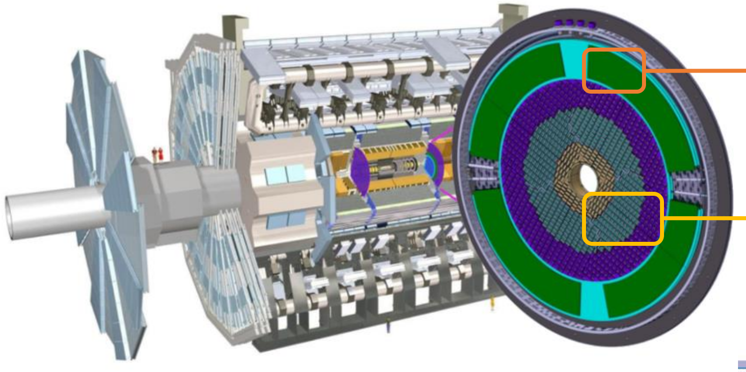
HGTD Demonstrator and DAQ software

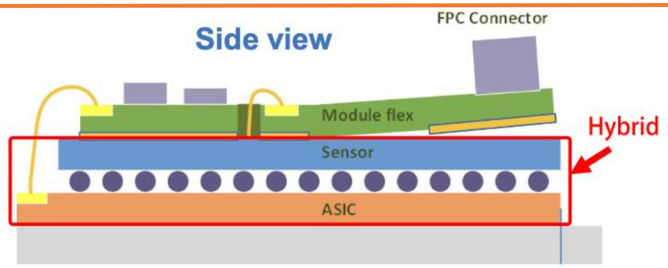
Aonan Wang

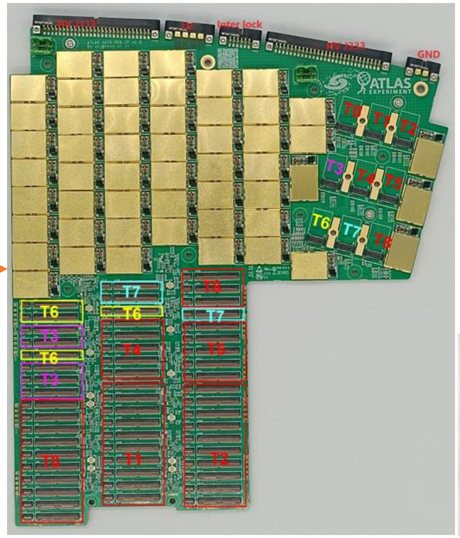
On behalf of USTC HGTD group

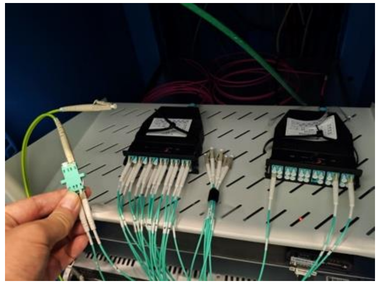
5th symposium on semiconductor radiation detectors, 4/18/2024

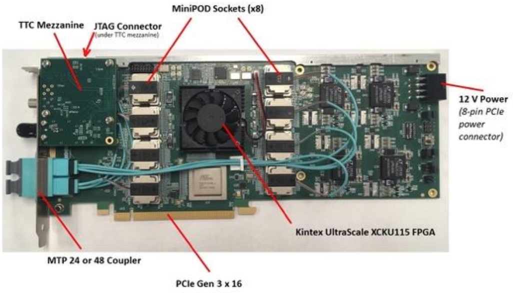
Introduction to the HGTD

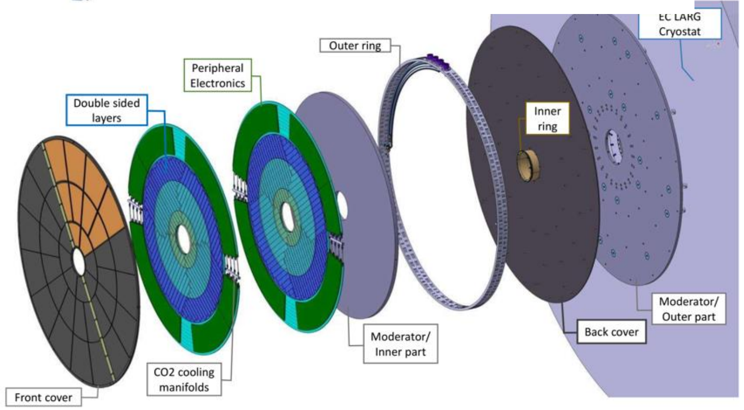


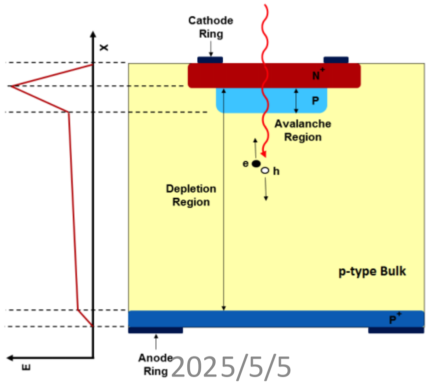


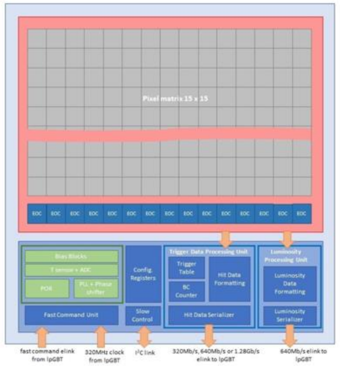






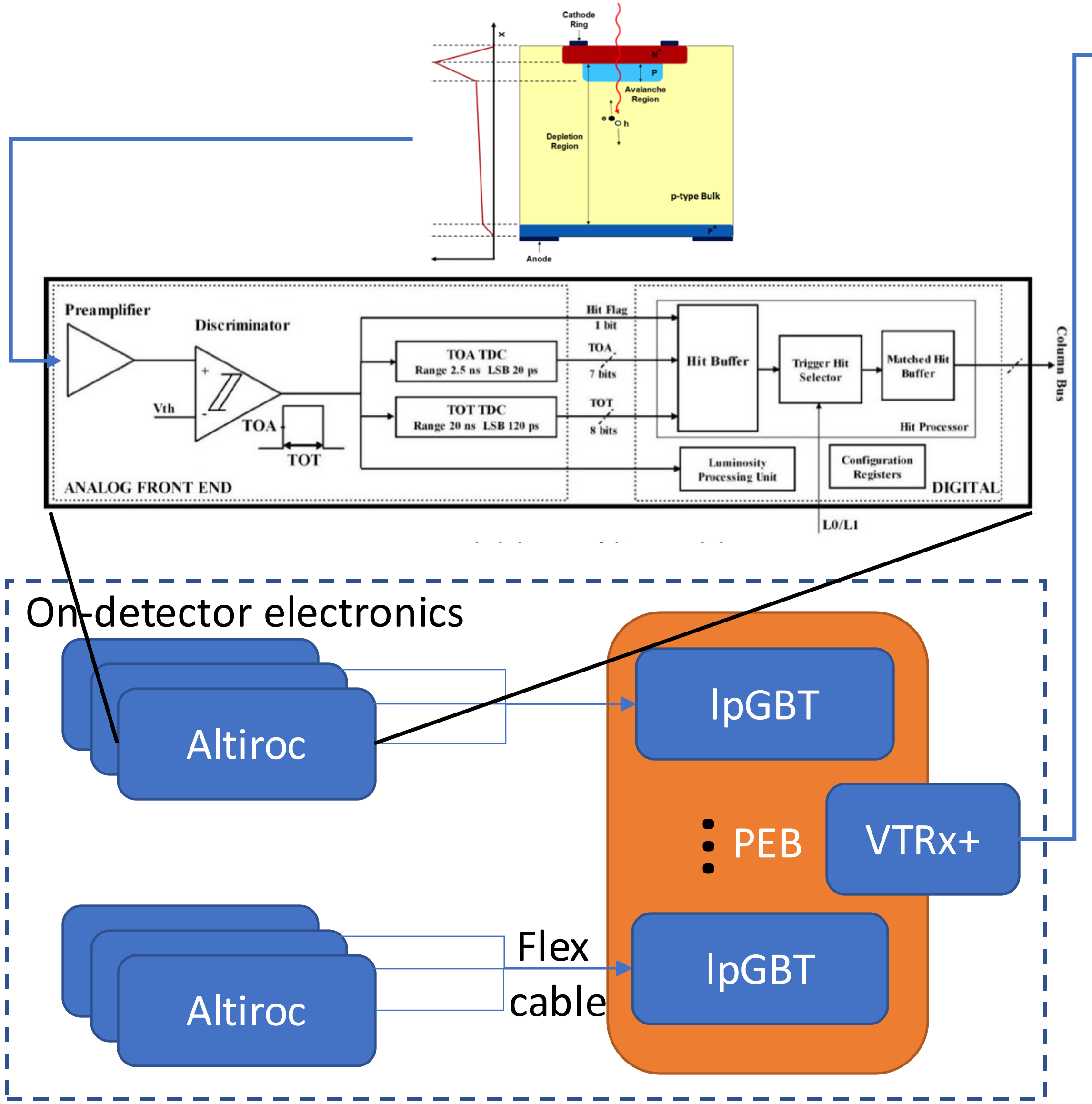


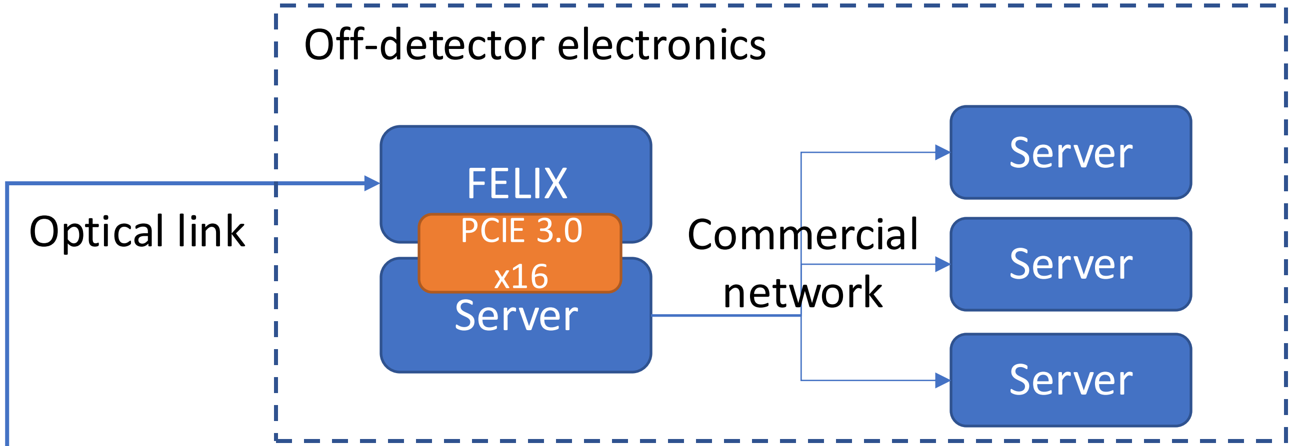




- By using high-precision timing information, the increase in pileup interactions from the LHC to the HL-LHC can be significantly mitigated.
- The sensor will be the Low Gain Avalanche Detector (LGAD).
- The front-end electronic ASIC is named ALTIROC.
- The module is composed of LGAD, ALTIROC and Module flex.
- The Periphery Electronic Board (PEB) is located at the outer ring.
- The PEB will be connected to the FELIX on the server through optical fiber.
- There will be 8092 modules, which means 3641400 channels for entire HGTD.

