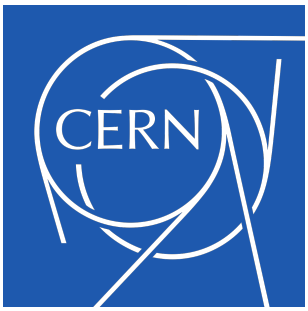


CERN

Summer Student Project Documentation

Design and Implementation of a Multi-Channel Oscilloscope Data Acquisition and Monitoring System

A Distributed Web-Based System for Remote Monitoring and Control of Laboratory Oscilloscopes for the LIS section

by Roman Michael Juhls

over the period of
02.06.2025 - 08.08.2025

Supervisor
Giulia Gnemmi

Table of Contents

1 Introduction **2**

2 Approach **2**

3 Implementation **3**
3.1 Frontend - Streamlit Dashboard 3
3.2 Backend - FastAPI Service 4
3.3 Database - PostgreSQL 4
3.4 Oscilloscope Communication 4
3.5 Overall Integration 4

4 Conclusion and Outlook **5**
4.1 Outlook 5

1 Introduction

The LIS (LINAC, Injectors and Synchrotron) section of the Radio Frequency group at CERN depends heavily on oscilloscopes to retrieve real-time voltage and current data from different parts of the accelerator systems. These oscilloscopes are used for analysis, monitoring, and control purposes. However, the lack of centralized access and data storage often leads to inefficiencies in workflow, data management, and collaboration. Each oscilloscope typically operates as a standalone device, requiring manual interaction for data retrieval, analysis, and saving, which can be time-consuming and error-prone. Even though most oscilloscopes have built-in web dashboards, the control possibilities on these dashboards are limited with no way to save data for future analysis. Additionally, with multiple oscilloscopes, it is not convenient to open each dashboard individually and type the designated IP addresses to access them.

This project aims to develop a unified web-based dashboard that connects to all oscilloscopes within the section, enabling centralized remote access, real-time waveform monitoring, and automated data retrieval. By integrating these instruments into a single system, users can conveniently interact with multiple devices from any authorized workstation. Additionally, the system allows waveforms and measurement data to be saved in a centralized database, making historical data accessible for review, comparison, and analysis over time.

The solution not only improves operational efficiency and user experience but also lays the groundwork for advanced data analytics, collaborative diagnostics, and long-term signal tracking—all essential in modern test environments.

2 Approach

The development of the system followed a modular and iterative approach to ensure flexibility, scalability, and ease of maintenance. The project was structured into distinct phases, encompassing planning, implementation, and integration of the various components: the frontend, the backend, and the database.

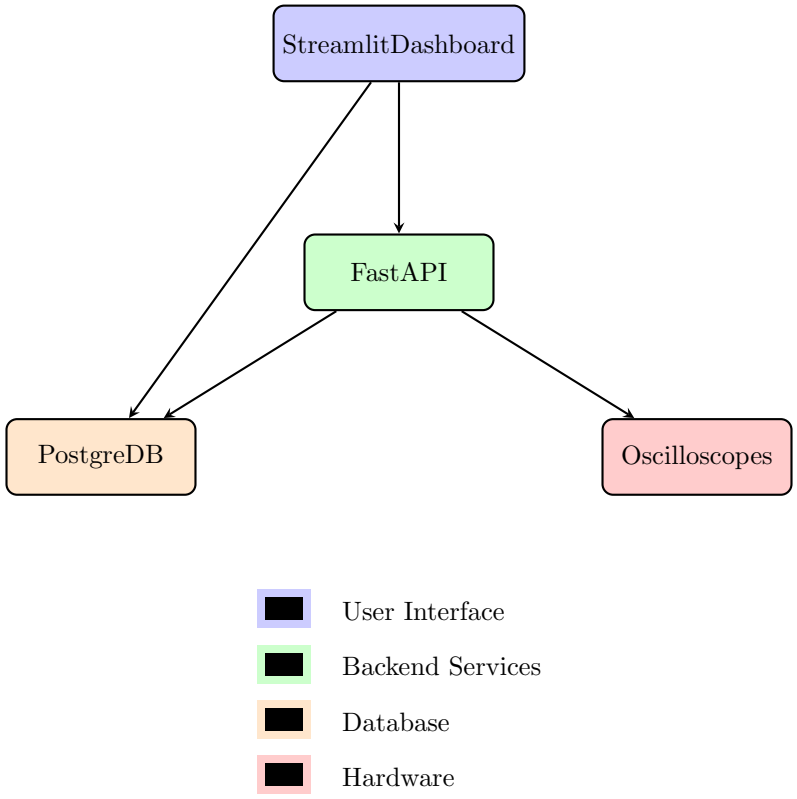
The first step involved analyzing the requirements. The key objectives included the ability to interact with multiple oscilloscopes remotely, retrieve real-time waveform data, perform automated data acquisition over defined time intervals, and access historical measurement data through a centralized system. Based on these requirements, an architecture was designed comprising a web-based user interface, a backend API service, and a relational database.

