LZ Slow Control

Tomasz Biesiadzinski
(based on my experience as L2 for LZ Detector Operations)
LNGS, 2025/07/01

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

LZ Slow Control Overview and Scope

- controls, monitors, and records historical data
  - Xenon handling system (circulation, recovery)
  - Detectors (TPC, outer detector): high voltage (grids, PMTs)
  - Calibration (e.g. source deployment system)
  - Cryo infrastructure (LN, thermosiphons) and vacuum systems
  - Backup generators
- monitors and records historical data
  - Electronics
  - Environment
  - DAQ (e.g. trigger rates)
- implements access permission and privilege management
  - System-wide: Everything needs an explicit permission
  - Centrally managed as ‘LZ roles’ which are comprehensive sets of permissions and stored in the Directory
- can generate alarms on any monitored quantity and derived quantities
  - ~ 460,000 ‘tags’ in the system (dominated by HV and DAQ)
- is NOT a personnel safety system
  - Personnel and major equipment safety implemented by proper engineering and passive/autonomous devices (burst disks, fuses, circuit breakers, etc.)
  - No Safety PLCs (no requirement, pointless without safety sensors and safety relays and safety actuators)
  - One of its main roles is to prevent activation of passive/autonomous safety devices
    - Can recover Xenon autonomously (active compressor-based recovery into bottle storage)
- Needs to be highly reliable and provide reliable and secure access from surface and off-site
  - Support our operational model - system experts distributed around the world
  - Underground access may be limited or unavailable

Hardware Components

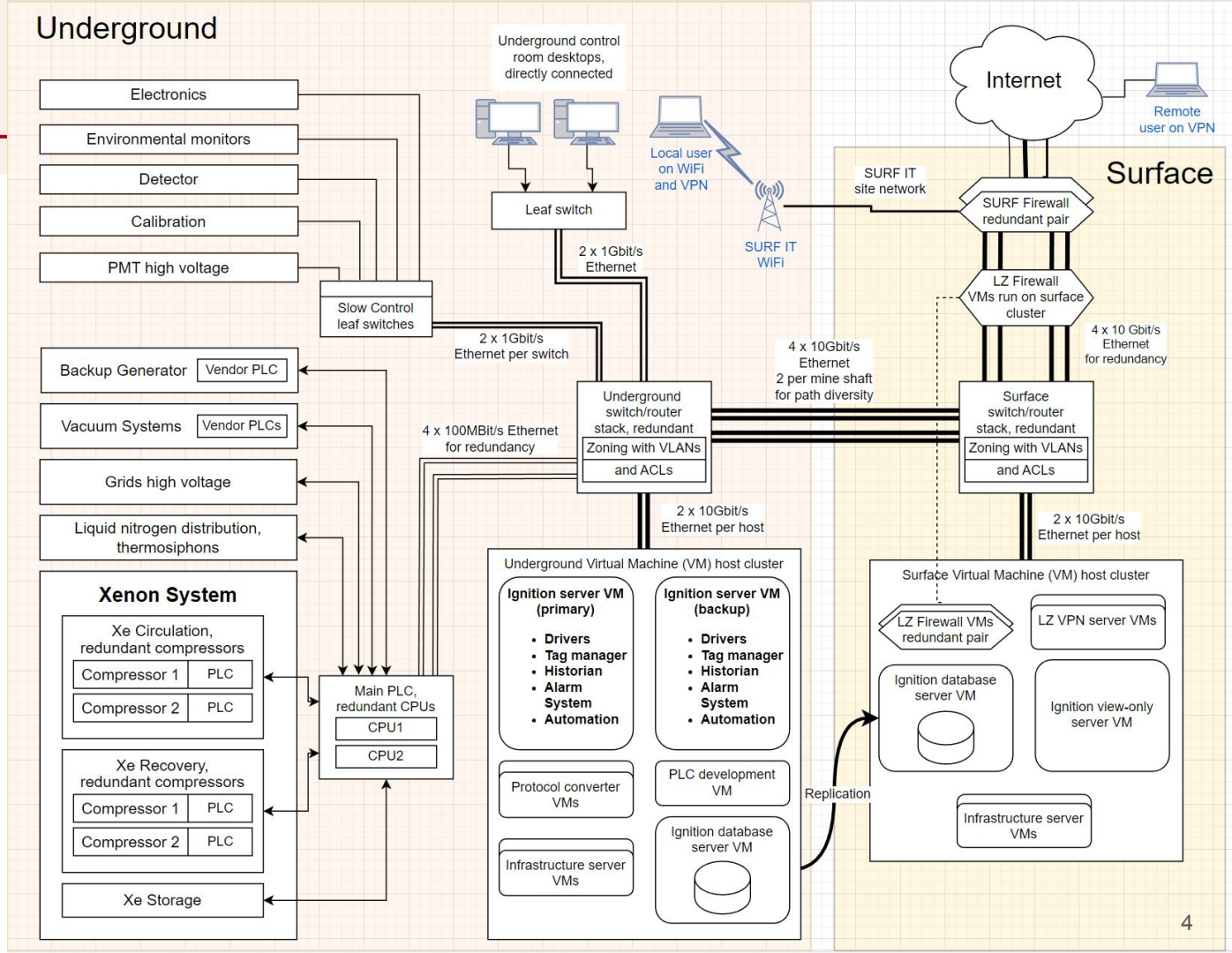
- Siemens PLC maintains operations of critical systems. Primarily:
  - Heating, cooling, insulating vacuum, circulation, Xe recovery (also backup PLCs)
- ADAM IO modules used for non-critical remote applications
- Custom precision sensors (thermometers, level sensors, other sensors)
- Individual stand-alone devices (vacuum pumps, PMT HV, environmental sensors etc)
- DAQ interfaces - PMT rates imported from a DAQ DB; allow alarming

