# 1.08 Control/DAQ Status

**Laura Newburgh**

CMB-S4 Collaboration Meeting
April 3-6, 2023

# Outline

- Subsystem Team
- Scope and adaptation to AoA
- L3 Progress
  - Controls
  - Acquisition
  - Monitoring
  - Lab Support
- Near-term plans
- Summary

# 1.08 DAQ/Control Team

**1.08**
**DATA ACQUISITION AND CONTROLS**

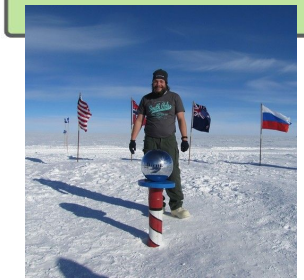
Laura Newburgh (Yale)
Gregg Thayer (SLAC)

**1.08.01**
**DAQ Management**
Gregg Thayer (SLAC)

**1.08.02**
**Observatory Control System**
Brian Koopman (Yale)

**1.08.03**
**Observatory DAQ**
Nathan Whitehorn (MSU)

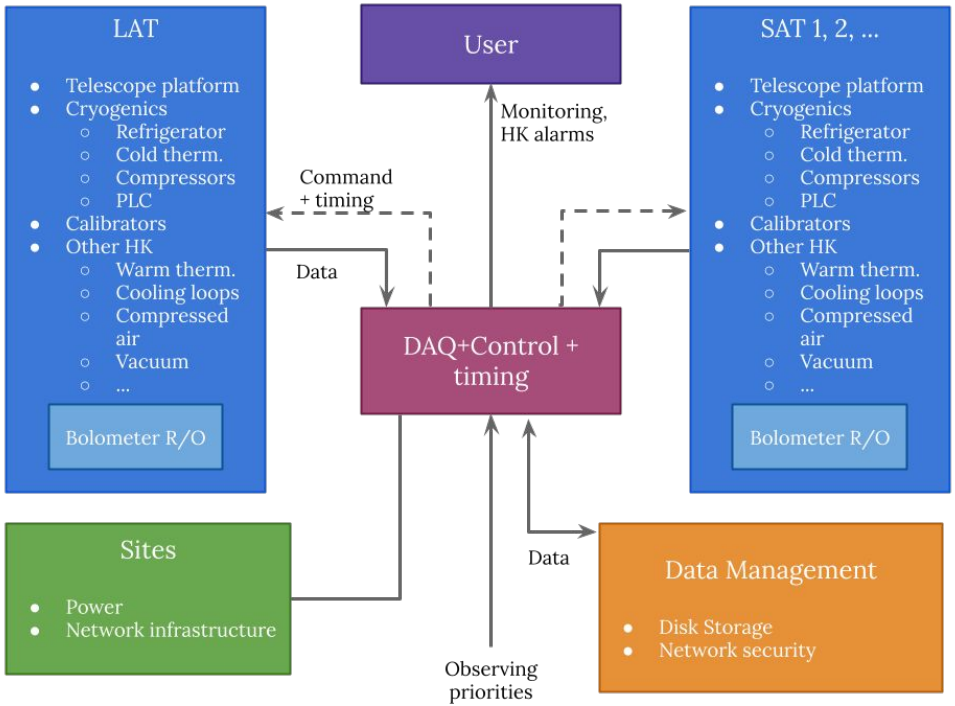
**1.08.04**
**Monitoring and Alarms**
Cosmin Deaconu (U. Chicago)



**1.08.05**
**Lab Subsystem Development and Support**
Abigail Crites (Cornell)

# 1.08 Scope and AoA

- Acquire and aggregate high-speed data from detectors, slow data from housekeeping and meta-data into specified format, hand-off to data management
- Provide control framework for commanding equipment in the labs, and at the observatories
- Browser-based live and historical monitoring of ‘housekeeping’ data and meta-data
- Hierarchical, **non-safety** alarms system based on housekeeping data
- Provide observatory-wide timing and frequency references to readout and telescope control hardware
- Provide support for running DAQ and Controls systems in development labs
- AoA only changes the scope of deployment, not development to be done



