# BELLA Center Controls System

**Anthony Gonsalves**

Lawrence Berkeley National Laboratory

LPA workshop on control systems and machine learning

January 24th 2022

**"What are the primary requirements of the future robust experimental control system?"**

BELLA

BERKELEY LAB

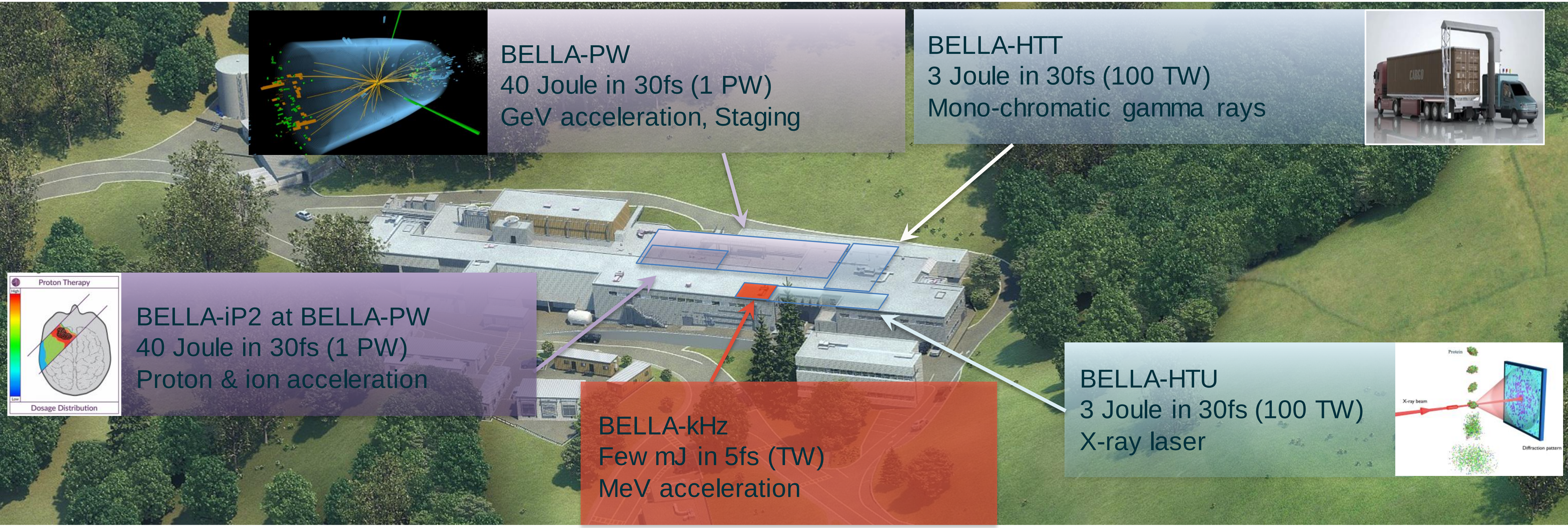
ACCELERATOR TECHNOLOGY & APPLIED PHYSICS DIVISION ATAP

U.S. DEPARTMENT OF ENERGY | Office of Science

# Overview

- **Control system requirements for the user and developer**
- Design of the BELLA Center control system - GEECS (Generalized experiment and equipment control system)
- GEECS user experience
- Integration of ML into GEECS
- The future of controls at the BELLA center

# BELLA Center houses multiple LPA facilities, each requiring a flexible & distributed control system




**Examples of what needs to be controlled**

Laser (alignment, energy…)

Targets (position, density…)

Diagnostics (plasma, electron beam…)

**Example scale of controls: BELLA-PW**

~90 computers

~300 devices

~60 cameras

>1000 process variables

≦1Tb/day data saved (1Hz)

# Key control system user features required at the BELLA Center

- Control and acquire data from all systems (e.g., laser with associated cameras, motors, etc)
  - Remote "real-time" view, analysis and control from arbitrary number of locations
- Continuously log certain parameters (e.g., temperature, laser energy)
- Perform experiments by scanning variables and saving the synchronized data a standard way, including "recipes"
  - Fire laser pulse on demand
- Repetition rate up to 10Hz (this one is now changing)
- Alarm system
- Load/Save experiment configurations
- Fully automated single click startup and shutdown
- Full system deployment, including devices, GUIs, experiment configurations etc with no programming knowledge (we have no software engineer)

# System must also be efficient for the developer

- Based on standard principles
  - ANSI/IEEE-1471-2000 (IEEE-2000a, “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems”) as described in Documenting Software Architectures: Views and Beyond by Paul Clements et al, Addison Wesley 2002.
- Efficient code reuse
  - Hardware abstraction
  - Object-oriented
- Easy to extend
- Open source
- At minimum, support data access and control to all common programming languages
- For us, developer time = scientist time. Less time spent on CS the better
  - Creating an average device driver should take a few hours not days

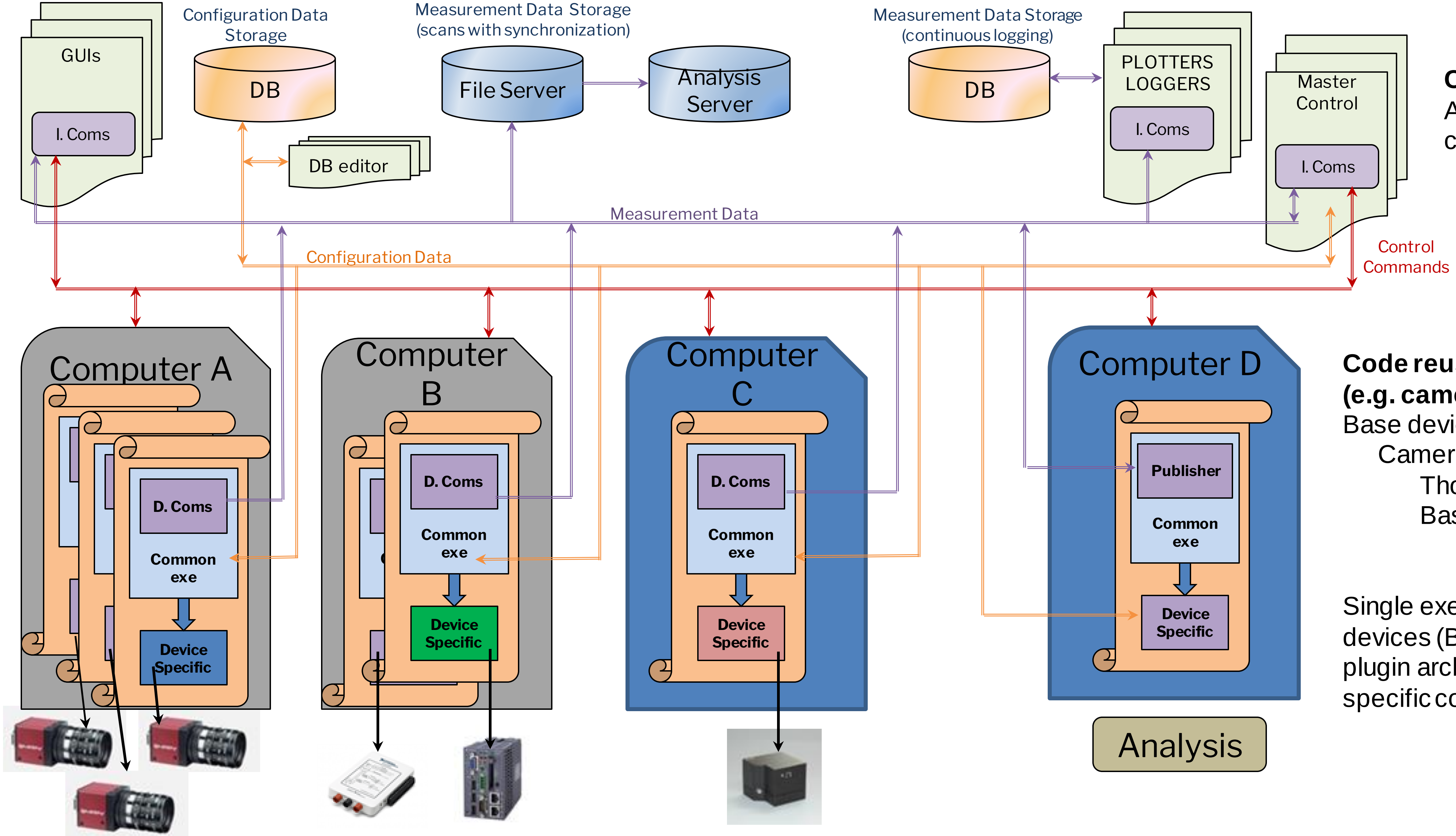
# In order to meet user and developer requirements with minimal support, we developed GEECS

