DSG Software Projects

Patrizia Rossi and the
Detector Support Group
April 27, 2022

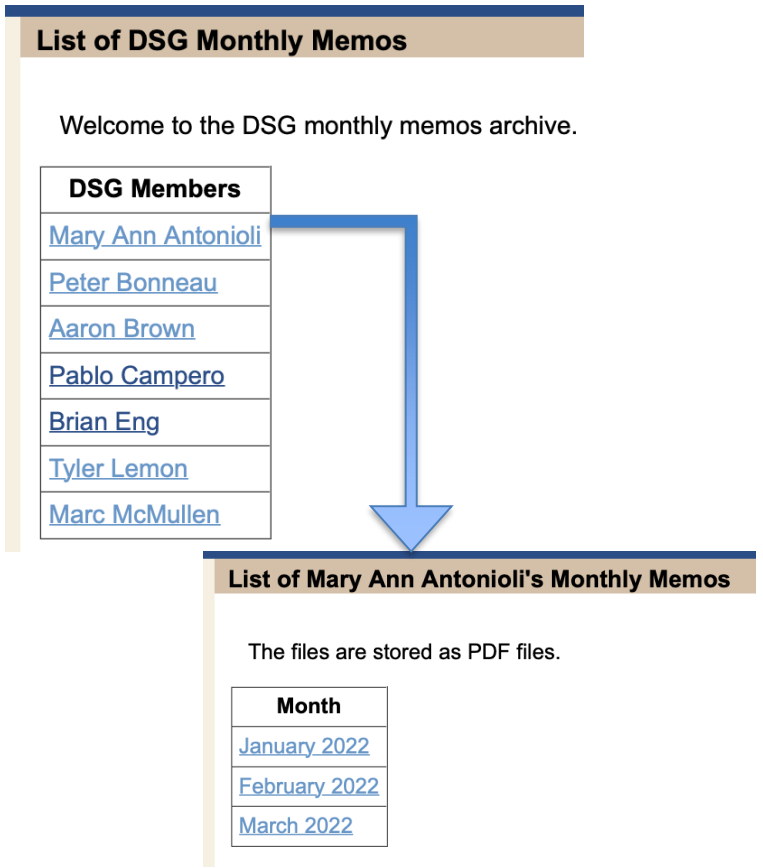
Contents

- **NPS**: Hardware Monitoring System
  Mary Ann Antonioli
- **EPICS**: CSS-Phoebus
  Peter Bonneau
- **NPS**: Crystal Array Thermal Analysis
  Aaron Brown
- **SoLID**: Test Station for PID Controls
  Pablo Campero
- **EIC**: Thermal Analysis of Beampipe
  Brian Eng
- **RICH-II**: Hardware Interlock System
  Tyler Lemon
- **GEM**: Gas Flow Monitoring Software
  Mac McMullen

Contents

- **NPS**: Hardware Monitoring System
  Mary Ann Antonioli
- **EPICS**: CSS-Phoebus
  Peter Bonneau
- **NPS**: Crystal Array Thermal Analysis
  Aaron Brown
- **SoLID**: Test Station for PID Controls
  Pablo Campero
- **EIC**: Thermal Analysis of Beampipe
  Brian Eng
- **RICH-II**: Hardware Interlock System
  Tyler Lemon
- **GEM**: Gas Flow Monitoring Software
  Mac McMullen

https://www.jlab.org/div_dept/physics_division/dsg/weekly_reports/monthly_memos.html



NPS: Hardware Monitoring System

Mary Ann Antonioli

2022-02

NPS Hardware Monitoring LabVIEW Code

Using LabVIEW software, I wrote code and created user interfaces for monitoring NPS hardware. In February, I completed the software portion for monitoring the two NPS chillers, one for the crystal array and one for the electronics zone.

Each chiller has its own tab on the user interface. Each tab indicates the model number of the chiller, its values for supply temperature, pressure, and flow, and a graph of each value type. The expert tab is for choosing to use random numbers or sensor values to run the software. Figure 1 is a screenshot of the user interface for the crystal zone chiller.

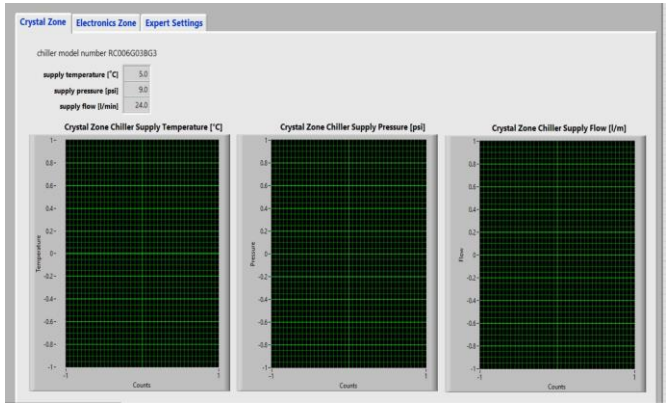


FIG. 1. LabVIEW user interface for the NPS crystal zone chiller, which will display supply temperature, pressure and flow.

