ECE/COE 1896

Senior Design

Position-based Automatic Lighting System Conceptual Design

Team #12

Prepared By:

Buka Agbim

Jarod Vickers

Andrew Tran

Jiacong(Peter) Liu

[This page is intentionally blank.]

# Table of Contents

*[When you want to update your table of contents (i.e. prior to turn-in), Right-click on table, select ‘update field’ and select ‘update entire table’]*

[Table of Contents i](#_Toc32861872)

[Table of Figures ii](#_Toc32861873)

[Table of Tables ii](#_Toc32861874)

[1. Introduction 1](#_Toc32861875)

[2. Background 1](#_Toc32861876)

[3. System Requirements 3](#_Toc32861877)

[3.1 Microcontroller 3](#_Toc32861878)

[3.1.1 The microcontroller shall connect to WIFI access points for data 3](#_Toc32861879)

[3.1.2 The microcontroller shall have a mechanism for on/off 3](#_Toc32861880)

[3.1.3 The microcontroller shall have a mechanism for adjusting light brightness 3](#_Toc32861881)

[3.1.4 The microcontroller shall connect to the software system via WIFI 3](#_Toc32861882)

[3.1.5 The microcontroller shall be battery operated 3](#_Toc32861883)

[3.2 IoT Devices 3](#_Toc32861884)

[3.2.1 The Smart Bridge shall receive HTTP requests from the software system 4](#_Toc32861885)

[3.2.2 The Hue Lights shall turn on when the user enters the room, and off when the user 4](#_Toc32861886)

[3.3 Backend/Server System 4](#_Toc32861887)

[3.3.1 The system shall continuously receive data from the microcontroller 4](#_Toc32861888)

[3.3.2 The system shall perform API calls to calculate grid coordinates 4](#_Toc32861889)

[3.3.3 The WIFI access point locations will be known by the system 4](#_Toc32861890)

[3.3.4 The system shall send HTTP requests via WIFI 4](#_Toc32861891)

[3.3.5 The system shall have a method of redefining the building it’s contained within 4](#_Toc32861892)

[3.3.6 The software system will be run remotely, independent from the rest of the system 4](#_Toc32861893)

[4. Design Constraints 5](#_Toc32861894)

[4.1 The system shall not interrupt normal daily usage of home light 5](#_Toc32861895)

[4.2 WIFI 5](#_Toc32861896)

[4.3 Philips Hue API 5](#_Toc32861897)

[4.4 Raspberry Pi Zero and Power constraints 5](#_Toc32861898)

[5. Conceptual Design 5](#_Toc32861899)

[5.1 System Overview 5](#_Toc32861900)

[5.2 Raspberry Pi Software 6](#_Toc32861901)

[5.2.1 Acquiring and Send Router Information to the Server 6](#_Toc32861902)

[5.2.2 Reading User Inputs From GPIO 7](#_Toc32861903)

[5.3 Backend/Server Software 7](#_Toc32861904)

[5.3.1 Receiving Information from Raspberry Pi 7](#_Toc32861905)

[5.3.2 Indoor Position Calculation Algorithm 7](#_Toc32861906)

[5.3.3 Data Structure Design 9](#_Toc32861907)

[5.4 Philips Hue API Interaction 11](#_Toc32861908)

[5.4.1 HTTP Request from Server 11](#_Toc32861909)

[6. System Test and Verification 11](#_Toc32861910)

[6.1 Key Performance Criteria 12](#_Toc32861911)

[6.1.1 The coordinate generated from collected data should be within 1 meter of precision of the user’s actual location. 12](#_Toc32861912)

[6.1.2 Smart Hue lights should respond to user input on the microcontroller within 2 seconds. 12](#_Toc32861913)

[6.2 Requirements Tests 12](#_Toc32861914)

[6.2.1 Microcontroller 12](#_Toc32861915)

[6.2.2 IoT Devices 13](#_Toc32861916)

[6.2.3 Backend/Server System 13](#_Toc32861917)

[6.3 Miscellaneous Tests 14](#_Toc32861918)

