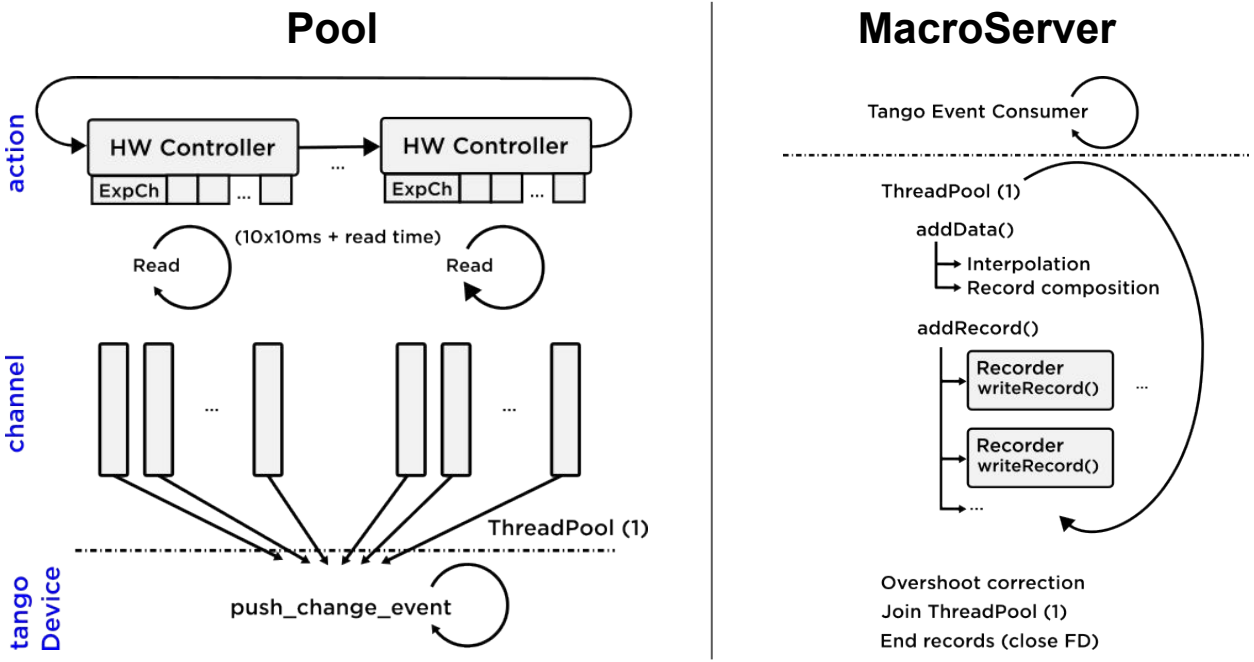
Controller plugin design: Ideas for future improvements - brainstorming.

- Probably the Singleton and Factory patterns are still the best for integrating versatile hardware controllers into Sardana
  - Providing some kind of MetaController class which would be a composition of the controller type classes would probably lead to similar kind of code in terms of amount and complexity.
  - We can not unify motor, counter/timer and trigger/gate into one element, they have different states e.g. when motion is done the position capture buffers may still need to be emptied…

- Make controller classes a composition of *capabilities* (inspired on LImA)
  - Experimental channels: Referable, HardwareSynchronized
  - Motor: ClosedLoop, Trajectory, PowerOn?

Improve support for high speed scans

Historically, we have experienced dead times in continuous scans that occur after the scan execution, just before the macro's conclusion. We suspected these delays could be due to factors like data storage delays, event accumulation on the client side, or data processing on the MacroServer side.



