UNH · NUCLEAR & PARTICLE PHYSICS

Polarized Target Lab:

Software & Automation Updates

ssRF GUI · Mattermost Autolog Bot · Helium Liquefier HMI · Python DAQ

Muhammad Farooq
Advisor: Prof. Karl Slifer
Department of Physics and Astronomy · University of New Hampshire

b_1/A_{zz} Collaboration Meeting · June 3-5, 2026

University of New Hampshire

UNH NPG

U.S. DEPARTMENT OF ENERGY | Office of Science

This work is supported by DOE contract *DE-FG02-88ER40410*

Overview

What this talk covers

01 ssRF GUI

Combined signal generator GUI for safe, reproducible ssRF & AFP burns with hardware interlocks and automated BLAST sequencing.

02 Mattermost Autolog Bot

Automated lab monitoring bot: screenshot capture, LabVIEW CSV data posting, message queuing, and edit detection.

03 Helium Liquefier HMI

Flask-based SCADA dashboard for the QT P663 reliquefier: Modbus TCP, live P&ID, alarm decoding, and setpoint logging.

04 Python DAQ Migration

Transition from LabVIEW to a Python/EPICS-based DAQ with soft IOC, pyVISA device bridge, and pyEPICS GUI.

ssRF GUI — Motivation

Why we need the GUI

⚠ Before the GUI: The Problem

- Required 2–3 operators for a manual multi-step protocol
- Hand-operated switches & separate instrument panels with tightly coordinated timing
- Any timing error risked exposing the NMR preamplifier to full RF power
- NMR preamp damage → weeks of downtime and expensive replacement
- No logging, no reproducibility, no automated safety checks

✓ The Solution: Combined Signal Generator GUI

- Single-button BLAST sequence orchestrates 5 instruments across 3 buses
- Hardware interlocks enforce NMR/ssRF mutual exclusion automatically
- Double-confirmed switch-state verification before every critical step
- Timestamped log file for every run — full reproducibility
- One operator can execute ssRF/AFP burns safely and reliably

ssRF GUI — Three-Tab Interface

How we use the GUI

Tab 1

SMC100A Control

- Connect/auto-detect VISA resource string
- Set carrier frequency (33.0 MHz) with ±1M/±10M quick steps
- Set RF output power (dBm) via field or slider (−120 to +30 dBm)
- Configure FM deviation (10 kHz) and source (EXT)
- Toggle FM/RF ON/OFF; switch port control for diagnostics
- SCPI console: send any raw command, see response

Tab 2

RF Level Sweep

- Configurable power ramp on SMC100A with Agilent 33220A trigger
- Parameters: carrier, start/stop dBm, step dB, dwell ms, TRI/SAW shape
- Click RUN → sweep executes with real-time power plot
- Click STOP → SMC returns to CW, Agilent output off
- Used for: calibration, pre-burn characterization, power sweeps

Tab 3

BLAST Sequence

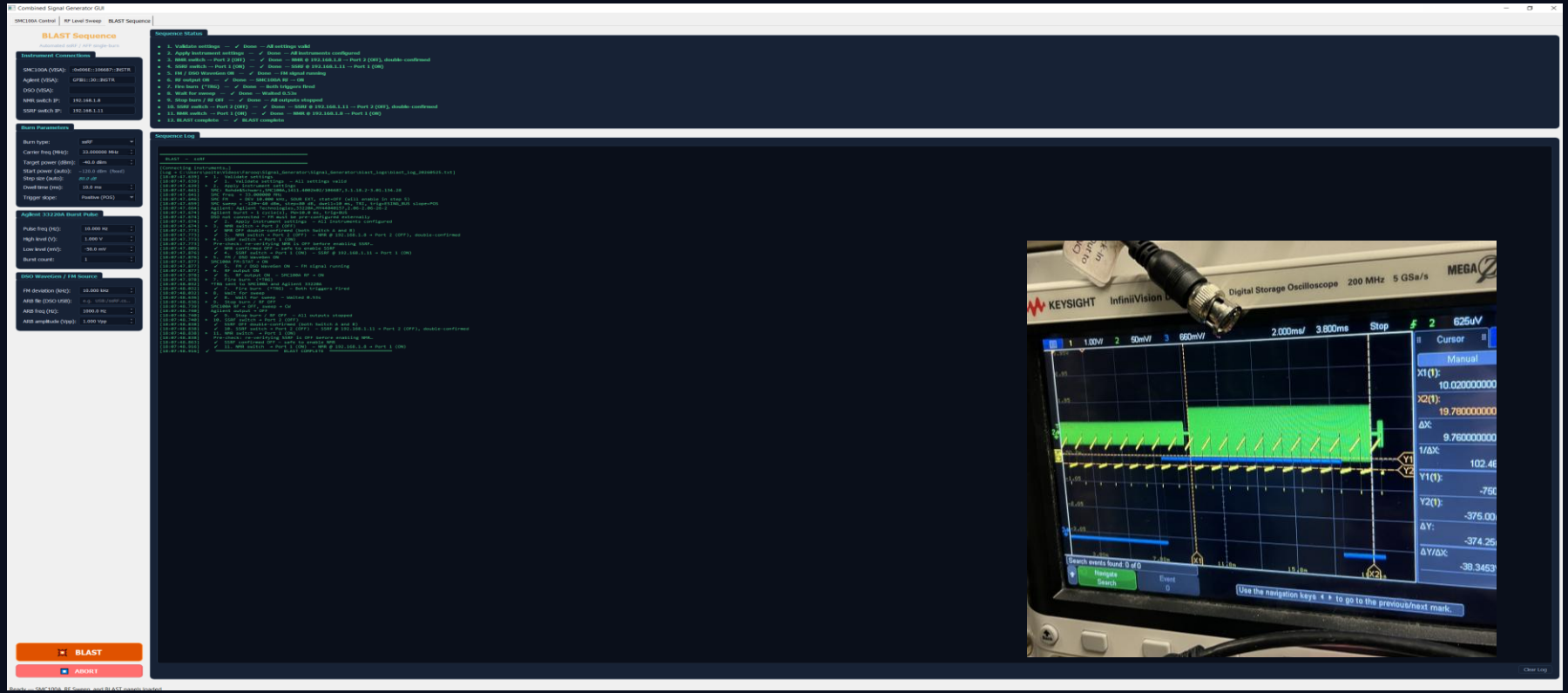
- Fill in: burn type (ssRF/AFP), carrier freq, target power, dwell time
- Verify NMR/ssRF IPs, SMC VISA, Agilent VISA (all pre-filled)
- Pre-load CSV waveform into DSO WaveGen from USB drive
- Click BLAST → 12-step automated sequence executes
- Sequence Status panel: green = done, orange = running, red = error
- Click STOP/ABORT at any time → emergency cleanup runs automatically

ssRF GUI — RF Level Sweep Tab

Real-time RF power ramp preview with TRI sweep shape and Agilent trigger timing

ssRF GUI — BLAST Sequence Tab

12-step automated burn sequence with interlocked switch control and timestamped log output



ssRF GUI — BLAST Algorithm

Decision points & error paths

Press BLAST

Validate Settings

Apply Instrument Settings

NMR → Port 2 / SSRF → Port 1

FM ON → RF ON → *TRG (sweep)