[7. Team 14](#_Toc32861919)

[7.1 Andrew 14](#_Toc32861920)

[7.2 Jarod 14](#_Toc32861921)

[7.3 Chiebuka 14](#_Toc32861922)

[7.4 Peter 14](#_Toc32861923)

[14](#_Toc32861924)

[8. Schedule 15](#_Toc32861925)

[8.1 Minimum Standard for Project Completion 15](#_Toc32861926)

[8.2 Final Demonstration 16](#_Toc32861927)

[To update the Table of Contents, Table of Figures, or Table of Tables, click on any text within the table. Press the F9 key on your keyboard (or right click and select “Update Field”). Then select “Update entire table” from the pop-up window and click “OK”. If you use the formatting styles as directed below, your tables will fill automatically and have the correct page numbers.]

# Table of Figures

[Figure 1: Concept Car. 3](bookmark://_Toc361750686#_Toc361750686)

[Figure 2: Example Risk Management Assessment Graph (descriptions of individual risks not shown) 5](bookmark://_Toc361750687#_Toc361750687)

# Table of Tables

[Table 1: Example Table for ECE/COE 1896 Final Report. 4](#_Toc361750719)

[If necessary, insert a blank page after your Table of Tables to make the front matter end with an even page. You should be able to print your final report double-sided and have the page numbers appear as they would in a book.]

[The following sections must be included in your final report. You may add sections or subsections to suit the needs of your project, but do not omit any of the sections provided here, unless instructed to do so. Follow the instructions and examples in each section to build your report. If you insert material from other sources, make sure the format is consistent with this document’s original format. If you have trouble with the formatting, please consult your course instructor.]

# Introduction

The Position-based Automatic Lighting System (PALS) is a real-time smart home lighting system that uses the position of the user to control smart light bulbs. PALS can be used to turn on lights in a specific room as soon as the user enters it without needing to physically turn on a switch. There are currently many interfaces that already exist that can also provide similar lighting systems, but there are issues with them in terms of how and when a user can interact with them. For example, many interfaces require the user to interact with a smartphone app or use voice commands. While these interfaces work in ideal scenarios, there are some scenarios where these interactions are difficult or impossible to perform.

PALS, on the other hand, does not require the user to directly interact with an interface as it determines the user’s position to determine which lights to control and whether they need to be turned on or off. Position tracking runs constantly in the background on a device carried by the user as well as a server on the same WIFI network. The device carried by the user will constantly send data to the server about the signal quality between it and at least three different access points distributed across a home. The server can use the data sent from the device to determine the position of the user and turn lights on and off by sending commands directly to a smart bridge. By using position instead of commands sent from a user, smart lights can be controlled without a user needing to interact with any device making PALS more accessible and easier to use.

# Background

As the adoption of home IoT devices has increased, there have already been different interfaces designed to control these devices. While these interfaces work well in ideal scenarios, there are some scenarios where they become difficult or impossible to use. PALS will not replace any of these interfaces, but it will address some of the problems that they face. The problems these interfaces have mainly relate to three different aspects: availability (in what scenarios they do and do not work in), accessibility (who is able to effectively use them), and convenience (how a user interacts with them).

One of the most popular interfaces currently is smart speakers, such as Amazon Alexa, which can be used to control most brands of smart lights by listening for voice commands spoken by any user. Smart speakers can work well when they are in the same room as the user but can be rendered useless if the room is too noisy or if the user is in another room where the speaker is too far away to detect voice commands. These issues severely impact the availability of smart speakers because the scenarios that prevent them from functioning as intended are not uncommon. Another issue with smart speakers is their ability to understand different accents. In a study performed by Globalme, it was found that “people who spoke Spanish as a first language, for instance, were understood 6 percent less often than people who grew up around California or Washington, where the tech giants are based” 1. These inaccuracies can require users to repeat voice commands multiple times due to the speaker not understanding certain accents. This can make using these devices frustrating because this is a problem that users cannot fix themselves causing IoT devices to be less popular to people that do not speak in a “typical” American accent. This raises both accessibility and convenience issues especially when speech disabilities are considered as it reduces the number of people that can use voice commands effectively.

