

# Indoor Navigation System with Beacons

Fall 2014 Group 14

Andre Compagno  
Josh Facchinello  
Jonathan Mejias  
Pedro Perez

May 1, 2015

## Table of Contents

|                                                       |         |
|-------------------------------------------------------|---------|
| 1.0 Executive Summary                                 | Page 1  |
| 2.0 Project Description                               |         |
| 2.1 Project Motivation and Goals                      | Page 2  |
| 2.2 Objectives                                        | Page 3  |
| 2.3 Project Requirements and Specifications           | Page 4  |
| 3.0 Research related to Project Definition            |         |
| 3.1 Existing Similar Projects and Products            |         |
| 3.1.1 Estimote Beacon                                 | Page 5  |
| 3.1.2 SPREO Navigation                                | Page 6  |
| 3.1.3 Motorola Bluetooth Smart Beacons                | Page 7  |
| 3.1.4 Indoo.rs Solutions                              | Page 8  |
| 3.1.5 Microsoft Location Provider Project             | Page 9  |
| 3.2 Relevant Technologies                             |         |
| 3.2.1 Solar Power in Microelectronics                 | Page 10 |
| 3.2.2 Bluetooth Low Energy                            | Page 11 |
| 3.2.3 Apple iBeacon Protocol                          | Page 12 |
| 3.2.4 Google Glass                                    | Page 13 |
| 4.0 Beacon Hardware Details                           |         |
| 4.1 Initial Design Architectures and Related Diagrams | Page 14 |
| 4.2 Power Design Specs                                |         |
| 4.2.1 General Power Supply Requirements               | Page 15 |
| 4.2.2 Current Power Supply Implementations            | Page 16 |
| 4.2.3 Solar Power Research and Design                 | Page 16 |
| 4.2.4 Different Battery Technologies                  | Page 17 |
| 4.2.5 Battery Safety                                  | Page 19 |
| 4.2.6 Battery Longevity Requirements                  | Page 19 |
| 4.2.7 Battery Charging Safety                         | Page 20 |
| 4.3 Radio Frequency (RF) Design and Optimization      |         |
| 4.3.1 Advantages of Bluetooth LE protocol             | Page 20 |
| 4.3.2 Antenna Designs - Omnidirectional               | Page 23 |
| 4.3.3 Antenna Designs - Unidirectional                | Page 26 |
| 4.3.4 Beacon Signal Broadcast Rate                    | Page 30 |
| 4.3.5 Multiple Beacon Placement Optimization          | Page 34 |
| 4.3.6 RF Health and Safety                            | Page 38 |
| 4.4 Nordic nRF51822 System on Chip                    | Page 39 |
| 4.5 Module Enclosure                                  |         |
| 4.5.1 3D Printed Plastic Case                         | Page 42 |
| 5.0 Beacon Software Details                           |         |
| 5.1 Initial Design Architectures and Related Diagrams |         |
| 5.1.1 Beacon Firmware Architecture                    | Page 43 |

|                                                                   |          |
|-------------------------------------------------------------------|----------|
| 5.1.2 Building and Flashing the Beacon's Firmware                 | Page 46  |
| 5.2 Beacon Configuration                                          |          |
| 5.2.1 Configuration/Debugging Module                              | Page 47  |
| 6.0 Application Software Design Details                           |          |
| 6.1 Mapped Building Database                                      |          |
| 6.1.1 Purpose of Database                                         | Page 50  |
| 6.1.2 Storing the JSON Building Information                       | Page 52  |
| 6.1.3 Data Structure                                              | Page 57  |
| 6.2 Beacon Detection Library                                      |          |
| 6.2.1 Bluetooth Low Energy within<br>the Android Operating System | Page 58  |
| 6.2.2 Filtering miscellaneous BLE devices                         | Page 59  |
| 6.2.3 Deriving Distance from the Beacon's Signal                  | Page 60  |
| 6.3 Trilateration Module Design                                   | Page 61  |
| 6.4 Pathfinding Design                                            |          |
| 6.4.1 Purpose of Module                                           | Page 64  |
| 6.4.2 Deciding the Right Algorithm                                | Page 65  |
| 6.4.3 Graph Design Comparison for Buildings                       | Page 69  |
| 6.4.4 Off-Course Determination Protocol                           | Page 72  |
| 6.4.5 User Direction Protocol                                     | Page 75  |
| 6.4.6 Floor Sequencer                                             | Page 77  |
| 6.5 Application UX Design                                         | Page 79  |
| 6.6 Concurrency                                                   |          |
| 6.6.1 Comparisons for different call patterns                     | Page 82  |
| 6.6.2 Ensuring thread safety                                      | Page 83  |
| 6.7 Security                                                      |          |
| 6.7.1 User Data                                                   | Page 84  |
| 6.7.2 iBeacon spoofing                                            | Page 84  |
| 6.8 DynamoDB                                                      | Page 87  |
| 7.0 Beacon System Prototype Construction and Coding               |          |
| 7.1 Parts Acquisition and BOM                                     | Page 90  |
| 7.2 PCB Vendor and Assembly                                       | Page 91  |
| 7.3 Final Coding Plan                                             | Page 91  |
| 8.0 Beacon Location System Design Summary                         |          |
| 8.1 Architecture Summary                                          | Page 92  |
| 8.2 Architecture Diagrams                                         | Page 93  |
| 9.0 Beacon Prototype Testing                                      |          |
| 9.1 System Testing                                                | Page 95  |
| 9.2 Hardware Test Environment                                     |          |
| 9.2.1 Site Survey                                                 | Page 96  |
| 9.2.2 Signal Strength Testing                                     | Page 98  |
| 9.2.3 Battery Tests                                               | Page 100 |

|                                        |          |
|----------------------------------------|----------|
| 9.2.4 Antenna Impedance Testing        | Page 101 |
| 9.3 Software Test Environment          |          |
| 9.3.1 Application-specific Testing     | Page 102 |
| 9.3.2 Beacon Firmware-specific Testing | Page 103 |
| 10.0 Administrative Content            |          |
| 10.1 Milestone Discussion              | Page 104 |
| 10.2 Budget and Finance Discussion     | Page 105 |
| 11.0 Specifications and Standards      |          |
| 11.1 Software                          | Page 106 |
| 11.2 Firmware                          | Page 106 |
| 11.3 Hardware                          | Page 107 |
| Appendices                             | Page 108 |
| Works Cited                            | Page 109 |

Page Count -108

# 1.0 Executive Summary

Our senior design project will consist of using beacon devices for indoor navigation. The project will be divided into software and hardware, we will be designing our own beacon devices and creating the software that will be use for the beacon to communicate with google glass. There are multiple companies that are working on making this technology better, for indoor navigation this technology can be used at a supermarket store to navigate users to a desired product location through their smartphone.

This technology can also be used at museums so that when you pass by an object of interest you get a pop up you your smartphone that tells about the significance and history of the object. They way this technology works is many small beacon (about the size of a quarter) are placed in a building, the beacons will then communicate with your smartphone when you are in close proximity to them, this is why you can use it at museums or as a greeting when you arrive at a store and tells you of special sales happening that day.

The other way beacon technology can be used is how we plan to use it, once you are inside the store and you are looking for an item or at a mall looking for a store you can just tell your smartphone what you want to find and it will give you step-by-step navigation as how to get there by communicating with all the beacons in the store, so it is not a GPS navigation system. Many companies are also working on using this technology to help navigate visually impaired people navigate through a city, and did consider doing this at a much smaller scale but decided against it because of many factor we can not take into account with a limited budget, and not like Microsoft that is creating a unique type of headphone to aid in helping the visually impaired.

After deciding on doing this project we decided our goal is to obtain a working product that will navigate the user to a specific room or office location in a building at UCF. The user will put on a set of google glass and say the desired location, such as room 474, and will then obtain voice navigation as to how to get there while communicating to the beacon devices where the user is located and in what direction it is headed. Since we are doing this project to a small scale, we decided to use the fourth floor of the engineering building to test our product, this is will be close to our senior design lab and we believe there will have less curious people that might want to take our beacon devices off the wall.

We also decided to do this project because each group member can gain knowledge on an area of interest, for this reason we will be designing our own antenna for the beacon devices and our own power supply. For the two group members interested in software programming, they will be creating the software that will locate the user through the beacons and will then give navigate then to the desired location. We will also be creating a 3-d printed case that will protect the pcb. Since our beacon devices will not be large in size, protecting them is important to prevent them from easily being damaged, and at the same time we hope to gain experience using a 3-d printer.

## 2.0 Project Description

The following sections will describe the motivation for doing this project and what we plan to accomplish.

### 2.1 Project Motivation and Goals

The reason we decided to design beacon devices and software is because we wanted a project that would provide a challenge and not just another project where we can repeat what multiple groups have done in the past. One of the group members has experience in working with beacons from a previous internship and we decided to use his idea, but in order to meet the requirements for a senior design project and have enough hardware we decided to build the beacon devices ourselves in order to have a more in-depth understanding of the devices in case something goes wrong while testing the final product. We will also be writing the code that will implement the indoor navigation of the final product, which provides us with the right amount of software and hardware to make this a good senior design project along with the challenges that will come up along the journey.

Our first project motivation was to be able to create a product that can potentially help visually impaired people navigate in an unfamiliar place, we wanted to create a product that can help impact the society in the near future. Through our research we found that there are several companies working on similar projects to help visually impaired people with the use of beacon navigational systems. These companies have spent years developing this technology and still don't have finished product available to the public, they are still performing tests to make the technology better and reliable. They also have researchers that have been looking for better ways to create the software for this technology.

For previously mentioned reasons and having a budget with limited knowledge compared to other companies, we decided to not create a product specifically for visually impaired people, but rather a small scale project of what other companies are doing. There are variables we cannot take into account in our project, for example, if someone leaves a chair in the middle of the hall and we blindfold a test subject, there can be injuries involved and we do not want such problems.

We will be doing the project the same way we had initially planned, but without having to blindfold the test subject, which we plan to use one of the professors in our grading panel on presentation day of senior design 2. We will be located in the fourth floor of Engineering 1 building and the user (volunteer professor) will put on google glass and say a room number as his destination, he will then hear turn-by-turn directions as how to get there. We also like this project because UCF can use our idea to locate beacons in every building on campus and create an app for students to use when looking for a classroom in a building they haven't been in before, especially freshman and transfer students. This would be different than the UCF map app because it would not use GPS navigation and can therefore have a good signal inside any building as long as there are enough beacons set up.

Since one of our group members had a pair of google glass we tried to look for a project in which we could implement using them, so we decided that instead of using a smartphone

to communicate with the beacons we would use google glass. Using the smart pair of glasses provides us with some advantages over a smartphone, for example one team member pointed out it will be easier for the software to know in which direction the user is headed or is facing forward in the hallway.

From this project we also hope to gain experience in working with a group of engineers and trusting each other. We will have to learn to trust each member's work because we do not all share the same knowledge, we have different backgrounds depending on our interests and internship experience. Being able to work in a group now will help us be more comfortable to work in groups in the workplace and be responsible for one part of a big project.

Another goal for our team is to become familiar with a 3-d printer, which is why we have decided to design a protective case for the pcb. 3-d printing has been an uprising technology for the past few years and we believe gaining experience design and using the 3-d printer can provide us with an advantage in our jobs when working on a special project or help get a job in the field of making such printers.

We will all be involved in process of ordering the pcb because we feel this is important for all electrical and computer engineers to know how to do. Knowing how the process works can be beneficial for us when doing a home project or working for a company that focuses on electronics devices, such things we do not learn in the classroom.

In summary, our motivation to do this project to take on a challenge and learn along the way qualities that will make us better engineers. Our goal is to create a product that in the near future will make a big impact on people's lives, change the way they shop at the supermarket by finding items faster, or by creating a system that will help visually impaired people.

## 2.2 Objectives

Our objective with this indoor navigation system is for each group member to gain knowledge in an area of interest they want to learn more about. With this project we will be working with pcb boards, designing antennas, writing software, and doing research among other things.

Research is very important in all fields of engineering because we create new technology. As one of our professors once mentioned, if we do not design and research, we are just technicians, and companies want to hire engineers to innovate and not to fix what has already been created. We took the advise of Dr. Richie and design the project in the way that each group member would take advantage of senior design class as if it was a two semester long elective.

One group member is interested in working with circuit boards and understanding how the power supply to an electrical system works, he will use this class to do research on these subject and decide the best options to implement our hardware. The other electrical engineer in our team wants to expand his knowledge on radio frequency performance optimization, power efficient radio transceivers, and advantages in different antenna

designs. This team member will be responsible for creating the best antenna design possible for the purpose of our beacon devices and do research on Bluetooth 4.0 technology, which our devices will be using.

Two group members with a background in programming from previous internships and related classwork will be responsible for writing the software to implement the indoor navigational system we desire. They will be doing research mapped building database, on using the beacons and android operating systems for google glass for the purpose of the signal. They will also be designing a trilateration algorithm to use on our software, they will also be doing a pathfinding design to determine the best path to get to the user's desired location.

After several weeks of discussion and changing the purpose and how to test our project we decided to use the fourth floor of Engineering one building to perform our tests and to use for our final presentation. We got permission from Dr. Richie to perform our final presentation in engineering one building because we want at least one professor in our grading board panel to use the equipment and share their thought with the others on the panel on how effective the project is.

We plan to get a volunteer professor and have them put on google glass and say room location he would like to go to. He will then hear turn-by-turn directions as how to get there. Since we are doing this project to a small scale it will seem as a big deal, and for that reason we will be making our project open source. We want future senior design students to keep building on our project and make it better as the years go by.

By having the opportunity to work with beacon in our senior design class, we believe any of us will have an advantage if we want to work for one of the companies working on creating and making these devices better. The programmers on our team will have experience with the algorithm to use to implement the software and communication of the devices with android products. The other members will have experience designing the device itself and will have an advantage by understanding how it all works and will have ideas to make a better product.

## 2.3 Project Requirements and Specifications

In order to do the indoor navigational project we must build our own beacon devices. In creating the beacon devices we will be designing our own antenna for each device in order to obtain an antenna that meets our needs based on where we will be testing the product. For this reason the group will have two antenna designs, omnidirectional and Unidirectional. Depending on the position of where the device will be place will depend what type of antenna it will use. We also want to create our own power supply system and have considered running some of the devices off ac power in order to avoid buying a lot of 3v batteries, and we have looked at our options of having a solar cell to obtain the power needed to run the device.

The project also has a good amount software design involved. We will be creating the software that will be responsible for communication between the beacons and google glass in order to navigate the user. Our group has two members with experience in programming



and coding, this is why we feel capable as a group to design the software. Our project will be different from other groups because it will require more testing than other groups, and for this reason we cannot compare our timeline with the progress of other groups, we must be ahead because after our first test, we must calculate to have up to three weeks in order to fix bugs with our software and make sure the hardware functions properly, especially the antenna.

Initially we discussed as a group to potentially test our final product in HEC but there are certain restrictions that would not benefit our product. For once, there will be more students wondering what we are doing and asking questions that will distract us. Also, the way the building is set up with curved hallways will create a more difficult environment to test our equipment and most likely need more beacon devices than if we use a better testing environment.

We decided to use the fourth floor of the engineering building because of the simply layout that is better to test our equipment and also has many offices that can be set as destination points. We are also focusing on using Bluetooth Low Energy (BLE) in order to save power consumption and take advantage of this new technology used on new smartphones as Bluetooth 4.0 which consumes less energy than previous models.

## 3.0 Research related to Project Definition

### 3.1 Existing Similar Projects and Products

This section explores different projects and products that use similar technologies to the ones being used in this project and offer similar features to those being offered by this project.

#### 3.1.1 Estimote

Estimote is start-up company focused on "building a sensor-based analytics and engagement platform." [1] Their goal is to change how brick-and-mortar business run and how people communicate with physical locations. They intend for users to use their smartphones to communicate with sensors (such as their Bluetooth modules) in order to be able to provide features such a micro-location context and product information.

Estimote currently produces a Bluetooth module which advertises the iBeacon signal. As shown in the teardown by Makezine [2], Estimote's module is built around a Nordic Semiconductor nRF51822 SoC, which has a 2.4 GHz radio which supports Bluetooth Low Energy, whose signal is transmitted through a PCB antenna. The module is powered by a lithium coin battery and is advertised to have a battery life of up to five years depending on how the Beacon is configured. Estimote claims that the beacons advertisement range is around 70 meters, but that can be adjusted due to Estimote's custom firmware running on the device. In addition to the Bluetooth capabilities, the module also includes both temperature and motion sensors. These sensors are meant to provide more information and

context which can be consumed by applications developed for the Estimote system. The Estimote website currently lists the development kit for \$99 which includes three Beacons.

Estimote is now also producing a Sticker Bluetooth module. These Stickers are significantly smaller than the regular Beacons, but are still able to offer much of the same functionality since they utilize the same SoC and contain the same motion and temperature sensors. The downsides of the Stickers in comparison to the regular Beacons include a lower battery life (one hour compared to the Beacons five hours) and lower customizability of the device (the Sticker comes pre-programmed with nearable-specific context whereas the Beacons are fully configurable). With the smaller form factor, Estimote mentions that these stickers are meant to work alongside the Beacons by turning objects they are stuck to into a "smart thing connected to the Internet." The Stickers have yet to be fully released, but a Development Kit which includes ten Sticker Beacons is now available for pre-order on the Estimote website for \$99.

In addition to producing hardware, Estimote has also developed frameworks that developers can use to interact with their Beacons and Stickers. Estimote provides a basic SDK for iOS and Android, which facilitates detection of Estimote devices and consumption of the information provided by the devices. More recently the company has started to develop a library dedicated to providing indoor positioning to systems using the Estimote Beacons. Unfortunately the indoor positioning library provided by Estimote is only available for iOS devices.

### 3.1.2 SPREO Navigation

SPREO is a company who is focused on providing many different services which center around indoor positioning [3]. In order to provide these services, SPREO uses Bluetooth Low Energy beacons and other unspecified data inputs to make up a platform which they call Indoor Positioning System. This platform is made up of many different services which include: "indoor navigation, location based notifications and content management, venue analytics, and additional location based services."

SPREO prides themselves in being "beacon-agnostic." [3] Their website mentions that their system has been successfully tested with beacons from many different manufacturers. This differs from other systems, such as the system offered by Estimote, which are designed to only work with their own beacons. The requirements for the beacons used are that they have to be "low-cost, low-energy, reliable, and tested to ensure indoor positional accuracy." The beacons also have to be compatible with both iOS and Android Devices.

The architecture for SPREO's Indoor Positioning System can be seen in Figure 3.1.2-1. In this system, mobile devices gather data from outside sources, such as WIFI, GPS, and the BLE beacons, and from sensors found on the device, such as the gyroscope, accelerometer, and compass. The mobile devices then access a server in order to download other required information. With all this data, the mobile device then uses the SPREO proprietary algorithm in order to derive the users location. SPREO claims that this algorithm differentiates their system from other indoor positioning systems. It uses combines information from other sensors to improve accuracy. It also utilizes Machine Learning to

progressively improve the accuracy as the system becomes more widely used. With all these features, the algorithm is able to be accurate to less than one meter. With the location the mobile device is able to provide the desired service to the user and it also relates information back to the server in order to provide analytics information.

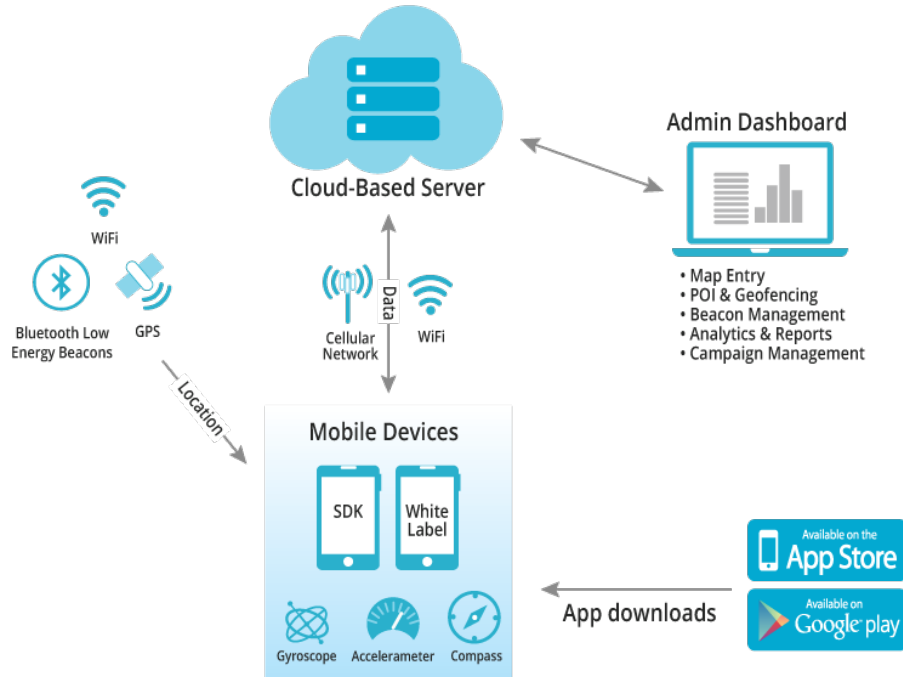


Figure 3.1.2-1: SPREO Indoor Positioning System architecture [4]

### 3.1.3 Motorola Bluetooth Smart Beacons

Motorola Solutions has developed their own indoor location provider. This system utilizes both Bluetooth Low energy and WIFI in order to deliver location-based features to both the consumer and the business. Motorola Solutions is providing this system as the MPact Platform [5].

The customer facing features for the MPact Platform include store maps, loyalty points and promotions and automated assistance. The store maps are using in conjunction with the location provider to efficiently direct customers to the location of their desired items. The MPact platform also allows for automated customer loyalty points and promotions. The system is able to automatically detect when a specific user walks into the business and then award that customer points or promotions. Lastly, the system is able to figure out when a customer has spent an extended amount of time in specific area of the store indicating that the customer might require assistance. The system can then direct an employee to the location to the customer in order to provide them with assistance.

Motorola’s MPact Platform also provides features to the business utilizing this platform. The system is able to capture and store information regarding the behavior of the customers. It is capable of detecting which aisles have the most foot traffic and which

products customers prefer. The system is also able to store customers shopping history and what influenced the customers buying decisions. All this information is then provided to the business which will then be analyzed to improve customer interaction possibly leading to improved customer experience and an increased number of transactions.

Motorola separates the detection granularity into three different categories: presence, zone, and position. Presence simply notifies the business when a customer has entered or exited the store/location. This can even be extended to cover the parking lot with the correct hardware. Zone allows the business to know when a customer has entered a specific area in the store/location. The zones can represent a specific department in a store or the lobby or a specific restaurant within a hotel. Position relates to the exact position of the customer within the store. Motorola claims that they can achieve an accuracy of 5-10 meters using WIFI and an accuracy of down to a meter using Bluetooth. With the pinpoint accuracy provided by Bluetooth, the system can get the location of the customer down to the product level.

Motorola provides their own Beacon modules named Smart Beacons. These modules support different modes of functionality. They are able to advertise Apple's iBeacon protocol, battery saver mode and Motorola's own MPact protocol. The Beacons are managed by Motorola's software. The software is able to change the transmission mode and set the frequency of each Beacon. It is also able to analyze the current setup of Beacons to ensure that they are not spaced too far apart. This helps achieve the best accuracy for the system.

### 3.1.4 Indoo.rs Solutions

Indoo.rs is another company offering their own indoor location provider using Bluetooth Low Energy. Their system is cross-platform and compatible with both Android and iOS devices. Their system offers many features which business can use. On their website they provide examples of systems set up for travel, retail, events, meeting room finder and public safety [6].

One of the more interesting implementations is the system set up at the San Francisco International Airport [7]. The airport worked in conjunction with Indoo.rs to create an application to help visually impaired passengers to get around the airport. The application is able to point out points of interest to the user as they walk around the airport. It is able to point out anything from restaurants and stores to bathrooms and water fountains. The application is also able to help guide travelers to their destination within the airport. Although the application was originally designed for the visually impaired, it has also been proved to be useful to passengers regardless of their sight. Indoo.rs helped the airport to efficiently set up the Beacons throughout the airport and helped map the airport.

Indoo.rs provides many different features with their system. First, all calculations are run on the users mobile device. This means that the device does not require an Internet connection. They also provide "fingerprinting" in which they use the current WIFI infrastructure instead of Bluetooth Low Energy to find the location of users. This system

is less accurate, it provides an accuracy of about 5 meters compared to the 2 meters of accuracy provided by using Bluetooth Low Energy, but it eliminates the cost of having to purchase and deploy a fleet of Beacons. Indoo.rs also allows different kind of maps to be used to display the position of the user. Businesses can choose to use a simple two-dimensional map created using jpeg or png files. They can also choose to create more complex three-dimensional or even augmented reality maps. Lastly, much like Motorola's MPact platform, Indoo.rs allows for data-collection and analizations. It allows businesses to do real-time data collection of customer behavior like traffic patterns, frequency and duration of visits.