The backend was implemented using **FastAPI**, which handles the communication with the oscilloscopes. It exposes endpoints for controlling data acquisition processes and provides interfaces for retrieving both real-time and historical data. To support automated measurements over time, background tasks were integrated using asynchronous routines, ensuring reliable and efficient operation.

The frontend was developed using **Streamlit**, providing a simple yet interactive web application for users to connect to different oscilloscopes, visualize live waveform data, and initiate or stop data acquisition. The dashboard also allows users to explore historical data stored in the database, offering a seamless experience without direct interaction with the backend logic.

Data storage was managed using a **PostgreSQL** database, chosen for its reliability and support for structured data. The schema was designed to store waveform data along with relevant metadata such as timestamps and device identifiers, enabling efficient querying and long-term archiving. A background service ensures that data is removed after a certain period to prevent storage overflow. The following diagram explains the main structure and interactions between the different subsystems.

System Architecture



3 Implementation

The system was implemented using a modular architecture that separates the user interface, backend logic, and data storage components. This separation ensures maintainability, scalability, and clear responsibilities within the codebase.

3.1 Frontend - Streamlit Dashboard

The frontend was developed using **Streamlit**, a Python-based framework for building interactive web applications. It provides a simple and responsive dashboard that allows users to:

- Connect to available oscilloscopes
- Visualize live waveform data in real-time
- Start and stop automated data acquisition sessions
- Load and filter historical measurement data from the database

Communication between the Streamlit frontend and the backend is handled via RESTful API endpoints. For data visualization, the system uses `matplotlib` and `plotly`, offering flexibility in how signals are displayed.

3.2 Backend - FastAPI Service

The backend service was implemented with **FastAPI**, a modern asynchronous web framework for building high-performance APIs. It serves as the communication layer between the frontend, the measurement devices, and the database. Core responsibilities include:

- Managing connections to multiple oscilloscopes via PyVISA
- Periodically retrieving real-time waveform data from devices
- Storing measurement data in the PostgreSQL database
- Providing API endpoints for frontend interaction

Background data acquisition is handled using FastAPI's asynchronous task features (`BackgroundTasks` and `asyncio`), ensuring uninterrupted data flow even without active user input.

3.3 Database - PostgreSQL

For centralized and reliable storage of measurement data, the system uses a **PostgreSQL** relational database. It stores:

- Raw waveform data (e.g., voltage vs. time)
- Device metadata (e.g., serial number, channel settings)
- Timestamps and acquisition context

The schema was designed to support both high-frequency data insertions and efficient querying for visualization and analysis.

3.4 Oscilloscope Communication

Oscilloscopes are accessed using standard command protocols such as **SCPI** (Standard Commands for Programmable Instruments), typically via USB, LAN, or GPIB. The system uses the `PyVISA` library for device communication, allowing for:

- Device discovery on the local network
- Querying of current waveform data
- Configuration of measurement settings (channels, time base, triggers)

3.5 Overall Integration

All components are designed to work together seamlessly:

- The frontend serves as the control and visualization layer
- The backend handles business logic and orchestrates data flow
- The database persists historical measurements for long-term analysis
- Oscilloscopes act as data sources accessible in real-time

This architecture enables users to remotely monitor, acquire, and review data from multiple oscilloscopes in a unified and user-friendly environment.

4 Conclusion and Outlook

The unified oscilloscope dashboard developed in this project successfully demonstrates the integration of multiple test instruments into a centralized web-based system. By combining a Streamlit frontend, a FastAPI backend, and a PostgreSQL database, the solution allows users to interact with different oscilloscopes, visualize live waveforms, perform automated data acquisition, and review historical measurements in a convenient and efficient manner.

The modular architecture proved to be flexible and scalable, making it possible to support various communication interfaces such as SCPI over LAN or USB. The backend services are capable of handling real-time data acquisition asynchronously, ensuring stable operation without blocking the user interface. The frontend provides a simple and user-friendly interface that abstracts technical complexity from the end user, enabling seamless interaction with devices and data.

Through this system, the process of waveform analysis becomes more consistent and accessible across the entire section. Storing historical data in a structured format opens the door for long-term monitoring, comparison of measurements, and possibly even automated anomaly detection in future applications.

4.1 Outlook

While the current implementation fulfills the core goals, several improvements and extensions can be considered in the future:

- **Device Management:** Adding support for dynamic registration and configuration of devices in the dashboard
- **User Authentication:** Implementing access control to manage user permissions and data visibility
- **Data Export and Reports:** Enabling export of waveform data as CSV, PDF, or integration into reporting pipelines
- **Advanced Analytics:** Applying signal processing or machine learning algorithms to detect patterns, trends, or anomalies in historical data
- **Containerization and Deployment:** Packaging the system into containers for easier deployment and scalability across different labs or sections
- **Long-Term Stability and Maintenance:** Evaluating long-term system behavior remains an open task due to time constraints. Potential bugs or performance issues that may emerge over prolonged use will need to be addressed in future iterations

In conclusion, the developed system forms a robust foundation for a modern, remote-access oscilloscope management platform. It improves usability, data consistency, and workflow efficiency, while offering significant potential for further development and integration in advanced testing environments.