A problem was encountered. Initially, the software would only run one loop for generating random numbers and displaying them. After debugging, I discovered that the stop icon for the chiller tab loop was wired to a different stop button. Once wired correctly, the loop, thus number generation and display, ran correctly until manually stopped.

- **Developing hardware monitoring system for operations safety**
- **Designed and developed user interface to monitor two chillers, one for the crystal array and one for the electronics zone.**

NPS: Hardware Monitoring System

I began a second LabVIEW project of writing NPS chiller drivers, which will be used to set and read back from the chillers. Nine drivers were written and a program to test them. Figure 2 is a screenshot of the user interface of the driver to read the chiller status. Chiller, a.k.a plant, temperature, the set temperature, and pressure are given. LEDs light green if various items are on. The response box shows the actual string from the chiller, used for debugging. Communication error turns red if the chiller is not responding.

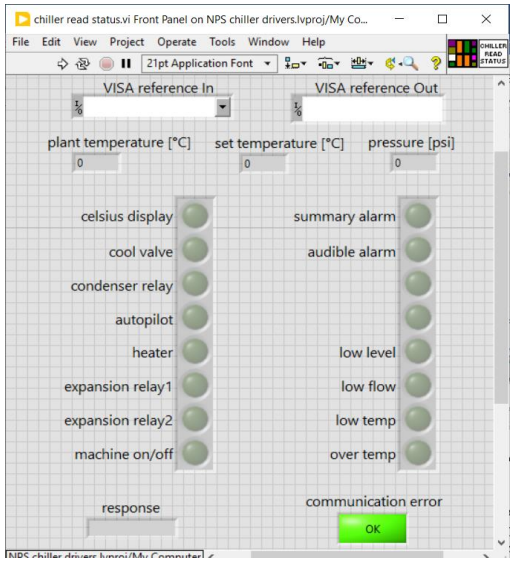
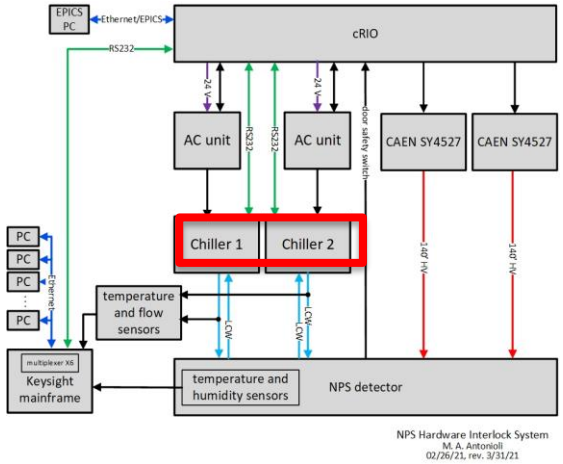


FIG. 2. LabVIEW user interface for the software to read a chiller status.

A significant problem in writing the drivers was trying to understand the chiller manual command instructions. A second problem is that the drivers cannot be tested until a chiller is connected to a test computer.

In the upcoming month, I plan to test the chiller drivers, after a chiller is connected to a test computer. I will also begin working on the Phoebus NPS hardware monitoring user interface.

- **Developed nine drivers to set and read back chiller values**



System diagram of monitoring and interlock system

Mary Ann Antonioli
2022-03

Phoebus Screens for NPS Hardware Monitoring

Using the previously made LabVIEW screens as a guide, I made 12 Phoebus screens for the NPS hardware monitoring system. Two main issues were encountered when creating the screens.

.....

- **Developing Phoebus user screens to monitor NPS hardware, based on LabVIEW NPS monitoring program**
- **Developed 12 Phoebus User screens**

EPICS: CSS-Phoebus

Peter Bonneau

2022-02 & 2022-03

EPICS Alarm System in Phoebus

I am developing an EPICS alarm system based on CS-Studio Phoebus. Phoebus will be used for new EPICS system development and will replace the existing Eclipse-based CS-Studio systems as detailed in my note DSG Note 2021-37 and talk DSG Talk 2021-17.

I am working on the configuration and initial start-up of the Phoebus alarm server. The server is an alarm system support program that monitors process variables (PV's) on the network via EPICS channel access.

Based on a template in the Phoebus source code, I wrote an alarm_server.service file. This file is used by Linux at computer boot time to automatically start the alarm server. The file contains parameters used by the alarm server at startup such as the environmental variables and the alarm system configuration name.

When starting the alarm_server.service, an error occurred stating: Failed to determine user credentials: No such process. I looked deeper into the code and found another program called procServ is used in conjunction with the alarm server. I looked through the Phoebus source code and it does not have procServ and the documentation does not mention this dependency.

From my research, I determined procServ is used as an interactive command wrapper which allows remote access via telnet to the alarm system command console while the system is in operation. Written by the EPICS collaboration, procServ is also used with other EPICS processes such as soft IOC's. I downloaded the procServ source code from GitHub. I also obtained and installed the telnet protocol handling library required by procServ. I successfully built, installed, and tested procServ.

I restarted the alarm_server.service and the alarm server console successfully read the configuration file and began the initialization process. When the alarm server initialization reported "Started Phoebus Alarm Server", an error from the alarm server console is shown indicating "Main process exited, status=2/INVALIDARGUMENT".