- Originally, controls at the BELLA center were monolithic
  - GUI, hardware communication, data processing all in single, large programs with ad-hoc communication with a few devices on other computers. Different beamlines had their own unique controls systems with little code reuse. Classic examples of terrible LabVIEW code (but worked)
- In Jan 2011, with BELLA PW on the horizon, we decided we needed a distributed and modular control system (CS) that could be used for any beamline/experiment
  - Considered using EPICS, but development time for new features and drivers seemed long (no experience in group). With no software engineering support, and deep LabVIEW experience among the scientists, decided to build our own CS
- With 2 people 2hrs a week, the core of the control system was built in about 6 months
  - Temporarily had support from software engineer: developed logger / alarm system, expanded supported device list, and created GUIs for configuring devices and experiments
  - By the end of 2011, we had the control system up and running on TREX (for staging experiments) and the BELLA PW

# Overview

- Control system requirements for the user and developer
- **Design of the BELLA Center control system - GEECS (Generalized experiment and equipment control system)**
- GEECS user experience
- Integration of ML into GEECS
- The future of controls at the BELLA center

# GEECS framework is modular, flexible, & uses hardware abstraction to minimize coding effort




**Code reuse in GUIs:**
All GUIs have same communication layer

**Code reuse in device drivers (e.g. camera):**
Base device class
- Camera class
  - Thorlabs Camera class
  - Basler Camera class

Single executable for all devices (Base device class), plugin architecture calls device specific code

# Overview

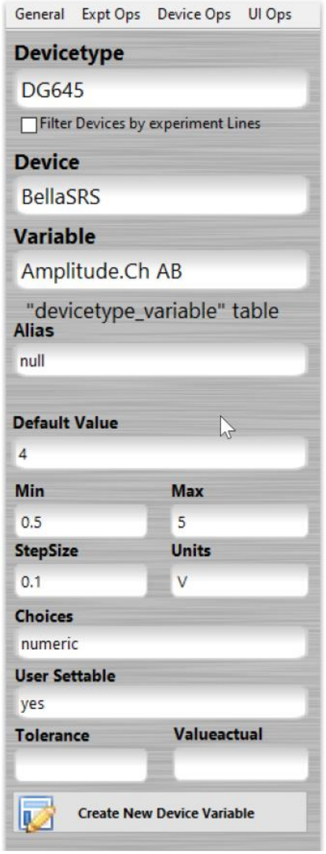
- Control system requirements for the user and developer
- Design of the BELLA Center control system - GEECS (Generalized experiment and equipment control system)
- **GEECS user experience**
- Integration of ML into GEECS
- The future of controls at the BELLA center