Wait (3×dwell + 500 ms)

RF OFF / FM OFF

SSRF → Port 2 / NMR → Port 1

BLAST COMPLETE ✓

Safety Checks (Double-Verified)

- Step 3: double-read NMR confirms Port 2 → if No: ABORT
- Step 4 pre-check: NMR still on Port 2? → if No: ABORT
- Step 10: double-read SSRF confirms Port 2 → if No: ABORT
- Step 11 pre-check: SSRF still on Port 2? → if No: ABORT

Emergency Cleanup (on ABORT or Error)

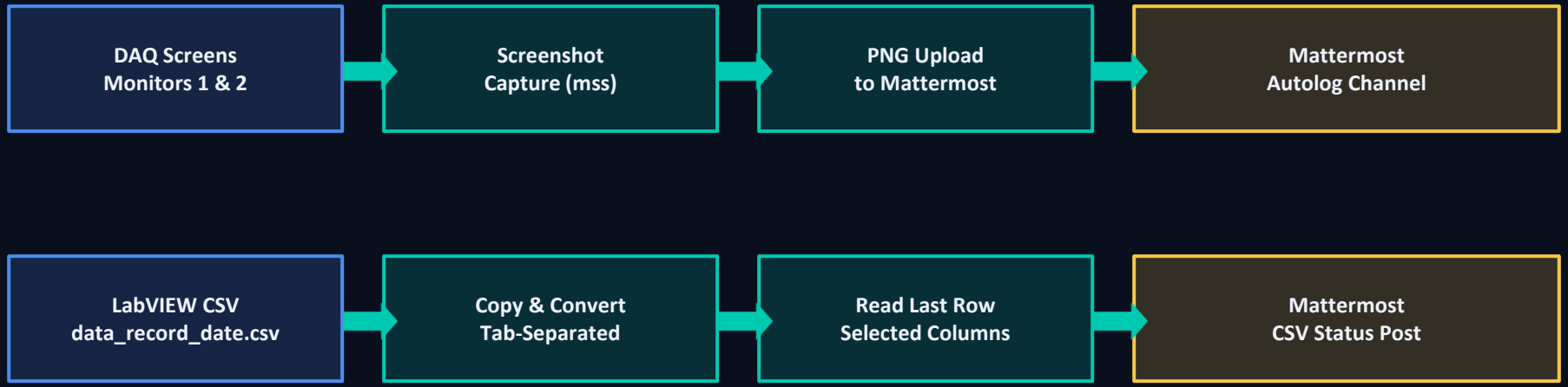
1. SMC100A: stop level sweep + RF OFF
2. SMC100A: FM:STAT OFF
3. Agilent: OUTP OFF
4. SSRF switch → Port 2 (50Ω isolated)
5. NMR switch → Port 1 (measurement path restored)

Note: Thanks to Prof. Long for helping out during developing, implementing, and testing this GUI.

Poll interval: 10 ms | Timeout: 5.0 s | Confirm mask: raw & 0x03 (Switch A & B bits)

Mattermost Autolog Bot — System Architecture

One merged application



Shared Infrastructure: Mattermost REST helpers · Structured logging · Scheduler · Lock file · Stop-file control

Mattermost Autolog Bot — Launcher Application

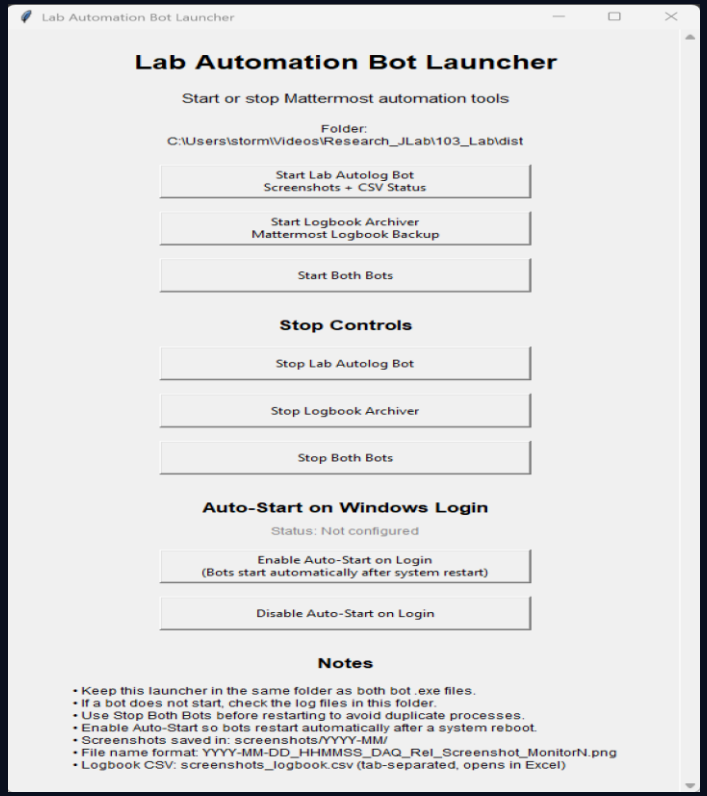
Start/stop both bots with auto-start on Windows login

Start Controls

Lab Autolog Bot (Screenshots + CSV)
Logbook Archiver (Backup)
Start Both Bots

Stop & Auto-Start

Individual or combined stop controls
Auto-Start on Windows Login
Bots restart after system reboot



File Conventions

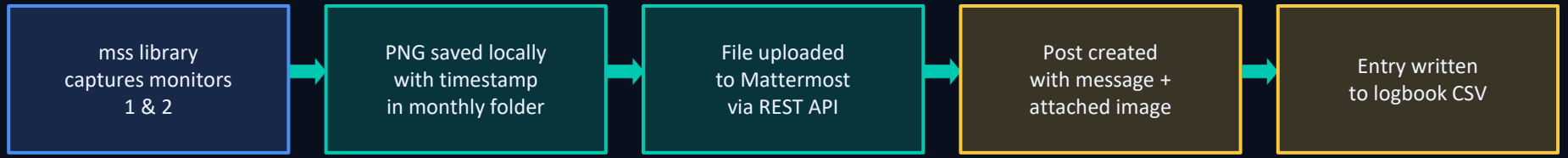
Screenshots saved in:
screenshots/YYYY-MM/

Filename:
YYYY-MM-DD_HHMMSS
_DAQ_Rel_Screenshot_MonitorN.png

Logbook CSV:
screenshots_logbook.csv

Autolog Bot — Screenshot Capture & Upload

Every 15 minutes at exact clock intervals (:00, :15, :30, :45)