Improve support for high speed scans

CS tests on BLs

Beamline	Ctrl (nr ExpChan)	Scan	Dead time*
BL16	NI-Pos (1) AlbaEM2 (4) Lima2D (1) LimaROIs (26) tango_attr_ctrl (1)	4000 points 5ms integ time output + spec + h5	160ms dead time
BL04	Ni-Ct (19) Ni-Pos (2) Dummy (1)	120000 points 1ms integ time output + spec	180ms dead time
BL22	Adlink (5) NI-ct (13) tango_attr_ctrl (1)	10000 points 5ms integ time output + spec + h5	30s dead time 230ms dead time

Not significant dead times for current applications

* Time between mg.Acquisition.HardwareAcquisition end and spock prompt recovery (no overshoot/postprocessing)

Improve support for high speed scans

Recently, taking profit of the development of a Redis recorder to publish scan data and metadata, we have prepared PoC that involves directly pushing the data to a Redis DB on the Pool side (instead of relying on Tango events to traverse through the MacroServer and the recorders).

The objective is to minimize the stress on the MacroServer and decouple data composition and storage from the data acquisition process. We saw that tango events were not a problem and to avoid file recording issues a Redis recorder should be enough.

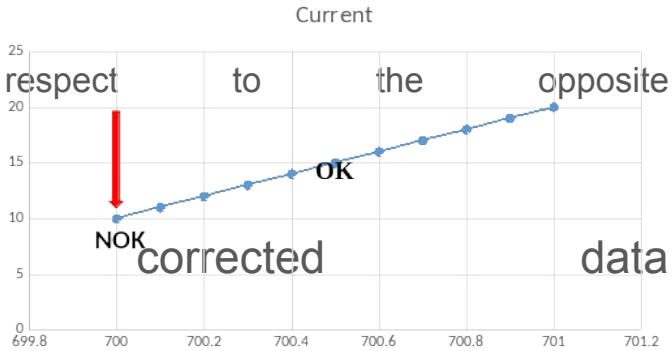
Other motivation for in-memory DB data publishing:

- Fast measurements with different synchronization definitions may cause issues with recomposing the data of different shapes (output recorder, online plotting,...). Some solutions involve file recording + republishing (+ subscription to individual equipment). Data recomposition is done now in the MacroServer level so something like Redis could be a common layer for intermediate publishing where the recomposition of the data could be performed in an efficient way.

Minor issues

Theoretical position of the record

- E.g. Read the Current in a continuous scan of the Mono Energy: `ascanct Energy 700 800 100 0.1`
- The Current is integrated during full intervals, e.g. between 700 eV and 701 eV, but the integrated value is associated to 700 eV.
- Middle of the integration interval should be used.
  - No half interval shift with respect to step scan.
  - No full interval shift with direction scan: ascanct Energy 800 700 100 0.1.
  - Here latency time should be taken into account.
- Workaround: add a with addCustomData() - ascanct_midtrigger



respect to the opposite corrected data

Latency time of trigger/gate controller

- Latency time of experimental channel controllers is considered when calculating the synchronization description.
- Latency time can also be adjusted manually as scan macro parameter.
- Currently there is no way to indicate the latency time of the trigger/gate controllers

```
class SpringfieldCounterTimerController(CounterTimerController):

    def GetCtrlPar(self, name):
        if name == "latency_time":
            return self.springfield.GetLatencyTime()
```

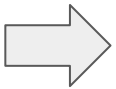
```
def SynchOne(self, axis, configuration):

    if passive < self.min_time:
        channel.write_attribute("LowTime", self.min_time)
        self._log.warning("Changing passive time to the ni660x minimum")
    else:
        channel.write_attribute("LowTime", passive)
```

Chain synchronization configuration

- Currently it is not possible to configure the main - *secondaries* relation between the synchronizers.
- Current workaround:
  - Add a *dummy* counter/timer as a placeholder for the main synchronizer - no visible relation between these two.
  - Set secondary mode to other synchronizers in the pre-acq hook.
- Possible implementations:
  - slave trigger/gate as a new entry in the expconf table
  - use tree instead of table

Channel	Synchronizer	Synchronization
ct01	slavetg	Trigger
placeholder	mastertg	Trigger

```
mntgrp01:
  channels:
  - ct01:
      synchronizer: slavetg
  - placeholder:
      synchronizer: mastertg
```



Element	Synchronizer	Synchronization
ct01	slavetg	Trigger
slavetg	mastertg	Trigger
mastertg	n/a	n/a

