# IoT Sensor Integration and Back-end Development for Sequoia

Design Document

**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: 2018-12-1 / Version 2

# *Table of Contents*

## List of Figures

## List of Tables

## List of Symbols

## List of Definitions

# 1 Introduction

## 1.1 Acknowledgement

Team 36 Client: Andrew Guillemette
Team 36 Advisor: Daji Qiao

## 1.2 Problem and Project Statement

### Problem Statement

With a large number of baby boomers becoming older there is a large need for systems to monitor the health of our senior citizens. Most senior citizens have habits that they follow every day, and a lot of information can be learned about them from these habits. The goal of the Sequoia project of our clients company is to put sensors in seniors homes to monitor their habits, and be able to tell when a senior might be ill based on changes in habits before they are showing other symptoms.

### Project Statement

Our group has 4 specific goals for this project. First we will add a smart outlet that will track when different appliances are being used to the suite of sensors that is already in place for the system. Second we will use a wearable device such as a smartwatch to track the the location of the senior within the apartment and use the sensors already on the smartwatch to monitor their health in more traditional ways. Third we will install a flow meter in the sink to track the amount of water that is used. Forth we will add a way to track guests in the apartment so their actions don’t affect the behavioral profile of the senior.

## 1.3 Operational Environment

The sensor network will be placed in the senior’s apartment or home to collect data data on their habits.  Due to being placed in the home, the sensors will not be exposed to any harsh weather conditions or environments.  The only physical danger for these sensors is the risk of being unplugged or removed from their power sources, this would compromise the effectiveness of the sensor network.

Another potential hazard is security. Due to the nature of the project, the data collected will need to be uploaded to the cloud to store. This information is quite personal (tracking location, habits, ect.) and therefore must be kept secure to ensure privacy. The sensor network will be placed in a seniors apartment or home in order to collect data on their habits so they will not be subject to any harsh weather conditions. The wearable tracking device will be worn both inside and outside of the residence so it should be able to function in some unfavorable weather conditions such as rain.

## 1.4 Intended Users and Uses

The overall system will include three intended users: the senior that will be monitored, the senior's visitors ( family and friends), and the senior's doctor's or nurses. Currently, there is a network of sensors that have been established. Our project will expand on this network by adding three additional components: the smart outlet, flowmeter, and the wearable tracking device. Our scope will thus be limited to these three components. Due to the limited scope, our only user will be the senior.

The smart outlet will be used to track which appliances are being used and when. It will report the time the event occured, along with specific power statistics. Primarily the current (Amps), voltage (Volts), and power (Watts) will be recorded from the outlet, and then sent to the current sensor network.

The wearable tracking device will be used to track the senior inside and outside of the home. It will also provide fall detection and heart rate monitoring. For outside of the home usage, the tracking device will retrieve GPS information to establish the senior's current location. For inside the home usage, the tracking device will utilize it's built in sensors such as the pedometer, accelerometer, magnetometer to assist in identifying events that occur in the home.

Additionally, a flowmeter will be used to track when water is used in the kitchen sink. This, alongside the smart outlet will be used to identify when a senior is cooking, or getting water. With the flowmeter data collected, the data will be forwarded to the server and will include information such as timestamp of when event occurred, the flowmeter id, and the flowmeter device name. Knowing this information will then provide whether the water was hot or cold, as there will be two flowmeters for both hot and cold water.

## 1.5 Assumptions and Limitations

### Assumptions:

- The senior will not take off the wearable tracking device
- The senior will not plug appliances to non-smart outlets
- The weable will be some kind of watch because the results of a survey of seniors showed they were must open to wearing that type of device.
- The senior will charge the wearable device when necessary
- The senior will not rearrange their apartment

### 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 of the chosen wearable
- The smart watch may need to connect to a phone

## 1.6 Expected End Product and Deliverables

The expected end product of this project is a smart outlet that can transmit data to the hub of the already existing system, a working solution to track the senior outside their apartment, and provide fall detection with an existing wearable device, a working solution to track guests in the apartment to filter out the the guest data from the behavioral profile, and a flow meter to know how much water is being used in a sink.