- **Developing CS-Studio Phoebus based controls, monitoring, and alarm system - to be implemented on Hall C detectors**
- **Debugging startup of alarm server**
- **Plan to implement message streaming (block diagram on next slide) interface for alarm server**

EPICS: CSS-Phoebus

I investigated this error and concluded that one or more command line arguments supplied to the alarm server program when invoked by *procServ* was incorrect. I tested the individual arguments and found the path to the alarm server directory was incorrect in the *alarm_server.service* file. When this was corrected, the alarm server initialization successfully progressed passed this initial error. Figure 1 shows the successful start of the alarm server initialization.

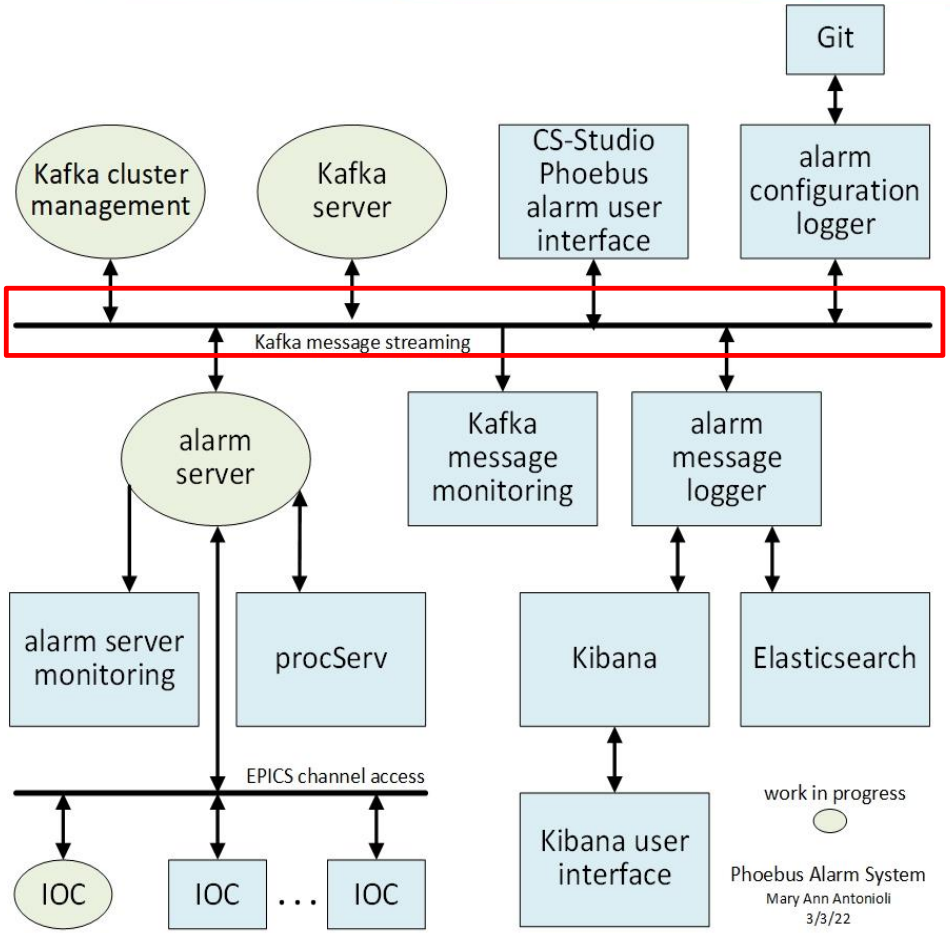
I restarted the alarm system initialization process. When the alarm server console reported *"Fetching passed alarm states"*, an error from the alarm server console is shown indicating *"connection to node could not be established"*. I investigated this error and determined that the database implemented with Apache Kafka must be configured for the alarm system and operational before starting the alarm server. This alarm server dependency was mentioned nowhere.

I plan to get Apache Kafka configured and operational as the next step in the development of the Phoebus alarm system.

```
[bonneau@fedora ~]$ sudo systemctl status alarm_server.service
● alarm_server.service - Phoebus Alarm Server
     Loaded: loaded (/etc/systemd/system/alarm_server.service; enabled
     Active: active (running) since Mon 2022-02-14 15:06:29 EST; 3min 28s ago
   Main PID: 3713 (procServ)
      Tasks: 1 (limit: 37986)
     Memory: 300.0K
        CPU: 11ms
     CGroup: /system.slice/alarm_server.service
             └─3713 /usr/bin/procServ --foreground --noautorestart --name alarm-server --chdir /home/bonneau/Downloads

Feb 14 15:06:29 fedora systemd[1]: Started Phoebus Alarm Server.
```

FIG.1. Start of the Phoebus Alarm System Server



Phoebus Alarm System Programs

[DSG Note 2021-37]
[DSG Talk 2021-17]

NPS: Crystal Array Thermal Analysis

Aaron Brown

2022-02

More on the Thermal Analysis of NPS Crystal Array with Ansys

As mentioned in my previous monthly memo [1], temperature probes were placed on the front and rear faces of each crystal of the NPS crystal array model. These temperature values were extracted from the Ansys thermal simulation solution to a text file using the IronPython script get-results4.py.