HGTD Data flow

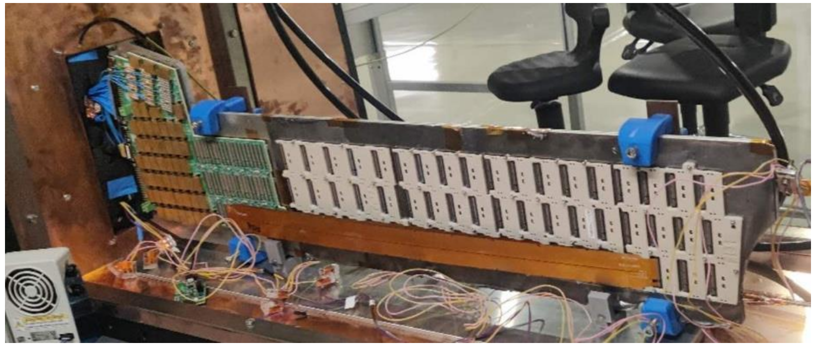


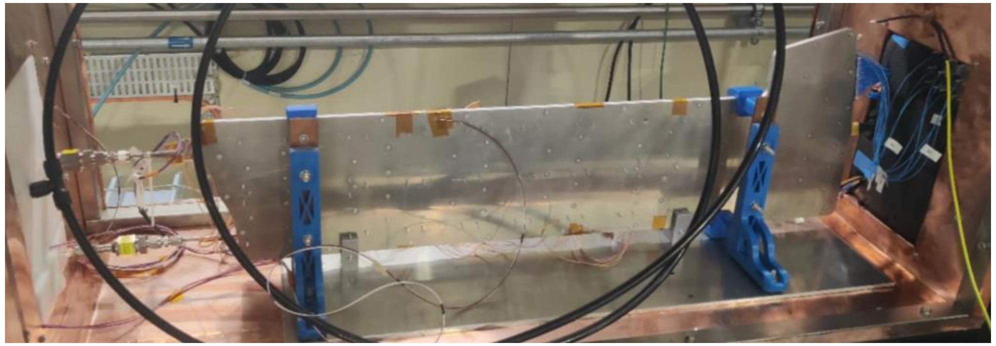


- The signal from LGAD amplified by PA will pass to the discriminator, and then it will be converted into a digital signal by the TDC within the ALTIROC.
- Upon receiving a trigger, the TOT and TOA will be transferred to the lpGBT through flex-cable.
- The lpGBT collects data from different modules and send it to the VTRx+.
- The VTRx+ converts the data into optical signal.
- The FELIX collects the data from multi lpGBT and tagged with e-link.
- The data is forwarded to the server memory via PCIe bus.

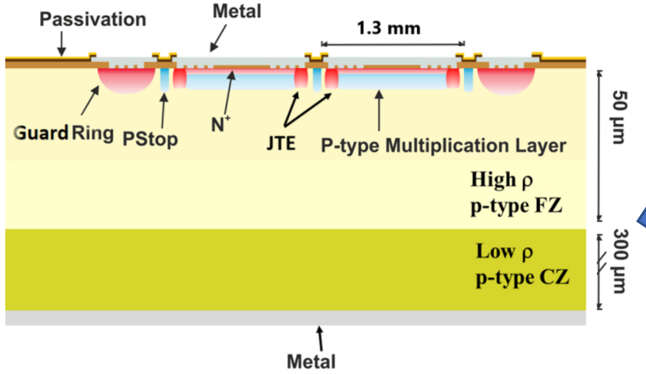
HGTD Demonstrator

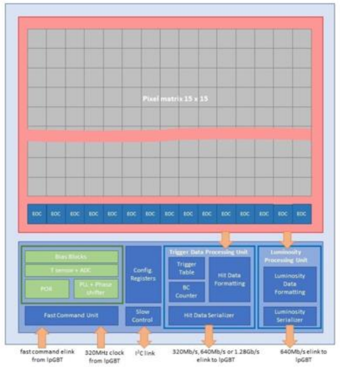
- At this stage, the demonstrator involves two parts.
  - The first is the DAQ demonstrator,
    - It utilizes the PEB1F and corresponding detector units (DU) as the first step.
    - It also includes the server with the FELIX card.
    - The ATLAS Local Trigger Interface (ALTI) is used for trigger and clock distribution.
    - The readout software is also a crucial component of the demonstration.
  - The second is the heater demonstrator,
    - The cooling plate is made of stainless steel, with an additional PGS layer to facilitate heat exchange.
    - The CO_2 is delivered into the plate through an aluminum tube.
- The system is consists of 54 modules, 108 LGAD sensors and ASICs, 24300 channels.
- This setup helps us validate the full chain and identify any possible issues.
- Meanwhile, it provides a platform for the development of data acquisition and detector control system.

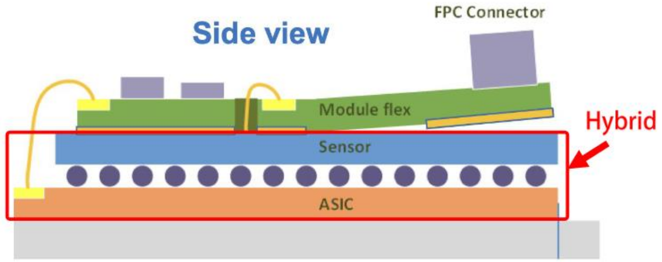


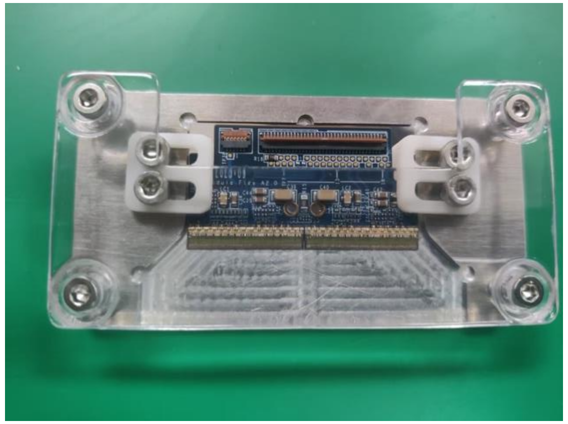


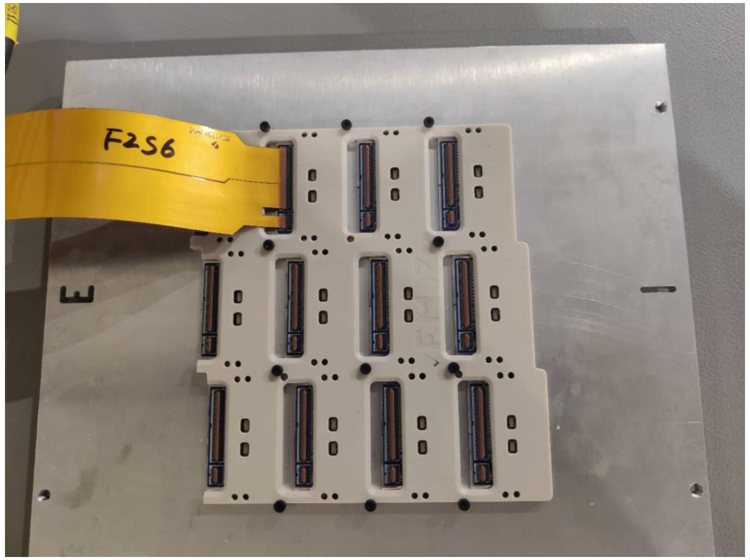
From sensor, ASIC to module, detector unit





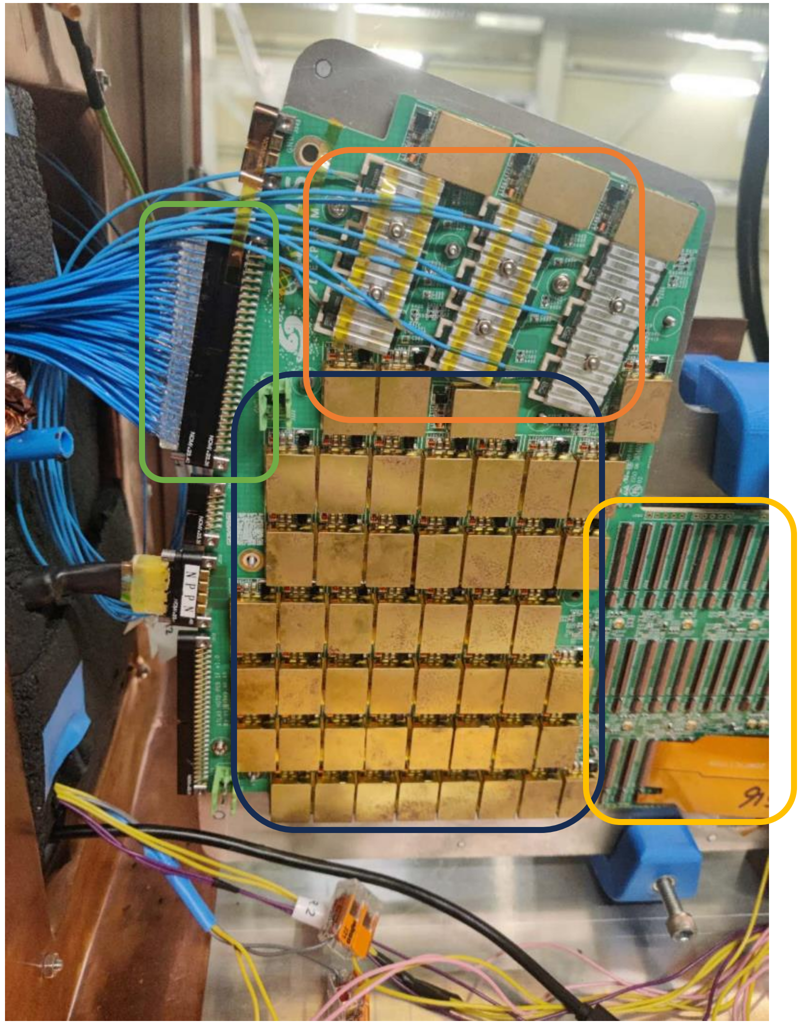






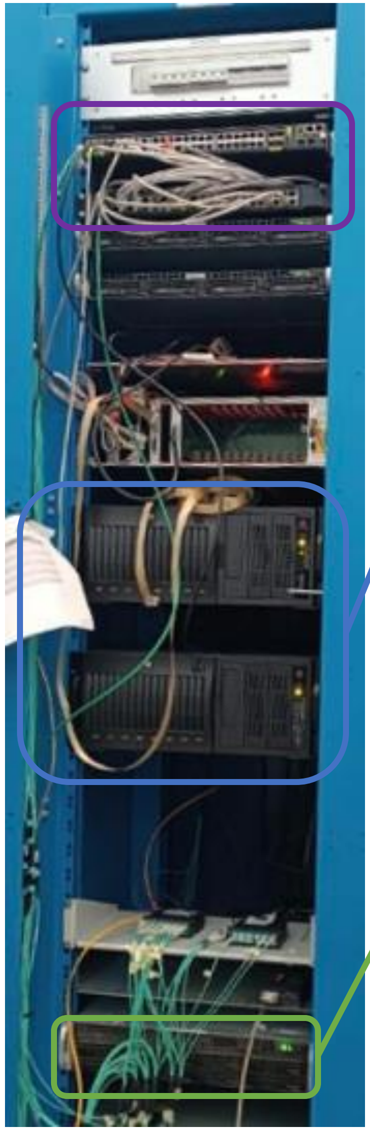
- A 15×15 LGAD sensor and an ALTIROC ASICs will be flip-chip bonded into a Hybrid.
- Two hybrids are glued to the module flex, which provides electrical routing, further filtering for power source and control voltage set. The wire bonding is used to establish the electronics connection.
- The modules are then attached to the 3d-printed support units, forming the detector units.
- The detector units are connected to the PEB via the flex-cable.