All those communicate with Inductive Ignition (now v8.1)

Also, network infrastructure that sustains the Slow Control server, along with all other online services. I'm not going into great detail here… Just a little bit of detail on the next slide

Infrastructure

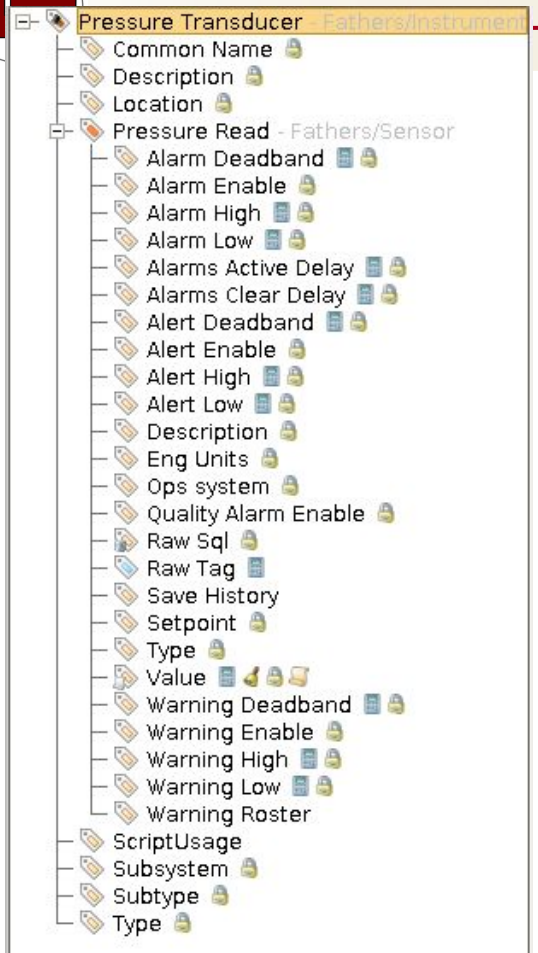
- Underground and Surface hardware
- Underground
  - Focus on redundancy!
  - Virtual machine cluster hosts main ignition, backup ignition and other services on separate VMs
  - Main Siemens PLC (Xe handling PLCs subordinated to it) and non-PLC devices communicate with the cluster via a network switch
  - Control room computers communicate with the cluster via that network switch but don't require internet connectivity
  - Multiple fibers to the surface LZ network switch
- Surface
  - Virtual machine cluster hosts a read-only ignition server, DB replica and other services (including VPN)
  - Firewall
  - Internet connection for remote access



