Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie
AGH University of Krakow

Evolution of the ATLAS TRT Gas Gain Stabilization System Software

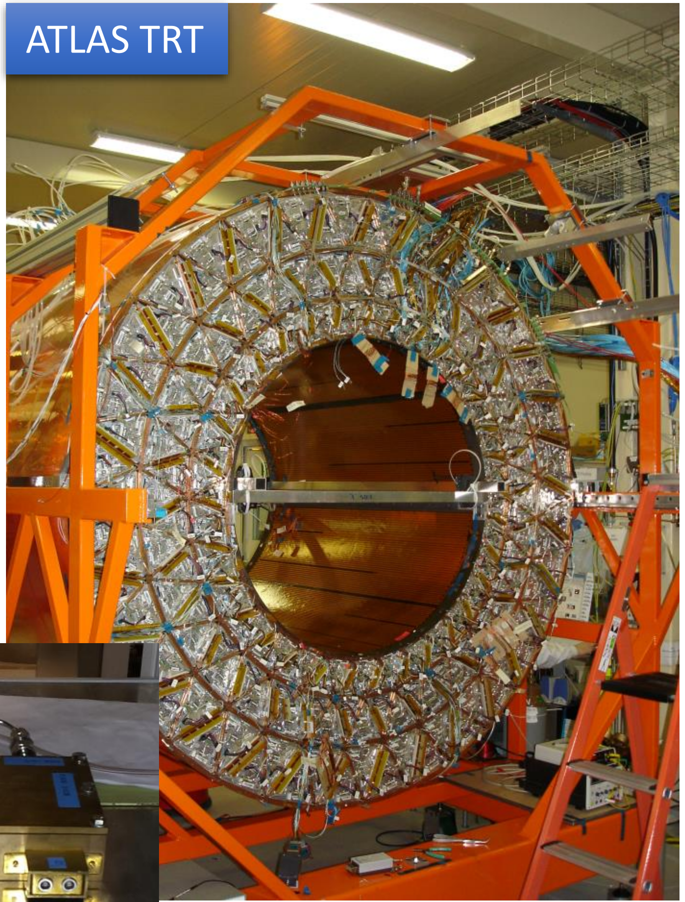
Bartosz Mindur
AGH University of Krakow

Conference on Computing in High Energy and Nuclear Physics
CHEP2024, 21.10.2024

Outline

Outline

- System overview
- Design and evolution
  - Pre LHC
  - Run1
  - LS1
  - Run2
  - LS2
  - Run3
- Conclusions
- Acknowledgements



System Overview

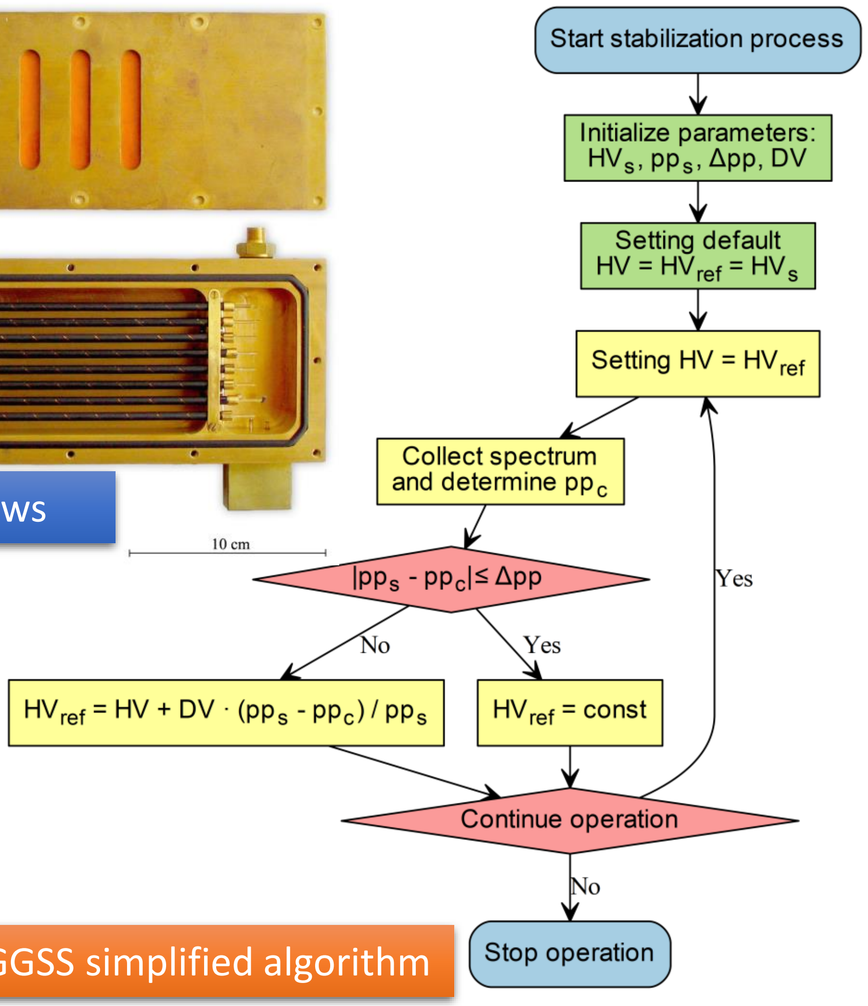
Requirements for GGSS

- Continuous operation
  - All the time, not only during data taking or stable beams
  - Producing reliable HV value for keeping the gas gain constant in the ATLAS TRT detector
  - Monitoring the quality of active gas mixture
- Failure resilient
  - Hot swappable hardware spares
  - Software able to detect and react automatically for (almost) any problem
  - Build in alarm system
  - Build in logging system
- Easley monitored by non experts
  - Clear messages and actions for shifters
- Build using (mostly) standard hardware components
  - Easy for repair or replace
- Compact design
  - Fit into a single rack in USA15
- Long term stable operation
  - Data recorded by GGSS to be used for validation of operation conditions over the decades

GGSS system overview

- Compact system located at USA15 cavern
- Operated by a single PC
  - Custom software connected to central ATLAS DCS
  - Dedicated hardware
    - Nonstandard drivers
- Active elements
  - Two boxes with straws
  - Measured constantly 24/7
  - Flushed by the TRT active gas

GGSS box with straws



GGSS simplified algorithm

2002 -2004

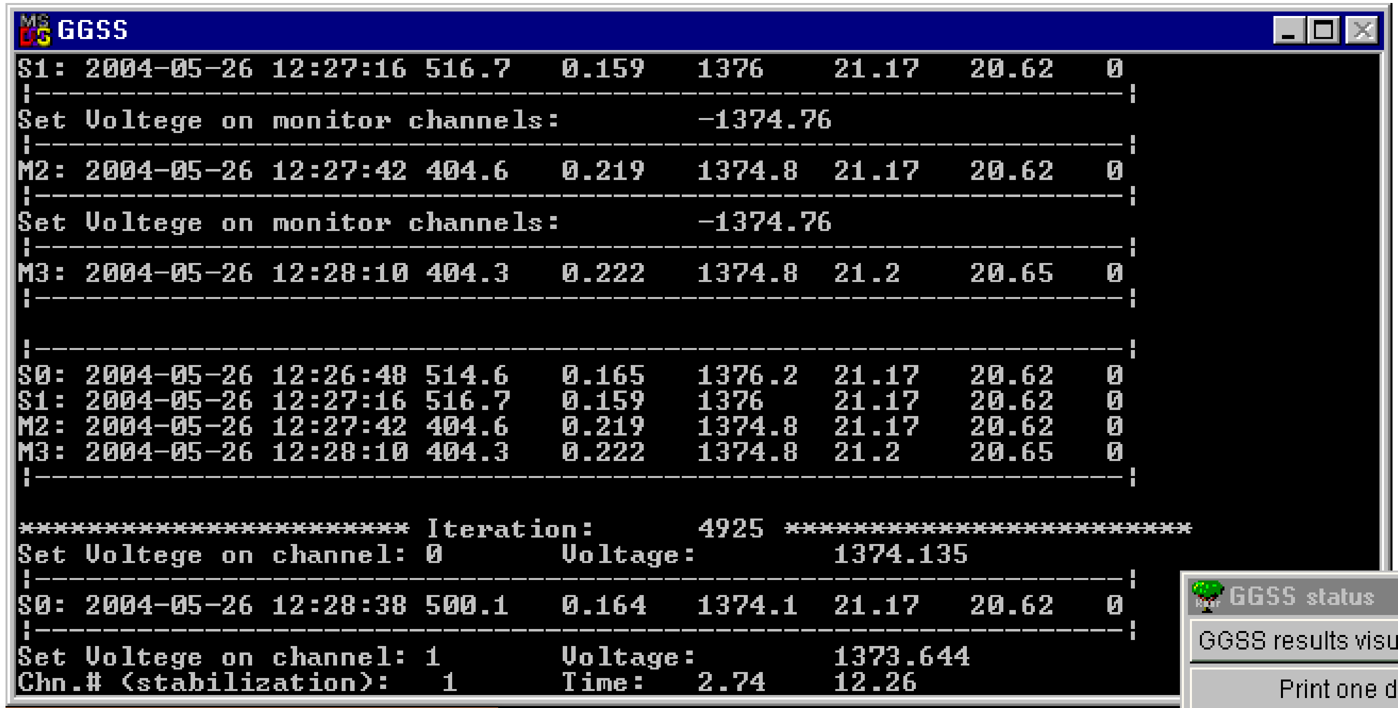
System for CTB 2004

- Software was a set of independent C++ Windows console programs
  - Platform dependent (e.g. threading)
  - No automatic running
  - Controlled by operator using “desktop shortcuts”
- MySQL database
  - Stored all recorded data
  - Provided inter-process “communication”
- ROOT based standalone visualization
- PCI hardware components
  - Drivers only for Windows XP
- System not suitable for 24/7 running
  - No internal safety build-in