The next task was to parse the data file from the Ansys simulation. To do this, I generated a new Python program (parse-temps.py) to remove header information from the text file leaving only the temperature value behind. The purpose for this is to be able to generate plots showing the temperature profile of the crystal faces only and not of the entire system as a whole. This way we can determine the temperature gradient from crystal to crystal.

To plot the crystal face temperature probe data, this new Python program stripped out the header information for each temperature probe. The temperature probe values had to be extracted in the proper order (crystal #0 to crystal #1079) for both front and rear crystal faces since the header information in the text file was to be removed. The remaining data still contained multiple delimiters and extraneous information, Fig. 1.

```
NaN,Time [s],Front Crystal 0 [°C]
1,1.,11.816
NaN,Time [s],Front Crystal 1 [°C]
1,1.,12.694
NaN,Time [s],Front Crystal 2 [°C]
1,1.,13.113
NaN,Time [s],Front Crystal 3 [°C]
1,1.,13.312
NaN,Time [s],Front Crystal 4 [°C]
1,1.,13.554
NaN,Time [s],Front Crystal 5 [°C]
1,1.,13.641
NaN,Time [s],Front Crystal 6 [°C]
1,1.,13.651
NaN,Time [s],Front Crystal 7 [°C]
1,1.,13.778
```

FIG 1. Screenshot of extracted temperature probe values text file

After removing the unnecessary information, the temperature probe values were separated into two arrays (one each for the front and rear crystal face temperature

- **Developed Python code to plot extracted Ansys temperature data**
- **Generated plot of front crystal face temperatures**
- **Determined central crystal region temperature dependent on temperature in hut (22°C); coolant at 10°C affects only crystals on the periphery, simulation results on next slide**

NPS: Crystal Array Thermal Analysis with Ansys

probe values). Each array, containing 1080 temperature values, was plotted, Fig. 2, temperature ranges are shown in Table 1

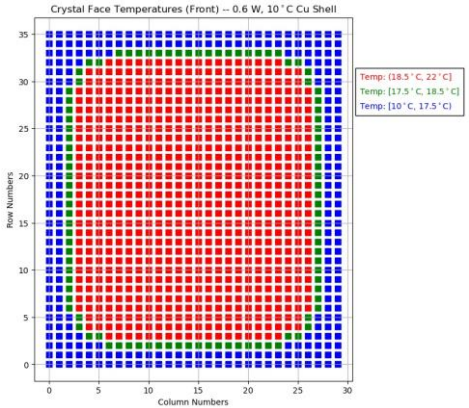


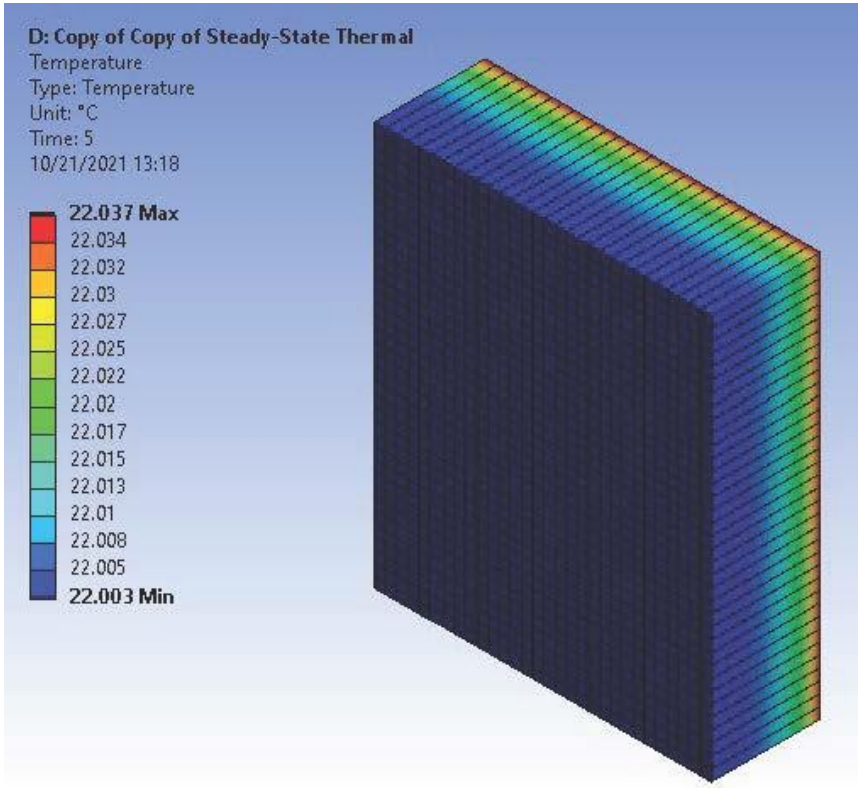
FIG 2. Python plot of front crystal face temperature probe values

Temperature range	Color
[10.00°C, 17.50°C)	Dark Blue
[17.50°C, 18.50°C]	Green
(18.50°C, 22.00°C]	Red

Table 1. Temperature ranges for Python plot of Ansys temperature probe values.