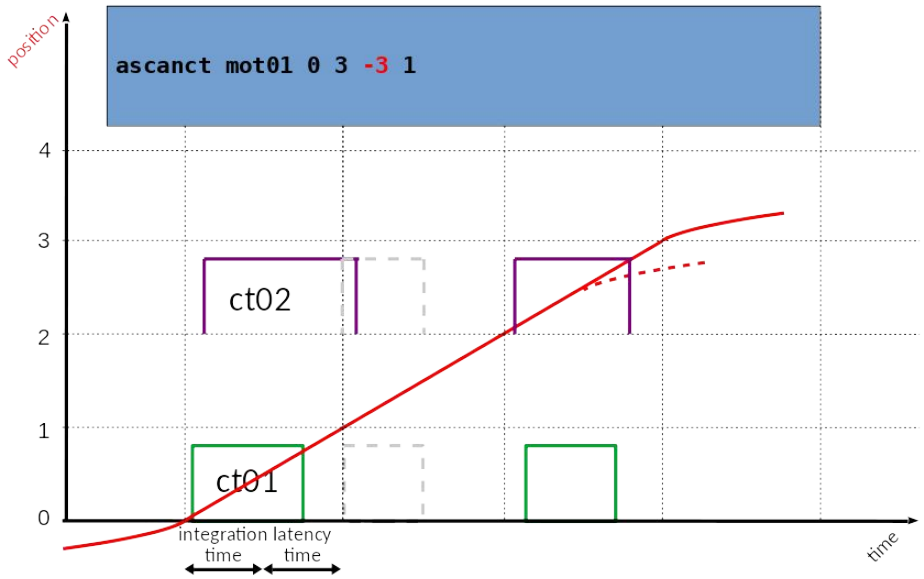
Facilitate measurement group configuration

- Channels added to the measurement group have some default configuration e.g. synchronized by software, no plotting, first added channel as timer, etc.
- Allowing to customize the default configuration could be appreciated by users and could save from errors.
  - default configuration could be associated to the element e.g. as an attribute, we already have one: timer for the *experimental channel acquisition*

```
ct02:
  axis: 1
  attributes:
    timer: ct01
    synchronizer: tg01
    synchronization: Trigger
```

Reduce dead time in software synchronization

- Currently all software synchronized channels are triggered to start acquisition at the same time.
- While any of them is still involved in the acquisition a new synchronization event is ignored.
- Optional data interpolation is available.
- Ideas for improvements:
  - Make preparation as soon as the previous acquisition finished.
  - Decouple channels.



Add timeout for motion when there is a problem with never ending motion

- Motion may also hang - rather strange e.g. due to the too low velocity (could be easily solved by setting a velocity limit).
- Calculate motion time and set a timeout (with some configurable tolerance).

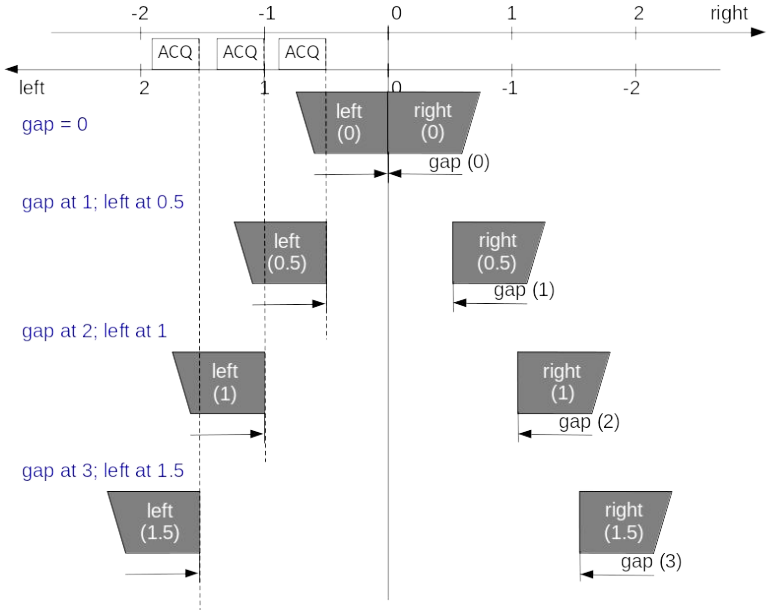
```
state, _ = motion.move(final_pos, timeout=estimated_motion_time)
if state in (Alarm, Fault):
    self.dump_information(motion.moveable_list, "moving to waypoint/end position")
    raise RuntimeError(
        "Moving to waypoint/end position ended with %s\n" % state.name.capitalize())
```