# 1.08.02 (Controls) Progress [Brian Koopman]

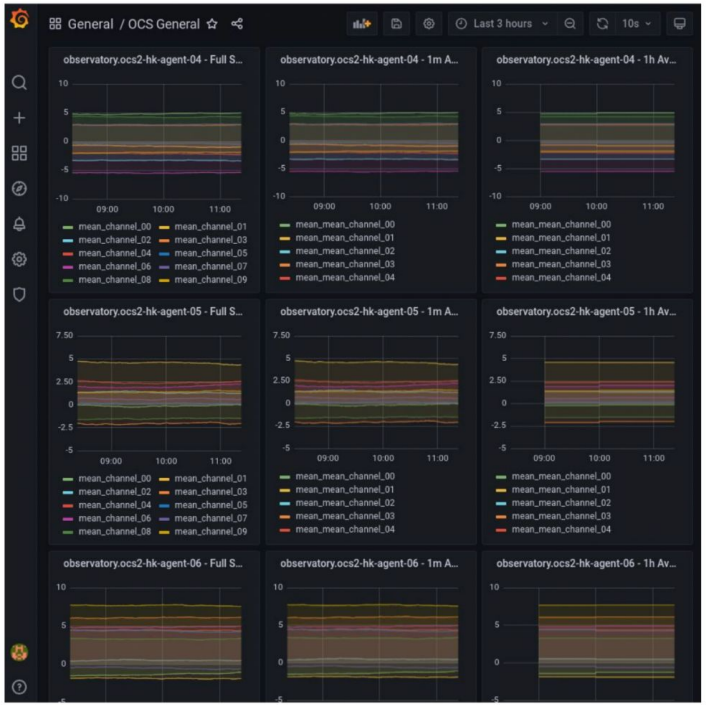
- Various upgrades to the OCS repository to make it easier for public collaboration (separate core utils from specific hardware agents)

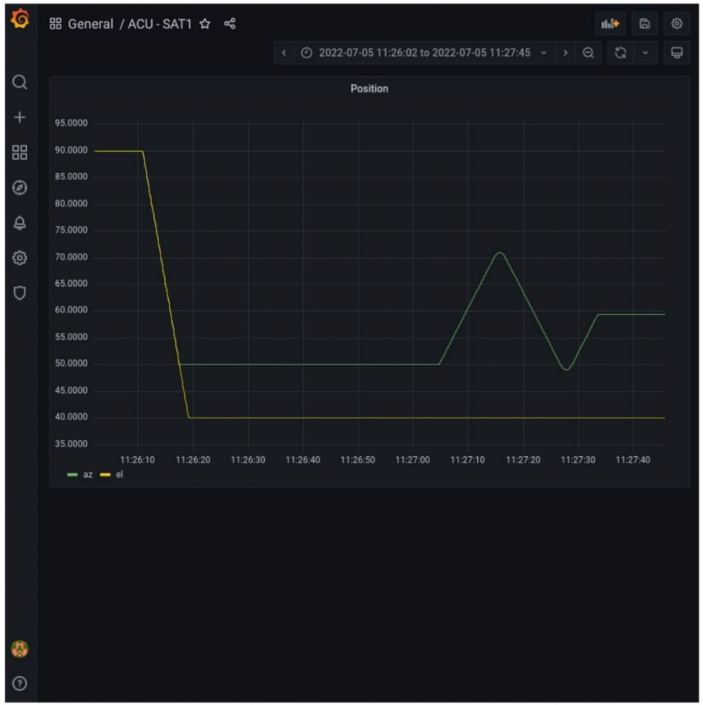
- Demonstrated large-scale testing leveraging effort for Simons Observatory:
  - In lab:
    - Many bugs worked out at UChicago testbed
    - Lab testing with full SAT receiver and partially populated LAT
    - Ongoing lab use for detector testing and next two SAT receivers
    - Demonstrated ‘user’ side of control, acquisition, monitoring

  - In emulation:
    - End-2-End testing with emulators for telescope and detector data + representative number of other ‘housekeeping/telemetry’ equipment
    - Above also exercised automated control through scheduling software
    - Demonstrated full operation (from scan to final data collation) at SO scale: verification of the architecture → This ‘trivially’ scales to a test for S4, although requires additional computation resources.

# 1.08.02 (Controls) Progress [Brian Koopman]

Fun Screen grab from the end-to-end tester (all data is simulation, producing data on the fly, shown is a set from one ‘SAT’)



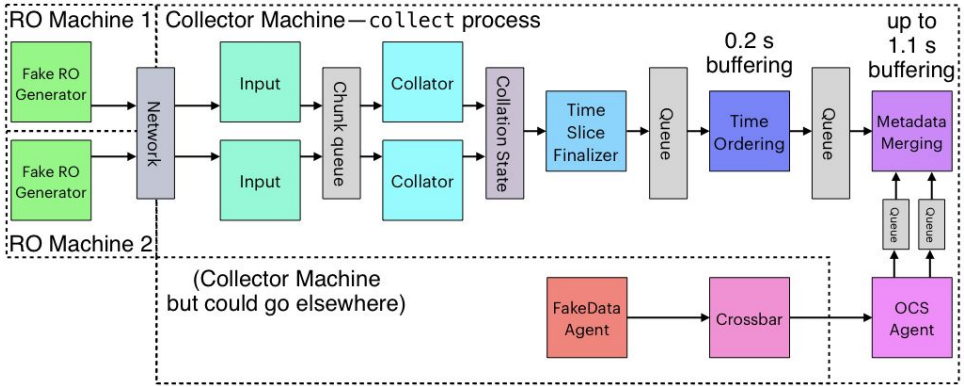
# 1.08.03 (Acquisition) Progress [Nathan Whitehorn]