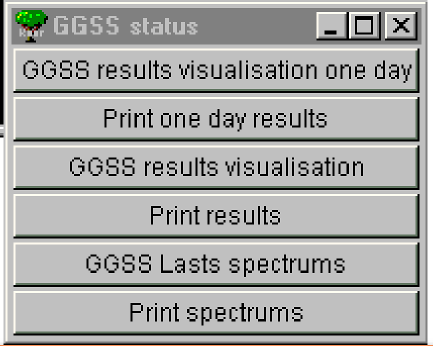
Lines of code

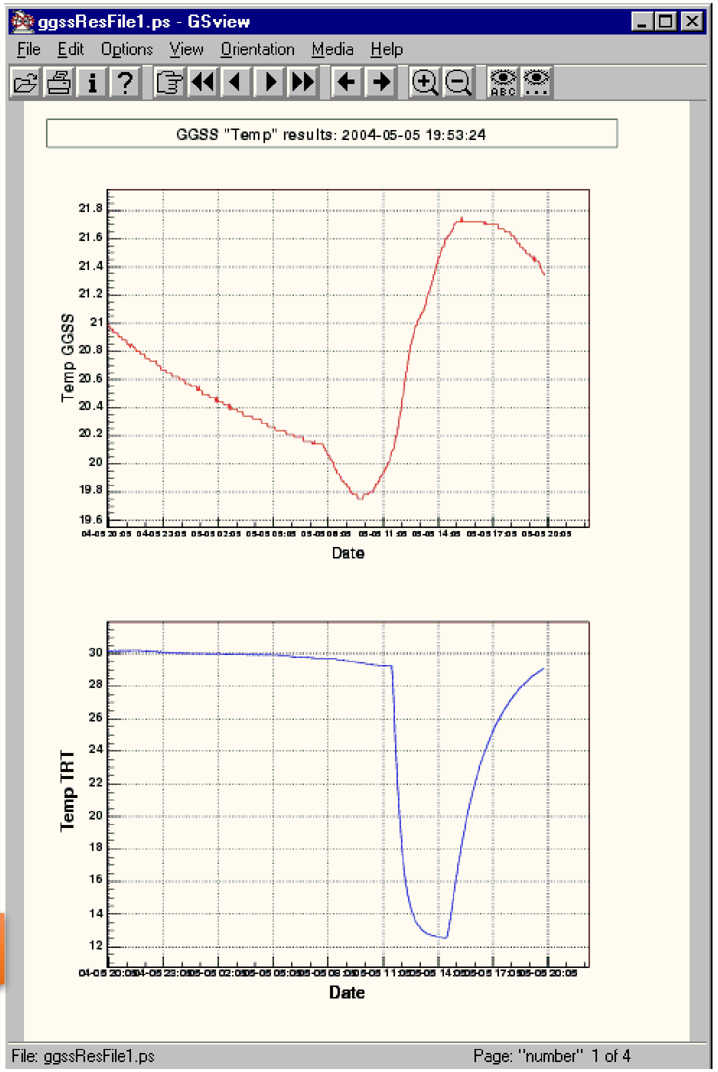
Language	Files	Code	Comment	Blank	Total
C++	43	3,762	558	1,278	**5,598**
Makefile	1	199	17	74	**290**
C	1	103	18	43	**164**
Properties	2	67	18	16	**101**

First Run During CTB 2004



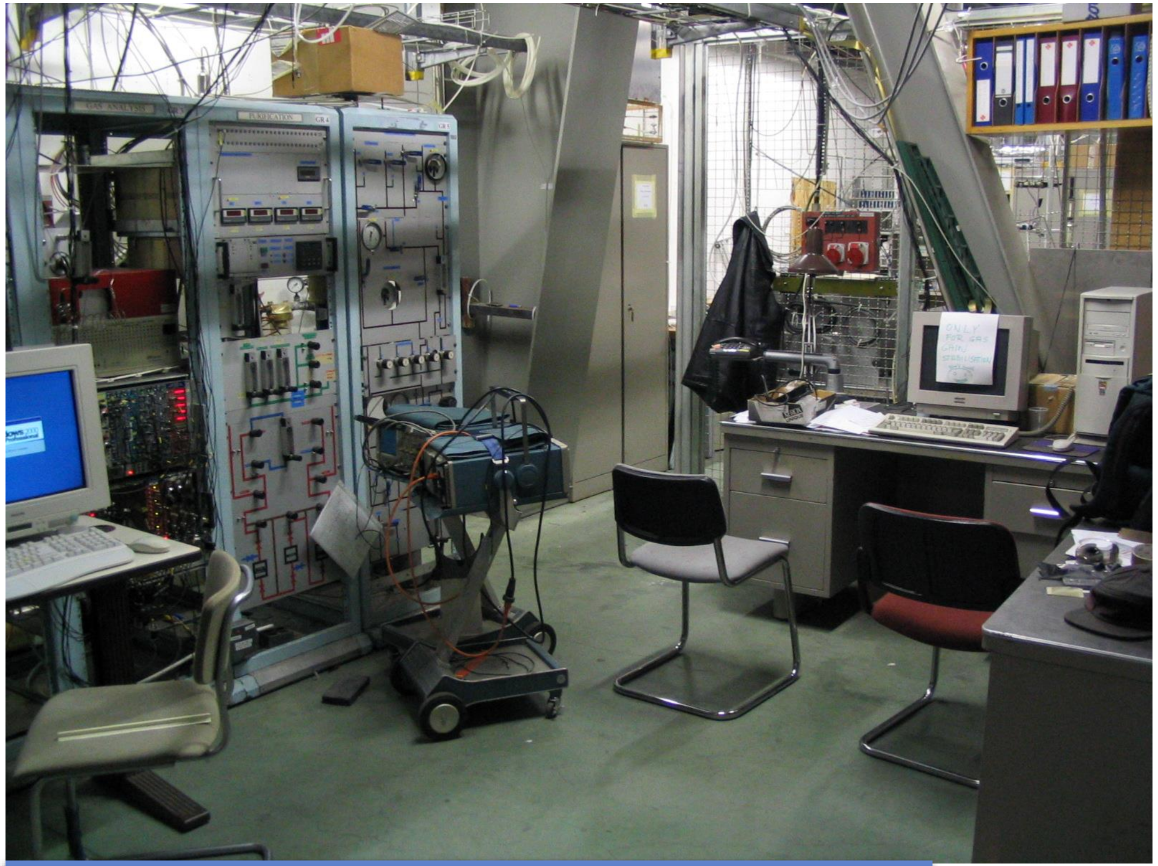
GGSS console application





GGSS standalone visualization

GGSS first run during CTB



SPS H8 beam line control room

PC running first version of GGSS

CTB 2004 outcomes

- Hard to control many independent application
- Safety must be build-in internally into the control application
  - The system must handle (almost) all failures by itself
- Application must be well prepared for long runs in 24/7 regime
- Reduce external dependency by removing not crucial dependencies
  - ROOT
  - MySQL
- Provide independent online and offline data access and visualization
- Provide efficient massage deliver in case of nonrecoverable critical failure

2004 -2008 (2010)

GGSS software for Run1

- Complete rebuild of the whole software
  - Still platform dependent
  - Sources controlled by CVS
  - No automated tests
  - MS Visual Studio build system
- Platform Windows XP 32-bit
  - Single application
  - Run as service
  - Loging to Windows event system
  - Communicate via DIM
    - Distributed Information Management System, https://dim.web.cern.ch/
- Interface and control
  - WinCC OA
  - Finally integrated with ATLAS DCS