The other popular interface for controlling smart lights is through applications designed by the smart light brands themselves. These apps can be installed on a smartphone which is convenient because it gives users access to any of their lights from anywhere that has access to WIFI. However, like the smart speakers, there are some scenarios where these apps are not very helpful. One of the most inconvenient aspect of smartphone apps is that they require the use of your hands. However, there are some scenarios where the user may not be able to interact with app like when their hands are full. There is also often a delay from the time between a user presses the button on their phone and when their lights actually turn on. Users have reported delays anywhere from 3-8 seconds2 which does not include the time taken to unlock their smartphones and open the app. Due to this delay, apps are mostly used to control lights before users have even entered the room and not necessarily right when they need to be turned on or off because by the time the user enter a room, it would be faster to physically turn on a light switch instead of using the app on their phone. Since apps need the use of hands as well as the having relatively long delays, they can be very inconvenient and not worth it to use.

The main problem with both of these interfaces is that they require the user to directly interact with them in a way that may not always be possible in all scenarios. PALS solves this problem by using the user’s position to control smart lights rather than the user directly interacting with an interface. By using position to control smart lights, most if not all the problems mentioned above are addressed. In terms of availability, the only problem that could occur is if the user is out of range of one or more of the three access points making it impossible to accurately determine their position. However, with current technology and the range of WIFI signals (approximately 150 feet)3 smaller homes and apartments should be covered with only three access points and more access points can be installed for larger homes. PALS also provides more accessibility and convenience by not requiring a specific method of interaction between the user and the interface. The only requirement is that the user carries a WIFI connected device with them.

To implement this system, user position will be calculated constantly in the background by determining the distance between the user and three different points within a home and using those distances to triangulate the user’s position. PALS will use WIFI signals between a device carried by the user and multiple access points to determine the three distances needed to triangulate the user’s position. Certain qualities of WIFI signals such as signal strength and attenuation can be used to estimate the distance between two connected devices.

# System Requirements

The following section outlines the requirements of PALS. The different main components of the system will be listed, as well as their main functional requirements. The system consists of a Microcontroller, a server/backend system, and a system of IoT devices.

## Microcontroller

The microcontroller used in our base development is the Raspberry Pi Zero. This microcontroller was used as our mobile location tracked device.

### The microcontroller shall connect to WIFI access points for data

The microcontroller will connect to different WIFI access points and collect data from each connection. This data will be network connection information. This data will be sent to the backend software for processing.

### The microcontroller shall have a mechanism for on/off

A user will be able to manually turn their IoT devices off when desired with a simple mechanism.

### The microcontroller shall have a mechanism for adjusting light brightness

A user will be able to adjust how soft and how bright the lights they are connected to are.

### The microcontroller shall connect to the software system via WIFI

The microcontroller will communicate with the rest of the system via WIFI. This allows the microcontroller to be portable.

### The microcontroller shall be battery operated

A battery-operated microcontroller allows the controller to be mobile, which is necessary for its position to be tracked within a building.

## IoT Devices

The IoT devices used in conjunction with this product are the Philips Smart Hue Lights. The Smart Hue Lights have an open API and are interactive through a Philips Smart Bridge and HTTP requests.

### The Smart Bridge shall receive HTTP requests from the software system

These HTTP requests are calls to the Phillips Smart Hue API. The bridge will receive these commands and take proper action on the lights.

### The Hue Lights shall turn on when the user enters the room, and off when the user

When the user enters or exits the room, a physical change in state should be seen in the lights. Note this will not occur if the system is set to “Off” on the microcontroller.

## Backend/Server System

A backend and server system will exist to perform most of the computations for the system. This includes receiving and processing data and creating proper requests for IoT devices.

### The system shall continuously receive data from the microcontroller

The system will collect WIFI information from the microcontroller for further processing. This will be done constantly to ensure we don’t lost precision in location. This will be done via WIFI.