# Adding, configuring devices and experiments done with simple GUIs

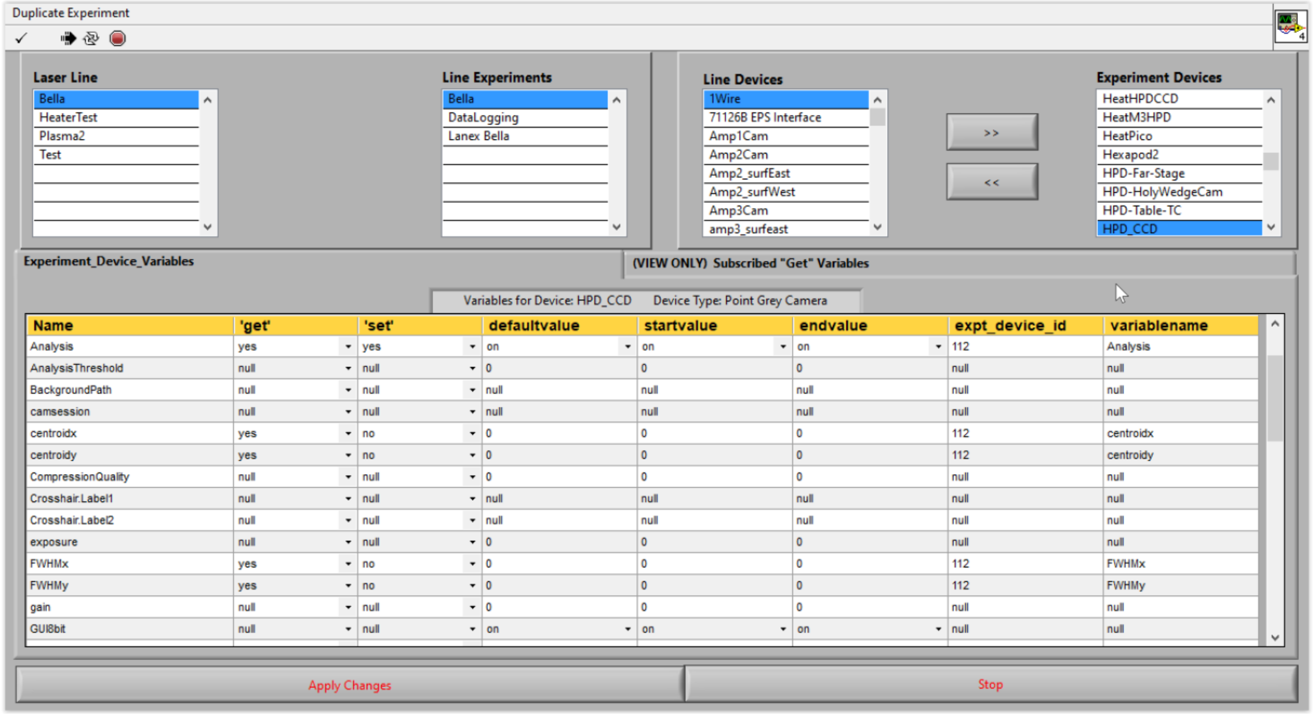
## Add device


Training videos and documentation are on the web

## Configure Device

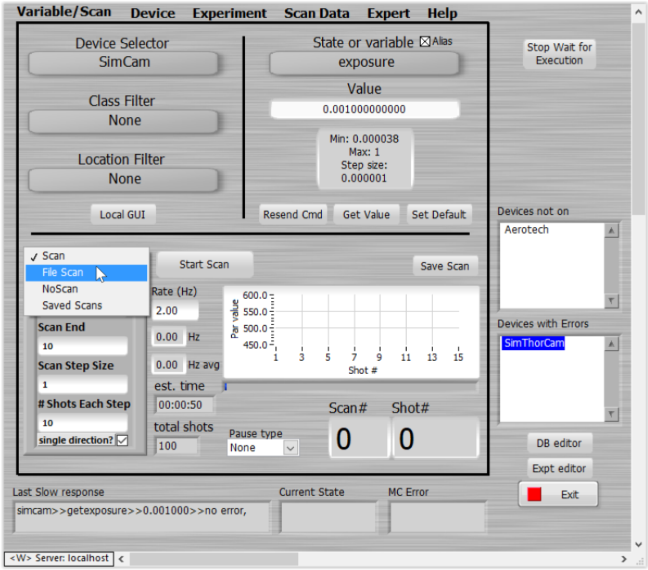



## Configure experiment



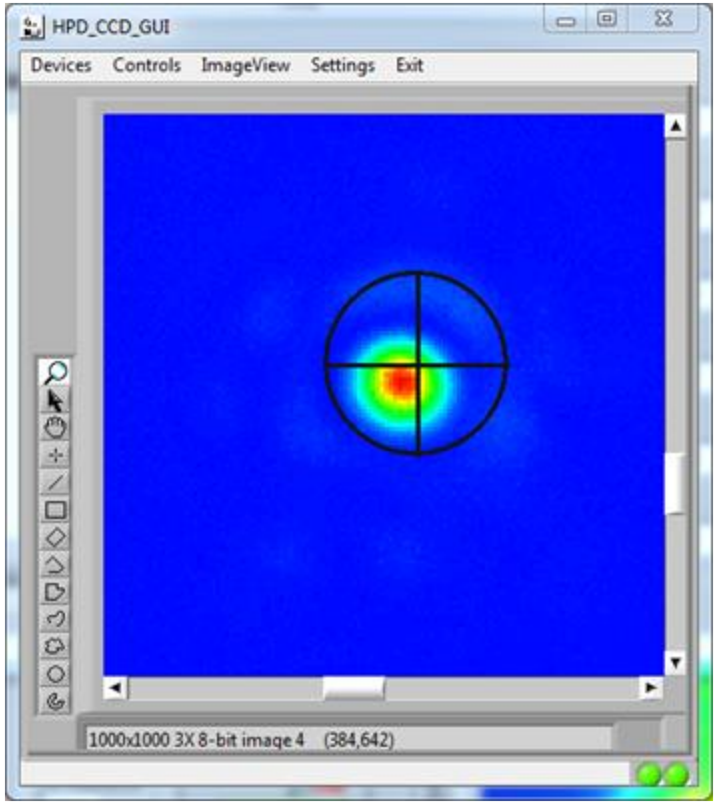
# Simple graphical interfaces provided for immediate access to device control and viewing