Filename & Folder Conventions

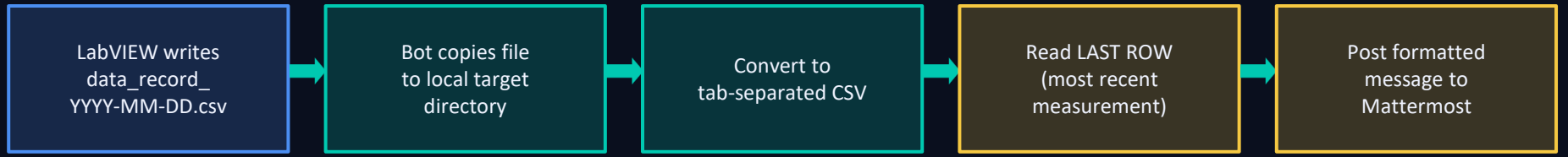
- Monitor 1: YYYY-MM-DD_HHMMSS_DAQ_Rel_Screenshot.png
- Monitor 2: YYYY-MM-DD_HHMMSS_DAQ_Screenshot.png
- Saved under …/Reliq_Screens/YYYY-MM/ (Monitor 1)
- Saved under …/DAQ_Screens/YYYY-MM/ (Monitor 2)
- SAVE_LOCAL_COPY = True → PNG kept after upload
- Monthly sub-folders are auto-created

Timing: Exact Interval Logic

- math.ceil(now / 900) × 900 → next exact 15-min mark
- Sleeps in 5-second steps → checks stop file
- First run: immediately (or wait for interval — config)
- Interval: 900 s (INTERVAL_SECONDS, configurable)
- Thread: ScreenshotWorker (daemon thread)

Autolog Bot — LabVIEW CSV Data Logging

46 parameters posted to Mattermost from each data record



Columns Logged (46 Parameters)

- mmWave Freq, Span, Power, RF Status
- Magnetic field: Central, Span, Fringe, PS/Magnet Current
- Temperatures: M1–M5, F1–F11, T1, T3, Vapor Pressure
- Pressures: Ionivac, Manometer Hi/Lo, Roots
- NMR: Central Freq, Freq Span
- Flows: Separator, Roots
- DAQ: VME Board Temp, Diode/Phase Tune, IF Attenuation

File Handling Details

- Source: Box Sync → autolog/data_record_YYYY-MM-DD.csv
- Target: local folder / YYYY-MM / same filename
- Auto-detect delimiter (tab vs. comma)
- Pad short rows to uniform column width
- Latin-1 → UTF-8 encoding conversion
- Thread: CSVWorker (daemon thread)
- Error posted back to Mattermost if file is missing

Autolog Bot — CSV File Locking & Message Queuing

Robust handling when CSV is open in Excel

Implementation

```
# Queue a row if file is locked
_PENDING_CSV_ROWS.append([timestamp, author,
                          message, attachments])
_flush_pending_csv_rows()  # try immediately

# In flush:
except PermissionError:
    log('File open — queued for next cycle')
```

Why This Matters in a Lab

- Scientists often leave CSV open in Excel
- No data loss — every message is eventually written
- Queue holds unlimited rows in RAM
- _flush_pending_csv_rows() called every 30 s
- Log shows how many rows are queued
- Zero configuration needed from user

Autolog Bot — Handling Edited Mattermost Messages

Updated versions are detected and appended to all archives

Detection Logic (state.json)

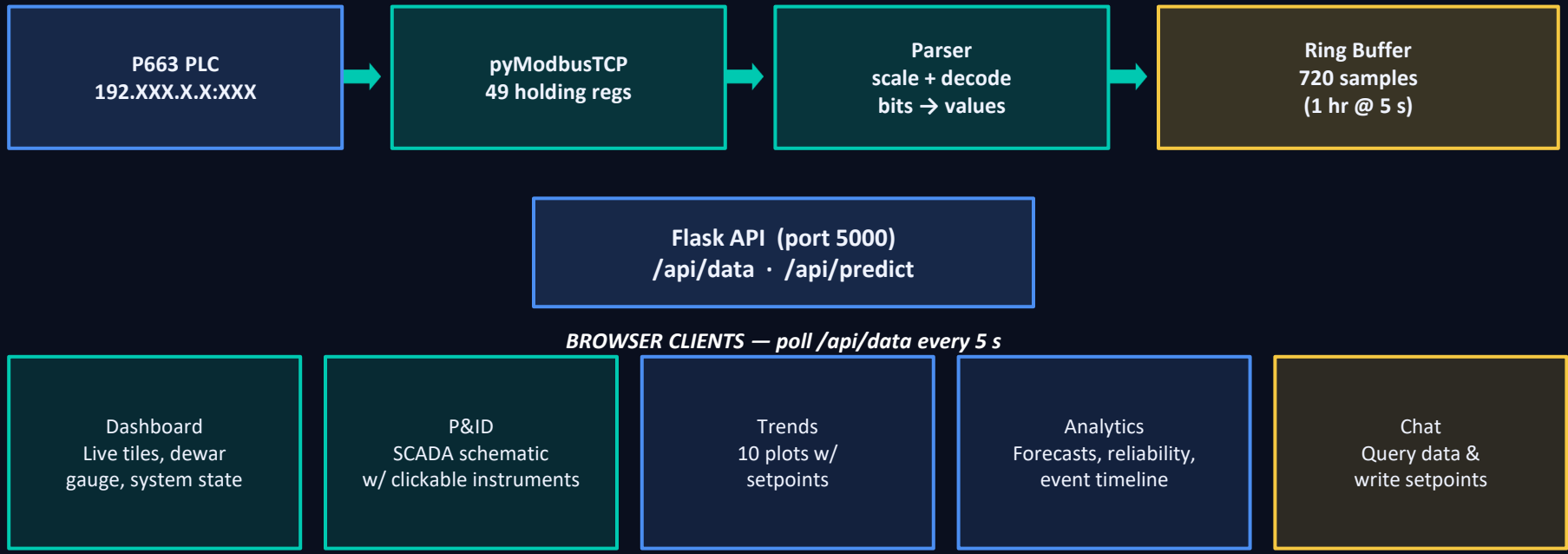
- Every post carries create_at and update_at timestamps
- state.json stores post_update_times{post_id: update_at}
- Each poll cycle: compare stored vs. current update_at
- If update_at > stored value → post was edited
- state.json trimmed to last 20,000 post IDs

Log Output Format

```
========================================
[EDIT] POST ID    : abc123xyz
USER              : Dr. Smith (@dsmith)
EDITED AT         : 2026-05-14 10:33:21
UPDATED MESSAGE   :
Corrected: target temperature was 1.0 K
========================================
# CSV row prefix:
[EDITED at 2026-05-14 10:33:21]
```

