OSCILLOSCOPE DATA PUSH PROGRAM

Jason Osei-Tutu*
Advisors: Jose Berlioz[1], Aisha Ibrahim[1], Carol J. Johnstone[1]
Fermi National Accelerator Laboratory, Batavia, IL, USA†

ABSTRACT

This paper details the development of a Python program designed to automate the data acquisition and conversion for an oscilloscope for the purposes of a one-off/temporary data acquisition system for users that readily need data, and do not have the option of obtaining a Data Acquisition (DAQ) solution. Creating DAQ systems for analyzing a system requires expensive electronics and a dedicated team of engineers for support. Traditionally, manual data collection and processing are time consuming and prone to error. By automating these processes, the cost, efficiency and accuracy of data handling are improved upon. This project involves the creation of a program that interacts with the oscilloscope. During this interaction, there are various functions being performed such as the acquisition of waveform data via floating points, generating plots with the acquired wave points, and storing of floating points in a CSV file format for future reference and plotting purposes. While the initial aim of the project included continuous logging to a cloud database, this was deferred due to time constraints. The results portrayed an almost-instant rate of data collection with a buffer time, showcasing the potential for further integration and real-time data processing.

INTRODUCTION

Data acquisition (DAQ) is a complex and costly process. Creating DAQ systems for analyzing a system requires expensive electronics and a dedicated team of engineers for support, posing a challenge for users who readily need data, such as scientists in the study of the nature of particles using a particle accelerator. Particle accelerators are devices that propel charged particles such as protons or electrons, to high speeds aimed at various targets. These collisions are studied to understand fundamental particles and forces. In these studies, oscilloscopes are used for capturing and analyzing the fast-changing electrical signals generated during particle collisions.

This project aims to serve as a proof of concept for creating a temporary or "one-off" DAQ system using the Rohde & Schwarz oscilloscope which commonly available to every team. This oscilloscope was chosen because of its gigahertz capability, providing a better resolution of signal display. We aim to automate the data acquisition process from the oscilloscope via the automatic conversion of binary to floating points predetermined by the oscilloscope, and store the results in a CSV file format.

By developing a Python program to handle these tasks, we seek to reduce the manual effort involved in data collection, significantly increasing efficiency, and decreasing time and cost. This automation is important in the context of particle accelerators, where rapid data analysis can lead to timely insights and more efficient experimentation, ultimately advancing our understanding of fundamental physics.

METHODS

Equipment

The primary equipment used in this project includes:

- Rohde & Schwarz RTO 1044 Oscilloscope with Windows OS.
- A computer connected via Ethernet or USB
- A function generator for simulating wave functions

Programming Language and Libraries

Python was chosen for its versatility and extensive library support. Most oscilloscopes use the PyVISA library to communicate with measurement instruments. However, in this project, we used the RsInstrument[1] library provided by Rohde & Schwarz. This library includes specific Standard Commands for Programmable Instruments (SCPI) that are unique to Rohde & Schwarz oscilloscopes, enabling more precise and tailored control over the data acquisition process. SCPI commands are ASCII strings, which are sent to measuring instrument (such as oscilloscopes and signal generators) over the physical communication layer.[2]

Development Process

The development process involved creating a Python class to manage data acquisition and conversion. The main functions implemented in the class are:

- **Initialization**: Establishing a connection with the oscilloscope using RsInstrument.
- **Trigger Settings**: Configuring trigger options to control when data acquisition occurs.
- **Data Acquisition**: Retrieving waveform data from specified channels.
- **Data Storage**: Saving the acquired data to a CSV file.
- **Plotting**: Generating plots of the acquired waveforms for visualization.

* First author
† Operated by Fermi Research Alliance, LLC.

Process Workflow

Due to the lack of access to a live beam, multiple wave functions were simulated using a function generator during the course of building the program. This approach ensured that the program could still be tested and validated under conditions that approximate real-world usage, despite the absence of live beam data.

Automated Data Acquisition: With the the development of the program, the data acquisition process is simplified:

- The Python program automatically acquires data from the oscilloscope via Ethernet or USB.
- Data is acquired as floating points.
- Floating points are saved directly to a user-specified CSV file.

Figure 1 shows the complete setup of the designated PC connected to the scope, receiving wave input from a function generator.

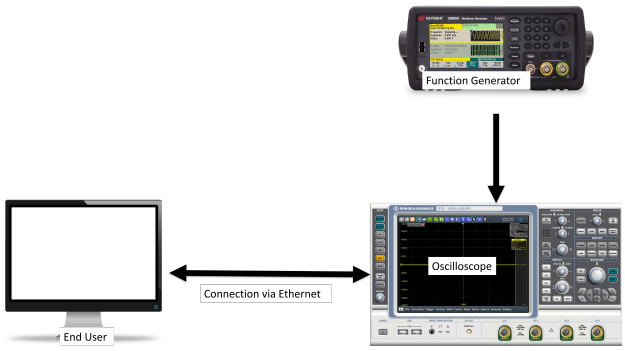

Figure 1: Complete Connection of Devices

RESULTS AND DISCUSSION

The program demonstrated the ability to successfully connect to and interact with a Rohde & Schwarz RTO oscilloscope using the `RsInstrument` Python library. The following results show the key functionalities and performance outcomes of the developed system.