Unlike some of the other indoor positioning systems, Indoo.rs does not make its own iBeacon modules. Instead, they work with hardware partners such as Kontakt.io, BEACONinside, StickNFind, and others to provide their customers multiple options for sourcing the Beacons for their facility. Indoo.rs guarantees that the Beacons provided by their partners will be compatible with iOS and Android devices and will allow for accurate locationing. Even though they recommend that their partners Beacons should be used, the system is designed to work with any Beacon.

One of the downsides of Indoo.rs is its accuracy is slighly worse than other solutions. Other systems like Estimote, SPREO, and Motorolas MPact platform offer an accuracy down to one meter whereas Indoo.rs only offers an accuracy down to two meters.

### 3.1.5 Microsoft Location Provider Project

Microsoft is currently working on a pilot project very similar to our beacon navigation senior design project. Since we have limited resources we will be setting beacon devices in the fourth floor of the engineering building and use google glass for the user to say the desired navigation and to be taken there through audio and calculating the location of the google glass.

Since Microsoft has unlimited resources as compared to us they want to implement the project to a big scale, in a city. Microsoft wants to set up beacons all throughout a city; in the streets, stores, restaurants, bus stops in order to potentially help 246 million visually impaired people. They are constructing a bone-conducting headphone that will communicate with the beacon devices and the smartphone of the user. These pair of headphones will be placed on the face below the ear so the visually impaired can hear environmental noise such as somebody walking by or a train far away.

Engineers are good at solving problems, but can often forget important details, like making a visually impaired person use a smartphone, I am not sure if this could be the most effective method. The headphones they are working sure do seem like a very good invention. If you are deviating off the path you are supposed to take you will hear a sound like "galloping coconuts", they also provide turn-by-turn direction and warning on approaching streets. Microsoft is currently running tests in a small city in England to learn how to make the product better and ready for the market. [8]

With the beacon technology available and the many potential customers out there, we believe starting a start-up company would be a good idea, but each group member has different plans after college as to what field to work in. In Florida, I believe the Publix supermarket chain would be interested in the indoor navigation so that customers can just say to the phone application what they are looking for and the app will then navigate them to that part of the store (ie go straight and make left on isle 5 and item will be located 15 feet to the right on top shelf). Another potential client for this product is the Mall at Millennia in Orlando, FL because they are a high-end mall that could afford the technology. They can use it to navigate tourists to their desired location instead of having them attempt to understand the map and get lost and not even find where the bathrooms are located.

With the potential market available for this technology it would be a good idea to start a company that can compete with the big guys now, before it becomes normal technology and cell phone applications of all types are available. I believe one app can be created that can be synced with other app like one for a mall, a supermarket, the hardware store will all operate under the same app because the user will want a user-friendly software.

Another reason we decided to do this project is because it is not something that has been done many times in the past by other senior design groups, we wanted something that could provide us with a challenge and that we can benefit from. Ten years from now when this technology is used a majority of people we will understand how it works because we worked on this in senior design, in a way, we didn't want to do a project using "old technology".

## 3.2 Relevant Technologies

This section covers some of the main technologies that make up the different components of this project.

### 3.2.1 Solar Power in Microelectronics

Over the past decade the technology of solar cells has improved and more companies are starting to invest in this “green technology” because it is more efficient. But the truth that there is still a long way to go for solar cells because they can have an efficiency loss of up to 20%. The world has so many power plants in order to provide electricity to the as many people as possible, but the truth is the sun can provide enough energy in one minute to supply all the energy the world needs for one year. The problem with solar cells is that there is not an efficient way to convert the sun’s energy, that is why solar cell energy can be expensive. Companies that manufacture solar cells are using good quality material so the product can last longer until they can find a cost-effective solution. [9]

## 3.2.2 Bluetooth Low Energy

The Bluetooth Smart – dubbed Bluetooth LE or Bluetooth 4.0 - protocol offers enhancements over its predecessor Bluetooth technology. The radio communications protocol operates around the 2.4 GHz radio frequency and is used for short-range wireless communications for devices; Bluetooth LE, or Bluetooth Low-Energy, aims at delivering low-power consumption RF communications in the short-range – i.e. typically below 100 meters of signal coverage. The Bluetooth LE protocol is much like to similar competing RF communications protocols in the 2.4 GHz ISM band such as Zigbee or Ant except with greater emphasis on power savings rather than data throughput or advanced modulation.

At 2.4 GHz, the Bluetooth LE technology can operate within small design size constraints using small antennas which make it a perfect wireless protocol for miniature devices. In addition to being able to support miniature design dimensions, the Bluetooth LE protocol emphasizes low current consumption. This is achieved by using simpler, adaptive modulation techniques (as compared to standard Bluetooth technology) and much more efficient duty cycles. Depending on the case, Bluetooth LE can have a power drain of as much as 1% as compared to standard Bluetooth. It should be noted that the Bluetooth Smart standard is not meant for as much data throughput as the standard Bluetooth technology. The data rate for Bluetooth Smart technology can peak at a third of standard Bluetooth's throughput, a trade-off for the increase power-efficiency.

The Bluetooth LE protocol uses a frequency hopping scheme much like its predecessor technology. Frequency hopping increases its robustness in counteracting potential interference from other radio frequency sources in the 2.4 GHz spectrum. Spread spectrum also increases the stealth and security of a Bluetooth LE signal as the carrier power tends to “blend in” with the background noise power in spread spectrum systems. There is also the increased real-estate of the spectrum within the 2.4 GHz to approximately 2.5 GHz ISM band – i.e. spread spectrum coding allows for a greater density of users to communicate in the same area using the same frequency band. Additionally, Bluetooth Smart supports advanced security features such as 128-bit AES, or Advanced Encryption Standard, and Counter with CBC-MAC (CCM) mode which gives developers great freedom in creating a secure wireless communications link.

Last of all, Bluetooth 4.0 LE technology comes with a large compatibility support base; many modern products roll out of the production line with the latest Bluetooth support. This makes consumer electronics utilizing Bluetooth easily accessible to the latest smartphones, tablets, laptops, and many more devices. This compatibility support base will only continue to expand as the market's demand for tiny, low-energy wireless devices increases; wearables technology – e.g. smart watches and Google Glass – all have the benefit of supporting Bluetooth LE for low-energy, efficient wireless data connections to systems utilizing the Bluetooth wireless protocol. It should be noted, older devices with only standard Bluetooth support (i.e. no Bluetooth 4.0 / LE support) will not be able to utilizing the energy-saving features of Bluetooth LE.

### 3.2.3 Apple iBeacon Protocol

iBeacon is a Bluetooth Low Energy protocol developed by Apple in order to extend Location Services in iOS [10]. This technology becomes especially useful indoors where other location services, such as GPS, are not accurate or reliable enough. Apple is currently using iBeacons in their Apple Stores in order to deliver contextual information based on the users location within the store. This means that the user’s iOS device will automatically display information of products close to the user. Although many companies are leveraging this technology for both Android and iOS devices, Apple has specified that any companies using the iBeacon name with Android are susceptible to legal action [11].

As defined by Apple, the iBeacon protocol carries three separate values: the UUID, the Major value, and the Minor value. The UUID (Universally Unique Identifier) is a sixteen byte ID. The UUID defines a broad group of iBeacons and would be common to all iBeacons being used by a specific application or location. The Major value is a two byte unsigned integer. It is usually used to define a more specific group of iBeacons within the group defined under a UUID. The Minor value is also a two byte unsigned integer. It is an even more. There are other values, which are not implicitly specified by Apple. One of these values is the calibration RSSI (Received Signal Strength Indication). This value is the 2’s complement of the measured RSSI value at one meter. It is used to calculate an estimated distance between the user and the iBeacon. The iBeacon prefix is nine bytes of information including manufacturer information and Bluetooth signal flags. All this data mentioned makes up the “Data” section of the signal shown in Figure 3.2.3-1. The rest of the advertisement is defined when setting up and configuring the Bluetooth Low Energy device and are abstracted out by the iBeacon specification.

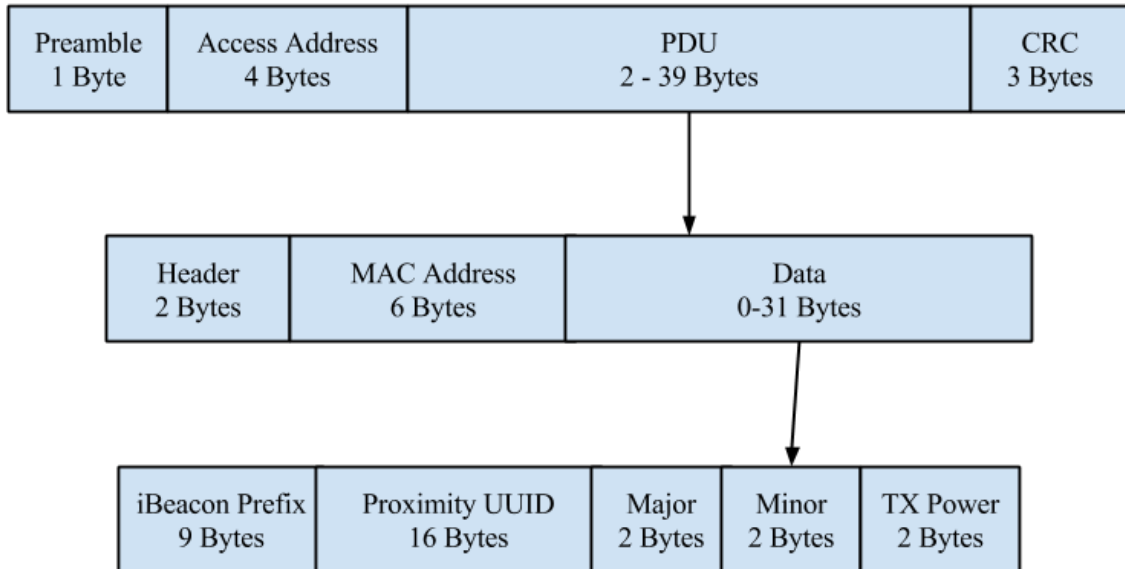


Figure 3.2.3-1 iBeacon advertisement package

Figure 3.2.3-2 shows an example implementation of the iBeacon protocol for a nationwide retail store. The UUID is among all the different locations of the stores. The Major value



then specifies each location of the store. Using this value, the application will be able to easily determine which store the user is currently in. The Minor value relates to a specific section in each store. This allows the application to determine which section of the store the user is in.

| Store Location |            | San Francisco                        | Paris | London |
|----------------|------------|--------------------------------------|-------|--------|
| UUID           |            | D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C |       |        |
| Major          |            | 1                                    | 2     | 3      |
| Minor          | Clothing   | 10                                   | 10    | 10     |
|                | Housewares | 20                                   | 20    | 20     |
|                | Automotive | 30                                   | 30    | 30     |

Figure 3.2.3-2 Example implementation of the iBeacon signal [10]

### 3.2.4 Google Glass

Google Glass is a wearable device developed by Google. It is designed to be worn like normal eyeglasses while its display sits right above the users line of sight. Figure 3.2.4-1 shows how the display on Google Glass works. The display is made up of transparent prism with a reflective layer inside. A projector that is in the main housing of the Glass, projects an image into that reflective layer which is then reflected into the users eye. This creates the illusion of a transparent floating screen. Google claims that the projected display is “equivalent of a 25 inch high definition screen from eight feet away.” In addition of the display, the Glass also includes a bone conduction speaker, a touchpad, and a microphone for I/O. Google Glass is powered by a Texas Instruments OMAP 4430 SoC and one or two gigabytes of ram depending on the hardware revision. It also has a five-megapixel camera mounted at the front of the device. In terms of connectivity, the Glass supports Bluetooth (with Bluetooth Low Energy support), and WIFI 802.11b/g. It also has a gyroscope, accelerometer, magnetometer, and light sensor.

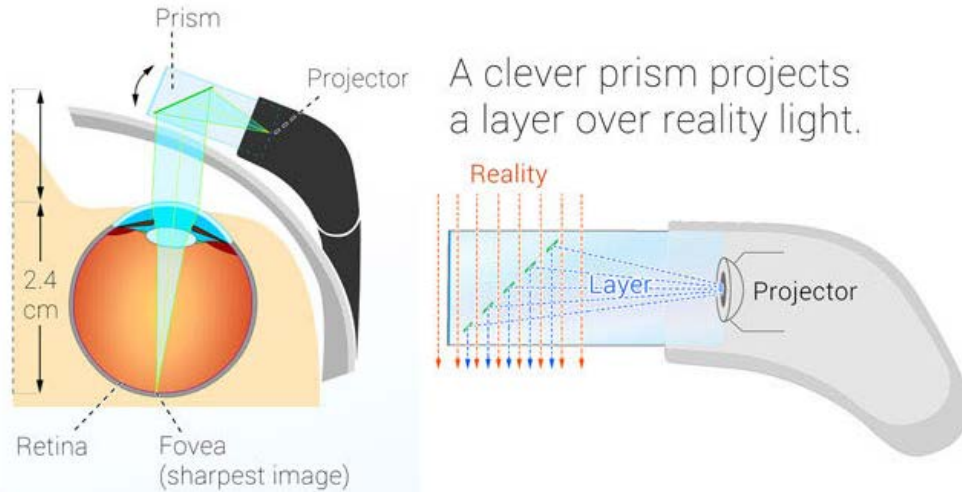


Figure 3.2.4-1 Google Glass display diagram [12]

In terms of software, Google Glass currently runs Android 4.4 with a modified user interface. The user interface is mostly based on a timeline of Cards centered around a Home Card. While on the Home Card, the user is able to interact with the device by using their voice. The user can speak the key phrase “Ok, Glass” and the menu of more key phrases will be brought up. This allows for mostly hands-free operation of the device. Development for Google Glass is done by using the Glass Development Kit (GDK), which is a modified version of the regular Android SDK which includes for Glass specific Classes, likes Cards and the voice recognition Activity, and removes support for a few features not compatible with Google Glass. Because of this, many of the libraries built for Android are also compatible with Google Glass. Most tools used for Android development, such as Android Studio, Eclipse with ADT, and ADB, are also used for Google Glass development. This allows for a relatively easy transition between Android development and Google Glass development.

## 4.0 Beacon Hardware Details

The following sections will explain the hardware section of our project.

### 4.1 Initial Design Architectures and Related Diagrams

For this section we had planned to add schematics of similar beacon devices available in the market to buy and then compare to designs we are making. Unfortunately because of copyright issues we cannot display the work of other companies on this report. Instead, we will describe several beacon product available today.

The first is a coin-sized beacon, the diameter is about the size of a coin because it uses a 3 volt button battery about the same size of the device. We like this design because of the simplicity, it does not even contain a protective plastic case. The other popular beacon

device is about the size of a fist and has a protective case with only one LED visible on the front view of the device.

We plan to incorporate both designs into our design by creating small beacons with a protective case and possibly use strong double sided tape to place our devices in the hallways of the engineering building.

## 4.2 Power Design Specs

### 4.2.1 General Power Supply Requirements

The beacon devices use Bluetooth Low Energy, often referred to as BLE. The user sets the rate at which the device transmits information, it can send the data to be found in intervals from 20 ms to 10 seconds. A faster interval rate is desired to connect to the phone, but a trade-off is a shorter battery life. For the purpose of this project we desire a battery life of at least 4 months in order to get all the testing done and final presentation. Another reason we will be using BLE technology is because we will not be using the maximum range and transfer rate. Our concern is using the device in a way that does not consume the energy of the battery at a fast rate. BLE power consumption is between 0.01 and 0.5 as compared to 1W for traditional Bluetooth. The BLE technology take more time to transmit data, but for the purpose of our experiment this will not be an issue since the most time it will take is 3 ms. Since we are using less power the maximum range each device will cover is about 150ft depending on interfering objects as compared to over 300ft for traditional Bluetooth, so we will be using more beacon devices (possibly twice as many) using BLE rather than "normal" Bluetooth. As a group, these are all trade-offs we have decided to make in order to take advantage of the new BLE technology. In order to save on battery use, we want our beacon devices to an ac and dc converter in the power supply. The way we intent to build the device is that if there is a wall outlet near the location we want to locate the beacon, we will use ac as the source. If the device will not be located near a wall outlet, we will use dc source from a 3v battery. This design will allow us make our own power supply and learn more how our devices will work and provide a small challenge in senior design. We are also looking at the possibility of adding a small solar cell strip to our device for when the battery runs out or does not function properly. We are currently looking at the options of a small cell like the ones calculators use or cell that will cover the front surface of the device. Our power design will include an ac dc converter circuit and an option to run off from solar energy. As a group we believe this is a good power design for the hardware portion of the project, among the antenna design and LEDs and LCD display on the device.

We will be using diodes in our power supply for safety purposes and as an ac dc converter for a potential design among other reasons. Diodes are used to prevent too much current going through a device and potentially causing damages. We will be using 1N4001 diodes in our project. The diodes will be used to build a full wave rectifier, connecting them in a loop. The cathodes of diodes one and two will be connected together, the anode of diode

2 will be connected to the cathode of diode 3, then the anode of diode 3 will connect to the anode of diode 4, finally the cathode of diode 4 will connect to the anode of diode 1.

In order to build an ac dc converter we will have to add a small transformer to our circuit, with a primary and secondary winding. The secondary will be connected to the cathode of diode 3 and to the cathode of diode 4, and the output to the circuit board will be connected from the wire connecting cathode of D1 and D2 and to where the anodes of D3 and D4 meet.

Next, a capacitor will be connected from the wire connecting cathode of D1 and D2 to where the anodes of D3 and D4 meet in order to act as a smoothing capacitor, the following formula will be used to determine the capacitance we will use:

$$\text{Capacitor (F)} = \frac{(5) \times (\text{current supplied by converter})}{(\text{secondary transformer rating}) \times (\text{frequency}) \times 1.4}, \text{ frequency} = 60 \text{ Hz}$$

Figure 4.2.1-2 Formula for smoothing capacitor

Since we have not performed any tests yet, we cannot provide the exact numbers at this time. The formula in figure 4.2.1-2 will be used to calculate our capacitor value. We will also be needing a voltage regulator to that regulates output of the converter. The schematic of the regulator is below, pin 3 will be connected to ground, pin 1 will be connected to the capacitor and pin 2 will be the output voltage.

The output from the voltage regulator will go into the Nordic nrf51822 chip pin A2.

## 4.2.2 Current Power Supply Implementations

There are two power supply methods available, linear and switching power supplies. The switching method uses a regulator in order to be efficient, and the linear method uses a resistor in order to regulate the output voltage. In order for a linear power supply to function properly it needs to have a complex regulator circuit. On the other hand, the switching power supplies are more efficient and smaller but they are more complex to build because the different currents can cause electrical noise.

## 4.2.3 Solar Power Research and Design

We considered adding a third power source option to our devices, solar. At the moment we do not see the need for our devices to have so many option from where to consume

power, we will discuss how the solar power would work in case we decide to add it to our design again.

The way solar energy would work on the beacons is the same way solar powered calculators work, it uses a coin-cell battery and small photovoltaic cell. The way this photovoltaic cell works is by converting photons to electrons using an average of 14 percent efficiently. Indoor lighting is enough to power up our devices, it will just not last as long because of the continuous use and therefore might be an ineffective power supply for the time being. [13]

Solar cells can be powerful enough to be used in the following products:

- Toys
- Watches
- Calculators
- Street lights
- Portable power supply
- and Satellites

From this list we can solar energy is involved in our everyday lives. Because the device only charges when it is exposed to light, the solar cells are charged to when exposed to light in order to provide a constant energy source, another way of charging is also to charge super-capacitors instead.

## 4.2.4 Different Battery Technologies

The battery might seem like a thing of the past, but there has been advancement on the technology of batteries because researchers believe they can solve the need of energy in the world by making advances in this technology, just like solar cell researchers believe they can solve the energy crisis. For example, there is a team of researchers at MIT currently working on lithium-air batteries, they are currently analyzing the best material to use for the electrode that causes the battery to be more powerful and lightweight than the current technology.

Another product this team of researchers are analyzing is carbon nanotubes, these devices can release powerful waves of electricity that can be capable to power up small to mid sized electrical appliances. An engineer in Stanford University is working on creating paper batteries and other fabrics that can conduct energy. The engineer takes regular paper and soaks it in an ink that has nanoparticles in order to conduct the energy, this technology advancement can provide a solution to lower battery prices, and what would be cool to see is a battery that is bendable. On the other hand, there is toy company that made a toy car that can run off sugary drinks such as sprite instead of batteries, and a bottle of sprite is way cheaper than a pack of batteries. Parents around the world could benefit from a technology like this because it is more cost-effective than buying batteries every time they run out. [14]

There is also a battery technology that can create its own power, researchers at Imperial College London have analyzed. This technology can be used on cellphones and laptops

because ever since the invention of the smartphone, battery life has been an issue because it consumes more power with multiple applications running at the same time. Just imagine if this technology can be used on hybrid cars, the miles per gallon efficiency of the car would improve significantly.

The invention of the lithium-ion batteries has been useful for many applications, one of the most popular being used to power hybrid cars. With hybrid car becoming popular there will be a shortage of lithium-ion materials to build the batteries. Scientists have found that the most important material that there might be a shortage of is the graphite for the anodes. Engineers have come up with a way to obtain sufficient material by having recycled tires go through an industrial shredder and then going through a process explained in the table below:

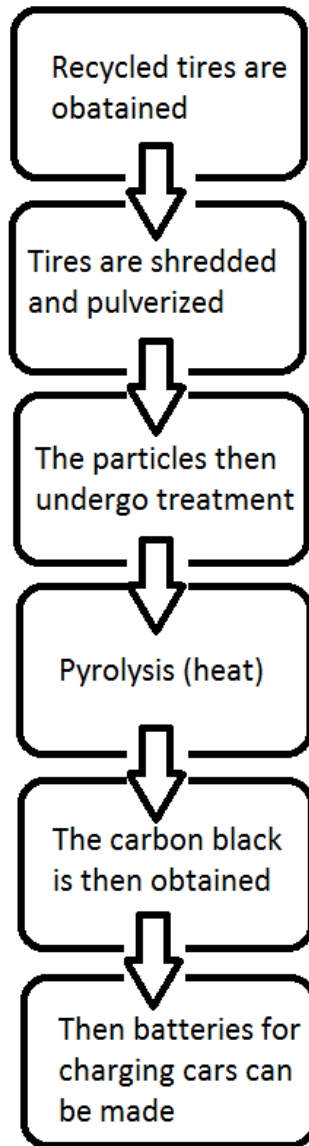


Figure 4.2.4-1 Process to make Carbon Black anodes in Lithium-ion batteries

The reason this system works is because the rubber in the tires is mainly composed of carbon, as is graphite, the important material in lithium-ion batteries. Once the tires are shredded, they are treated in order to produce a sulfur-rich rubber, which then goes through a heating process in a nitrogen environment that produces carbon black, which makes for a perfect substitute for the Lithium-ion anodes. The carbon black is better than the graphite because it is more efficient after 100 charge cycles. [15]

## 4.2.5 Battery Safety

Safety is a concern in many fields, not just engineering, and since we will be dealing with batteries and designing our power supply, I feel it is important I go over the concerns of battery safety. Batteries are used in our lives everyday, from a remote control to a watch and charging our cell phones, tablets or computer. Batteries can be dangerous if they are not used correctly or kept in a good environment. Battery manufacturers spend a lot of resources in package design of the battery because they know how dangerous it can be if they are misused, that is why batteries come with instructions to use. In this country, one of the biggest issues with the small button 3v batteries we will be using is small children swallowing them and the battery releasing a dangerous chemical into their body. This makes it important for us to keep the batteries in a secure place when we are dealing with the project in our houses over winter break or spring break. For safety purposes when we test the project for the first time we must make sure that the batteries do not overheat. Batteries do include a Circuit Interrupt Device which stops working if the internal gas pressure is too high, but in case this does not work and there is something wrong with our device, the battery can potentially explode and ruin the beacon. The following are concerns we must keep in mind when dealing with batteries:

- cannot consume more than recommended current, will cause short circuit
- must use correct size of battery, 3v is ideal for beacons
- power design cannot cause severe voltage drop, will not be able to supply the required current
- cannot store it in high/low temperature
- excessive vibration is not recommended

We plan to use a CR1025 3v battery to power our beacons. This battery has a nominal capacity of 30 mAh, a continuous standard load of 0.1 mA, and operates between temperatures of -22 and 140 degrees Fahrenheit which should not be a problem for us in the environment we live in Florida.