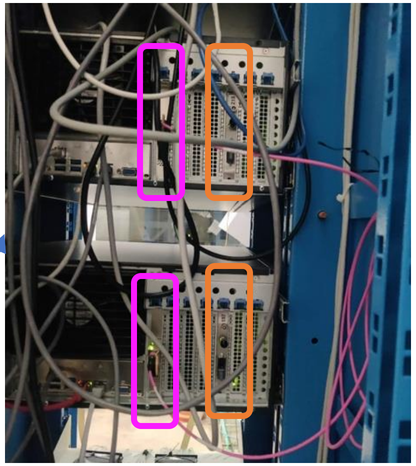
Periphery Electronic Boards



- The periphery electronic board (PEB) is designed to provide connection between module and backend server, and HV/LV source.
- It includes some several components
  - **Bpol** – Converts 12V to 1.2V and supplies power to the module for both VDDD and VDDA.
  - **VTRx+** – Performs electronic-optical convert.
  - **HV connector** – Connects external HV module by module.
  - **Flex-cable connector** – Interfaces with the module via a flex-tail.
  - **lpGBT** – Provides $\mathrm{I}^2\mathrm{C}$ for all chips and GPIO to control other chips. It also merge data from multiple modules and handle data/command transfer.
  - **MUX64** – A 64-to-1 analog multiplexer ASIC.
  - **NTC** – A temperature-sensitive resistor.

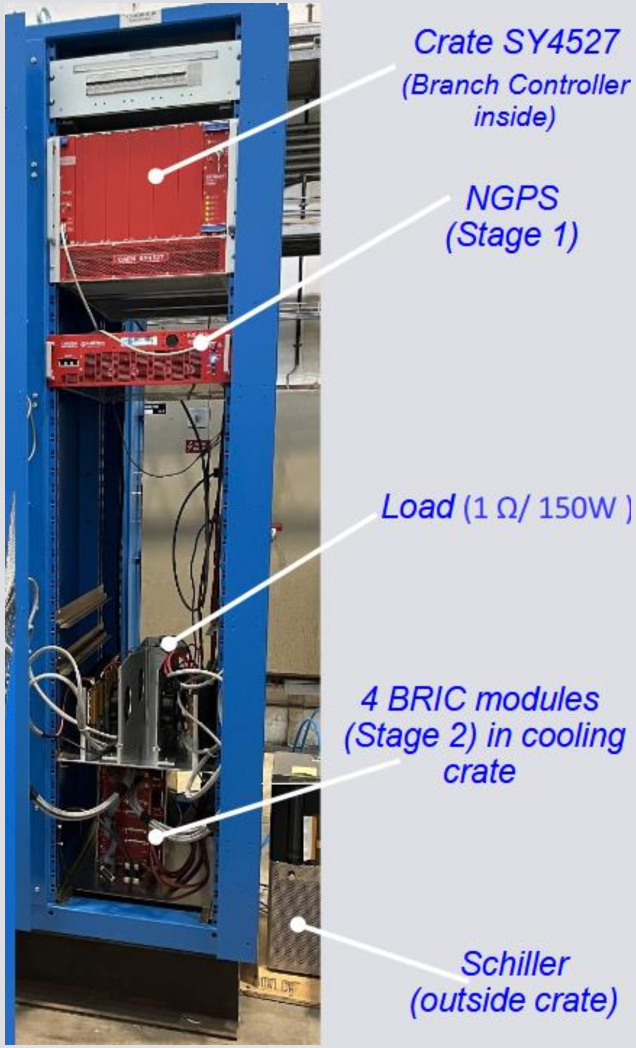
Readout Server





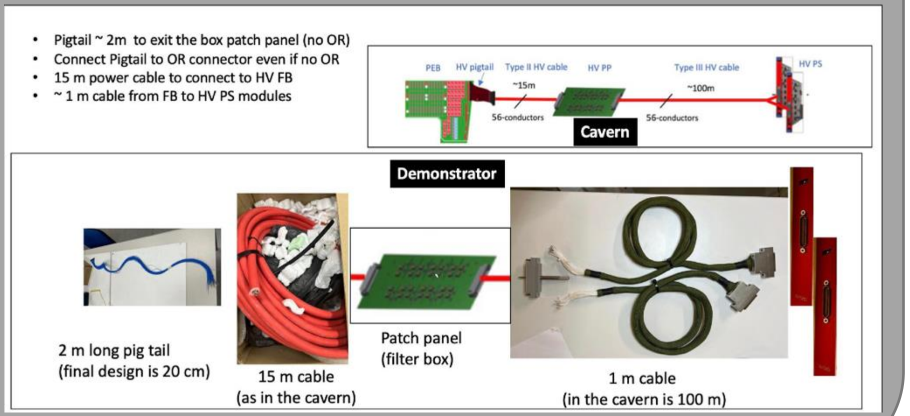
- Currently, three servers are used for DAQ demonstrator testing,
  - Each is equipped with an FELIX-182 card.
  - Two of them are newer one and are connected via high-speed optical network(pink), and one of them also connected to the Demonstrator through FELIX card.
  - All servers are connected to the same router, sharing a local network.
- A VME craft is used for ALTI module, it will provide the trigger and clock distribution.

LV and HV power source



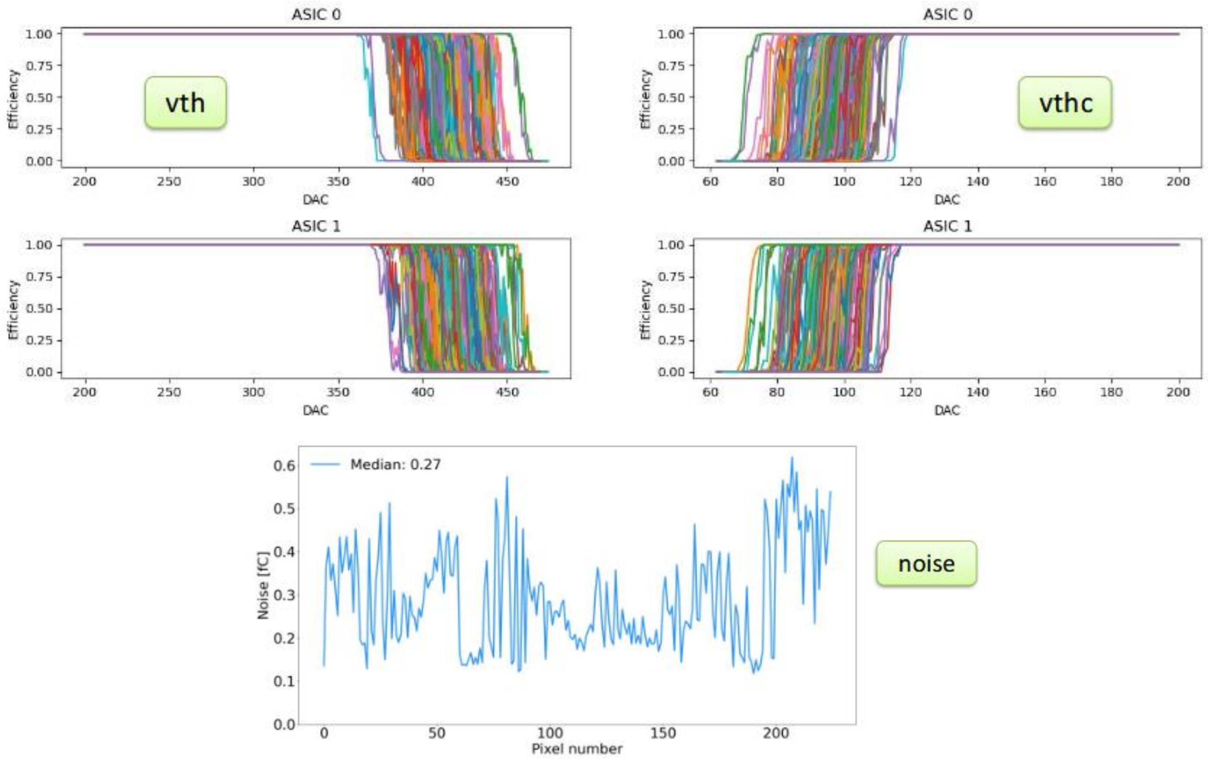
Low voltage system

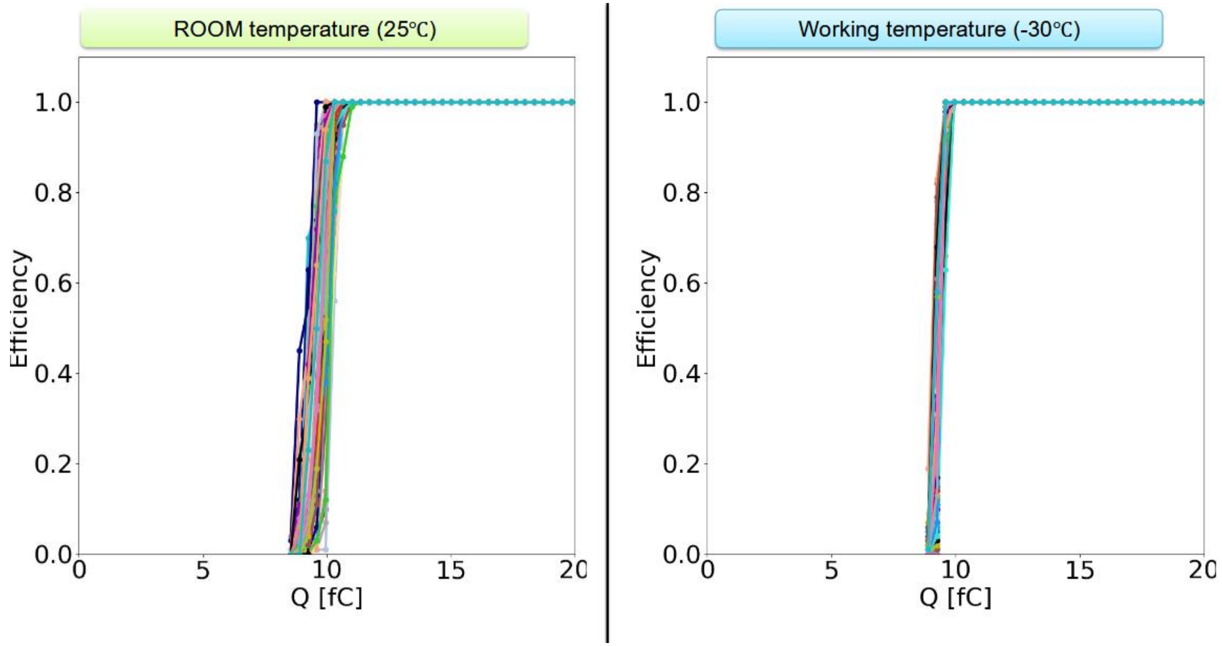
High voltage source



- The Low Voltage power supply is composed of three stages.
  - Stage 1: NGPS will convert 400V (three phase AC) into 300V DC.
  - Stage 2: BRIC modules will convert 300V DC into 12V DC.
  - Stage 3: The bpol on the PEB will convert 12V into 1.2V and 2.5V.
- The High voltage system will be integrated into VME crates,
  - Now only one module is used for demonstrator test.
  - An additional patch panel is used for filtering.

High speed test and low temperature test





- The Altiroc3 Modules has been tested for following conditions
  - High-speed performance: Datat transferring rate up to 1280 Gb/s. The matched the expectation.
  - Low temperature operation: Tests were conducted with the temperature controlled to $-30℃$. ASIC tuning showed improved results compared to room temperature, as expected.