This new Python program, *parse-temps.py*, will be used to plot temperature probe values for future thermal simulations of the NPS crystal array. Eventually, a plot of the temperature gradient from a crystal face to adjacent crystal faces will be generated using the temperature probe values extracted from the simulation solution.

[1] Brown, Aaron DSG Monthly Memo 2022-01



Snapshot, after 5 s, of thermal conduction along the length of the crystals

SoLID: Test Station for PID Controls

Pablo Campero
2022-02

Test Station for PID Controls of SoLID

I continued with the development of PLC code to test PID control over the electric valves that will be used for the Hall A SoLID magnet cryogenic operations. The code under development is part of the proposed PLC test station.

I added code to simulate the behavior of the liquid level sensor readout in sub-routine Liquid Level. Liquid level sensor is commonly installed inside the helium or nitrogen cryogenic reservoirs. The readout of the liquid level is critical for the control of the valves since the opening or closing of the valve depends on a reliable readout of this variable.

The added code simulates the filling and emptying of a tank while a valve is open and its outlet is open. I added conditions to limit the filling and emptying of the tank from 0 to 100%. I implemented timers to control the flow rate for filling so the readout of the liquid level is close to a real scenario.

For the HMI programing required for controls and monitoring, I created and configured the HMI screen to monitor and control liquid level. I started adding the indicators for liquid level, filling, and interlock status. I implemented controls to simulate a tank opening and closing, Fig.1.

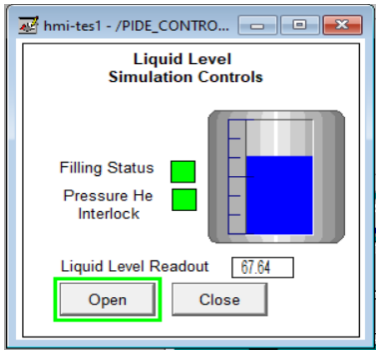


FIG.1. HMI screen under development to simulate the liquid level variable. Screen shows data read from the PLC emulator

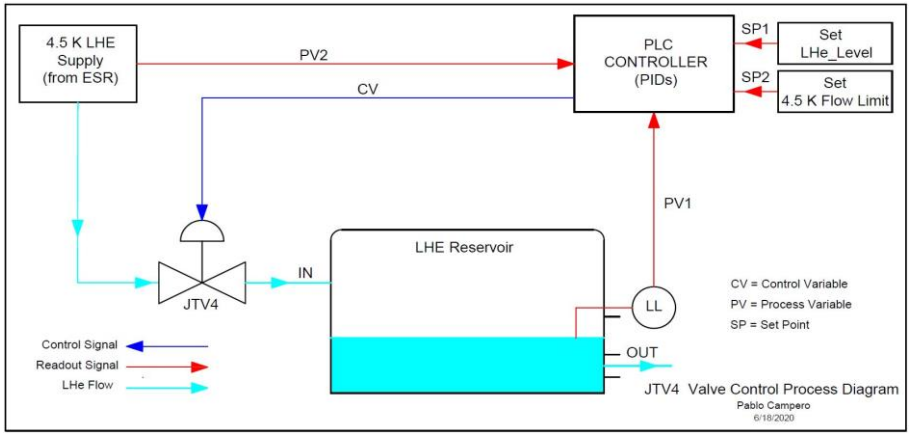
- **Developing PLC code using PID functions to control and monitor electric valves**
- **Developing HMI screen to control and monitor liquid cryogen levels**
- **Plan to implement code and screen to operate electric valves in automatic mode**

SoLID: Test Station for PID Controls

I solved the problem presented last month about the remote access to the PLC software. I added the laptop host name to the list of allowed computers/users by using FactoryTalk Administration Console software. I ran a test accessing PLC programming software remotely and accessed software with no issues.

I had some issues while developing the testing HMI screen for the valve controls and monitoring since I was running FactoryTalk View Studio (HMI development software) in Demo mode with no available license on my computer. Running HMI software in Demo mode allows the user to run the software for a maximum of two hours, after which the software is closed automatically, losing modifications done to the project. Saving must happen before that period to prevent losing any progress. Currently DSG has a standalone license that enables access to the HMI software, which runs on a dongle and can be connected via USB to any computer.

PLC code to operate the valve in automatic mode will be added.



Block diagram of PLC controls for liquid helium in reservoir

EIC: Thermal Analysis of Beampipe

Brian Eng
2022-02

Ansys Computational Fluid Dynamics (CFD) Analysis

To better understand the interaction between the heated beampipe (mainly done to remove any residual contaminants) and the first layer of the silicon pixel detector, the analysis solution is being switched from a steady state to fluid.

Per vacuum experts, the inside wall of the beryllium beampipe must reach at least 100°C at a minimum. The first layer of silicon in the current detector model is separated from the beam pipe by ~1.25 mm radially.