## 4.2.6 Battery Longevity Requirements

Some team members are concerned with the battery issues, this section is for the group to understand the battery so it will not ruin our beacon devices once in use. We have decided to use a 3v lithium battery model CR2032 to power devices when running off dc power. Since we can potentially have over one dozen devices, it is important for us to maintain the

batteries in good condition as last as long as possible in order for us to not go over budget in buying a lot of batteries, that is another reason why we want our devices to run off ac power also. We must make sure our batteries are not put under any circumstances were they might be able to deform, or put under extreme temperatures that they might leak acid. We must also keep the batteries from staining. For all of these reasons we will consider buying a battery storage container, to keep them from being harmed. The batteries we will be using have a low self-discharge rate of less than 2% and if kept in good condition can last over 900 hours of use. We must also test the batteries every few weeks during our testing procedure is senior design 2 to make sure it is still in good condition and not potentially stop our project from functioning correctly during testing.

## 4.2.7 Battery Charging Safety

The 3v button battery we will be using cannot be charged, the manufacturer recommends using diodes in order to prevent the power source from charging it. We must take into account protective resistances to regulate the current in case the diode fails. The manufacturer recommends using diodes with a small leak current in order to charged capacity within 1% of nominal capacity. Our other concern with battery charging is if we add a solar cell to our devices, which then will require another battery that can be charged. If this design can be beneficial to our devices, we will add another battery, but for now it is not necessary.

## 4.3 Radio Frequency (RF) Design and Optimization

The following section extensively goes over the method and processes of designing the radio frequency (or RF) component of the beacon modules which will serve the indoor positioning system area with Bluetooth Smart coverage signals.

### 4.3.1 Advantages of Bluetooth LE Protocol

Development of an indoor positioning system faces the challenges of being accurate, low-cost, and ease-of-use (compatibility). The Bluetooth 4.0 LE protocol was chosen for this project for several reasons which deemed this wireless protocol advantageous as compared to other potential solutions for this project. Other indoor positioning system schemes may utilize similar approaches of indoor positioning schemes such as the trilateration of radio signals not transmitted by the Bluetooth LE protocol (e.g. Zigbee, ANT, etc), which has the disadvantage of the following three factors: reducing ease-of-use and compatibility with devices, heightened power consumption, and optimal range. Likewise, there are other solutions which are limited by these three factors for indoor positioning systems utilizing visible light communications rather than radio by using LEDs with uniquely identifiable



identification tags which correlates to the location of the LED. These different schemes all have their pros and cons which will be analyzed in the following body.

The “LE” in Bluetooth LE protocol stands for “low-energy,” which implies its advantage in low-energy applications. The Bluetooth LE protocol allows for power consumption as low as 1% of the power consumption in classic Bluetooth technology. The indoor position system scheme does not require a large bandwidth or very low latency which removes the need for any other power-hungry network technology such as the commonly used wireless standard WiFi. Even amongst the most energy-efficient wireless protocol technologies available today for consumer devices - i.e. classic Bluetooth, ZigBee, and ANT, - Bluetooth LE remains the optimal candidate for reducing the power required to run a multitude of these wireless indoor position system beacons. This is illustrated in a study by Microsoft and the University of Washington Researchers which analyzed the power consumption between the big three competitors in the 2.4 GHz consumer radio protocol market; researchers emulated cyclical sleep cycles which resembles the operation of these devices with common consumer level devices; analyses concluded that the Bluetooth LE protocol has the most efficient duty cycles in conjunction with the most optimal sleep cycles which largely led to superior power consumption reductions [16]. Figures in the aforementioned study demonstrate the advantage of Bluetooth LE in providing superior power consumption reductions compared to the other 2.4 GHz short-range wireless protocols. This highly efficient, low power consumption makes Bluetooth LE a prime choice in wireless protocol for a low power indoor navigation system requiring an array of multiple beacons necessary for trilateration which implies a large multiplication of the power consumption for the total system and thus each module should be designed with minimal power consumption in mind – i.e. use Bluetooth LE.

Aside from power consumption, adequate signal range is another obstacle in designing an optimal indoor positioning system. In an obstacle-free environment such as the outdoors, wireless protocols such as Bluetooth or ANT can easily reach upwards of 100 meters. This project, however, aims to provide indoor positioning rather than outdoors. Indoor environments eliminates the assumption of line-of-site communications which, effectively, disrupts signal range. Obstacles exist in every facet of the environment; radio waves reflect off of all sorts of objects and end up propagating in all sorts of directions leading to effects such as multipath fading. These effects are visualized in figure 4.3.1-1 below. Interference in an indoor environment are plentiful and can arise from even the simplest of household appliances such as the microwave (which operate at a similar frequency as most of these wireless devices).

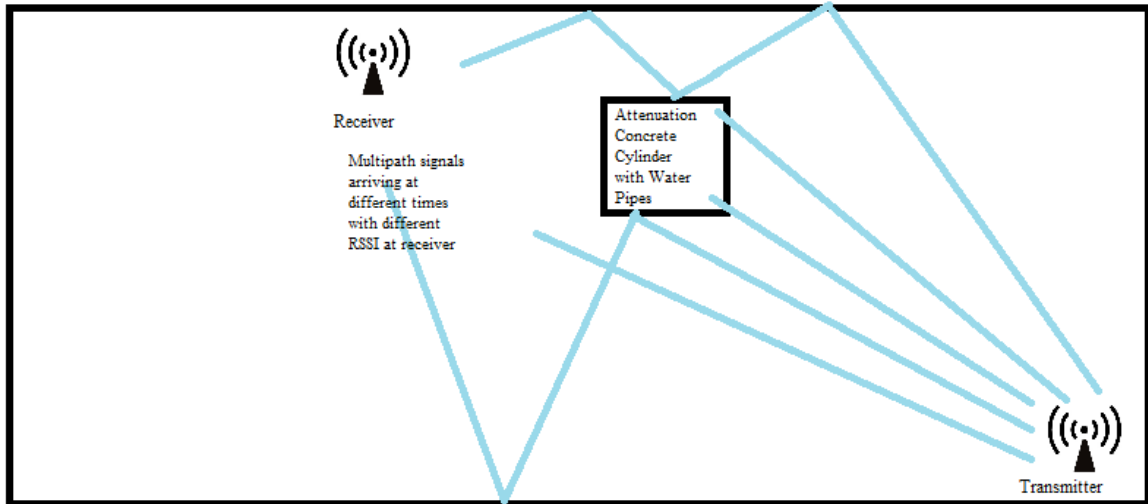


Figure 4.3.1-1 The above diagram shows the effects of obstructions present in an indoor site causing multipath fading and general attenuation.

Operating at 2.4 GHz, Bluetooth LE protocol's performance can also be disrupted by an increased noise floor arising from other wireless protocols operating in the 2.4 GHz – e.g. ANT and WiFi can operate in the 2.4 GHz spectrum and cause some interference from the increase in signal noise and effective decrease in the signal to noise ratio. According to the FCC's list of allocated frequencies within the 2.4 GHz range, other relevant standards allocated within the same frequency range as Bluetooth also include ISM, IEEE 802.11, 802.11b, 802.11g, 802.11n Wireless LAN WIFI, IEEE 802.15.4-2006, ZigBee, and many more. These devices operate in an unlicensed band which increases the chance of miscellaneous wireless devices in the similar frequency band interfering with the Bluetooth LE beacon wireless protocol beacon. Bluetooth protocol's channel hopping scheme reduces the loss in signal quality from environmental signal degradation such as multipath [17]. Bluetooth LE uses an adaptive frequency hopping spread-spectrum within an 83 MHz ISM band which also increases its robustness against interfering radio signals in the similar 2.4 GHz – 2.483 GHz band such as WiFi, ANT, or even ZigBee. This frequency hopping scheme also allows for a greater number of users to communicate within the same area without inter-symbol interference which is a great plus within an indoor environment.

Regardless of interference from the environment, this project does not require high data throughput because the Bluetooth LE beacons being designed only need to transmit a relatively small "advertisement packet" which contains the unique ID of the beacon; thus, throughput degradation from indoor signal attenuation should be considered but should not be a substantial issue in properly transmitting the beacon ID packets; indoor obstacle signal attenuation may still be cause for alarm for calculating approximate distance from beacons as the software relies on signal strength and trilateration for location approximation. Site survey should be recommend in case of any strong interfering signals such as microwaves or other 2.4 GHz operating devices which may disrupt and cut off signals our beacon signals. All in all, the Bluetooth LE wireless protocol demonstrates robustness against interference from indoor multipath effects and interference from the average 2.4 GHz band consumer radio electronics.

Other schemes for indoor positioning systems avoid aforementioned issues inherent in wireless radio communication theory. For example, schemes using visible light communications with LED's transmitting unique beacon IDs instead of wireless radio beacons avoid a lot of the interference (e.g. LED communications will not be affected by 2.4 GHz radio emitting devices) associated with the typical indoor office environment. There is a large drawback with visible light communications in that one must depend on line-of-sight as light does not propagate through obstacles as easily as radio waves. Thus, advantages of using radio, specifically Bluetooth LE protocol, still outweigh its disadvantages.

The biggest alluring characteristic of the Bluetooth LE wireless protocol is its compatibility with a multitude of wireless devices in the current market. Most consumer devices, which our indoor positioning system project will serve, support Bluetooth LE. – e.g. the latest Android and iPhone devices support the latest Bluetooth standard. In fact, the Bluetooth low energy technology protocol is expected to be massively deployed in the smartphone market; 700 million units are expected to support the new Bluetooth standard by 2015 [18]. This is opposed to other protocols and schemes such as LED visible light communication or ANT which has yet matured into the consumer technology market and would require extra peripherals to interface if the system were implemented with those alternative protocols. Bluetooth low energy's high compatibility "future-proofs" this engineering project as the wireless protocol utilized to interface to devices will reach a large volume in the near future.

### 4.3.2 Antenna Designs - Omnidirectional

The indoor positioning system demands that the beacons are optimally placed to provide maximum indoor signal coverage with the least number of beacon modules possible in order to save in costs. The modules constructed in this design will have a varying antennas depending on the best suited environment. Another aspect of this project is that the positioning system should be flexible, expendable, and thus have user-friendly installation. These criterion present the challenge of designing radio modules capable of dealing with the main hurdles of indoor wireless radio transmission: obstructions.

Providing adequate radio signal coverage requires the proper choice of antenna design for the Bluetooth beacon modules. The antenna design's resulting radiation pattern will ultimately affect the quality of the signal, the effective coverage area, and the overall effectiveness of the indoor positioning system.

The antenna for this design must be compact in order to keep the beacon module form factor small; in order to achieve a compact design, the antenna will be designed employing PCB tracing as opposed to having a setup with unnecessary wires and an external, unnecessarily big antenna which would create unneeded material costs and perhaps undesired electromagnetic interference from the wires attaching to said external antenna. Due to the short-wavelength nature of radio frequencies in the ultra-high frequency band, antennas operating in the ultra-high frequency spectrum tend to be very small in order to

produce the appropriate small wavelength radio waves; this allows the design and implementation of a 2.4 GHz antenna on a small portion of a PCB via board traces. If the utilized wireless protocol fell into lower frequency ranges of under <433 MHz, PCB antenna designs falter in size (too large) and efficiency [19].

In addition to constraining the antenna system to the most possible compact design, the antenna design must also match impedances of the transmitter in order to achieve maximum power transfer and retain the highest possible signal quality and range. The Nordic System on a chip device incorporated in this design utilizes an antenna output impedance of approximately 50 Ohms. The goal is to design or find an antenna with matching characteristics for 2.4 GHz radio broadcasting. Afterwards, the built prototype will be tested in order to ensure that adequate power transfer to the antenna is achieved.

In order to provide optimal signal coverage in the indoor premises, the antenna design one must keep the radiation patterns in mind. Several basic antenna designs with varying radiation patterns exist and each one is ideal for certain scenarios. Radiation patterns can change the gain of the signal in a certain direction. For this indoor positioning system, one must consider the environment in which the design will operate in: indoor hallways and large rooms.

To easily provide adequate radio coverage in a large room, an antenna design capable of a full-rotation radiation pattern should be considered. One should also consider the fact that the signal radiation pattern does not need to go towards the ceiling and should largely remain level, preferably above head height in order to avoid potential interference from human traffic. The omnidirectional antenna design would theoretically be optimal for this situation due the horizontal 360 degrees of service; albeit, one must consider a property of a typical omnidirectional antenna design: the area directly under and over the antenna often provides poor coverage in these designs. Optimal placement of the module and antenna will be discussed in later section of this document.

For this design, a 50 ohm balun will be implemented onto the Nordic nRF51822 SoC. This allows for ease of installation of a variety of open-ended 2.4 GHz antenna designs. An Inverted-F Antenna design will be considered for the omnidirectional module antennas. Basic visual layout of an inverted-F antenna is shown in Figure 4.3.2. This antenna configuration is commonly used in cellular phones due to its advantageous omnidirectional signal coverage. One-draw back to the Inverted-F Antenna design is its size; a smaller omnidirectional antenna design would be a Meandered Inverted-F Antenna Design. However, Inverted-F Antenna provides higher signal power efficiency (which is highly needed in a large indoor environment) than its smaller variant and size isn't a tremendous issue on this beacons as opposed to tiny baby handheld mobile devices.

It is much more cost effective and time effective to use a reference design for an antenna configuration. Reference diagram for a general case of the Inverted-F Antenna PCB design antenna is as shown below. Dimensions from reference design source will be traced onto the main PCB board, which will be multi-layered, and then routed to the transmitter's balun. The dimensions for the entire antenna should be approximately 25.7 x 7.5 mm.

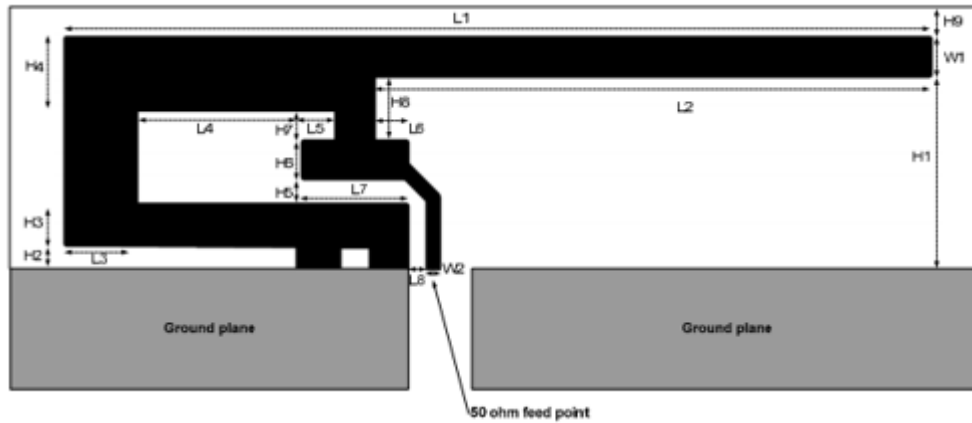
Reference designs are to be based on the Texas Instruments D007 2.4 GHz PCB transceiver [20].

The reference Inverted-F Antenna design should be able to efficiently broadcast service for the entire Bluetooth LE spectrum (Bluetooth 4.0/LE ISM band is 83 MHz-wide, surrounding the 2.4 GHz center frequency).

The balun chip utilized in this design will be the BAL-NRF02D and will be interfaced with the ANT1 and ANT2 output of the nRF51822 SoC's RF transmitter. The balun chip output is to be coupled with a 2.2 nH in series with the antenna and a 1.2pF in parallel with the antenna for a tuned impedance match. Antenna input will be 50 ohms impedance to increase compatibility with antennas.



Figure 4.3.2-1 Above diagram shows a basic inverted-F antenna implementation.



|    |         |    |          |
|----|---------|----|----------|
| H1 | 5.70 mm | W2 | 0.46 mm  |
| H2 | 0.74 mm | L1 | 25.58 mm |
| H3 | 1.29 mm | L2 | 16.40 mm |
| H4 | 2.21 mm | L3 | 2.18 mm  |
| H5 | 0.66 mm | L4 | 4.80 mm  |
| H6 | 1.21 mm | L5 | 1.00 mm  |
| H7 | 0.80 mm | L6 | 1.00 mm  |
| H8 | 1.80 mm | L7 | 3.20 mm  |
| H9 | 0.61 mm | L8 | 0.45 mm  |
| W1 | 1.21 mm |    |          |

Figure 4.3.2-2 Above diagram shows a TI reference design of an inverted-F antenna. Reprinted with permission from Texas Instruments

Above shows reference design, figure 4.3.2-2, for the Texas Instruments PCB inverted F antenna reference design with permission - “Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.” [20].

### 4.3.3 Antenna Designs - Unidirectional

Continuing on the discussion of varying antenna design implementations for the indoor positioning system, signal coverage of corridors and long hallways must be solved. The antenna for this design will also be restrained in size by implementing PCB antenna trace in order to sustain a small form factor of the overall product and reduce the usage of any wires which tend to bring unwanted electromagnetic interference. As with the omnidirectional antenna design, impedance of the antenna must match with the transmitter module on the nRF51822 greatest possible signal power efficiency. Standardized output impedance of 50 Ohms remains.

The indoor environment arrangement of corridors decreases the effectiveness omnidirectional radiation pattern covers the serving area. In order to cover this new environment, an antenna design capable of transmitting a radiation pattern in a singular direction should be considered; the radiation pattern does not require signal coverage in any direction except the corridor, so having the signal go into the walls behind the beacon, ceiling, or floor is undesired and redundant. A unidirectional antenna design works best for

covering indoor corridor environments. The caveat in dealing with unidirectional antennas is the decreased user-friendliness in installing these transmitting sites – i.e. a site survey is highly recommended in order to fully understand the area you are servicing with the unidirectional antennas since, unlike the omnidirectional antennas, unidirectional antenna design has higher margins of error in placement in terms of creating “dead-zones” for signals. Despite the increased man-power required to install these unidirectional modules, unidirectional antenna designs experience an increased gain as compared to omnidirectional antenna designs. This increased gain can be exploit in several ways: Increased gain could theoretically increase range and reduce the number of beacons required to cover an area; Transmitter power can be reduced to match the gain of the omnidirectional antennas which results in energy consumption reductions effectively making this a “greener” product and increasing battery life.

A 50 ohm balun will be coupled with the Nordic nRF51822 SoC which will increase compatibility with 2.4 GHz antenna designs. A Yagi antenna design will be considered to satisfy the requirements of unidirectional signal coverage of corridors. Reference diagram for the Yagi antenna PCB design is shown below. These dimensions will be traced onto the main PCB board, which will be quad-layered, and then routed to the transmitter’s balun. The dimensions for the entire antenna should be approximately 100 mm x 150 mm [21].

The balun chip utilized in this design will be the BAL-NRF02D and will be interfaced with the ANT1 and ANT2 output of the nRF51822 SoC’s RF transmitter. The balun chip output is to be coupled with a 2.2 nH in series with the antenna and a 1.2pF in parallel with the antenna for a tuned impedance match. Antenna input will be 50 ohms impedance to increase compatibility with antennas.

In effort to save product and development costs, modules will fit both antenna design in one module with the option of being able to switch between uni- and omni-directional gain modes using a de-multiplexer with a selector switch controlled by the SoC software and the outputs leading to the appropriate antenna.

The reference Yagi antenna reference design in figure 4.3.3-1 and figure 4.3.3-2 provides frequency range from 2126 MHz to 2623 MHz, which is sufficient for the entire Bluetooth LE spectrum (Bluetooth 4.0/LE ISM band is 83 MHz-wide, surrounding the 2.4 GHz center frequency).

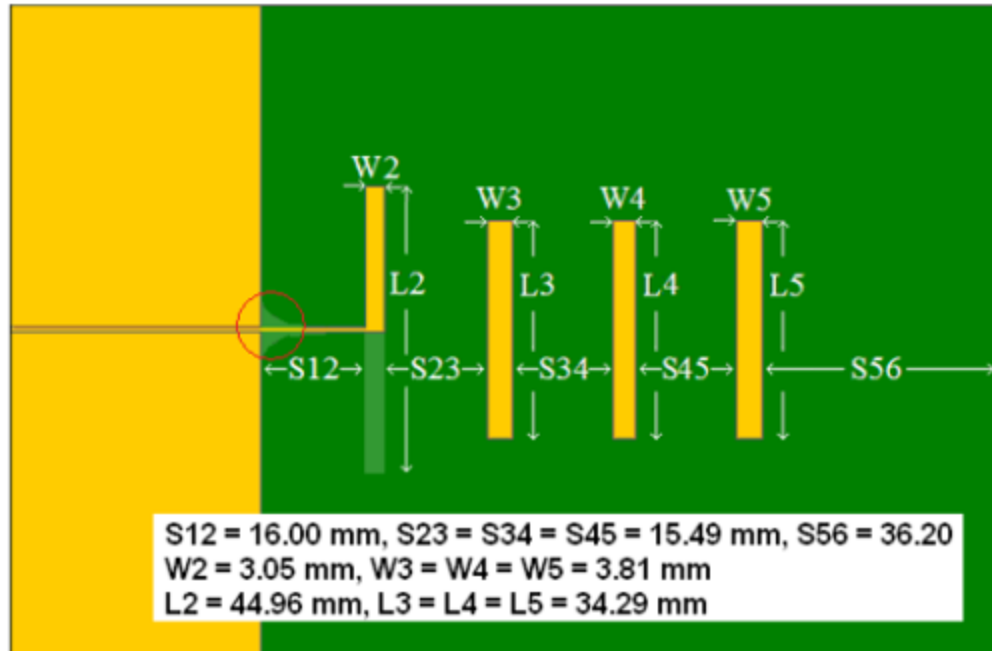


Figure 4.3.3-1 Above figure shows reference design of a Yagi PCB trace antenna. Reprinted with permission from Texas Instruments.



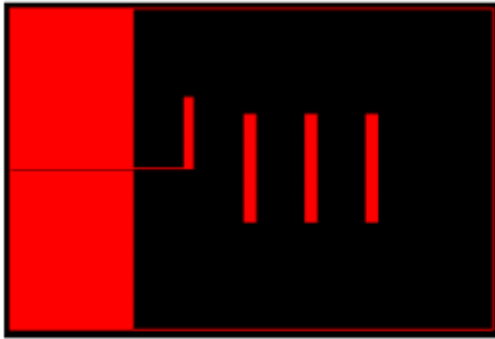


Figure 3. Layer 3 (Top, Antenna Top Layer)

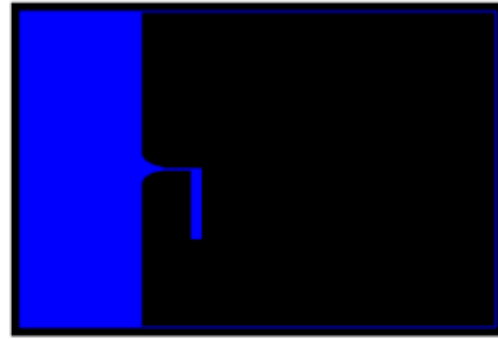


Figure 4. Layer 2 (Inner, Antenna Bottom Layer)

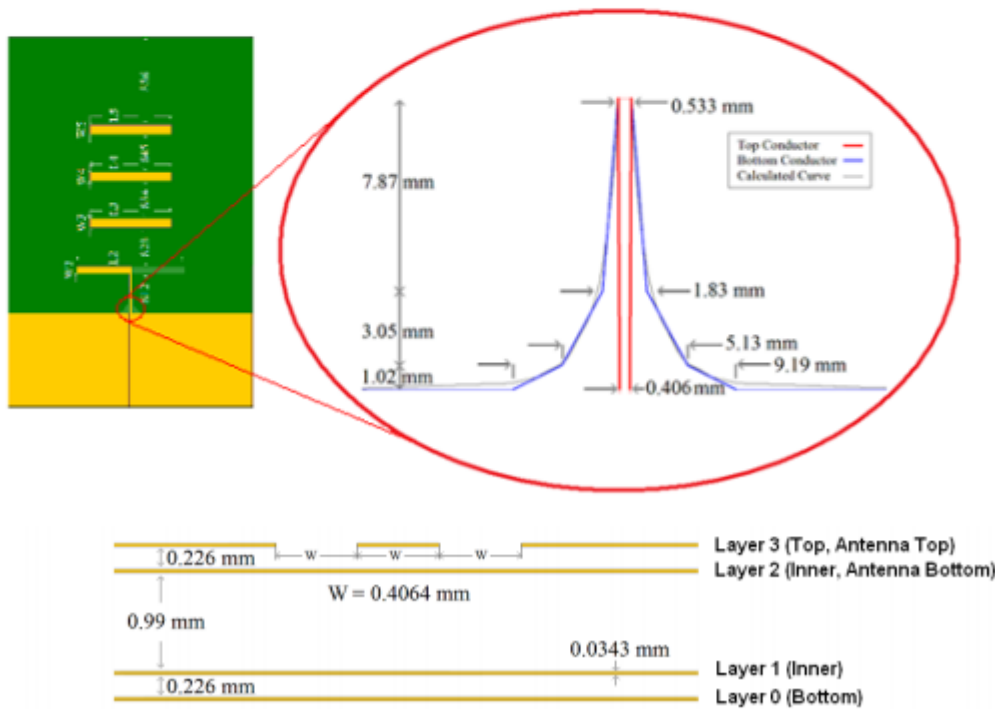


Figure 4.3.3-2 Above figure shows reference design dimensions of a Yagi PCB trace antenna. .Reprinted with permission from Texas Instruments.