Next step for the hardware

Detector Units

- A more refined version of the ASIC, ALTIROC-A, will be used in the production of new DUs,
- The module flex has been updated to add additional filters for both HV and provide external control voltages.

HV System

- Additional HV modules are under production and will match the final setup, offering more channels.

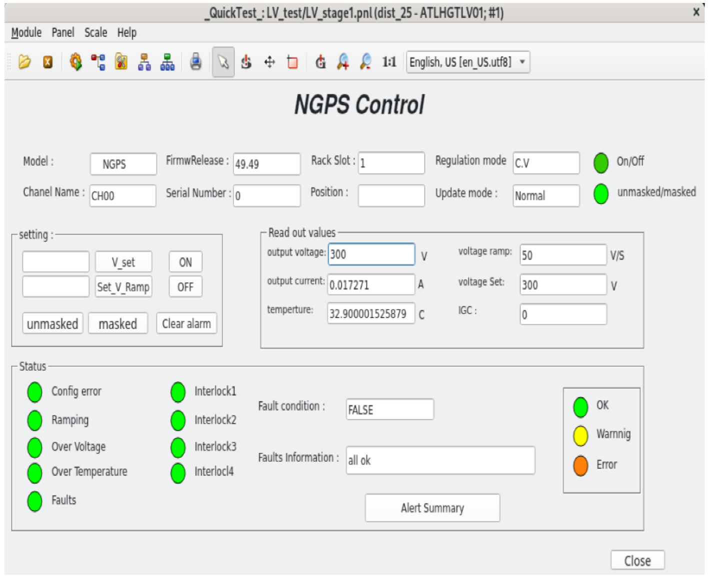
Mechanics

- A new Faraday cage box with an identical Outer-Ring, Internal moderator, DUs and PEB alignment, Holding brackets to the final design will be implemented.

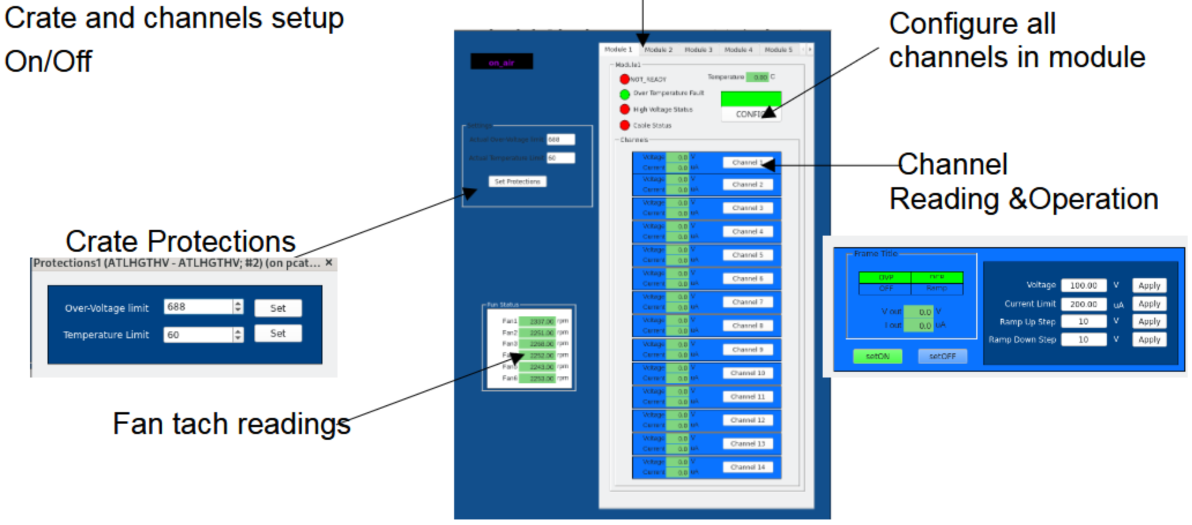
Interlock

- The crate has been partially installed, will be integrated with the demonstrator.

Detector control system



- Current Uis are based on first version
  - Crate and modules
  - Crate and channels setup
  - On/Off

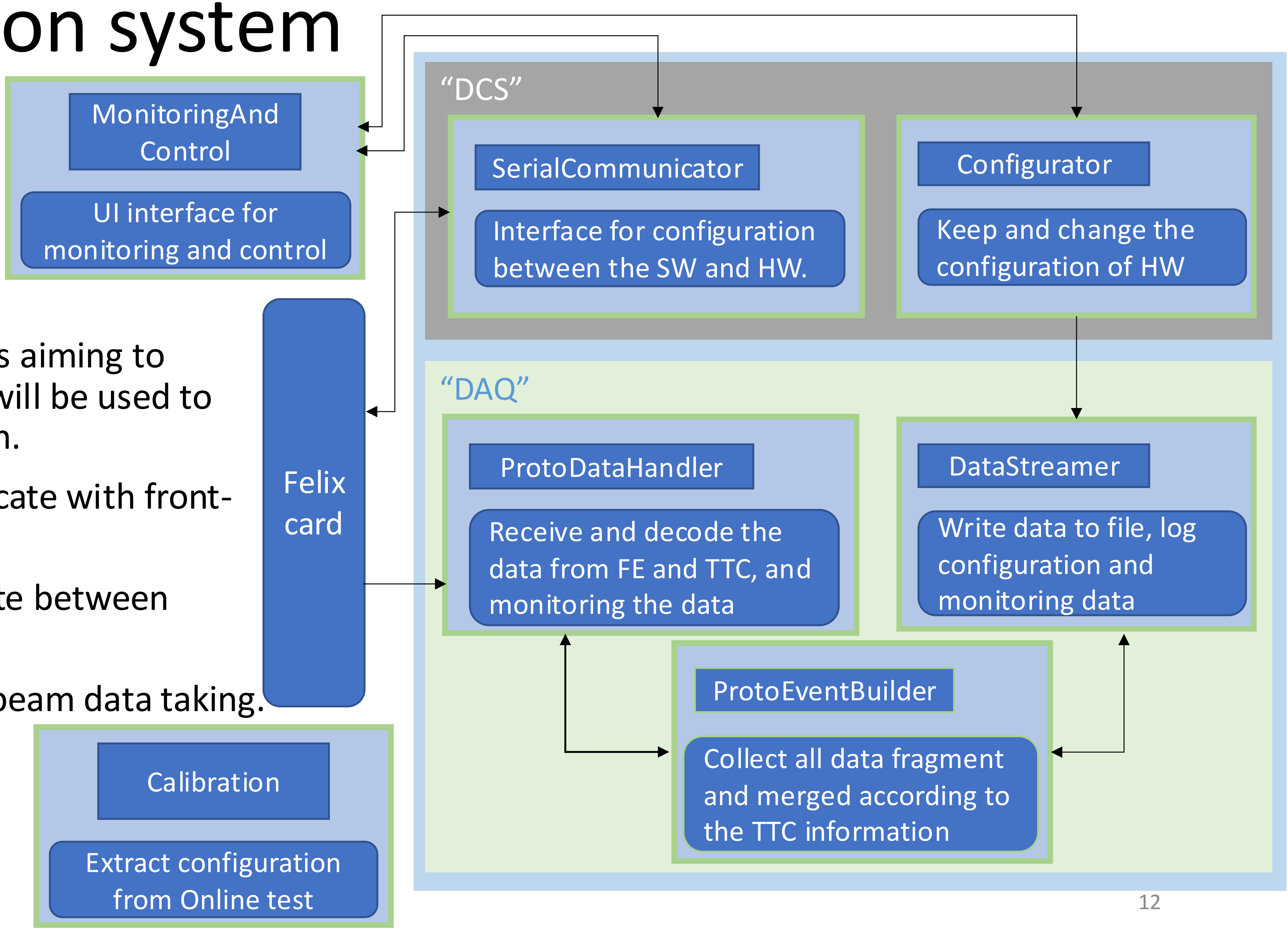


- A GUI controller for the low-voltage system has been developed based on WinCC and OPC UA (Stage 1 and Stage 2), with monitoring capabilities.
- For the stage three, now under the control and monitoring of DAQ software
- The high-voltage controller, also based on WinCC and OPC UA, is still under development.

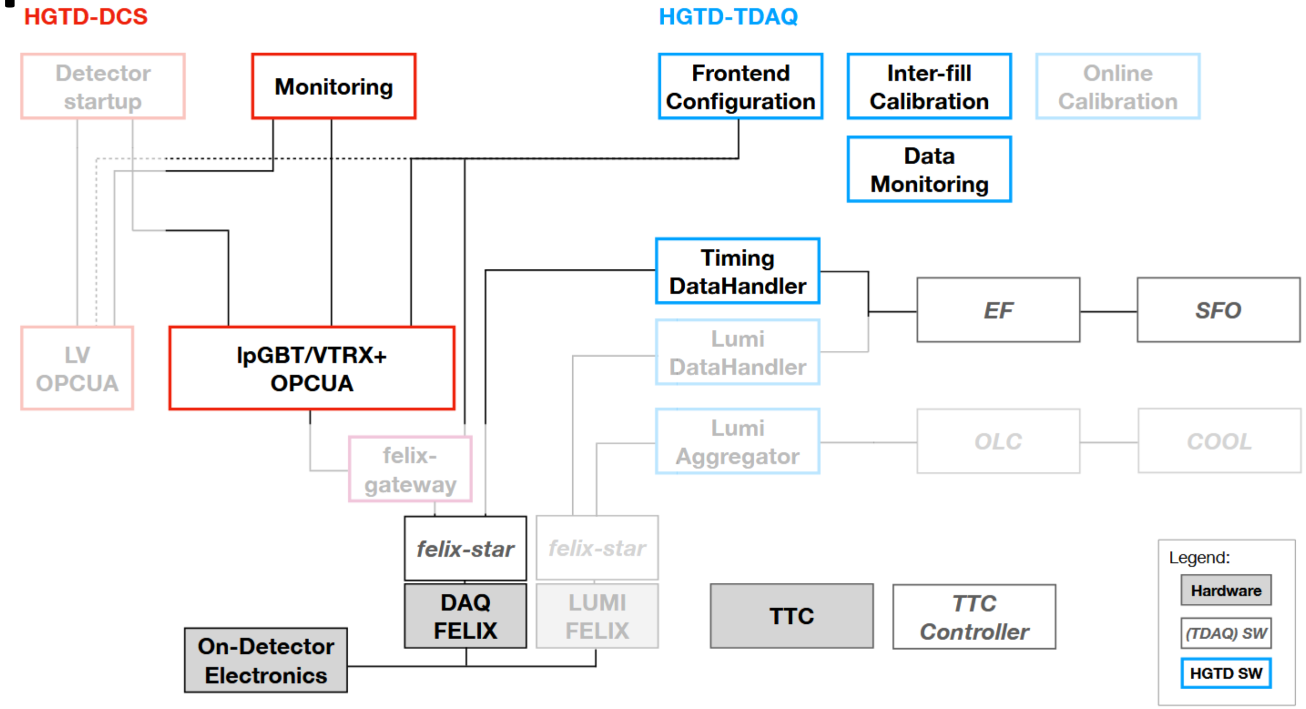
Data acquisition system

This suit of readout software is aiming to develop a demonstrator that will be used to implement a full readout chain.

- Using felix-star to communicate with front-end electronics.
- Using TCP/IP to communicate between different components,
- Planning to be used in test beam data taking.



