IoT Sensor Integration and Back-end Development for Sequoia

Final Report

Client:
Andrew Guillemette

Team Members:
Cody Brooks
Guan Lin
Josh Hatton
Josh Lang
Justin Somers
Mike Ludewig

Team Email: sdmay19-36@iastate.edu
Team Website: http://sdmay19-36.sd.ece.iastate.edu
Revised: 2018-4-30 / Version 1

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

0 Executive Summary

0.1 Acknowledgment

Team 36 Client: Andrew Guillemette
Team 36 Advisor: Daji Qiao
Pilot Program Facility: Green Hills Retirement Community
Pilot Program Resident: Bob Kern

0.2 Problem and Project Statement

Problem Statement

On average the American population is getting older. With this increase in age comes an increase in health care needs, and an increase in the need for systems to monitor their health. Since many senior citizens have habits in their daily routine, a lot of health information can be gathered from these habits. The goal of the Sequoia project is to put sensors in the homes of senior citizens to monitor their habits. Then use machine learning with that data to build a behavioral profile for the senior citizen, and to see when a deviation from those habits might indicate a health issue before other symptoms are showing.

Project Statement

Our team has been extending an existing project. The focus of our group within the project was to add more sensors to the suite of sensors that was already in place. The three new types of sensors that we added were smart plugs on multiple appliances in the kitchen, a flow meter to track water usage from the kitchen sink, and a smartwatch to track the heart rate and step count of the senior citizen. All this information will be used to help us build a better behavioral profile for the senior. We also extended an Android app that was started by a previous senior design group to display the new data we gathered. This will allow the seniors loved ones to keep better track of their important health data.

1 Requirements specification

1.1 Functional Requirements

Functional requirements:

FR.1: The data from the Smart Plug, Flow Meter, and Smart Watch shall be transferred to the AWS server.

FR.2: All data on the local network, including the smart plug and flow meter, should be transferred to the AWS server from the smart hub.

FR.3: The AWS server shall process incoming events from the smart plug and flow meter to determine the duration of time the event happened.

FR.4: The AWS server will distribute the information to the database and to the mobile app to view the information.

1.2 Non-functional Requirements

Functional requirements:

NFR.1: 3rd party products used are readily available for purchase in large quantities to be scaled to large amounts of customers.

NFR.2: The implemented systems operate wirelessly to transfer data and communicate with the smart hub and user.

NFR.3: The product requires no user interaction or maintenance to continue to monitor and transmit data.

1.3 Users and Use cases

The overall system includes three intended users: the senior that is monitored, the senior's visitors (family and friends), and the senior's doctor's or nurses. Initially, the project inherited a door sensor & smart hub from a previous senior design group. Our project expanded on this network by adding three additional components: A smart outlet, flowmeter, and a Fossil Smart Watch. Therefore, our scope was limited to these three components. Due to the limited scope, our only user will be the senior.

The smart outlet is used to track which appliances are being used, and for how long. It reports the time the event occured, along with specific power statistics. The outlet primarily records the current (Amps), voltage (Volts), and power (Watts). Once this data is recorded, it is sent to the smart hub, which then transfer the data to the AWS server to compute the time duration of the event.

The Fossil smart watch is used to provide the senior's heart rate and step count throughout the day. This information is then directly transferred to the AWS server for safe keeping. For future senior design groups, this information will be monitored to check if there are any irregularities in the senior's heart rate, and thus contact the senior's loved ones and health care professionals.

Additionally, a flowmeter is used to track when water is used in the kitchen sink. This, alongside the smart outlet are used to identify when a senior is cooking, or getting water. With the flowmeter data collected, the data is forwarded to the server. The data sent includes information such as the timestamp of when the event occured, the flow meter id, the rate of flow, and also the amount of water used throughout the duration of the event.and will include information such as timestamp of when event occurred, the flowmeter id, and the flowmeter device name. Depending on the flow meter id, the event then identifies if the senior uses hot or cold water, or both.