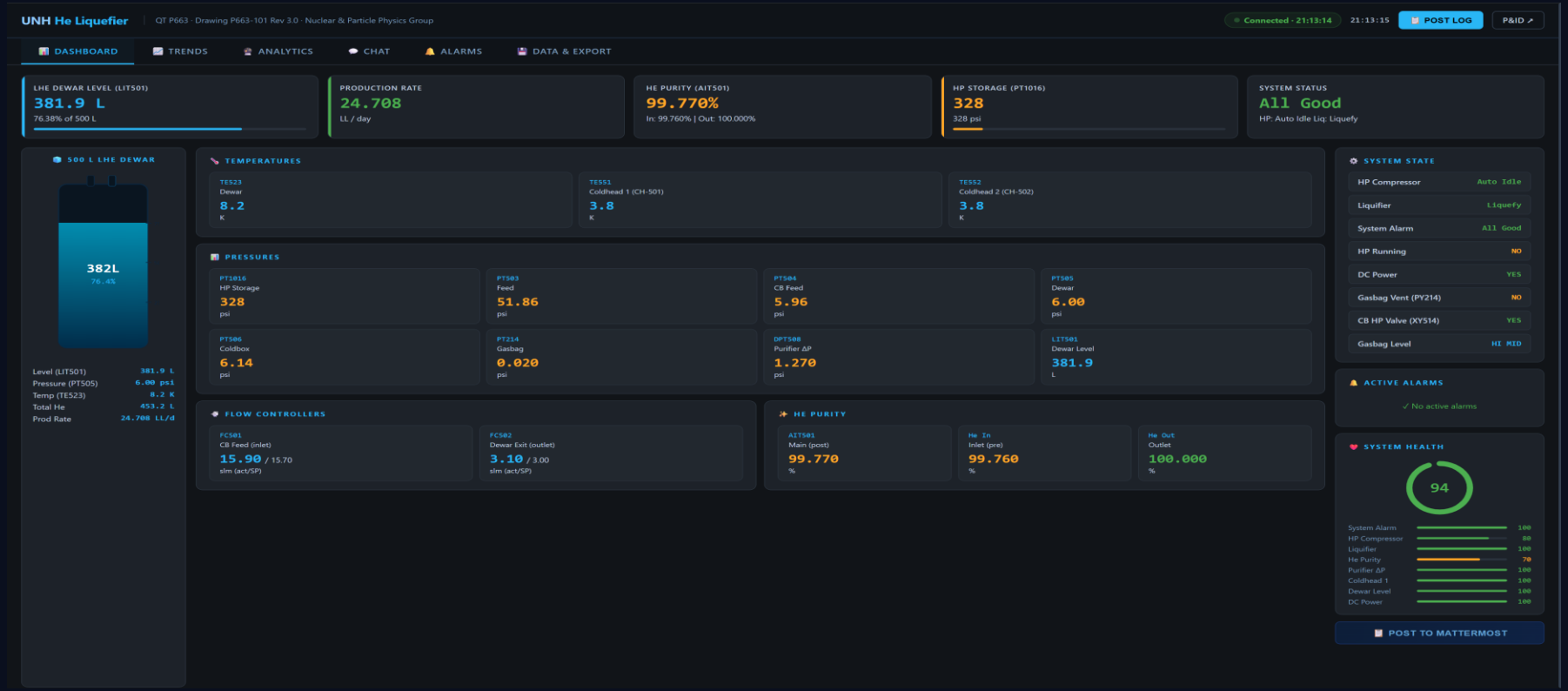
Helium Liquefier HMI — Software Architecture

QT P663 Reliquefier · UNH Polarized Target Lab



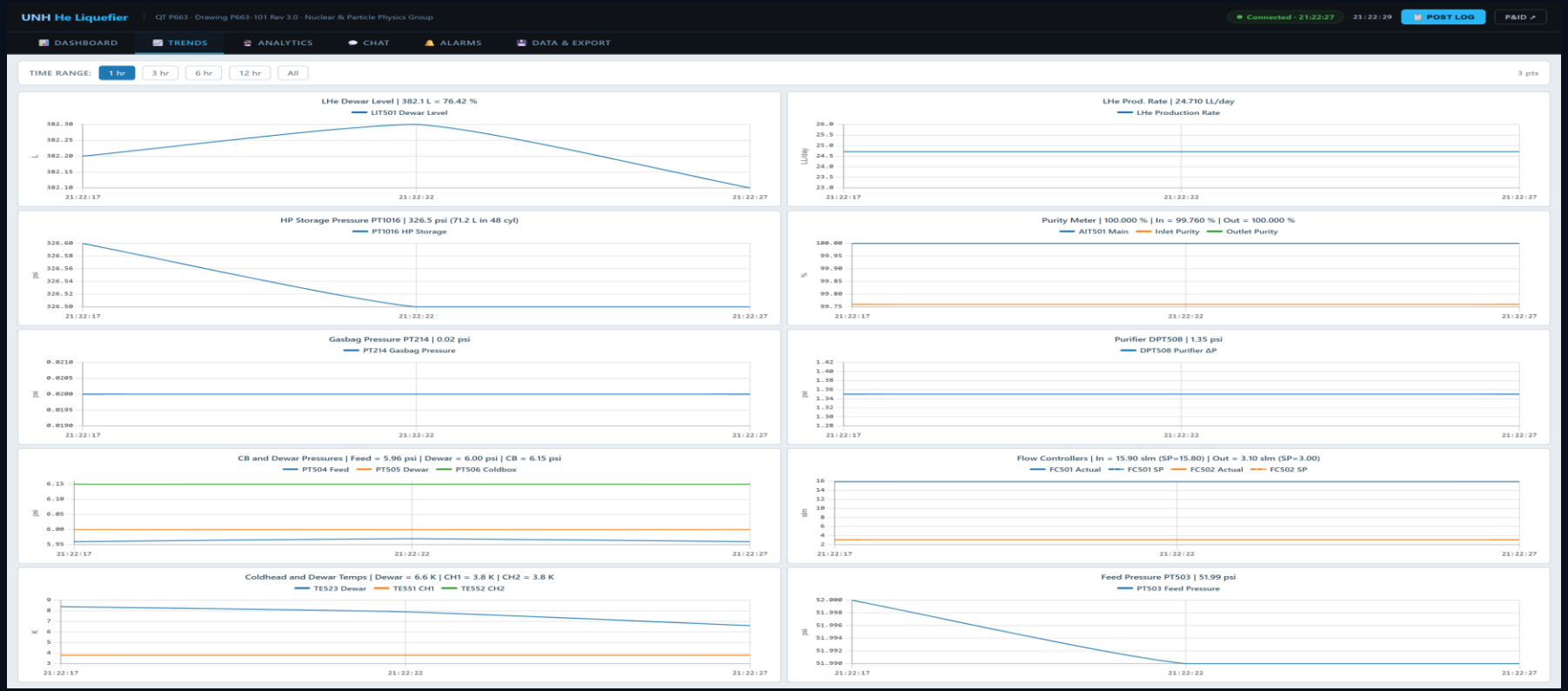
Helium Liquefier HMI — Dashboard

Live tiles: dewar level, production rate, He purity, HP storage pressure, system state, temperatures, pressures, flow controllers