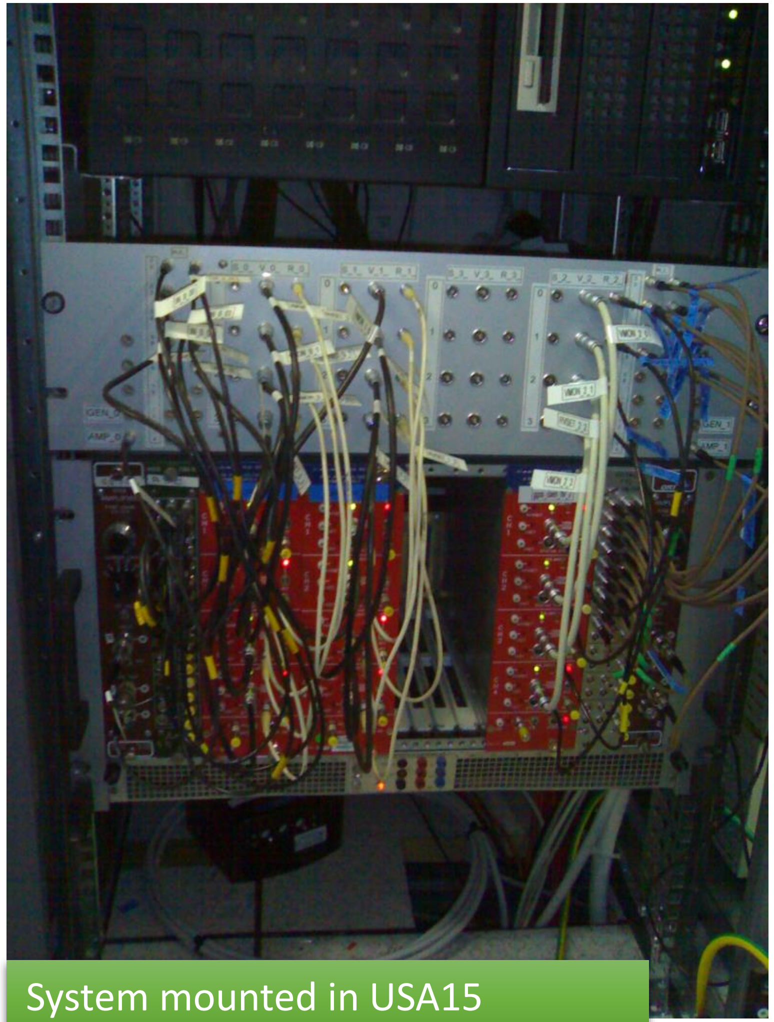
Lines of code

Language	Files	Code	Comment	Blank	Total
C++	462	60,911	19,916	14,705	95,532
HTML	31	3,163	0	1	3,164
XML	10	799	3	8	810
Visual Basic	10	644	620	210	1,474
Properties	1	98	0	34	132
Batch	3	5	1	0	6
Shell Script	1	1	0	1	2

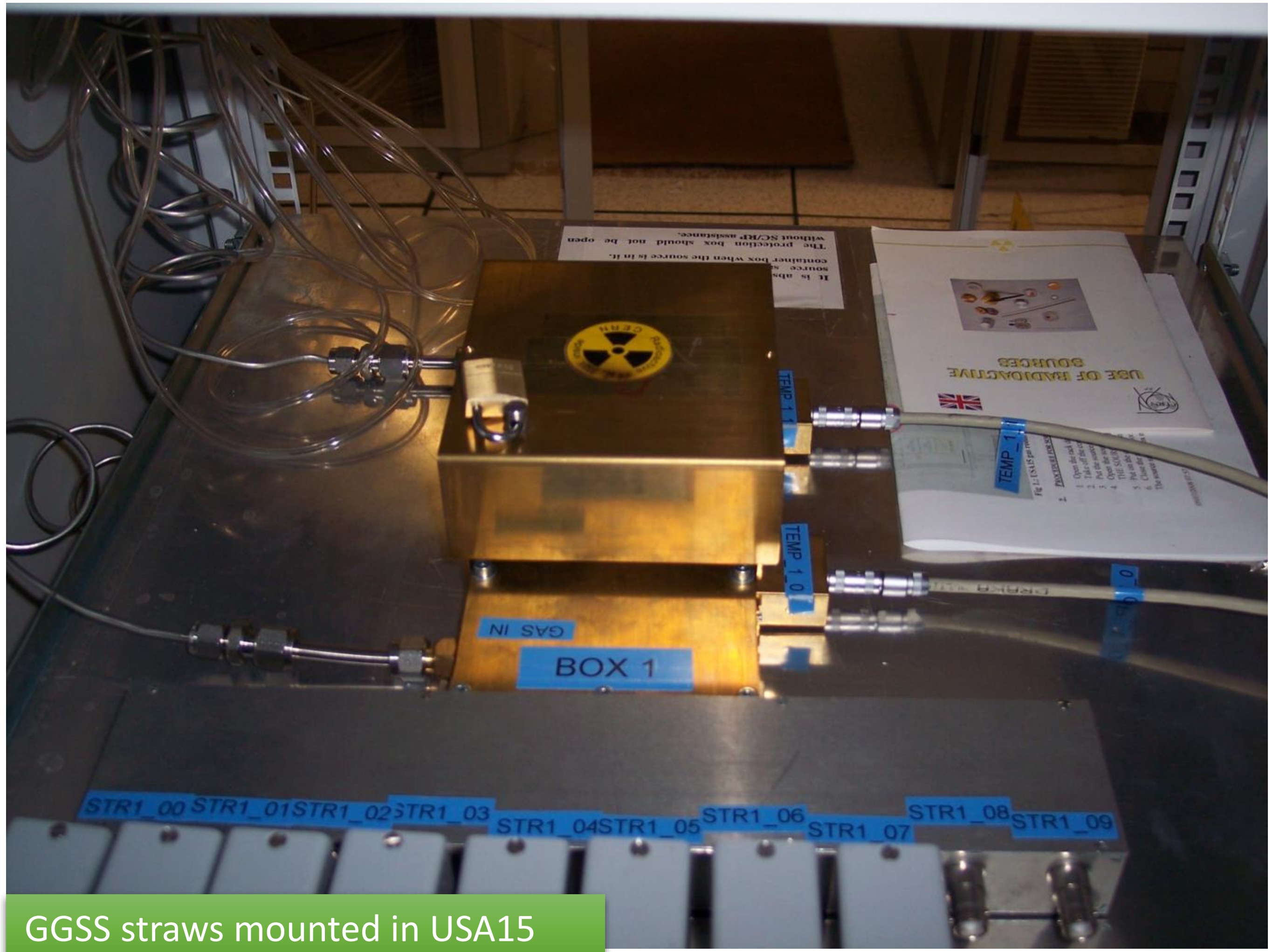
External dependencies

- Boost ver. 1.35
- GNU Scientific Library (GSL) ver. 1.11
- Hardware dependent (MEN, Ortec)