Early timeout for lack of data

- Disconnected hardware synchronizers and experimental channels, or wrong synchronization configuration can cause the acquisition to never start.
- Timeout could save some time and improve user experience:
  - Should be configurable based on the fastest experimental channel reporting data.
  - Possible implementation with a Timer thread started just before the motion and checking if any data arrived on the first trigger time + timeout.
- Eventually clear feedback messages should indicate that we just wait for data.

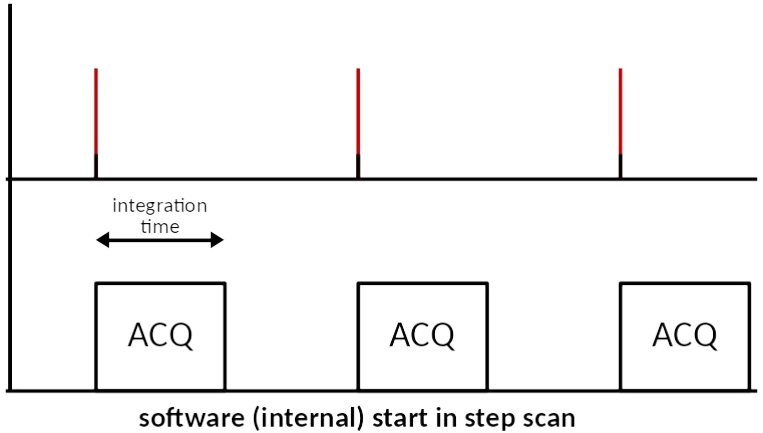
How to select the master motor (the one used for synchronization purposes) in scans of pseudomotors?

- Synchronization description in the position domain is based on one physical motor.
- In case of scanning a pseudo motor e.g. a gap of a pair of slit blades, we need to specify which of two motors should be used for that purpose.
- Configuration point could a pseudo motor attribute.



Step scans with Start synchronization

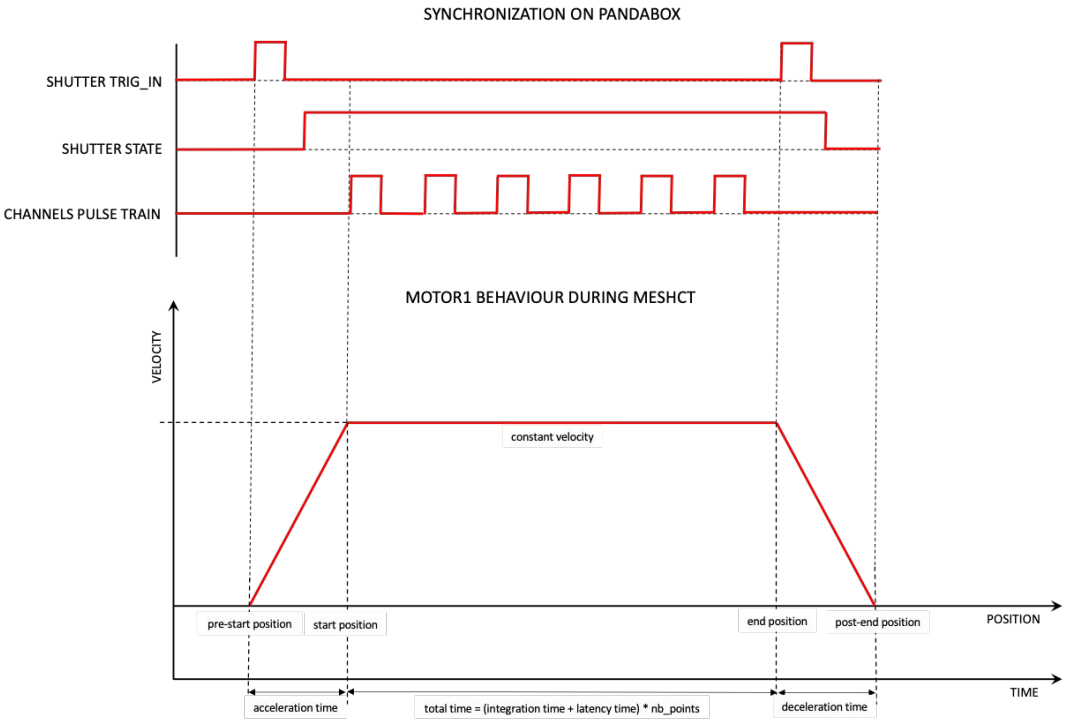
- Start synchronization is actually compatible with step scans.
- It is simply equivalent to Trigger synchronization.
- Example of step scan:
  5 acquisitions x 0.1 s of integ. time



