CERN

Flexible configuration and monitor system for medium scale detector tests

at CERN

Rui Filipe Teixeira Alves

Supervisor: Nicola Minafra

Supervisor: Tommaso Isidori

September 6, 2023

Abstract

The ALICE (A Large Ion Collider Experiment) project at CERN explores the fundamental properties of matter through heavy-ion collisions. During the Large Hadron Collider (LHC) long shutdown scheduled to start in 2027, the experiment will undergo the installation of new detector systems for expanding the current physics program. In particular, the FoCal (Forward Calorimeter) aims to enhance our understanding of small-x gluon distributions and investigate the dynamics of hadronic matter.

This report presents the design and implementation of a comprehensive configuration and monitoring system for the FoCal project, with a particular focus on the versatile *Parsing System* and the intuitive web-based interface, called *RuiGUI*.

The *Parsing System*, the central component of the system architecture, allows abstract and flexible configuration of different devices. Utilizing YAML files, detector experts define device properties and parsing rules, enabling the seamless integration of terminal-based and web-based data sources. This system efficiently extracts critical data from devices, ensuring accurate and reliable data collection.

RuiGUI, the web-based interface, enables users to monitor and configure devices with ease. The interface features two tabs: *Configurations*, where users define variables to monitor and configure instances, and *Visualization*, which displays real-time insights into device status and performance. *RuiGUI* streamlines device oversight, and enhances operational robustness, ultimately contributing to improved data quality.

The success of this project lies in the ability of the *Parsing System* to accommodate various devices and the user-friendly *RuiGUI* interface, providing detector experts and operators with tools for efficient device management.

The ideal application of this solution is to simplify the characterization of medium-complexity systems. In fact, small systems don't usually justify the use of a centralized control and monitoring interface. On the other hand, large systems have a dedicated Detector Control System (DCS), Medium scale systems, or prototypes of large detectors (as for the FoCal R&D) would benefit greatly from a centralized system, without compromising on the flexibility needed during the development phase.

Keywords

ALICE, LHC, FoCal, monitoring system, controlling system, *Parsing System*, *RuiGUI* interface, device configuration, web-based interface, data visualization, data quality, heavy-ion collisions, small-x gluon distributions, hadronic matter dynamics, CERN, electromagnetic calorimeter, spaghetti hadronic calorimeter.

Contents

1 Introduction

The **ALICE** [1] (A Large Ion Collider Experiment) is one of the major experiments conducted at the Large Hadron Collider (LHC) [2] at CERN. *ALICE* is specifically designed to study the properties of strongly interacting matter at extreme energy densities, which are created in heavy-ion collisions.

ALICE focuses on understanding the behavior of quarks and gluons, the fundamental building blocks of matter, under conditions of high temperature and energy density. These extreme conditions can be achieved by colliding heavy ions, such as lead nuclei, at ultra-relativistic speeds.

It contains a wide range of detectors and sub-detectors, each designed to measure different properties of the particles produced in heavy-ion collisions. These detectors include tracking systems, calorimeters, particle identification detectors, and more. The data collected from these detectors provide valuable insights into the complex physics of these high-energy collisions.

1.1 FoCal Background

The forward electromagnetic and hadronic calorimeter (**FoCal** [3]) is an upgrade to the *ALICE* experiment. Thanks to the combination of a highly granular Si+W electromagnetic calorimeter and a transversally segmented spaghetti hadronic calorimeter, *FoCal* will provide unique capabilities for the measurements of small-x gluon distributions via prompt photon production and will significantly enhance the scope of *ALICE* for inclusive and correlation measurements with mesons, photons, and jets to explore the dynamics of hadronic matter at small x down to about 10^{-6}.

The **FoCal-E**, shown in Fig. 1, consists of a Si+W sampling calorimeter hybrid design using two different Si readout technologies:

- **18 Pads**: 18 Si "low granularity" detectors ($\approx 1cm2$) that are needed for collecting the energy of the showers. These are interleaved with layers of absorbers (Tungsten plates)
- **2 Pixels**: *FoCal* needs 2 "high-granularity" layers (positioned in L5 and L10) for having a space resolution good enough ($\approx 30um2$) to distinguish tracks of two electrons with a minimum distance of 5mm at the surface

Independent front-ends are used to provide triggers, slow control, and read out the data coming from the two FoCal-E subsystems. A custom-made board, called **Aggregator**, collects the signals generated by each of the Si pads layers, processes them in FPGA, and provides clock and slow control via Ethernet. The aggregator is connected via fiber (GBT protocol) to one of the CRUs installed in the FLP.