General GUI for all devices

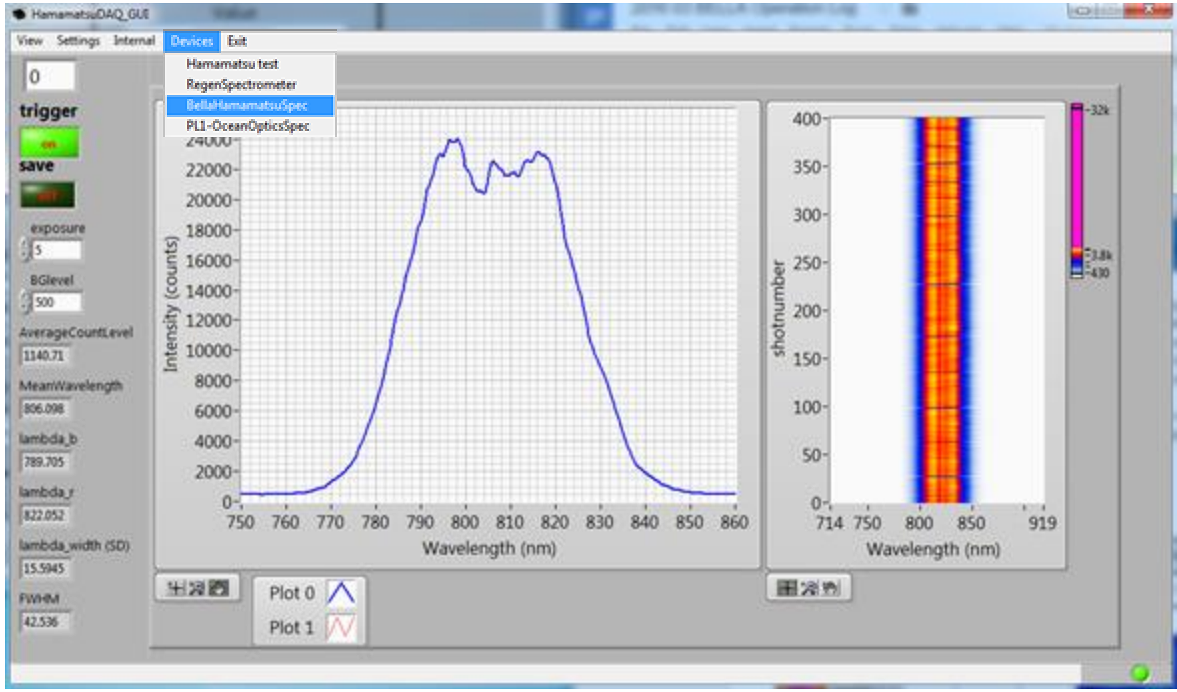


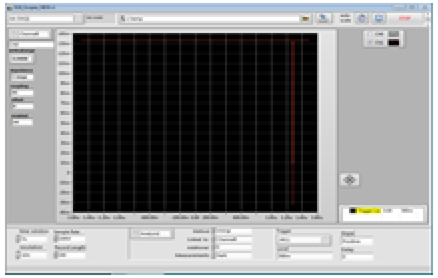

Immediately control any device

General GUI for all cameras



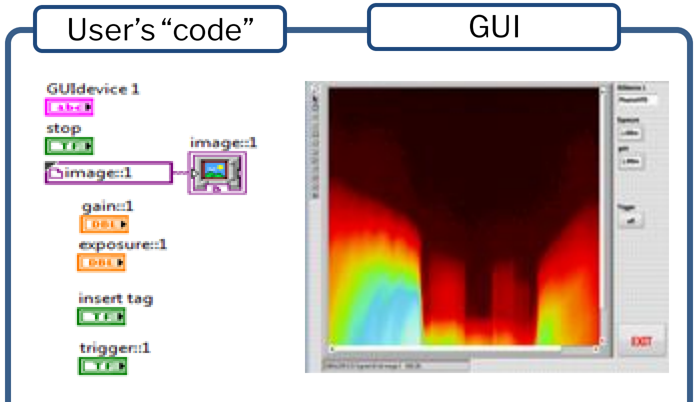

General GUI for spectrometers



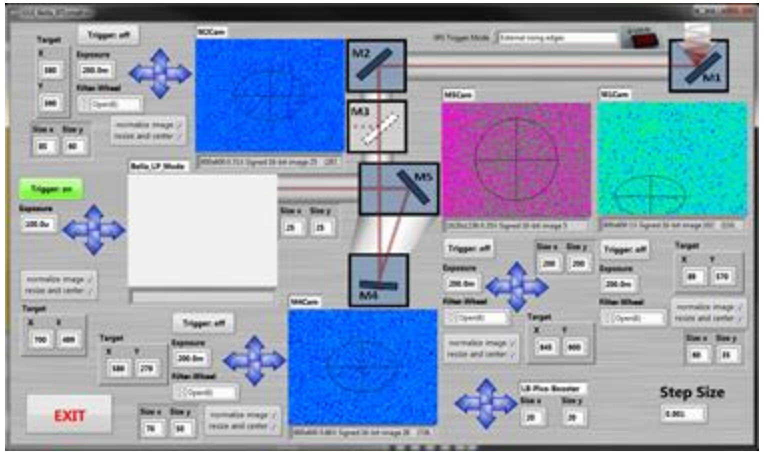