Next step for the DCS and TDAQ integration



- So far, the development of the DCS and DAQ has not been tightly integrated. Each is focused on implementing its core functionality.
- The controller for voltage supply and signal monitoring are based on WinCC and OPC UA.
- Configuration and monitoring of the front-end electronics are handled by self-developed software based on the ATLAS TDAQ application felix-star.
- An additional layer is under development to provide unified interface: DCS will be operated via WinCC, while DAQ will focus on data handling.

Summary

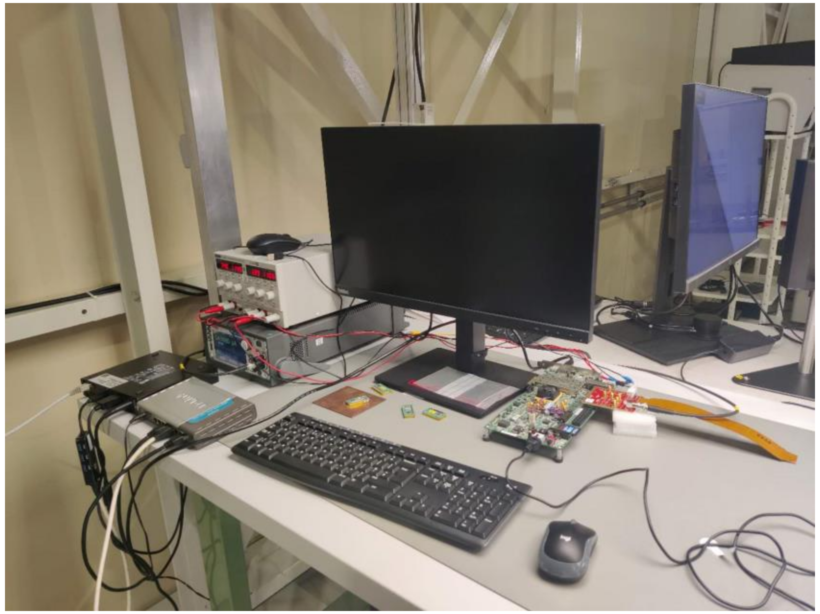
- The HGTD consists of sensors, front-end ASICs, module flex, support units, PEB, power sources, the interlock system, the DCS and DAQ systems, cables, mechanical supports, and the cooling and monitoring system. These components will be integrated into the demonstrator step by step.
- Signals from the sensors are digitized and collected by ALTIROC and lpGBT, and are ultimately saved on a server with a FELIX card via optical fiber.
- The demonstrator has been setup to validate the readout chain.
- A preliminary version of the DAQ and DCS software is under development, and the demonstrator provides a solid platform for testing and integration.

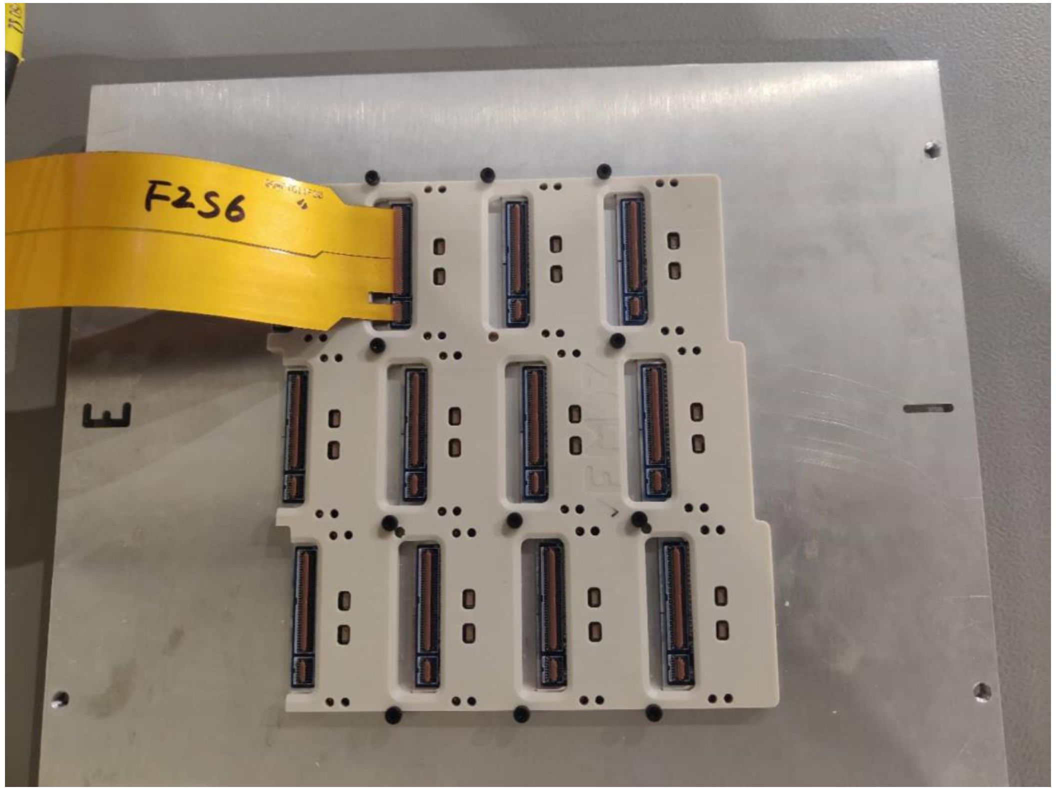
Thanks for your attention!

acknowledgement : The speaker acknowledges the contributions by many members of CERN/IFAE/IHEP/IJClab/JSI/NJU/USTC ... groups

Backup

Detector unit





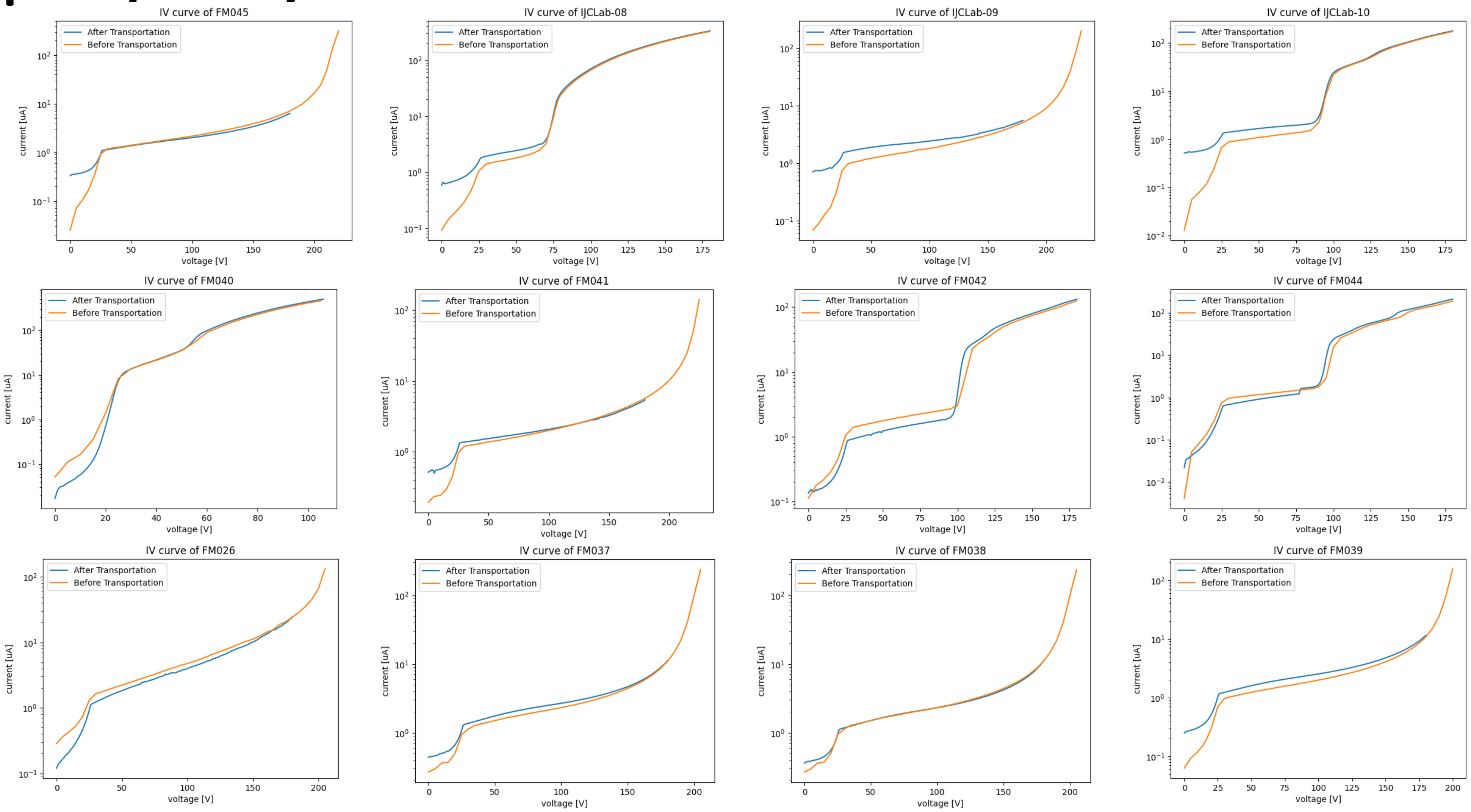
- The modules will be loaded onto a 3D-printed plastic frame, referred to as the support unit.
- A new detector unit has been delivered during last HGTD week. This DU is used to replace the dummy DU.
- After the loading and transportation, the reception test has been performed on the new DU
- The test setup uses the FADA pro, and the same HV source and controlling script as the USTC is used.
- The detector unit has been installed on the demonstrator.
- The impact from loading and transportation is minimal.

Test result

Module\Difference	disconnected	noisy	luminosity	Current @ 120V
FM026	0:14,224	none	0:work 1:not work	15.42%
FM037	none	none	0:work 1:work	-11.9%
FM038	none	none	0:work 1:work	2.55%
FM039	none	none	0:not work 1:work	-23.08%
FM040	1:1 (connected)	1:157,158	0:work 1:not work	-2.88% (@50V)
FM041	none	none	0:not work 1:work	-0.85%
FM042	none	none	0:not work 1:work	-16.52%
FM044	none	1:180	0:not work 1:work	-12.54%
FM045	none	none	0:work 1:work	9.91%
IJCLab-08	none	1:196	0:not work 1:not work	-3.38%
IJCLab-09	0:224 1:195, 210	none	0:work 1:work	-25.81%
IJCLab-10	1:210	0:70	0:not work 1:work	-3.38%

- Colors are used to differentiate results between two tests, green for same, red for different.
- If test result is different, the result is what we get at CERN.

IV property check



- We check the IV from 0 to 180V with current limitation at 500 μA.
- For most of the early breakdown modules, noisy channels can be found, except for FM042.

Hgtd-peb, temporary solution

1. PEB

PEB configuration includes lpGBT configuration and VTRX+ configuration.

For lpGBT configuratin, it enables the bpol12vs to power on the front-end modules, sends primary clocks to the front-end modules, sets the data rate of the eLinks and also sets the pre-empahsis for signal transmission.

For VTRX+ configuration, it sets the pre-empahsis for signal transmission and enable the lumi channels for T0~T2.