The pixel layers communicate with the CRU using dedicated Readout Units (RUs). This electronic chain, as well as the sensors technology, were inherited from the Inner Tracking System (ITS) of the ALICE detector and re-adapted for the FoCal physics program.

1.2 Motivation and Expected Results

Efficient monitoring and control of the devices are paramount for the successful execution of experimental procedures and the acquisition of accurate and reliable data. As the complexity and scale of scientific endeavors continue to expand, it becomes increasingly challenging to maintain effective oversight of

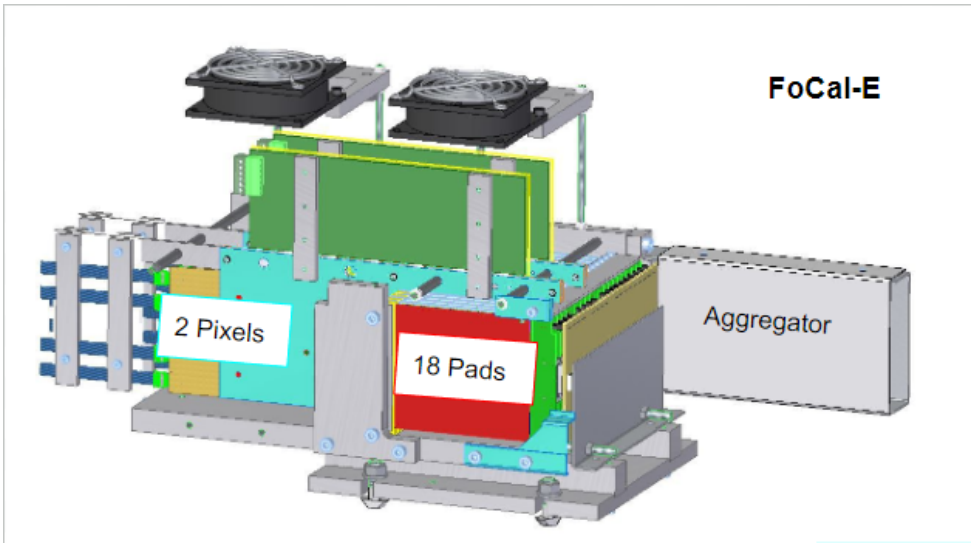

Figure 1: Three-dimensional rendering of the FoCal-E detecting apparatus. The position of the 18 layers of Si pads and 2 high granularity Si pixels are highlighted in the figure.

numerous interconnected systems simultaneously. In the case of the *FoCal* project, this challenge is evident in the need for workers to manage and control multiple devices, such as the pixels and pads detectors, often requiring them to monitor their outputs concurrently.

At the current prototyping stage, the workflow for monitoring and controlling the sub-devices necessitates the simultaneous use of multiple terminals, each displaying critical information from distinct components. This disjointed approach poses several significant drawbacks. First and foremost, the manual coordination of multiple terminals demands considerable cognitive effort and attention from the operators. Maintaining focus across multiple displays often leads to potential errors due to information overload or inadvertent oversights.

The management of certain devices may necessitate a specialized level of familiarity and technical understanding. This expertise is particularly crucial during the configuration phase, as these devices require specific settings and adjustments that may not be readily apparent to the operator during regular usage and control.

In response to these challenges, the development of an integrated monitoring and controlling system for *FoCal* devices becomes not only a matter of practicality but also an imperative for the effectiveness of the users.

Centralizing the monitoring and control processes into a unified and user-friendly interface will empower workers to more efficiently and easily oversee the operation of the devices, ensuring timely responses to anomalies and improved data quality. By storing the data persistently, the system will allow the detector experts to track old data or study the existence of patterns in error detection, using the previous image of the devices behavior. Furthermore, the system will enhance the quality of recorded data by ensuring online visual feedback when data values are outside critical or warning required values. This comprehensive approach will help identify and rectify errors promptly, ultimately leading to a reduction in the instances where errors go unnoticed.