### The system shall perform API calls to calculate grid coordinates

The system will use collected data from the microcontroller to generate coordinates. The calculation will be comprised of 2 steps. Distances from each access point will be calculated, then a coordinate will be triangulated based off these distances.

### The WIFI access point locations will be known by the system

The coordinates of the access points will be known to the system to streamline calculation of coordinates.

### The system shall send HTTP requests via WIFI

The system must send HTTP requests to the Smart Bridge to interact with IoT devices. This will be done via WIFI.

### The system shall have a method of redefining the building it’s contained within

The system will have a method of recreating coordinates for key features. This will allow the system to be robust and easily be translated to a new location.

### The software system will be run remotely, independent from the rest of the system

The software backend will be hosted on a computer or server independent of other components to ensure upkeep time.

# Design Constraints

## The system shall not interrupt normal daily usage of home light

The system shall at no point interfere with users’ physical interaction with lights through on/off switch. All the light should function as normal regardless the system being on or off.

## WIFI

We have decided to use WIFI as our primary communication method between different components of our project. We will have to account for the network delays, signal range, as well as to determine which network protocol to use for our project.

## Philips Hue API

We are using the Philips Hue series LEDs, which means we can only use Philips’ public smart home API and software standards when controlling the LEDs.

## Raspberry Pi Zero and Power constraints

For better user experience the device for user to hold on for collecting router data should be small and light weight. This means the client software must support Linux and Raspberry Pi in a low power consumption manner.

# Conceptual Design

There are three main components to our project – the raspberry pi software, backend server software, and Philips Hue API interaction. Details regarding to each component will be discussed in following sub-sections.

## System Overview

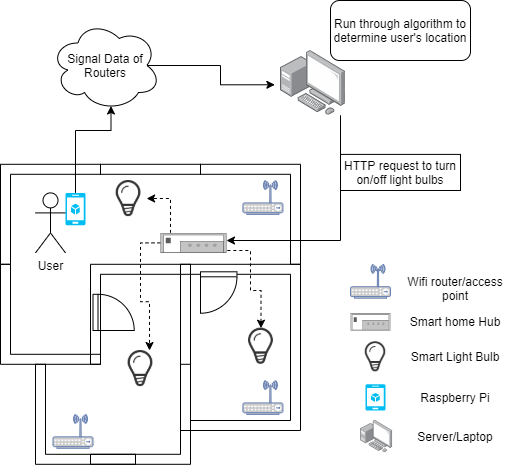


Figure 1: System Interaction Diagram.

The general system interaction diagram is shown in Figure 1. First, the Raspberry Pi Zero carried by the user will detect and collect signal information of all the WIFI routers we set up in the apartment. These data will then be sent to the backend server, where the data will be processed through the algorithm to calculate user’s estimated location. The server will record user’s current room location, therefore once a change in room is detected, the server will send out HTTP requests to turn on and off the corresponding LEDs in the rooms involved. The HTTP requests will be delivered to the Philips’ Smart Home Bridge, which will send out ZigBee signals to targeted LEDs.

## Raspberry Pi Software

### Acquiring and Send Router Information to the Server

Since the Raspberry Pi Zero will be running on Linux, we could easily get the connectivity information from each available WIFI router because Linux has countless command line tools. Figure 2 shows one example as how we can acquire such information. The command “nmcli dev wifi” will list all the available WIFI and their signal strength. By redirecting the result, we can put all the information to a .txt file, which can then be sent to the server. There are other commands to show more detailed information including noise level and attenuation, but the general idea is the same.

