JOINT INSTITUTE FOR NUCLEAR RESEARCH
Dzhelepov Laboratory of Nuclear Problems

FINAL REPORT ON THE START PROGRAMME

“ Software for data acquisition from Rigol digital oscilloscopes ”

Supervisor:
Mr. Aleksei Trifonov

Student:
Wasi Ur Rehman Qamar, Pakistan
Nazarbayev University

Participation period:
June 30 – August 10,
Summer Session 2024

Dubna, 2024

Table of Contents

Abstract

The software developed for data acquisition from Rigol (e.g. DS1000Z series) digital oscilloscopes is a comprehensive solution tailored for DOS / Linux systems. Leveraging Django, Flask, and React, the application facilitates seamless interaction with oscilloscopes connected via Ethernet, each with a unique IP address on a local network. The system features three main modules: an initialization module for oscilloscope configuration through IP address input; a visualization module that generates individual sections for each oscilloscope, displaying oscillograms data from four channels with toggling options for each channel; and a data archiving module that integrates Timescale DB for efficient data storage, accompanied by interfaces for starting/stopping archiving and visualizing archived data. This software ensures efficient data acquisition, real-time visualization, and robust data management, enhancing the usability and functionality of Rigol DS1000Z series oscilloscopes.

Introduction

The linear accelerator Linac-200 at the Joint Institute for Nuclear Research (JINR) is a newly constructed facility designed to provide electron test beams for particle detectors research and development, studies of advanced electron beam diagnostics methods, and applied research. This facility represents a significant upgrade in terms of both technology and capability. The core of the Linac-200 is a refurbished MEA accelerator from NIKHEF, with key subsystems such as controls, vacuum, and precise temperature regulation having been completely redesigned or deeply modernized [1].

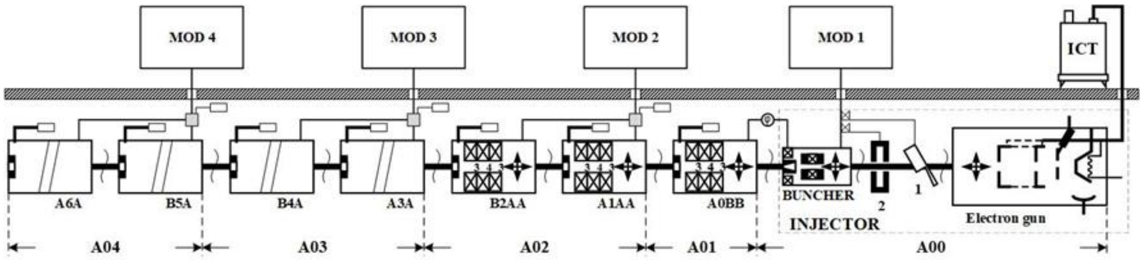

Figure 1 : Linac-200 Accelerator Layout

The electron beam is generated by the 400-kV DC triode-type electron gun with a thermionic cathode. The beam is available for users at two extraction points: after stations A01 (EP1) and A04 (EP2) [1].

Figure 2: Beam extraction points at Linac-200 , Left – EP1 , Right – EP2

Rigol Technologies is a leading manufacturer of electronic test and measurement equipment, renowned for its high-quality oscilloscopes, spectrum analyzers, and waveform generators. The Rigol DS1000Z series digital oscilloscopes, particularly the DS1054Z model, are highly regarded for their advanced features, affordability, and reliability. These oscilloscopes are essential tools in electronics research and development, troubleshooting, and educational environments, offering detailed insights into electronic signals through their multi-channel configuration.

Oscilloscopes work by capturing and displaying electronic signals, allowing users to observe signal waveforms and analyze their properties such as amplitude, frequency, and noise. The Rigol DS1000Z series features a four-channel input, enabling simultaneous observation of multiple signals, which is crucial for complex signal analysis. These oscilloscopes use a command system, as detailed in their documentation, to facilitate data exchange and remote control, making them versatile tools in various scientific and engineering applications.

To fully harness the capabilities of these oscilloscopes, a dedicated software solution is required. This software will run on Linux (Debian/Ubuntu) systems and use Ethernet connectivity for seamless communication with the oscilloscopes. It aims to enhance the usability of Rigol DS1000Z series oscilloscopes by providing a robust graphical interface for configuration, real-time data visualization, and efficient data archiving.