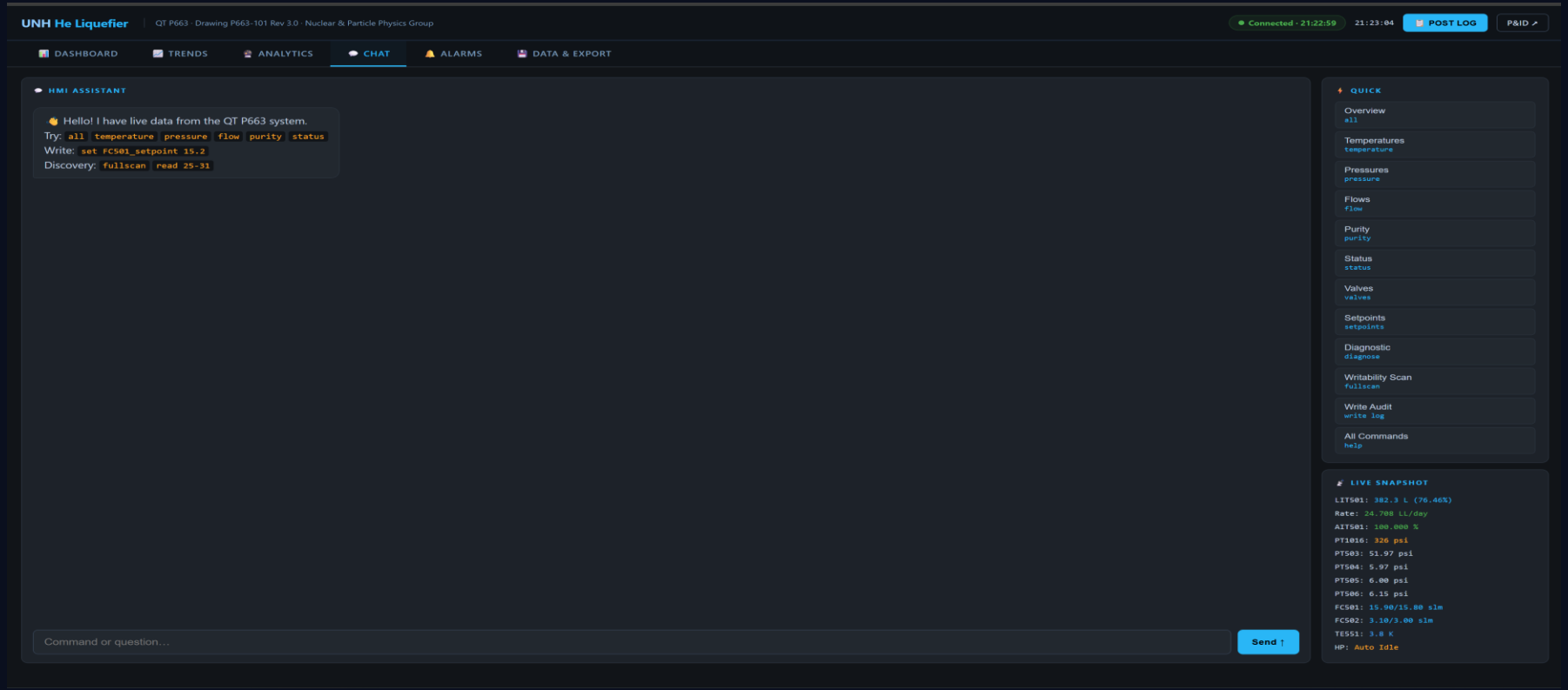
Helium Liquefier HMI — Trends

10 matplotlib-style trend plots with setpoints: dewar level, production rate, purity, HP storage, pressures, flow controllers, temperatures