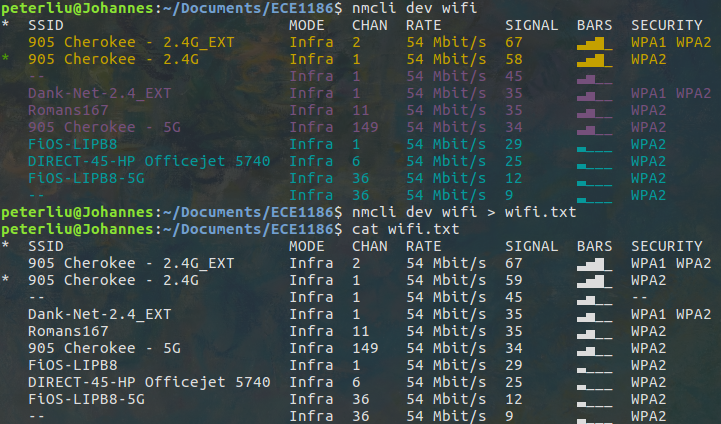


Figure 2: Example Linux Command

### Reading User Inputs From GPIO

The Raspberry Pi Zero will also have other control functionalities such as adjusting the brightness of the light in current room. This can be done by connecting a potentiometer to an input voltage from the board and a GPIO pin. The GPIO pin will read the input voltage through the potentiometer. Input voltage will change based on the position of the nob and we can convert it to a percentage by dividing the maximum input. This information will also be sent to the server, where the brightness setting will be added to the HTTP requests for the LEDs.

## Backend/Server Software

### Receiving Information from Raspberry Pi

The server-client communication between the Raspberry Pi and the laptop/server will be implemented through existing network socket library in either C or Python. Once the router information file is generated on the Raspberry Pi, the server will start receiving the file and starts extracting signal information of targeted router from the file. The file will be send using the UDP protocol to reduce delay and transmission time.

### Indoor Position Calculation Algorithm

The indoor position calculation algorithm consists of 2 primary computations. The first of these computations is the calculation of the distance from the microcontroller to each WIFI access point. Each distance is calculated individually from data received from each access point. The second computation is the triangulation of distances to calculate a specific coordinate.

#### Distance Algorithm

To determine the distance from the Raspberry Pi and the router we will be using the common path loss formula. The server will extract the signal strength, noise, and power information of each router from the file sent by the Raspberry Pi and use this information to calculate the approximate distance to each router.

#### Triangulation Algorithm

The second algorithm makes use of circle theory to determine the location of a point given the distances from three other known points. In this scenario, the three known locations are the locations of the WIFI access points, which according to REQ 3.3.3, are known to the software system. The access point locations are used as centers of circles, with the distance to microcontroller serving as the radius for each circle. The coordinate of the microcontroller will be the intersection of these 3 circles, as seen in Figure 3.

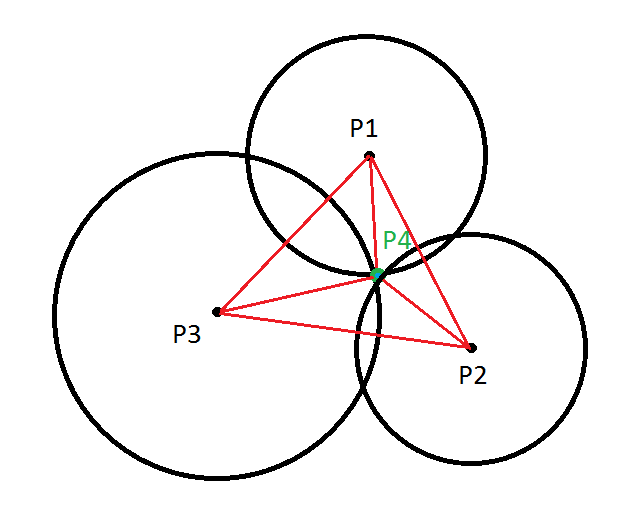
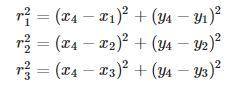
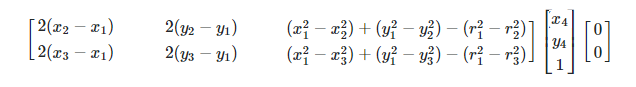


Figure 3: Triangulation of point from circles and known distances

The calculation of x and y coordinates of P4 can be calculated from the following set of equations, where (x1, y1), (x2, y2), and (x3, y3) are the locations of each access point, (x4, y4) is the location of the microcontroller, and r1, r2, and r3 are the respective distances:



From these equations, linear relationships can be created by first expanding these equations via FOIL and setting them equal to 0. The next step is to subtract equation 2 from equation 1 and subtract equation 3 from equation 1 to get equations 4 and 5 respectively. Due to the nature of these equations, all nonlinear unknown terms (x42 and y42) are eliminated and thus you have a linear system of 2 equations with 2 unknowns:



From this point, there solving for x4 and y4 can be done using tools such as Numpy or any library that can solve linear equations.

