ECE 558 - Fall 2018
Final Project - Progress Report

Brian Henson
Jamie Williams

Final Project: Security and Theft Detection App and Base Station

Progress Report

Project Description:

Our project is a security and theft detection system consisting of a portable base station (a Raspberry Pi with various sensors), to be placed where the user wants to detect an intruder (for example, at a door, inside of a dresser, or on top of a valuable item), and a paired phone app. When the alarm is armed, the phone app would be notified upon any disturbance sensed by the base station. The user could view a photo sent by the base station, and send audio output through the base station to warn or communicate with any intruder.

Project Status Summary:

We are currently behind our proposed timetable for this project. Neither the base station, nor the phone app progress have lined up with the schedule. Instead, we have changed several parts of our implementation, to utilize and work around the capabilities of Google Firebase. Additionally, the sensors for the base station will not arrive until Monday, December 3rd, which added a delay in several of the earlier timeline objectives.

The original timeline had finals week objectives as “test and finalize”, which gives us some room for adjusting. Therefore we feel that we can still get our project finished on time, and even meet a few of our stretch goals in addition to the basic functionality!

The biggest change we have made so far is that we moved all of the “active” communication (while the app is open) between the phone and the Pi away from the Messaging service and into the Realtime Database instead. This is because the Messaging service isn’t designed for sending from device to device; it is possible, but more complex than we expected. It is instead intended for server-to-device transmissions. However, we still need to use Messaging for “background” communication and to send notifications to the app; to do this, the Pi will write to the Database when it wants to send something, and a Node.js script that Jamie wrote will detect that and send a Messaging ping from the server to the app.

This scheme has two added benefits: first, the user’s phone will be using less battery overall, because it won’t need to receive constant pings from the Pi to know the Pi is still connected. And second, because the “is the Pi still connected” logic is in the Node.js script, the app user won’t get an alert for “the Pi just went offline” if they go through a tunnel and lose both WiFi and cell service.

We have also decided to implement Firebase Authentication, meaning that our hypothetical product could be used by multiple users (app-Pi pairs) simultaneously without interfering with one another, so long as each user uses a different username/password combo. This was mainly done because a) it wasn't very difficult, and b) Firebase Storage strongly recommends that we have authentication enabled, to prevent unknown persons from storing whatever data they want on our system.

Proposed Schedule with Progress Notes:

Week of Nov 11th-17th
Phone app can send and receive data to Raspberry Pi app (Project # 3 Requirement) **DONE**

Week of Nov 18th-24th
~~Phone app~~ Firebase triggers notifications upon an alarm trigger or base station lapse. Half done
Phone app can retrieve sensor config from the database. **DONE**
Raspberry Pi app correctly reads sensor data.... Delayed
...and sends it to the database. No longer needed
Raspberry Pi app has 'setup' mode to establish normal sensor range. Delayed

Week of Nov 25th-Dec 1st
Phone app can play alarm or send microphone data to base station. Not done
Phone can view photos taken by the Raspberry Pi. Not done
Raspberry Pi can take photos on command or trigger and send them to the database. Not done
Raspberry Pi speaker can play sounds sent from base station. Not done

Week of Dec 1st-7th
User interface of the phone app is improved, tested, and finalized. TODO
Raspberry Pi app sensor thresholds are tested and finalized. TODO

Phone App and Database Setup - Brian Henson

In my work on the Phone App, I have gotten the Firebase Messaging Service added in a demonstration capacity. I am able to receive messages and generate notifications, and I know how to launch the app when a notification is clicked, but I haven't implemented it yet. I've got Firebase Authentication working flawlessly, and the communication channels through the Database aren't yet set up, but I know how to work with the database so it should be straightforward. The overall UI structure is halfway done; I decided to go with a ViewPager that contains 3 fragments, and 2 of the 3 UIs are done. I still haven't figured out how to download images from the Firebase Storage, nor how to record and upload sounds. I have, however, done some research on VOIP and think that a live voice call is not impossible to arrange, and it could even work in a "drop-in" configuration. I need to do further research before committing to either the "upload/play recorded message" or "establish mono-directional audio-only VOIP call" strategies.

Overall, many half-done and half-explored features… so the timeline looks bad, but I'm actually making acceptable progress.

Final note: I've been having serious struggles with the Styles/Themes of the app, and have resigned myself to a functional but ugly app interface.

Raspberry Pi Base Station and Firebase Functions - Jamie Williams

For my part of the project, I have been focusing on developing Firebase functions to send notifications to the phone app. Originally, this was not a part of the project, but was necessary to implement a notification when the base station was not communicating to the Firebase database. After examining the capabilities of the database, and the likely use cases and issues, my partner and I decided that using Firebase functions would be the best option.

Firebase functions are written in Node.js, which uses JavaScript to define database functions. As I do not have experience with either JavaScript or Node.js, this portion of the project has taken longer than I anticipated. Currently, I have a Firebase function that generates a notification when the base station writes an alarm has been triggered. However, I have been having issues with figuring out the correct method to determine if the base station has been disconnected. I have several different option on how it could be possible to do this, and I plan to experiment to find one that works.

For the base station app code, I have been integrating various pieces of code, from my partner and from previous projects, to give it database connection. Unfortunately, I will not receive the sensors needed for the project until Monday, so I have not been able to work on the sensor processing code (which made up most of the milestones). If necessary, I might have to simplify some of the data processing code for the presented project.

I feel that I have made significant progress on my part of the project, but the order in which I have done so has been very different from the expected timeline, mostly due to the delay in receiving the sensors. However, the timeline for the last week was lenient, so I feel that I can get everything done in time for the demonstration.