software (internal) start in step scan

```
PrepareOne(axis=1, integ_time=0.1, repetitions=1, latency_time=0, nb_starts=3)
for acquisition in range(3):
    LoadOne(axis=1, integ_time=0.1, repetitions=1, latency_time0)
    StartOne(1)
```

meshct - active time calculation



In the current implementation

$$\text{flyscan\_time} = nb\_points * (int\_time + latency\_time)$$

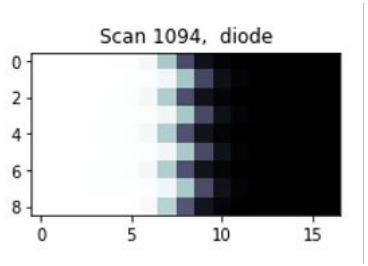
When running time-based scans:

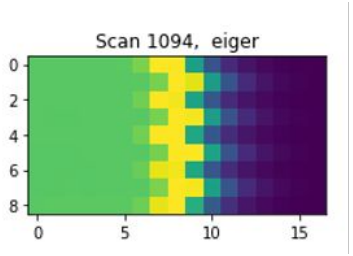
- trigger rising edge matches start position
- trigger falling edge happens before end position
- if snake scan is performed, the trigger edges don't match in both directions

meshct - active time calculation

In the current implementation

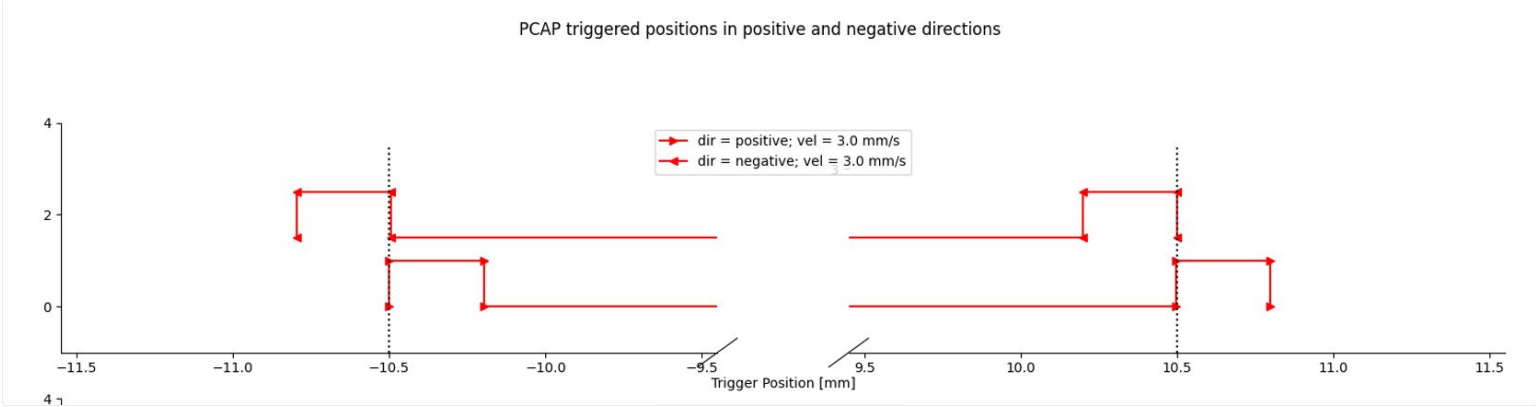
$$\text{flyscan\_time} = nb\_points * (int\_time + latency\_time)$$