Allowing the users to define *a-priori* fixed configurations of the system, the operator itself does not need the training and understanding of the system to monitor it, greatly simplifying long test campaigns, both in the lab and during beam tests.

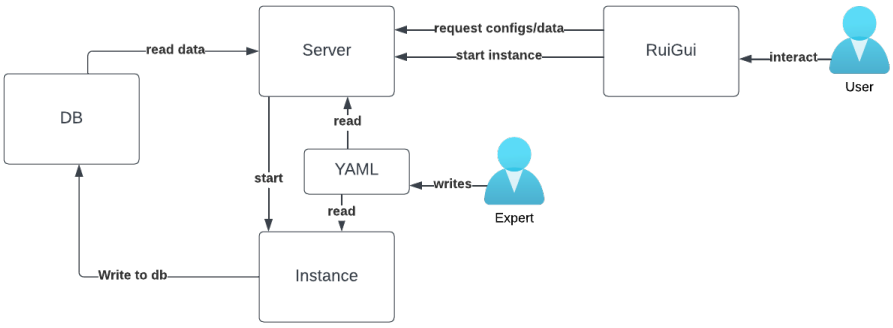

Figure 2: Architecture of the system and its relations. It consists of 4 main components: Instance, Server, DB, and *RuiGUI*.

2 System Architecture and Design

The system architecture, shown in Fig. 2, is composed of 4 main components:

- **Instance**: An instance is the centralized component of the system. It starts the device and all the necessary elements to parse its content. It can be of type *terminal* or *webpage*, according to how the output is provided by the device specified. The instance is also defined in a YAML file.
- **Server**: The Server plays the role of an intermediary between the user and the rest of the system. By communicating with the database and creating instances, it is the controller of the system and performs actions based on the users' input on the interface.
- **DB**: is the database where the data is stored. It should be preferably a time series database. The main DB used is *InfluxDB*, but *PostgreSQL* is also supported for convenience.
- **RuiGUI**: it represents the user interface that displays the configuration of each device and the respective monitored data. It is used to provide the last data monitor and possible errors on that data.

The interactions between each component are also shown in Fig. 2. The detector expert writes the *YAML* files describing each device and how the system should get the data from it, which is then read by the *Server* to create configurations. When the operator interacts with *RuiGUI* to get the output provided by the devices, the server creates an instance based on the *YAML* file for this device, which writes the data obtained from the device to the *DB*. The *Server* reads then that data from the *DB* and retrieves it to *RuiGUI*, displaying it to the user.

2.1 Parsing System

The *Parsing System* consists of the system that's responsible for parsing the data from the device. Currently, the devices are outputting data into a terminal or directly to a web interface. This system parses the *YAML* files to obtain the configuration of each device and its output and proceeds to obtain its data. The data is then persistently stored in the *DB*.

This system, shown in Fig. 3 is composed of 6 main components:

- **Instance**: the centralized component explained above. It initializes all the necessary components for the correct parsing of the content for the specified device.

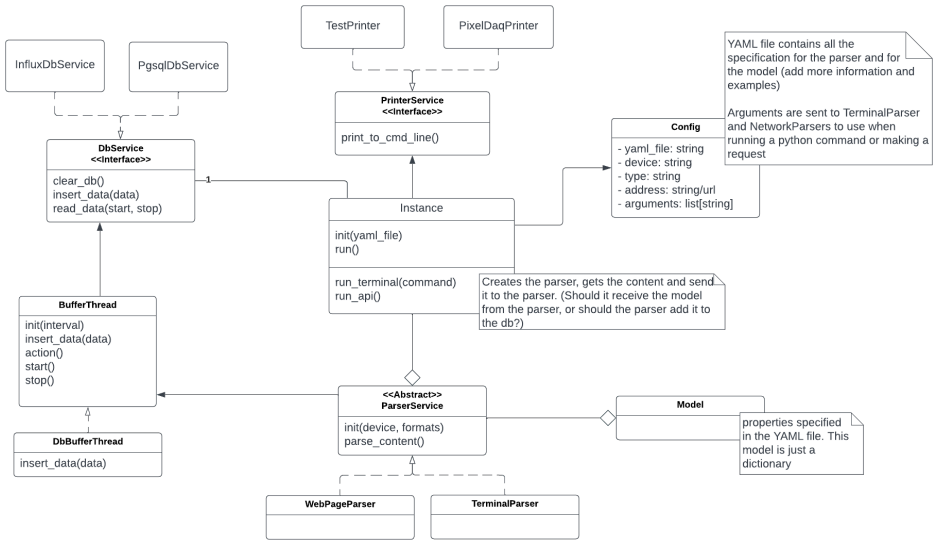

Figure 3: Architecture of the Parsing System. It is composed of 6 main components: Instance, ParserService, BufferThread, DbService, PrinterService, and Config