2 System Design and Development

2.1 Design Plan

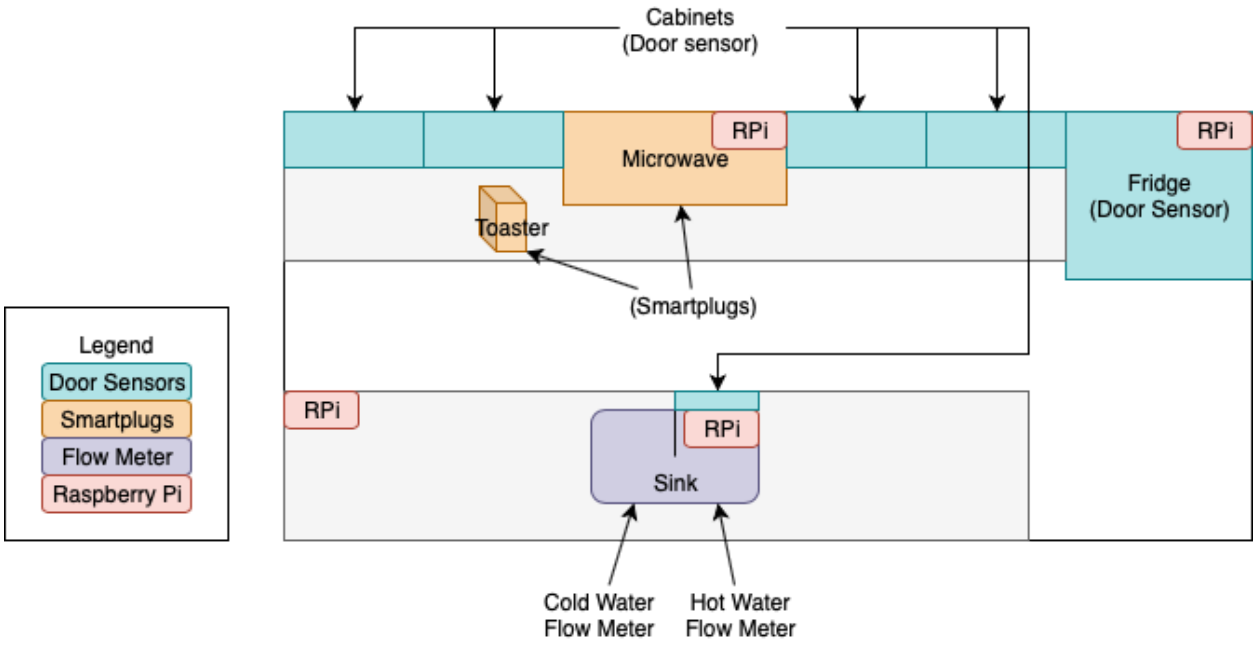

Figure 1: Conceptual Diagram

The focus of our project for monitoring is in the resident's kitchen. The sensors are connected to Raspberry Pi Zeros that send data to the Raspberry Pi 3 that works as a sensor hub. The hub uses a WiFi connection to then send the door sensor data to an AWS server for storage. This data will be used to create event identifiers that will help determine things like the resident was eating. The events will be then be used to build a behavioral profile of the resident.

2.2 Design Objectives, and Constraints

Having only door sensors predicting if a resident is eating is fairly inaccurate. Our team has been tasked with adding components to make the event identification more accurate. Our extension adds three major components to the existing design.

Smart plugs will be added to capture when a device is in use. Knowing when devices such as a coffee pot or a microwave are used can help with event identification. If a resident used a coffee pot and microwave, the probability of the resident eating breakfast increases dramatically.

The next extension we are adding is a flowmeter. The flowmeter is a device to measure water flow through a pipe. For our project, we are using the flowmeter to determine

when the sink is being used. Knowing when the sink is being used can also help event identification probabilities.

The resident will be outfitted with a smartwatch. The smartwatch will be used to get heart rate data of the user. The heart rate data will also be included in the behavioral profile.

Assumptions:

- The senior will not take off the smart watch
- The senior will not plug appliances to non-smart outlets
- The senior will charge the wearable device when necessary
- The senior will not rearrange their kitchen cabinet drawer contents

Limitations:

- The smart outlets must not greatly restrict outlet usage
- Not all appliances run on standard 120 volt outlets so a different method is needed to monitor their power.
- The amount of data we can get is restricted by the API watch
- The smart watch may need to connect to a phone