Reference design for a Yagi unidirectional PCB antenna from Texas Instruments 2.4 GHz PCB Yagi PCB Reference Design – with permission: “Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.” [21].

## 4.3.4 Beacon Signal Broadcast Rate

A positioning system which aims at personnel indoor navigation must heavily consider a standard for precision and accuracy of the reported user's location versus the actual location. The trilateration used in the software side, which relies on perceived power of the Bluetooth signal, is completely dependent on the properties of the received beacon signal. These quality of these signals will assuredly widely fluctuate due to the hurdles of navigating through complex environments and interference from other signals within the 2.4 GHz communications band. Aside from the reception of the signal, the rate at which signals are being transmitted, received, and processed will also have a profound effect on the magnitude of precision that which the system can provide – i.e. the more the signal can be transmitted, the faster the user's position can be updated. All of these lead to opportunities for imprecision of the indoor positioning system on the user's position which is sought to be drastically avoided. There should also be a consideration of the power consumption of the device in relation to the broadcast rate of the beacon. This section will focus on the beacon signal broadcast rate – i.e. how short the intervals between beacon signal pulses are – and its effect on the performance of the indoor positioning system.

Indoor corridors and atrium halls offer different acceptable margins of errors of precision due to the size of the area. This gives way to many optimal broadcasting rates to choose from depending on the situation. One must also consider the standard velocity of an indoor pedestrian, which will be the standard user that we are catering to in this indoor positioning system until later developments of the indoor system progresses into covering indoor settings such as warehouses with much quicker vehicles.

Foot traffic can vary greatly in an individual's walking speed. The preferred walking speed can be influenced by impairments, age, gender, and population density. Impairments, such as an individual being visually impaired, leads to a decreased preferred walking speed. Similarly, older demographics tend to walk, on average, slower than lesser-aged foot traffic; women, on average, also experienced slower walking speed according to studies. Since these systems are dealing with a college indoor setting, the expected demographics will tend to reside greatly within the young-adults spectrum with roughly an even amount of females and males. Aforementioned studies reported that the average walking speed of male subjects in their 20s is around 1.5 meters \* seconds<sup>-1</sup>, or approximately 3.36 miles \* hours<sup>-1</sup>. On average, female subjects within the age group of 20s centered around 1.3 m/s, or 2.9 mph [22]. Averaging the two allows for an ideal preferred walking speed for the average young adult college student to be approximately 1.4 m/s or 3.13 mph. We must also analyze the effects of visual impairment on walking speed so that this indoor positioning system can be most effectively utilized by the visually impaired. Studies have shown that the visually impaired will walk short distances in a mean speed of approximately .84 m/s or 1.88 mph [23]. With this new information, there is a larger distribution of potential optimum walking speeds that could represent different key demographics that our system serves.

We must now analyze how the frequency of beacon signal transmissions affects the precision of our system in our aforementioned walking speed groups. The precision of the

system was calculated using basic kinematic equations to calculate the difference in distance between two points at a certain walking velocity and varying sampling frequencies of the beacon device. There is an assumption made that the pedestrian is already in motion and is experiencing no acceleration and thus the resulting derived equation necessary to perform the following data is  $Precision = \frac{Average\ Velocity}{Sampling\ Frequency}$ . Precision, in this case, is the change in position (in meters) of the indoor positioning system user at a set velocity being sampled at a fix rate. A lower value in this change in position equates to a more precise system since there is an increased possible resolution of the indoor positioning system. Analyzing the derived equation shows that it is an inverse relationship,  $1/x$ , and thus will form a hyperbolic distribution. This should lead to diminishing returns and give way to an optimal sampling rate at which further increase in the sampling rate does not provide for adequate benefits. Looking at the derivative of the inverse function,  $y = -1/x^2$ , one can quickly see how the rate of change of the precision versus sampling frequency quickly decays to very low numbers; Past 10 Hz (substituting for x in the derivative function), the change in precision improvements related to the sampling frequency begins to reach numbers in the thousandths order and past 30 Hz, numbers begin to reach the ten-thousandths levels.

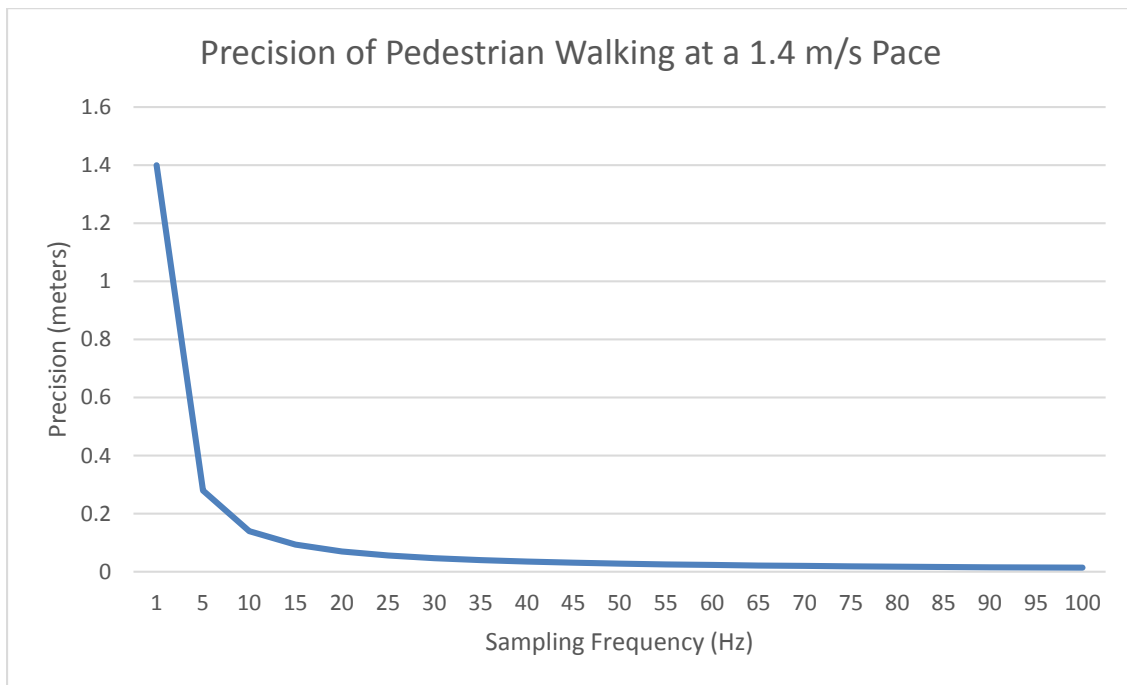


Figure 4.3.4-1 Above figure demonstrates relationship between sampling frequency and positioning precision

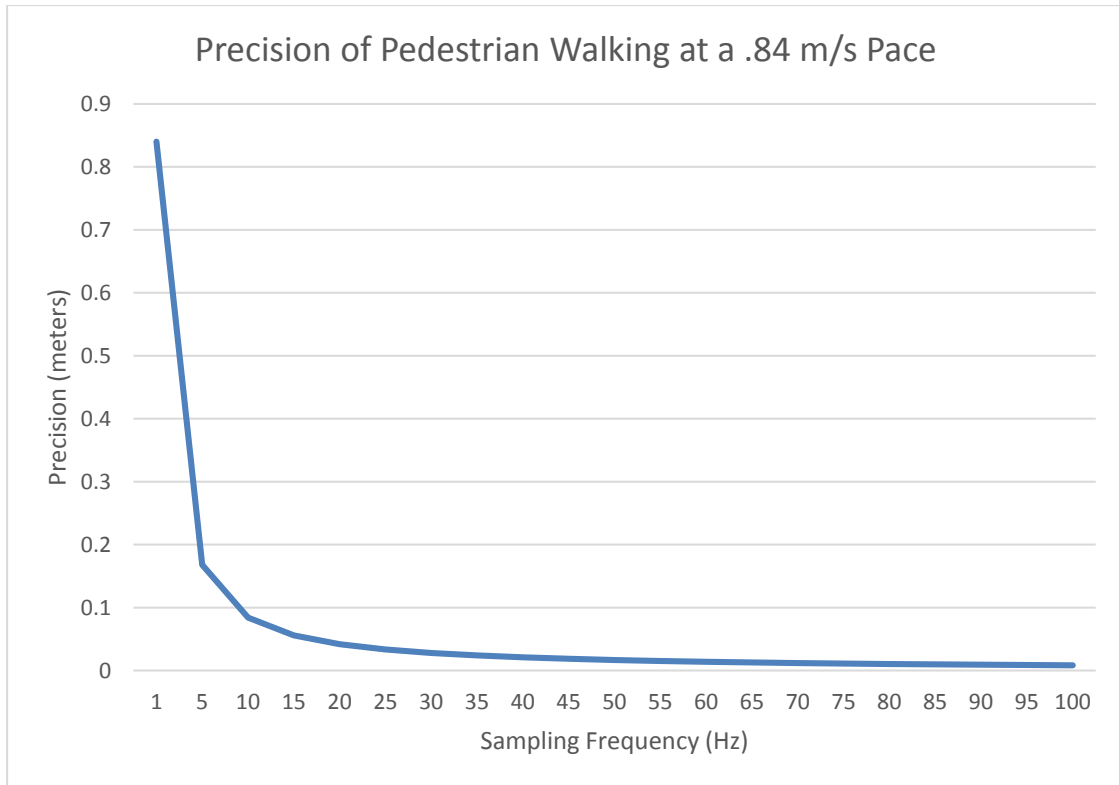


Figure 4.3.4-2 Above figure demonstrates relationship between sampling frequency and positioning precision

Visual inspection of the generated Excel spreadsheet data as shown in figures 4.3.4-1 and 4.3.4-2 illustrates that the precision does not increase greatly after a certain sampling frequency has been reached. The line graph for 1.4 m/s walking speed levels out around 30 Hz as does the line graph for the .84 m/s. It can be assumed that sampling and sending out beacon signal transmissions past 30 times a second does not improve precision much. With that declaration, it will be inefficient for the beacon system to transmit past 30 Hz.

Hallways on campus that support pedestrian traffic are typically around approximately 3 to 4 meters. A pedestrian walking at 1.4 m/s, using the data above, could get a location precision of around 1.4 meters and around .84 meters for the .84 m/s pedestrian. This is demonstrated in figure 4.3.4-3 This is plenty of precision for a pedestrian to navigate in an indoor setting and thus our system should not have to use a very high broadcast rate that might potentially drain the battery. In fact, other industry products that are Bluetooth LE-based use a standard sampling frequency of 1 Hz, even for the application of indoor positioning beaconing.

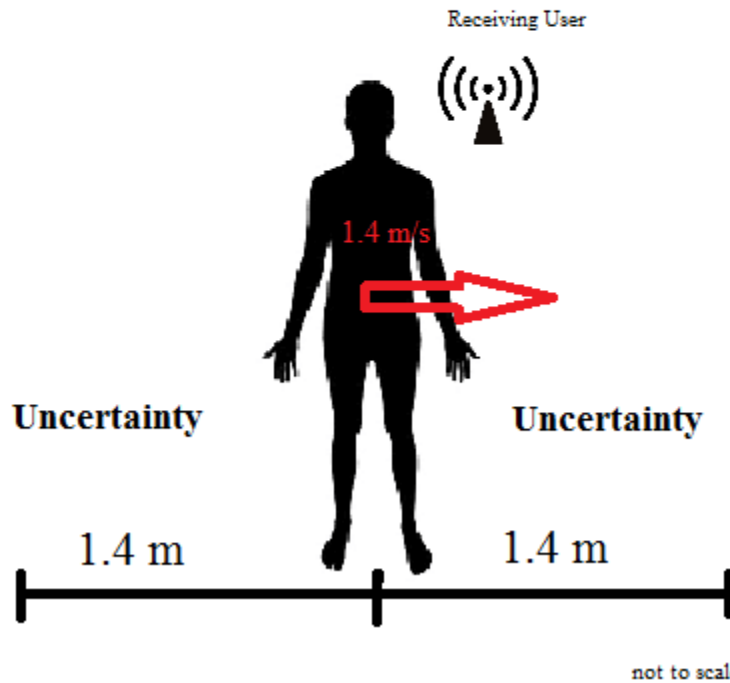
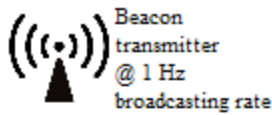


Figure 4.3.4-3 The above is a visual representation of the distance uncertainty magnitude that arises from a 1 Hz broadcast rate (height of silhouette is approximately 1.7 meters)

It should be noted that the above calculations assume that each signal received by the device and calculated are a result from clear, line of sight, no error system. In the real world, signals will typically be attenuated and distorted as they propagate in a complex indoor environment, causing some signals to be slightly off. The trilateration on the software side can account for this, to a certain extent, by averaging out extraneous signal samples – i.e. any signal received that is perceived as way off from the previous signal will be ignored or have a lessened effect on the perceived location of the user. These corrections from erroneous signal samples will only lessen the precision as there will be greater variation in the actual user's location that will be added with the corrected, mean values. The effects of signal attenuation, multipath, etc. can be mitigated with proper, optimized placement of the beacons. Regardless of design, a prototype build of the system should be tested and fine-tuned in order to overcome the unpredictability of the indoor environment and propagating radio signals.

An additional consideration in choosing the optimal Bluetooth beacon broadcast rate is the power consumption. Our product aims to be power efficient and thus must consume the least manageable power and still maintain reasonable performance in tracking a user's

location precisely and accurately. A study by researchers at Microsoft concerning the power consumption of the Bluetooth LE with varying “Sleep cycles” protocol demonstrated that the utterly insignificant power savings in sleep cycle times of above 30 seconds; at high rates of 0 – 10 second sleep intervals, power consumption is still fairly low for the Bluetooth LE protocol [16]. It should be noted that extrapolating data from this research study is not recommended due to the differences in power consumption of other applications utilizing the Bluetooth LE protocol – e.g. increasing transmission power to adequately cover large indoor areas will decrease how power efficient the devices are.

### 4.3.5 Multiple Beacon Placement Optimization

This indoor positioning system project faces the challenge of mitigating error from the several drawbacks of indoor radio utilization. Operating a radio, especially in the microwave region, can be sensitive to the present obstructions of walls, glass, metallic surfaces, moving pedestrians, etc. in addition to plenty of interference from other possible radio frequency emitting devices in the similar spectrum such as WiFi or even a simple microwave. Aforementioned metallic surfaces and other reflective surfaces which radio waves perceive as having an impedance difference lead to attenuation and reflection of signals causing effects such as “multi-path fading” which can only be corrected to a certain amount through techniques such as channel hopping in wireless protocols. Thus, choosing a proper placement is a necessity for any wireless system’s performance, especially an indoor system with all of interference and obstructions.

Dynamic factors such as the movement of pedestrians should be considered. Pedestrians are likely to be equipped with interfering sources of radio frequency devices such as their cell phones in addition to serving as another obstacle for beacon signals to circumvent. At 2.4 GHz, radio waves generated by our wireless system will be unable to largely penetrate the human skin, let alone their clothing. From this fact, it is recommend that the indoor position system has beacons deployed at above the pedestrian level to maintain as much line-of-site communications as possible. Human height, taking into consideration the differences between male average height and female average height, around the age of 20 – 30 years old is approximately 1.7 meters.

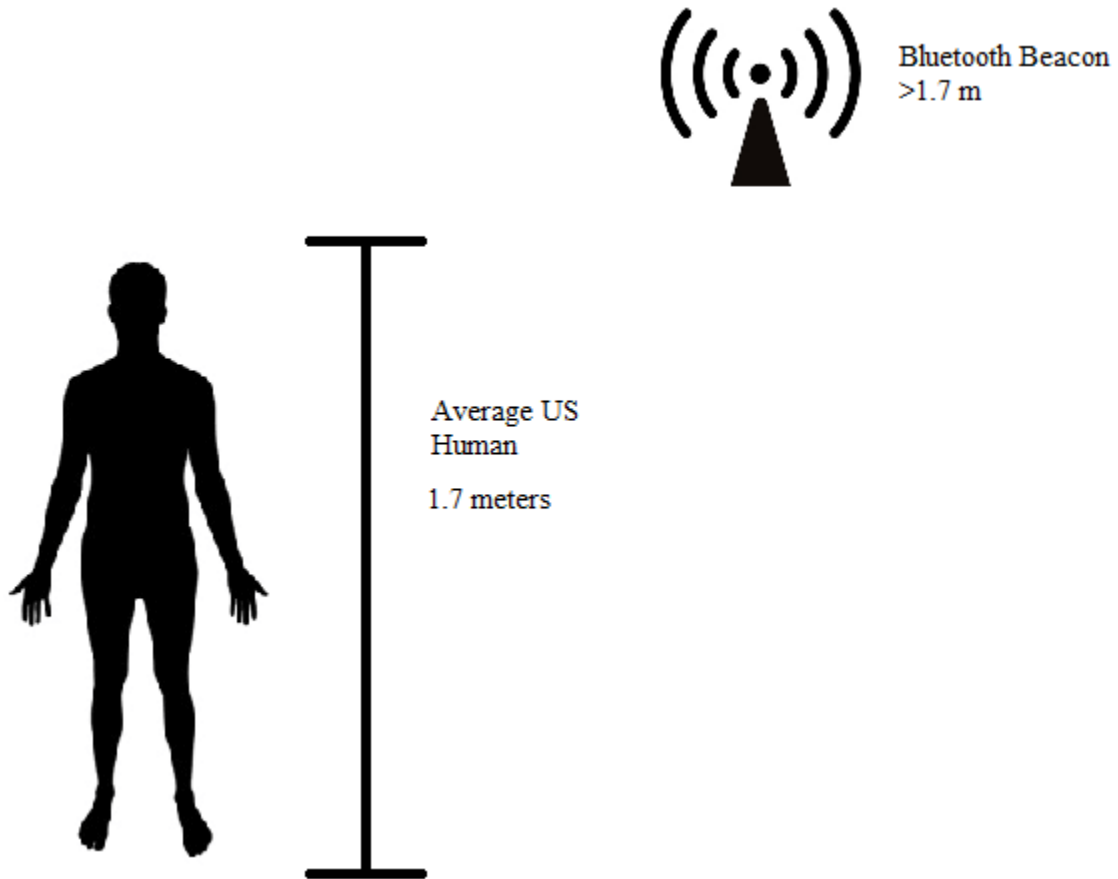


Figure 4.3.5-1 Above figure represents the proper height placement of the Bluetooth transmitter beacon module

The indoor positioning system that we are to design will utilize two potential antenna configurations contingent on the topology of the beacon module placement. Long corridors will be serviced by directional gain antenna beacons and large room such as an atrium will require an omnidirectional configuration. Referring to the radiation patterns of the unidirectional yagi antenna reference design from the previous section of this chapter, beacon modules in unidirectional gain mode would theoretically perform best placed against the wall with the radiation pattern's highest gain direction facing down the corridor. This will ensure that there is an optimal coverage of the corridor topology with the Bluetooth signal. Note that due to limitations of the Bluetooth wireless protocol and the perils of indoor radio attenuation, it is not recommended to have an isolated beacon module covering a corridor of close to, or greater than, 100 meters.

### Beacon Module Antenna Facing Towards Corridor



Figure 4.3.5-2 The above figure shown demonstrates the proper beacon placement of a unidirectional antenna in a long corridor

The omnidirectional antenna gain mode configuration of our beacon modules will be most efficiently be placed in an area where it is not easy to control the direction of radio coverage such as in large rooms or atriums. In an area where the receiver device is not constrained by a narrow passage (e.g. corridors), omnidirectional antenna configuration shall be utilized for optimum coverage. The radiation pattern characteristics of this omnidirectional configuration reduces the effective gain as compared to the unidirectional configuration due to the fact that the effective gain is spread out in multiple directions. This has implications in even having a shorter operational range than the unidirectional configuration and will require additional beacons in very large rooms. Because of this, it is advised that the placement spacing between the beacon modules is kept much below 100 meters. Reports show that at standard transmission power (0 dBm = Tx), the range in a typical office environment is around 15 meters.

The indoor positioning system's software components highly relies on the concept of trilateration which, in principle, requires RSSI (Received signal strength indication) from at least three beacon modules. Thus, it should be kept into consideration that the modules' servicing areas should overlap. A solution for this is to arrange the beacons in a similar arrangement that of cellular phone towers – hexagonal tiling. The hexagon shape offers no "blank spaces" when tessellated with each other thus treating the omnidirectional configuration's servicing area as hexagons, modules can be efficiently placed in a large room with maximum overlap and efficiency.



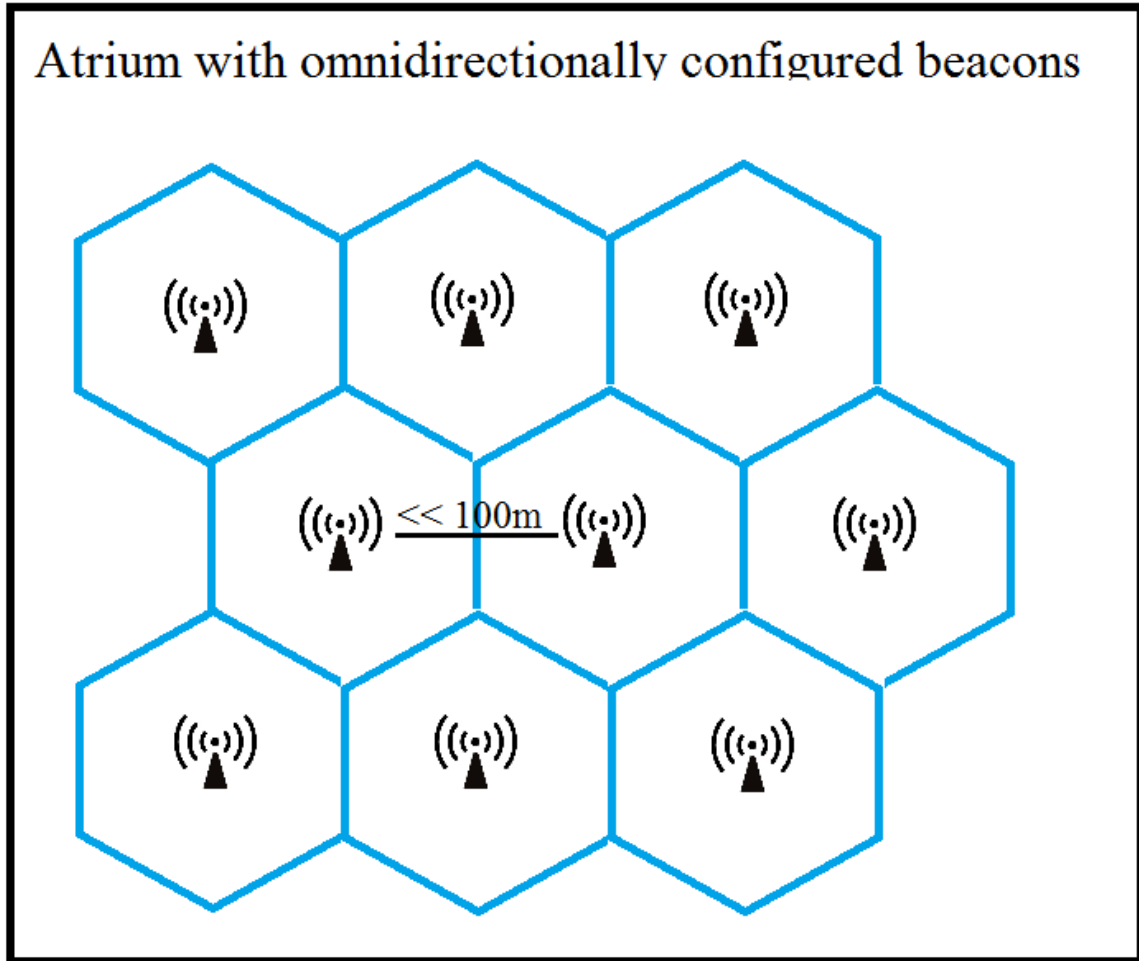


Figure 4.3.5-3 The above figure shows the hexagonal tiling configuration on beacon modules in a large, indoor open space

The placement of the beacon modules needs to also consider how the PCB trace antenna is oriented so that the gain is distributed in the proper directions. The reference designs aforementioned in previous sections show the orientation of the radiation pattern gains in conjunction with the PCB trace shapes. The Yagi antenna design requires special care in orientating the PCB in the correct direction due to the highly unidirectional gain distribution. The inverted F antenna design configuration, however, is less of an issue as it is omnidirectional; thus the orientation of the PCB board and module as a whole will be largely reliant on the Yagi Antenna PCB traces. If issues arise from the omnidirectional-configured beacons, it should be noted that the one plane of the inverted F antenna design configuration has the characteristic of having increased radiation pattern gain as compared to its other two planes.