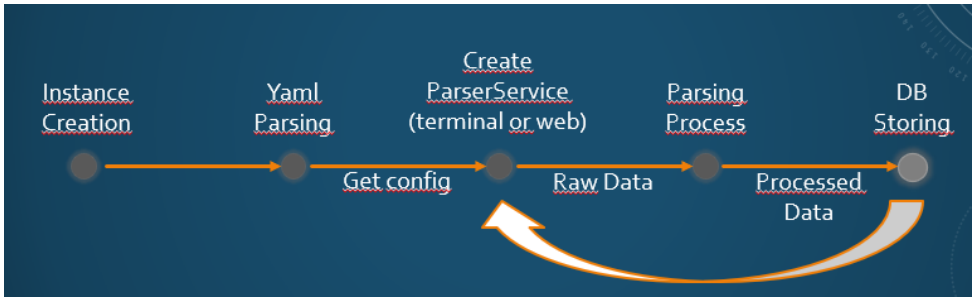

Figure 4: Sequential Diagram representing the Workflow of the *Parsing System*

- **Config**: the configuration component that reads the YAML files and parses the content. It is then passed to the instance to provide more information about the device to run.
- **ParserService**: the parser component that parses the content of the specified device. It receives the data from the instance and parses it according to the formats specified in the *YAML* file. It then sends the parsed data to the *buffer*. It can be of type *terminal*, parsed by a *TerminalParser*, or *webpage*, parsed by *WebPageParser*, according to how the output is provided by the device, also specified in the *YAML* file.
- **Buffer**: The data received from the parser are stored in a buffer. It also uses the information specified in the *YAML*, periodically storing the data in the *DB*.
- **DBService**: the component that makes the connection with the DB. It receives the data from the buffer and stores it. It is also used to retrieve the last data to the *Server* when requested. It can be of type *influxdb*, an *InfluxDBService*, or *pgsql*, a *PgsqlService*, according to the *DB* used.
- **PrinterService**: the printer service is used to simulate a physical device. This service should follow the output rules of the devices, replicating devices' behavior in the different situations they may be in, allowing a much easier debugging and study of the overall system.

A sequential diagram representing the workflow of this system is shown in Fig. 4.

```yaml
devices:
    - terminal
    - test
    - focalesp
    - aggregator
    - aggregatorDAQ
    - pixels

instances:
    terminal:
        device: terminal
        address: 192.100.100.100
    focalesp:
        device: focalesp
    test:
        device: test
    aggregator:
        device: aggregator
    aggregatorDAQ:
        device: aggregatorDAQ
        address: 192.168.100.100 -i -g -e -r normal -s -1 -f /home/flp/softs/testPadTimeOffsets/
        cwd: /home/flp/softs/aggregator-sw/aggregatorConsole/
    pixels:
        device: pixels
        address: -c ../config/daq_test_focal_ob.cfg
        cwd: /home/flp/softs/cru_daq_SPS_2023/cru_daq/pixel/software/py
```

Figure 5: ALL YAML file used to define all the instances and devices.

3 Configurations

3.1 All YAML File

To provide information about the device, the system uses *YAML* Files. On these files, the *detector experts* defines which variables the system should pass and all the required information for its good functioning, as explained below.

All devices are specified in *all.yaml* file, shown in Fig. 5. Along with this, also the instances are defined with the respective address/arguments and working directory.