2.3 Architecture Diagram, Block Diagram

Pre-Existing Architecture

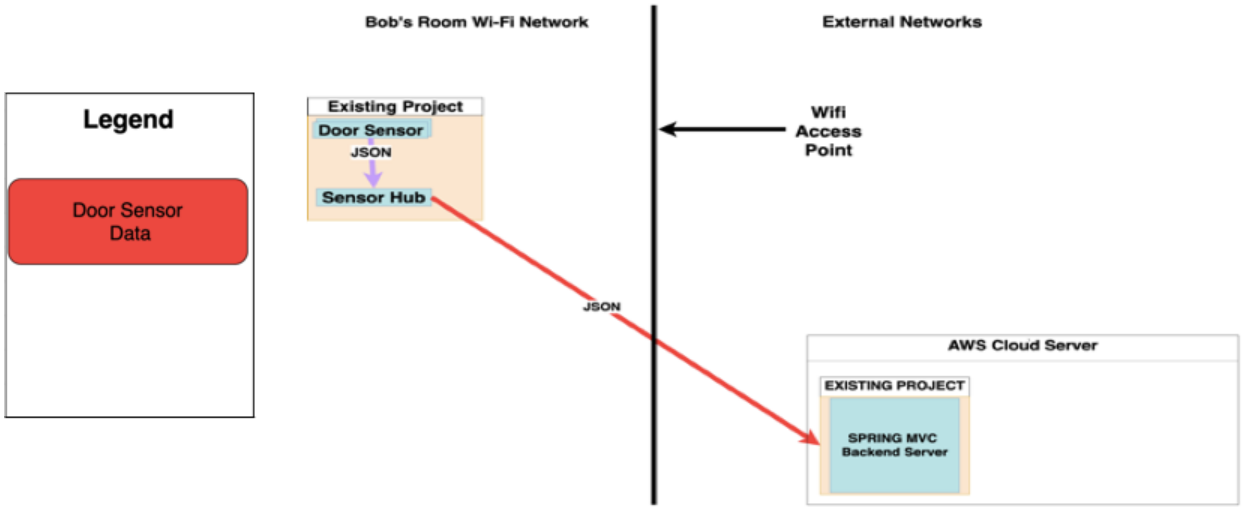

Figure 2: Pre-Existing Architecture Diagram

As stated, our team has inherited this project from a previous team. This figure is shown in figure 2**.** The previous team had added multiple door sensors to the drawers in the resident's kitchen. These sensors would send data on when and how long the drawer doors are open. This data is sent to the sensor hub through a local network. The sensor hub will then send this data to the backend server to be stored on the cloud through JSON.