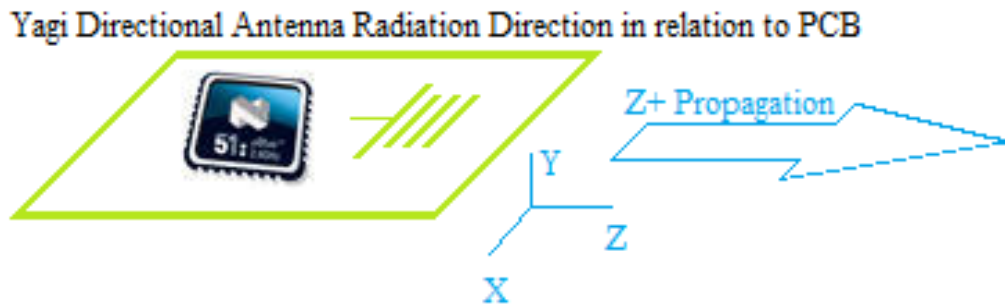


Figure 4.3.5-4 The above diagram demonstrates the direction of the unidirectional gain in relation to the PCB traces of the Yagi antenna design. Module must be oriented so that the Z+ direction is going towards the necessary serving area.

With the inherent challenge of the variability and randomness of signal wave propagation in indoor radio systems, it should be noted that it is impossible to accurately predict the performance of the beacons in the indoor positioning system. It should also be noted that radiation patterns from antennas might vary from how the module units are assembled, PCB trace imperfections, and general variability of the environment. Thus, regardless of how the system is planned, field tests of the prototype product will be required and will be analyzed in order to find the most optimal configuration of the beacon devices. It is also recommended to perform a wireless site survey in order to avoid the installation of beacons near a high-interference source such as a rogue microwave oven. All in all, this design procedure coupled with the prototype test and debugging should essentially provide optimal Bluetooth wireless coverage in the serving area.

### 4.3.6 RF Health and Safety

Any product aimed for public use must highly consider any ill effects to the user. In this case, the potential effects radio frequency radiation transmitted by our wireless beacon devices on relevant user systems must be investigated.

Usually, radio frequency radiation becomes a health issue when there are either: 1) incredible strong sources of RF transmission or 2) RF transmission in very close proximity to the human body. The first concern deals with the likes of cellular sites where large amounts of transmission power is generated and radiated from antennas which may actually cause bodily harm to those in close proximity to it. Usually these areas are marked off as being hazardous to human health and cellular tower workers take a multitude of precautions when servicing these cellular towers due to the inherent dangers of microwave radiation at high levels such as causation of glaucoma as the radiation boils away at the proteins in ones' eye like a boiled egg. The dangers of high energy radio in the microwave region comes from the microwave-absorbing properties of water molecules which cause a heating affect to biological tissue adequately exposed. The second concern is of a direct concern to any system design which use consumer wireless devices such as cellular phones which utilize WiFi, Bluetooth, and other UHF wireless protocols.

There is plenty of inconclusive research reporting on the potential effects of microwave radio radiation on the human body. These usually deal with cellular devices being held at close proximity to the skull which may be close to non-radioisolated regions of the body such as the eyeballs. Our beacon modules will not transmit in very close proximity to any human being which will greatly reduce any potential harmful effects from RF.

In addition to the low proximity of the beacon devices to the human subjects, Bluetooth Smart's standard typically has very low output power which further reduces any concerns for the health and safety of others.

Often when designing a wireless system, one must keep in mind the potential interference in the function of other devices which may, for example, be medically significant. Bluetooth Smart operates in a radio band that is pretty unregulated by the FCC (to a certain extent) and thus consideration relies on the designer and admin of the system. The indoor positioning system uses beacons with such low power that it should not pose a significant threat to existing wireless infrastructure or devices.

With the safety of the beacon module devices established, one must also consider the receiving users' devices – the Google Glass, or the smart phone. These are unlikely to pose any threat via Bluetooth due to the one-way traffic nature of this system. There is still the concern that the device might still operate in cellular bands with high powers but that is beyond the scope of this design.



There is still a lot of debate and inconclusive scientific research for the potential health dangers of RF communications at the consumer level such as in Bluetooth devices. A lot of facts about RF health and safety in consumer devices is riddled with inconclusive evidence and thus requires plenty of additional investigation.

## 4.4 Nordic nRF51822 SoC

This indoor positioning system design project utilizes an array of devices which must be able to broadcast Bluetooth LE protocol. Data being broadcasted must consist of a simple string which identifies the module in question. In addition, the module form factor should be as compact as possible and should consume the least possible power possible. The approach chosen for designing this module was to use a "system on a chip" which provides a bulk of the tasks under one IC. The Nordic nRF51822 system on a chip (SoC) provides onboard memory, an ARM processor, a 2.4 GHz transceiver, and several auxiliary functions which will be utilized for maintenance operations on the module device.

There are plenty of alternatives to this approach which were thoroughly analyzed. For example, a competing Bluetooth 4.0/LE capable system on a chip from Texas Instruments, the CC2540 provides a similar flexibility on their "system-on-a-chip" for Bluetooth LE class devices much like the Nordic nRF51822. Unfortunately, the CC2540 is burdened by higher unit costs and vastly limited free-license with limited stack size utilizing code-

composer studio. To unlock the available stack size for utilization of the TI Bluetooth SoC, a very license for Code Composer Studio development kit must be purchased. Nordic provides the same basic features of the Bluetooth SoC at less of a cost without requiring a heftily-priced license for using the hardware you bought to its full potential. Cost figures from Mouser Electronics website are shown below in figure 4.4-1.

| Product Info.           |                                                                                   |                                                                                     |
|-------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Image:                  |  |  |
| Mouser Part #:          | <a href="#">595-CC2540F128RHAT</a>                                                | <a href="#">949-NRF51822-QFAA-R7</a>                                                |
| Mfr.'s Part #:          | <a href="#">CC2540F128RHAT</a>                                                    | <a href="#">nRF51822-QFAA-R7</a>                                                    |
| Manufacturer:           | <a href="#">Texas Instruments</a>                                                 | <a href="#">Nordic Semiconductor</a>                                                |
| Description:            | RF Microcontrollers - MCU RF Bluetooth SMART SOC with USB BLE                     | RF System on a Chip - SoC 2.4GHz BT low Energy 256KB Flash 16KB Ram                 |
| RoHS:                   | Yes                                                                               | Yes                                                                                 |
| Specifications          |                                                                                   |                                                                                     |
| Operating Frequency:    | 2.4 GHz                                                                           | -                                                                                   |
| Package / Case:         | VQFN-40                                                                           | QFN-48                                                                              |
| Packaging:              | Reel                                                                              | Reel                                                                                |
| Series:                 | CC2540                                                                            | nRF51                                                                               |
| A/D Channels Available: | 3                                                                                 | -                                                                                   |
| Brand:                  | Texas Instruments                                                                 | Nordic Semiconductor                                                                |
| Data RAM Size:          | 8 kB                                                                              | 16 kB                                                                               |
| Mounting Style:         | SMD/SMT                                                                           | SMD/SMT                                                                             |
| Program Memory Size:    | 128 kB/256 kB                                                                     | 256 kB                                                                              |
| Core:                   | 8051                                                                              | ARM Cortex M0                                                                       |

|                                       |                           |                     |
|---------------------------------------|---------------------------|---------------------|
| <b>Data Bus Width:</b>                | 8 bit                     | 32 bit              |
| <b>Maximum Operating Temperature:</b> | + 85 C                    | -                   |
| <b>Minimum Operating Temperature:</b> | - 40 C                    | -                   |
| <b>Number of Timers:</b>              | 21 Timer                  | 2 Timer             |
| <b>Operating Supply Voltage:</b>      | 2 V to 3.6 V              | 1.8 V to 3.6 V      |
| <b>Processor Series:</b>              | CC2540                    | nRF51822            |
| <b>Product Category:</b>              | RF Microcontrollers - MCU | -                   |
| <b>Standard Pack Qty:</b>             | 250                       | 1000                |
| <b>Supply Voltage - Max:</b>          | 3.6 V                     | -                   |
| <b>Supply Voltage - Min:</b>          | 2 V                       | -                   |
| <b>Frequency:</b>                     | -                         | 2.4 GHz             |
| <b>Output Power:</b>                  | -                         | + 4 dBm to - 20 dBm |
| <b>Program Memory Type:</b>           | -                         | Flash               |
| <b>Sensitivity:</b>                   | -                         | - 92.5 dBm          |
| <b>Type:</b>                          | -                         | Bluetooth           |
| <b>A/D Bit Size:</b>                  | -                         | 10 bit              |
| <b>Data RAM Type:</b>                 | -                         | RAM                 |
| <b>Interface Type:</b>                | -                         | 2-Wire, SPI, UART   |
| <b>Maximum Data Rate:</b>             | -                         | 250 kb/s            |
| <b>On-Chip ADC:</b>                   | -                         | Yes                 |
| <b>Supply Current Receiving:</b>      | -                         | 9.5 mA              |
| <b>Supply Current Transmitting:</b>   | -                         | 6.3 mA              |

Figure 4.4-1 Sample comparison between the leading Nordic and Texas Instrument Bluetooth 4.0-capable SoC – taken from Mouser Electronics website. It is clear that the nRF51822 leads in having lower voltage capability, larger secondary and primary memory space to work with, and a much cheaper cost per unit.

Another solution that was considered for this design project was assemble a Bluetooth-capable module in a much more discrete manner – i.e. purchase a separate IC for the 2.4 GHz transceiver, purchase a microcontroller, etc. This method is much less feasible due to several reasons: higher difficulty in troubleshooting component failure versus just having a SoC to diagnose, relatively; much more costly compared to simply buying an SoC; less compact; likely less power efficient. The nRF51822 provides the capabilities of a microcontroller and radio transceiver integrated on a single chip.

All in all, the Nordic system-on-a-chip offers great advantages in trouble shooting the prototype beacon system, affordability, performance, and efficiency. In a future implementation of this project, a customized SoC-type device could be developed to serve the specialized purpose of low-power, short-range radio RF broadcasts; this device could be specialized in a manner that would eliminate costs from unnecessary features – e.g. not much memory real-estate required for these beacons in this system and a quick processor is also not a necessity. Further specialization of the design of these chips tailored to the needs of our devices would lead to ultra-efficient beacons leading to savings in costs and improvements in performance of the RF system.

## 4.5 Module Enclosure

This section goes over our plans to protect the pcb.

### 4.5.1 3D Printed Plastic Case

We have not determined the size of the case we will be using because we still have not determined the final size of our finished product circuit for the beacon device until we perform the final testing on a breadboard and determine the product will work as desired. But we do plan on using the 3D printer provided for engineering students to use. We plan on printing a case to enclose the pcb and provide small curved-shaped holes in the case to prevent the device from overheating. Since we will have many beacons set up in the engineering building we want to protect them from damage if it gets dropped using the protective case. We will also have a small hole for the LED that will display the battery status and device status.

The following diagram will show an idea of what we want our case to look like:

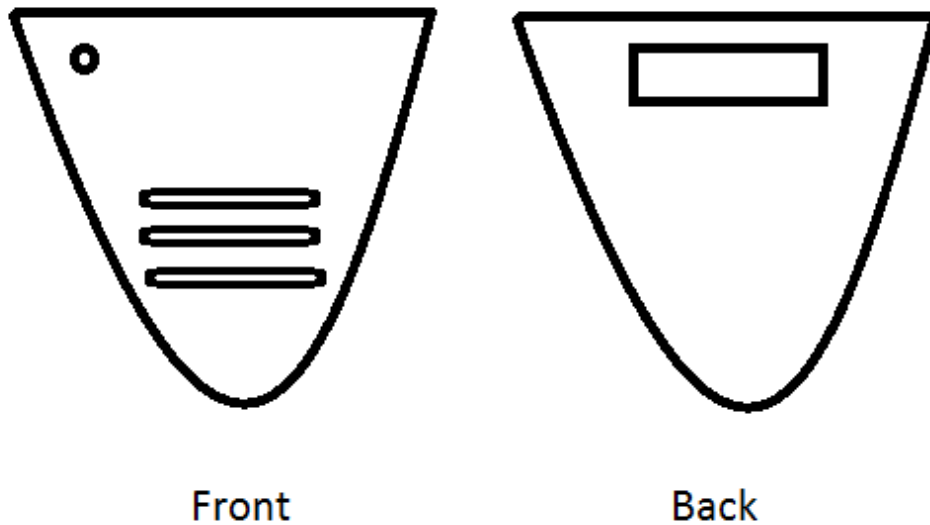


Figure 4.6.1-1 Preliminary design for 3-D printed enclosure

We believe something like this will be a nice design because we will be placing these devices on the walls of the engineering building and do not want it to look like we are placing random circuit boards in the halls and have intruders take it. For this reason we will most likely have to place signs in close proximity to the devices that say "senior design testing in progress- do not remove devices from wall."

## 5.0 Beacon Software Details

The following sections explain the software portion of this project.

### 5.1 Initial Design Architectures and Related Diagrams

This section covers the architecture and project structure of the firmware being developed to run on the Bluetooth Low Energy module and the process.

#### 5.1.1 Beacon Firmware Architecture

The Beacon firmware will be made up of two different operating modes: Advertising Mode and Configuration Mode. The Advertising mode is the mode in which the Beacon will start up and by default. The architecture for this mode can be seen in Figure 5.1.1-1. It will start off by initializing all the necessary components. First it will initialize the status LED. This LED will indicate whether the Beacon is advertising and whether the Beacon is low on battery. The configuration button is then initialized. Whenever this button is pressed, the

Beacon will restart into Configuration mode, allowing the user to set the advertising data. The firmware will then continue to initialize the Bluetooth Low Energy stack and will finally initialize the advertisement data from memory.

Once the firmware is done initializing all the different components, it will then start advertising the iBeacon protocol. Once the Beacon is advertising, it will enter the main loop which will keep getting executed until the Beacon is shut off. In each iteration of the main loop, the Beacon will carry out the same steps. First it will check the current battery level. If the battery is considered to be low, the LED color will be set to red to indicate the battery status. The loop will now check if the configuration button has been pressed. If the button was pressed, the Beacon will be restarted into Configuration Mode. If the button was not pressed, the firmware will perform some power management tasks. After this the loop will start over until the Beacon is powered off.

The second mode for the Beacon is the Configuration Mode. The high-level architecture for this mode can be seen in Figure 5.1.1-2. This mode allows the configuration module to communicate with the Beacon in order to set and display the Beacon's advertisement data. When the Configuration mode is first started, all configuration and initialization of components is carried out. First the UART communications is configured to allow communication with the configuration module. The non-volatile memory controller in order to allow the data being sent from the module to the Beacon to be written in the SoC's flash memory. Lastly, the button which is used to exit configuration mode is initialized.

Once all components are initialized and configured, the UART communication is started. This is then followed by the main loop of the Configuration Mode. This loop is run until the button is pressed or until the Beacon is powered off. The loop starts off by reading the current advertisement data from memory and sending it to the module over UART. The program then tries to receive new advertisement data from the configuration module. Once data is received, it is checked to assure that the data is valid. If it is invalid, the program will not write anything to memory. If it is valid, the program will write the new configuration to the Beacons flash memory. This allows the Beacon to use the new data once it is restarted into Advertising Mode. The loop will now check if the button has been pressed. If it has not been pressed, the main loop will simply be restarted allowing the user to further edit the advertisement data. If the button was pressed the program will break of the main loop. It will then close the UART communications and restart the Beacon into Advertisement Mode.



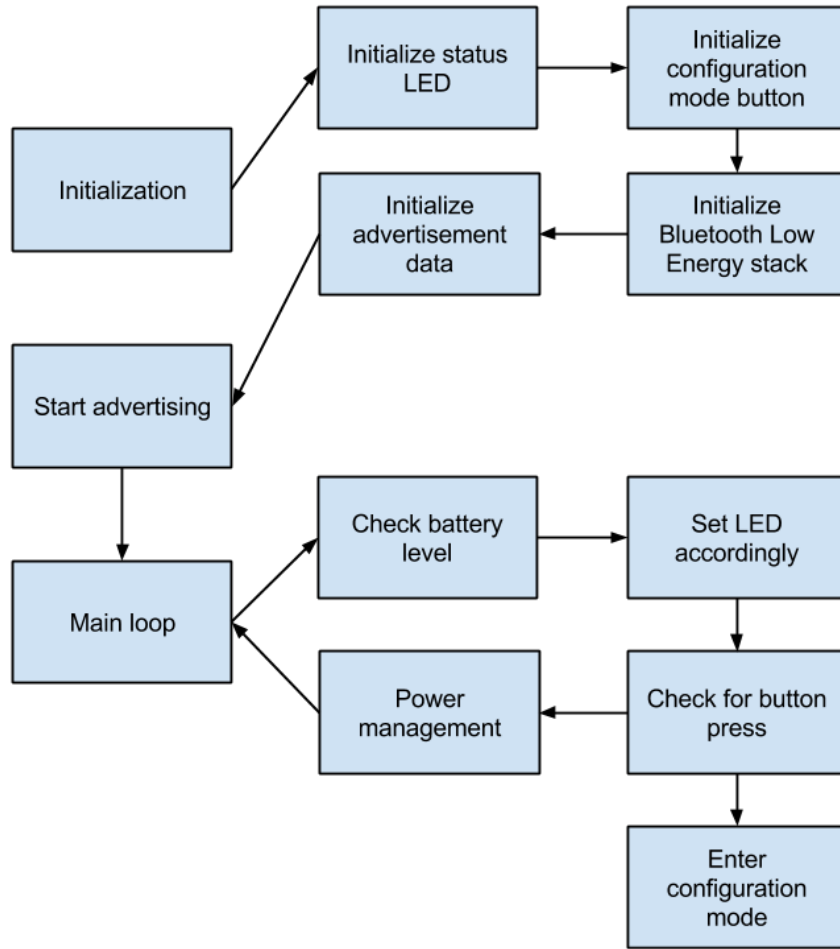


Figure 5.1.1-1: High-level architecture for the Beacon's Advertising Mode

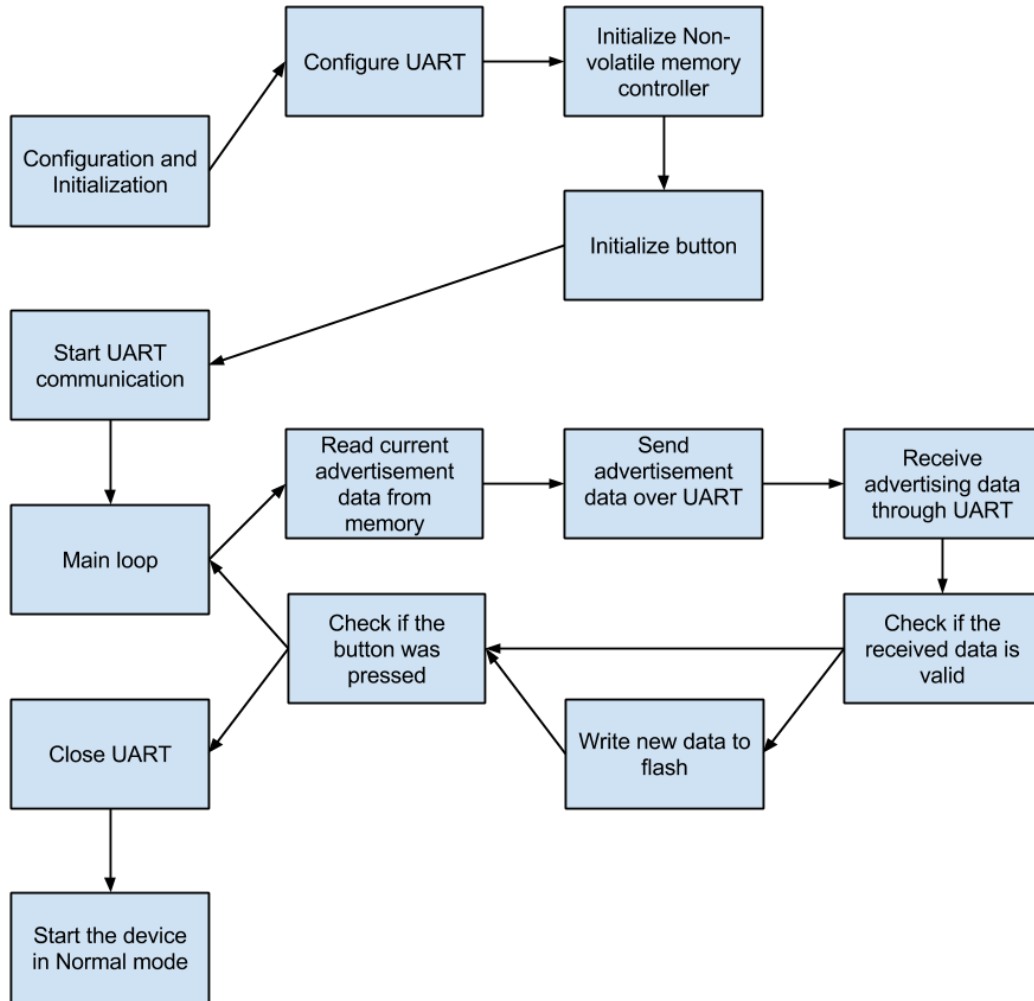


Figure 5.1.1-2: High-level architecture for the Beacon's Configuration Mode

## 5.1.2 Building and Flashing the Beacon's Firmware

In order to simplify build process for the Beacon's firmware, GNU Make will be used to build all the source files, link them to any files they use, and to flash the binary file onto the device. The Makefile will have four different base targets: build, run\_tests, flash, and clean. These build targets can be run by simply executing the command "make <build\_target>" on the root directory of the project. The "build" target will simply build and link all the source files for the firmware and generate a binary file. The "run\_tests" target will build and run all the unit tests. If the source files have not yet been built or any of the unit tests fail, the build will also fail. The "flash" target will check for an existing binary file and flash it onto the Beacon module. If no device is connected or there is currently no built binary file, the build will fail. Lastly, the "clean" target will delete any built files or binary files on the project directory created by the other targets. From three of the four base targets, the "all" target is created. This target is the default target executed when running the "make" command in the project directory. This target will build all the

source files by executing the “build” target, will build and run and the unit tests by executing the “run\_tests” target and will finally flash the resulting binary file onto the Beacon using the “flash” target. The structure of the Makefile can be seen in Figure 5.1.2-1. If any of the sub-targets being executed fail, the full build will fail and the user will be prompted with an error message. By using this structure for the makefile, a lot of time is saved by not having to manually build all the sources and not having to manually flash the binary onto the Beacon. It will also help make the compilation compatible with both Windows and Posix (Linux, Mac OSX) systems as long as the location of the SDK, the location of the GCC ARM Toolchain, and the location of the J-Link binaries are specified. All the source files will be linked to the latest version of the NRF51822 SDK provided by Nordic Semiconductors.

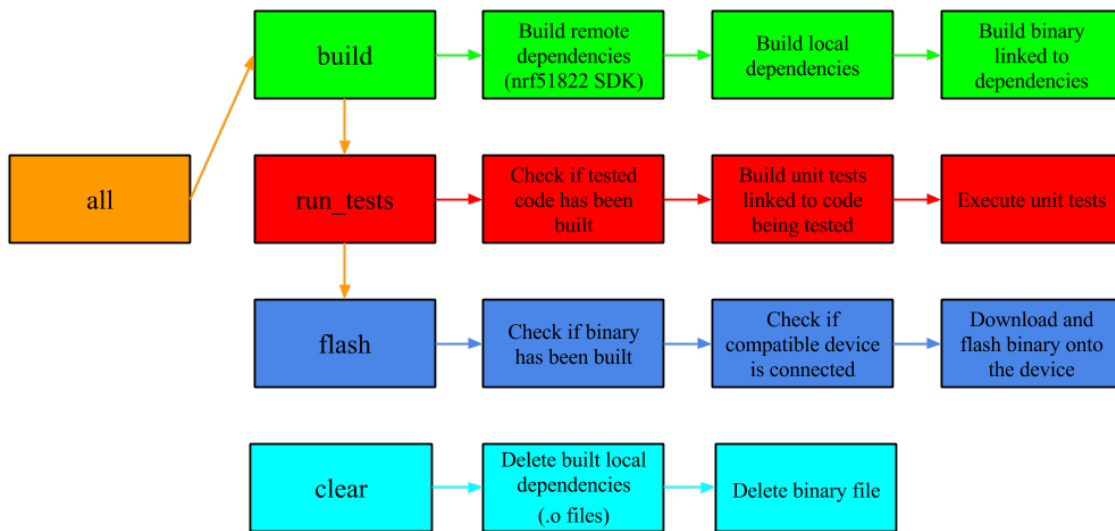


Figure 5.1.2-1: Structure of the Makefile for the Beacon’s firmware

The compiler being used to build all the sources is distributed with the GCC ARM Toolchain. This is a toolchain being actively maintained and tested by ARM to support embedded ARM processors. The specific compiler being used is called “gcc-arm-none-eabi.” The “none-eabi” section refers to the assembler used to create object code in the Embedded ABI format.