Scope Identification and Connection

The program was able to establish a stable connection with the oscilloscope via a TCP/IP interface. We verified this connection by querying the identification of the oscilloscope: Oscilloscope Identification: Rohde&Schwarz,RTO,1316.1000k44/400199,3.70.1.0

Data Acquisition and Configuration

The oscilloscope was configured for single acquisition to a specified channel, with a time range of 50 milliseconds, an acquisition point count of 1,000,000 (The maximum number of points the scope allowed), and specific triggering functions. These settings aimed to achieve high-resolution data capture over the specified time period. The acquisition settings were applied as follows:

```
ACQ:POIN:AUTO RECL
CHAN1:STAT ON
CHAN1:RANG 2
CHAN1:POS 0
CHAN1:COUP AC
TIM:RANG 0.05
ACQ:POIN 1000000
TRIG:MODE AUTO
TRIG:SOUR CHAN1
TRIG:LEV 0.04
```

We later included an "autoscale" function for testing purposes. The autoscale[1] function was used to automatically adjust the horizontal, vertical, and trigger settings, ensuring stable waveform display. Below is the SCPI command for the autoscaling feature:

```
AUToscale
```

Waveform Data Retrieval and Visualization

After completing the acquisition, the waveform data from Channel 1 was retrieved in binary format. The program successfully captured the waveform data points, and the total number of acquired points was consistent with the configured settings:

```
Maximum acquisition points supported: 1000000
```

The acquired data was then visualized using Matplotlib. The generated plot displayed the waveform over the 50 ms acquisition time. The plot produced a clear and stable waveform, indicating the effectiveness of the autoscale and acquisition configuration. The waveform was plotted in red for a better visibility. Figure 2 shows the plotted waveform obtained by the oscilloscope:

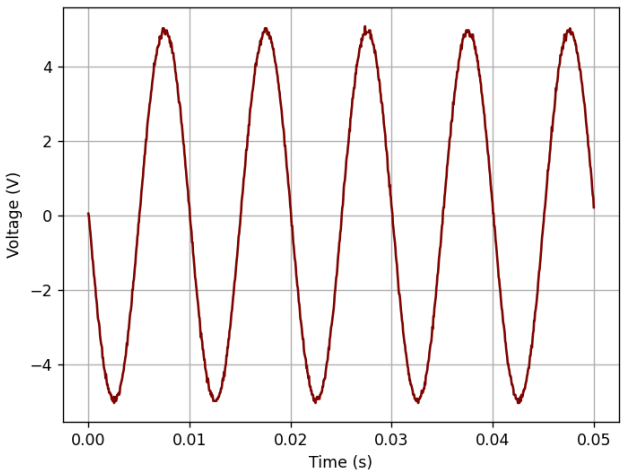

Figure 2: Plotted Sine Wave function

[1] The autoscale fucntion can be removed if user has premade display settings

Data Storage

The acquired waveform data, along with the corresponding time values, was successfully saved to a CSV file with a specified path of the user's choice. This function makes sure that the captured data can easily be accessed and analyzed further. Below is an example of data stored in the file:

Time (s)	Voltage (V)
0.0	-0.002
0.00005	0.003
...	...
0.04995	0.001
0.05	-0.002

Table 1: Example csv file with wave data

Discussion

The developed oscilloscope control program effectively demonstrated the ability to configure, acquire, visualize, and store waveform data from a Rohde & Schwarz RTO oscilloscope. The integration of the `RsInstrument` library facilitated seamless communication with the oscilloscope, and the implementation of key functions ensured robust data acquisition and handling. The results indicate that the program can serve as a reliable tool for oscilloscope data management in various research and engineering applications.

The results demonstrate that automating data acquisition and conversion for the R&S RTO 1044 oscilloscope significantly improves efficiency and accuracy. The ability to precisely convert binary data to floating points and automate the storage process reduces the potential for human error and speeds up data handling.

While the project did not achieve continuous cloud-based data logging due to time constraints, the current implementation provides a strong foundation for future enhancements. Integrating cloud storage and real-time processing capabilities will further streamline workflow and enhance data accessibility and collaboration.

The use of a function generator to simulate wave functions provided a live testing environment, ensuring the program's functionality in the absence of an active beam. This approach allowed for thorough validation of the Python class, demonstrating its effectiveness and reliability.

FUTURE WORK

In the future, we plan to enhance the program's functionality by allowing it to collect data from multiple channels of the oscilloscope simultaneously. This will help users capture and analyze signals from multiple sources at the same time, providing a more complete picture of the system being tested.

We also aim to build a cloud database where the data is automatically stored whenever a signal is detected by the oscilloscope. This will make it easier for users to access the data from anywhere, without needing to connect directly to the PC that is linked to the oscilloscope. The cloud-based storage will also facilitate data sharing and collaboration among users, making the program even more useful.

CONCLUSION

In conclusion, the oscilloscope control program simplifies the process of configuring and acquiring data from a Rohde & Schwarz RTO oscilloscope. It can establish a reliable connection, set up acquisition settings, retrieve and visualize waveform data, convert and save the data to a CSV file. This makes it a valuable tool for researchers and engineers.

The successful implementation of the program lays a strong foundation for future improvements, such as multi-channel data acquisition and cloud-based data storage. These enhancements will make the program even more powerful and versatile for oscilloscope data acquisition and analysis. Overall, the project has met its primary objectives and opened up new possibilities for future development and research.

ACKNOWLEDGEMENTS

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.
This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internship (CCI).

REFERENCES

[1] `https://rsinstrument.readthedocs.io/en/latest/RsInstrument.html`
[2] `https://www.rohde-schwarz.com/us/driver-pages/remote-control/remote-programming-environments_231250.html`