### Data Structure Design

The Data structures in the background must be designed in a way such that we can easily store all the information needed, as well as provide necessary functionality to the system. As such, we need a structure of the overall house to store the information of rooms within the house, as well as locations of objects of interest. These objects of interest include the location of WIFI access points, as well as the location (or containing room) of an IoT device. All the following concepts of data structures assume a grid layout, with x and y values, as the underlying idea behind location. This imaginary grid has (0,0) at its bottom left corner, with x ascending to the right and y ascending upwards. All location values are stored as a tuple (x, y). This idea can be seen in Figure 4.



Figure 4: System grid layout

#### House Design

The House data structure will be a simple structure that contains the rooms, which are stored in a list, and is able to perform simple calculations given input data. These calculations are get\_room which returns the room the given coordinates exist within, and get\_closest\_room\_center, which returns the room whose center the given x, y coordinates are closest to. Both methods could be useful in determining HTTP calls to the Smart Bridge device. The class diagram for the House can be seen in Figure 5.

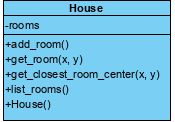


Figure 5: House class diagram

There are several different concepts for room structure that will allow for different advantages and disadvantages on the backend, all of which assume relatively rectangular rooms.

#### Room Concept 1

The first and desired room concept idea stores the center of the room, as well as the length and width of the room. This structure would allow for easy creation of rooms, as there are only 3 input values, 1 of which is a coordinate. Also, storing length and width allows us to maintain our assumption of rectangle rooms with no overlap between rooms, as we can calculate exact coordinates for the bounds of a room. The resulting class diagram and physical structure can be seen in Figure 6.

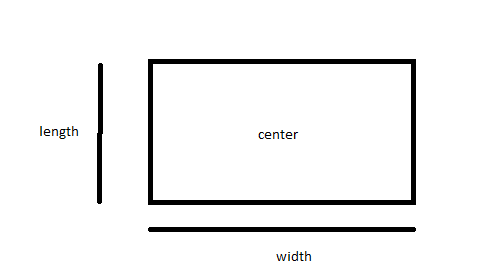
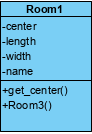


Figure 6: Room Concept 1 class diagram and visualization

#### Room Concept 2

The second room concept idea stores the center of the room, as well as the rough radius we would consider. This would be the easiest to implement, as there are only 2 data fields for the structure. It would also prove to be a simple calculation for a room’s area. This however would break our assumption of rectangle rooms and lead us to need to approximate for room location, as room areas would overlap. This concept is more suitable for large scale, approximate locations.

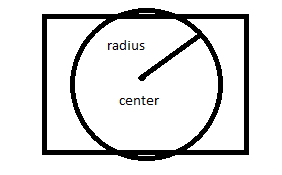
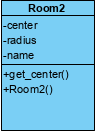


Figure 7: Room Concept 2 class diagram and visualization

#### Room Concept 3

The third room concept idea stores the 4 corners of the room. This structure would contain all the required information for determining room bounds and passes our assumption that rooms are rectangles. However, this design would be harder to instantiate, as 4 coordinates are needed rather than just 1 for the other concepts. This thus would possibly invalidate REQ 3.3.5, as it would make this requirement a much more tedious process.