In order to flash the generated binary files onto the Beacon, it must be connected to the computer through the Segger J-Link Lite programmer provided with the NRF51822 development kit. This connection between the Beacon and the computer will allow the J-Link program to download the necessary binary files onto the device and then properly flash them to the memory of the Beacon.

## 5.2 Configuration/Debugging Module

In order to facilitate the configuration of the Beacons, a separate module was designed. This device is able to display and configure the different parameters (UUID, Major, Minor, and calibration RSSI) of each Beacon. This can be achieved in two different ways: through a separate hardware module and though an Android application.

An overview of the Hardware-based configuration/debugging module is shown in Figure 5.2.1-1. The module is built around an MSP430 due to its ease of use and widespread support. The user can define a new configuration for the Beacon using the input of the module. This new configuration will be sent to the Beacon over a UART connection from the MSP430. The current values being used for advertisement by the Beacon will also be read through the UART connection. The MSP430 will then write those values on the display.

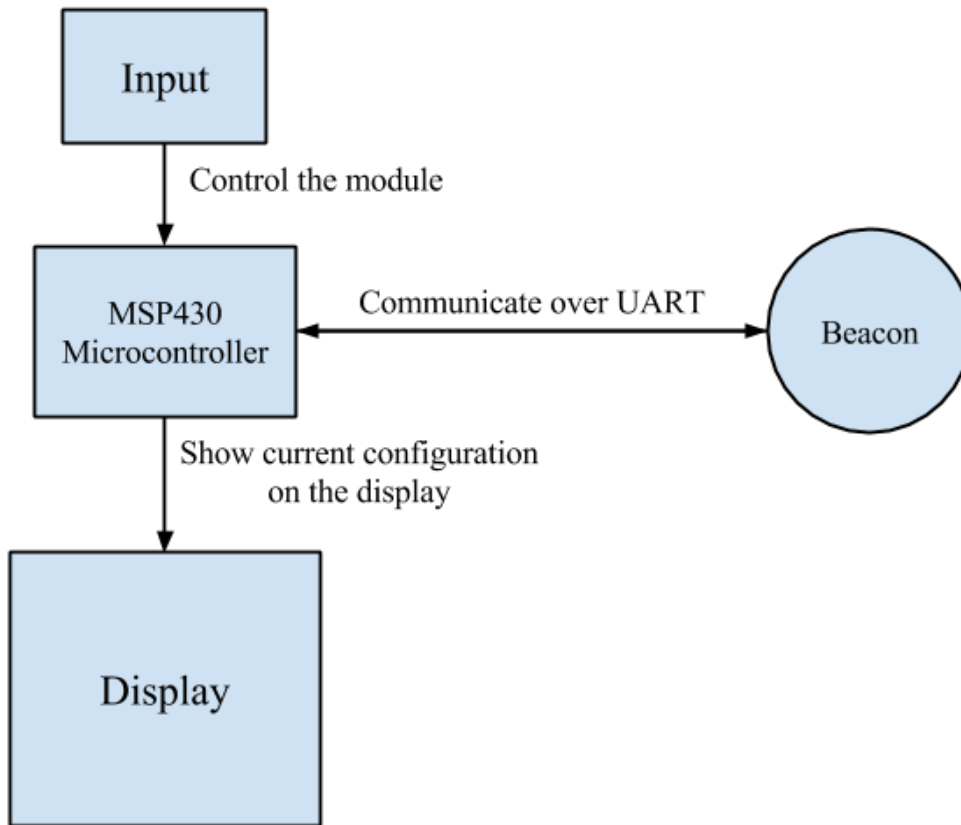


Figure 5.2.1-1 Overview of the hardware-based configuration/debugging module

An overview of the Android-based module configuration/debugging can be seen in Figure 5.2.1-2. This is based around an Android application running on a Android device. The Android device will establish a Bluetooth (not Low Energy) connection with the Beacon. The application will then use this connection to read and display the current values being used for advertising on the Beacon. The user will also be able to create a new configuration on the Android application and then send it to the Beacon over the Bluetooth connection. The Android application will also allow for more features like storing configurations on the device and general Beacon administration by interacting with the database.

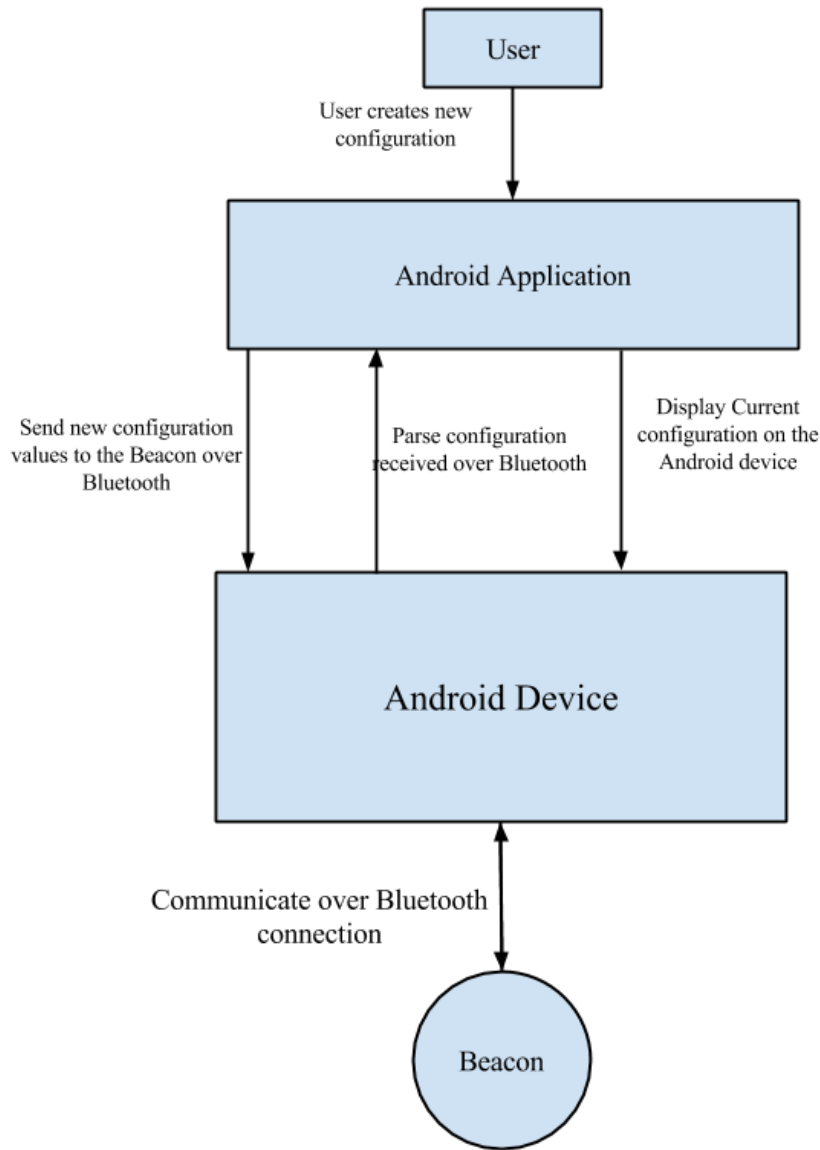


Figure 5.2.1-2 Overview of the configuration/debugging Android application

One advantage of creating the hardware module is that it will be physically connected to the Beacon, allowing it to access the exact values being used by the Beacon and ensuring a more reliable connection. The Android application will use a Bluetooth connection, which can lead to some complications when trying to connect to a specific Beacon when there are many other Beacons nearby. The advantage of the Android application is that it will allow for more features to be implemented, especially by integrating the database storing building information. The hardware-based module will have a limited feature set that will be harder to update. The hardware module will also be more expensive due to the cost of hardware whereas the Android application will only require software work.

## 6.0 Application Software Design Details

### 6.1 Mapped Building Database

The mapped building database module is responsible for holding all of the data needed by the system to run effectively. The module takes a file from an external source and parses the information into the data structures that make up the database. Technically it is not a database in traditional sense of having a large amount of data being stored on a server, but rather the word database is used because it holds information that the system grabs at runtime and uses. Once the application exits, the data is erased, because it is only held in RAM.

#### 6.1.1 Purpose of Database

In order for the user to be able to navigate through a building, the application must have access to the building information that is important for navigating. Important aspects of traversing through a building include room names, number of floors, beacon positions, and a collection of virtual nodes that map out the structure of the building, which is used within the pathfinding module. The database will be responsible for holding all of this building information and for giving the information whenever the application requests it.

When the user enters a building with the application running, a file containing all of the important building information will be retrieved from an external source. The specifications for the file will be discussed in the next section, Parsing JSON Building Information, and will follow the JSON format. Once the file is received, it will be read with the information contained in the file being stored within the database of the application. The term database is used loosely related to the navigation system, because it only holds data temporarily while the user is within the building and using the application. The database is created when it parses the JSON file and is queried when either a beacon location is needed or the user's desired destination is needed. Also, it will not have the range of querying operations that a typical database would have. It will only allow operations that are specific to the navigation system's needs, which makes it somewhat limited.

One of the main purposes of the database is to give the beacon locations to the system whenever it is requested. Each beacon will contain a preset, unique identification string (UUID). Upon input of the UUID string, the database will give back the three-dimensional coordinates of the beacon requested. The positions of the beacons are needed by the trilateration module for the calculation of the user's position.

Another purpose for the database is to hold the tags of specific nodes that are in the building. Within the pathfinding module, some virtual nodes will contain special names or tags that describe the particular node and its role within the building. Tags can be viewed as words that describe the node's location within the building. For example, if the building

being mapped is a school, one of the nodes might be at the entrance to a classroom. Some tags for that node might be the room number of the classroom, the name of the teacher that teaches within the classroom, or the purpose of that classroom, such as biology lab or chemistry lab. In order for the mapping from tags to nodes to work properly, each node must have tags that are unique from any other node within the building, so that a unique tag maps to one and only one node. Only nodes that mark building landmarks need to have tags. The rest of the nodes do not need tags, because they are only used to traverse to the landmark nodes as sort of a middle point. Some examples of landmark nodes are nodes that are entrances to rooms, elevators, staircases, or any node that can be used as a possible destination of the user. Once the tag-to-node mappings are stored in the database from the JSON file parsing, the database will be able to accept a tag as an input and return the node mapped to that tag. This specific query will be used to determine where the user would like to go within the building. The application will ask the user where they would like to go and the user will be able to input tags for their desired destination. If the tag cannot be found in the system, then the user will be notified. The user will also be able to view all of the tags or rooms within the building in a list format, so that if the user's given tag does not match any in the database, they can still select the room they want from the list if somehow they did not input the right tag.

Not only will the database hold the mappings of the tags to nodes, it will also hold the collection of nodes used by the pathfinding module. Discussed in section 6.4.3 Graph Design Comparison for Real World Buildings, a proper structure for delivering the nodes must be used so that the pathfinding module can perform efficiently. The database module's structuring of the nodes will be determined by the way in which the pathfinding module needs the nodes, so that no conversion is needed to be done before passing the nodes from the database module to the pathfinding module. Once the database gives the pathfinding module the collection of nodes, it will not need to be done again unless the user moves to a new building, because there is one node collection per building.

Figure 6.1.1-1 outlines the basic design of the database. The blue lines represent inputs that the database needs in order to perform properly. The database is created from the input of the JSON file, which the JSON Parser reads into memory and stores the data of the beacon information and node information into the structured database. The Beacon Information block holds the mapping of beacon UUIDs to beacon positions. The Node Information block holds the mapping of tags, which are strings, to nodes. The module that inputs each data is described in the figure, as well as the module that receives the output. Currently, the beacon information is only needed by the trilateration module to find the user's real world position, while the node information is used just to find the user's desired destination.

Ultimately, if anything within the navigation system is needed to be stored, it will be held in the database module. Currently, there is just the beacon and node information stored, but if any other data is needed to held for the system to work properly, it can be added to the database module. The next section will discuss the formatting behind the JSON file that is inputted into the database and the reasons why JSON was picked as the best formatting option for holding the building structure information.

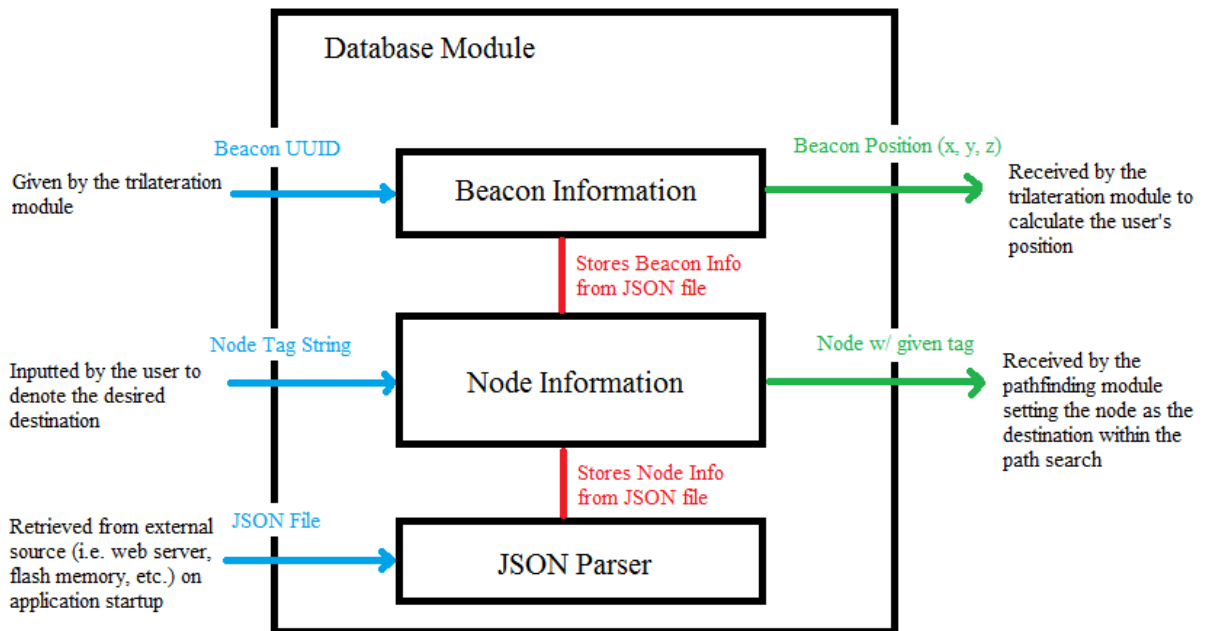


Figure 6.1.1-1 Database Module Design Overview

## 6.1.2 Storing the JSON Building Information

In order for the database module to be filled with correct data, a file will be responsible for holding the initial data and passing it to the module to be parsed. The formatting of the file will be an important aspect, because it will directly translate to the ease of parsing the data into the needed format and structure. Although the file, as long as it holds all of the information needed, could be formatted in a number of ways, a simple design will allow for easier construction of the file and easier parsing by the database module parser.

There are many different formats that can be used for the file that holds all of the building information. One approach is to use comma-separated values (CSV) to store the needed information. Within a CSV file, each piece of information is delimited by a character, typically a comma. Using a CSV format is beneficial for information that can be broken into rows and columns similar to what is found in a relational database. The downside is that the information needing to be stored for the navigation system to run properly does not necessarily relate over the entire file. In other words, the beacon information, node information, and node tags would each have different columns and parsing instructions. This becomes an issue, because visually within the file, all one would see is values and commas with no way of knowing what each value is without looking into the parsing algorithm, which would store those values in meaningful locations. Because the file must remain simple and readable, CSV would not be the best format to use to store the building information. A better approach would be to use Extensible Markup Language (XML), which is both human and machine readable. XML groups information into elements and attributes providing a simplistic way to hold data. For example, a virtual node within the



pathfinding module might have an attribute giving the name or number of the node and an element nested under the node element for the position, which might have more nested elements for the x, y, and z coordinates of the node. XML would be a good format to choose for the building information file, but it is very wordy and requires a lot of extra syntax information in order to be able to read the relevant building information into the database. This is where JavaScript Object Notation (JSON) comes in. JSON can be seen as an alternative to XML that uses attribute-value pairs to hold the information. The plus side of JSON is that it requires less extra syntax. It's not extensible like XML, but still does everything that is needed by the navigation system to work properly and it is easier to construct the file with the necessary building information than would be if XML were used. Because of its simplicity, JSON is the perfect choice of format to hold all of the building information. It will allow the owners of each building to write their own files containing their own building's information, which would be required if the navigation system were to be widely used.

The file must contain all of the data that is specific to each building. One file will be used for one building, so that when the user walks into a building with the application running, the file containing that building's information will be retrieved. This building information includes all of the nodes that are used by the pathfinding module, the real-world beacon locations and their unique UUIDs, and tags for the landmark nodes. The tags are separate from the nodes themselves, because not all of the nodes have tags. Each node within the file will also contain information. Some of this specific node information includes the node index, the node position, and possibly information on whether or not the data can be walked through or not depending on the method in which the pathfinding module uses the nodes. This information does not have to be explicitly stated within the file, but can be implicitly determined based on the parsing algorithm within the database module. For instance, a visual representation of the building being mapped can be stored in the file with an array of ones and zeroes where the ones represent walkable areas within the building and zeroes represent unwalkable locations in the building. This approach is similar to the grid approach in the pathfinding module. It provides a quick and easy way to display the building's structure without having to include a lot of information. The distance between the nodes would be stated in the file so that every node would be the same distance apart from its adjacent neighbors. Figure 6.1.2-1 shows how this group of ones and zeroes would appear in the JSON file. On the left side of the figure is the grid approach, while on the right the adjacency list approach that explicitly states each nodes location. The list approach is an abbreviated version of what would actually be in the JSON file if it were used, because it takes up a significant amount of space. One can see that the grid approach visually shows the example building's structure, while the other approach does not. Though the other approach would work, the grid format will be a much easier format to create and edit during testing. In the figure, the red numbers represent the walkable paths and are stored as the ASCII character for the number one. The non-walkable areas are stored as the ASCII character for the number zero. The letter E represents an elevator and the letter S represents a staircase within the building. These two act very similar within the navigation system, because they both lead the user from one floor to another. They will also be used within the floor sequencer in the pathfinding module to determine the correct path between floors to take. Notice that the elevator leads to the same indexed node on the second floor

(technically floor one, because the floors start at zero). Also notice that there is not a direct staircase node overhead of the staircase node on the first floor. This is because staircases do not have to lead to the same location on a different floor. In fact, most staircases end at a different horizontal location compared where they started, unlike elevators, which only move vertically. In this case, the staircase leads to two locations, because it splits in the middle. This information cannot be determined directly from what is shown in Figure 6.1.2. Extra information is needed in order to know that the first floor stairs leads to both stair nodes on the second floor, because if there were another floor, then maybe one of the stairs goes from the second to third floor, while the first floor stairs node only leads to the other stairs node on the second floor. The navigation system must be able to handle these special cases in order to be effective and, in order to handle them, more information is needed to be added to the JSON file.

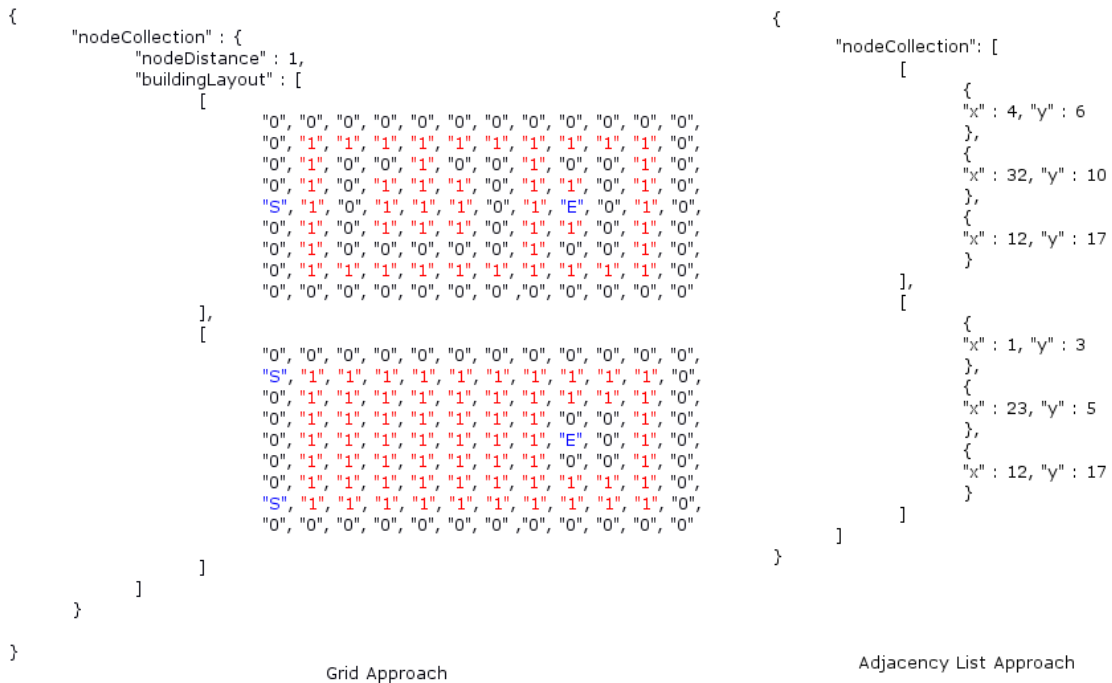


Figure 6.1.2-1 Two approached on storing node locations.

In order to have pathfinding between floors, the elevators, staircases, and each of their connections to other elevators or stores must be known. This will make up another component in the file, which is shown in Figure 6.1.2-2 labeled “stairs/elevators”. This section simply goes through every floor and lists each elevator and staircase and each of the others they are connected to. The label “location” is the specific node location for the either the elevator or staircase, while the “connections” label is an array of all of the node locations that are connected to that elevator or staircase. Also, notice that the stairs and elevators that are on the same floor as each other are also listed as being connected to one another. This is needed by the pathfinding module to determine all of the paths that can be taken. Elevators and staircases on the same floor must be connected, because one possible path could be to walk from one to another across the floor, which might occur if one set of stairs goes from the first to the second floor and a different set of stairs in a different location goes from the second to third floor. The path on the second floor would start at the

staircase the user just exited from and the destination node would be the staircase that leads to the third floor. For this reason all elevator and stair nodes must be considered connected to one another in the JSON file.

```

"stairs/elevators" : [
  {
    "location" : {
      "floor" : 0,
      "index" : 57
    },
    "connections" : [
      {
        "floor" : 1,
        "index" : 57
      },
      {
        "floor" : 0,
        "index" : 49
      }
    ]
  },
  {
    "location" : {
      "floor" : 0,
      "index" : 49
    },
    "connections" : [
      {
        "floor" : 1,
        "index" : 13
      },
      {
        "floor" : 1,
        "index" : 85
      },
      {
        "floor" : 0,
        "index" : 57
      }
    ]
  }
],
{
  "locations" : {
    "floor" : 1,
    "index" : 57
  },
  "connections" : [
    {
      "floor" : 0,
      "index" : 57
    },
    {
      "floor" : 1,
      "index" : 13
    },
    {
      "floor" : 1,
      "index" : 85
    }
  ]
},
{
  "location" : {
    "floor" : 1,
    "index" : 85
  },
  "connections" : [
    {
      "floor" : 0,
      "index" : 49
    },
    {
      "floor" : 1,
      "index" : 57
    },
    {
      "floor" : 1,
      "index" : 13
    }
  ]
}
]
}

```

Figure 6.1.2-2 Elevator and Stairs Component within JSON file

Figure 6.1.2-3 shows the JSON formatting for the tags component that will hold all of the tags that will map to nodes. The specific node location must be given and a list of the tags/strings that identify that particular node. In this particular case, there are only three special nodes of interest. the lab, the break room, and another entrance to the same breakroom. In Figure 6.1.2-1, the break room can be seen as the narrow room that has the elevator inside of it. Because there can be multiple doors to the same room, as seen with the break room, the system will have to be able to handle finding the path to the room that takes the closest entrance to the user.

```

"tags" : [
  {
    "floor" : 0,
    "index" : 29,
    "nodeTags" : ["Lab", "100"]
  },
  {
    "floor" : 0,
    "index" : 32,
    "nodeTags" : ["Breakroom", "Break", "Soda"]
  },
  {
    "floor" : 0,
    "index" : 80,
    "nodeTags" : ["Breakroom", "Break", "Soda"]
  }
]

```

Figure 6.1.2-3 Tags component within JSON file

The next component of the JSON file consists of the beacon location mappings. Each beacon is uniquely identified by a major and minor number that is part of its UUID. The JSON file will hold each beacon's major and minor number as well as the three-dimensional coordinates of the physical beacon within the building. Both the beacon coordinates and the node coordinates must be on the same coordinate mapping so that if a node is located at (4 m, 5 m, 3 m) within the building and a beacon is located at (4 m, 5 m, 3 m), they will be at the same exact spot within the building. Figure 6.1.2-4 shows three total beacons for the example building. The major number is a grouping of the beacons according to the floor they are on. Thus, there are two beacons on floor 0 and one beacon on floor 1. The minor number can be seen as the equivalent of the index number for nodes as it uniquely identifies the beacon on the major number's particular floor. The x, y, and z values just give the exact location of the beacon within the building.

```

"beacons" : [
  {
    "major" : 0,
    "minor" : 0,
    "x" : 4.3,
    "y" : 4.4,
    "z" : 2.0
  },
  {
    "major" : 0,
    "minor" : 1,
    "x" : 8.5,
    "y" : 4.3,
    "z" : 2.0
  },
  {
    "major" : 1,
    "minor" : 0,
    "x" : 6.5,
    "y" : 1.0,
    "z" : 6.3
  }
]

```

Figure 6.1.2-4 Beacon component within JSON file

The Android API will be used for the actual JSON reading and parsing into structures within the system's database. This will be used because it provides an easy, ready-to-use classes for parsing JSON files, so that the parsing algorithm will not need to be designed.

