Group 12: “The Internet Explorers”
Logan Whitmire (lwhit@vt.edu)
Alex Tejada (alexart@vt.edu)
Neil Singh (neils@vt.edu)
Austin Lamicella (lustin@vt.edu)

ECE 4564, Assignment 2

1. Objectives

The purpose of this assignment was to communicate a message between three different Raspberry Pi’s, and store that message into a noSQL data repository. The three Raspberry Pi’s involved in the communication process are the Mobile RPi, the Bt/WiFi Bridge, and the Repository RPi. To utilize this system, the user simply enters input that they want to store in the database into the Mobile RPi. From there, the other RPi’s transfer the message to the database.

We approached this assignment by separating it into various parts. These parts included the Mobile RPi setup, the bluetooth integration, the Bridge RPi setup, the zeroconf integration, the rabbitMQ integration, the GPIO integration, the Repository RPI setup, the Regex integration, and the noSQL database integration.

To help with our integration of zeroconf, we used source code from:
https://github.com/jstasiak/python-zeroconf/

To help with our integration of rabbitMQ, we used source code from:
https://www.rabbitmq.com/tutorials/tutorial-six-python.html

We started by setting up the Mobile RPi. This allowed us to ensure that we could successfully parse input and store it in a noSQL database. We then setup the WiFi/Bluetooth Bridge RPi and ensured that we could send data via bluetooth. As we finished with our parts we pushed our code to a Github repository. From there we focused on creating a shared message queue to communicate with the Repository RPi. Our greatest struggle was getting the Pi’s to talk to each other over a RabbitMQ messaging service using zeroconf. This seemed a simple task, but ended up giving us a fairly large headache. We persisted through a couple bugs that were holding us back and preceded to upload the final code to our repository.

2. Responsibilities

Logan developed the ZeroCONF and bluetooth connectivity for the bridge.py, as well as assisted with debugging the final implementation of the project.

Austin developed the GPIO functionality, and ZeroCONF implementation, assisted with RabbitMQ messaging protocols, and helped with the debugging of the final system.

Neil developed the Mobile.py file, implementing the noSQL database, assisted with the setup up of the bluetooth connection between the Mobile and Bridge, and helped with debugging of the final system.

Alex worked on the integrating the RabbitMQ messaging protocols, and assisted on Bridge.py. He also set up the RabbitMQ server, and helped with debugging of the final system.

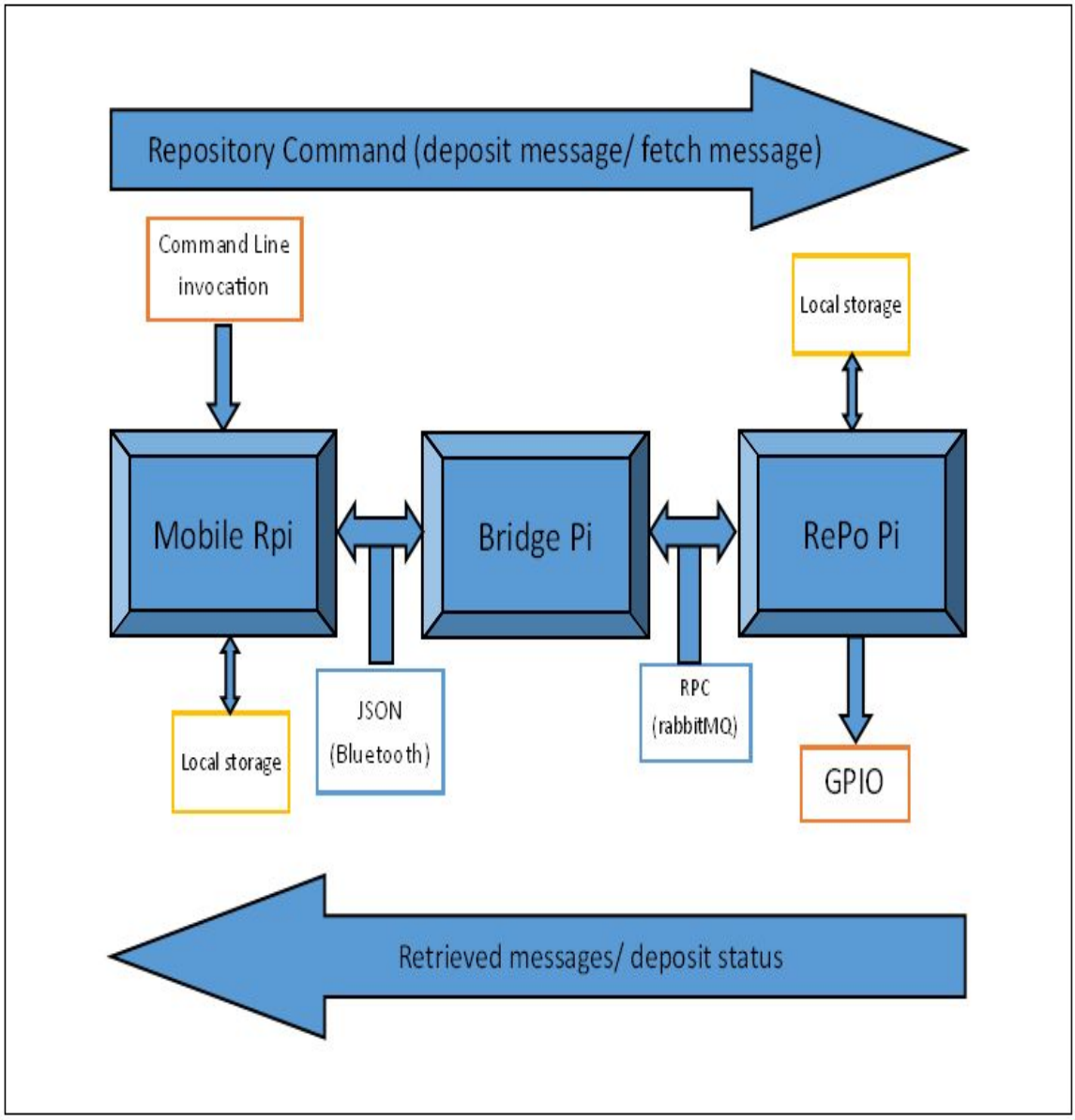
3. Conclusions

One of the biggest issues we faced as a team was the lack of monitors that we could use when we would meet up to work on the assignment together. We worked around this by constantly writing small bits of code in a modular fashion that would work with other teammates' code and kept checking to make sure that it would work when we all ran it separately.

Additionally, the RabbitMQ was a big hurdle for the team to work through. We spent most of our time debugging our rabbitMQ protocol. The proficiency with Python this project was much better than our last project. Also our overall chemistry as a team as greatly improved.

Upon completion of this assignment the team learned how to communicate between multiple devices via sockets and make use of an existing API. This will be useful information for working on our final project as we will need to communicate between different devices to deal with the incoming data from sensors.

Diagram on Next page.


Repository Command (deposit message/ fetch message)
Command Line invocation
Local storage
Mobile Rpi
Bridge Pi
RePo Pi
Local storage
JSON (Bluetooth)
RPC (rabbitMQ)
GPIO
Retrieved messages/ deposit status