Implementation of a Web Interface to Display Real-Time Statistics of a Dilution Refrigerator

Hammad Rasheed
Fermi National Accelerator Laboratory - CCI Internship
University of Illinois at Urbana-Champaign
August 11, 2023

Abstract

The initial purpose of the project was centered around the communication with a Bluefors Control Unit. The unit is a piece of experimental equipment that is affixed to a dilution refrigerator. The connection allows for the control unit to access and temporarily store data at multiple different points in the refrigerator. Additional functionality also comes with the control unit, as it has its own API and a Bluefors control software user manual with instructions on how to communicate with the API. After understanding the API, a program was created in Python to connect to the Bluefors control unit via connecting to its websocket server using an API key generated from the control unit. Furthermore, communication was established by sending a JSON request to the server which asks for specific data, then retrieving the JSON objects that the server sends in response which contains the data that was requested. After successfully communicating with the control unit in a Python program, the next goal was to display the data in a webpage. In the first instance, HTML and CSS were used to create a webpage to display the data, however it became more advantageous to switch to using Next.js, React, NodeJS, and JavaScript to create a webpage. With these tools, a webpage was created that can access the data of multiple dilution refrigerators by accessing the Python code from the web interface.

Introduction

The basis of the project was to find a way to create a webpage that can display the data of multiple different dilution refrigerators dynamically. Before the introduction of this project, it was necessary to learn about the tools involved to make the project possible to accomplish. The first tool that was prioritized was understanding the mechanics of the programming language Python. Python is a very high-level general purpose programming language that has many functionalities. With a basic understanding of Python, the next goal was to understand the websockets library. Websockets is a library that is used to build WebSocket servers and clients in Python, which serves to be invaluable in communicating between computers. After understanding the library by reading the documentation provided by the creators, a program was created to act as a WebSocket client and another was created to act as a WebSocket server. These programs were both written in Python and were able to communicate between two different computers. It was also necessary to learn the json and asyncio libraries, the json library allows for handling JSON objects and the asyncio library allows for an indefinite runtime for a program, which makes it easier for a program to run as a server. The final components that were necessary to succeed were the components required for the web page. Initially, HTML and CSS seemed to be the most reliable tools for this, however it was realized that using Next.js, REACT, NodeJS, and JavaScript would be much more advantageous and practical. Therefore, after understanding all of these tools, it was possible to start the project.

Bluefors Control Unit

After understanding the basics and the practicality of the Python language and the websockets library, its relevance was explained, for the main component of the project was introduced. The method in which the data from the dilution refrigerators, also known as DRs, would be extracted is through a piece of experimental equipment called a Bluefors control unit. The Bluefors control unit is integrated on each of the DRs and is configured to access and temporarily store the data at multiple different points on each DR. The control unit also has a very thorough application programming interface, also known as API, that provides functionality to access the data of each DR. The API is complex and requires an understanding of the Bluefors control software user manual. The manual explains that the Bluefors control unit hosts a websocket server that can be accessed with an API key that can be generated directly from the Bluefors unit itself. With the generated API key, where each key has its own permissions to read and write data, the websocket server of the control unit can be accessed from a program and communication can be established between the control unit and the program via JSON requests and JSON objects. It must also be added that in the Python program, if there is no valid SSL certificate, then added functionality has to be added to the program to ignore SSl verification to establish a connection to the control unit.

JSON Communication

With the API key and the name of the computer with the Bluefors control unit attached, a WebSocket client can be created in Python to access the Bluefors unit. Once a connection is established, the WebSocket client has to send a JavaScript Object Notation, also known as JSON, request to the control unit to specifically ask for which data it intends to access. The format of the JSON request is specifically described in the user manual and must be followed coherently. After a JSON request is sent to the control unit, it will respond with two separate JSON objects. The first JSON object will have the status “RECEIVED” and it will be a confirmation of the request that was sent to the control unit with the same information that was provided, and the

second JSON object will have the status “SUCCEEDED”. This object will contain the data that was requested and can be parsed in Python to access the data specifically.