### 6.1.3 Database Structures

The database contained on the application will consist of several components previously discussed in the last section. These components must be taken from the JSON file and stored into data structures within the data allowing for easy access from other modules in the system to grab the information they need.

The information containing all of the nodes and their positions will be broken down into a three-dimensional array if the grid approach is used by the pathfinding module or an adjacency list if the graph approach is used. Figure 6.1.3-1 shows all of the data structures and components that will be held within the database. The information holding the connections between the elevators and staircases in the building will be stored as an adjacency list. A hashmap will be used to hold the tag-to-node mappings. It's important to note that because multiple nodes can lead to the same room, a list of the nodes that have the same tag will be given. It is also important to note that the hashmap will not directly contain the nodes, because the physical nodes are being held in the structure holding the collection of nodes, but will hold the location of the nodes according to the collection of nodes. So if a tag is inputted, the locations of the nodes will be grabbed and used as an index within the collection of nodes to retrieve the real node. Then, this node will be returned to the caller needing the node. The extra indexing of the node collection will be done internally within the database module, so the caller will not need to do any other processing to get the list of nodes. A two-dimensional array will be used for holding all of the beacons' coordinates within the building. The row of the array will be indexed by the major ID of the beacon, while the column of the array will be indexed by the minor ID of the beacon, so that upon input of the major and minor ID's, the beacon's location can be returned. All of these structures within the module will be initialized by the parsing of the JSON file, so that the JSON file will give all of the required information for the system to run effectively.

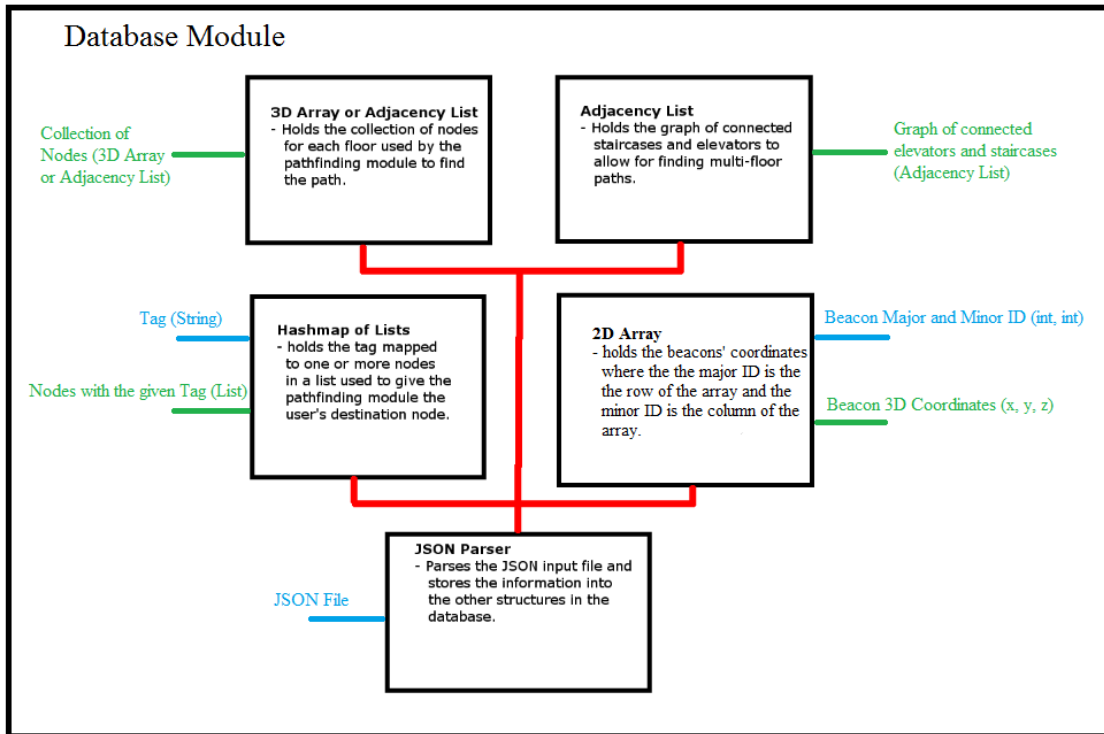


Figure 6.1.3-1 Database Structures within the Database Module along with inputs (Blue), outputs (Green), and initializers (Red) of each structure.

## 6.2 Beacon Detection Library

This section explores the library being built around Android's Bluetooth Low Energy SDK that will be used to facilitate the process of scanning, filtering and interpreting the Beacons modules.

### 6.2.1 Bluetooth Low Energy within the Android Operating System

Even though it had already been supported by the hardware in the past, Bluetooth Low Energy support wasn't introduced into the Android operating system until the release of Android 4.3 (Jellybean). With this update, Google introduced the `startLeScan` method into the `BluetoothAdapter` class in Android which allows the developer to scan for Bluetooth Low Energy devices and consume the data being advertised by those devices. Although this allowed for some Bluetooth Low Energy support, it was still behind the Bluetooth Low Energy APIs offered by other mobile operating systems like iOS and Windows Phone 8. Since Android only allows for scanning, it means it can only act as a Central device and does not support acting as a Peripheral device. With the recent release of Android 5.0 (Lollipop), Google has added software support for the emulation of multiple simultaneous Peripheral devices.

In order to use Bluetooth Low Energy in an Android Application, two system permission are need to be declared in the application manifest: `android.permission.BLUETOOTH` and `android.permission.BLUETOOTH_ADMIN`. If the permissions are not declared, the operating system will not allow the application to access the Bluetooth hardware. All Bluetooth activity is done through the `BluetoothAdapter`. This object is shared across all applications which are using Bluetooth.

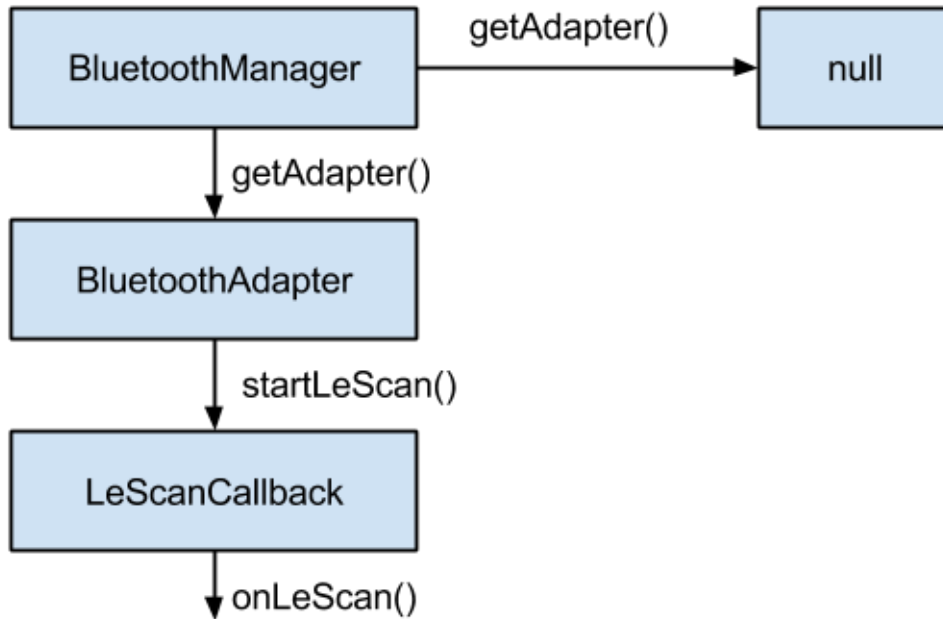


Figure 6.2.1-1 Architecture of Bluetooth Low Energy in Android

Figure 6.2.1-1 shows a high level view of the architecture of Bluetooth Low Energy in Android. With the permissions correctly set in the applications, the `BluetoothAdapter` can be extracted from the `BluetoothManager`. If the device doesn't support Bluetooth or Bluetooth is not enabled, the `BluetoothManager` returns `null`. In order to define what happens whenever a device is found, a `LeScanCallback` object is created with the `onLeScan` method overridden. Whenever a Bluetooth Low Energy device is found the `onLeScan` method from the `LeScanCallback` object is called. In order to start the scan, the `startLeScan` method from the `BluetoothAdapter` is called with the `LeScanCallback` object passed in as a parameter.

## 6.2.2 Filtering Miscellaneous BLE Devices

While scanning for Beacons, all Bluetooth Low Energy devices are going to show up in the scan. This requires that the application filter out these miscellaneous Bluetooth Low Energy devices. In order to accomplish this, the advertisement data for the device is analyzed to find the expected pattern for Beacons. This check is performed when the data

scanned in is being transformed into a Beacon object. The method uses a Regex expression to check the advertisement data. If the check fails, a Null value is returned instead of the Beacon object telling the application to ignore the data scanned. A diagram representing this process is shown in Figure 6.2.2-1.

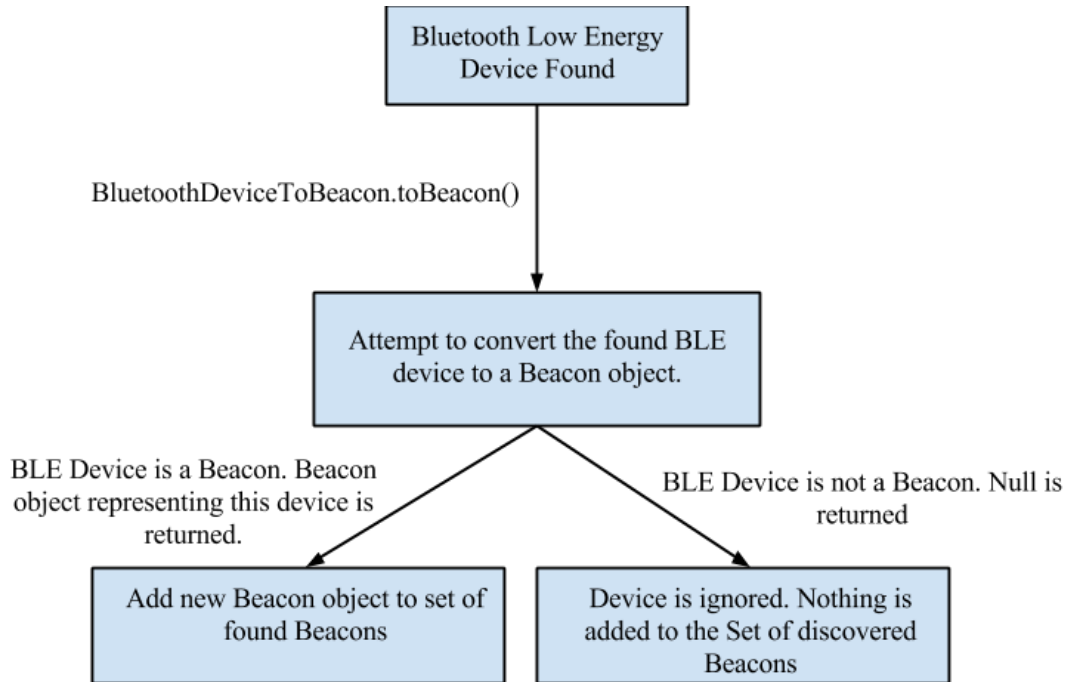


Figure 6.2.2-1 Diagram for filtering miscellaneous Bluetooth Low Energy devices

With each device scanned, three parameters are received: a BluetoothDevice object, the RSSI value, and a byte array with the advertisement data. The advertisement data is converted from a byte array to a String so the Regex expression can be used. The Regex expression being used is the following: `^0201[a-f0-9]{2}1aff[a-f0-9]{4}0215[a-f0-9]{42}.*`. The “0201” section is found in all Beacons. This is followed by “[a-f0-9]{2}” which matches by any 2 hexadecimal digits which represent Flag values in the advertisement. The following section, “1aff” is also constant across all Beacons. This is followed by the company identifier “[a-f0-9]{4}” which is made up of any four hexadecimal digits. The next three digits, “0215”, correspond to the iBeacon advertisement indicator and is constant across all Beacons. Lastly, the “[a-f0-9]{42}” section ensures that there are forty-two hexadecimal digits. This corresponds to the UUID, Major value, Minor Value, and the calibration RSSI value.

### 6.2.3 Deriving Distance from the Beacon's Signal

In order to derive the distance between a Beacon and the user, the calibration RSSI value found in the iBeacon protocol will be used. This calibration value is the measured strength of the signal emitted by the Beacon at one meter. This value can then be used in conjunction with the measured Tx power value in order to derive an estimated value for the distance of the Beacon. This is done by getting the ratio of the calibration RSSI value to measured TxPower value. These values are measured in dB and therefore must be converted to the



linear ratio. This converted value is then known to be the distance in meters of the Beacon. An example of this process can be seen in Figure 6.2.3-1. This process is a naive implementation and will be slightly modified in the final implementation.

$$\begin{aligned} \text{Calibration RSSI value} &= -59 \text{ dB} \\ \text{Measured TxPower value} &= -45 \text{ dB} \\ \text{Ratio(dB)} &= \frac{-45}{-59} = 0.7627 \\ \text{Ratio(linear)} &= 10^{0.7627} \\ \text{Distance(meters)} &= \text{Ratio(linear)} * 1 = 5.7904 \end{aligned}$$

Figure 6.2.3-1: Example for calculating the distance of a Beacon using the calibration RSSI value and the measured TX Power value

In order to increase the accuracy of the derived distance, a couple of measures will be taken. First, the application will scan for Beacons for a set amount of time. Each time a Beacon is scanned within that allocated time period, the measured value of the TX Power will be stored. Later on when the distance for that specific Beacon is being calculated, an average of all the measured TX Power values will be calculated. This will help eliminate the effect of spikes in the measured values giving a more accurate value of the measured distance. Another way to increase the accuracy will be to test the device being used to scan the Beacons at multiple distances and using the gathered data to create a best-fit curve for the specific device being used. This method will help increase the accuracy for the given device.

Problems with the calibration RSSI value arise when using different devices. When measuring the signal strength of the same Beacon using different Android devices, different signal strengths are measured at the same distance. This can be due to the difference in Bluetooth hardware being used in each device. When the iBeacon protocol was developed by Apple, this was not an issue for their use case since iBeacons are intended to be used solely by iOS devices which Apple can make sure have equal or similar Bluetooth hardware which would measure the same signal strengths. This issue will not affect the project, as it is currently designed to only work with Google Glass, but if it were to be scaled to support other Android devices and iOS devices, the derived distances using the calibration RSSI value would not be accurate for all different devices. This would then lead to deriving an inaccurate location of the user.

## 6.3 Trilateration Module Design

The trilateration module will be responsible for pinpointing the user's location according to the beacons that are transmitting Bluetooth signals to the user's Android device. The module will be a computationally extensive part of the system due to the mathematical calculations involved with determining a user's position from only having a few distance measurements. The actual trilateration process involves determining a point in three-dimensional space according to the distances between at least three other known points in the same three-dimensional space. Relating the trilateration process to the beacon

navigation system, the point that needs to be determined is the user's location within the building. The known points are the known beacon locations positioned around the building with the distances being determined from the beacon signal strengths that reach the user's phone. Trilateration works effectively in pinpointing a specific location by constructing three spheres with the centers of the spheres being the three known locations. Then, these spheres will intersect one another at a specific three-dimensional coordinate.

There are few methods of solving for the user's position in the building with each one calculating a more accurate estimate, while also being more involved. The first, simplest method of determining the user's position uses the exact linear model, which is the most direct method and involves using at least four sphere equations coming from the four beacon positions and their approximate distances from the user along with a reference beacon to get the user's  $x$ ,  $y$ , and  $z$  coordinates. The next, more accurate model is the linear least squares model, which uses the results from the exact linear model and applies matrix algebra to determine a solution with less error. The last, most complex model is the nonlinear least squares model, which involves using the sphere formula along with derivatives and summations in order to reach the solution with the smallest error. Because coming up with an efficient trilateration algorithm is a very involving process containing numerous mathematical calculations, Yu Zhou's nonlinear trilateration algorithm will be used [24]. His algorithm provides an efficient and robust solution to the trilateration problem that can be used on real time systems due to the use of only standard linear algebra techniques while keeping the programming logic to a minimum. The algorithm also went through significant testing comparing the difference in accuracy using a smaller set of beacon measurements versus using more beacons to determine the unknown locations. The results are promising, showing that the more beacon measurements used, even if not completely accurate, ultimately improve the accuracy of the estimate. Below are a list of the equations used in the trilateration algorithm as well as the programming steps involved. One will notice that step 7 involves choosing from two answers that are produced. Typically, one of the answers is unreasonable, because it is unusually far from the previous location calculated. For this reason, the solution that is closer to the previous calculated user location will be chosen. Within the list of equations are some important variables:  $p_0$  is an estimate of the user position and the end goal,  $p_i$  is the pre-mapped position of the  $i$ th reference point,  $r_i$  is the measured distance between  $p_0$  and  $p_i$ ,  $I$  is the identity matrix,  $N$  is the number of reference points/beacons used,  $Q$  is the orthogonal matrix of  $H'$ , and  $U$  is the upper diagonal matrix of  $H'$ .

$$\begin{aligned}
1) a &= \frac{1}{N} \sum_{i=1}^N (p_i p_i^T p_i - r_i^2 p_i) & 2) B &= \frac{1}{N} \sum_{i=1}^N [-2p_i p_i^T - (p_i^T p_i)I + r_i^2 I] \\
3) c &= \frac{1}{N} \sum_{i=1}^N p_i & 4) f &= a + Bc + 2cc^T c \\
5) f' &= [f_1 - f_n, \dots, f_{n-1} - f_n]^T & 6) H &= -\frac{2}{N} \sum_{i=1}^N p_i p_i^T + 2cc^T \\
7) H' &= [h_1 - h_n, \dots, h_{n-1} - h_n]^T & 8) q^T q &= -\frac{1}{N} \sum_{i=1}^N p_i^T p_i + \frac{1}{N} \sum_{i=1}^N r_i^2 + c^T c \\
9) & \left[ \left( \frac{u_{12}v_2}{u_{11}u_{22}} - \frac{v_1}{u_{11}} \right) + \left( \frac{u_{12}u_{23}}{u_{11}u_{22}} - \frac{u_{13}}{u_{11}} \right) q_3 \right]^2 + \left[ \frac{v_2}{u_{22}} + \frac{u_{23}}{u_{22}} q_3 \right]^2 + q_3^2 = q^T q \text{ where } v_k \\
& \text{denotes the } k\text{th component of } Q^T f' \text{ and } u_{kj} \text{ the } (k, j) \text{ entry of } U. \\
10) & \left[ \frac{v_1}{u_{11}} + \frac{u_{12}}{u_{11}} q_2 \right]^2 + q_2^2 = q^T q \\
11) q_1 &= \left( \frac{u_{12}v_2}{u_{11}u_{22}} - \frac{v_1}{u_{11}} \right) + \left( \frac{u_{12}u_{23}}{u_{11}u_{22}} - \frac{u_{13}}{u_{11}} \right) q_3 & q_2 &= -\frac{v_2}{u_{22}} - \frac{u_{23}}{u_{22}} q_3 \\
12) q_1 &= -\frac{v_1}{u_{11}} - \frac{u_{12}}{u_{11}} q_2 & 13) q_1^2 &+ q_2^2 = q^T q \\
14) q_1^2 &+ q_2^2 + q_3^2 = q^T q & 15) p_0 &= q + c
\end{aligned}$$

- 1) Calculate a, B, c, f, f', H, H', Q, and U.
- 2) Calculate  $q^T q$  from (8).
- 3) For 3D trilateration, calculate  $q_3$  from (9); for 2D trilateration, calculate  $q_2$  from (10).
- 4) For 3D trilateration, calculate  $q_1$  and  $q_2$  from (11); for 2D trilateration, calculate  $q_1$  from (12).
- 5) For 3D trilateration calculate  $q$  from (14); for 2D trilateration calculate  $q$  from (13).
- 6) Calculate  $p_0$  from (15).
- 7) Choose one of the two candidates of  $p_0$ .
- 8) Return  $p_0$ .

In order to work properly, the module will require at least four beacon distance measurements. However, the more beacon measurements, the more accurate the end estimate will be, so using as many measurements as possible will be the best design choice. Though the calculations may take a little longer due to the summations, the accuracy improvement should far outweigh the runtime increase. These beacon measurements will come in from the Bluetooth module that receives the actual Bluetooth signals from the beacons and calculates the approximate distance the beacon is from the user according to the received RSSI value. The Bluetooth module will give the trilateration module the needed number of beacons in the form of their major and minor numbers along with their respective distance approximations. Because the beacon positions are needed for the calculations, the trilateration module will then use the major and minor numbers given by the Bluetooth module to retrieve their locations that are stored within the database module. Once the locations are known, then the user's location can be determined using the equations above.

Overall, the trilateration module is a component made largely of mathematical formulas without really any data structures. The previous calculated user coordinates will have to be stored for the algorithm to choose the correct solution of the two calculated, but other than that, not much else will need to be contained within the module. Its space complexity will remain small, while its mathematical calculations will most likely make the module take up more resources than some of the other modules.

## 6.4 Pathfinding Design

The pathfinding module is responsible for creating a virtual structure of the building that is being navigated through. Using the mapped building database to retrieve the structure of the building, the pathfinding module calculates the shortest path from a starting location to a ending location using virtual nodes that mark where the user can and cannot walk throughout the building.

### 6.4.1 Purpose of Module

Pathfinding within any system involves moving from one point to another within some sort of space. The process involves determining the area one can traverse, as well as determining obstacles that block the path needed to get from one location to another, if possible. The process of finding a path can be found in many systems from games to robotics. Anything that involves artificial intelligence and traversing through a physical or virtual world will most likely involve some sort of pathfinding algorithm in order for the artificial intelligence to move properly. Even though the beacon navigation system does not robot or game character moving around, one can view the person using the system as a blind user that requires instructions on where to move. The pathfinding algorithm does not force the user to move in a restricted path as is found in moving inanimate objects, but rather gives instruction to the user on the path that would be most beneficial to them to get to their desired destination. Thus, the purpose of the pathfinding module within the system is to use the virtual node mapping provided by the mapped building database and give the user updated directions to get them from their current position to their requested destination within the building they are in.

Most pathfinding algorithms use a collection of nodes giving the positions of all obstacles and open areas within the traversal space. Within the beacon navigation system, these nodes represent either open, walkable areas within a building, or walls that block off movement from the user. The user's specific location will be inputted from the trilateration module of the application, while the user's requested destination will be determined from the user's inputted tag that will attempt to be matched with one of the room's tags, which, once found, will give the room's node location for the pathfinding algorithm to use as a destination. Once the start and destination nodes are given and the collection of nodes mapping the building's structure is provided, an adequate path can be found for the user to walk through. If a path cannot be found, the user will be notified that there is no valid path that can be used to reach the user's destination. This, however, should never happen, because a given room should always be able to be visited if the building is of sound design

and the mapped building database is without error. If the destination node is located on a different floor than the user is currently on, it should be able to guide the user to the staircase or elevator that would lead to taking the shortest, most efficient path. However, the actual algorithm that handles finding the shortest path between two nodes should not have knowledge of multiple floors, but rather just one floor searching a two-dimensional area. The processing of multiple floors should be done outside of this step while remaining in the overall pathfinding algorithm. It will be able to guide the user through multiple floors by using the user's current floor search space to find the closest elevator or staircase. Then, when the user reaches the node, which will be specifically marked as either a staircase or elevator, and the user's location on the z-axis reaches a certain height, the next floor will be passed in and used to search to the destination. By having the specific pathfinding algorithm only need to find a node in a two-dimensional space and having that wrapped around by the part of the module that determines which floor to use as a search space, the design will be able to be kept simple and modular.