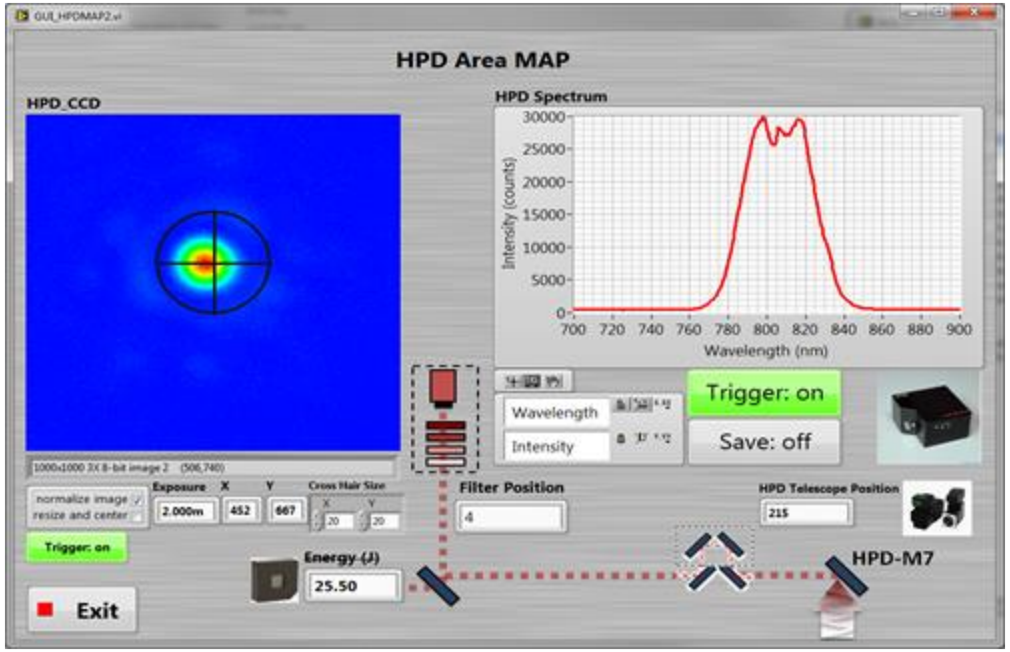
Other general GUIs include oscilloscopes, FROGs, hexapods, digital delay generators, filter wheels…

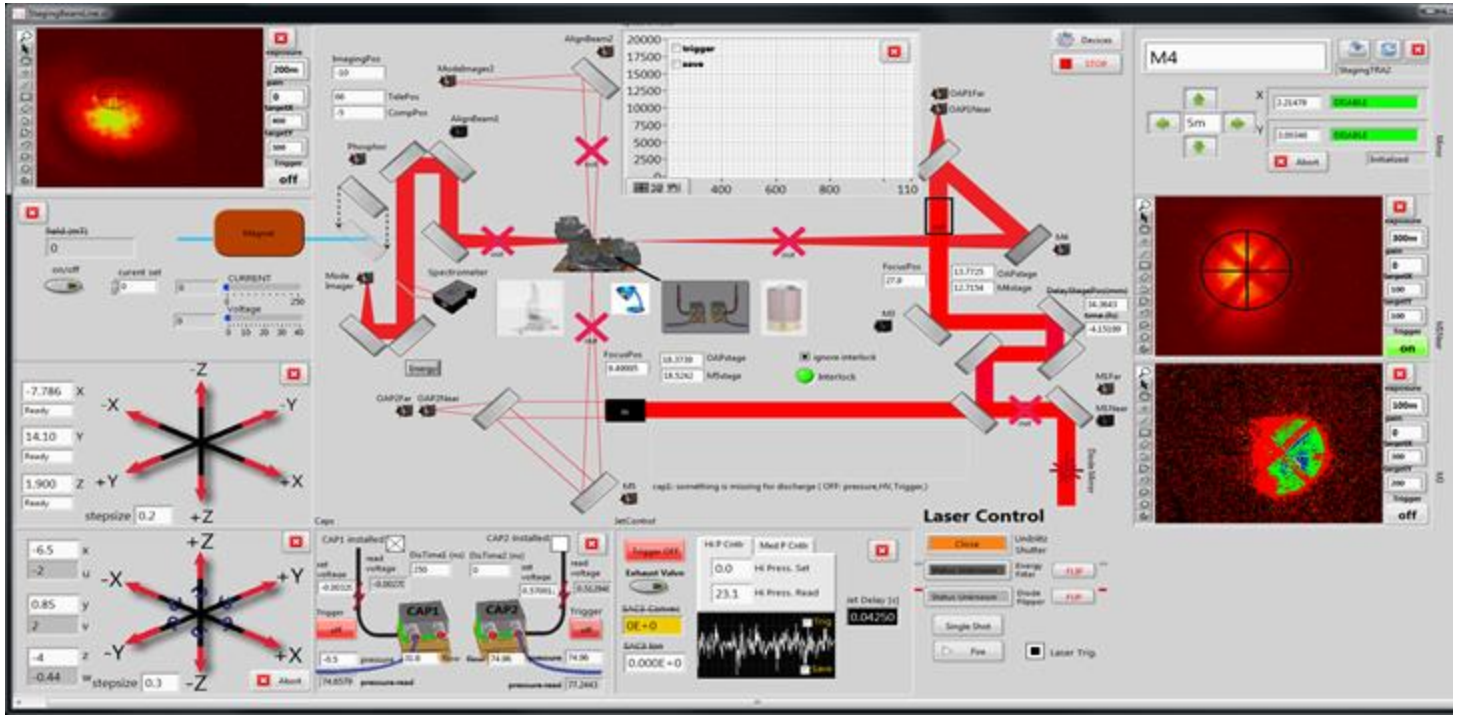
# Custom GUIs communicating with many devices are simple and quick to create