GGSS System in USA15

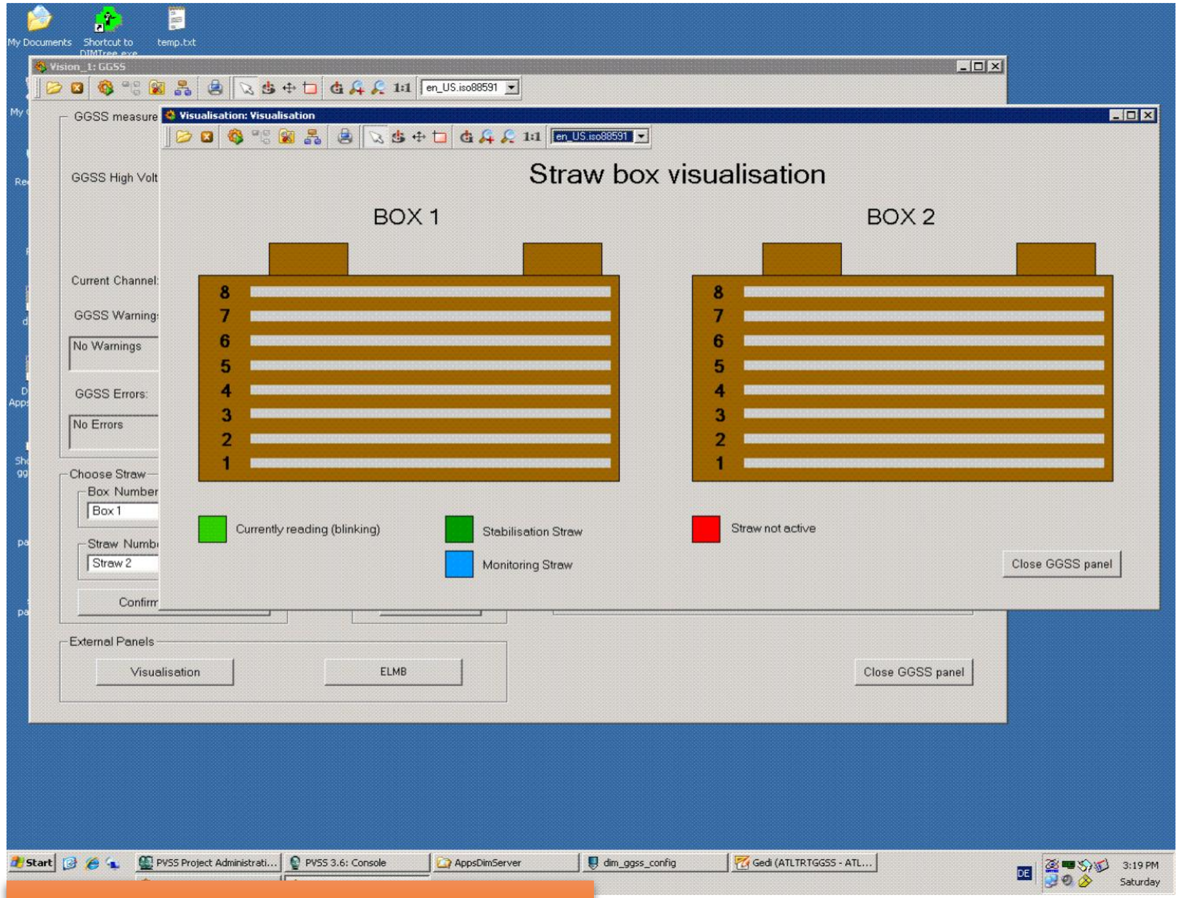


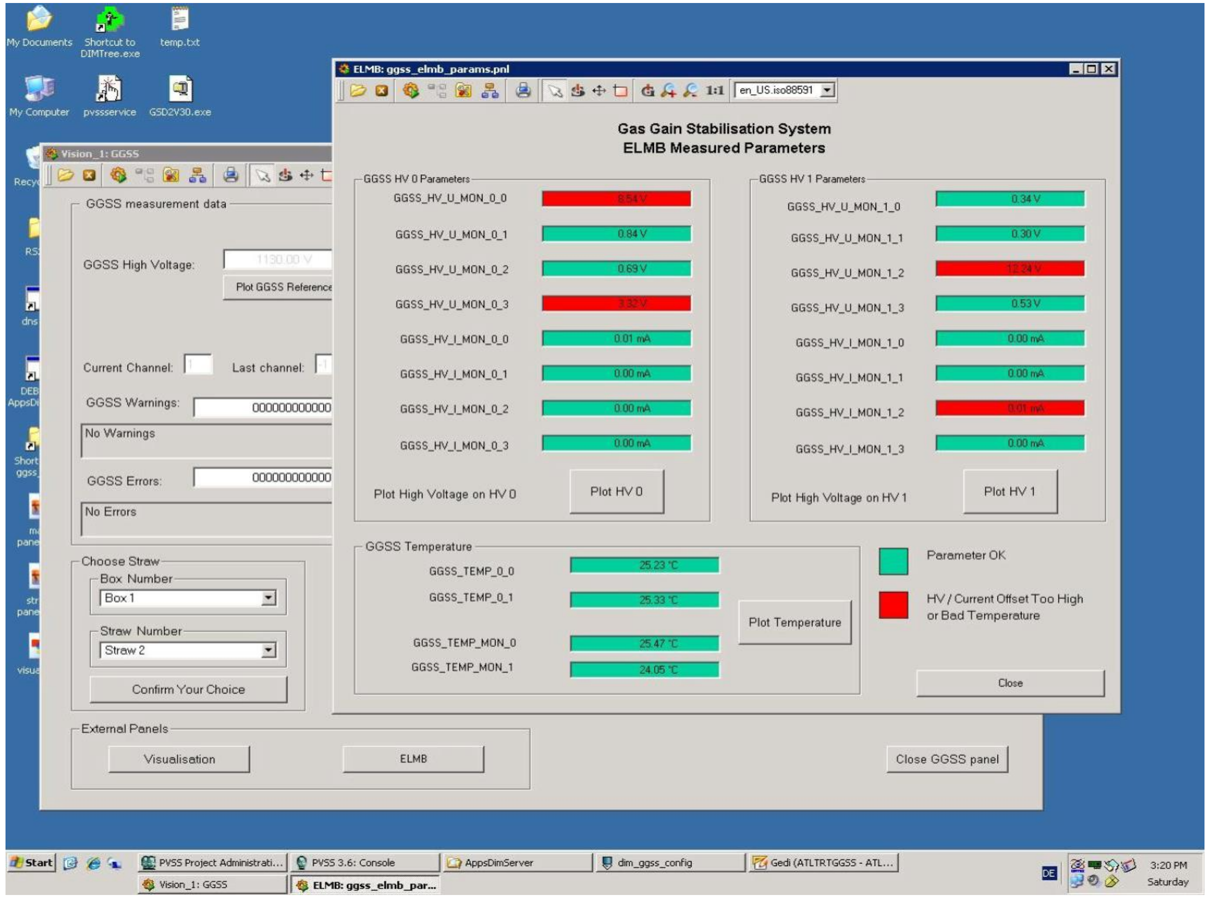
System mounted in USA15



GGSS straws mounted in USA15

Early GGSS interface

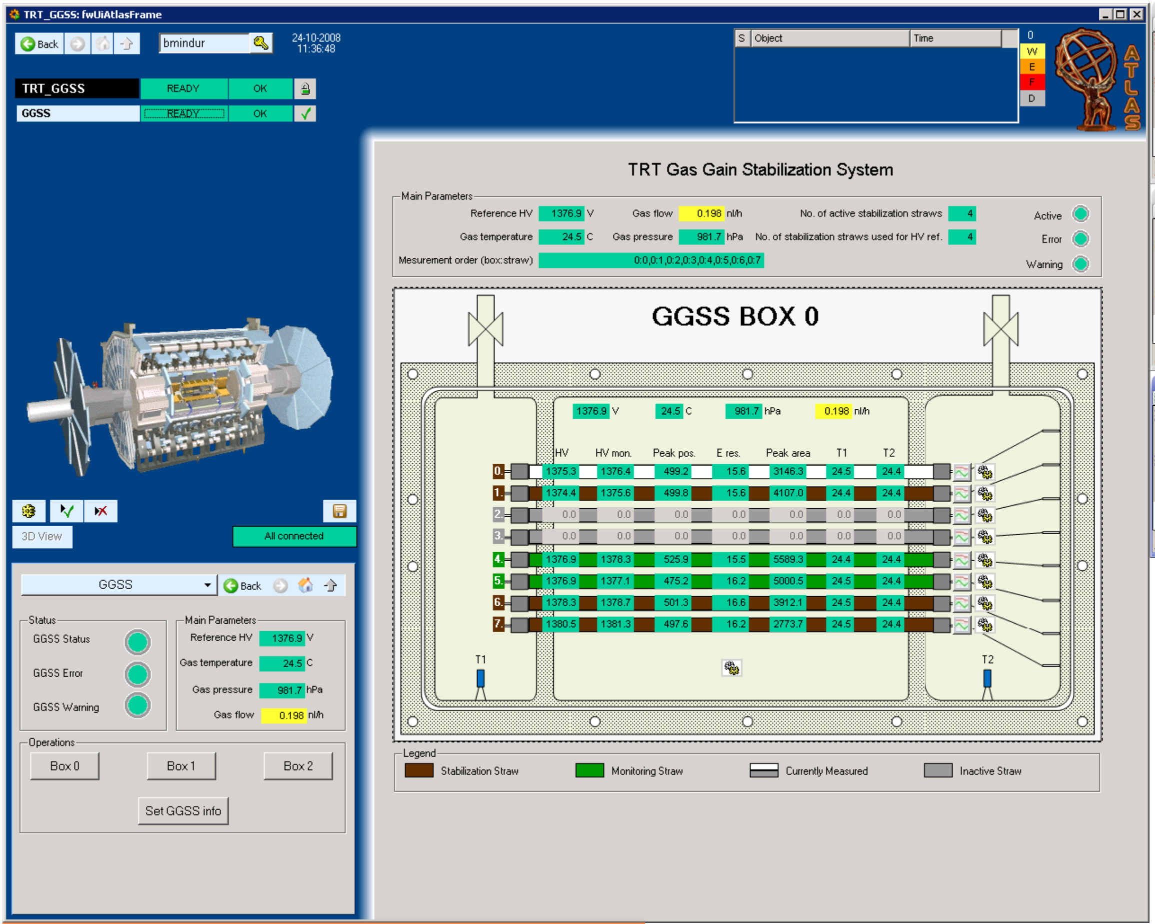




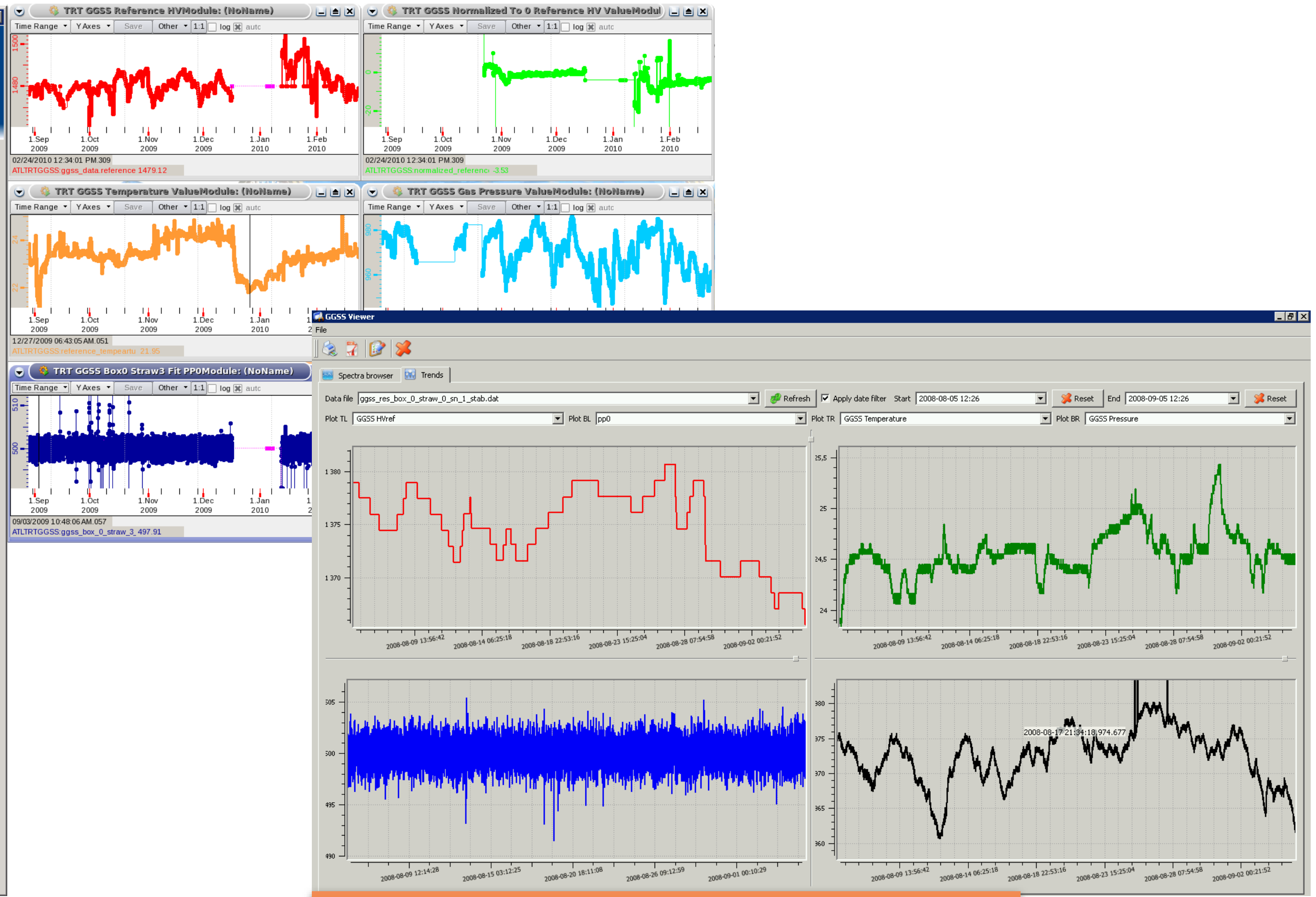
First version of GGSS UI

2010 -2013
Run1

Updated GGSS interface (FSM)



UI for GGSS monitor and control



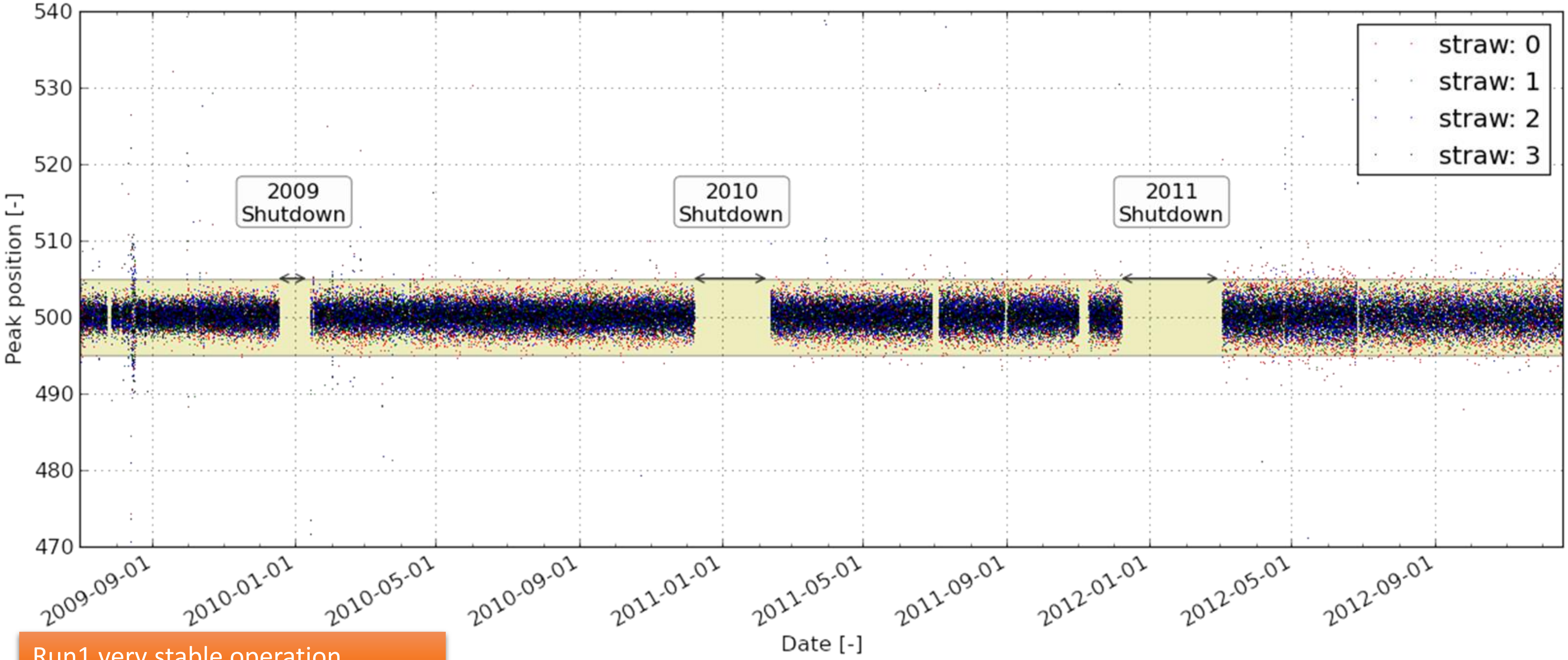
Qt based offline GGSS data visualizer

GGSS during Run1

- No major updates of software
  - Except Windows XP and WinCC OA patches
  - GGSS uses admin account all the time