Details On Inductive Ignition

- Commercial Industrial Control System Platform
- Multi-platform for servers and clients (Mac, Windows, Linux)
- Good support (perpetual license, annual support contract)
  - Major recent effort to upgrade to the current version
- Good documentation; easy to get started using and developing
- Communication protocols focused on industrial controls
  - OPC/UA, Modbus/TCP, various PLC drivers, Simple UDP or TCP (very limited!)
  - Device drivers can be written in Java. SDK, ‘advanced topic’
- Redundancy option
- User Interface (GUI/HMI)
- Historian
  - Relational database
  - Documented schema
- Alarm Handling and Notification
  - Email, SMS, voice
- Scripted Controls (Python)
- Comprehensive user and permissions management

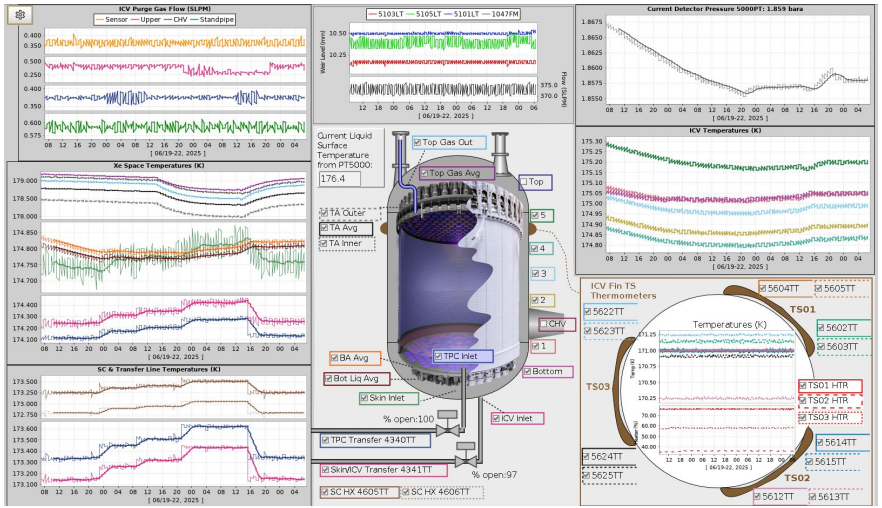
LZ implementation of ignition: instruments and scripts