Ansys Workbench is the main program where the analysis systems for the simulation can be chosen for a project, e.g. steady-state thermal, transient thermal, harmonic response, fluid flow, etc. Within the fluid flow analysis, there are several choices available for the analysis system, but this depends on the options selected during Ansys installation. Initially only CFX was available; after modifying the installation, Fluent and Polyflow became available, Fig. 1. The focus on this paper will be between CFX and Fluent, namely their workflow.

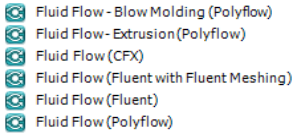


FIG. 1. Current Ansys Workbench standalone analysis systems for fluid flow.

I found that both CFX and Fluent projects have the same steps, Fig. 2, for a fluid flow simulation, namely: Geometry, Mesh, Setup, Solution, Results.

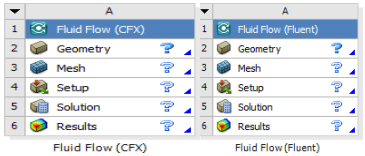


FIG. 2. Ansys Workbench standalone project steps for CFX (left) and Fluent right).

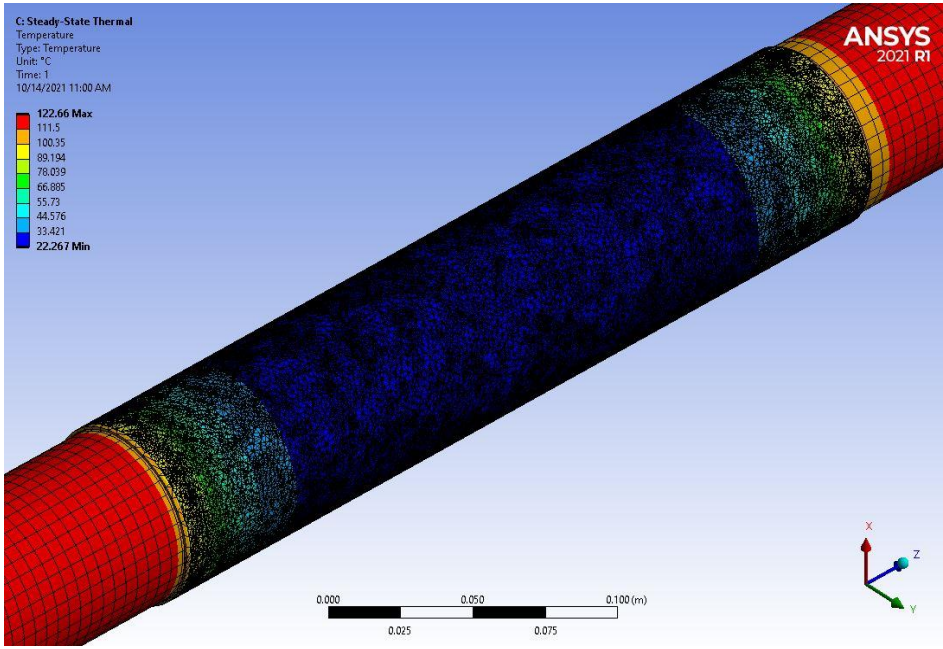
- **Setting up simulation, using CFD, for thermal interaction between heated beampipe and first silicon layer**
- **Within the fluid flow analysis, several choices available for the analysis system, depending on the options selected during Ansys installation.**
- **Initially only CFX available now also Fluent and Polyflow available. Comparison between CFX and Fluent workflow ongoing**

EIC: Ansys Thermal Analysis for Beampipe

The functions for CFX and Fluent are the same but how they ended up being used in practice presented some issues. I determined that all of the steps involve configuring various options, which must be completed successfully prior to the next step being able to execute, e.g. if the geometry step is incomplete or fails the mesh step will not run. The geometry step allows one to either import or design a new geometry for the 3D model being simulated. The mesh step applies a mesh to geometry from the previous step. The first two steps are similar between CFX and Fluent. The setup step is where all the details for the simulation are added: what materials are being used, the boundary conditions between bodies, initial parameters (flow, temperature, etc.), among some of the similar parameters, but as CFX and Fluent perform their calculations slightly differently (CFX being vertex-centered and Fluent being cell-centered), their setup is slightly different. The solution step allows for changes to the simulation itself, number of runs, convergence, etc. The final result step takes the output from the solution step and allows for various plots and visualizations to be set up.

Aside from the calculation method and slightly different setup steps, I found that the biggest difference between the CFX and Fluent analysis is when a license is checked out for the software to run. Both software requires the following licenses to run: cfd_base, cfd_solve_level1, and cfd_solve_level2. For CFX these are checked out only when the solution step is actually running, meaning it is possible to set up the initial conditions and the solution itself without any licenses. For Fluent the licenses are checked out at the start of the setup step, meaning if there are no licenses available no work can be done.

As there is really only a single available CFD license, this has made progress very slow, as very often all available licenses have been in use. This makes iteration and refinement on the analysis slower than it could be. Currently another CFD license package is being procured, which should help alleviate this issue.