Along with this we will provide any information and documents needed about the setup of the devices that will be needed to set up the system in more units, and scale up the use of the system.

The smart outlet will be able to send the following data to the sensor hub of the system: Plug ID, time of reading, date of reading, and power data. The plug that we have chosen to use is the TP-Link HS110. The expected delivery date for the prototype is December 1st. A stretch goal for the smart outlet is to allow it to be configured in an app so that the threshold for saying a device is being used can be changed based on what device is plugged into it.

The senior tracking solution will be able to get the location of the senior both when they are inside, and when they are outside, and provide fall detection for the senior. It will also be able to send that information to the existing AWS server. The expected delivery date for an idea for the solution is November 14th and the expected delivery date for the implementation is March 24th

The guest tracking system will be able to to track the location of the guest in the home. It will send the data that is collected to the existing AWS server. The expected delivery date for the idea for the solution is November 28th and the expected delivery date for the implementation is March 24th.

The flow meter will be able to tell how much hot and cold water flows through the pipes that it is connected to. It will send that data to the server, and it will be used to tell when the sink is being used, and for what purpose. The expected delivery date for the implementation is January 27th.

# 2. Specifications and Analysis

## 2.1 Design Specifications

### Functional requirements:

**FR.1:** The smart sensor hub will be notified when power goes through the smart outlet

**FR.2:** The data from the smart outlet shall be sent to the existing AWS server

**FR.3:** The smartwatch will provide an alert when a fall is detected.

**FR.4:** The guest tracking system will track guests with one meter accuracy.

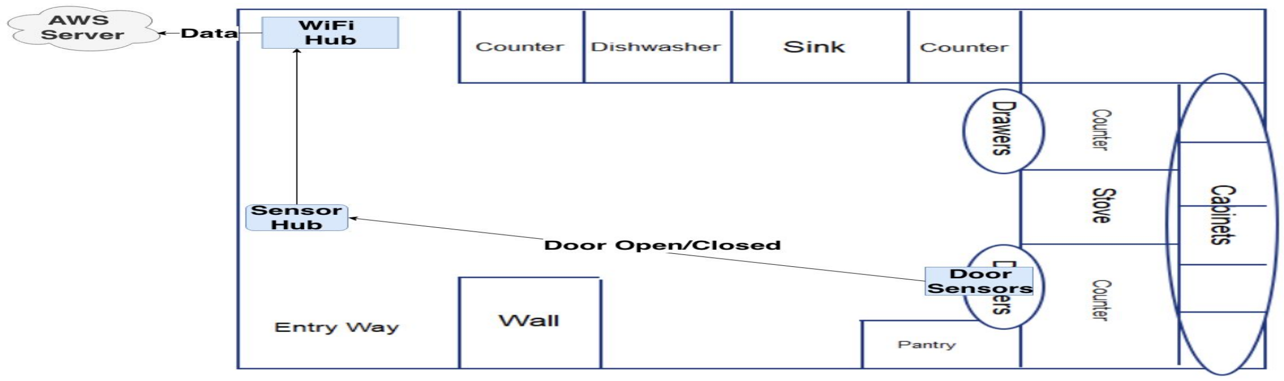
### Non-Functional Requirements:

**NFR.1:** All products used should be easily available to allow for scale up.

**NFR.2:** Number of wires used should be minimized.

## 2.2 Proposed Design

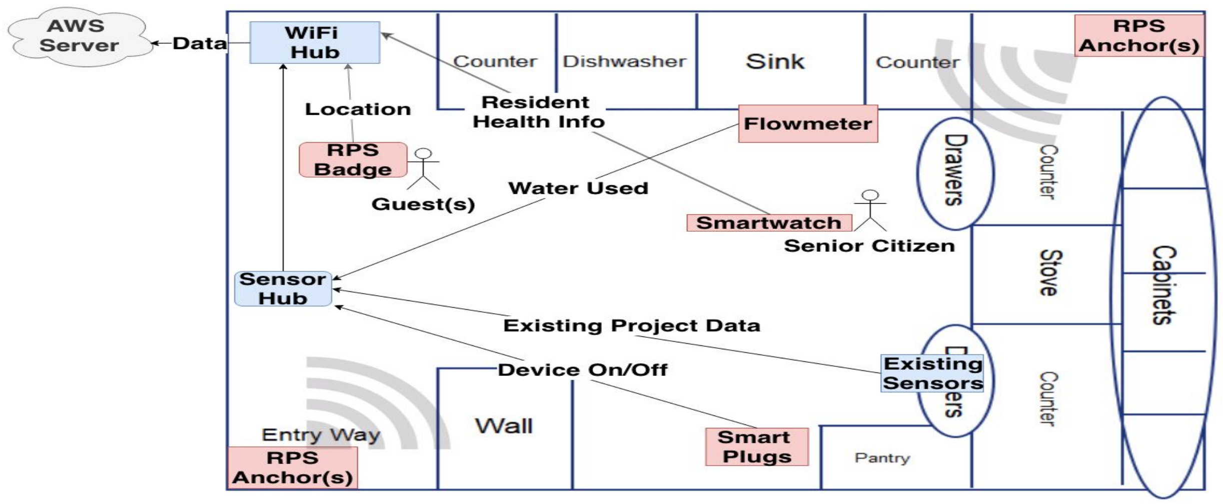
### Existing Design



[Figure 2.1.1] Schematic of existing plan for sensors in pilot program

The existing project uses door sensors that sense when a door is open or closed. This information is then sent to Raspberry Pi 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.

### Extension of Existing Project



[Figure 2.1.2] Schematic for design extension for sensors in the kitchen

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 5 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.