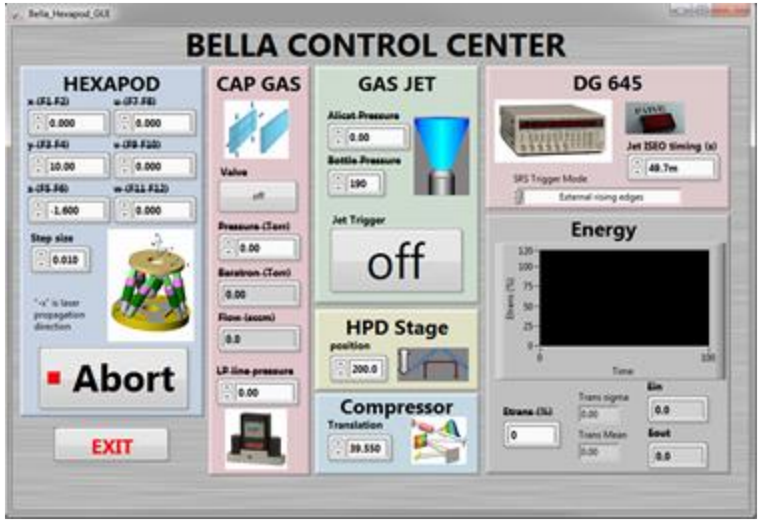



Control & viewing without "programming" in minutes

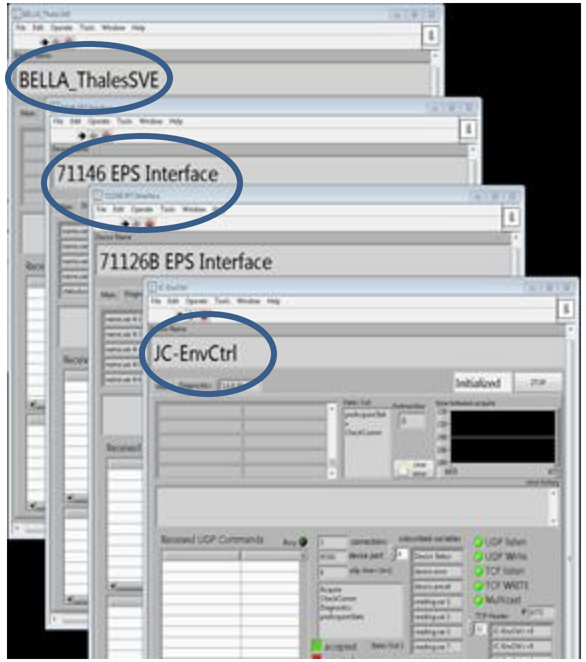


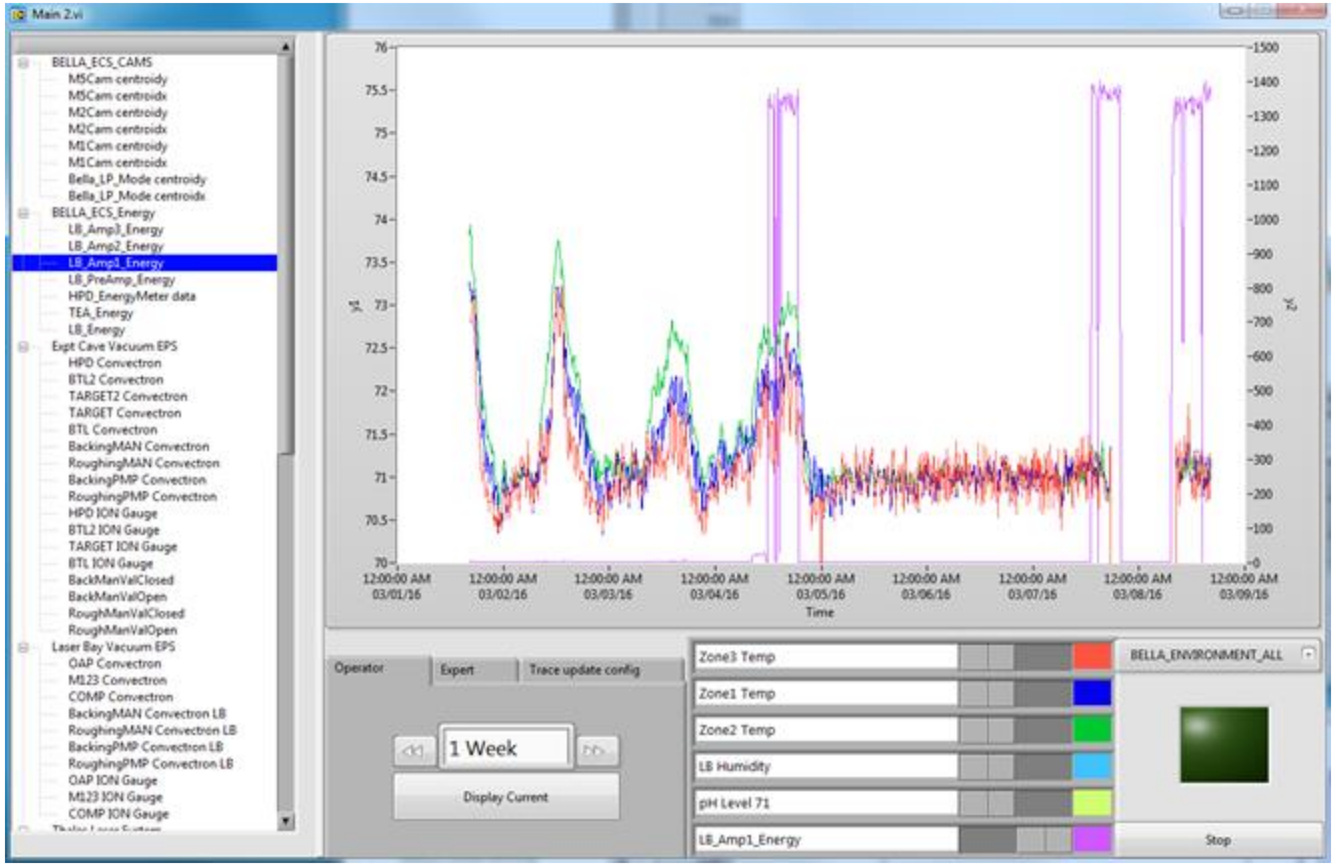



# Selected variables are continuously logged



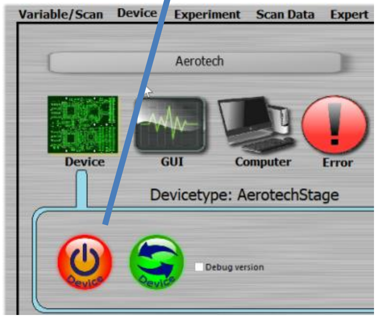

Logger continuously gathers data from devices and writes to database

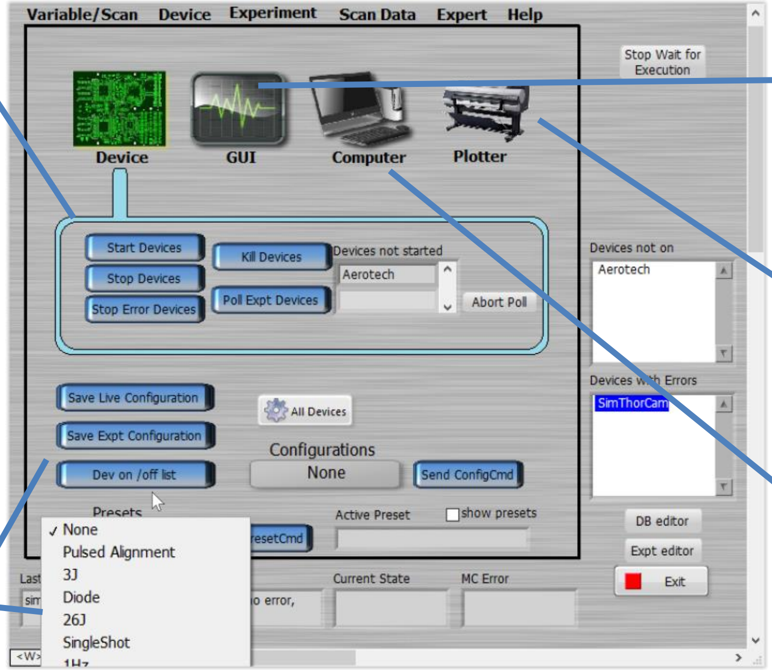
Plotters access database & show trends over arbitrary time scales. Alarming also implemented.



# General GUI provides many of the required user features for starting, configuring, and performing experiments

Start/stop all or individual devices




Start/stop individual GUI or complete pre-configured set for experiment

Online plots

Remote computer control

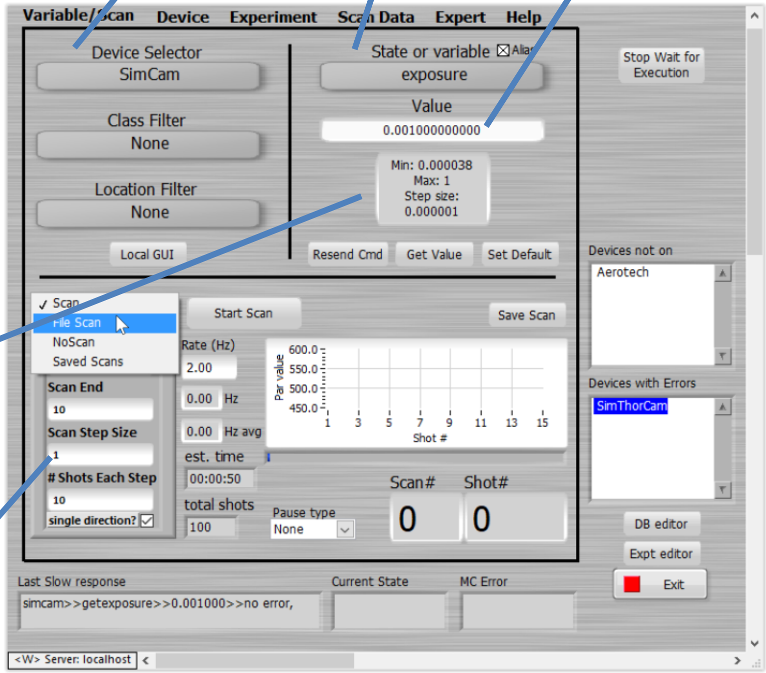
Load/Save partial or complete configurations

- Can start everything (all devices/GUIs) for a given experiment with a few clicks

