EE 250 Final Project

Team Members:

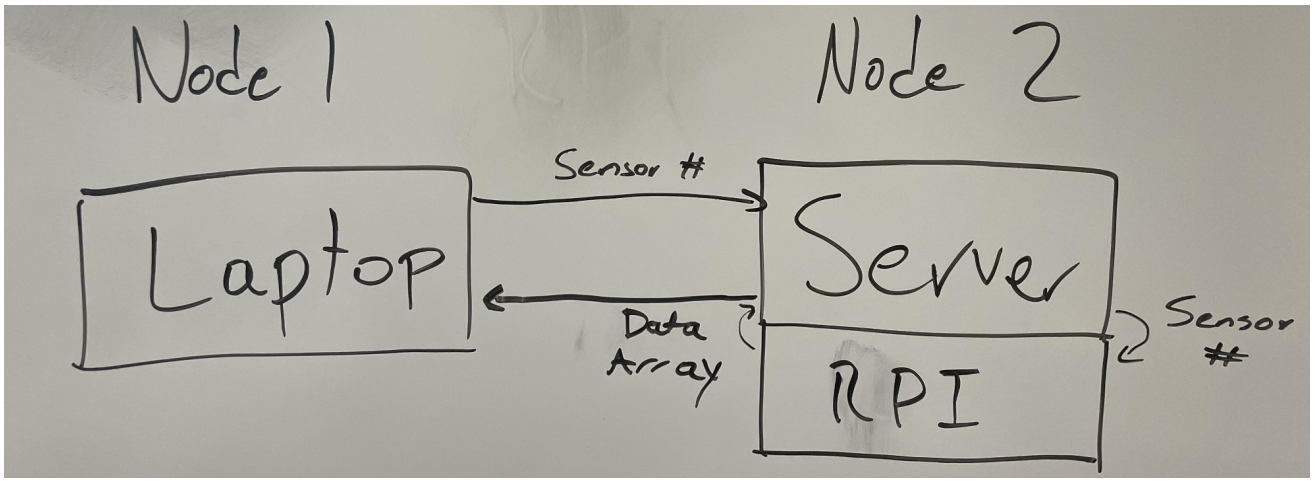
Isaiah Lee
Michael Ruiz

Link to Video:
https://drive.google.com/file/d/1UVBXP_7mLey34AJWaSK0Gowb9Z13MoWr/view?usp=sharing

Project Description:

Our project collects data from the surrounding environment using one of three sensors: light, sound, or ultrasonic range finder. First, the user is prompted on one node (a laptop in our case) to select which of the three sensors it would like to collect data from. This node publishes this information to a broker hosted on our Raspberry Pi, which the Raspberry Pi itself is also subscribed to. The Raspberry Pi then polls the sensor for a brief period, stores the data in an array, and publishes the array to the original node, which then graphs the data over time and applies polynomial regression to more effectively visualize the results.

Block Diagram:



\Components:

This project uses a Laptop and a Raspberry Pi as physical nodes. The Raspberry Pi runs a MQTT broker as well as its own code and operations.

MQTT: A form of Publish-Subscribe architecture allowing for asynchronous data flow and data decoupling, effectively allowing each node to act independently until such a time that it receives data. This is the protocol which was used to connect the nodes used in this project. (Source: Lab 4)

Mosquitto Broker: MQTT Broker which runs on the RPi, allowing for only two nodes to be used in this lab, despite the fact that both nodes were running code. The broker acts independently of the RPi and simply distributes information to and from each node when called on. (Source: Lab 4)

Matplotlib Graph: Python library allowing the generation and storage of graphs to visually represent the data.

Polynomial Regression: Polynomial regression tries to fit data into an Nth degree polynomial, similar in concept to how linear regression works, finding the data points with the best fit using the least squares method. In our case, since our data would act sporadically as we took measurements from sensors, we used a 20th degree polynomial in order to catch every rise and dip in our measurements.

MCP Chip: The MCP3008 is a 10-bit Analog-to-Digital Converter. It has 8 analog input channels and supports SPI interface. The sampling frequency is 200 kilo samples per second when powered at 5V. This allowed us to choose exactly how much we would want to sample without being restricted, as the maximum sample rate was much smaller than we needed it to be. (Source: Lab 7)

Light Sensor: Captures environmental light intensity and fluctuations which were used for data analysis and visualization.

Sound Sensor: Records surrounding sound intensity and fluctuations which were used for data analysis and visualization.

Reflection:

The freeform nature of this project meant we had to be uncomfortable. Whether it was working around MQTT, working with python lists and numpy arrays, and researching linear/polynomial regression, it was uncomfortable, but we learned so much. In terms of limitations for the project, our original goal was to perform a fourier transform on the audio data, and possibly implement some sort of low-pass filter to get rid of background noise. Our main issue was that the sound sensor used in this project reported a sound level rather than a signal. This meant we could not directly apply the transform and would have to convert from a series of integers to a wave. For this to work we could use a different sensor that would return a full audio signal. For lessons learned, we found it was important to routinely check hardware to ensure that all wires and components were properly plugged in. We also found it was useful to check our code every time we added a functionality, to make sure things were working.