At this point, there are already implementations of similar software for data exchange with the DS1000Z series oscilloscopes. These software's are DS Remote (Linux) and Rigol DS1054 Viewer/Controller (Windows). However, these existing solutions have limitations in some features and may not have been thoroughly tested during development. Additionally, they primarily run on specific operating systems, limiting their usability.

Our developed system aims to address these shortcomings and to facilitate data acquisition and management from the Linac-200, a specialized software application that has been developed. This software is intended to enhance the capabilities of the Linac-200 by providing efficient data collection, real-time visualization, and robust data storage. The software is compatible with all major operating systems, including Windows and Linux making it versatile and accessible to a wide range of users.

Project Goal

The primary goal of this project is to develop a comprehensive software application that enables the collection, visualization, and storage of data from Rigol DS1000Z series digital oscilloscopes. The software will include three key functional modules:

Initialization Module: This module will provide a graphical interface for configuring the oscilloscope's connection. Users will be able to input the oscilloscope's IP address manually.

Visualization Module: After configuration, the software will generate separate sections for each connected oscilloscope, displaying wave signals from all four channels. Users will have the ability to toggle the display of each channel on and off using buttons.

Data Archiving Module: This module will facilitate the archiving of collected data into a database called Postgres SQL with Timescale DB extension. The visual interface will include controls to start and stop data archiving and provide visualization tools for the archived data.

By achieving these goals, the software will significantly enhance the capabilities of Rigol DS1000Z series oscilloscopes, making them more versatile and user-friendly for various applications. The project will draw on existing implementations such as DS Remote and the Rigol DS1054 Viewer/Controller as references to inform its development process.

System Implementation

The software for data acquisition from Rigol DS1000Z series digital oscilloscopes is implemented using a combination of modern web technologies and frameworks to ensure a robust, scalable, and user-friendly application. The system is divided into three main modules: control, visualization, and data archiving. Each module is designed to handle specific functionalities and is integrated to provide a seamless experience.

1. Frontend Implementation

The frontend is developed using the React framework, utilizing HTML, CSS, JavaScript, and Bootstrap to create an intuitive and responsive user interface. React component-based architecture allows for efficient management of UI components and dynamic updates.

- **Control Module**

  This module communicates with the oscilloscope to control various settings, including turning specific channels on and off, adjusting horizontal and vertical scale values, and capturing screenshots of the oscilloscope display. The control module sends HTTP requests to the backend Django server to perform these actions and updates the UI based on the oscilloscope's response.

- **Visualization Module**

  A separate visualization module is developed using Flask for the frontend and Python for the backend. This module fetches waveform data from the oscilloscope and displays it using Plotly graphs.

2. Backend Implementation

Flask is used to handle socket connections for real-time data fetching, providing a reliable two-way communication channel between the oscilloscope and the application. The rationale behind using socket connections is to ensure fast and reliable real-time data transfer, as sockets enable continuous two-way communication, unlike HTTP, which waits for a request before sending a response.

- **Django**

  The backend for the control module is implemented using Django, which handles communication with the oscilloscope via Ethernet. It processes incoming HTTP requests from the React frontend to control the oscilloscope settings and returns the updated status.

- **Flask**

  For real-time data visualization, a separate module is developed using Flask and Python. This module establishes socket connections with the oscilloscope to fetch live waveform data. The use of sockets ensures quick data transfer and updates, enhancing the reliability and responsiveness of the visualization module.

3. Data Archiving Module

A dedicated data archiving module is developed using Flask and Python, with PostgreSQL and Timescale DB for data storage. This module provides functionalities to save oscilloscope data to the database and retrieve it based on specified IP addresses, start times, and end times.

- **Data Storage**

  The archiving module saves waveform data to Timescale DB, a time-series database extension for PostgreSQL, ensuring efficient storage and retrieval of large volumes of time-series data.

- **Data Retrieval**

  Users can query the archived data by specifying the oscilloscope's IP address and a time range. The module fetches the relevant data from the database and displays it using the developed visualization framework.

4. Integration

The control and visualization modules are integrated into the React frontend to provide a unified interface for users. This integration allows users to control the oscilloscope settings and view real-time waveform data from a single application, enhancing usability and efficiency.

Results

1. Visualization Module

Figure 1 shows the visualization module before connecting. You can connect to an oscilloscope using IP address or by uploading a .csv file. Figure 2 shows the visualization module after connecting to oscilloscopes. This module usually opens automatically after opening bash file and running on port 3000.