- Maintenance
  - No central updates
  - Custom PC
  - User full access with admin rights to GGSS PC
    - (Even during stable beams)
- YETS
  - No changes in core software
  - Calibrations etc.

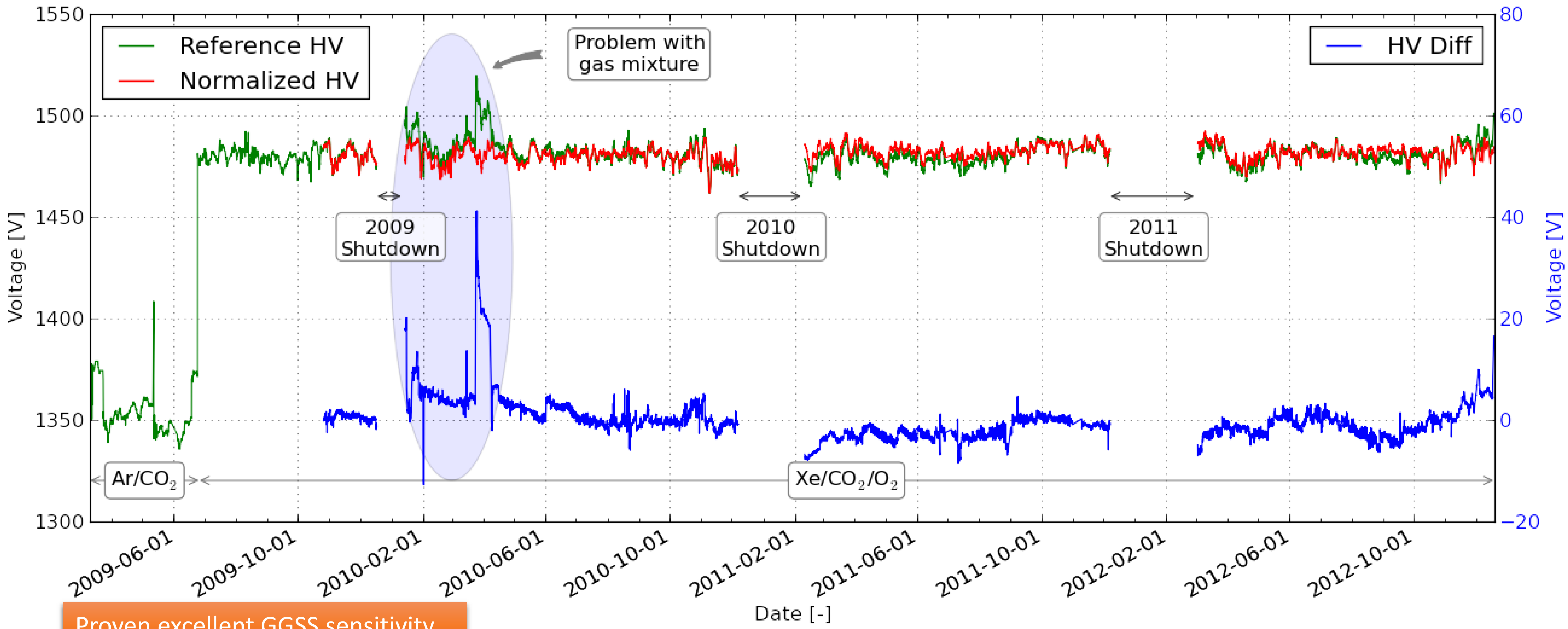
- All raw data, logs and configs stored locally on a hard drive
  - About 70,000 (text) files generated during Run1 operation
  - About 1,5 GB of data
- To central DB only crucial information were sent
  - Primely for save space
  - Massively reduce network occupancy
    - Smaller number of events
    - Better access time