L1 silicon layer around beampipe (red)

- **Found license checkout process depends on which CFD software is selected**
- **Procured additional license as Jlab has only a single floating CFD solver license and it is in use continually**

RICH-II: Hardware Interlock System

Tyler Lemon

2022-02

Shared Variables for the Hardware Interlock System

I am working on the LabVIEW program for the RICH-II hardware interlock system. I am adding network shared variables and EPICS communication to the program, which allows remote monitoring of all relevant variables using an expert LabVIEW user interface, a CSS user screen, or a CSS expert screen.

To make it easier to add network shared variables to the LabVIEW program and link EPICS PVs to the network shared variables, I developed a separate Python program to create a configuration file for the LabVIEW network shared variable library and a database file for the EPICS PVs. The Python program reads and parses an Excel file, which contains the name of the network shared variable, the data type of that variable, and the EPICS PV to assign to that network shared variable. The Python program then creates a CSV that can be imported into LabVIEW to add all network shared variables to the LabVIEW program's shared variable library and a .db file that can be imported into the sbRIO's EPICS server or client to create the EPICS PVs that are linked to the network shared variables.

Without this Python program, the only way to create and assign properties to network shared variables and EPICS PVs in LabVIEW is to manually add the items to the project file. Because RICH-II's program will have over 300 shared variables, this process would be tedious, time-consuming, and prone to typos.

A problem that was encountered was there is very little documentation on the syntax and format of the file that LabVIEW can use to import variables into the network shared variable library. To overcome this, I had to manually create a template variable in the LabVIEW project, export that variable to a CSV (a built-in capability of LabVIEW projects), and then use that CSV as a template for the overall format of the output of the Python program.

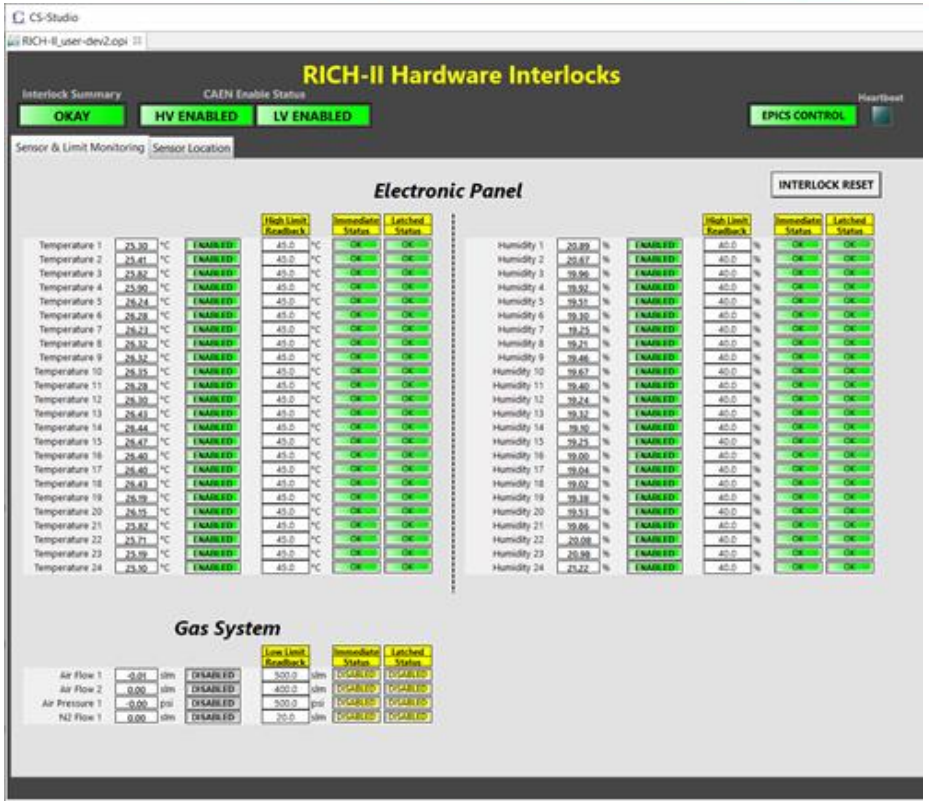
- **Adding EPICS communication capability to LabVIEW program**
- **Developed Python program to create a configuration file for the LabVIEW network shared variable library and a database file for the EPICS Process Variables (PV's).**
- **Plan to test interlock system on Hall B development subnet in EEL building and review acquired data with a separate analysis program - being developed**

RICH-II: Hardware Interlock System

Another problem encountered was the sbRIO did not have the correct EPICS software installed on it. This was resolved by reconfiguring the sbRIO with the correct software.

As of February 15, 2022, the software for this task has been completed,fig1, but not used and tested. The sbRIO and hardware interlock system has only recently been moved to EEL and set up on the JLab Hall B development subnet (previously, the system was at home to allow work-from-home on the project).

In the next month, the network shared variables will be tested on the Hall B development subnet. Additionally, the sbRIO will be configured as an EPICS server to allow the shared variables to be published to EPICS. After data are able to be read in EPICS, all sensor data (with no dead band) will be archived using the existing MYA infrastructure. At this point, while data is recorded, a separate analysis program (most likely in Python) will be developed to determine the stability and behavior of the sensors over long periods of time.



Portion of CSS-BOY screen developed to monitor and control RICH-II hardware interlock system using EPICS.