In the visualization module you can perform the following functions:

- ✓ Turn On or Off a specific channel
- ✓ Start or Stop data archiving
- ✓ Change vertical or horizontal scale
- ✓ Capture the screenshot of oscilloscope from device
- ✓ Connect oscilloscope thorough IP by entering manually or by uploading CSV file

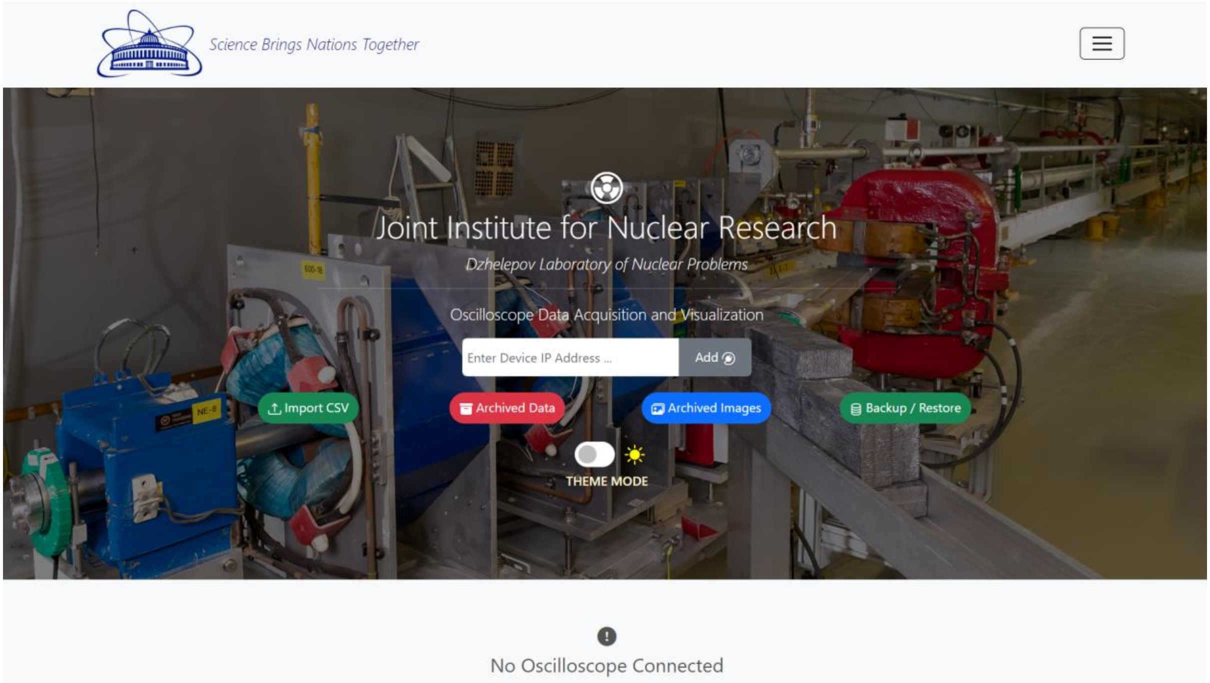

Figure 3 : Before Connection to Oscilloscopes

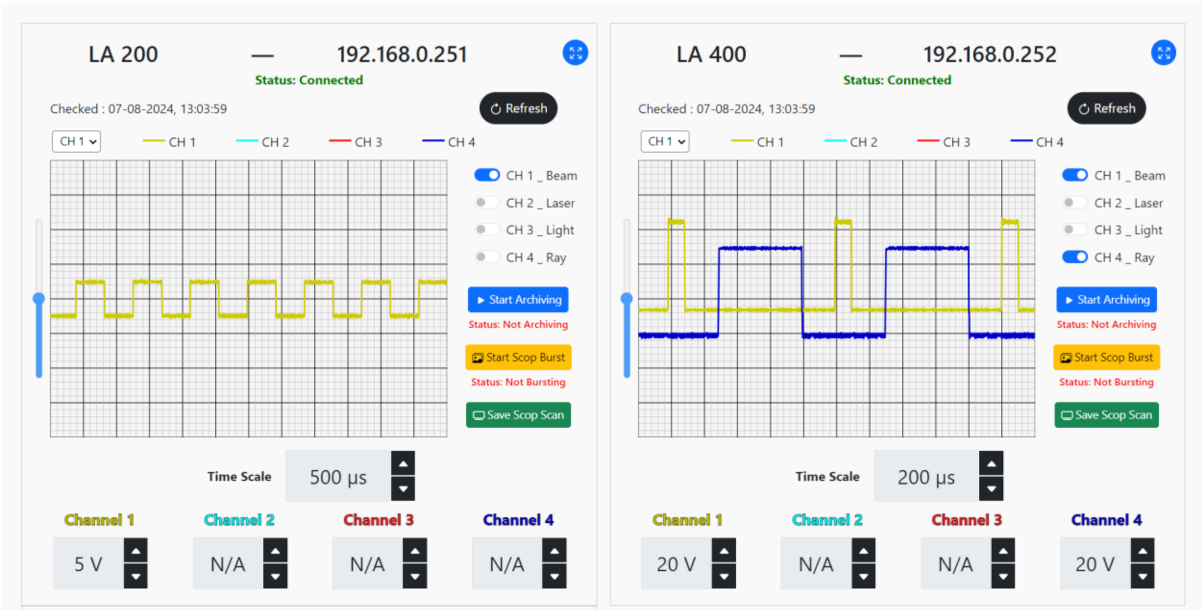

Figure 4 : After Connecting to Oscilloscopes

2. Archive Module

Figure 3 shows the archive module before checking the data. You need to enter the IP address of an oscilloscope for which you want to see the data. You have the option to apply filter like seeing all the channel of only a specific channel and use the start and end time to see the waveform data of only a specific time. It usually run on port 5000. You can access this page by clicking the button on visualization module. This archive module has two functionalities. We can see archived data in original form having noise and can view smoothed data by applying smoothing and down sampling.