Current Architecture Diagram

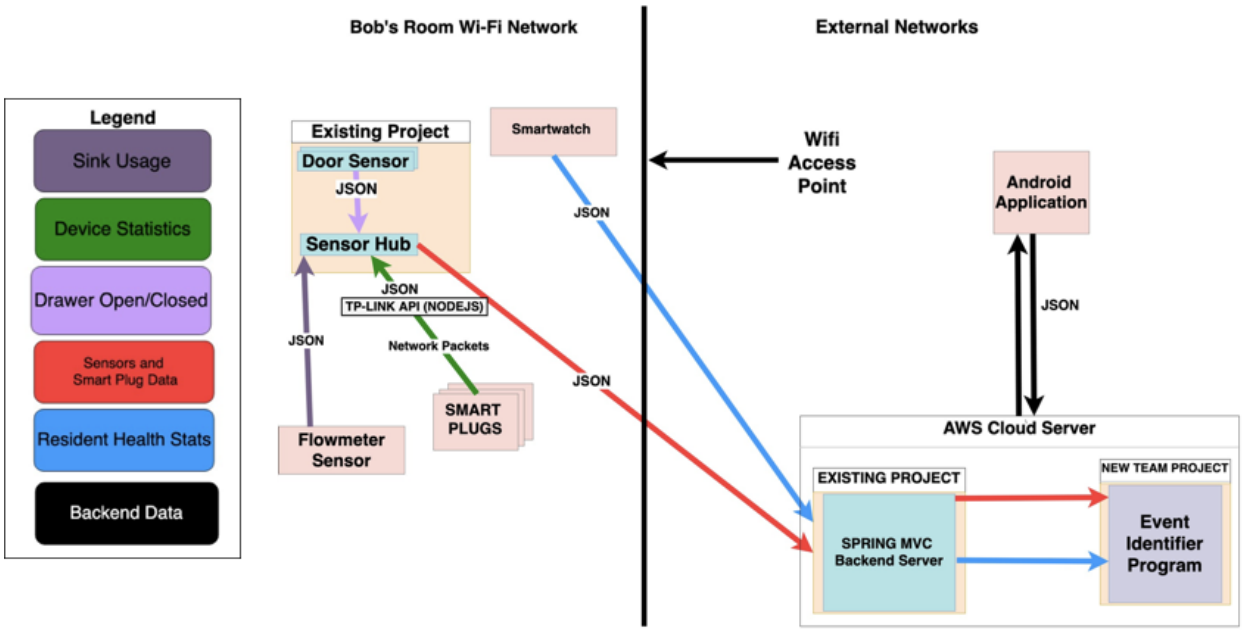

Figure 3: Current Architecture Diagram

In order to get more data to get a more accurate behavioral profile, our team has integrated three more components to the resident's apartment. The first is a smartwatch to monitor heart rate as well as daily step counts. This data is sent directly from the watch to the server through Wi-Fi. The second component is the smart plugs. The plugs send data on what time a device is used and for how long. Finally, the flowmeter sensor was added to measure water usage from the kitchen sink. The data that is being collected includes whether the hot or cold tap was used, the duration the water was on, and the duration the tap was on. Both the flowmeter and smart plugs communicate with the backend server through the sensor hub.

In addition, our team has developed an android application to display the data collected from all of the sensors. This information is transmitted between the server and the application through JSON as well.

The data collection was the main focus for our phase of the project. The next phase will entail taking the data that our project stores and to begin identifying events to build a behavioral profile. This component is marked as the "Event Identifier Program" in figure 3.

2.4 Modules and Interfaces

Modules

Door Sensors

Magnetic door sensors that can detect when the two magnets are apart and for how long. When the magnets are apart, our system sends a door open event to the backend server. These sensors are wired to individual Raspberry Pi Zeros. This component was done by a previous senior design team.

Figure 4: Door Sensors

Sensor Hub

A Raspberry Pi that acts as the main hub of communication of data between the sensors located in the apartment and the backend server. The hub has a local server to get data from the the Pi's connected to the sensors and also runs the TP-Link Smart Home API to get data from the smart plugs.

Smart Watch

A Fossil Gen 4 Sport smartwatch that is to be worn by the resident of the apartment. Our team has written a watch app that gets the heart rate and the

step count for the resident. The watch or the tethered phone will then send the data to the backend server directly through Wi-Fi.

[1]

Figure 5: Fossil Gen 4 Smart Watch

Smart Plugs

A series of TP-LINK HS110 plugs that are used to get power level statistics on some of the appliances in the resident's kitchen. The plugs can be seen monitoring the resident's toaster and coffee pot in figure 6.

Figure 6: TP-LINK smart plugs in resident's kitchen

TP-LINK API

The HS110 plugs are able to give us power level statistics but is not able to tell us when the device is actually being used. In addition, our team was not able to

get the data from the plugs except through the provided TP-Link app. As our team wanted an event-driven system where we wanted the plugs to send off a notification when the device was being used we are using an open source API called tplink-smarthome-api (https://www.npmjs.com/package/tplink-smarthome-api).

The developers of this API have reverse engineered the network protocol of the smart plugs in order to get the data directly from the plugs and bypass the TP-LINK servers. Our team has extended this API to send an event to the backend server whenever the connected device reaches past a certain power threshold.

Flowmeter Sensor

Installed meter seen in figure 7. A sensor that is able to detect water flow by measuring the electrical power that is generated by water flowing through the sensor. There are two sensors within the casing that detects hot and cold water, respectively. In between the two sensors is a Raspberry Pi that has a Python program to detect and send data from the sensors to the sensor hub.

Figure 7: Installed Flowmeter Sensor

Spring MVC Server

The backend server for the project and initially started by a previous senior design team. Our team has extended the server by adding controllers for the smart plug, smart watch, and flowmeter data. In addition, our team has a added a process to calculate durations for the smart plug events for each device.

Android Application

An android application that is used to display all the data that is collected by the different sensors in the residents room. Communicates with the backend server through JSON.

Interfaces

Our system has two main interfaces. The interface between the 3 sensors and the sensor hub, and the interface between the sensor hub and the backend server. Both interfaces really on HTTP requests with JSON payloads in order to send data between the interfaces.

3 Implementation

3.2 Technologies Used

The project uses five main technologies, Python, NodeJS, Java Spring, mySQL, and Android. Together, all five of these components are used to implement our integrated system.

The python component of the project was inherited from the previous senior design group. They decided to use python to transmit the door sensor data to the sensor hub when a door is opened or closed. This was achieved through a python script running on the door sensor, and also a python HTTP server running on the sensor hub. Our project expanded upon this component by using python similarly to transmit data from the flow meter to the python HTTP server. However, not all data for the project could be collected via python, due to the smart outlet being limited to a specific external API.

