Carson Wittwer: bbplayer@vt.edu
Jason Nelson: njason18@vt.edu
MH Charles-Etuk: mh6698@vt.edu
ECE 4564, Assignment 2

Objectives

The objective of this assignment was to gain experience in programming with message middleware, Bluetooth communication, database managers, and GPIO as well as their practical use in facilitating the use of data-driven network applications. In this project we designed a rudimentary bookkeeping system that could handle basic requests on the part of the user. We used two Raspberry Pi 3+s outfitted with the message queue platform RabbitMQ to serve as the Client & Processor, and a third Raspberry Pi 3+ with the database manager MongoDB to serve as the storage unit for the books in question. The Client would request an action, such as adding or buying a book: the Processor would listen for this request and relay it via Bluetooth to the Storage Pi, who would use the MongoDB framework to carry out the requested action against the existing database.

Design

Client:

The client first parses the command arguments for the processor's IP, the desired user action, the book in question, and (optionally) the number of books in question. Then, after assigning these arguments to variables, a package is built with th relevant information to be sent and then wrapped in JSON, before a connection is established to the Processor using the RabbitMQ protocol. This connection also creates a unique queue that verifies that each message is legitimate and upon production it waits for a response from the Processor. After receiving one, it decodes the Processor's package from JSON and displays it to the users, readying itself for any request. -MH

Processor:

The processor file is relatively simple. It hosts a RabbitMQ server, which listens for a request from the client. After printing the data sent by the client to output, it forwards it to the storage device using a Bluetooth connection. It then receives a reply message from the storage device, and forwards it back through RabbitMQ to the Client. -Jason

Storage:

The Storage file is the most complicated of the project. When initially starting the file, the program initiates a database and sets up the GPIO output. The database is not actually created until the first item is put into it. The processor Pi connects to the Storage Pi via Bluetooth connection, which is also set up on launch of the program. It takes in the payload from the processor as a JSON object. Then, after unpacking the payload, it reads the payload to determine what action needs to be taken on the database. After determining what action needs to be taken, it then acts, or doesn't on the database. From there, the answer payload is put together and then sent back to the processor over the same Bluetooth connection. While the database action is running in one thread, another thread is concurrently running ensuring that the GPIO is output. These are running at the same time, so no matter what is going on both are working. When the program is shutdown, the database is cleared and the GPIO is reset. The functions to implement MongoDB changes and GPIO where built in separate files so nothing outside of the class could change the data. -Carson

Conclusions

By the end of this assignment we had gained a substantial understanding of how to create continuous links between our Pis and send information between them. We came in with little to no experience using these platforms and through first building and debugging them we were able to gain familiarity with them and in addition worthwhile coding practices that will doubtless carry over into our future endeavors.

We learned how to pack and unpack JSON messages, how to deal with dictionaries and their elements in Python, how to send complicated messages over both Bluetooth & message queue platforms, and how to create GPIO algorithms that respond to user-driven conditions dynamically.

This served to further increase our knowledge of Python overall as these are very integral aspects of programming.