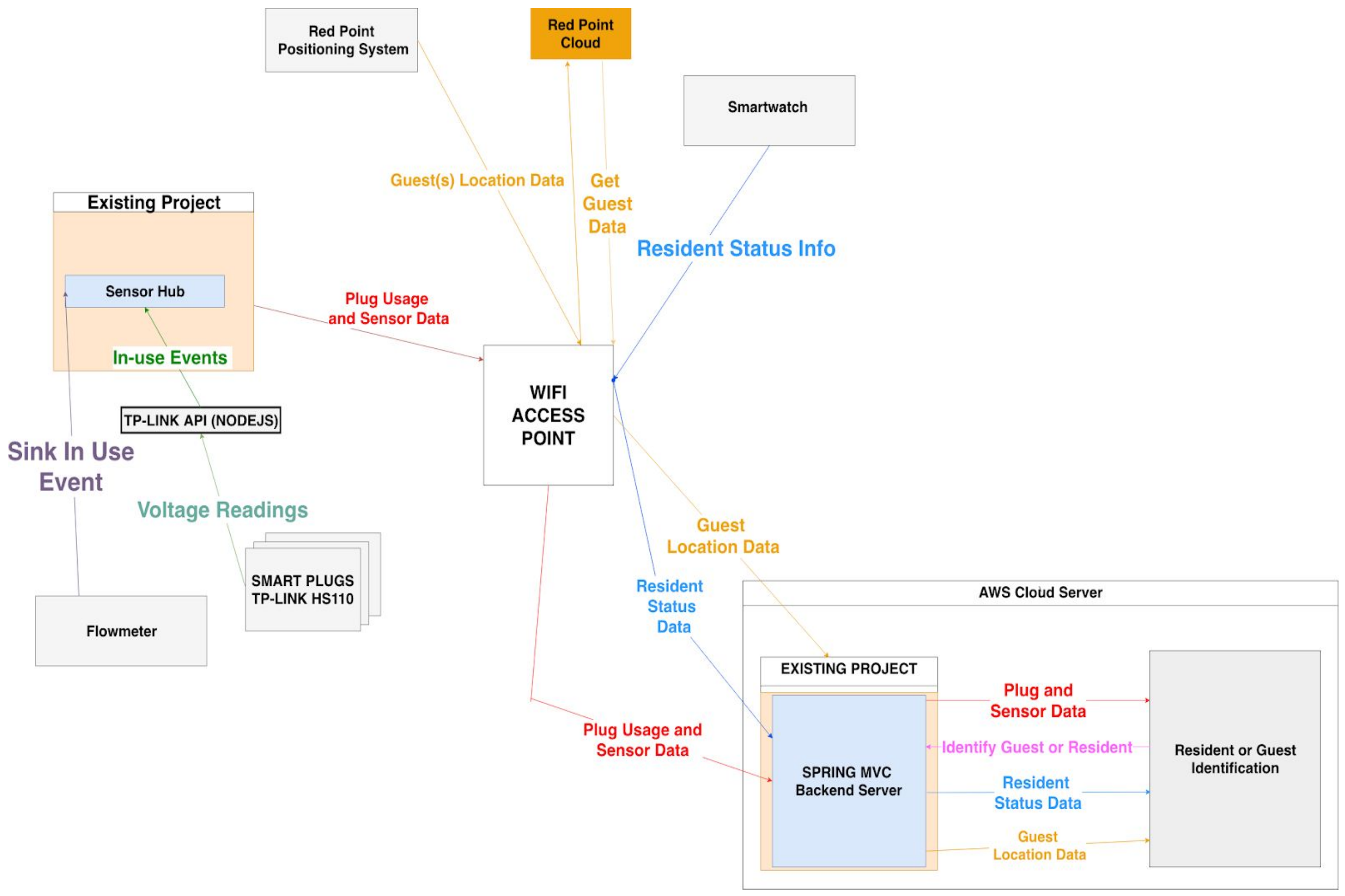
The next extension we are adding is a flowmeter.  The flowmeter is a device to measure water flow through a pipe.  For our project, we are using the flowmeter to determine when the sink is being used.  Knowing when the sink is being used can also help event identification probabilities.

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.  In addition, our team will install a fall detection on the smart watch.  The app will sense a fall and alert authorities unless the user manually clears the fall alert message.

Most of the sensors in this project are not able to determine who caused the event.  This creates a problem when guests come into the resident’s apartment.  In order to remedy this problem, our extension will also include tracking of guests.  This is done through the Red Point Positioning System (RPS).  The system includes anchors that send out ultra-wide band bluetooth signals through the room.  Guests will be required to have a RPS badge on their body.

Application Data Flow  The badge is used to triangulate the position of the guest within the room[1].

## Application Data Flow



[Figure 2.1.3] Application data flow

The flowmeter and TP-Link smart plugs will communicate with the server through the existing sensor hub. The smart plug cannot send data to the server natively, so a Node.JS API will serve as a message sender. The Red Point Positioning system will have its cloud server, so our system will have to interface with that server. Our selected smartwatch will have built-in wifi capabilities, and we will build software to send data to the server.

Once information makes it to the server, the system will use plug and sensor data, and guest location data to determine if an event occured. Guest data location will be sent to a program to determine whether an event is a resident event. Resident status data such as heart rate will also be added to the behavioral profile of the resident.

## 2.3 Design Analysis

Our team has bought a TP-Link HS110 smart plug and are in the process of using the Node.JS API provided from TP-Link. Connection to the smart plug has been made and data can be viewed. Our team is currently meeting with the team who is working on the existing project in order to start getting data into the server. In addition, our team is has been using an open source TP-Link API to be able to send data based on an event driven architecture.

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.

The Red Point Positioning system was chosen for tracking because our client will be using the system for another project and should be able to port the technology to our project easily. Our team had spent some time discussing building an in-house solution, but decided that this would be outside the project scope.