GGSS Run1 operation



Run1 very stable operation

GGSS monitoring tool



Proven excellent GGSS sensitivity

Run1 outcomes

- Software was stable with very successful and smooth operation
  - No single failure reported
- Efficient implementation
  - Experience gained during CTB and pre Run1 period
- Windows XP 32-bit executable file
  - Very stable
  - Still can be run with modern OS releases
  - Windows provides very good backwards compatibility

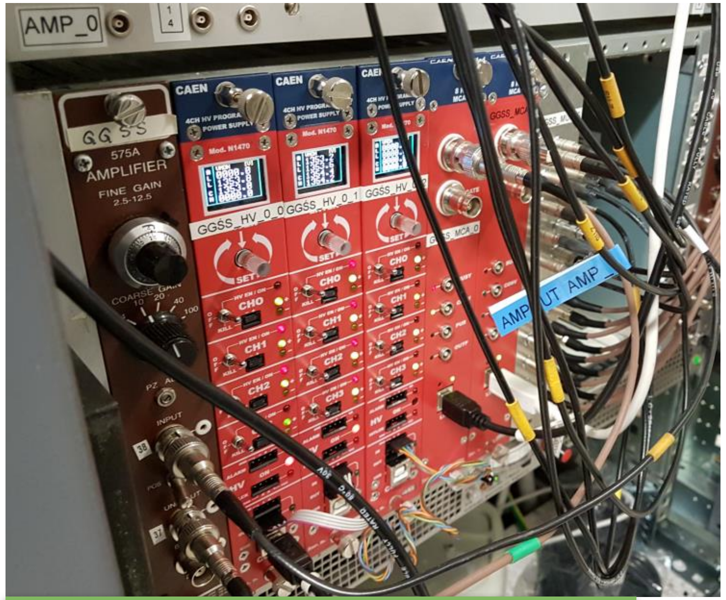
- Custom PC and nonstandard hardware
  - Huge disadvantage
- Nonstandard C++ libraries
  - Significant bottleneck
  - Windows threading libs
    - C++11 appeared during the Run1
- Windows XP – obsolete OS
  - But no updated drivers for PCI hardware (Windows 7 64 bit)
  - Admin rights required all the time
  - Access to desktop (via RDP) needed

2013 -2015
LS1

GGSS hardware upgrade

- PC replacement
  - Standard DELL PowerEdge R610
  - Centrally maintained
- PCI cards phaseout
  - New devices controlled via USB
  - (Significant cost of exchange)
- Linux based OS
  - System SLC6
- Sources migrated to SVN

Standard PC: DELL PowerEdge R610



USB based hardware modules

GGSS software refactoring during LS1

- Driven by OS change
  - Windows to Linux
- Access rights change
  - No more admin account used for running
- GGSS functionally unchanged
  - Significant effort to migrate threads library to *boost::thread*
    - Problems of support the C++11 standard by compiler toolchain at SLC6
  - Significant updated of logging facility
  - Development of driver package for new hardware
- Improvements
  - Quite a few new tools for monitoring and checking purpose
  - Introduction of code coverage test

Lines of code

Language	Files	Code	Comment	Blank	Total
C++	121	15,133	5,179	3,349	23,661
Python	24	4,839	772	910	6,521
XML	1	626	0	1	627
CMake	25	473	0	84	557
Shell Script	11	216	25	50	291

External dependencies

- Boost ver. 1.57
- Gnu Scientific Library (GSL) ver. 1.15
- Hardware dependent (CAEN – only MCA)
- DIM (managed internally by CERN)

LCOV - code coverage report

Current view: top level
Test: coverage_clean.info
Date: 2015-02-09 17:58:51

	Hit	Total	Coverage
Lines:	2267	2515	90.1 %
Functions:	141	151	93.4 %

Directory	Line Coverage			Functions	
ggssLib		90.1 %	2267 / 2515	93.4 %	141 / 151

Generated by: LCOV version 1.11

LS1 outcomes

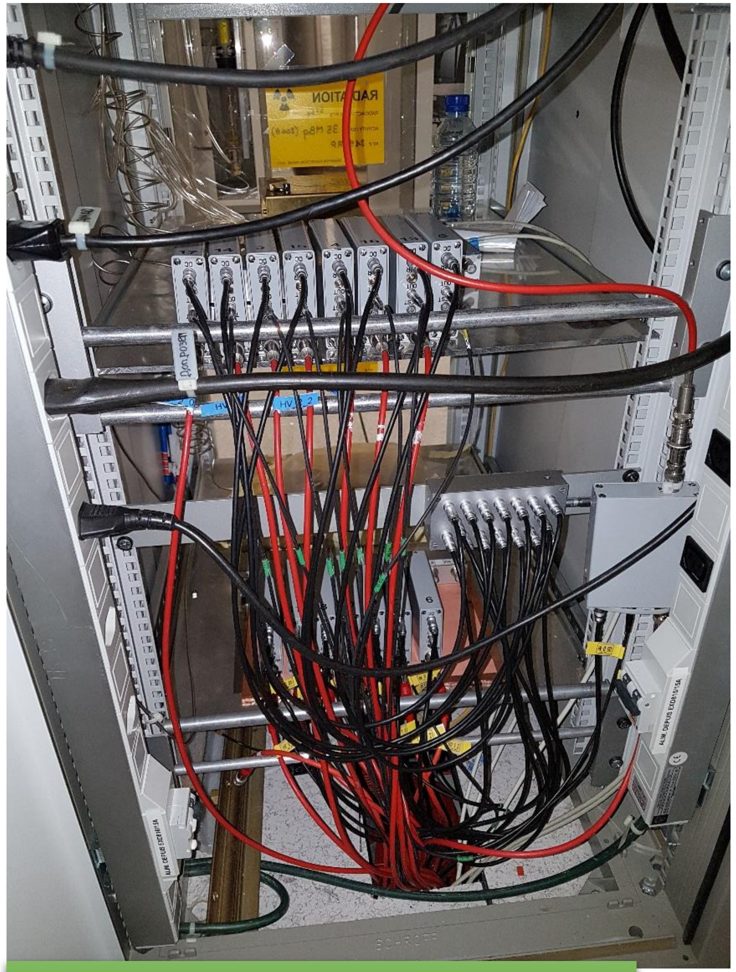
- Very successful migration
  - Usage of a standard centrally managed PC
    - Enforcement of access control and system package installation
  - Replacement of hardware
    - USB connection - TTY devices
    - Dedicated USB driver for CAEN MCA
- No significant changes in WinCC OA project
  - Well designed project
- Single biggest change of the GGSS during LHC operation

- Huge effort to change OS environment
  - Software depended
    - Threads, logging system, access control etc.
  - Hardware dependent components
    - New hardware drivers with completely different interface
    - Costs of hardware
- Problems with external build environment
  - Supported and allowed C++ complier without C++11 standard
    - Therefore *boost::threads* instead of *std::threads*