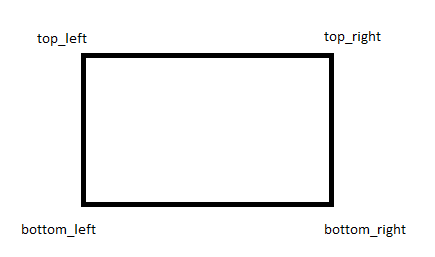
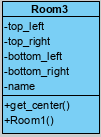


Figure 8: Room Concept 3 class diagram and visualization

## Philips Hue API Interaction

### HTTP Request from Server

Once the server determines which room the user is in it will send the appropriate HTTP requests directly to the bridge. This includes turning on lights in the room the user has just entered as well as turning off lights in the room the user has just left. Additionally, the user can adjust the brightness of the lights in the room they are currently in. The PUT request will be formatted following the Philips Hue Light API standard which includes the ID of the room the lights are in and any attributes such as state (on/off) and brightness.

# System Test and Verification

Include preliminary test plan. Specify key performance criteria that are measurable and how you will test these criteria (specify at least 3, you can include more if necessary). Saying that the prototype simply ‘works’ or ‘doesn’t work’ won’t be good enough. Your test plan is how you verify all of the requirements. List the expected/desirable test outcomes and materials/equipment that you made need to carry out your plan.

## Key Performance Criteria

### The coordinate generated from collected data should be within 1 meter of precision of the user’s actual location.

**Smart Hue lights should turn on within 2 seconds of a user entering a room.**

### Smart Hue lights should respond to user input on the microcontroller within 2 seconds.

## Requirements Tests

### Microcontroller

#### Unit testing REQ 3.1.1

The microcontroller’s connection to WIFI will be unit tested. Tests will attempt to connect to each WIFI access point and will attempt to fetch required data from said access points.

#### Unit Testing REQ 3.1.2

The microcontroller’s on/off switch will be unit tested. Tests will attempt to switch lights on/off. The API function will be called to ensure proper output.

#### Integration Testing REQ 3.1.2

The microcontroller’s on/off switch will be tested interactively within the scope of the entire system. Upon button press, Smart Hue lights should change states appropriately (from on to off, or off to on).

#### Unit Testing REQ 3.1.3

The microcontroller’s brightness editing will be unit tested. Tests will attempt to change the input to the method and will verify proper expected numerical output.

#### Integration Testing REQ 3.1.3

The microcontroller’s brightness editing will be tested interactively within the scope of the entire system. Upon editing of the mechanism, Smart Hue lights will change brightness states.

#### Unit Testing REQ 3.1.4

The microcontroller’s WIFI connection will be unit tested. The tests will attempt to send data from the microcontroller to the server and will verify that proper information has been transmitted.

#### Integration Testing REQ 3.1.5

The microcontroller’s portability will be tested interactively. The microcontroller will be connected to a battery pack and moved around to ensure the microcontroller can operate without an immobile power source.

### IoT Devices

#### Unit Testing REQ 3.2.1

The Smart Bridge HTTP request will be unit tested. Different HTTP requests will be sent to the Smart Bridge via internet, and HTTP response codes will be checked for correctness.

#### Interactive Testing REQ 3.2.2

The smart bridge lights will be tested interactively for powering on and off when the user enters the room. This will be accomplished by walking into and out of the room, and recording whether the light turned on, as well as the response time.

### Backend/Server System

#### Unit Testing REQ 3.3.1

The servers receiving of data will be unit tested. The tests will attempt to send data from the microcontroller to the server and will verify that proper information has been transmitted. This can be combined with Test 6.2.1.4.

#### Unit Testing REQ 3.3.2

The grid coordinate calculation will be unit tested. This will be done by using arbitrary data and values and validating the calculated grid coordinate is correct. This will also include unit testing for our distance algorithm.

#### Unit Testing REQ 3.3.3 & 3.3.5

Unit tests will be created to test the structure on functionality of backend data structures for the system. These tests ensure data structures can be created and manipulated with ease. The data structures included are the room and house layout, which contain WIFI access point information.

#### Unit Testing REQ 3.3.4

Different HTTP requests will be sent to the Smart Bridge via internet, and HTTP response codes will be checked for correctness. This can be combined with Test 6.2.2.1.

