

CRIS daq

CRISTAL – CRIS Tuning, Acquisition and Logging

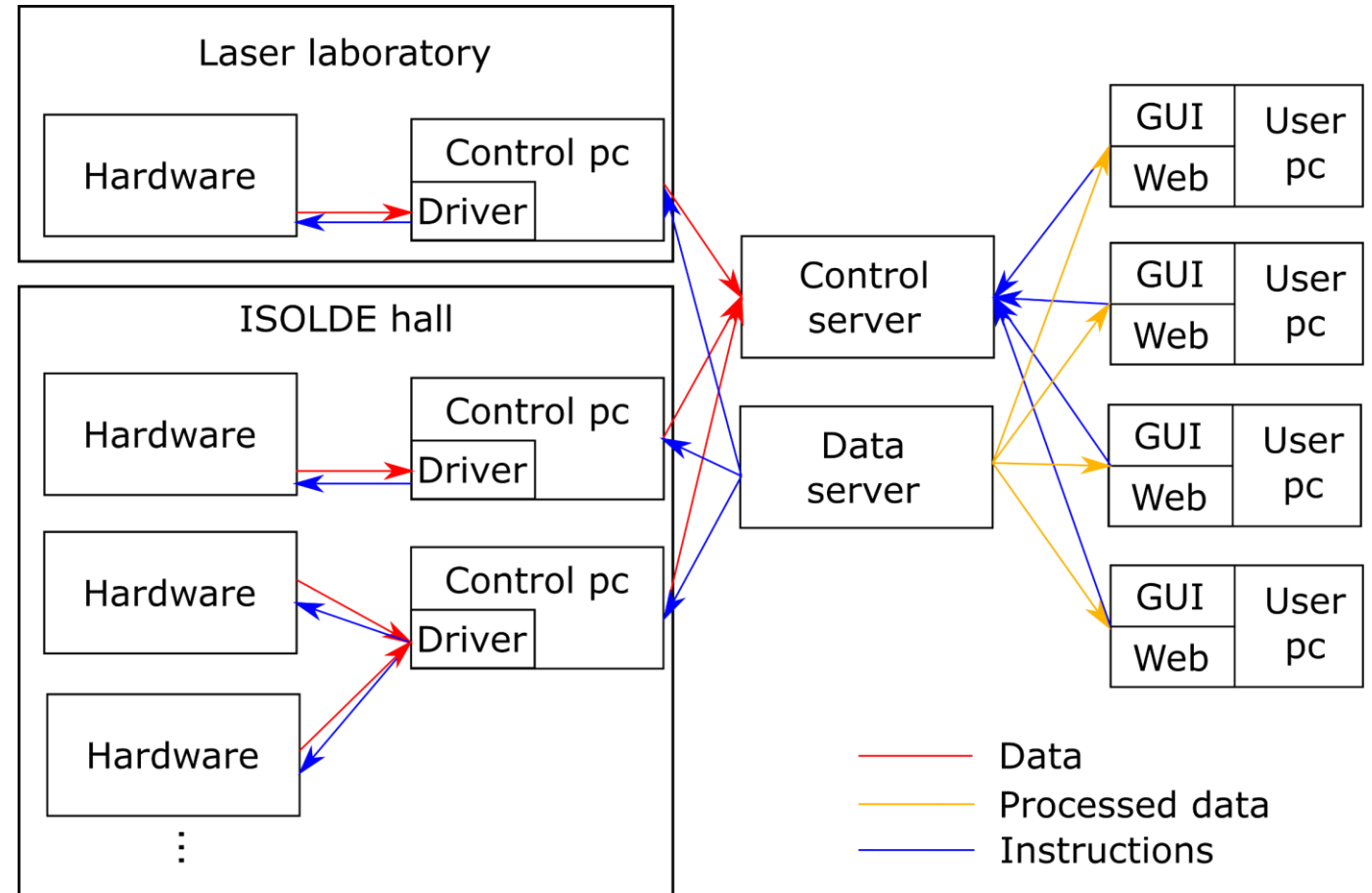
General features

CRIS collaboration signaled a need for

- a centralized control system to monitor and control hardware, to initiate measurement cycles, record data,
- That is accessible from many locations,
- Can run 24/7, 365 days a year,
- And furthermore user-friendly
- Written in an easily maintained language (Python)
 - Mostly standard library, some additional libraries for high-performance data manipulation and fast data plotting