**Strengths:**
Modularity- Each component will function independently of each other
Scalability - Project should be able to scale to more rooms.

**Weaknesses:**
Security- This project relies on wifi communication to send data to the backend server. Our team will need to ensure data is not stolen through the wifi network.
Integration - The project will build on top of an existing project. Our team is assuming that the existing project will be done to specification.
Redundancy - There is no current plan for wifi outages within the environment of the project.

# 3. Testing and Implementation

## 3.1 Interface Specifications

There will be a significant amount of hardware/software interfacing due to the nature of the project. We will have several discrete embedded systems working together to enable features like event identification, effective data collection and safety alerts. Software will need to be developed for each of these systems in order to provide data that is usable from the sensor hub and AWS server.

The wearable device that will track the seniors and the wearables that will track the guests will have custom software running on them in order to process and communicate sensor data from the wearable devices to a server where that data can be analyzed and processed into the resultant predictive health model.

On the AWS server we will be creating software that will monitor and process data from the Wi-Fi connected smart outlet, to which home appliances will be connected. This software will interface with the TP-Link Cloud to retrieve data collected by the smart outlet.

## 3.2 Hardware and software

For this project we will be using a few different hardware components to test the different phases of our project. Most of the components used in development will also be the components we will use in testing. All of the hardware components will be provided by the client, and each group member will have access to the component when needed. On the software side of things we only have a few software components.

The first hardware component is the TP-Link Smart Plug. Even though this was used for development no additional hardware components will be needed to test the plug. For testing we will insert the plug into an outlet and plug some sort of appliance into the plug. The plug should then automatically be sending data to our server. The data that need and can be collected from the plug is the current, power, and voltage being drawn when the appliance is turned on.

The second piece of hardware that will be needed is the smart watch. This component, like the smart plug, was also used during development and will not need additional resources for testing. During testing when the watch is turned on it should automatically be collecting the patients data. The watch will contain many different sensors including, GPS, heart rate monitor, and others, to be used for the collection of data. Some of the collected data should include heart rate, patients movement, daily health statistics, fall detection, and etc.

Lastly, the final hardware component used for testing will be the Red Point Positioning (RPP) hardware. At this point we don't know specifically what the testing components will be needed. However, one component that we know will be needed is the tracking chip provided by the RPP development kit. This kit should include all of the new sensors needed and the tracking device used to track an individual. When all of this is set up we can begin reading the data being collected on whoever is wearing the tracking device inside a certain geofenced location.

The main software component is going to be the Raspbian operating system on the Raspberry Pi's. Raspbian is the main operating system on all Pi's, this means that anything we use that communicates with a Pi needs to be through the operating system.

## 3.3 Functional Testing

During the testing phase our functional requirements will be the most important due to that this is the area where we will see if everything we have implemented is working up to par with our standards. Most of these tests will be integration testing and system testing. For example, the smart plug will need integration testing in order to see how well it is working with the previous system. Then, the smartwatch will need both integration and system testing because it will be a component in the system and finally the Red Point Positioning component will need lots of system testing before it can be fully functional.

For testing the main goal is to make sure the data taken by the plug is collected and sent to our existing AWS server. To make sure this is happening we will need make sure the smart outlet API is correctly implemented into our program. Then, we will need to make sure the smart plug is plugged into an outlet with an appliance plugged into the plug and turned on all while having the server up and running waiting to see what data is showing up. When the data is collected we will need to verify it is correct and it is feasible compared to the appliance we tested.

Next, to test the smart watch we will need a volunteer to wear the watch for a certain period of time. This way the watch has enough time to collect the wearer's health data.

The main goal of this test is to also make sure the data from the watch is collected and sent to the server. When the watch is fully implemented we will have the test user wear the watch until data collection is completed. Then, we will have to make sure all traditional health data is being collected and specific data, like location, is also being tracked and sent to the server. Then, like before we will have to manually verify all collected data is correct.