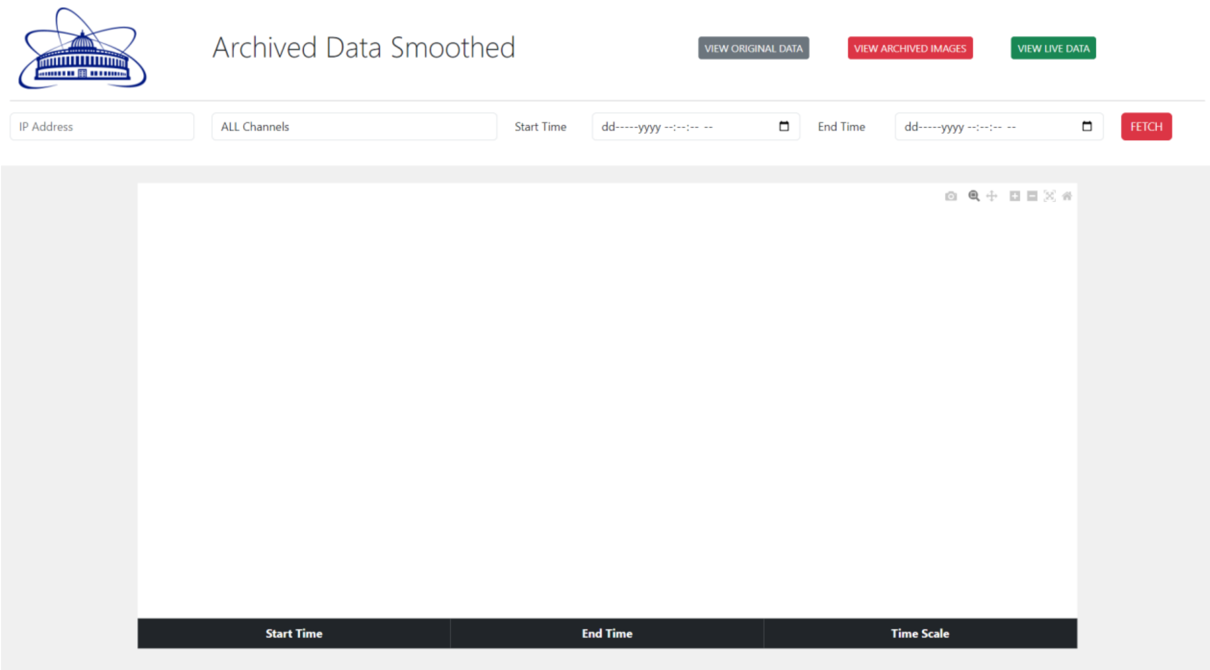

Figure 5 : Archived Module Before Fetching Data

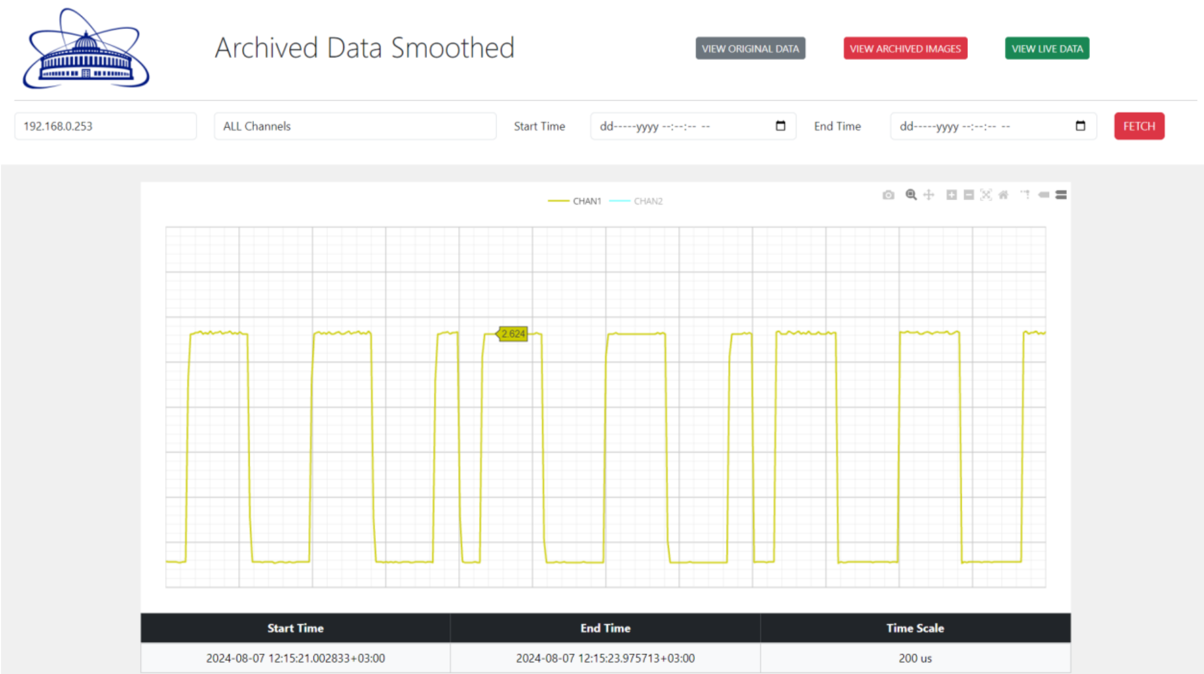

Figure 6 : Smoothed Archive Module After Fetching Data

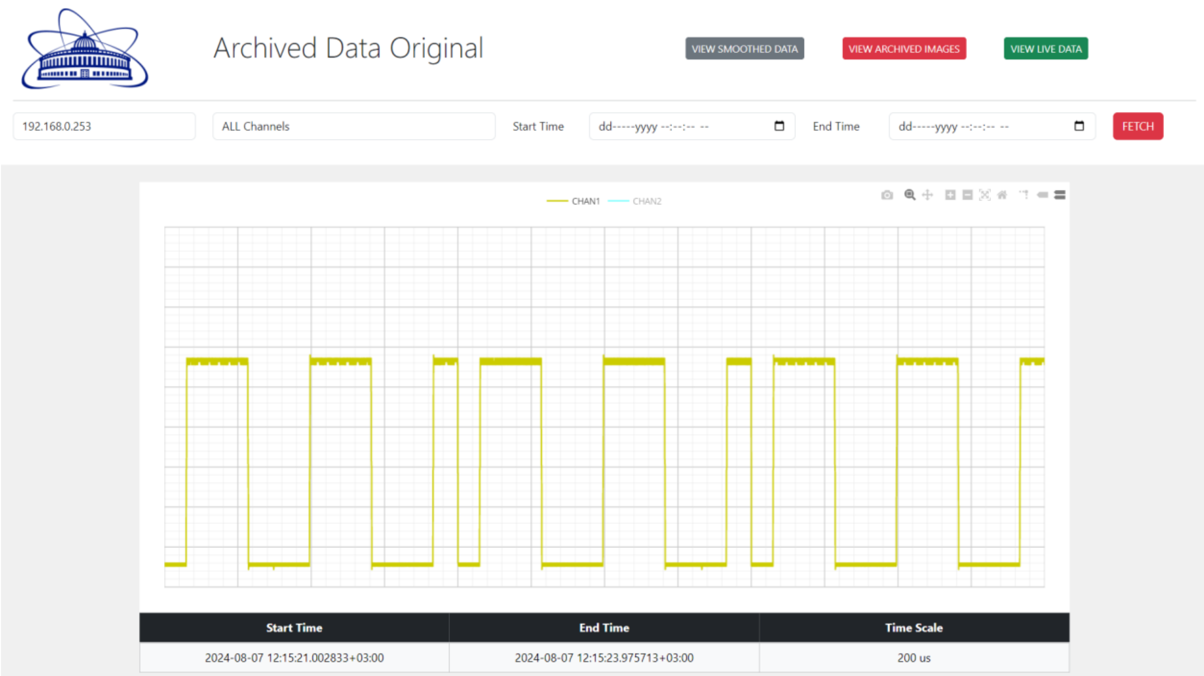

Figure 7 : Original Archive Module After Fetching Data

Additional Features

In addition to visualization and archiving module, we can view the images saved by our software on computer using the integrated module of Archived Images. It will open the directory where all the images are saved on the computer. In addition, we can create a backup and restore our database using a single button click.