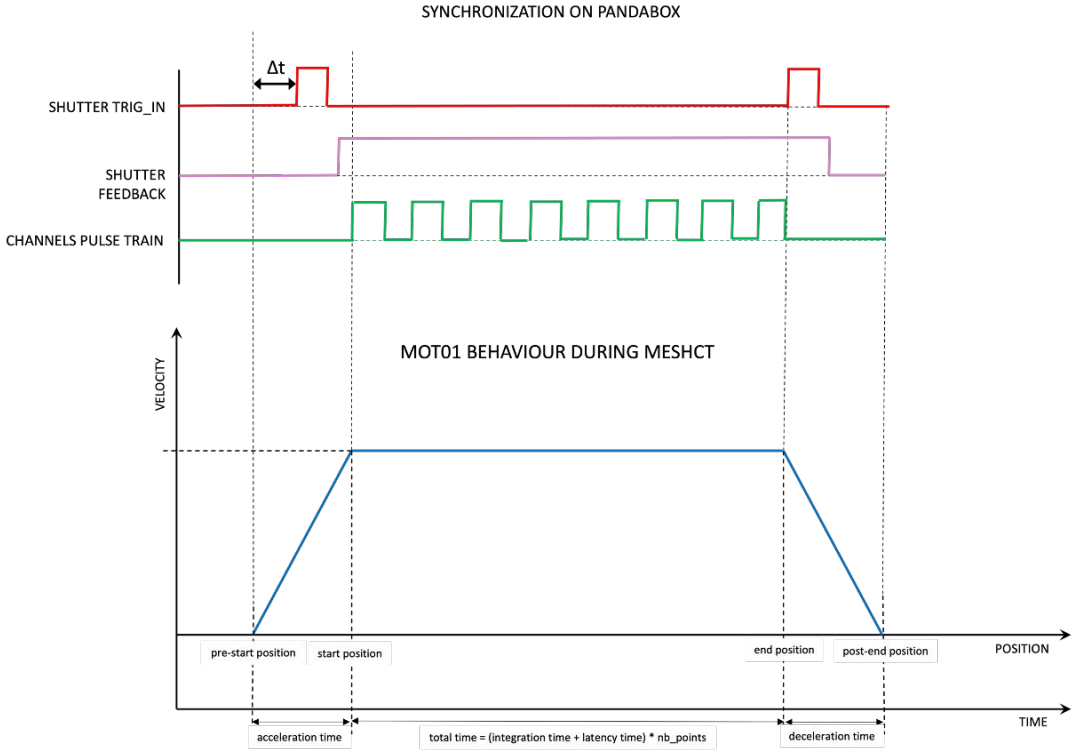


When running time-based scans:

- trigger rising edge matches start position
- trigger falling edge happens before end position
- if snake scan is performed, the trigger edges don't match in both directions

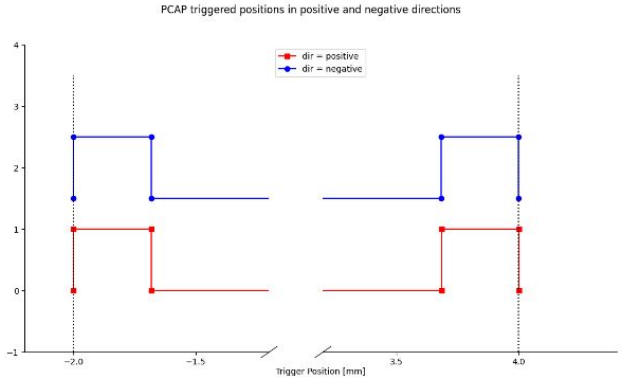


meshct - active time calculation proposal




```python
def _generator(self):
    """Overrides meshct _generator to adjust active_time."""
    for step in super()._generator():
        step["active_time"] = (
            self.nb_points * self.integ_time
            + (self.nb_points - 1) * self.latency_time
        )
        yield step
```