Finally, for the Red Point Positioning (RPP) testing the main goal is to be able to track the senior to within 1 meter of accuracy in their home. After the RPP system is fully implemented we will need a test user to wear the device and have him/her walk around the apartment to test location tracking. When the tester is moving around the apartment we need to see what locations are being sent to the server and determine if they are the correct.

## 3.4 Non-Functional Testing

Some general non-functional requirements is to minimize the number of wires being used in the project and also all products must have the ability to be scaled up. We will need to make sure we use a minimal amount of wire so the senior citizen will not have to mess around with any wires. Also, while we plan and complete this project we must have scaling as a requirement in the back of our minds.

Having the requirement of minimizing the amount of wires won’t be as difficult as it sounds however we will still have a good amount of wires. Not only will every Raspberry Pi need a power supply but it might also need an ethernet cable. The smart plug, and smart watch won't be needing any wires to be able to function at optimal standards. Finally, the Red Point Positioning system as far as we know will not be needing any wires to function beside the wires needed to power the hub. For this requirement we won’t necessarily be able to test for anything however we will have to make sure if a wire is not needed we don't use it and for wires that we do use we will have to make sure they are hidden from the senior.

Next up is scalability of the project. For this functionality we will have to make sure all products used in the production of this project needs to be future proofed. For example, the smart outlet we used was the TP-Link HS110, this outlet will be able to use the same backend code as the first outlet. This way when we scale the project we won’t have to be rewriting code. When products are being chosen for the project we will need to make sure support for the project is continuous and will not be ending soon.

## 3.5 Process

At this point in the project, testing needs to be completed for each component of our solution separately after research and choosing a platform.

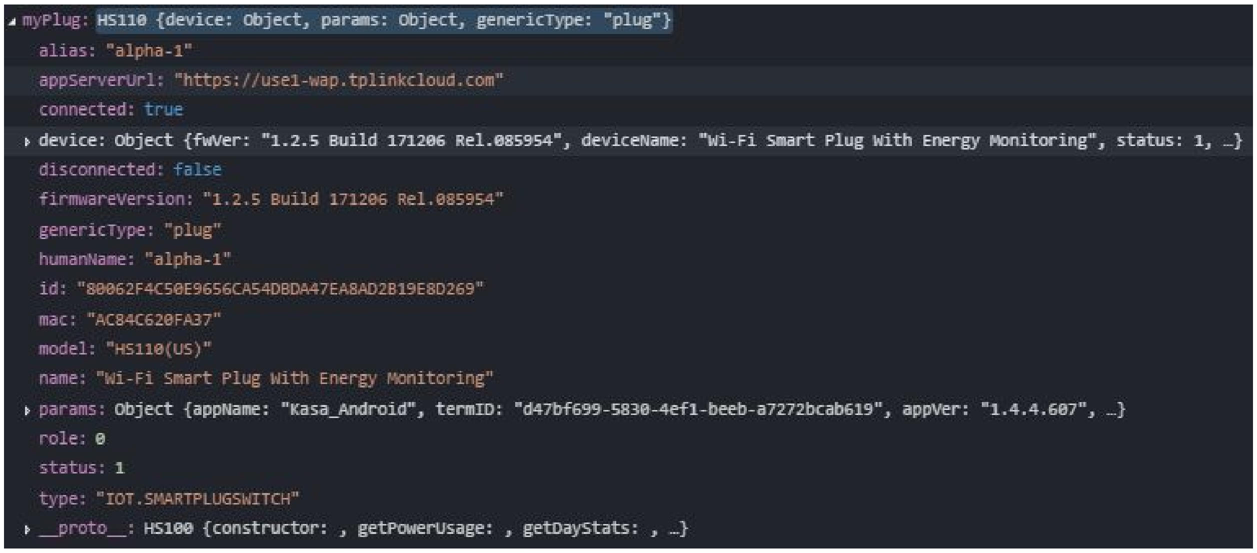
After choosing the TP-Link HS110 as our smart outlet platform, we began testing what kind of data was available with different third party APIs. Once we chose the API, we began testing by writing some simple code to show us what kind of data was available through the API, and how the device responded to different electrical loads. This process showed the type and granularity of the data that could be expected from the device as well as the responsiveness of the NodeJS API and the TP-Link Cloud.