Each instance takes 3 parameters to be defined:

- **device**: name of the device we want to run
- **address**: address of the device/parameters we want to pass into the device.
  - e.g: device: ‘aggregator‘, address: ‘192.168.100.100 -i -g -e -r normal -s -1 -f /home/flp/softs/testPadTimeOffsets/aggregator-sw/softParams/physicsParameters.ini‘
- **cwd**: directory from where we want to run the device

3.2 Device YAML File

To define a device’s properties, an additional YAML File must be created. In this file, the *detector expert* provides detailed information about the device and its functioning:

- **device**: name of the device
- **type**: type of the device/parsing system (‘terminal‘ or ‘webpage‘)

- **interval**: interval in seconds between each parsing
- **db_interval**: interval in seconds between each db insertion
- **path**: path to the running script or webpage URL
- **formats**: list of formats to parse

3.2.1 Format Structure

The device YAML files works with specified formats. These formats represent the variables that we want to collect from the output of the device, and the structure of the output to be parsed. Each format follows the following structure:

- **name**: name of the variable to insert in the DB
- **type**: type of the variable to insert in the DB. It supports the use of int, float, str, bool, list, and checkbox. The latter type is used on checkboxes of HTML tags. It will parse the HTML and check if it contains 'checked' on its text and behave accordingly.
- **number_variables**: the name of the variable, as the formats, are flexible, as they allow parsing multiple variables with the same format. This key allows the user to specify the values of each flexible variable specified in the name or format.
- **format**: The regular expressions to parse.
  - lines: list of regular expressions to parse. It contains multiple lines in case the user wants a specific pattern on sequential lines
- **critical_when_not**: This field specifies the critical values that the variable should take, otherwise, it will be considered as critical. It can be a list of values or a specific value that the variable should take.
- **warning_when_not**: This field specifies the warning values that the variable should take, otherwise, it will be considered as a warning. It can be a list of values or a specific value that the variable should take.

An example of the *YAML* file of a *terminal* and a *webpage* device is shown in Fig. 6 and Fig. 7.

3.3 Server Configurations

Different from the *YAML* files that represent the information about a device and instance, the configurations stored on the server represent the configuration of the visualization of each device that is being monitored, i.e., the variables that we want to display. These configurations are stored in the configs folder and contain the following information:

- **name**: Name of the configuration
- **instance**: Name of the instance that we want to run.
- **id**: Id of the configuration
- **device**: Name of the device that is being monitored.
- **type**: Type of the device that is being monitored (terminal, webpage).

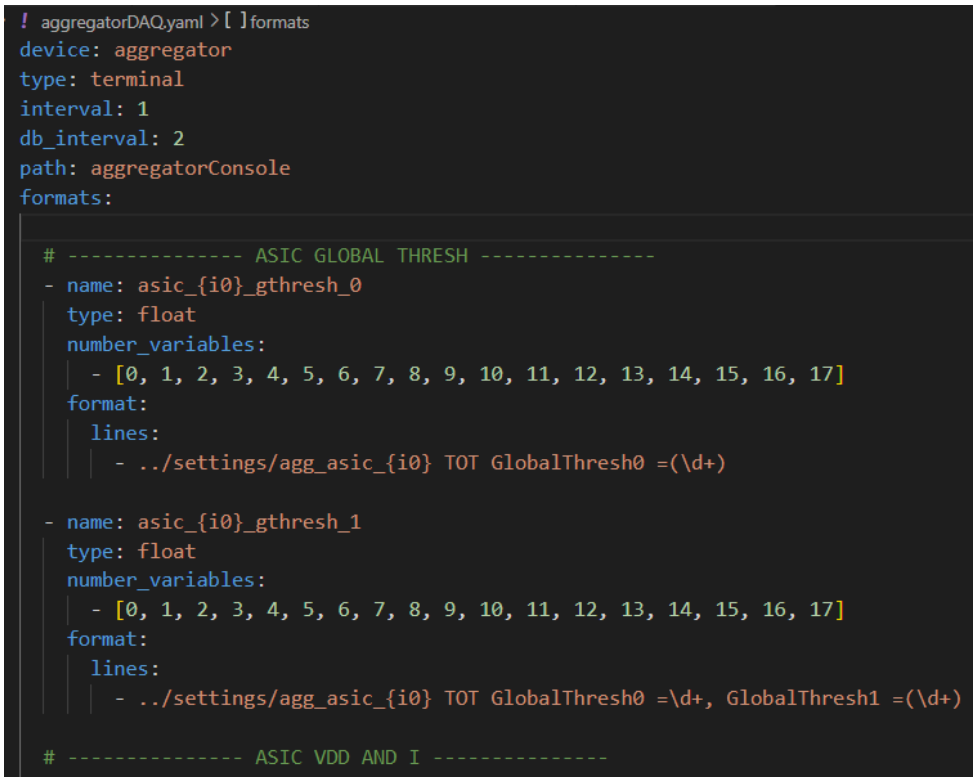

Figure 6: YAML file used to define the properties of a *terminal* device and variables to parse from its output.