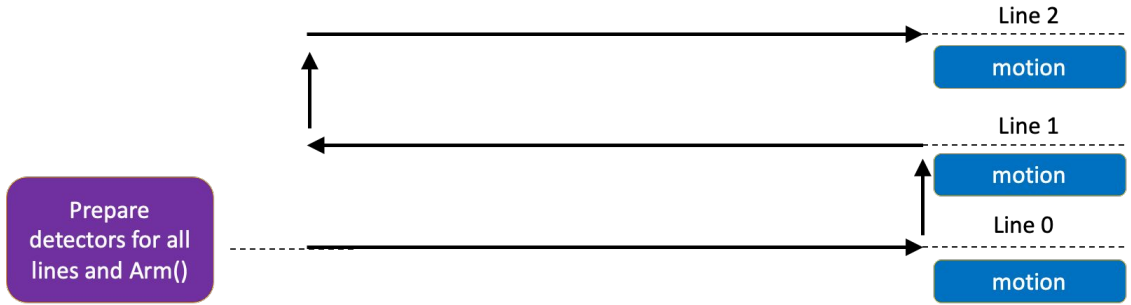
Time-based



meshct - improve overhead time between lines

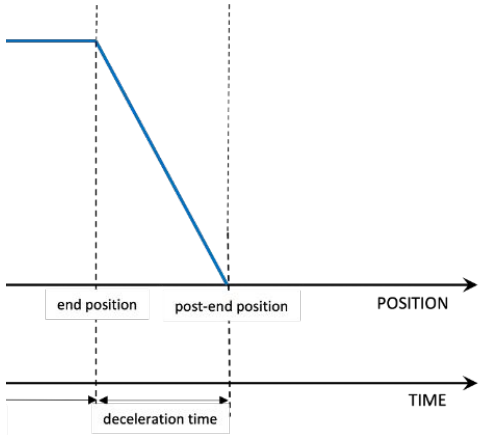
Proposal 1: prepare measurement group only once, in the beginning of Line 0