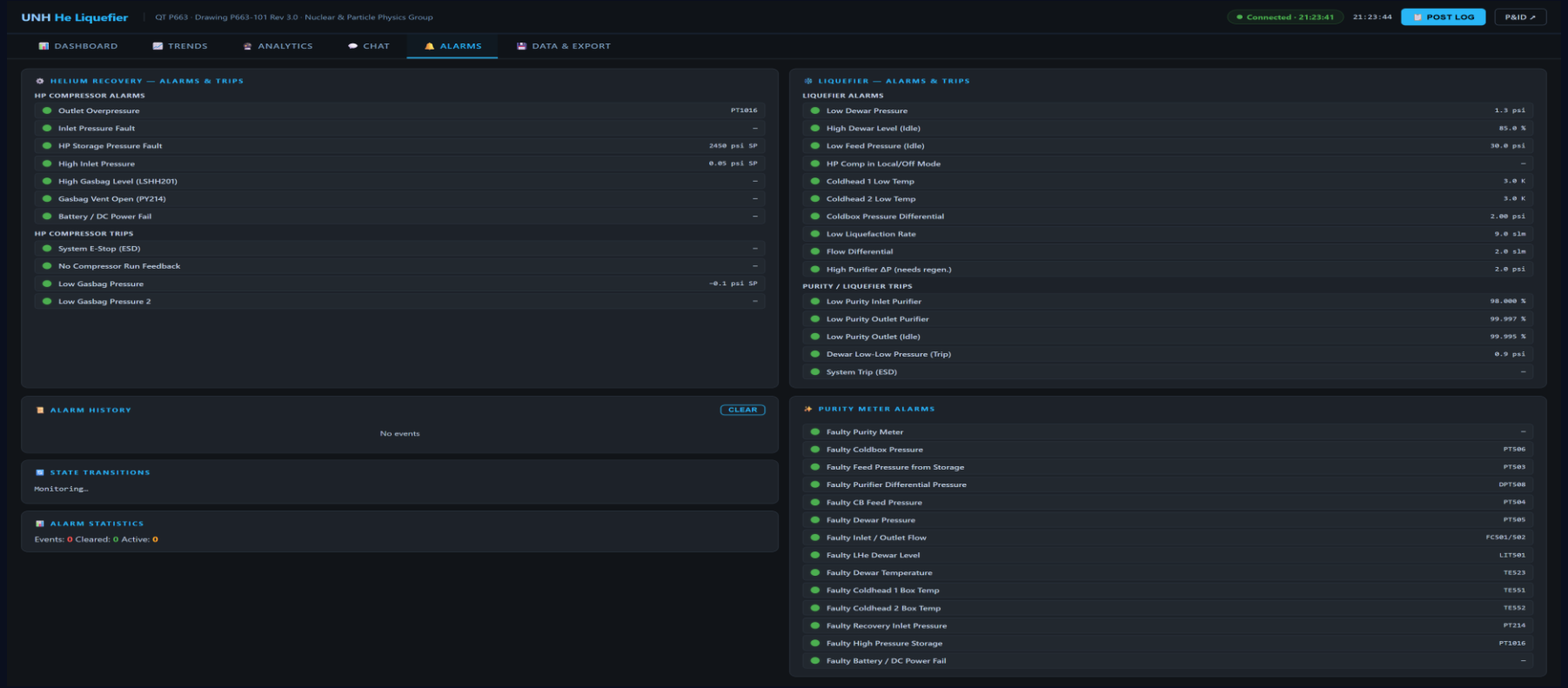
Helium Liquefier HMI — Chat Interface

Natural-language queries and setpoint writes with live snapshot sidebar



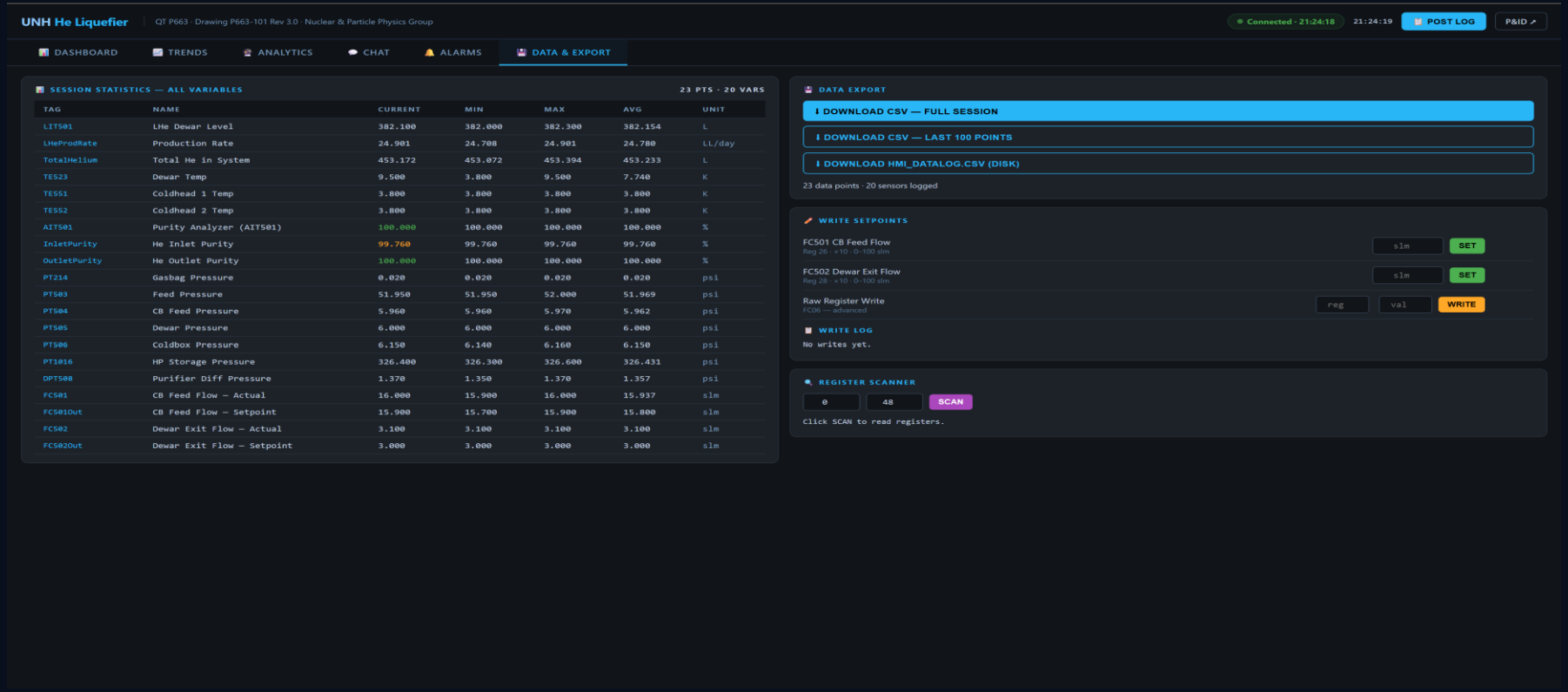
Helium Liquefier HMI — Analytics

Forecasts, reliability scores, and event timeline



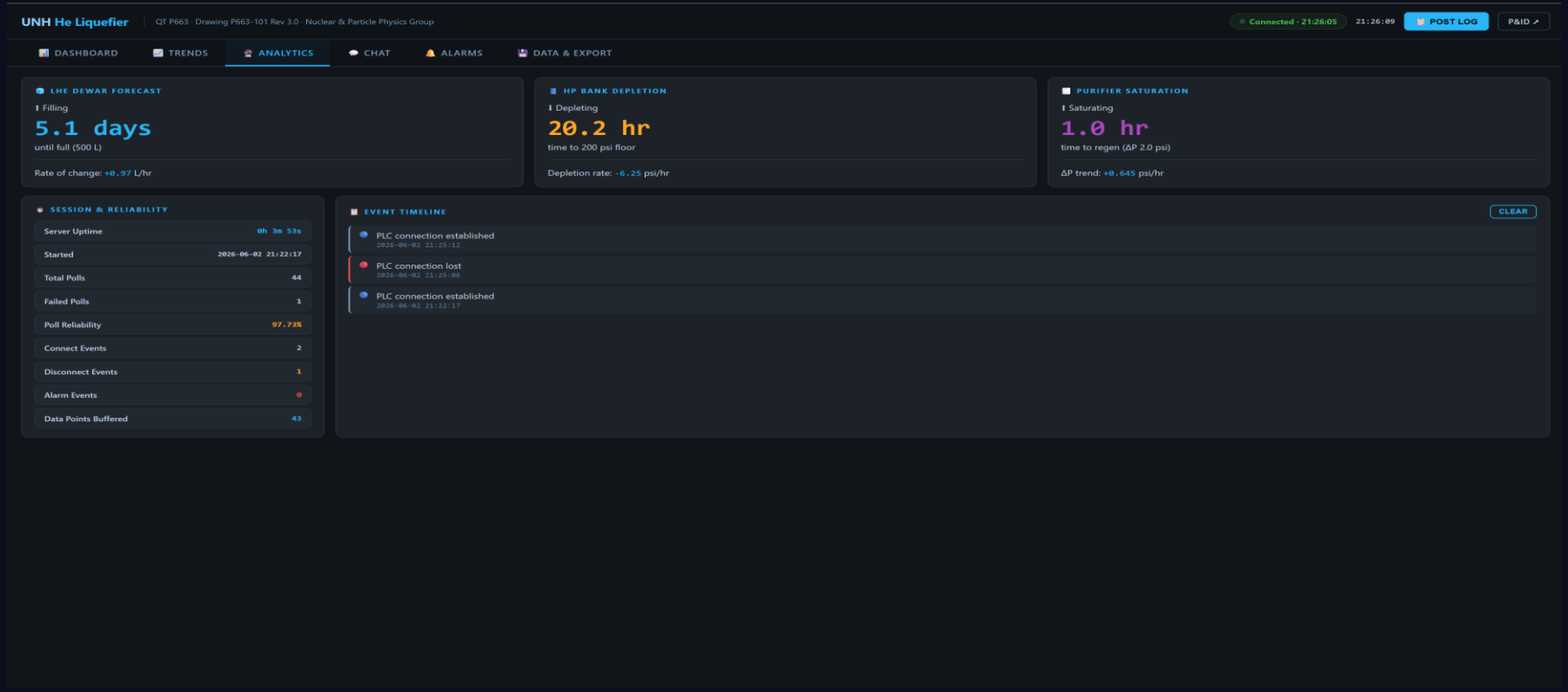
Helium Liquefier HMI — Alarms Panel

Active fault display with bit-decoded alarm names



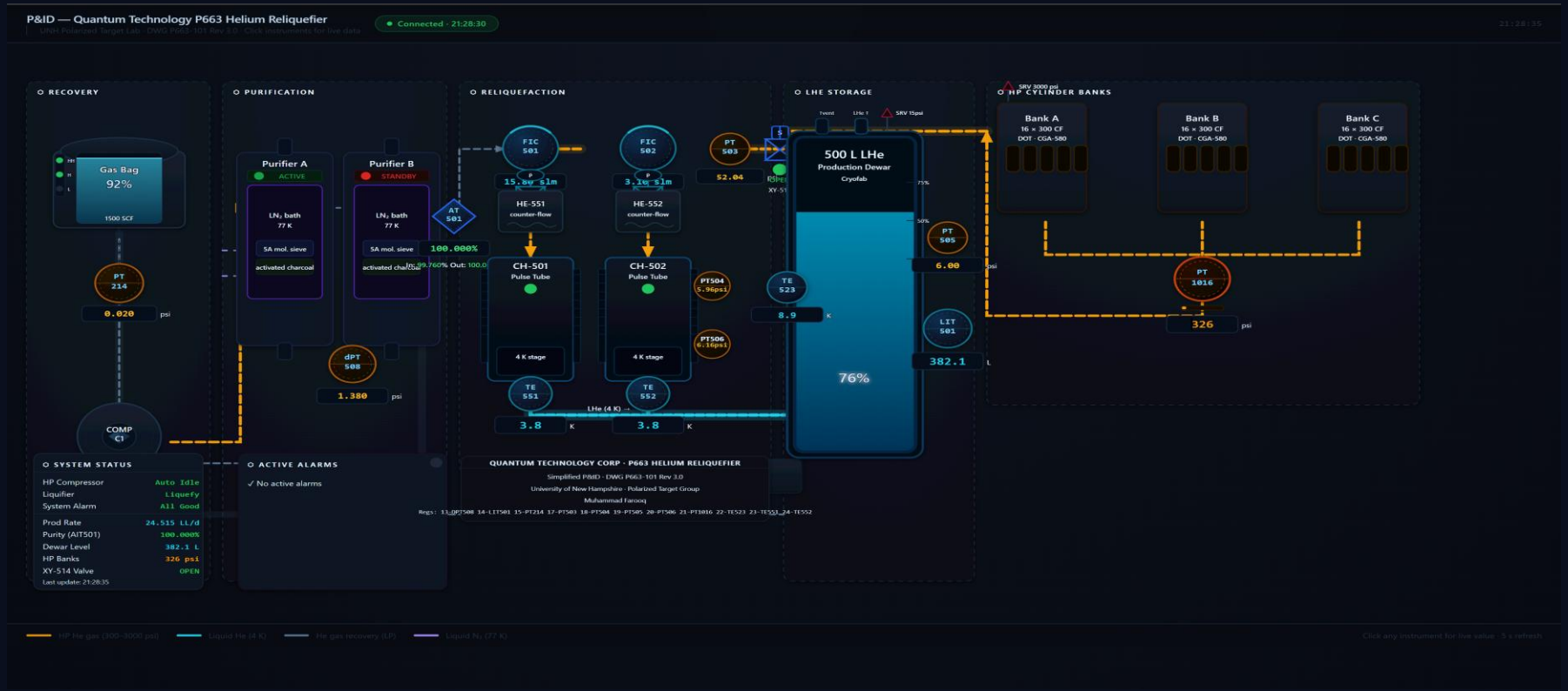
Helium Liquefier HMI — Data & Export

Data export and historical log access



Helium Liquefier HMI — P&ID / SCADA View

SCADA schematic with clickable instruments and blinking alarm indicators



Helium Liquefier HMI — Step 1: Transport

Connecting to the PLC over Modbus TCP

Connection Parameters

Host IP	**192.XXX.XXX.X**
TCP Port	**XXX**
Unit ID (slave)	**X**
VPN	**StrideLink tunnel**
Mode	**auto_open / auto_close**
Poll cadence	**every 5 seconds**

read_hmi_data() — pyModbusTCP client

```
client = ModbusClient(
    host="192.XXX.X.X",
    port=XXX,
    unit_id=X,
    auto_open=True,
    auto_close=True)

regs = client.read_holding_registers(0, 48)
client.close()

if regs is None:
    # PLC unreachable
    connection['connected'] = False

# FIX: datetime → JSON
# custom encoder so jsonify() never 500s
```

Helium Liquefier HMI — Step 2: Decoding

Raw 16-bit registers → engineering units

Reg	Tag	Raw → Scaling	Result
14	LIT501	regs[14] / 10	Dewar level (L)
21	PT1016	regs[21] / 10	HP storage (psi)
13	DPT508	regs[13] / 100	Purifier ΔP (psi)
22	TE523	regs[22] / 10	Dewar temp (K)