As we researched fall detection in smart watches, we came across an open source Android application for fall detection using a smartphone or tablet. In our initial testing we downloaded the source code and loaded it onto an Android tablet that was checked out from ETG. Once we proved that it could correctly detect falls, we began the process of adapting this application for a WearOS device. This development is current and ongoing. After we prove that we can detect falls using a WearOS device, we will begin testing how that data can be communicated in a meaningful way to the AWS server. After we finish fall detection, we will begin looking into how to best communicate other sensor data from the smart watch. Testing will need to be done to determine how this sensor data can corroborate location tracking and event identification from the Red Point Positioning system.

## 3.6 Results

The results of the initial testing of the TP-Link Smart Outlet showed us what kind of data we could expect to receive from the device.

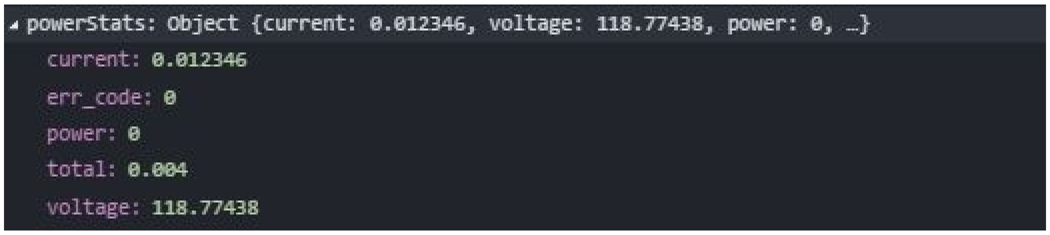
The first test was to collect data about the Node.JS API's object that represented the smart outlet. This allowed us to plan ahead to see what kind of data is possible to collect in the future.



```
▴ myPlug: HS110 {device: Object, params: Object, genericType: "plug"}
    alias: "alpha-1"
    appServerUrl: "https://use1-wap.tplinkcloud.com"
    connected: true
  ▸ device: Object {fwVer: "1.2.5 Build 171206 Rel.085954", deviceName: "Wi-Fi Smart Plug With Energy Monitoring", status: 1, …}
    disconnected: false
    firmwareVersion: "1.2.5 Build 171206 Rel.085954"
    genericType: "plug"
    humanName: "alpha-1"
    id: "80062F4C50E9656CA54DBDA47EA8AD2B19E8D269"
    mac: "AC84C620FA37"
    model: "HS110(US)"
    name: "Wi-Fi Smart Plug With Energy Monitoring"
  ▸ params: Object {appName: "Kasa_Android", termID: "d47bf699-5830-4ef1-beeb-a7272bcab619", appVer: "1.4.4.607", …}
    role: 0
    status: 1
    type: "IOT.SMARTPLUGSWITCH"
  ▸ __proto__: HS100 {constructor: , getPowerUsage: , getDayStats: , …}
```

