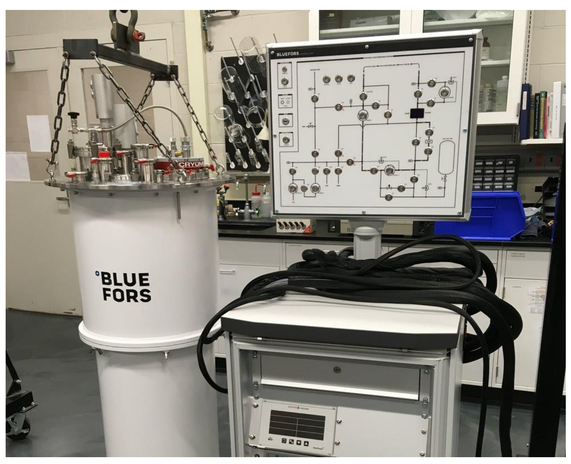
FERMILAB-POSTER-23-252-STUDENT

Quantum Computing – Real-Time Data Processing from a Dilution Refrigerator

Fazal Quadri, Fermilab & SQMS Intern, Supervisor: Nicholas Bornman

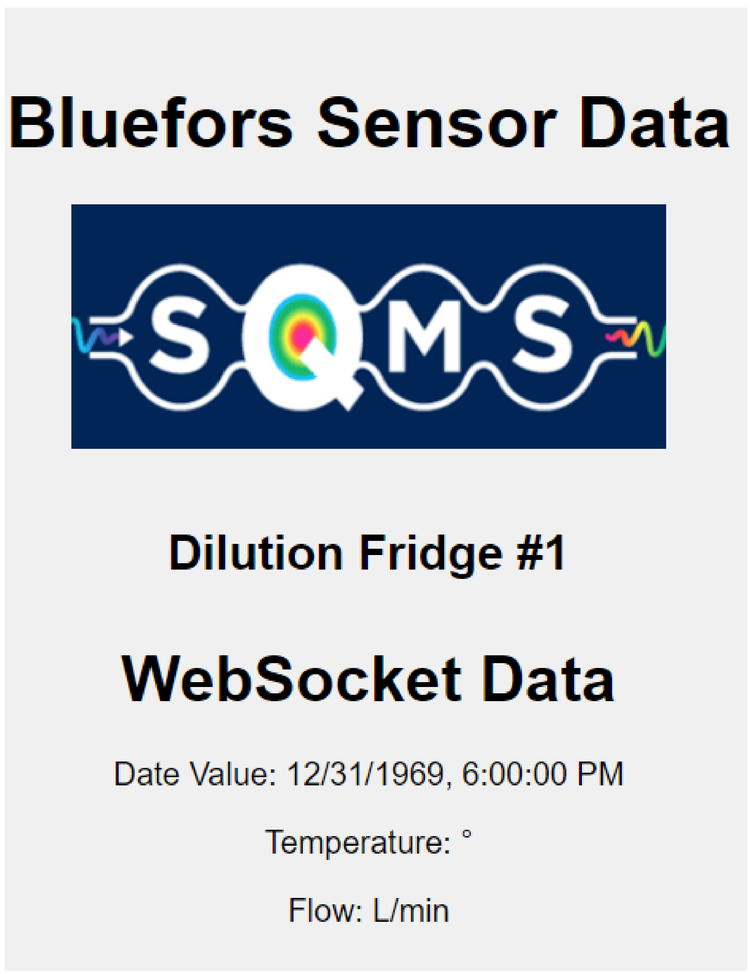
Overview of Project

- Stream data from the Bluefors Control unit to access physical quantities, such as temperature and helium flow, of the system at different points in the fridge
- Display these values, which are continuously updated, on a monitor with an aesthetic webpage
- Incorporate numerous connections of the dilution refrigerators into the WebSocket server



Bluefors Control Unit that monitors various temperatures, pressure, flow, etc., inside a dilution refrigerator.
(https://www.hybridquantumlab.com/new-blog/2019/1/3/santa-brings-lhqs-new-ld400)

The inside of dilution refrigerator that provides continuous cooling to temperatures as low as 2 mK
(https://bluefors.com/products/ld-dilution-refrigerator/)



Sample output of data exported from the Bluefors Control Unit

Note:Values are not displayed as the Bluefors Control Unit isn't on

- Prints the date, time, temperature and flow on the webpage
- Features a visually appealing and professionally designed user interface
- It is possible to add more refrigerators with similar layout

```
    if status == "SUCCEEDED":
        value = parsed_data['data']['mapper.bf.flow']['content']['latest_value']['date']
        value = str(value)
        #print(value)
        return value
    elif status == "RECEIVED":
        #print("")
        return 0

async def send_values(websocket, path):
    while True:
        if values:
            value = values.pop(0)
            await websocket.send(value)
        await asyncio.sleep(1)  # Wait for 1 second before sending the next value

async def main():
    # Start the WebSocket server to receive data
    receive_task = asyncio.create_task(receive_data())

    # Start the WebSocket server to send values
    server_uri = ""  # Update with your desired server address
    server_port = 8000  # Update with your desired server port
    async with websockets.serve(send_values, server_uri, server_port):
        print(f"WebSocket server started at ws://{server_uri}:{server_port}")
```

Python Script

- Creates a WebSocket server to establish real-time communication between the Bluefors Control Unit and the client computer
- Server receives live data from the Bluefors Unit, such as temperature and pressure readings, and parses it
- Asynchronous handling allows the server to manage multiple connections and serve multiple refrigerators simultaneously.

```
<!DOCTYPE html>
<html>
<head>
  <title>Bluefors Data</title>
  <script>
    var socket = new WebSocket("ws://131.225.94.167:8000");  // Update with your WebSocke

    socket.onmessage = function(event) {
      var value = event.data;
      updateValueDisplay("Received value: " + value);
    };

    function updateValueDisplay(content) {
      var valueDisplay = document.getElementById("value-display");
      valueDisplay.textContent = content;
    }

    setInterval(function() {
      updateValueDisplay("Waiting for new value...");
    }, 3000);  // Update the interval duration as desired (in milliseconds)
  </script>
</head>
<body>
  <h1>Bluefors Data</h1>
  <p id="value-display">Waiting for new value...</p>
</body>
</html>
```

HTML/JavaScript

- WebSocket connection listens for incoming data from the server, capturing measurements sent by the Bluefors Control Unit.
- Using JavaScript, the web page dynamically updates the displayed data as new measurements arrive, ensuring real-time visualization for the user.

Acknowledgement & Reference

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 Internships Program (CCI)