Architecture

- Centralized control and data servers
- Communications via custom TCP-IP protocol
- Every piece of hardware is polled as fast as possible, data is pushed to the servers
 - Data server timestamps all incoming data
 - Synchronization ~ms



Architecture

- Servers summarize information and make it available to user programs
- User programs request updates periodically
- ‘Multi-user’ - servers are multi-threaded to allow many points of interaction

Idle, on setpoint

Wavenumber 1: -100.06923

ISCOOL voltage: 30213.1978846641

M2 not locked

Wavenumber 2: -100.06923

Scan number: 1966

matisse lock: N/A

Wavenumber pdl: -100.06923

Booster info: 0/30 (0 for HRS)

100%

- Graphical User Interface (GUI) programs - control hardware and monitor experiment from anywhere in CERN

Update install files

Modify config file...

Launch listener

Launch data server

Launch control server

Launch device...

Run scanner

Run data viewer

Run timing data viewer

Run beam tuning

Run cups and valves

Kill local programs

Scan and device control

Settings

Devices

Scanning

beamline	controller	Data Server	not saving scans	not saving stream	data rows: 14526911	Settings...	Remove
current	controller	Data Server	not saving scans	not saving stream	data rows: 3445832	Settings...	Remove
m2	controller	Data Server	not saving scans	not saving stream	data rows: 5906202	Settings...	Remove
wavemeter	controller	Data Server	saving scans	not saving stream	data rows: 26706069	Settings...	Remove
wavemeter_pdl	controller	Data Server	saving scans	not saving stream	data rows: 1344186	Settings...	Remove
proton	controller	Data Server	saving scans	not saving stream	data rows: 2022622	Settings...	Remove
iscool	controller	Data Server	saving scans	not saving stream	data rows: 288705	Settings...	Remove

iscool Add Device

100%

Idle, on setpoint
Wavenumber 1: -100.06923
ISCOOL voltage: 30213.1978846641

M2 not locked
Wavenumber 2: -100.06923
Scan number: 1966

matisse lock: N/A
Wavenumber pdi: -100.06923
Booster info: 0/30 (0 for HRS)

Connected to controller and Data Server

Scan and device control

Settings

Devices

Scanning

11836.0 Go to setpoint

Change laser setpoint

Define scan regions

13002.72	450	13002.755	2.3 MHz/step	Remove
13002.845	280	13002.875	3.2 MHz/step	Remove

Reverse Add

Repeats: 1

units/step: 1.0 seconds

Total Frequency: 4.7 GHz

Total time: >12 min

Start!

Back Back to home

100%

Idle, on setpoint
Wavenumber 1: -100.06923
ISCOOL voltage: 30213.2148393762

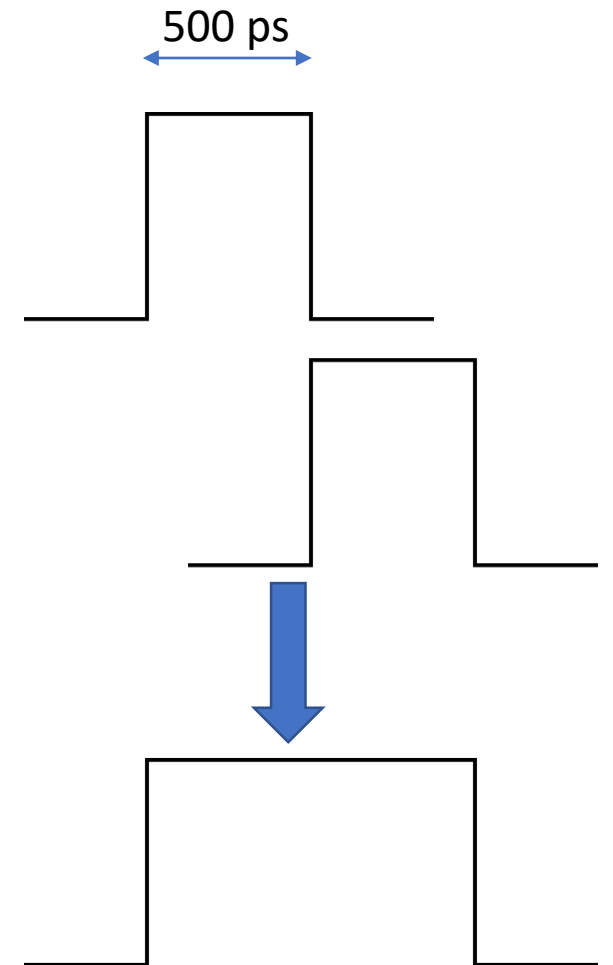
M2 not locked
Wavenumber 2: -100.06923
Scan number: 1966