- **address**: Parameters of the device that is being monitored (path, ip, etc.).
- **variables**: List of the variables that are being monitored.
  - *name*: Name of the variable.
  - *type*: Type of the variable (float, checkbox, etc.).
  - *critical_when_not*: List of the values that are considered critical.
  - *warning_when_not*: List of the values that are considered warning.
  - *visualize*: Boolean that indicates if the user wants to visualize the variable in the web interface.

One example is shown in Fig. 8

4 *RuiGUI* - Configurations and Visualization

The data acquired by the *Parsing System* is stored in the *DB*. Furthermore, the server reads from the database and provides the last inserted data to *RuiGUI*, which consequently displays it.

On the webpage, there are two tabs:

- **Configurations**: Here, the user defines which variables want to visualize and which instance wants to work on.
- **Visualization**: Here, all the variables that the user configured to visualize are displayed, showing their name, last value, and recording time.

Fig. 9 and Fig. 10 show, respectively, the Configurations and Visualization tabs and their content.

```yaml
device: aggregator
type: webpage
interval: 1
db_interval: 2
path: http://focalesp2/
formats:
  - name: total_av
    type: float
    critical_when_not:
      - [20, 50]
      - [10, 15]
      - 100
      - 0.0
    warning_when_not:
      - [30, 40]
      - [10, 20]
      - 0.0
    format:
      lines:
        - "<h2>Total currents: AV: (\\d+)"
  - name: total_dv
    type: float
    format:
      lines:
        - "<h2>Total currents: AV: \\d+   DV: (\\d+)"
  - name: ina_dv_m_enable
    type: checkbox
    format:
      lines:
        - "<input type=\"checkbox\" id=\"enableCh2\"(?:\\s+checked)?>"
  - name: total_current
    type: float
    format:
      lines:
        - "<h2>Total currents: AV: \\d+   DV: \\d+   TOT: (\\d+)"
```

Figure 7: YAML file used to define the properties of a *webpage* device and variables to parse from its output.

5 Conclusions

In this work, we have presented a comprehensive monitoring and configuration system for the devices within the ALICE *FoCal* project. The system addresses the challenges of efficiently managing and overseeing a complex array of devices, allowing for streamlined data collection, monitoring, and user interaction. The overarching goal of enhancing operational efficiency, reducing human error, and improving data quality has been central to the design and implementation of the system.

Throughout the course of this report, the components that constitute the system were explored, from the foundational device specifications to the interplay between the *Parsing System*, database, server, and user interface. With the use of YAML files to define device properties, configurations, and visualization preferences, it was achieved a flexible and adaptable framework that caters to the specific needs of the devices and detector users.

One of the most significant achievements of this system is the development of a powerful and adaptable *Parsing System*. This system forms the backbone of the entire monitoring and control framework, providing an abstract and flexible configuration mechanism. Through the utilization of YAML files, the *Parsing System* enables users to seamlessly define and parse a diverse set of instruments, including both terminal-based and web-based sources. This flexibility empowers the detector experts and operators to interact with and monitor devices with ease, irrespective of their specific characteristics. The *Parsing System* ability to dynamically adapt to various formats and data sources allows for efficient and accurate data collection.

```
{
    "name": "terminal",
    "instance": "terminal",
    "id": 2,
    "device": "aggregator",
    "type": "terminal",
    "address": "",
    "variables": [
        {
            "critical_when_not": [
                [0, 100]
            ],
            "name": "alignment_status",
            "type": "float",
            "visualize": true,
            "warning_when_not": [
                [0, 10]
            ]
        },
        {
            "critical_when_not": null,
            "name": "alignment_array",
            "type": "list",
            "visualize": true,
            "warning_when_not": [
                [true,true,true,true,true,true]
            ]
        }
    ]
}
```


Figure 8: YAML file showing a configuration on the server according to an aggregator device.