GEM: Gas Flow Monitoring Software

Marc McMullen
2022-02

GEM Gas Flow Monitoring Software: Final stage

I am working to improve the reliability of the Hall A GEM gas monitoring system. I am modifying the code to read all flows and both pressures using a single channel, Fig. 1. The rendering of the gas panel is shown in fig.2.

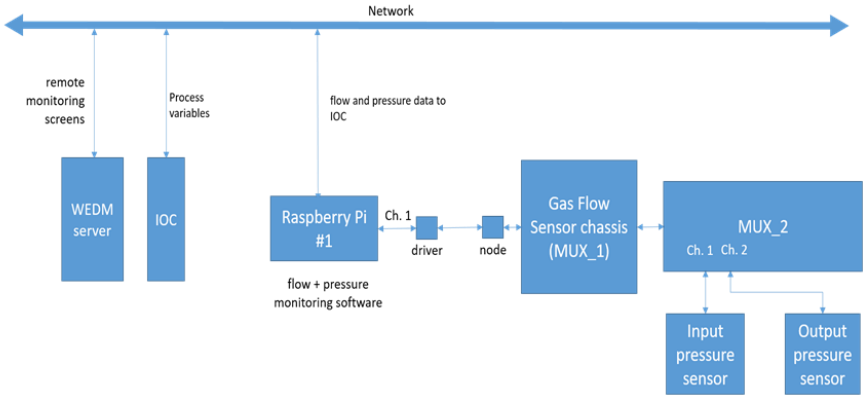


FIG. 1 The GEM BigBite data diagram will use a single Raspberry Pi channel to read all flow and pressure sensors

To do this I am using a second DSG-developed multiplexer to alternate between the input and output pressure transducer before reading the pressure data on the channel. I am trying different methods to sequence the change in multiplexer channels to accomplish switching. On the initial try, the software reads all the channels of the flow multiplexer (MUX_1), then closes that multiplexer. The flow multiplexer is installed in the Gas Flow Sensor chassis and has eight gas flow sensors connected to it.

- **Modifying hardware and software to improve robustness of gas flow and pressure readout system**
- **Tested different software modifications to use a second multiplexer to read pressure sensors**
- **Plan to implement code on Raspberry Pi using LabVIEW**

GEM: Gas Flow Monitoring Software

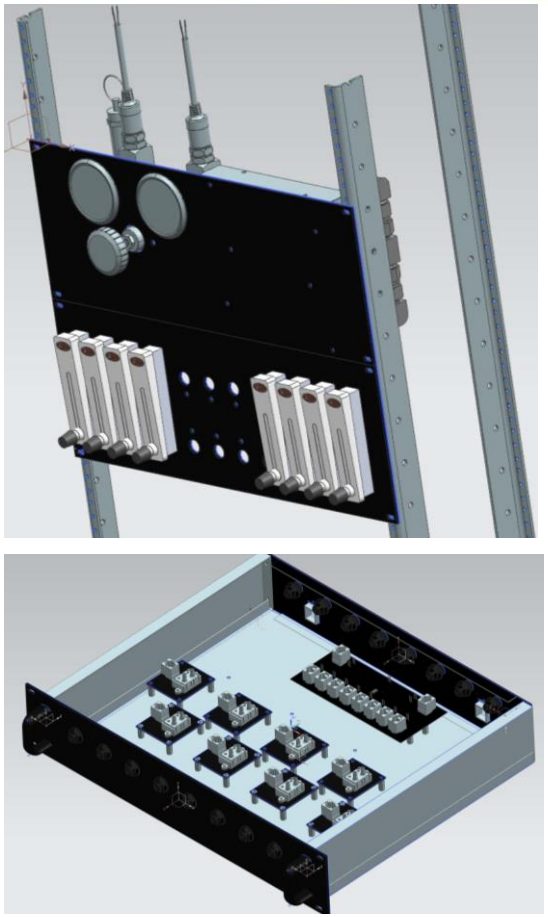
The code then calls the pressure reading program, which uses a three iteration for-loop to change the channels of a second multiplexer (MUX_2), read, and record the data from the two pressure transducers. Table 1.

Pressure program for-loop	
Iteration 1	Read input pressure sensor and write the data to PV
Iteration 2	Read output pressure sensor and write the data to PV
Iteration 3	Close MUX_2 and re-enter the main loop to read all the flow sensors

Table 1. Pressure program for-loop iteration steps

The issue with this method is that the program works for a while until it stops reading the pressure sensor. When the program stops, the program must be restarted. To mitigate the issue, I will change how the program reads the pressure. For instance, instead of using a for-loop to switch the pressure channels, I will only use the pressure program to read one of the pressure sensors at a time and write the data to a process variable. I will alternate the pressure channel on each loop of the main program. This will update each pressure reading every other loop of the main program, which is approximately six seconds for the BigBite.

Next month I plan to download the LabView module for Raspberry Pi and write a LabView version of the GEM flow and pressure readout software.



Rendering of the GEM gas panel and gas flow sensor chassis in NX12.
Top: gas panel.
Bottom: gas flow sensor chassis

Conclusions

- DSG making significant contributions to several software projects
  - Software memos on progress submitted monthly