2015 -2018
Run2

Operation during Run2

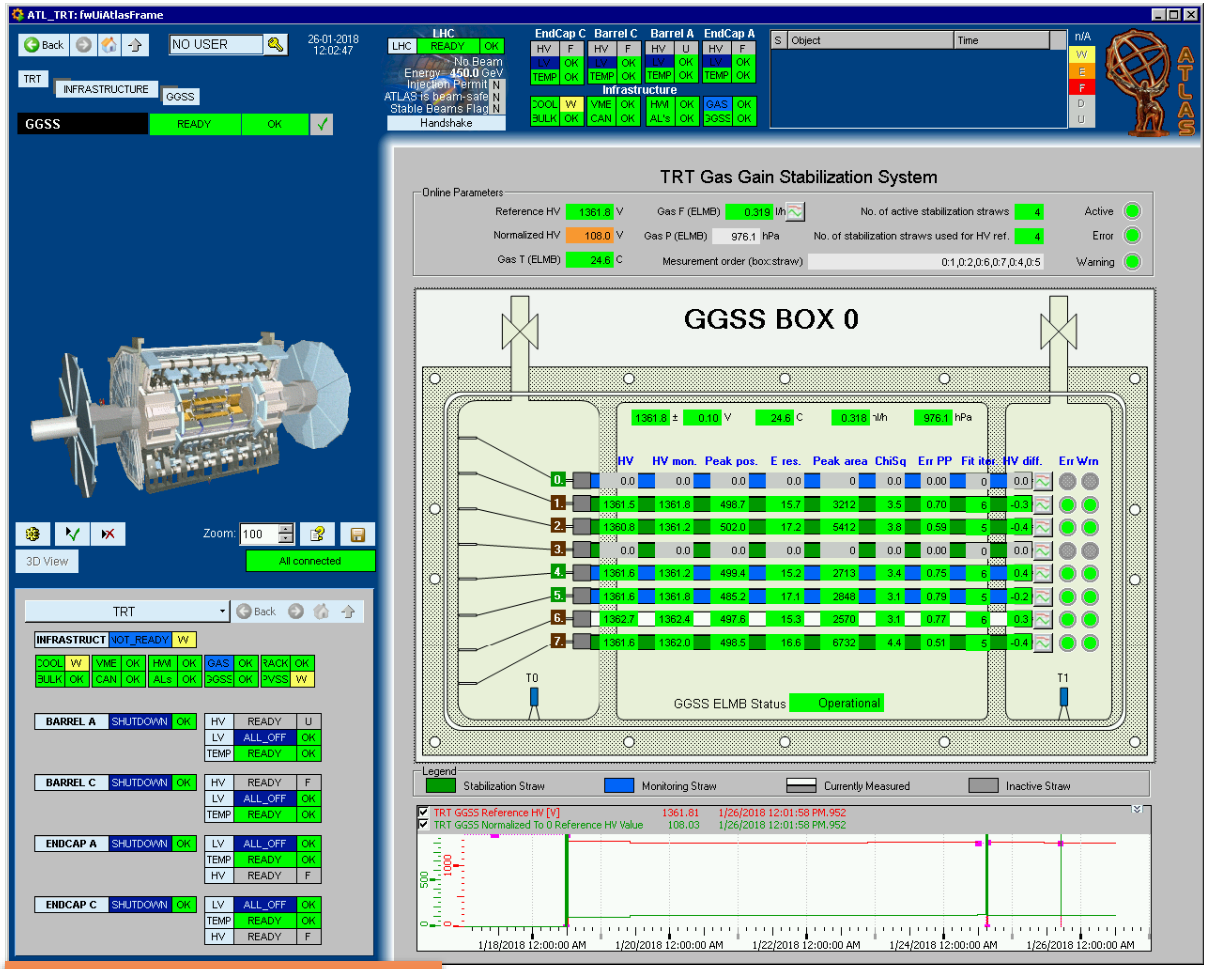
- Part 1 - SLC6
  - Till 2017-12
  - No problems
  - No changes in software
- Part 2 – CC7
  - From 2028-01
  - System successfully upgraded to CC7
    - Unexpected delete of the */localdisk* partition
    - Moving the most *results* and *logs* to dedication production repository with automated backup
  - Recompilation of GGSS core and drivers
    - Update to newer version of the libraries



Core GGSS hardware for Run2

Run2 outcomes

- Again, very stable operation
- OS changed during the run
  - Not very good idea
  - Necessity of software and driver recompilation
  - Unexpected cleanup of entire drive during system migration
- Changes to core software
  - Needs a lot of testing and monitoring before run starts
  - Finally validated during the run



GGSS UI during Run2

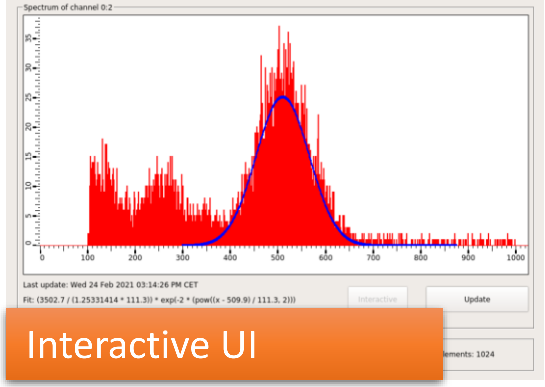
2018 -2022
LS2

GGSS changes during LS2

- Source code migration to Git
  - Git submodules
  - Usage of the CERN GitLab portal
- Introduction of CI/CD
  - Automated generation of the RPM packages
    - Core GGSS
    - Drivers for CAEN MCA
    - https://openstack.cern.ch/
- PC upgrade
  - Dell EMC PowerEdge R440 T

GGSS PC after upgrade

- Significant upgrade of the WinCC OA project
  - No more external dependencies
  - Removed unused DPEs
  - Integrated help system for GGSS dedicated command
  - Updated visualization



Interactive UI

GGSS software modification (LS2)

- Core GGSS software
  - Upgrade of the data analysis procedure
    - Triggered by low statistics due to weakening Fe-55 source
  - Added external commands handing for hardware
    - Mostly for monitoring during runs
- Complete rearrangement of the directory structure
  - More then dozen of git repositories with submodules
  - Automatic download of external libraries updates (if needed)

Language	Files	Code	Comment	Blank	Total
C++	285	21,854	11,502	6,477	**39,833**
CMake	115	5,188	0	1,274	**6,462**
Markdown	31	1,631	0	353	**1,984**
YAML	19	710	8	104	**822**
Python	7	491	2	115	**608**
Shell Script	8	272	22	59	**353**
XML	2	41	0	1	**42**
Properties	1	1	0	2	**3**

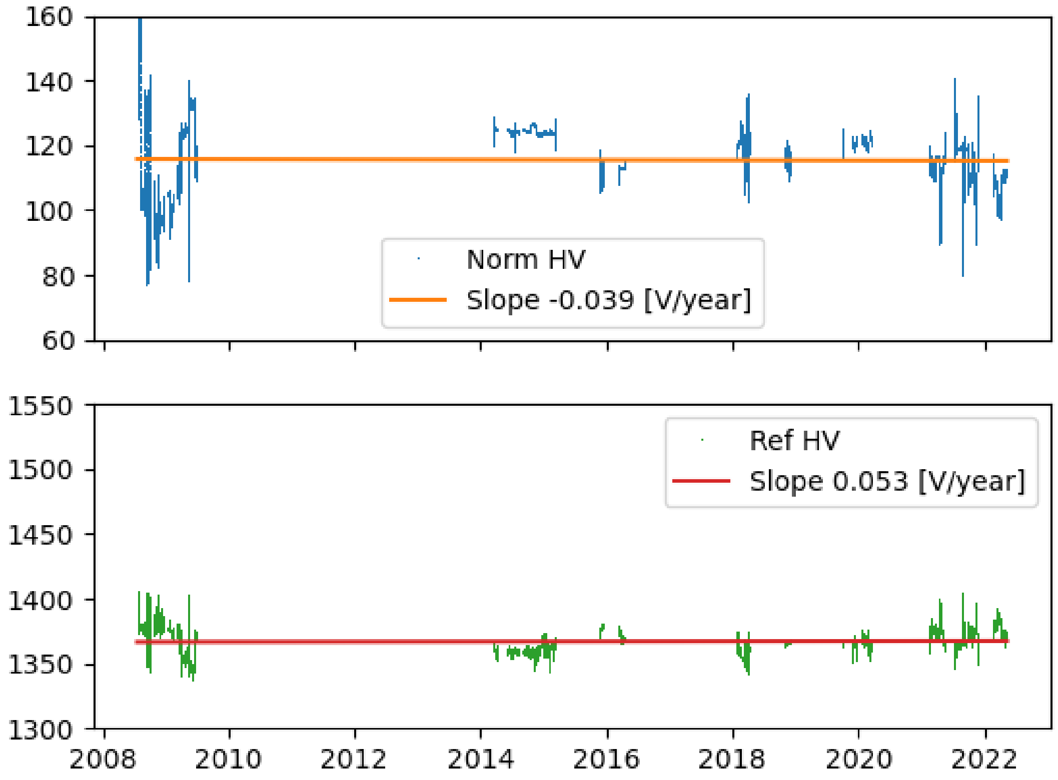
External dependencies

- Boost ver. 1.57
- GNU Scientific Library (GSL) – external (broken)
- Hardware dependent (CAEN N957 v.1.6)
- DIM (managed internally by CERN)