To solve this issue, we utilized the Node.JS API to gather data from the TP-Link Smart Outlet servers. This was selected, as it was the the only available API for the current time. We wrote a NodeJS script to subscribe to the API to receive notifications when the smart plug was in use. This script ran on the sensor hub, and the data received from the script, was then transmit to the AWS server.

For the AWS server, we utilized Java Spring to create our API to process all of our incoming data and events. The AWS server used GET and POST HTTP requests to receive data from the sensor hub & smart watch, and send data to the android app (and for future senior design groups: a website or another front-end service.) The server also performed data processing for certain requests, such as the smart outlet, to determine how long the smart outlet was in use. Once the information was processed, this data was stored on our mySQL database.

The mySQL database is only accessed from the AWS server, as both are hosted on the same linux instance. When the AWS server receives a request, it stores the information in the database. When an external service requests information from the AWS server, it uses our Java Spring API to make calls to the database.

Finally, the project utilized android for two components, the android app and the WearOS smart watch app. The Android app displays all of the sensor information collected, making API requests to the AWS server for updated information. The WearOS Smart Watch uses android to collect heart rate & step count information from the Fossil watch itself. It then makes a HTTP POST request with a JSON string body to the AWS server with the updated information.

Figure 8: Various screenshots of the Android App

3.3 Design Analysis

For the smart plug, our team had to choose between the TP-Link smart plug, Koogeek smart plug, and a University of Michigan voltage detector that connected to the cord of the device called PowerBlade [2]. The TP-Link was chosen because it is cheaper than the Koogeek plug and would take less time to integrate into the existing system than the PowerBlade. The PowerBlade was also not available on the market. Finally, we were able to find a 3rd party application to send event data directly to our own database only when the smart outlets are being used.

In the decision to pick a smartwatch, we settled on using a WearOS smartwatch as it is compatible with both Android and Apple phones. For the specific smartwatch, we initially chose to use a LG Sport due to its ability to function as a standalone device (can

connect to LTE without needing a connection to a smartphone). We later found out that this would not work with our client's carrier, and instead ended up using a Fossil Gen 4 smartwatch as it worked best with our client and their carrier. While it did not have standalone LTE capabilities, we were still able to use it to send heart rate and step count data over a bluetooth connection to a paired smartphone, at which point it would then be sent to our database.

For the flow meter, we ended up selecting to use the Hadronix Water Flow Sensor. The reason for choosing this over other flow sensors we looked at and tested, the Handronix sensor was food grade, meaning it would be safe to use with measuring water that could be used for consumption.

In terms of the indoor location tracking and fall detection, these requirements fell out of scope as our client was not able to come to an agreement with vendors to supply their product for these solutions.

3.4 Standards and Best Practices

There is one main standard, and one other best practice that we used in our project. The standard that we followed was IEEE standard P11073 Part 10471: Device specialization— Independent living activity hub. This standard told us the best way to implement our sensor hub so to make sure that all the data was being aggregated in one place and only sent to the server from one device. The best practice that we followed was to make sure that all of the products that we used that would contact food or liquid are NSF certified food safe. We didn't want anything in the kitchen to be contaminated by our product so we made everything food safe.

4 Testing, Validation, and Evaluation

4.1 Test Plan

Our main objective was to run integration testing on all of our components that our team worked on. In order to test the sensors we developed, a test environment was set up to see how it would run when we added it to the resident's apartment. In the test environment, the team worked out any errors that arose during testing and made sure they were fixed before the sensor was added to the system of sensors.

Smart Plugs

Testing for the smart plugs consisted of using devices and manually checking whether the event timestamp matched the recorded time our team had for that Event. In addition, event durations were manually checked to see if the duration calculations were reliable.

Smart Watch

The smartwatch testing consisted of two distinct parts that happened separately before integration testing occurred. The first component was WearOS application testing. This testing consisted of one of the group members wearing the watch while carrying out a normal day, for several days. While the watch was being worn, we verified that the watch was sending heart rate and step count data through both bluetooth connection with a smartphone as well as through a direct WiFi connection from the watch to the router. We also tried testing the smart watch with the resident, but encountered difficulties with the heart rate sensor not working for the resident.

The second component was the Java Spring and backend testing. Testing of the backend consisted of using an application called Postman to send various HTTP requests with JSON data. This was first completed using a local server for rapid development before the same tests were done on the AWS server once a new version of the application was deployed. At each step, we used MySQL Workbench to verify that the data being sent to the server was being properly handled.