- High-speed data collector prototype:
  - Accepts readout data via UDP from multiple readout machines
    - Currently, mock-up fake data generators due to lack of real equipment/specifications
  - Collates samples into coherent time slices which are output in time order
  - Can merge metadata published over crossbar/OCS by other DAQ components
  - **Operation demonstrated for 150,000 simulated detector channels read out at 400+ Hz**
  - Able to produce G3Frames as output over the network, suitable for direct ingestion by SPT3G_software

**Test Output (320 Hz)**

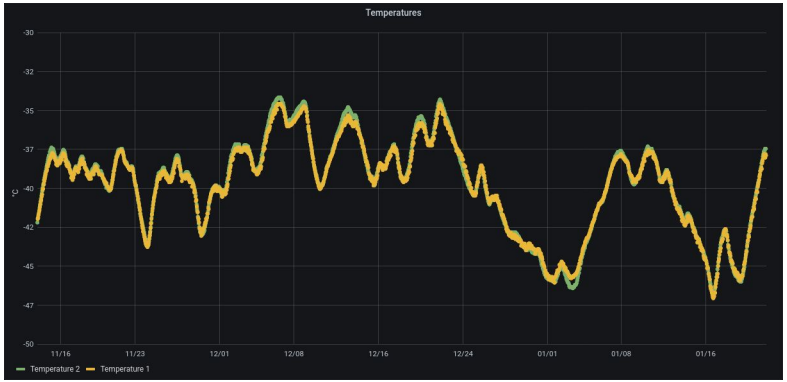
```
5785733 chunks received with 22603 syscalls
Spent 58.7333 seconds receiving, 2.06992 seconds on other work
5738129 chunks received with 22418 syscalls
Spent 58.7323 seconds receiving, 2.07103 seconds on other work
5753423 chunks received with 22477 syscalls
Spent 58.7307 seconds receiving, 2.08458 seconds on other work
5762715 chunks received with 22512 syscalls
Spent 58.7492 seconds receiving, 2.06613 seconds on other work
Acknowledged 23040000 packets using 45618 packets
Processed 11588520 chunks in 181234 batches with 416834 insert calls
Processed 11451480 chunks in 179094 batches with 412730 insert calls
Time ordering processed 19200 slices and discarded 0 which were too late
Total slice processing latency (s): 50%: 0.5051, 90%: 0.5092, 99%: 0.5101, max: 0.5141
23040000 chunks collated
Formed 19200 complete slices
Made 17239 finalization sweeps
```

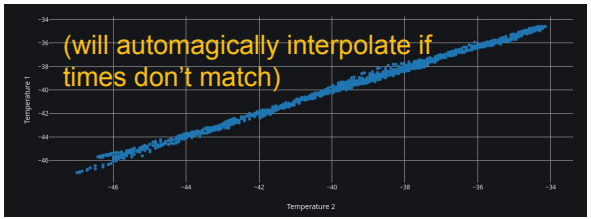
**Test Setup Schematic**



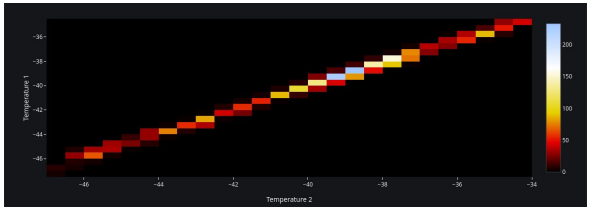
# 1.08.04 (Monitoring) Progress [Cosmin Deaconu]

- “Interposcatter” - interactively plot time-dependent variables against each other
  - Grafana, the selected housekeeping monitoring solution, mostly supports time series.
  - Proof-of-concept Grafana plugin implemented, supporting scatter plots and histograms
  - Boring example (two temperatures):




or




- Experimental support for JSON output for .g3 Frames
  - Uses same serialization framework as binary files, so supports any data
  - Eventual goal is to facilitate web-monitoring of non-housekeeping-like data and custom webviews
  - HTTP server to serve .g3 files from a directory hierarchy as json under development

# 1.08.05 (Labs and Integration) Progress [Abby Crites]

- Scope:
  - ensure the software works in the labs, and (thus) smooth transition to operations.

  - Develop software for reading in data (to be coordinated with DM)
  - AND ALSO lots of unglamorous but essential tasks like documentation, training, feedback, agent status tracking, repository cat-herding
    - Currently working on lab installation at Cornell as a test case
    - Defining training and documentation tasks
    - Developed some training documentation, need to feedback in with formal documentation

- Up next (summer ‘23):
  - Interface with L2s to determining first lab sites this will be deployed.
  - Work to build framework for reading in data (to be coordinated with DM)

# Plans for 2023

- Our goal is to deliver DAQ and OCS to labs together with warm electronics from Readout
  - Bootstrap system development with Readout teststand at SLAC
  - Funding allocations reflect this priority

- Define interface specifications
  - Fast-cadence data from Readout
  - File format definition to DM (both fast-cadence and housekeeping data)
  - Finalize data aggregation from readout system

- Once file format is established, develop data access software
  - Coordinating with DM to prevent duplication of effort and ensure compatibility

- Define control sequences for OCS readout commands (eg, in python, etc)
  - Allow “routine” operation of Readout system through OCS