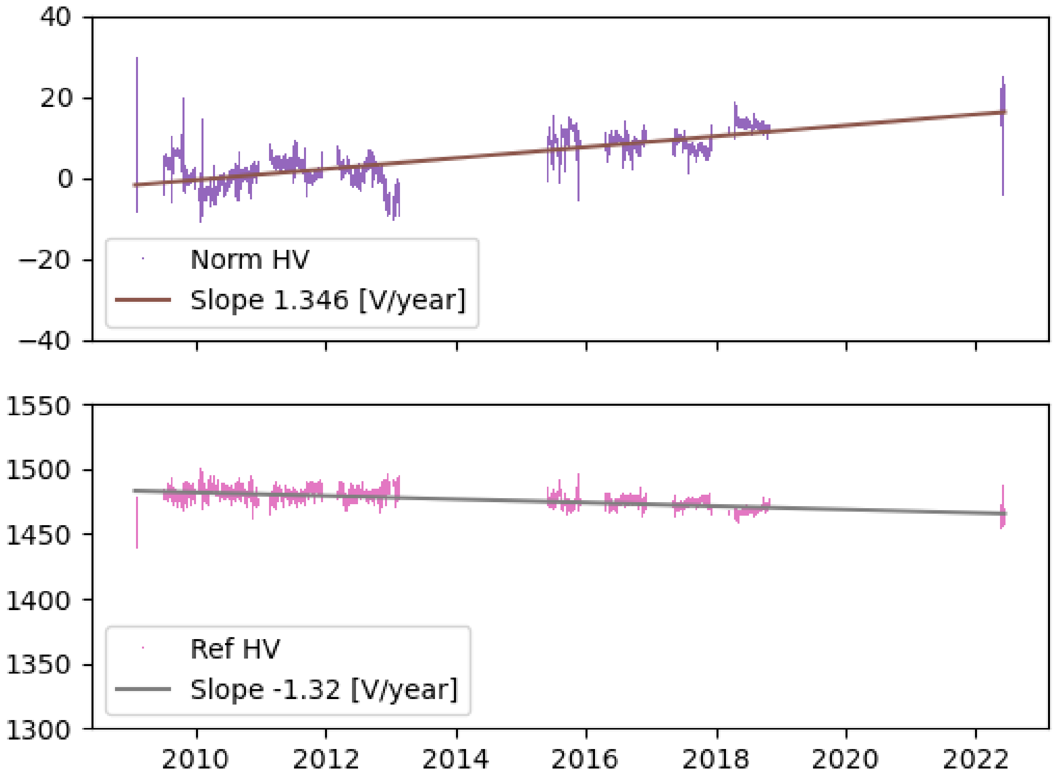
2022 – (2026)
Run3

GGSS very sensitive device

GGSS operation with Ar-based gas mixture

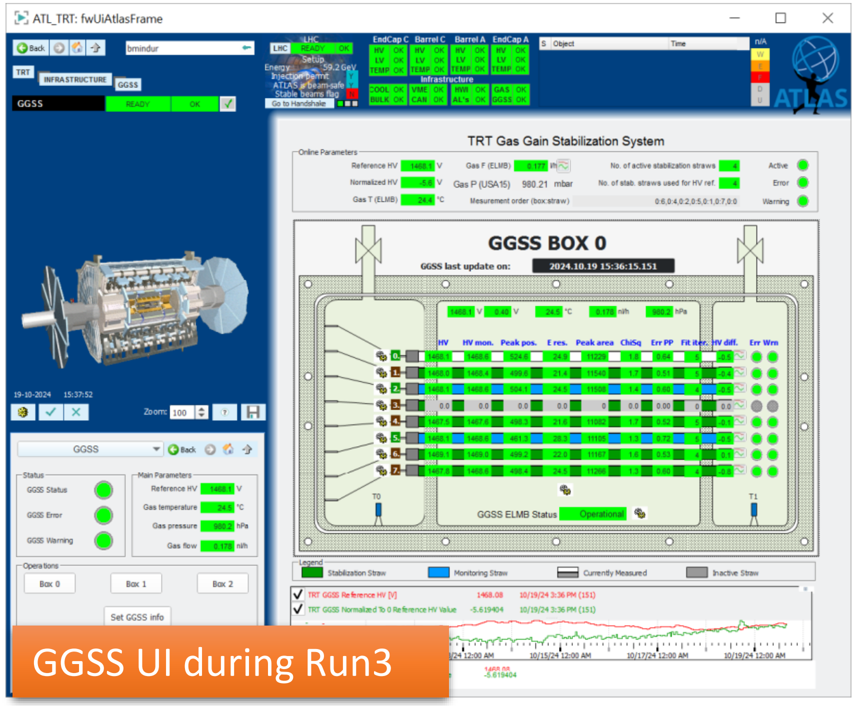


GGSS operation with Xe-based gas mixture

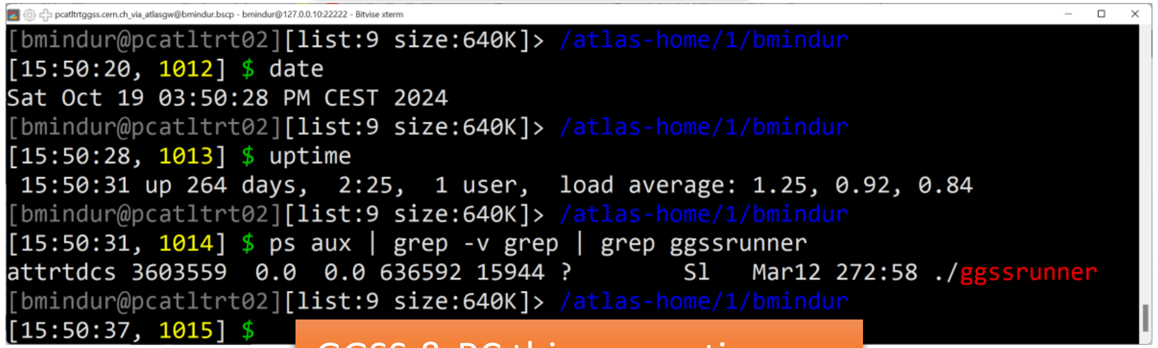


Ongoing operation during Run3

- Generally smooth operation
  - First ever error triggered by GGSS during the run
    - At 2023-07-07 problem with a single straw
    - Automatically recovered
- Migration to Linux ALMA9 in 2024
  - Recompilation of driver
    - No problems
  - No recompilation of GGSS core software
    - Breaking changes in GNU Scientific Library
      - v2.xx not compatible with 1.xx
    - Binary combability is very important
  - Still running binaries from 2021



GGSS UI during Run3



GGSS & PC this yearuptime

Conclusions

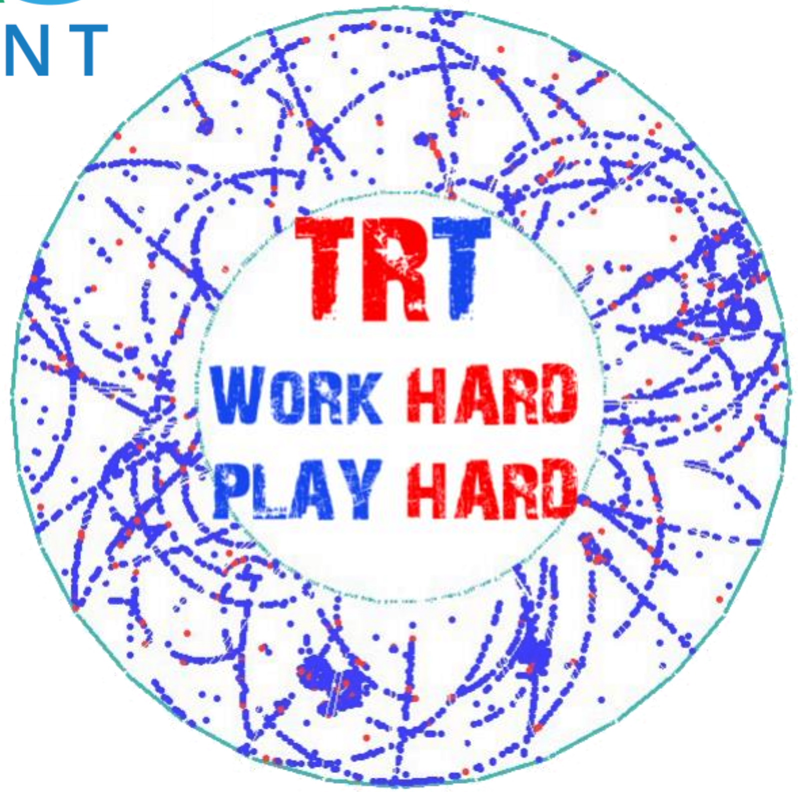
Conclusions

- GGSS system maintained and running for almost 20 years
  - Provides stable and reliable data
- Any breaking changes requires huge effort
  - Mostly underestimated
  - GGSS software testing is very hard
    - Dependence on external devices
- Good design pays off
  - Core functionality of the GGSS software was NOT changed at all
- Environment stability
  - Constantly decreasing
  - More and more effort required
- Security
  - At the beginning almost nonexistent
  - Nowadays strict (maybe too much?)
- Usual system and GGSS uptime
  - ~300 days each year – the whole run period
- GGSS software experience no single failure during all LHC runs!

Acknowledgements

Acknowledgements

- ATLAS Collaboration
- TRT Collaboration
- AGH University of Krakow
  - Faculty of Physics and Applied Computer Science
    - Department of Interactions and Detection Techniques



Thank you

Q&A

Backup slides

Abstract

I will be presenting the history of the design, implementation, testing, and release of the production version of a C++-based software for the Gas Gain Stabilization System (GGSS) used in the TRT detector at the ATLAS experiment. This system operates 24/7 in the CERN Point1 environment under the control of the Detector Control System (DCS) and plays a crucial role in delivering reliable data during the LHC's stable beams.

The uniqueness of this software lies in its initial release around 2004, followed by subsequent refactoring, improvements, and implementation for the Run1 period of the LHC in 2008. Another significant change occurred during Long Shutdown 1 when the operating system transitioned from Windows to Linux for Run2 in 2015. More recently, there have been frequent updates and upgrades to the operating system and external libraries.

My aim is to present the evolution of the software, highlighting changes introduced from an external perspective due to shifts in the environment or requirements. Additionally, I'll discuss the evolution of the C++ standard, compiler changes, security considerations, and modifications to the build and test environment. During the conference, I will focus on the most compelling and significant milestones, as well as key aspects relevant to the lifecycle of this software.