Flow Meter Sensor

Testing the flow meter consisted of two parts, one on the software end of the spectrum and the other on the physical end. For testing the software, the flow sensor must be able to record the duration of time the sink is turned and the amount of liquid that came out of the sink during that specified time. Then after a short period of nonusage, the Raspberry Pi should automatically send the data to the sensor hub, which should then, in turn, send the data to the AWS server.

Next up was the hardware testing. For the flow sensors, we had to convert our breadboard circuit onto a circuit board. This way the components would stay in

place. After the soldering was finished the board was checked for shorts and faults to make sure there were no unwanted connections.

Android Application

Testing for the Android application consisted of running the application, and using all of the different views and options to make sure that the app ran successfully, and that the data that was displayed at each point matched that data that was in the database for each of the sensors

4.2 Interface Testing

HTTP Request Testing

Unit Testing for data sent through JSON/HTTP Requests were made using the POSTMAN developer tool. Model JSON strings would be sent to the Server and debugging/further testing would stem from the results of the HTTP response.

4.3 Integration Testing

Smart Plug and Flowmeter

Testing for these components began with the unit testing described in 4.1. Once these tests were successful in the dev environments. The components were moved to a test environment in our client's office. During the office environment testing, the main focus was on reliability. Our team tested to make sure the programs that were collecting data could run for multiple days in a row. In addition, feedback from our client on when the devices and water were used were also used to check the accuracy of timestamps and durations.

Smart Watch

Once the testing described in 4.1 was completed, we moved to test with data collected from the smartwatch. We began by putting the watch on one of our team members and collecting data on campus. We made sure that the data collected was sent to the AWS server and our SQL databases using the tool

MySQL Workbench. While testing, the student would make sure the watch was staying connected, would continue sending data even after a reboot, and not crash when the watch was not connected to the internet.

4.4 Validation and Verification

To validate and verify whether our components were working we made sure all data was being collected when it needed to be and no data was being lost at any moment in time. Also, since all sensors are must be turned with power that means they are all prone to crashing which meant before any installation process all sensors must be able to restart whenever a crash event happens.

During smart plug verification, the plug not only had to successfully send data but had to distinguish between whether the device was on or in use. Therefore, for each appliance plugged into a smart plug a power threshold was established to determine the difference between on and in use events.

5. Project and Risk Management

5.1 Task Decomposition

We did our best to divide the work as evenly as possible between each of the 6 members of our team. Table 1 shows a description of the main tasks each team member completed.

Name	Work Description
Cody Brooks	Smart plug research Smartwatch back end development
Guan Lin	Flow meter development and testing
Josh Lang	Smartwatch Research Smartwatch front end development

Josh Hatton	Smart Plug Event Recording and duration calculations
Justin Somers	Smart Plug Development AWS server management
Mike Ludewig	Android App Development