#### Integration Testing REQ 3.3.6

Once all components are in place, the system will be tested. This will include the microcontroller, IoT devices, and a laptop/server of some kind performing calculations. Different edge cases must be evaluated, such as overlapping regions, lights in a room where no access point is located, and effectiveness of walls are examples of some scenarios that will be tested.

# Team

## Andrew Tran

Andrew has software development and testing experience so his main role will be to work on software development with the Phillips Hue API. He will work to setup the connection between the device and smart bridge. He will also create and perform test cases for the prototype of the device.

## Jarod Vickers

Jarod similarly has software development and software testing experience, but his focus will be on the system support software development. He will create some of the necessary data structures for the location tracking software. He will work on programming the Raspberry Pi being used in the device as well as develop API calls and creating a system to send proper controls to smart devices through the lightbulb smart bridge.

## Chiebuka Agbim

Chiebuka has software development and big data engineering experience. Chiebuka will focus on rudimentary algorithm design for the location tracking system and on general software development for the project. He will also help with PCB synthesis and the device hardware design. He will also manage the budget for the duration of the project.

## Jiacong (Peter) Liu

Peter has software development and embedded system programing experience. Peter will take the lead on overall PCB and hardware design, as well as creating low level device driver software for the project. As far as software development, he will have the responsibility of encoding the client-server connection.

# Schedule

|  |  |
| --- | --- |
| Week of February 19th | * Submit initial order for parts:   - (Raspberry Pi Zero W) (Philips Hue Smart Bridge) (Philips Hue LED Smart Bulbs)   * Finalize position tracking algorithm used for the project. Clarify the details of the triangulation algorithm being used. |
| Week of February 26th | * Upon retrieval of the parts, begin working on PCB design and synthesis and Raspberry Pi programming. * Begin server development. |
| Week of March 4th | * Finalize rudimentary server software capabilities. Ensure that majority of the workload is on the server side, not the device side. * Refine PCB design, begin working on establishing connection between device and sever. |
| Week of March 18th | * Get PCB printed and begin programming functionality. * Begin product testing, using minimal standard for project completion. * Review parts and make any necessary orders (second round of orders) |
| Week of March 25th | * Expand project capabilities to include more rooms, access points, and more light functionality (dimming, etc.) |
| Week of April 1st | * Clean up any loose ends in the server/device communication, continue to reduce the margin of error in locating the device |
| Week of April 8th | * Finalize demonstration for design expo, refine final product |
| Week of April 15th | * April 16th: Design Expo - Showcase video and live demonstration of location tracking system. Video should include detailed description of project and demonstrate the final product working in a real home setting. |

Table 2: Project Semester Schedule

## Minimum Standard for Project Completion

As a minimum, we should have the handheld device fully built and the connection between it and a server should be established. We should be able to have the server accurately track the location of the device in at least three different rooms and accurately toggle the lights in each room based of the device's location. The server should be able to send the appropriate HTTP requests directly to the Philips Hue Bridge.

## Final Demonstration

For our final presentation, we plan to include significantly more functionality to the light tracking system. We plan to include capabilities for the device to automatically turn off lights when leaving a room, have multiple programmable modes that allow the user to change the lights behavior such as leaving the light on in a room when leaving and the ability to set a timer on the lights. Having lighting modes such as a flash setting and a light dimming capability is also something we would like to include. All of this will be included in the demonstration video given that it will be difficult to showcase this in a design expo setting.

# References

* 1 <https://www.washingtonpost.com/graphics/2018/business/alexa-does-not-understand-your-accent/>
* 2 <https://community.smartthings.com/t/smartlighting-delay-when-turning-on/57631>
* 3 <https://openweb.co.za/how-far-will-your-wifi-signal-reach/>
* <https://developers.meethue.com/develop/hue-api/lights-api/>
* <https://developers.meethue.com/develop/application-design-guidance/hue-bridge-discovery/>
* <https://pypi.org/project/UPnPy/>
* <https://ieeexplore-ieee-org.pitt.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=6802701>