25	FC501	regs[25] / 10	CB feed flow (slm)
31	AIT501	regs[31] / 100	He purity (%)

Fixed-point scaling

Integers carry an implied decimal — divide by 10 or 100 to restore the real physical reading.

Sign correction

neg() and ngl() subtract a 16-bit wrap offset so small negative pressures read correctly.

State codes → human text

HPCompState 2 → "Auto Idle" | LiquifierState 4 → "Liquefy" | SystemAlarm 0 → "All Good"

Helium Liquefier HMI — Step 3: Alarm Decoding

Packed alarm words → named faults

Alarm and trip registers are bit-fields. Each register is expanded to binary; every set bit maps to a named condition.

0	1	0	0	1	0	0	0
Battery DC	Gasbag Vent	High Gasbag	High Inlet P	HP Storage P	Inlet P Fault	Outlet OverP	—

regs[4] → format(v, '08b')

HP Compressor

8-bit alarms + 4-bit trips
e.g. Outlet Overpressure, E-Stop

Liquefier

16-bit word, sparse map
e.g. Low Liq Rate, Coldbox ΔP

Purity

Inlet/outlet thresholds
flagged below specification

Active faults surface three ways: Dashboard alarm panel · P&ID blinking red LEDs · Event timeline + Mattermost push

Helium Liquefier HMI — Step 4: Logbook Integration

Posting status & write-audits to Mattermost

Mattermost REST v4 Config

Server	**endeavour.unh.edu**
Channel	**reliq_datalogger**
Auth	**Bearer token**
Notify	**@farooq mention**
Threading	**async, non-blocking**

📋 Full Status Report

Markdown tables covering: system state, 3 temperatures, 7 pressures, 2 flow controllers, dewar & production rate, He purity, valve states, and the full alarm/trip roster by category.