Select variable

Select Device

Set variable

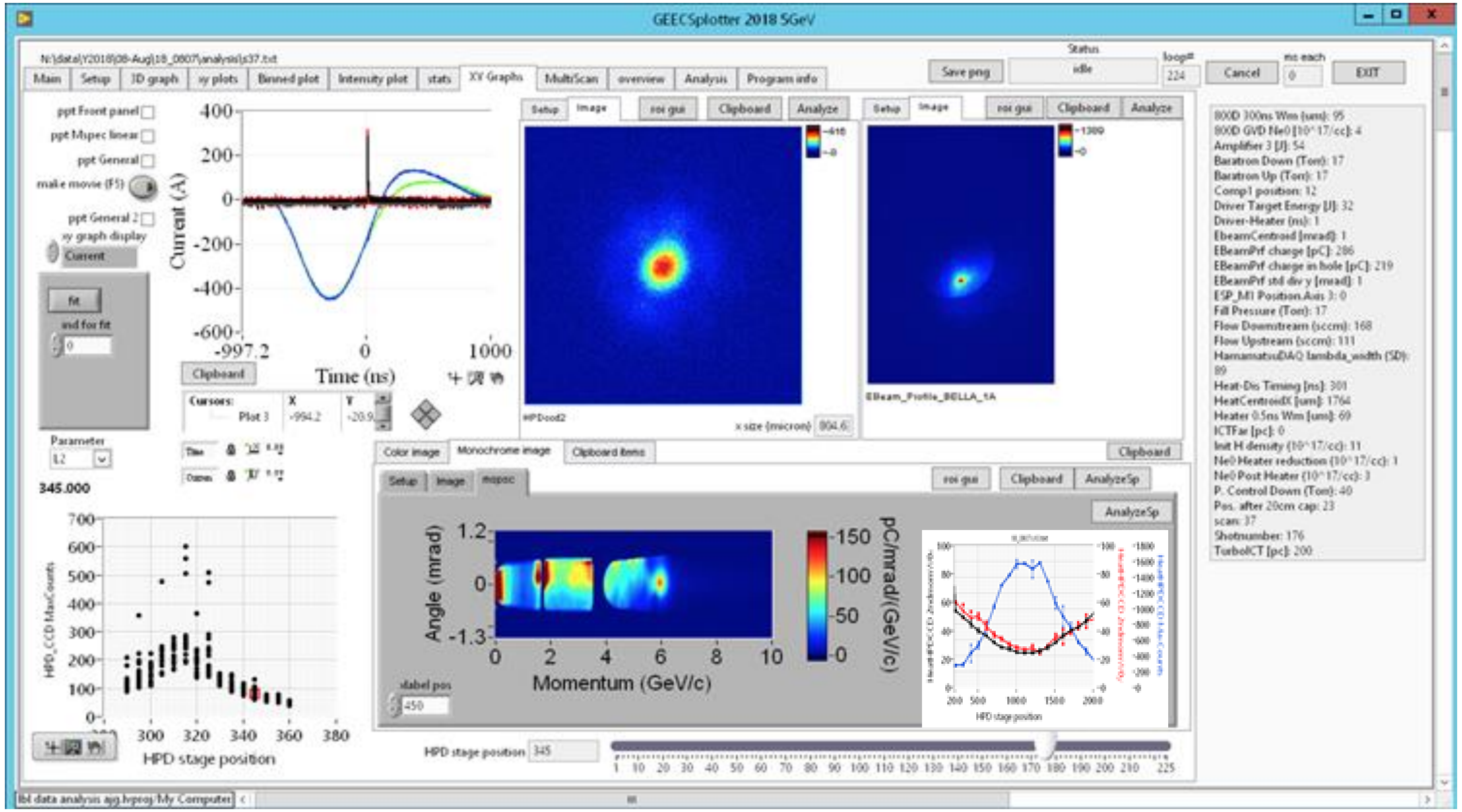



Variable limits from database

Simple or "recipe" scans

# Users write analysis codes but GEECS also has standard tools

- Post-scan analysis
  - GEECS plotter visualizes data
    - Plotting
    - Fitting
    - Comparison with simulation
    - Scripting & report generation
    - Can call Python functions
  - GEECS analyzer analyzes data
    - Images
    - Spectra
    - DAQ
    - Etc

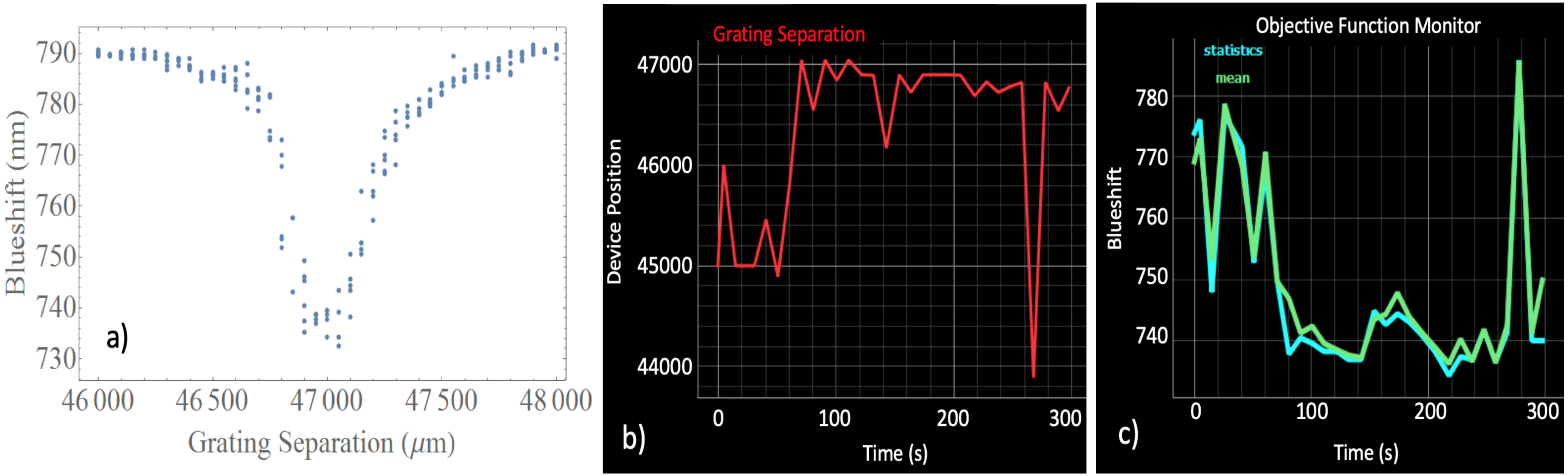


# Overview

- Control system requirements for the user and developer
- Design of the BELLA Center control system - GEECS (Generalized experiment and equipment control system)
- GEECS user experience
- **Integration of ML into GEECS**
- The future of controls at the BELLA center

# GEECS augmented with Ocelot framework. Initial optimization tests successful

- Large, robust community is developing ML/AI tools and techniques, e.g. tensorflow, pytorch etc. which we need to leverage
- Recently interfaced Ocelot generic optimizer[1] to GEECS enabling a wide range of optimization schemes (simplex, extremum seeking, Bayesian, etc.)




Courtesy S. Barber

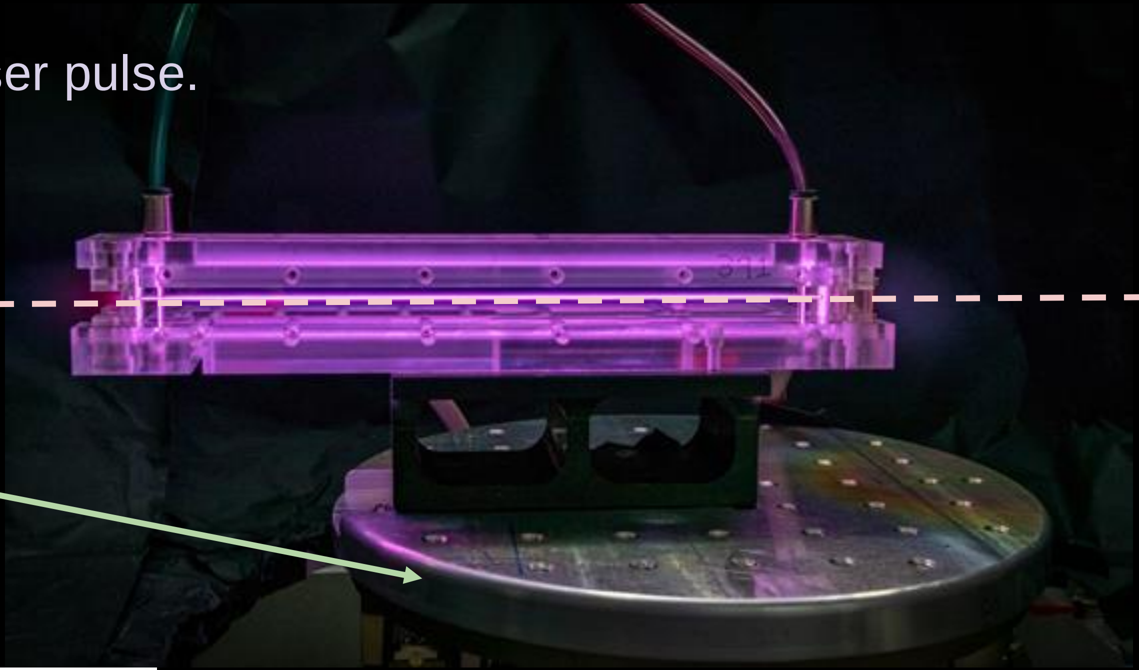
(a) Measured spectral shift of drive laser interacting with gas jet as a function of separation distance of the compressor gratings (i.e. pulse compression). (b-c) Online maximization of spectral shifting using Ocelot framework using Bayesian optimization with Gaussian process. (b) shows the history of positions explored, and (c) shows the value of the objective function at those positions.