Channels 0~2. Pattern: Mix046

```
python3 initialization.py -F ./config/lpGBT/mix046.yelc -d 0 -G 0
python3 initialization.py -F ./config/lpGBT/mix046.yelc -d 0 -G 1
python3 initialization.py -F ./config/lpGBT/mix046.yelc -d 0 -G 2
```

Channels 3~5. Pattern: Mix230

```
python3 initialization.py -F ./config/lpGBT/mix230.yelc -G 3
python3 initialization.py -F ./config/lpGBT/mix230.yelc -G 4
python3 initialization.py -F ./config/lpGBT/mix230.yelc -G 5
```

Channels 6~8. Pattern: Mix310

```
python3 initialization.py -F ./config/lpGBT/mix310.yelc -G 6
python3 initialization.py -F ./config/lpGBT/mix310.yelc -G 7
python3 initialization.py -F ./config/lpGBT/mix310.yelc -G 8
```

The configurations above are for the final detector according to the different data rates for different modules. If you only do the scanning test, for convenience, you can configure the data rate of all lpGBT eLinks to 320 Mbps by running the following command:

```
python3 initialization.py -F ./config/lpGBT/all_320M.yelc -d 0 -G <0/1//2/3/4/5/6/7/8>
```

Note: All above commands may not succeed at once, so if it fails, you need to run it again.

After above configuration, the front-end modules are powered on, now you can communicate with them through I2C bus.

You need too have a first time monitoring now, to find if there's anything wrong in temperature, GND, VDDA and VDDD:

```
python3 monitor_interested.py -G <0/1//2/3/4/5/6/7/8> > monitor_before_configure.txt
```

You need to pay attention to PEB-related monitor results -- if temperature makes sense, if GND almost equals 0, if VDDA and VDDD are not too low. At this stage, don't be so nervous if any ERROR found in altiroc monitoring results, they may fix after configuration.

Next step is to check the I2C bus is ok or not, you can try the following commands:

```
python3 i2c_check.py -d 0 -G <0/1//2/3/4/5/6/7/8>
```

The script i2c_check.py tries to read out some register values of the front-end modules, if any abnormal found, you will find a message looks like `!!!!!!!!!!!!I2C wrong number, may need configuration!!!!!!!!!!`

2. Front-end module

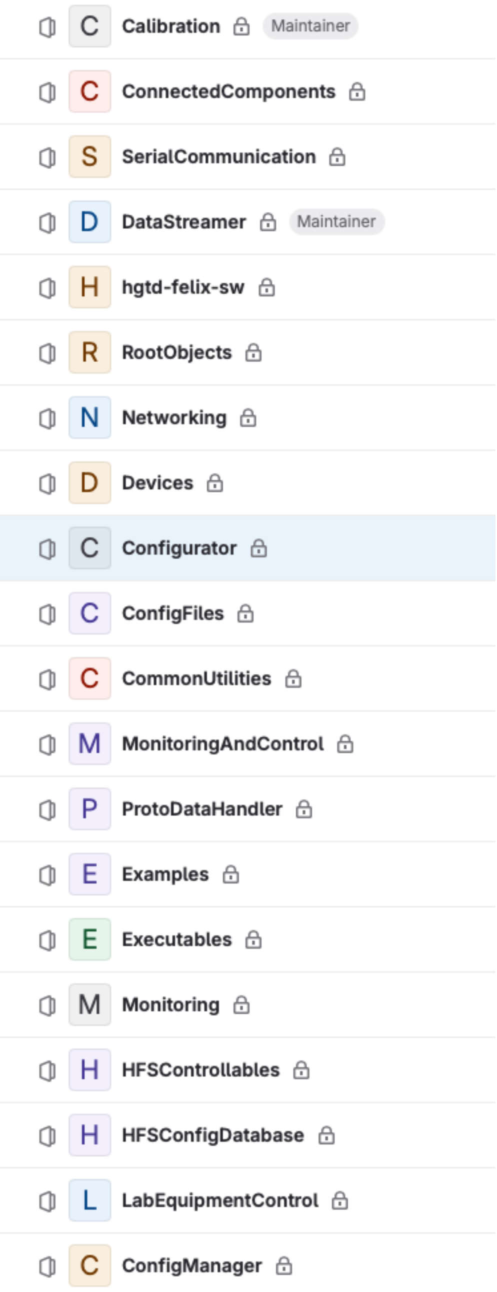
Now need to configure modules for further tests:

```
python3 configure.py --pixelOn col7 --pixelInj col7 --dacCharge 24 -F ./config/altiroc/mix046.yelc -d 0 -G 0 <0/1//2/3/4/5/6/7/8>
```

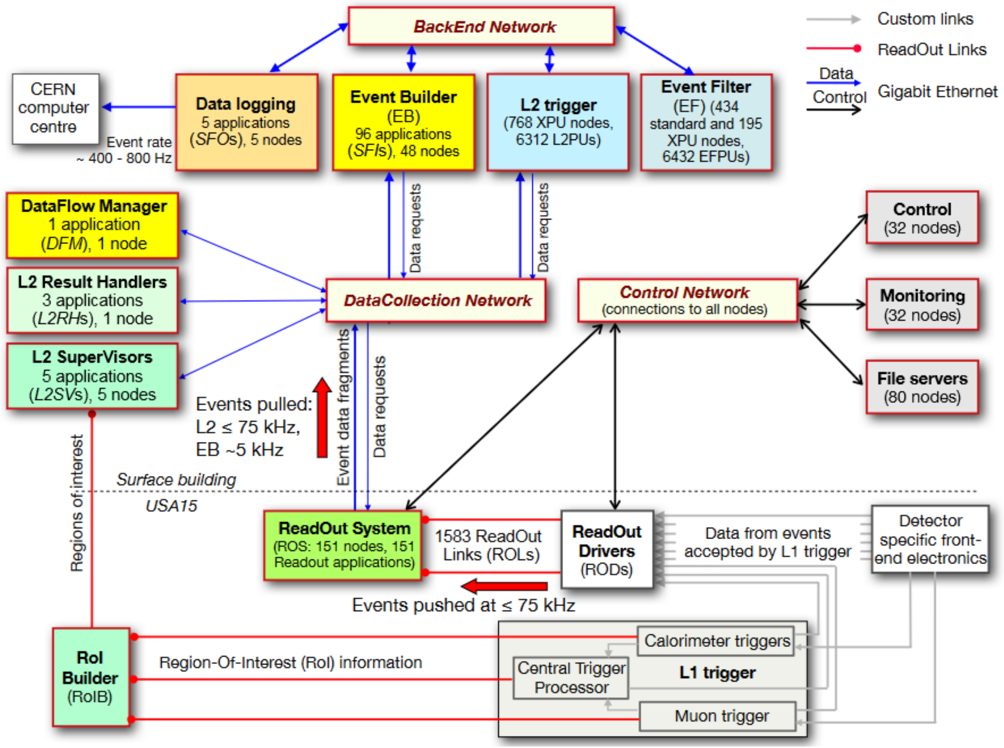
- The hgtd-peb is used for debugging tests of the demonstrator.
- This package utilizes a C++ library written by Alex, which will interact with felix card.
- The concept is similar to FADA, which can be used for configuration and basic scanning.
- The code is focusing on the read/write of register for lpGBT VTRx+
- For the running of the code,
  1. Initialize the FELIX card, and align the link with VTRx+
  2. Initialize the lpGBT, which will adjust the configuration for lpGBT and VTRx+, also enable the b-pole, which will provide power to modules.
  3. At this moment, the module should work properly, we can check the related monitoring bias with the MUX64 and the ADC in lpGBT.
  4. We can start the scanning to check the status of the demonstrator.

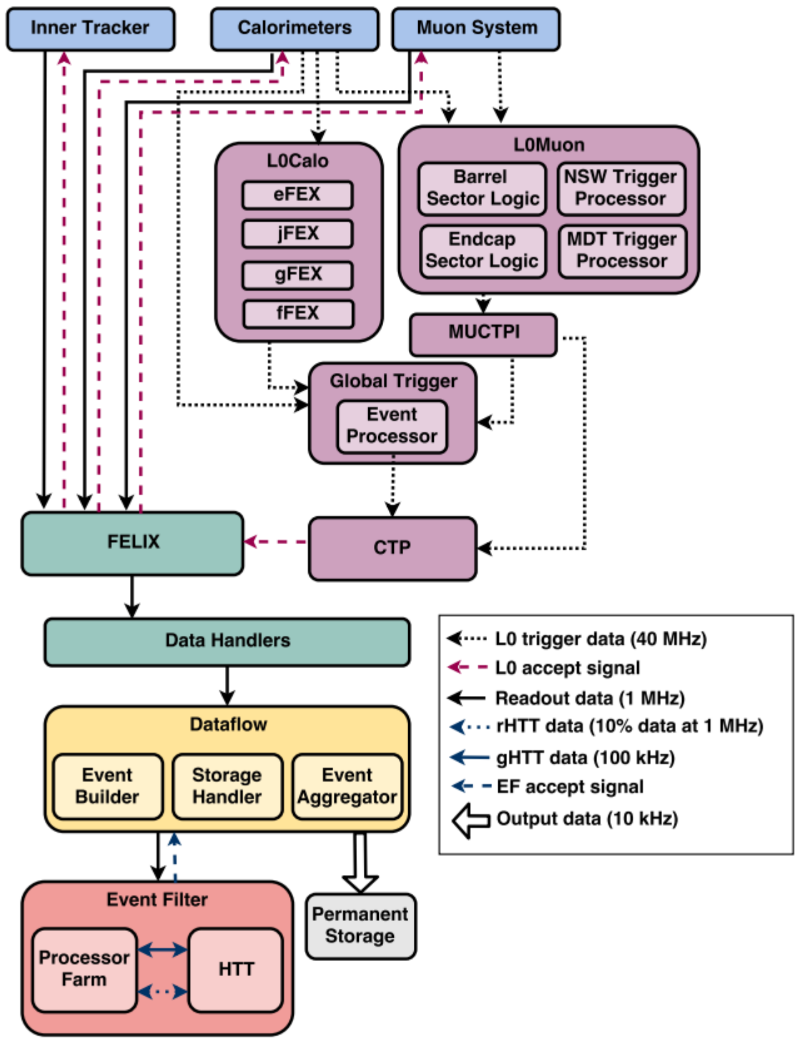
hgtd-felix-sw

- Next attempt to get closer to the final situation.
- This DAQ software uses TCP/IP to pass messages between different processes, which makes following function possible.
    1. The distribution of process to different server.
    2. Dividing different functions into separate processes.
    3. Scaling up the capability for data processing.
    4. Develop the different package independently and parallelly.
- Beyond this we also develop other function.
    1. Web UI for the operation and the status displaying.
    2. Collecting data from different lpGBT and according to the BCID
    3. Online monitoring of all the monitoring bias from lpGBT.
    4. Online scanning result analysis, and configuration result extraction (ongoing).



