Oren Moreno

ITAI 3377

Professor Patricia McManus

February 26, 2025

**Reflective Essay: My Journey Through an IoT Sensor Simulation Project**

**1. Introduction**

This project focused on simulating and visualizing real-time environmental data, specifically temperature and humidity, using three key Internet of Things (IoT) protocols: MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), and OPC UA (OPC Unified Architecture). The setup involved creating a virtual Industrial IoT (IIoT) environment where I generated sensor data, transmitted it via these protocols, and displayed the results through dynamic graphs. The visuals, line graphs plotting temperature and humidity alongside terminal logs of raw data, serve as evidence of the simulations’ successful operation. My aim and goals were relatively straightforward: to deepen my grasp of IoT protocols, sharpen my technical abilities in programming and data visualization, and build confidence in tackling practical tasks. I anticipated challenges, particularly around configuring the protocols and resolving unexpected issues, but I viewed it as an opportunity to blend conceptual knowledge with real-world application.

**2. Personal Contributions**

I completed this project individually from start to finish. Handling it independently meant managing all aspects of the simulations and repository population. I configured an MQTT broker to manage real-time data flow, publishing and subscribing to streams of JSON-formatted temperature and humidity readings. These readings, such as { "temperature": 20.09, "humidity": 32.71 }, were logged and visualized in a graph with sharp temperature swings and steady humidity trends. For CoAP, I built a server and client to exchange data, plotting the outcomes in a graph where humidity spiked near 50 while temperature held steady around 20-25, showcasing periodic updates. With OPC UA, I set up a server, defining nodes (in my case, ns=2;i=2 for temperature), and coded a client to fetch and graph values like 23.99 for temperature and 39.76 for humidity, reflecting consistent data trends. Using Python, I scripted the sensor simulations and utilized Matplotlib to craft real-time visualizations as graphs with blue temperature and orange humidity curves over short, successive increments of time. Besides the simulations and visualizations, I also populated the GitHub repository with all files including a brief “Readme” and a comparative report. Taking full responsibility pushed me to better understand the system’s intricacies, though it demanded significant troubleshooting to complete the project.

**3. Learning Outcomes**

Diving into MQTT, CoAP, and OPC UA hands-on and individually revealed the nuances in their respective strengths and weaknesses. MQTT, a lightweight, publish-subscribe protocol, shone for real-time updates with low-bandwidth restrictions. Configuring the broker and scripting data publication showed me its efficiency. Rapid graph updates mirrored the terminal’s steady stream of JSON entries, proving its fit for scalable IoT networks. CoAP, built for constrained devices, operates on a RESTful model over UDP, teaching me about efficiency in limited settings. The graph’s reliable, if slightly delayed, updates underscored its role in low-power applications. OPC UA, with its robust data modeling and security, felt like a leap into more industrial-grade systems. Setting up nodes and syncing client-server data (e.g. temperature at 23.99, humidity at 39.76) was intricate but interesting, as the graph’s steady trends highlighted its precision and reliability.

**4. Challenges and Solutions**

I ran into a few difficulties during this project that tested my patience and troubleshooting abilities. The first issue that I ran into was a stubborn port conflict that stalled my MQTT broker. Running netstat to free up port 1883 got it working, though, with terminal logs printing out data matched by the graph’s live updates. Another issue with UDP caused spotty graph updates in the CoAP visualization. Binding the server to 127.0.0.1 and tweaking request timing smoothed out the graph updates. Mismatched node IDs broke the OPC UA client’s data pull. Debug prints in opcua\_sensor\_simulation.py pinpointed the fix, correcting IDs to “ns=2;i=2” and “ns=2;i=3,” syncing terminal values with the graph. Finally, MQTT’s visualization faltered due to apparent threading issues. Adding a queue to shuttle data safely from subscription to plotting threads kept the graph running smoothly. These hurdles helped me practice my troubleshooting skills and better understand how the different simulations and visualizations work.

**5. Future Applications**

This project could be adapted to things like smart environmental monitoring or industrial prototypes by using networks of real sensors and improving the basic graph visualizations into full-fledged dashboards for real-time insights. I’d be very interested to see how machine learning could be used to predict trends from the graphed data or blend MQTT’s speed with OPC UA’s security for a hybrid system. Additionally, the skills that I used for this lab will be helpful for successfully implementing real-world IIOT applications.