Backend Development

After understanding the necessary steps to access the Bluefors control unit and communicate with the unit to access the data which was expected to be displayed on the web page, a program was created in Python that accomplished these steps. The Python program first connects to the WebSocket server hosted by the Bluefors control unit, while ignoring the SSL verification requirement. Then, the program sends a JSON request to the server and retrieves both of the JSON objects that are sent in response. The status of the object will be checked and it will wait until the object with the status, “SUCCEEDED”, is received. Once this object is received, the program will host its own WebSocket server and send the object to that server. The program will continue to repeat these operations over a set interval of time, which is called polling. This is not the most optimal solution, however it was necessary for the data to be updated in real time.

Webpage Development

After successfully accomplishing the goal of retrieving the data of the DR from the Bluefors control unit in the Python program. The next step was to find a method in which the data can be streamed to a webpage. The first instance of a webpage was created using HTML and CSS. This was done by creating an HTML file and formatting the webpage in CSS, in this file, a JavaScript file was called that accessed the websocket server and displayed the data from the Bluefors control unit. After this version of the webpage succeeded, it was switched to a different form of webpage by creating a web interface using Next.js, REACT, NodeJS, and JavaScript. Using Next.js is much more efficient because it is a web development framework that supports REACT, which is a JavaScript library that can be used to create web applications. It also is much more coherent than HTML and CSS because it allows for a local development environment to be created, which allows for much faster testing. It also uses server-side rendering which makes the webpage load faster and is much more practical in terms of functionality and user interface.

Troubleshooting

In the development process of the webpage, there were many difficulties that were necessary to overcome. The first problem that was encountered was with the websockets library. Programs that were developed that tried to create a WebSocket client or server would not work at all. This problem was rectified by configuring the computer’s firewall to ignore insecure activity on specific ports to allow for the WebSocket connections to go through. There were also many future problems regarding security due to the lack of an SSL certificate. A SSL certificate is a digital certificate that authenticates a website’s identity. Without using this certificate, it was necessary to find a way to ignore this authentication measure in the connections that were made. This was able to be done in Python using lines of code that specifically rejected the SSL confirmations. However, this could not be done in JavaScript through the webpage. Due to this problem, it was necessary for the Python program to access the Bluefors control unit for the data and host its own WebSocket server to send the data. The JavaScript was then able to access the locally hosted WebSocket server to retrieve the data because there would be no SSL errors. Other than the difficulties caused by the SSL certificate, other programming issues were resolved by reading different documentation and understanding the tools and components used to a higher degree.

Future Work and Impact

Moving forward, with more time, this can become a very important project because the program can progress into obtaining the functionality to remotely access and change values regarding the dilution refrigerators. This can be a very important contribution because remote access can allow for more efficient experiments. Also, it is possible for this project to be very important because it can be expanded to find trends in data or to store the data of all of the DRs in a database. The project was funded by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists. This investment can prove to be very beneficial if the research continues to expand as expanding on the program created can allow for intuitive additions that can increase overall efficiency and progress made.

Conclusion

In the final analysis, the goal of the project was to create a web interface that has the practicality to display data from multiple DRs and update the data in real-time. The goal was achieved after understanding how to create a program in Python that can communicate with the Bluefors control unit and retrieve the information that is expected to be displayed. Also, it was necessary to understand how to create a web interface using different methods. Initially, HTML and CSS were used to create the web page, but in the end it was updated to use Next.js, REACT, NodeJS, and JavaScript. With the understanding of these tools, the project was able to be completed by creating a sufficient web page that has the ability to display data and access the Python program to retrieve the data that is communicated via the Bluefors control unit. All in all, the goal was accomplished, however there could have been more optimizations made, such as changing the method in which the data is transferred to be more simple and changing the data retrieval technique from polling to listening for a less intensive program. Also, this project could have been optimized much further with an SSL certificate. This is because having an SSL certificate would allow for the web page to directly communicate with the Bluefors control unit and it would get rid of the need for the Python program to act as a proxy server for the data.

Appendix

Participants:

Name	Institution	Role	Impact
Fazal Quadri	University of Illinois at Chicago	CCI Intern	Supported in the process of programming multiple different components.
Nicholas Bornman	Fermilab - SQMS Center	Supervisor	Overlooked the programming process, provided goals to achieve, and sanctioned the use of the Bluefors control unit.

Scientific Facilities:

The work done in this project took place at the Superconducting Quantum Materials and Systems Center, which is led by the Fermi National Accelerator Laboratory. The laboratory was used to access the Bluefors control unit and the dilution refrigerators that are used by the SQMS center.