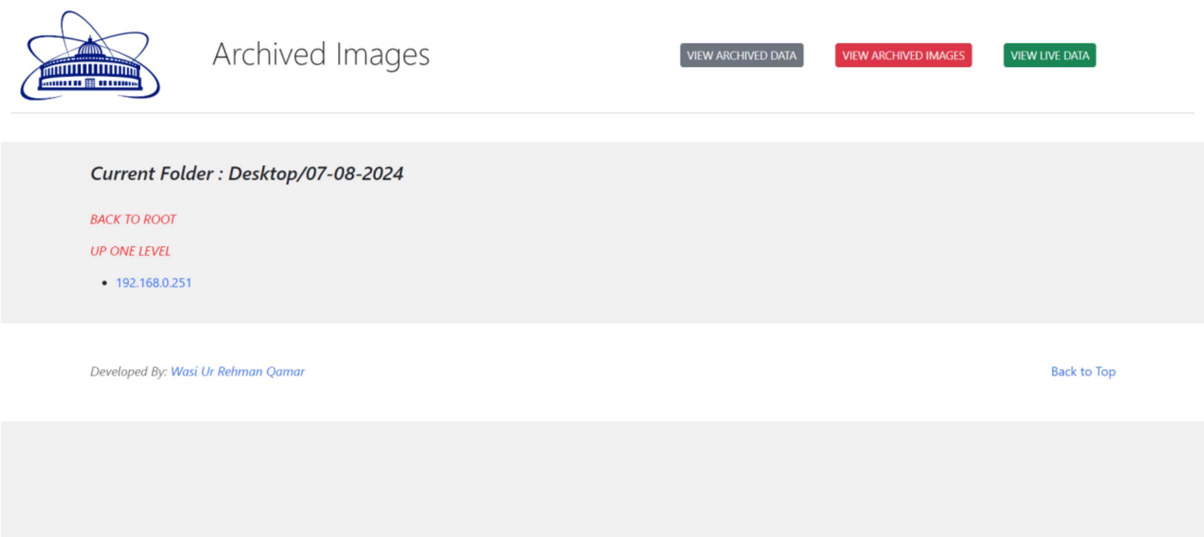

Figure 8 : Viewing Archived Images Web Interface

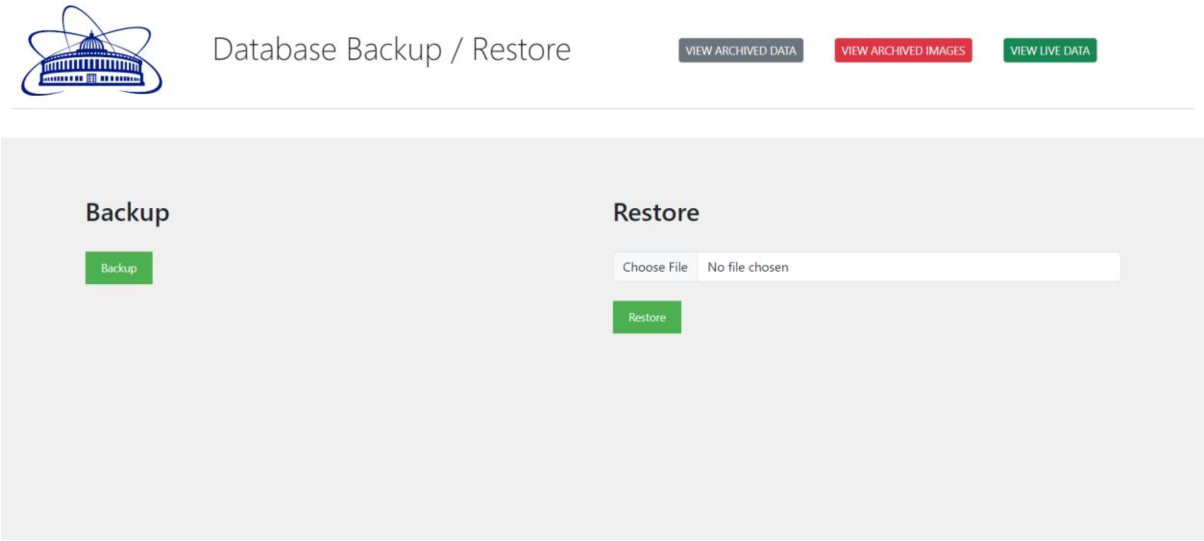

Figure 9 : Backup / Restore Web Interface

Conclusion

The developed software provides a comprehensive solution for data acquisition, visualization, and storage from Rigol DS1000Z series digital oscilloscopes. By using modern web technologies such as React, Django, Flask, and Plotly, the system offers a robust and user-friendly interface that enhances the functionality and usability of the oscilloscopes.

The integration of HTTP and socket communication ensures reliable and fast data exchange, while the use of PostgreSQL and Timescale DB for data storage provides efficient management of large volumes of time-series data. The software's compatibility with all major operating systems, including Windows, Linux, and macOS, makes it versatile and accessible to a wide range of users.

By addressing the limitations of existing implementations and providing additional features such as real-time visualization and data archiving, this software significantly improves the capabilities of Rigol DS1000Z series oscilloscopes. The intuitive graphical interface simplifies the process of controlling the oscilloscope and viewing the data, making it a valuable tool for researchers, engineers, and educators.

In conclusion, this project successfully delivers a powerful and flexible software solution that meets the needs of users working with Rigol DS1000Z series digital oscilloscopes, fostering enhanced productivity and insight in various scientific and engineering applications.

References

[1] Linac-200: A New Electron Test Beam Facility
https://pos.sissa.it/414/1094/pdf