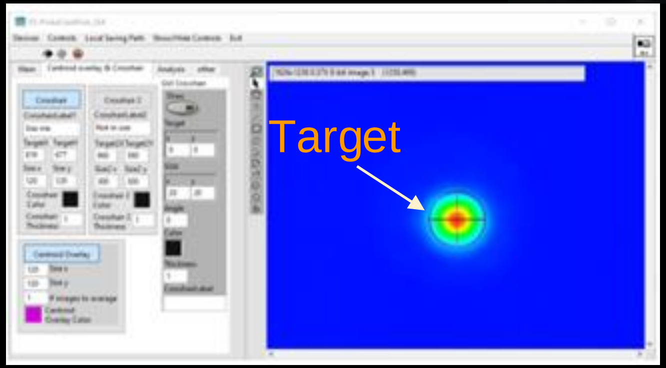
[1]Duris *et al.*, "Bayesian Optimization of a Free-electron laser," *PRL* 124, 124801 (2020).

# Straightforward integration with Dragonfly allows for automated capillary alignment

**Task:** Align capillary to laser pulse.

Courtesy M. Turner




Rms pointing fluct. = ~ 5 - 10 um
Mean centroid dist. <2 um

**Conclusions:**

ML based alignment similar or better than manual alignment.

Alignment time ~ 20 min.

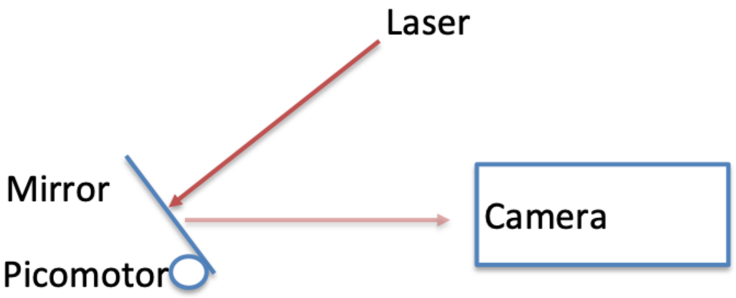
Bayesian Optimization Algorithm

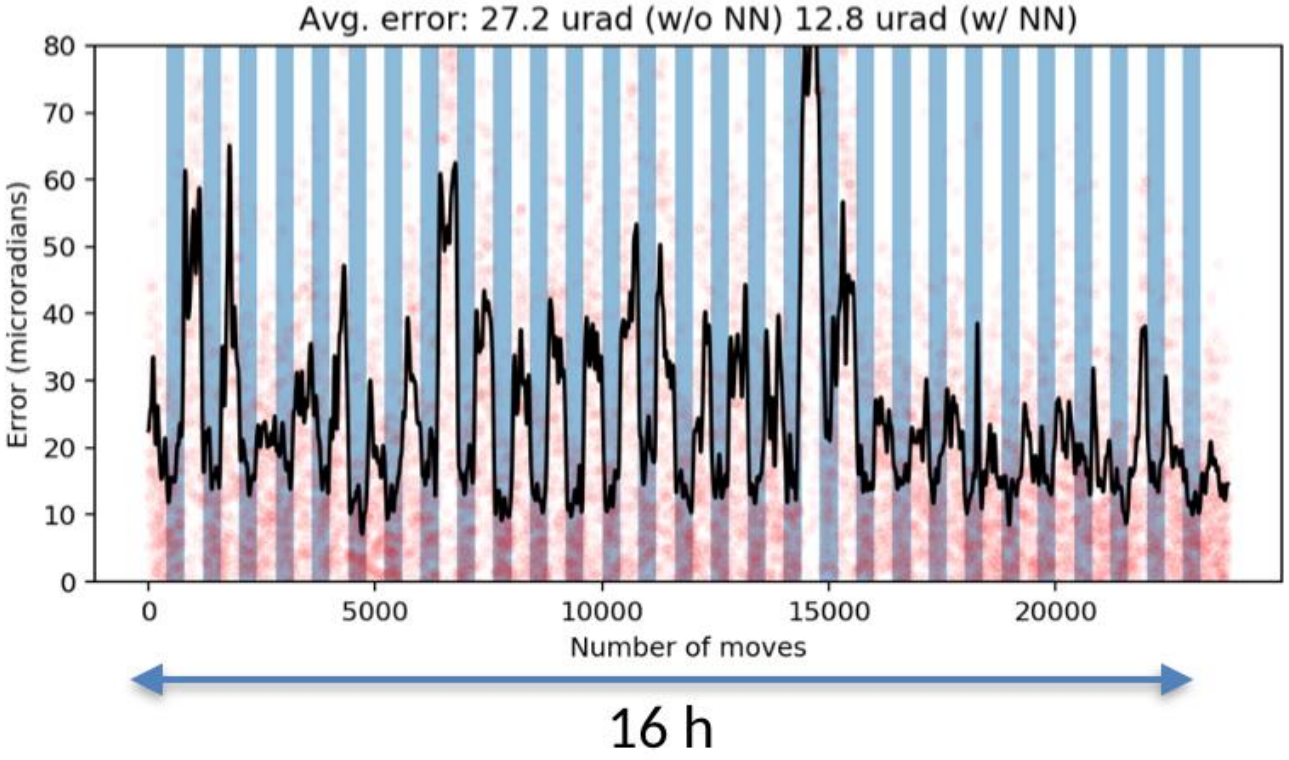
Dragonfly
Scalable Bayesian Optimisation

https://dragonfly-opt.readthedocs.io/en/master/

# GEECS devices controlled by Python ML code can improve corrections of laser pointing by taking into account hysteresis in motors

- Mirror needs to be **continuously adjusted** in order to maintain laser pointing on target
- However, picomotors have **hysteresis**, which limit accuracy and requires multiple moves.
- ML techniques (LSTM neural network) can predict the **hysteresis** and improve corrections of pointing




White areas: correcting pointing with a **linear model** of the motor response

Blue areas: correcting pointing with an **ML model** of the motor response

Keras

Courtesy R. Lehe

# Overview

- Control system requirements for the user and developer
- Design of the BELLA Center control system - GEECS (Generalized experiment and equipment control system)
- GEECS user experience
- Integration of ML into GEECS
- **The future of controls at the BELLA center**

# Conclusion and the future

- Low maintenance, scalable (modular/flexible) LabVIEW distributed control system allowed us to run efficiently with minimal software engineering support for **>10 yrs**
  - Installed on many beamlines (about a dozen, including one facility outside BELLA)
- For the future at BELLA, repetition rate increase needed (kHz)
  - Currently when we need kHz rates, we use standalone codes
  - GEECS was designed to display every shot every diagnostic in arbitrary number of locations.
    - Need modifications to latest shot only & improvements to communication layer. But at some point (**maybe we are here already?**), EPICS/TANGO/Other CS could provide a simple out of the box solution with close to no effort
  - More data to store
    - Larger file servers (leverage larger facilities e.g. NERSC)
    - Data reduction