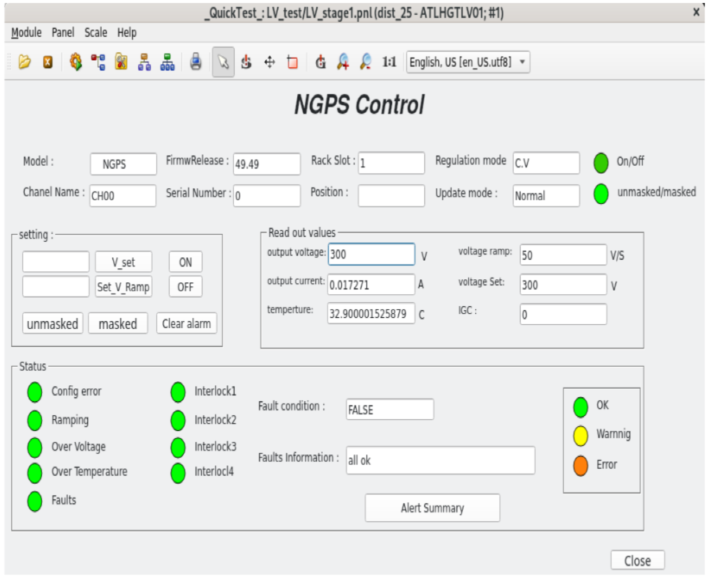
ATLAS Trigger and Data AcQuisition (TDAQ)



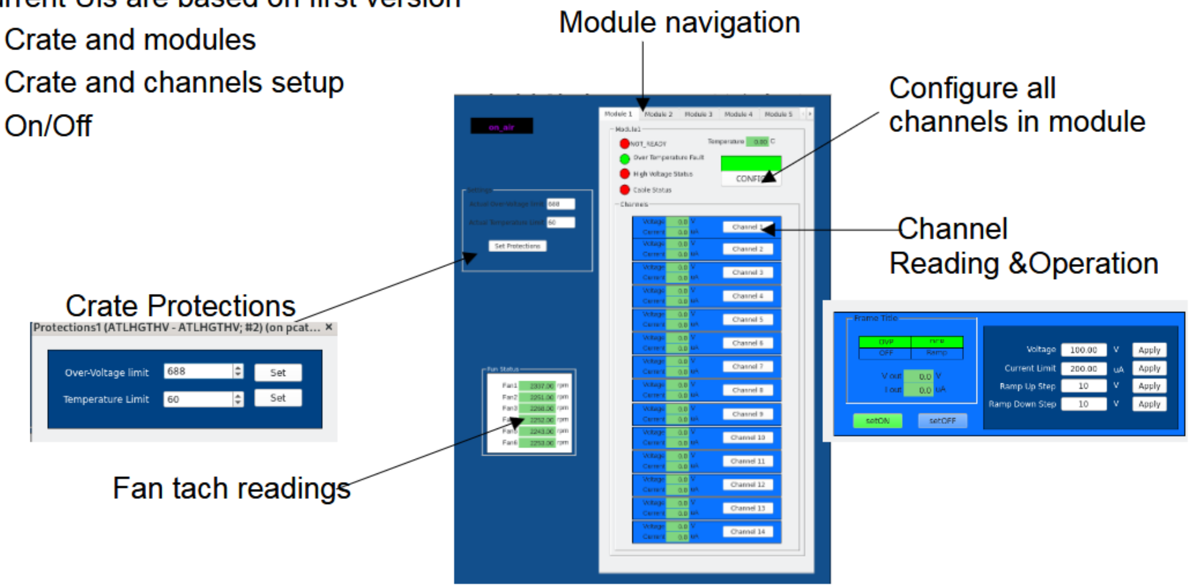


- ATLAS has dedicated TDAQ structure. Our target is to merge the code we developed.
- Besides the application shows in the diagram, we need to ensure the communication between process, error report, and a configuration process and monitoring.

DCS



- Current Uis are based on first version
  - Crate and modules
  - Crate and channels setup
  - On/Off



- For the Low voltage controller, an GUI controller based on the WinCC and OPC UA has been developed for stage 1 and stage 2. This software also provides monitoring.
- The development for HV control is also based on the WinCC and OPC UA and still on-going.

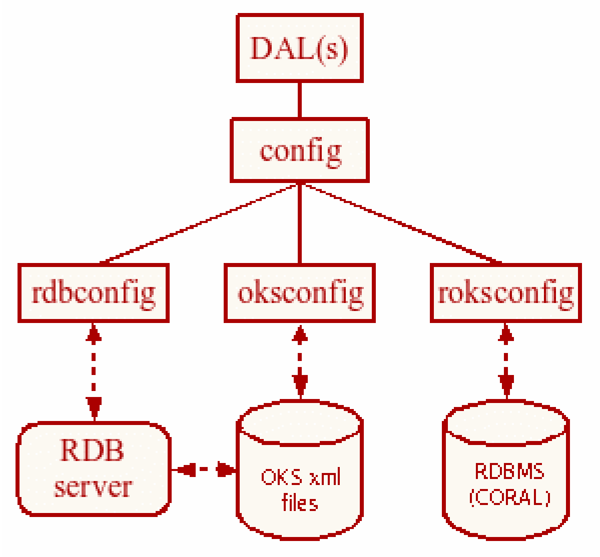
ATLAS TDAQ infrastructure

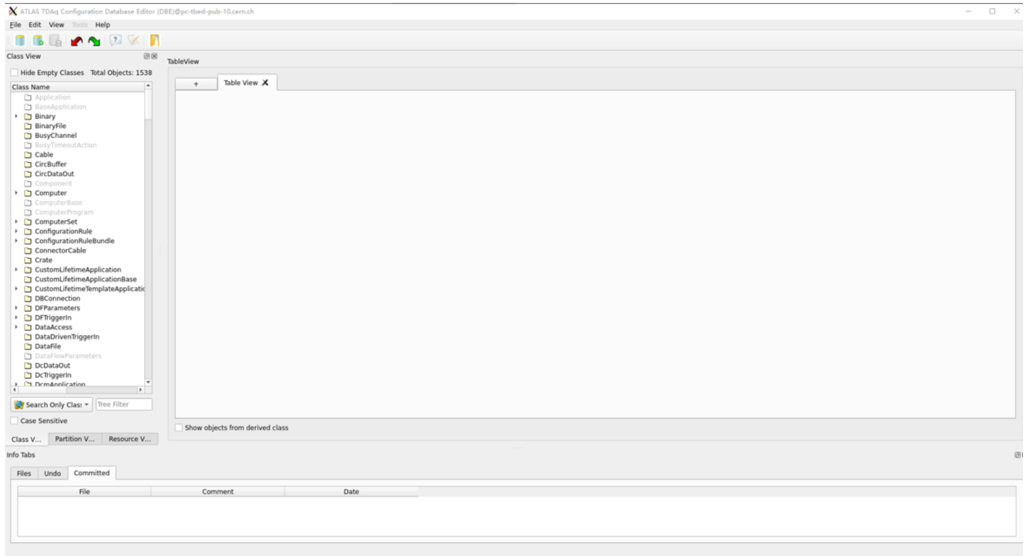
Service Name	Description
Access Manager	This service allows to introduce authorisation for actions based on users/roles. The same role based access management model is used in ATLAS at system administration level and in the Detector Control System
Configuration Service	This service allows to prepare a configuration for a data taking session, access the configuration from applications, archive configurations that have been used to take data (i.e. associated to a run number). The configuration service is central to the TDAQ, as it defines the content and behaviour of any data taking session. Due to its critical nature, at the experiment site operations are controlled via the Access Manager.
Resource Manager	This service allows to lock TDAQ resources for exclusive or limit-shared usage as described in the TDAQ configuration database. As an example it can be used to allow a limited number of application instances to run globally, or per partition, or per host.
Process Manager	This service allows to start, stop and monitor TDAQ processes. It uses the Resource Manager and the Access Manager to evaluate whether an operation can be carried out based on the resources utilisation and the requesters privileges.
Message Transport System	This service allows ERS messages to be exchanged across applications using a publish/subscribe scheme.
Run Number Service	This service allows to retrieve a unique run number that can be used for a data taking session. Tun numbers are meant to be unique throughout the lifetime of the ATLAS experiment.
Enabled Resources Information Service	This service allows to dynamically change the status of resources during a run and keep track of their status.
Test Manager	This service allows to execute tests on different components as described in the TDAQ configuration database.
Diagnostics and Verification	This service uses the Test Manager to execute sequences of tests on different components in order to verify the functioning of the TDAQ or diagnose problems.
Run Control	This service allows to coherently control a data taking session as configured in the TDAQ configuration database. The Run Control uses most other services to carry out its task.
Central Hints and Information Processor	This application processes the information from the TDAQ Configuration, the Run Control, the Information Service and the Error Reporting System to detect and handle errors, as well as perform automated actions on the TDAQ, such as changing the experiment's clock, perform the warm start, etc.. The documentation of this package also contains the description of automated procedures in ATLAS.
Log Service	This service allows to archive ERS messages and visualize them via a Java application or web based dashboard.
IS Archiver, PBeast	This service allows to archive permanently IS information for post-mortem analysis of data taking sessions.

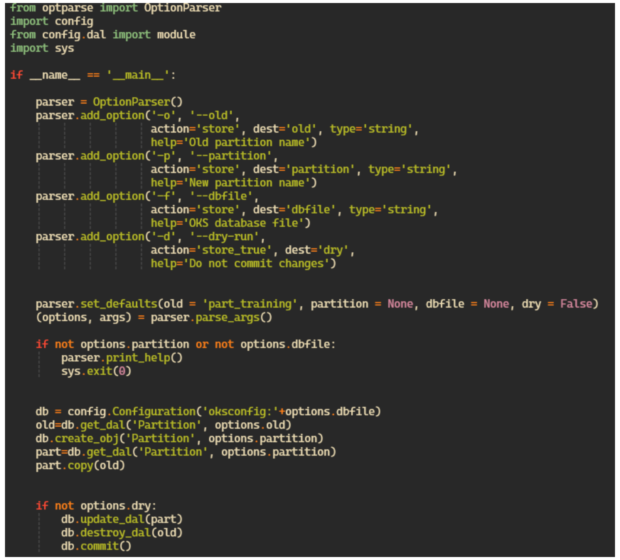
Tool Name	Description
Integrated Graphical User Interface, IGUI	This Java based tool allows the operator to control and monitor data taking sessions. It can be customised to include additional views and control panels.
Shifter Assistant	This application processes information from different sources, including TDAQ, analyses the status and behaviour of the ATLAS experiment, and, when appropriate, issues alerts to the operators and experts via Web, e-mail, SMS, ... This tool is customisable by system experts to include additional processing directives and alerts. The documentation of this package contains the list of defined directives.
ELisA logbook	This web application provides the electronic logbook for the ATLAS experiment. Messages can be inserted and visualised interactively over the web, or programmatically.
P-Beast Dashboard	This web application shows IS objects provided by P-Beast server over dashboards. Dashboards can be customised according user's needs to add or remove panels, queries and other features.
DBE	This tool allows the user to edit configuration databases.

- All the service is based on tcp protocol, and uses a CORBA layer through the inter-process communication (IPC) and error reporting system (ERS) software.
- So many process is needed for setting up the TDAQ system, we need another software to record where and how to setup all the software. We need OKS database.

Object Kernel Support Database







```python
from optparse import OptionParser
import config
from config.dal import module
import sys

if __name__ == '__main__':

    parser = OptionParser()
    parser.add_option('-o', '--old',
                      action='store', dest='old', type='string',
                      help='Old partition name')
    parser.add_option('-p', '--partition',
                      action='store', dest='partition', type='string',
                      help='New partition name')
    parser.add_option('-f', '--dbfile',
                      action='store', dest='dbfile', type='string',
                      help='OKS database file')
    parser.add_option('-d', '--dry-run',
                      action='store_true', dest='dry',
                      help='Do not commit changes')


    parser.set_defaults(old = 'part_training', partition = None, dbfile = None, dry = False)
    (options, args) = parser.parse_args()

    if not options.partition or not options.dbfile:
        parser.print_help()
        sys.exit(0)


    db = config.Configuration('oksconfig:'+options.dbfile)
    old=db.get_dal('Partition', options.old)
    db.create_obj('Partition', options.partition)
    part=db.get_dal('Partition', options.partition)
    part.copy(old)


    if not options.dry:
        db.update_dal(part)
        db.destroy_dal(old)
        db.commit()
```

- Object Kernel Support (OKS) basically is a database with very few functions, the data is stored in an XML files or a remote Oracle database.
- The data includes:
  - The definition of the class, which named the scheme.
  - The instances of these classes, which will ultimately be used as the configuration.
- The OKS file can be checked and edit using both GUI and process API.