[2] RIGOL Programming Guide DS1000Z-E Series Digital Oscilloscope
https://tw.rigol.com/tw/Images/DS1000ZEProgrammingGuideEN_tcm17-2850.pdf

[3] Oscilloscope Introduction
https://www.tek.com/en/blog/what-is-an-oscilloscope

[4] RIGOL Oscilloscope Products Details
https://www.rigolna.com/products/digital-oscilloscopes/1000z/

[5] Flask User’s Guide
https://flask.palletsprojects.com/en/3.0.x/

[6] Django documentation, everything you need to know about Django.
https://docs.djangoproject.com/en/5.0/

[7] React Overview, Creating a new project
https://react.dev/learn/start-a-new-react-project

[8] Timescale DB Installation Guide
https://docs.timescale.com/self-hosted/latest/install/

[9] RIGOL Programming Guide MSO1000Z/DS1000Z Series Digital Oscilloscope
https://www.batronix.com/files/Rigol/Oszilloskope/_DS&MSO1000Z/MSO_DS1000Z_ProgrammingGuide_EN.pdf

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Mr. Aleksei Trifonov of the Joint Institute for Nuclear Research, for his invaluable guidance, support, and encouragement throughout this project. His expertise and insights were instrumental in the successful development of the software.

I am also deeply thankful to the Joint Institute for Nuclear Research for providing me with the opportunity to participate in the summer session of the START program. This experience has been incredibly enriching and has significantly contributed to my professional and personal growth.

Special thanks to my brother, Dr. Safi Ur Rehman Qamar, for his unwavering motivation and encouragement, which pushed me to participate in this program. His support was crucial in helping me seize this valuable opportunity.

Thank you to everyone who supported and encouraged me throughout this journey.