EE 250 Project Writeup
Owen Zeng & Felix Chen

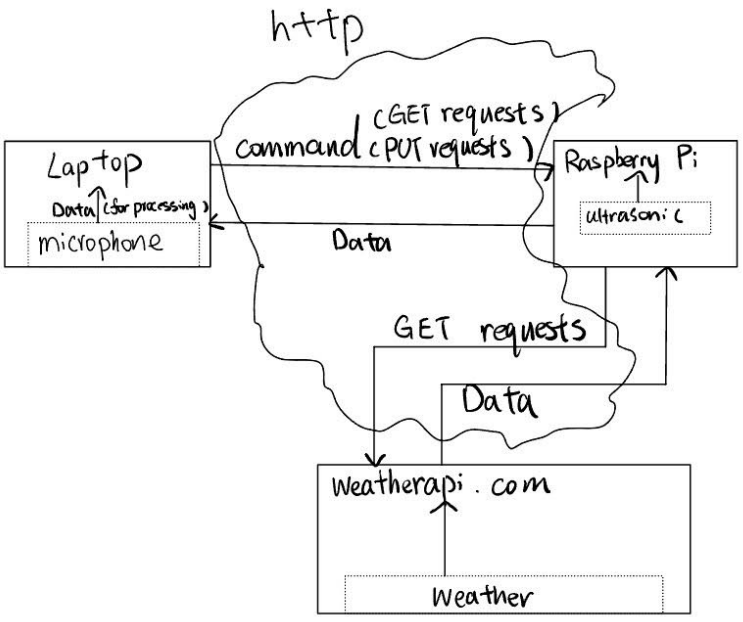
Project Overview & Functionality

The Voice-Controlled IoT System is our project. It is a multifaceted system that combines the Internet of Things (IoT) devices with voice commands. The project was meant to mimic a smart assistant like "*Alexa*." Voice commands can be used to connect with a Raspberry Pi and do things like get weather updates, control an LED, and see data from ultrasonic sensors. We have two nodes: a laptop and a Raspberry Pi. And we have three sensors: a microphone on the laptop, an ultrasonic sensor on RPi, and a virtual weather sensor.

A key feature of our system is its real-time signal processing capability, where voice commands are captured, processed, and translated into actionable commands for the Raspberry Pi. This processing includes noise filtering and audio data transformation, which are essential for accurate voice recognition. The communication between the laptop and Raspberry Pi exemplifies efficient node-to-node interaction in an IoT ecosystem.

In terms of visualization and control, our project offers a dual approach. Firstly, a user-friendly web front-end allows for direct control of the Raspberry Pi's functionalities, like toggling the LED and accessing sensor data. This interface provides an intuitive way for users to interact with the IoT system without voice commands. Secondly, our project features a unique web-based visualization tool (web.py) that displays signal diagrams of the processed audio input. This visualization is particularly crucial in demonstrating the signal processing aspect of the voice commands, offering insights into how voice data is transformed and interpreted by the system. While our system primarily focuses on voice command processing and response, the added layer of web-based control and visualization complements the overall functionality.

Block Diagram



Components

Laptop: Acts as the primary interface for voice input using a microphone. It processes the voice commands using Python scripts.
Raspberry Pi: Serves as the IoT node, connected to sensors and actuators. It receives commands from the laptop and performs the corresponding actions.
Ultrasonic Sensor: Measures distance data, showcasing real-time sensor interaction.
LED: Acts as an actuator, demonstrating physical response to voice commands.
Virtual Weather Sensor: Fetches real-time weather data from an external API.

Communication

For communication, we employed HTTP protocols to send requests between the laptop and Raspberry Pi. The Flask framework facilitated a lightweight server on the Raspberry Pi.

Processing Techniques and Design Choices

The project heavily relied on signal processing techniques, especially for voice command recognition. We implemented Fast Fourier Transform (FFT) for noise filtering in the audio signal and used the Google Speech Recognition API for converting speech to text. For text-to-speech functionality, we integrated pyttsx3, allowing the system to provide verbal feedback to the user.
A key design choice was to process the voice commands on the laptop rather than on the Raspberry Pi to reduce computational load on the latter. Also, the voice recording is typically around 150 KB. This is a significant amount of data compared to typical JSON payloads used in IoT applications. Transferring these relatively large audio files between the laptop and Raspberry Pi posed a challenge in terms of data transmission time and efficiency. Large file sizes could lead to noticeable delays in processing and responding to voice commands, especially over wireless networks where bandwidth and stability can vary. This decision ensured smoother operation and quicker response times for the IoT device.

Limitation & Reflection

One problem we ran into was that the weather API and speech recognition services needed to be connected to the internet all the time. Any problems with the network had a direct effect on how quickly and correctly the system worked.
Through this project, we learned about threading by using it to run the Flask server on the Raspberry Pi and receive data from ultrasonic sensors at the same time. This real-life use of threading, an idea we learned in class, helped us understand how IoT systems can do more than one thing at once better. We also had trouble with visualisation. We had planned to use Grafana at first, but instead we made a simple web frontend and a web.py script to show how voice order signals are processed. This change forced us to learn more about the web development, which showed how hard it is to make data easy for people to understand. These events made us realise how hard it is to make technology easy for people to use and how important it is to learn new skills.