Table 1: Task Decomposition

5.2 Schedule

Proposed Schedule

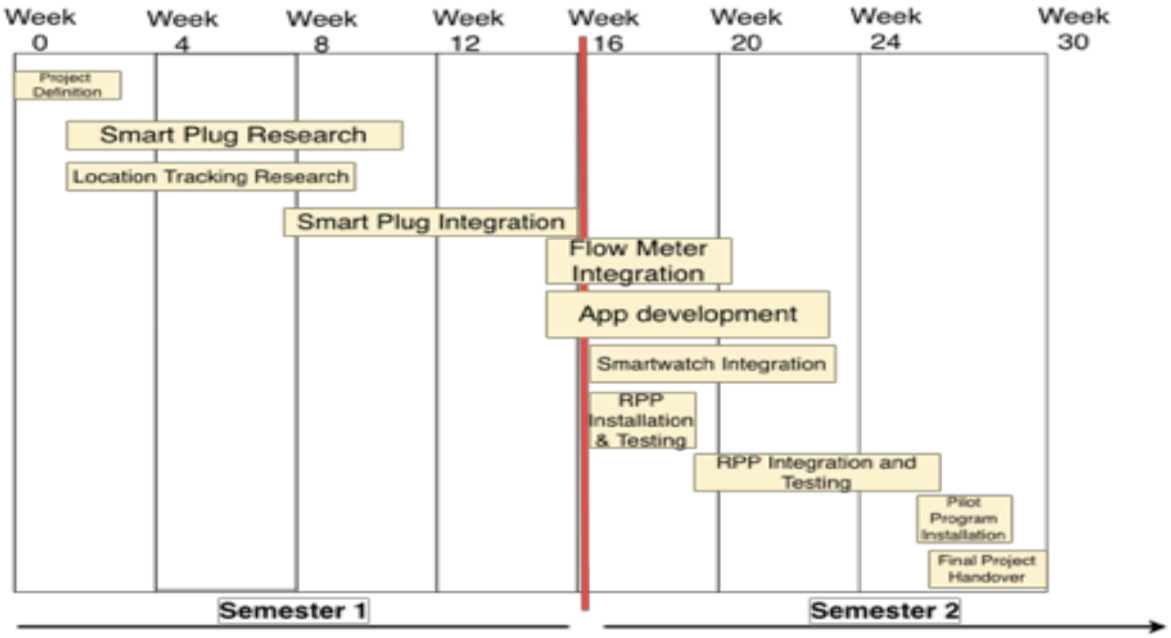

Figure 9: Proposed Schedule

Along with the components discussed above, our initial schedule called for integrating a location tracking system called Red Point Positioning (RPP). This component fell out of scope for our project because our client was not able to acquire the technology in enough time for our team to feasibly integrate and test the system.

Actual Schedule

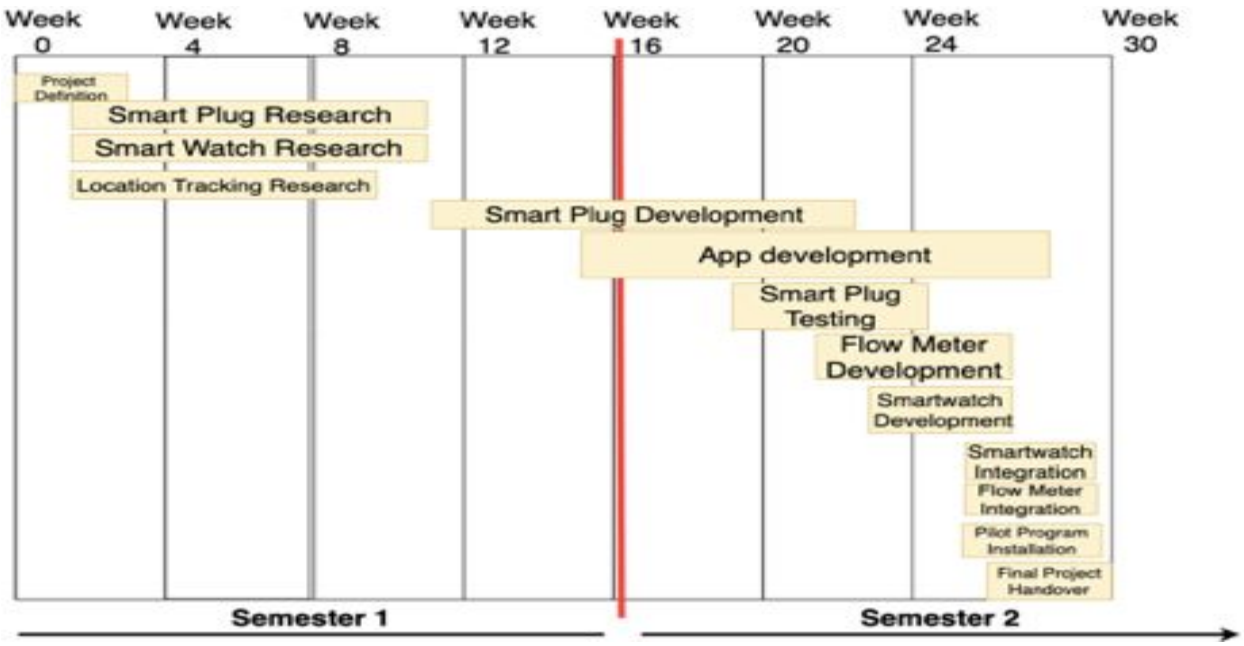

Figure 10: Actual Schedule

In addition to removing Red Point Positioning from the scope, another big change was the amount of time it took our team to develop, test, and integrate the components into the system. Our team had underestimated how long this would take. In addition smartwatch development was pushed back because our client bought the watch later than initially expected.