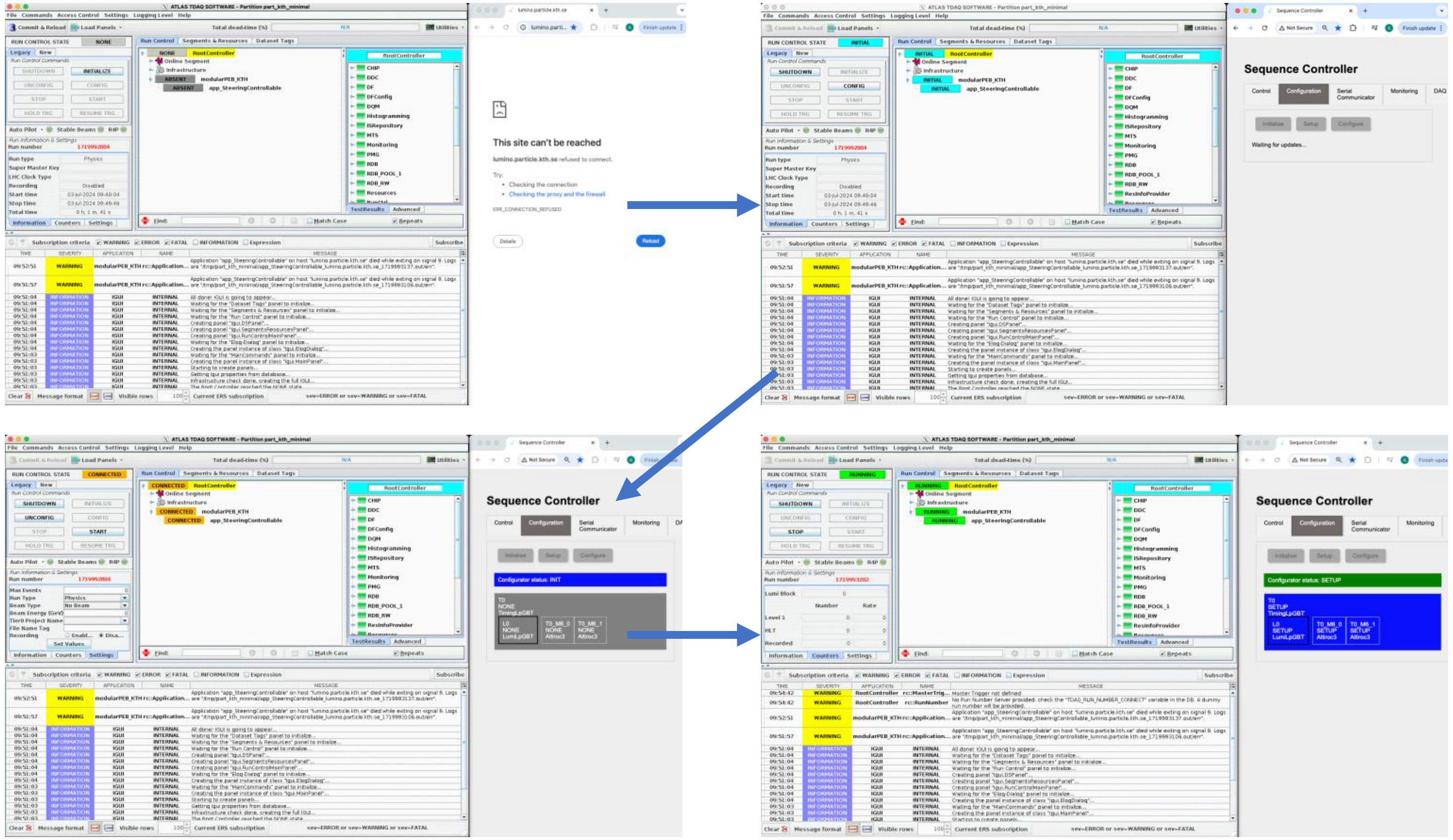
Partition and segment

```
$ tree data/Training/
data/Training/
├── hw
│   └── hw.data.xml
├── partitions
│   └── part_training.data.xml
├── schema
│   └── training.schema.xml
├── segments
│   └── ddc_test.data.xml
└── sw
    ├── packages.data.xml
    └── resources.data.xml
```

```xml
<obj class="Partition" id="part_training">
 <attr name="IPCRef" type="string" val="$(TDAQ_IPC_INIT_REF)"/>
 <attr name="DBPath" type="string" val="$(TDAQ_DB_PATH)"/>
 <attr name="DBName" type="string" val="$(TDAQ_DB_DATA)"/>
 <attr name="DBVersion" type="string" val="$(TDAQ_DB_VERSION)"/>
 <attr name="DBTechnology" type="enum" val="rdbconfig"/>
 <attr name="LogRoot" type="string" val="/tmp"/>
 <attr name="WorkingDirectory" type="string" val="/tmp/${TDAQ_PARTITION}"/>
 <rel name="Segments">
  <ref class="Segment" id="TDAQ"/>
  <ref class="Segment" id="MyDetector"/>
 </rel>
 <rel name="OnlineInfrastructure" class="OnlineSegment" id="setup"/>
 <rel name="DefaultHost" class="Computer" id="default_host.cern.ch"/>
 <rel name="DefaultTags">
  <ref class="Tag" id="x86_64-centos7-gcc11-opt"/>
  <ref class="Tag" id="x86_64-centos7-gcc11-dbg"/>
  <ref class="Tag" id="x86_64-centos9-gcc11-opt"/>
  <ref class="Tag" id="x86_64-centos9-gcc11-dbg"/>
 </rel>
 <rel name="ProcessEnvironment">
  <ref class="VariableSet" id="CommonEnvironment"/>
 </rel>
 <rel name="Parameters">
  <ref class="VariableSet" id="CommonParameters"/>
 </rel>
 <rel name="DataFlowParameters" class="DFParameters" id="Default-DFParameters-1"/>
 <rel name="IS_InformationSource" class="IS_InformationSources" id="part_training-counters"/>
 <rel name="MasterTrigger" class="MasterTrigger" id="HLTSVMaster"/>
 <rel name="TriggerConfiguration" class="TriggerConfiguration" id="PuDummyTriggerConfig-1"/>
 <rel name="ResourcesInfoConfiguration" class="ResourcesInfoConfig" id="Training"/>
</obj>
```

- All the component needed in the system must be defined in the database:
  - HW, including computer, network and so on.
  - SW, the software, the environment variable, the resources needed by process.
- A segment essentially includes all the configuration needed by an application, that is composed of resources, applications and nested segments.
- The partition serves as the root of other objects and is the starting point for initiating a TDAQ system.

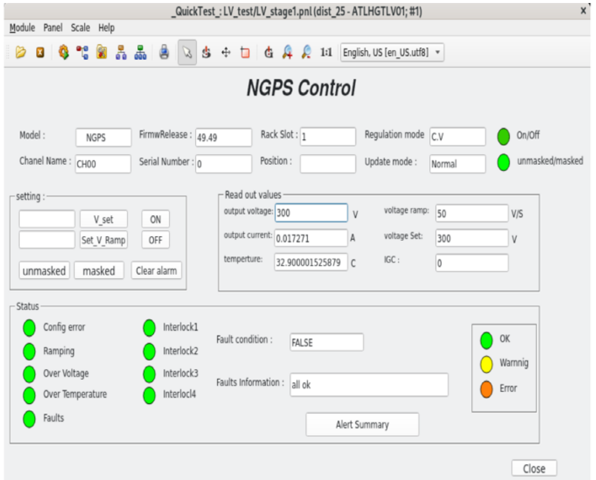
hgtd-felix-sw2

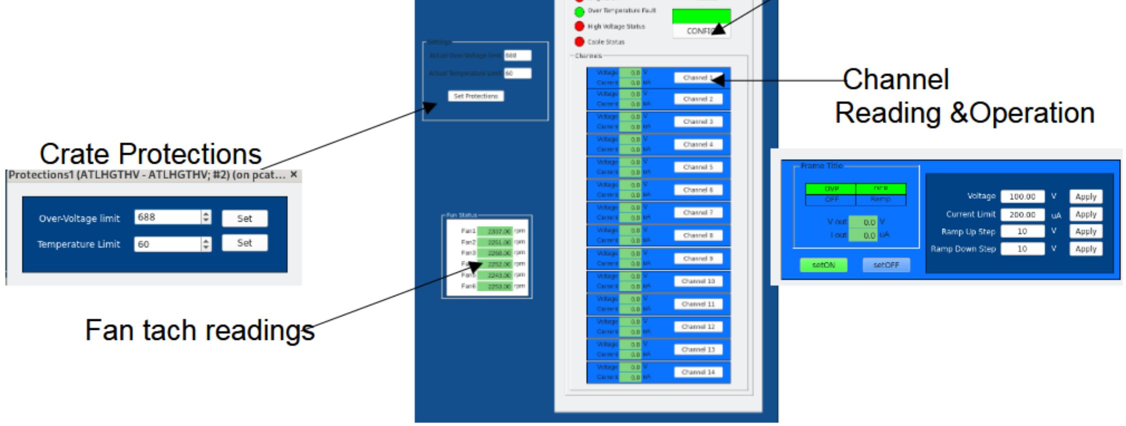


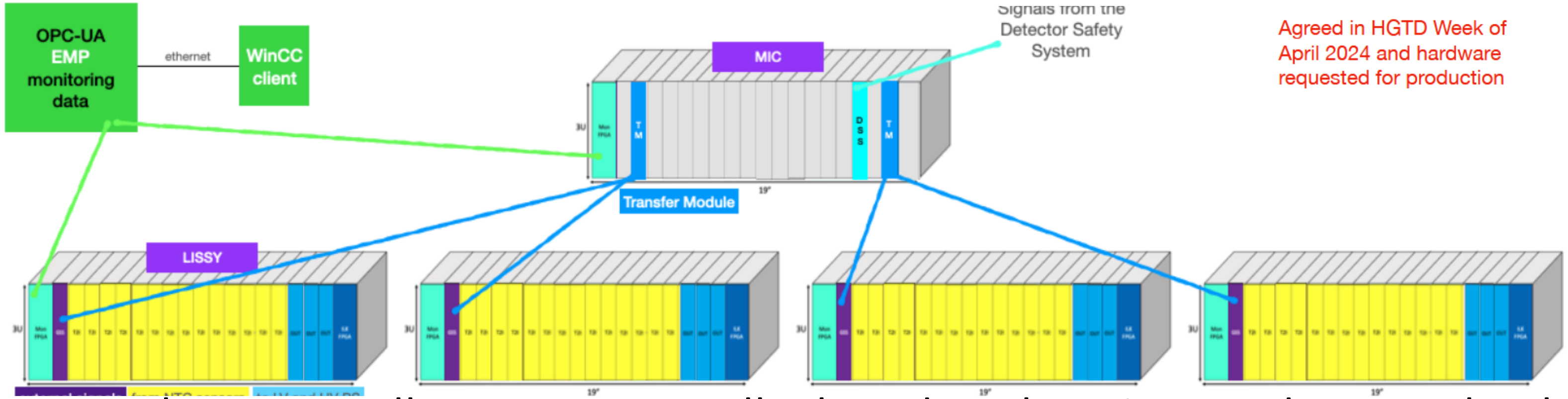
- The application can be coordinated by the ATLAS TDAQ software needs to follow the finite state machine (FSM) and the state transfer function.
- We have finished some attempt to utilize this suite to coordinate a demo process.
- We still focus on the previous version, for this version, we are now learning to use the infrastructure.

DCS and Interlock

- Current Uis are based on first version
  - Crate and modules
  - Crate and channels setup
  - On/Off







Agreed in HGTD Week of April 2024 and hardware requested for production

- For the Low voltage controller, an GUI controller based on the WinCC and OPC UA has been developed for stage 1 and stage 2. This software also provides monitoring.
- The development for HV control is also based on the WinCC and OPC UA and has not yet been integrated into the demonstrator.
- An Inter-lock system is under the development, which will cut off power when over heating is detected