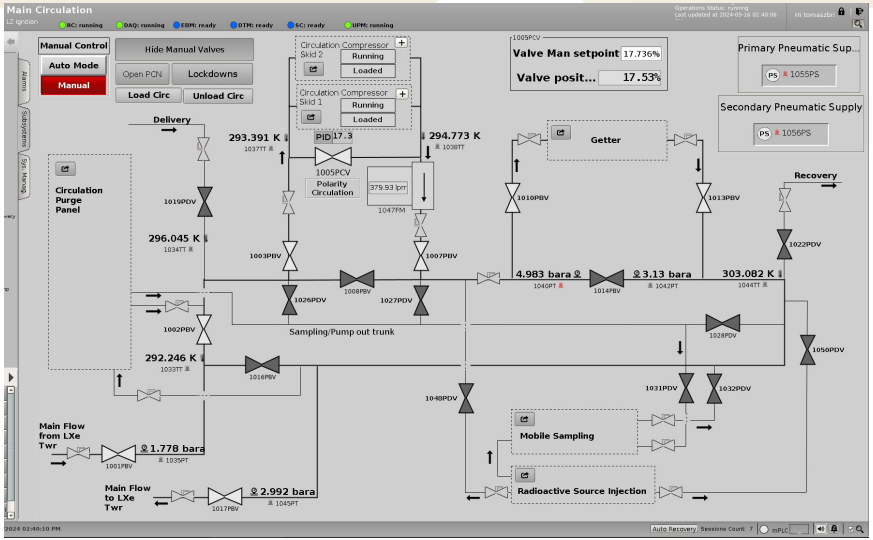


- An Instrument can be a single pressure transducers, or a 32 channel HV crate
  - Instruments using “nested” UDTs (User Defined Types)
    - Designed for individual instruments however composed of more basic types
    - Lesson: Standardization helps BUT sometimes too “one size fits all”, especially with feature creep for some specialized ones
    - Lesson: Sometimes NOT standardized enough - once ignition gets a value for temperature, it shouldn’t care about the readout type, for example
  - Data can be read and saved at differing rates
    - Lesson: someone has to specify these at some point, and not be too greedy
- Scripts used to automate processes (do things when a button is pressed; always do things in background; configuration of ignition itself)
  - Lesson: Document better!

LZ implementation of ignition: user interface

- Three rough flavors of pages
  - “P&ID” like controllable plumbing diagrams
  - Crate-like electronics interfaces
  - Plot-heavy monitoring pages
- Separate windows for interlock and lockdown interfaces
- Very easy to get started





- Lessons:
  - Developers NEED to be trained; hard to do with students and postdocs volunteering their precious time
  - Consistency is important
  - But, sometimes one size doesn’t fill all (what does “Auto mode” vs “Manual” mean for a given page
  - Color selection is important. My personal hot take, despite what guides might say, gray on gray is NOT good (“How do I tell if a button is pressed in?”)

Interfaces and boundaries in LZ

My view is skewed by operations, I'm less familiar with pre-commissioning phases

- Subsystems own instruments
  - They communicate what they need. Often an afterthought that comes late, followed by frustration and quick solutions
  - They **should** work with controls to, if possible, use "easy-to-control" instruments, and if not, to help develop the interface/driver/script
  - Lesson: be realistic about what you think you can do. Don't plan to have automation IF you don't know how to automate it. Maybe do it in the future, if possible and resources are available
  - But, lesson: If you know what you want, do it early when there is time
- Subsystems find problems and need to work with controls to diagnose if it's a software problem, or a hardware problem
  - In LZ, coordination often occurs outside of higher level coordinators, in private channels between subsystem operators, PLC programmer and slow control developer. Not what I'd like a priori, but quick when coordinators may be too busy to quickly respond.
- Process control narrative (PCN) documents used by PLC logic
  - Lesson: something similar should be used for ignition logic/scripting
  - Lesson: Documents don't differentiate between aspiration/to-do vs what's implemented clearly - sometimes things get missed
  - Lesson: Sometimes hard to interpret - need more human legibility if this is the ONLY reference for someone to know what's happening
- Lesson: Major disconnect between developers and system owners. Info gets lost in assumptions, background information (like, what is this thing?) doesn't get recorded.

General Lessons Learned - My view alone, feel free to disagree

- Differences between pre-commissioning, commissioning and operations
  - Pre-commissioning: Time for careful work but a lot of it used up by grand ideas for what slow control can do that then can end up wasting time
  - Commissioning: rush to get instruments online. Turns out they're broken; quickly get something in that we will fix later (we won't fix it). Turns out we missed instruments, add something quickly
  - Operations: Quick bug fixes and "slow" instrument additions and upgrades. Attempts to fix things left over. **Not enough experienced people with enough time to do the work and train (students and postdocs who rotate in and out)**
- A divide between the PLC and the rest of the slow control born out of different group responsibilities. Not good. Hard for other subsystems to track what they need to do
- Ticket system to track "requests". Great in principle, but when we get busy, tickets don't get checked and we revert back to slack.
- Early, logical subsystem divisions (based on operations, NOT WPS) should be built in. Will help with monitoring and alarming
- **Challenges in using a commercial software solution BUT, in my opinion, it was still better than trying to make our own thing**