5.3 Risks and Mitigation

Title: Project Inheritance
Risk: Mitigate
Information: As our project is part of an ongoing development, we inherited a lot of partially done work and had to work with changing requirements due to unfinished work from previous groups coming into our project scope (such as the flow meter).
Mitigation: We established lines of communication with the previous group we were inheriting work from and made sure to allocate work from new requirements out accordingly to make sure they would not stop new work from getting done while making sure the new requirements are met themselves.

Title: Sensors (especially flow meter) being safe for use in a residence
Risk: Mitigate

Information: This applies particularly to the flow meter, as it was handling water that could be used for consumption.
Mitigation: We were able to mitigate this by finding a flow meter that was food grade.

Title: Heart rate sensor on smart watch does not work well for everyone
Risk: Mitigate
Information: When testing the smart watch with the resident, it was found that the heart rate sensor was possibly not going to work with every person as it was not working for the resident.
Mitigation: This will be mitigated by setting up the smart watch app to have a better debug mode for future groups and the residents testing the watch to take advantage of. This debug mode will consist of large text to allow a senior resident to read values such as heart rate values changing and verifying that the watch is connected to the internet. Mitigation will require much more testing and possibly trying different smartwatches to find one that works for the most amount of people

6. Conclusions

6.1 Closing remarks

Overall we were successfully able to implement all parts of the project that were in the final scope of our project. We added smart plugs to the set of sensors and we were successfully able to collect more than 3 weeks of actual data from the senior in our pilot program. We were able to connect the smartwatch, and flow meter to the server and get a few days of data from the pilot program. The Android app was also successful in displaying both the test data and data from the senior and the pilot program.

All of the data that we collected from the sensors will be used in the future to build the behavioral profile for the senior citizen. With this profile we will be able to increase the reliability of the heath predictions and allow us to better determine health issues. This will enable us to provide better care for our senior citizens allowing them to live healthy independent lives longer.

6.2 Future Work

Now that the Sequoia project is at the stage where meaningful and accurate data has been collected by field integrated sensors we are now able to begin analyzing that data for behavioral patterns. As these behavioral patterns are observed and established,

machine learning can be used to analyze new incoming data to determine if it is within the bounds of ordinary behavior, or if it could be symptomatic of an underlying health concern. If it is determined that the abnormal behavior is concerning, family members and healthcare professionals will be notified.

During the course of the project, implementing fall detection using the RightMinder API fell out of scope. This was due to contract negotiations with RightMinder being delayed beyond what was feasible for our team to complete during the remainder of our time to work. Future work with fall detection will include integrating the RightMinder API with the WearOS application that we developed for smartwatch sensor data collection. This smartwatch app development will result in the immediate notification of emergency services and family members when it detects that the senior citizen has fallen.

Passive tracking is performed by observing the data collected from the actions the seniors take in interacting with the things in their home. In addition to this passive tracking, this project will be extended to also include active tracking. Active tracking will work by constantly tracking the location of the senior inside of their homes in order to corroborate and better inform passively tracking data. Our team was originally going to research and create an active tracking solution using Red Point Positioning (RPP), which fell out of scope for our team when our client's contract negotiations with RPP fell through.

Seniors will often have guests come into their home to visit as well as to administer care. When these guests use the different facilities in the home, by default the behavioral profile would assume that those actions were taken by the senior and throw off any analysis that could be given. To mitigate this problem, guests need to be able to be tracked separately from the seniors themselves.

Because of the intentional scalability of this project, additional sensors can and will be added throughout the home. Because our team worked primarily in the kitchen, different sensors will be needed to track data appropriate and useful from other locations within the home. Adding these sensors will include the research, purchase, and integration into the home. It will also include extending the current Java Spring backend implementation for our AWS server to properly handle the new types of data. In addition to backend development, the Android application will need to be extended to appropriately display the additional data.

In order to create a minimum viable product for our client, our team decided to send and handle data in an unencrypted and insecure manner. This was acceptable during

prototyping and did not cause any security concerns since the data was used in a field testing environment as opposed to production. Future teams will need to implement HIPAA compliant security measures in order to secure the privacy of the senior citizens. Such measures would include encryption of data from the sensors.

References:

[1]https://www.fossil.com/us/en/products/fossil-sport-smartwatch-41mm-black-silicone-sku-ftw6024p.html?recid=fossil_product--FTW6024P--7.html?recid=fossil_product--FTW6024P--7.html

[2]. S. DeBruin, B. Ghena, Y.-S. Kuo, and P. Dutta, “PowerBlade: A Low-Profile, True-Power, Plug-Through Energy Meter.” University of Michigan, Ann Arbor .