✍️ PLC Write-Audit Log

Every setpoint write logs a row — operator, register, label, old → new value, and a read-back verified flag. A tamper-evident trail of who changed what, and when.

Acknowledgement: Thanks to Prof. Long python script to use the IP address and QT-HMI variables.

Python DAQ — LabVIEW to Python/EPICS Migration

Transitioned half of the LabVIEW DAQ to a Python/EPICS-based system

Soft IOC

caproto serves ~70 named EPICS PVs from a CSV template — pure in-memory data store

Device Bridge

Reads 8 instrument classes via pyvisa/serial and writes values to IOC every 2 s

Fridge Thermo

11-channel LS-224 read over GPIB; CCS calibration; F1–F11 T_K PVs

Magnet Thermo

7-channel LS-224; M1–M5 CCS2; T1/T3 CCX Chebyshev (8-term $\log_{10}$R fit)

Flow & Pressure

Roots flow, separator flow, ionivac, manometer, TPG362 CH1/CH2, manometer low

LHe Levels

AMI-1700 fridge (ASRL7, 115200 baud) and AMI-135 magnet (ASRL5, 9600 baud)

SMT03 NMR

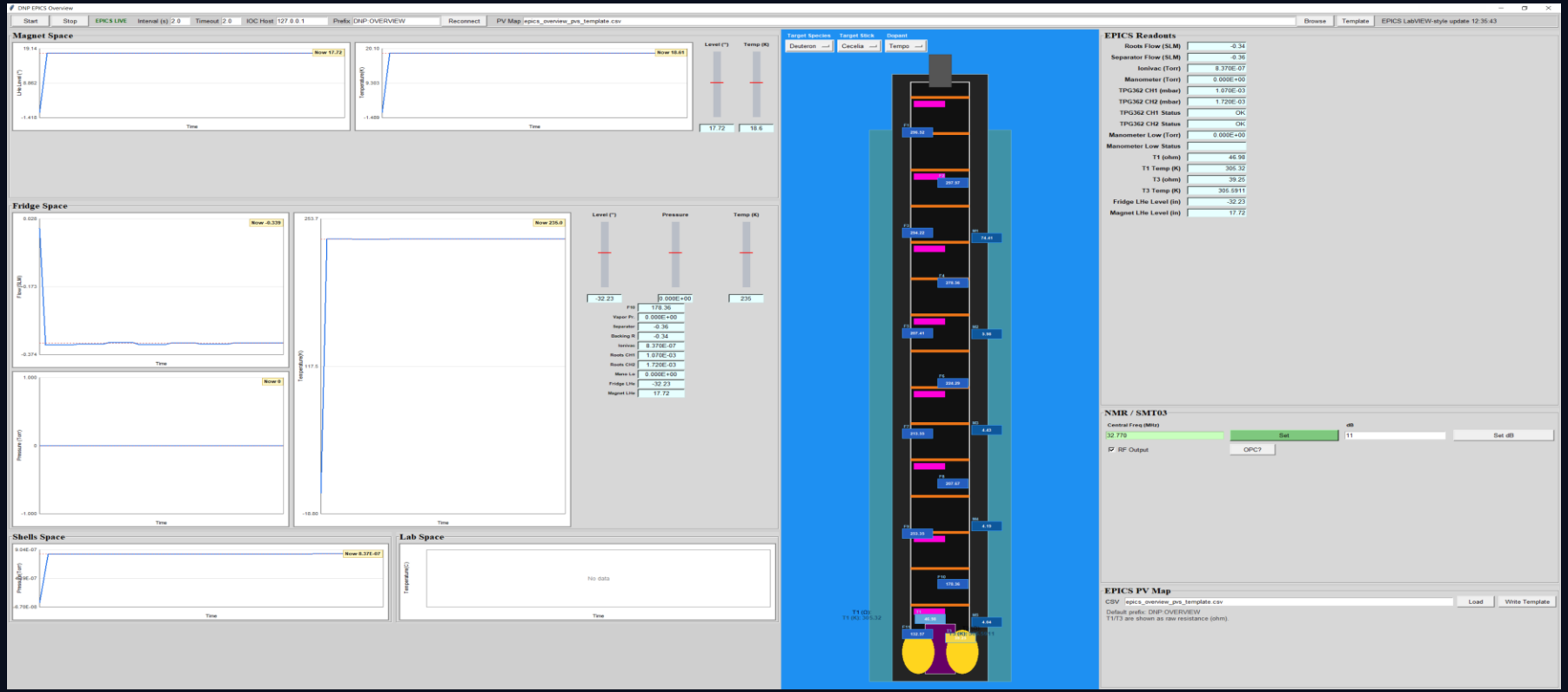
Frequency, level, RF output PVs written by GUI; read-back via OPC? PV

pyEPICS GUI

Two Tkinter views; caget() every 2 s; EPICS_CA_ADDR_LIST configurable at runtime

Python DAQ — pyEPICS Overview GUI

Live EPICS PV readouts, fridge/magnet space trend plots, target stick visualization, NMR/SMT03 controls



Summary & Outlook

Key accomplishments

01 ssRF GUI

Deployed single-operator BLAST system: 5 instruments, 3 buses, hardware interlocks, timestamped logs. NMR preamp protected; ssRF/AFP burns are now safe and reproducible.

02 Mattermost Autolog Bot

Lab monitoring fully automated: 15-min screenshot posts, 46-column LabVIEW CSV status, CSV file-lock queuing, and post-edit detection — zero data loss.

03 Helium Liquefier HMI

Flask/Modbus SCADA dashboard deployed: live P&ID, alarm bit-decoding, ring-buffer trends, Mattermost logbook, and PLC write-audit trail for the QT P663.

04 Python DAQ Migration

Transitioned half of the LabVIEW DAQ to Python/EPICS: caproto soft IOC (~70 PVs), pyVISA device bridge for 8 instrument classes, pyEPICS Tkinter GUI.

Please join the effort

Please join the effort

Please join the effort:

- Karl Slifer (karl.slifer@unh.edu) [Co-Spokesperson of b_1(E12-13-011) Experiment]
  - **E12-13-011: The Deuteron Tensor Structure Function b_1.** Link
- Elena Long (elena.long@unh.edu) [Co-Spokesperson of A_{zz} (E12-15-005) Experiment]
  - **E12-15-005: Measurement of the Quasi-Elastic and Elastic Deuteron Tensor Asymmetries.** Link

University of New Hampshire

UNH NPG

U.S. DEPARTMENT OF ENERGY | Office of Science

This work is supported by DOE contract *DE-FG02-88ER40410*