[Figure 3.6.1] TP-Link Cloud API smart outlet object data

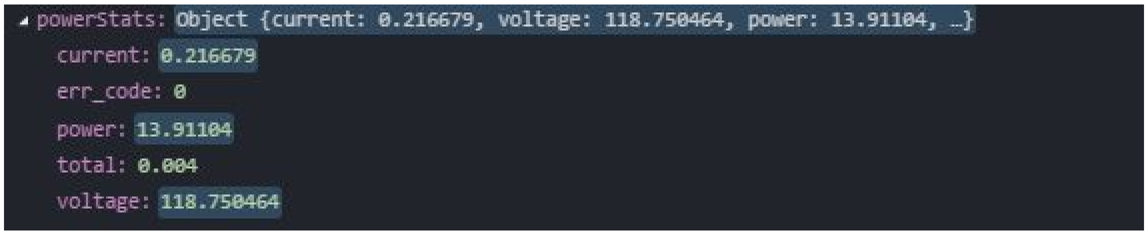
After it was known what methods and data were available through the myPlug object, we conducted a test where there was no electrical load to see what the power monitoring method returned.



```
▴ powerStats: Object {current: 0.012346, voltage: 118.77438, power: 0, …}
    current: 0.012346
    err_code: 0
    power: 0
    total: 0.004
    voltage: 118.77438
```

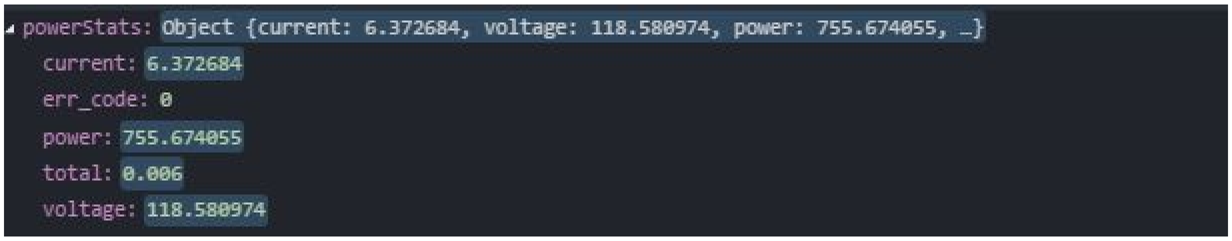
[Figure 3.6.2] Power monitoring method data, no load

Once we had a baseline for what electrical load and power monitoring data looked like, we tested the plug with different electrical loads. The first load was a charging iPad Pro and the second load was a toaster.



```
▴ powerStats: Object {current: 0.216679, voltage: 118.750464, power: 13.91104, …}
    current: 0.216679
    err_code: 0
    power: 13.91104
    total: 0.004
    voltage: 118.750464
```

[Figure 3.6.3] Power monitoring method data, charging iPad Pro



```
▴ powerStats: Object {current: 6.372684, voltage: 118.580974, power: 755.674055, …}
    current: 6.372684
    err_code: 0
    power: 755.674055
    total: 0.006
    voltage: 118.580974
```

[Figure 3.6.4] Power monitoring method data, toaster

The only challenge we faced during this testing was the experimental nature of it. These were tests of discovery rather than proving data or reliability, so we had to adjust the testing process as we moved on and learned more about the device.

# 4 Closing Material

## 4.1 Conclusion

So far our team has researched the major parts of our project: tracking with a smartwatch and getting power data with a smart plug. We have also ordered and received the smart plug that we have decided to use for the project, and have written both skeleton code and test code for getting the data from the smart plug.

Our main goal is to provide health monitoring for senior citizens by placing sensors in their home and building a behavioral profile from the data in order to diagnose medical conditions before symptoms are showing allowing seniors to stay in their homes longer. Our team specifically is working on adding a smart outlet to the sensor network to detect if an appliance is turned on, and add senior and guest tracking to the system. Our best idea for the senior tracking is to use a smartwatch that the senior would wear that can track their location because it is both a plausible solution, and based on a survey of seniors they are more open to wearing a smartwatch than any other wearable that could be used for tracking. Our best idea for guest tracking is to use sub gigahertz tracking to accurately track guests in the home to make sure the guest action does not get added to the behavioral profile of the senior.

## 4.2 References

[1]. “Redpoint RTLS Products & Services,” Redpoint RTLS Products & Services, 2018. [Online]. Available: https://www.redpointpositioning.com/products-services/. [Accessed: 27-Oct-2018].

[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 .

## 4.3 Appendices