matisse lock: N/A
Wavenumber pdi: -100.06923
Booster info: 0/30 (0 for HRS)

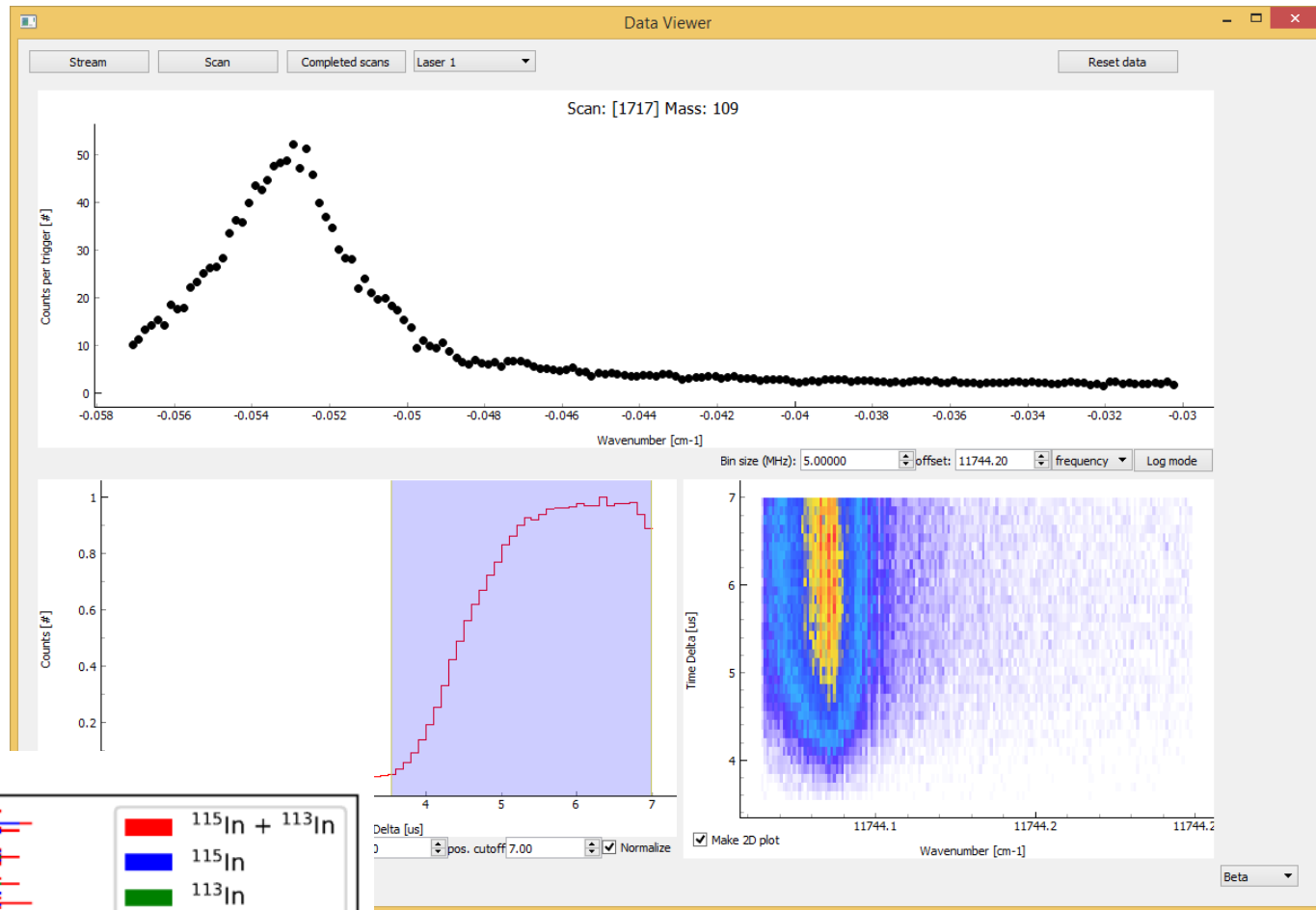
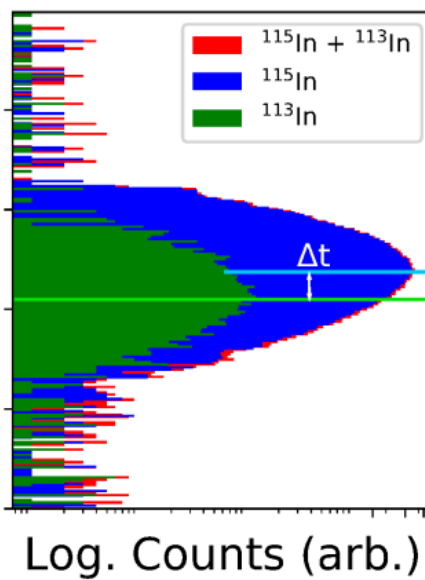
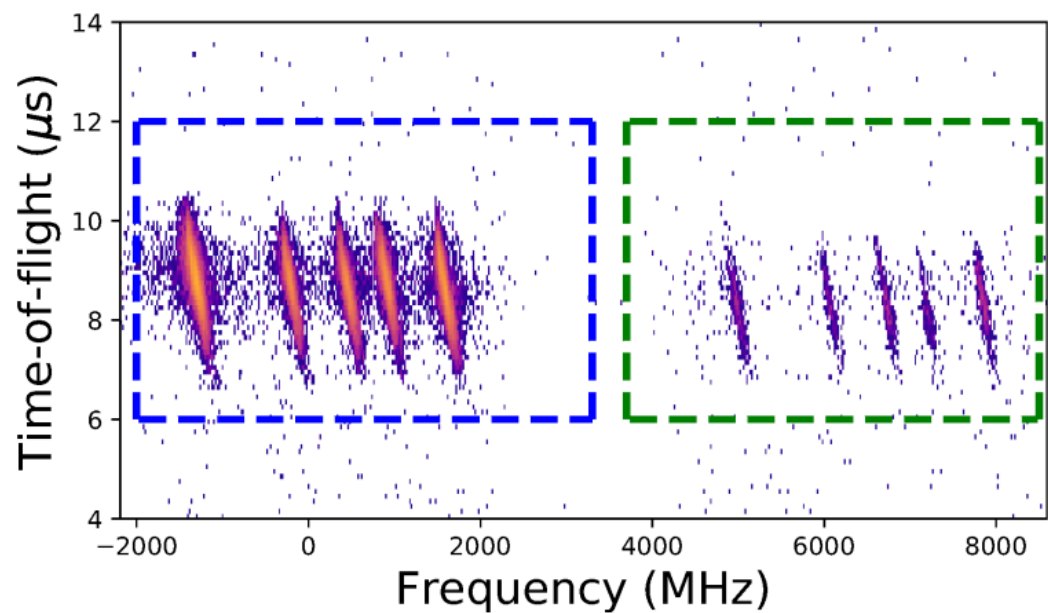
Connected to controller and Data Server

New addition in late 2017

- TDC card: timetagger 4-2G
 - 500 ps time bin
 - 'dead time': 1 ns double pulse resolution
 - 4 independent stop channels with software-controlled trigger levels
 - Up to 1000 events in a burst can be recorded
 - 2.5 kEUR



• Online view →



Lessons learnt

- Writing a full network protocol from scratch is very time-intensive
 - Program is easily expanded with simple features (e.g. adding a thermocouple readout, etc.)
 - But, I notice that changing the user-interface programs and working on the 'internals' is not something colleagues have taken up
- > I would now opt for EPICS-based networking solution for everything but high-throughput data transmission where synchronization is important
 - Experience in JYFL shows that this is a better route to take
- 'Modular approach' with many small programs that communicate is in my opinion the only scalable solution
- Note also: spending a bit more money on a piece of hardware with good driver support >>>>> having a student work hard on a cheaper product