- prepare for all lines at once
- possibility to Arm detectors only once



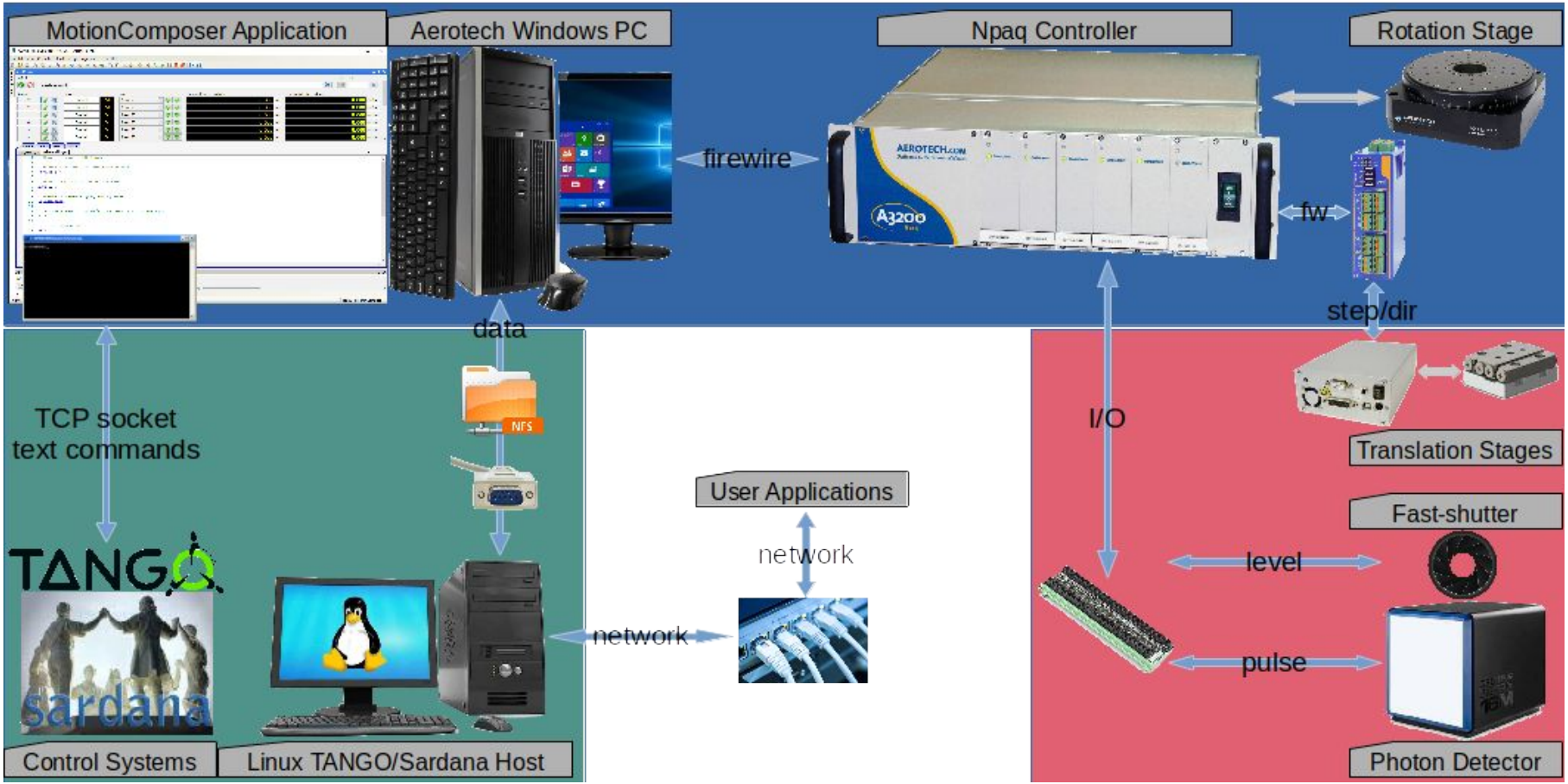
meshct - snake	meshct prepare once - snake
2.32 seconds	0.73 seconds

Proposal 2: between the lines the main action happening is motion [if detectors can be armed only once]. Improve motion time with simultaneous motion: start to move step scan motor while flyscan motor is decelerating



Thank you for your attention!

ALBA XAIRA Diffractometer: Aerotech A3200 Npaq (+SmarAct SDC2)



Diffractometer Sample-scan in BL06 - XAIRA: Detector & Fast-shutter

Requirements:

- Trigger detector by position, within ??? deg error-margin; velocities range from 1deg/s to 360deg/s
- Trigger fast-shutter at least 4ms before the detector

Aerotech A3200 Constraints:

- Single-channel incremental-encoder counter in Aerotech A3200
  - Must trigger fast-shutter by software: 1KHz clock in Aerobasic operations
- 16MHz encoder quadrature frequency: must reduce encoder resolution by 5, to 40'000counts/deg

Proposal:

- Trigger the fast-shutter immediately after acceleration, when already at desired scan velocity
  - Must adapt the motion start position further back than just for the acceleration profile

Sardana Solution:

- Dedicated A3200 TriggerGateController (+aerobasic for configuration) to trigger detector by position
- Added feature in A3200 MotorController (+aerobasic for monitoring accel. profile) to trigger fast-shutter
  - General scan-hooks to enable this feature for an *ascanct* macro

Measured Results:

- Detector trigger jitter: 0.000259deg
- Shutter timing: between -4.6ms and -5.1ms (dependent on scan velocity)