Furthermore, the *Parsing System* contributes to the core philosophy of the project by fostering a unified approach to device management. By centralizing configuration details and parsing logic, the system minimizes the complexities associated with handling multiple devices. This abstraction layer not only streamlines the configuration process but also promotes maintainability and scalability as new devices are integrated into the ALICE FoCal project.

As a result of these efforts, our system offers a holistic solution to the challenges posed by the management, monitoring, and control of *ALICE FoCal* devices. By unifying diverse components into a coherent and efficient framework, we have established a foundation for enhanced operational efficiency, improved data quality, and a simplified user experience.

While this report has outlined the design and implementation of the system, it also opens avenues for future enhancements and developments. As the *ALICE FoCal* project evolves and new devices are integrated, the presented system can serve as a foundation for continuous improvement and expansion. Potential areas for future work include the incorporation of advanced data analysis techniques, and the extension of the

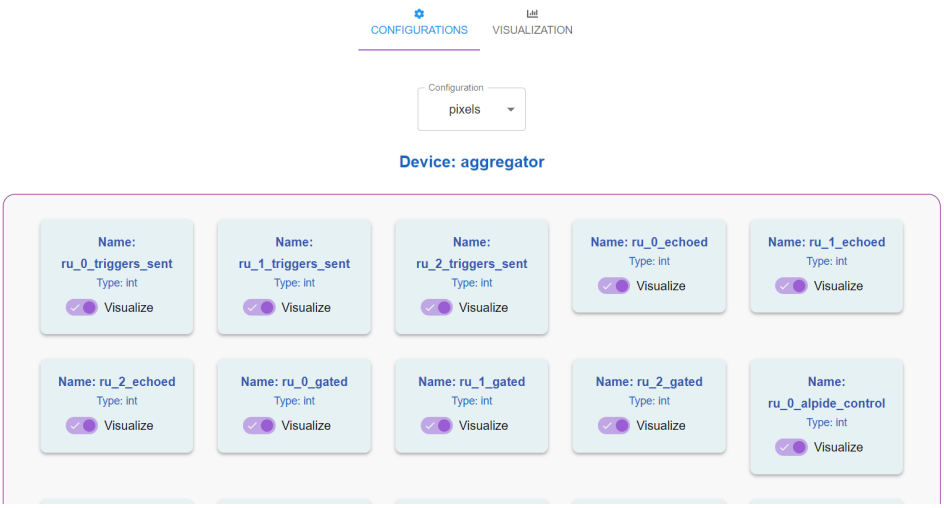

Figure 9: Configurations tab on the web page showing the configuration for pixels instance.

Figure 10: Visualization tab on the web page for pixels instance showing the last value of some variables.

system's capabilities to accommodate additional devices and experiments.

In conclusion, the monitoring and control system presented in this report represents a significant step toward realizing a more efficient and effective operation of the *ALICE FoCal* prototypes. By providing a cohesive framework for device management, data collection, and visualization, the system contributes to the ongoing pursuit of scientific discovery within the realm of heavy-ion collisions.

6 Acknowledgments

I express my heartfelt gratitude to Tommaso Isidori and Nicola Minafra for their invaluable guidance and unwavering support throughout this project. Their expertise and mentorship have been instrumental in shaping the development and success of the monitoring and controlling system for the ALICE FoCal project.

I would also like to express my sincere appreciation to the CERN Summer Student program for providing me with the opportunity to be a part of this remarkable scientific community. My sincere thanks go to the contact person Barbara BINDER and all the ALICE FoCal Collaboration, and in particular the ALICE group from the University of Kansas, to let me do this wonderful experience here at CERN

7 References

[1] Kenneth Aamodt, A Abrahantes Quintana, R Achenbach, S Acounis, D Adamová, C Adler, M Aggarwal, F Agnese, G Aglieri Rinella, Z Ahammed, et al. The alice experiment at the cern lhc. *Journal of Instrumentation*, 3(08):S08002, 2008.

[2] O Brüning, H Burkhardt, and S Myers. The large hadron collider. *Progress in Particle and Nuclear Physics*, 67(3):705–734, 2012.

[3] ALICE collaboration et al. Letter of intent: a forward calorimeter (focal) in the alice experiment. Technical report, 2020.