Pathfinding within the indoor navigation system will require a significant amount of computing power. Because of this, the module should be limited to run only when needed for the user to be able to reach their destination. Updating the user's path constantly would use excess battery within the android device and would prevent other modules within the application, such as the trilateration algorithm, which determines the user's position, from receiving adequate time to compute. More on this topic will be discussed later.

The overall design of the pathfinding module should be modular and efficient, being able to use as little resources for processing as possible. Varied methods should be included in the design to allow for quick and easy tweaks of the module when the need for testing the system arrives. This should allow for smoother changes to take place if one method is found to not work as well as planned.

## 6.4.2 Deciding the Right Algorithm

There are a variety of different pathfinding algorithms that have been used in many systems throughout the past. A number of tweaks and variations have been made to long-used pathfinding algorithms that introduce small improvements that give more efficiency to particular use cases. However, even though an algorithm may add to a previous one does not automatically make the newer one better. Each specific system must look at what the requirements are for that system and which algorithm is better suited for the system. This not only holds true for pathfinding algorithms, but also any other type of algorithm in general. However, because the discussion is on choosing the right algorithm for pathfinding within the indoor navigation system, a general overview of different algorithms for pathfinding will be conducted.

Perhaps one of the more well known pathfinding/graph traversal algorithms is Dijkstra's Algorithm, which finds the shortest path between one node to every other node within the graph. The reason Dijkstra's Algorithm would not be the best choice within the indoor navigation system is that it visits every node within the collection of nodes making it slower than a more direct approach that only visits that advance the path closer to the destination

node. This better approach is known as the A\* (pronounced A Star) Algorithm. A\* is based off of Dijkstra's Algorithm, but uses some extra information that Dijkstra's does not use to reach the destination node faster visiting fewer nodes along the way. The downside of A\* is that it only finds the shortest path from one node to another instead of one node to every node in the collection, which is why it is faster than Dijkstra's Algorithm. However, this is not an issue because navigating through a building does not require one to visit every corner of the building to get to a particular room, but rather only requires one path from where the person is to where they need to go. For this reason the A\* Algorithm is a better choice.

Following the creation of the A\* Algorithm a number of improved variations of A\* developed including D\*, D\* Lite, and Theta\* to name a few. Both D\* and D\* Lite are typically used in robotic systems when the area that is being traversed and the obstacles within are unknown. Because the mapped building database will hold the building's layout, all of the information will be known about the building, making D\* and D\* Lite not the most beneficial choice for pathfinding within the navigation system. Theta\*, however, is a very good contender as it is very similar to A\*, with a few extra benefits. The main benefit of using Theta\* is that it can give a straight line-of-sight path to corners and turns that have to be taken. Figure 6.4.2-1 illustrates the difference between the path generated between A\* and Theta\*. The red squares in the figure represent the start and destination, whereas the gray nodes are walls and the white nodes are open, walkable areas. Oftentimes A\* will generate a path that involves unnecessary zig-zags to reach the destination, because it follows the nodes' locations exactly, being sure to move from neighbor to neighbor not jumping over a node along the path. Because the collection of nodes is in a virtual space being used to direct movement through a real world continuous path, the user of the navigation system does not have to follow the nodes exactly to reach their destination, but rather can take a loose path along the generated path, which is where Theta\* shows it's benefits. Each node in Theta\* does not have to point to a direct neighbor along the path, but can point to any node in the collection as long as there is a line-of-sight between the nodes. In order for this line-of-sight approach to work properly, the real world path between the two line-of-sight nodes must not have any obstructions along the way, which should not be a problem because any obstructions should be mapped within the building layout in the database. The only downside of Theta\* is that it's runtime is slightly slower than A\* and it is not guaranteed to find the shortest path like A\* is. However, most of the time the path found by Theta\* will be shorter than the A\* because of the lack of zig-zags in Theta\* that is found A\*'s paths a seen in Figure 6.4.2-1, which is a basic example that made more complex would cause even more changes in direction to get to the destination.

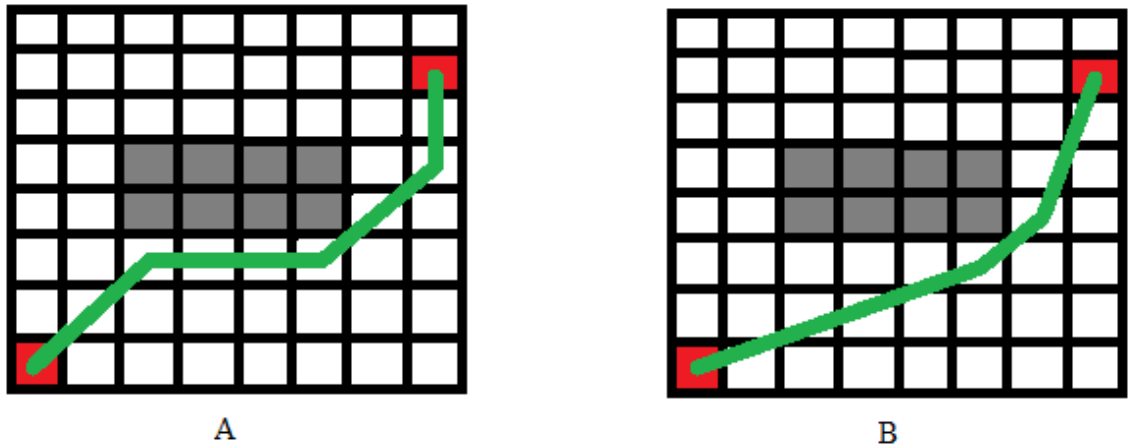


Figure 6.4.2-1: A\* (A) versus Theta\* (B) path comparison

So far, Theta\* seems like the best choice for pathfinding within the indoor navigation system as it provides smoother paths easier for real world navigation and direction following than the A\* approach. However, A\* cannot be written off completely because certain methods of implementation, which are discussed in the next section, can allow for simple implementation and smooth paths, while using less memory. For this reason, both A\* and Theta\* will be looked at more in detail, because only during testing will one be able to be determined which one is more beneficial.

Looking more in depth at the specific algorithm of A\*, one finds two different collections or lists of nodes. One of the lists is considered the open list, which holds all of the nodes that must be visited. The other list is the closed list, which holds all of the nodes that have been visited previously along the search for the destination node. Each node has several values that help advance the search closer to the destination node. The first of these values is typically called the G value and holds the distance from the start node to the current node that is being visited. The second value is called the H value and holds an estimated distance from the current node being visited in the path to the destination node. It is an estimate, because there is no way to tell exactly how far the node is from the destination without first going through to find the path, because of the possibility of obstacles between the current node and the destination node. The typical way of estimating the distance between these two nodes is by using the Manhattan method, which works only if an array is being used as the structure for holding the collection of nodes. It must have an array or grid-like structure, because the method takes the difference between the row indices and the difference between the column indices of the start and destination nodes and sums them together to get the estimated distance. It is given the title, the Manhattan method, because it similar to how one would move from one building to another in Manhattan by going down different blocks. The third member within each node is a parent node, which holds the node that was used to reach the node, which is considered the parent. To continue, both the G and H values are combined to form the F value, which is the total estimated distance from the start node to the end node. The algorithm begins at the start node and looks at all of its direct neighbors and checks their G and H values. If the neighbor node is not allowed for travel or has been visited already, then do nothing. If the neighbor node is within the open list because a visited node had it as a neighbor, then check the node's F value to see

if it is larger than if the current path were used. If it is, then change the G, H, and F value and update the parent, because the current path is shorter than the other path that was used to get to the neighbor. If the neighbor node is not in the open list, then simply update its G, H, and F values and the parent according to the current path. This loop continues until the destination node is reached. The path is then reconstructed by following the parent node's all the way back to the start node adding each node to a stack or list along the way. This is the basic process of the A\* algorithm and can be written simply without any complex logic as long as the steps are followed correctly.

Theta\* follows the same general logic as A\*, so explanation will only be made on the ways in which it differs from A\*. One of the differences is the choice of parent. Theta\* looks at two nodes when deciding a parent node for a neighbor, while A\* only looks at the current node as being a possible parent for the neighbor node. Theta\* uses this choice as one option, but also looks at the possibility of using the current node's parent as a parent for the neighbor node. The two options are decided based on whether the current node's parent and the neighbor node have a line-of-sight with one another. If they have a line-of-sight with one another, then the second option is chosen. If they don't have line-of-sight, then the first option is chosen, which would be the same as A\*. The process of determining whether two nodes have a line-of-sight with one another can be done with integer arithmetic and logic, which allows for faster execution than if floating point operations were used. The algorithm basically looks for any node that is not walkable in between both nodes to determine whether there is line-of-sight.

Another large part of the pathfinding within the navigation system is the process of mapping the user's location to a node within the collections of nodes. The trilateration module will determine the location of the user, which will then be passed into the pathfinding module. The specific A\* and Theta\* algorithms only deal with finding a path between two nodes. Thus, the missing piece is taking that real world position and matching it to a virtual node in the system. To do this accurately, each virtual node must have a radius stored that marks the space it will take up within the real world. It must have a radius, because the chances of the user being at the exact position of a virtual node cannot be relied upon. If the user is within a node's radius, then that node will be passed into the chosen pathfinding algorithm as the start node.

The destination node must also be passed into the pathfinding algorithm. This process will be done by the database that holds all of the "landmark" nodes that are either doors, staircases, elevators, or anything that could be used as a destination. Within the database, these special nodes will be distinctly tagged with a number of keywords to help identify it. Thus, when the destination node is identified within the database, it will be passed to the pathfinding module which will use it as the end node for the path search. Thus, the specific A\* or Theta\* algorithms will not have to worry about the real world, but only the virtual one involving a collection of nodes, a start node, and a destination node. The layer outside of A\* or Theta\* will be responsible for formatting the information so that it can be used by A\* or Theta\*.



### 6.4.3 Graph Design Comparison for Buildings

One aspect of the pathfinding module that has not been discussed is the type of data structure that will be used to hold the collection of nodes that represent the building mapped out in virtual space. Using one data structure has certain advantages in certain areas, while another data structure has advantages in other areas. The most common data structures used for pathfinding are two-dimensional arrays that overlay the exact layout of the search space, adjacency lists that is an array of lists that hold each node and its neighbors, or an adjacency matrix that is a two-dimensional array with one dimension being the node and the other being the node's neighbors. The actual pathfinding algorithm will change depending on which one of these structures is chosen, which is why it is important to choose the most optimal data structure to work with the indoor navigation system.

One of the data structures mentioned above is the adjacency matrix. Figure 6.4.3-1 shows the layout of the adjacency matrix. In this example, the rows represent the nodes within the graph, while the columns represent the neighbors to those nodes. Within the array cells is a boolean value to denote whether the node is a neighbor or not. An integer value could be used instead of a boolean to tell the distance between the nodes, with a special integer marking that the nodes are not connected. An advantage of using an adjacency matrix is the ease of implementation of the matrix. Using a two-dimensional array, traversing through the graph involves just indexing through a row to find the neighbors of a node. The downside, which is the reason why it would not be used within the navigation system, is the space that the matrix takes, which happens to be the size of each node squared. Adjacency matrices are usually beneficial for non-sparse graphs, such as the one in Figure 6.4.3-1, because most of the nodes connect to every other node. The graph that will hold the building map will be a sparse graph because each node only connects to direct neighbors and not nodes across the building. Depending on the total number of nodes, each node will only connect to a few, making it a sparse graph rendering the adjacency a lot of wasted space holding many false values its cells.

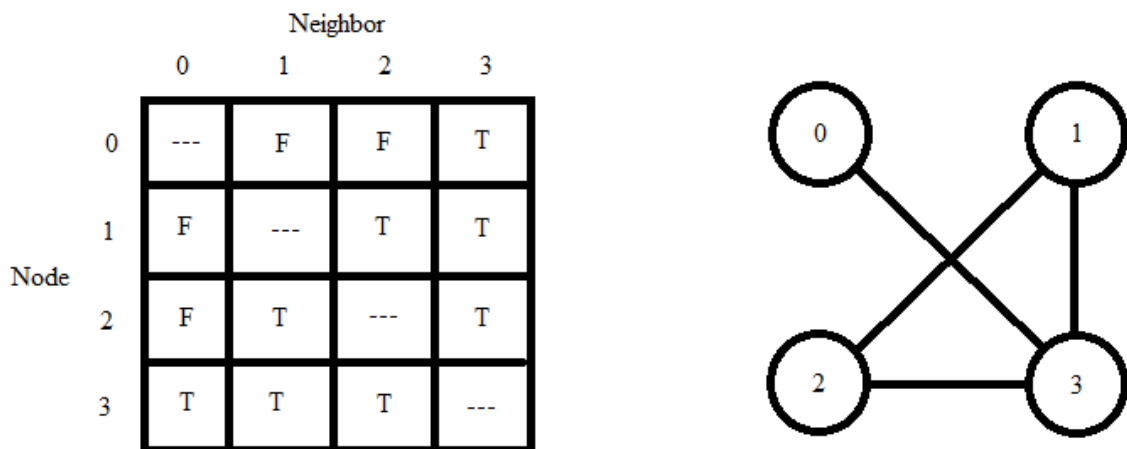


Figure 6.4.3-1 Adjacency matrix example

Another option to use the collection of nodes for the mapped building is a two-dimensional array that directly outlines the building layout. This grid layout is used in many pathfinding

examples, because it gives the actual layout of the search space without any needed conversion. For example, in the adjacency matrix, the structure of the building/graph cannot be easily determined just by looking at the matrix. However, with a grid layout such as the one found in Figure 6.4.3-2, the layout of the building can be easily seen just by looking at the array. Each cell represents a node in the collection with every node being separated by some predefined distance from its neighbors. This provides a logical, easily traversable space with each node having up to eight neighbors if diagonal neighbors can be reached or four neighbors if diagonal traversal is not allowed. Also, with the adjacency matrix each node is assumed to be traversable, unlike the grid option, which must mark whether nodes are traversable because the grid itself is the building layout marking where untraversable walls are and traversable open spaces are. The space complexity is also of order  $n$ , with  $n$  being the number of nodes in the collection. This is smaller than the  $n$  squared space complexity of the adjacency matrix. The retrieval of neighbor nodes is also kept pretty simple with the most complex part being determining which nodes are diagonal neighbors and which are not, which only requires some simple arithmetic.

|   | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    |
|---|------|------|------|------|------|------|------|------|
| 0 | Wall | Door | Wall | Wall | Wall | Wall | Wall | Wall |
| 1 | Wall | Open | Open | Wall | Wall | Open | Open | Door |
| 2 | Wall | Open | Open | Open | Open | Open | Open | Wall |
| 3 | Wall | Wall | Wall | Wall | Wall | Door | Wall | Wall |

Figure 6.4.3-2 The grid layout option

The final option is to use an adjacency list, which is, in a way, the opposite of the adjacency matrix. It is slightly more difficult to implement, but is made for sparse graphs and only does not contain wasted space, because only connected nodes are stored. An adjacency list's structure is an array of lists, with the array index being the specific node and the integers in each list being the nodes that are connected to the node with the index of the array. Figure 6.4.3-3 provides a good concept of an adjacency list using the same graph as in Figure 6.4.3-1. For the pathfinding module within the indoor navigation system, the neighbors of each node needed for the pathfinding algorithm would be known just by iterating through the list at the array index that matches the node index.

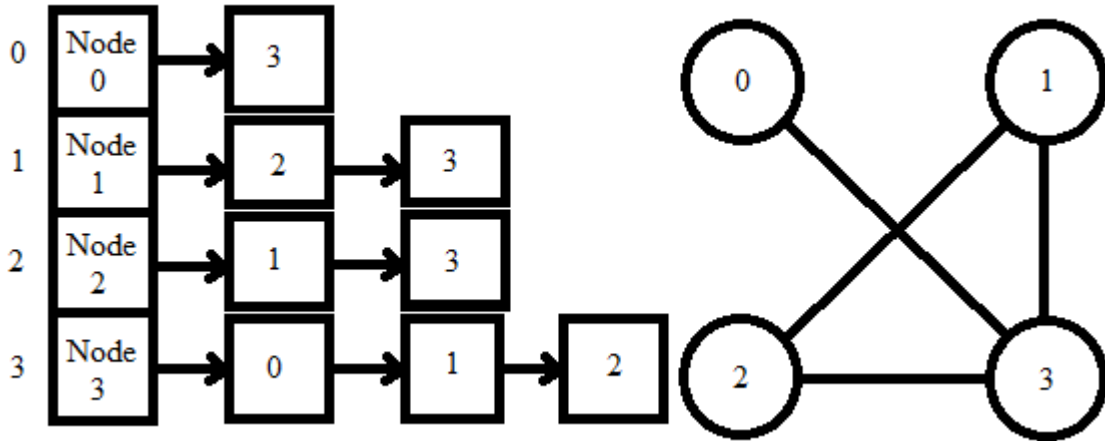


Figure 6.4.3-3 The adjacency list option

Both the grid option and the adjacency list have their specific benefits. At first glance it might seem that the grid option would be more beneficial by taking less space because it is essentially an array of nodes, while the adjacency list is an array of nodes plus lists of integers. The grid option is also more intuitive and would be simpler to create the building map connecting the real world with the virtual, because each node can be given a square area and the two-dimensional array overlaying the building. Using the adjacency list the same way in which the grid layout is used by having nodes separated into a grid, but just mapping the nodes into an adjacency list would render the grid option a much better option. However, an adjacency list might be more beneficial than the grid option by giving only certain locations nodes to mark decision points within the building. Figure 6.4.3-4 shows an example of a building with only nodes being placed in locations that are necessary for moving through the building. Certain necessary locations would be doors, staircases, and elevators. This approach to mapping the building would use far less memory than the grid option, because only the nodes that are needed for moving through the building would be stored, instead of a node for every square area in the building. Moving from the start node to the destination would also be faster due to there being fewer nodes in the graph. In the end, however, the correct option to choose will not entirely be known until testing of the system. Thus, both options will be taken into account as input methods for the pathfinding algorithm with the final choice being decided during testing.

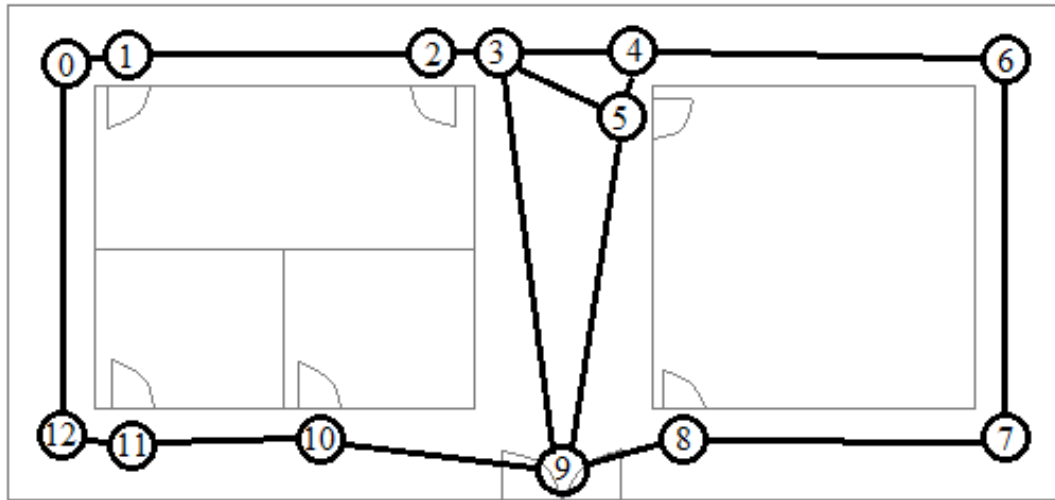


Figure 6.4.3-4 Node layout for using an adjacency list

## 6.4.4 Off-Course Determination Protocol

The indoor navigation system's pathfinding module must also be able to handle the possibility of the user not following the system's instructions and moving off the given path. The system should be able to provide a simple, energy efficient method of determining whether the user is on the path or off of the path. If the user is off of the path, the system should be able to guide the user from where they are, back to their desired destination.

One way in which to handle the user going off course is to have the pathfinding algorithm, either A\* or Theta\*, run at a certain low rate, so that the path is constantly being updated relative to where the user is currently located. Thus, the user would technically always be on the correct path, because the path is always calculated from the user's location. This would be the simplest implementation as no extra functionality would have to be developed. Only the algorithm would have to be written and given a rate at which it should run. The downside of doing this that it would most likely use a large amount of battery, because of the constant calculations occurring at rate. However, if the rate is limited to once every two to five seconds, the amount of power used will be less, making the power efficiency be more reasonable. One way to further improve efficiency is to only find the path for only the floor the user is currently on, so that if the destination is on a different floor, the path only guides the user to the closest set of stairs or closest elevator. Using this method assumes that every elevator and staircase leads to the floor that the user needs to get to. This might work for most buildings, but some buildings might not follow this layout, making the shortcut an inadequate method for finding paths. Thus, all of the building's floors must be taken into account when finding a path for the user.

Perhaps the most efficient, but more difficult implementation of determining whether a user is off course or not, is to run the pathfinding algorithm only once to get the entire path and rerunning the algorithm only when the user is off of the path. It is a more difficult option, because an extra algorithm must be developed to determine whether the user is indeed off of the path. However, if a proper, efficient algorithm is constructed, this method would be the best option for the indoor navigation system.

Looking more into the way in which to determine whether or not a user is off course when using the one-time pathfinding option, all cases of travel by the user must be taken into account. For example, if a user walks along the same path, but facing backwards or if a user faces the right direction, but walks backwards along the path. The off-course protocol should be able to handle these fringe cases, as well as the most obvious case of the user just walking in the wrong direction away from the path. One thing becomes clear upon further study; the user's direction cannot be used to determine if a user is off-course. The user will ultimately be able to look in any direction without affecting the path. There will be an on-screen arrow guiding them towards the next node in the path, which will be discussed in the user experience section, but this arrow or any direction the user is facing will not be used within the off-course protocol. The user's position, however, is the only input that will be able to be used in determining their on-path status. In order to determine their on-path status, the user's location must be taken at a certain rate from the beacons within the building. To conserve time spent computing, the rate should be as slow as possible, while still remaining effective. There will be three states of the user in relation to the path: on-course state, warning state, and off-course state. The on-course state simply means that the user is on the correct path going towards the next node in the path. The warning state is given whenever the user start to go off of the path. When the user reaches this state, the system will alert the user that they are going off-course. If the user continues, then the user will be in the off-course state triggering a recalculation of the entire path.

The off-course protocol will vary somewhat depending on which pathfinding algorithm is used within the system. If the A\* algorithm is used within the system, determining the user's state will consist of finding the current node the user is located at and calculating the distance between that node and the next node in the path that the user has yet to get to. This can be done simply using the Manhattan method, if a grid is used for the search space, or the distance formula, if the adjacency list is used. With Theta\*, this approach cannot be used, because the list of nodes in the path are not necessarily direct neighbors. One node in the path may be at one location, while the next node the user has to get to might be very far away, which means that there would be a blind spot in between each node in the path where the system would not have any information. To determine if the user is truly off-course, a virtual perimeter will be created around the last node the user passed and the next node they have to get to. One perimeter will mark the transition between the user's on-course and warning states, while another wider perimeter will mark the transition between the user's warning and off-course states. Using a circle as the perimeter would work, but it would leave large areas to the sides of the correct path that would allow the user to walk a significant distance before being warned that they are heading off-course, especially if the distance between the two nodes is very large. The shape of a rectangle would be a better choice for deciding on the perimeter's geometry, because it is more similar to the straight

line between the two nodes than the circle. Using a circle results in large areas to each side of the path that would grow the farther the nodes are separated. Using a rectangle would prevent this and follow the path between the two nodes more precisely. Figure 6.4.4-1 shows an example of how a rectangular perimeter. In order to create a rectangular perimeter, four lines must be made using the known positions of the two nodes. These four lines are separated into two groups. One group of lines' slopes are parallel to that of the line between the two nodes, while the other group's lines are perpendicular. For the group that has perpendicular slopes to the path, one line goes through the first node, while the other line goes through the second node. Each of these lines is then shifted either vertically or horizontally by some constant to allow for more off-course leniency. They are shifted vertically if the slope is less than one and greater than negative one. Otherwise, they are shifted horizontally. The lines parallel to the path are created by shifting the path slope up and down or left and right, depending on the slope, to form the two parallel lines. These four lines and their equations, which can be seen in Figure 6.4.4-1, form the rectangular warning perimeter in the yellow section of the figure. Further shifting of these lines outward forms the off-course perimeter, which is covered in red lines. Once the rectangles are calculated, all that needs to be done is to check which section the user is located in to determine the user's status.

Using Theta\* is obviously more complex when it comes to implementing the off-course protocol. However, the end result will have more straight lines with fewer turns compared to A\*, which leads to a simpler protocol for producing directions for the user, because of the fewer turns involved. The next section will further discuss the importance of the rectangular perimeter as it pertains to giving the user directions if in fact the option of calculating the path only when the user goes off-course is used within the system and not at a rate.

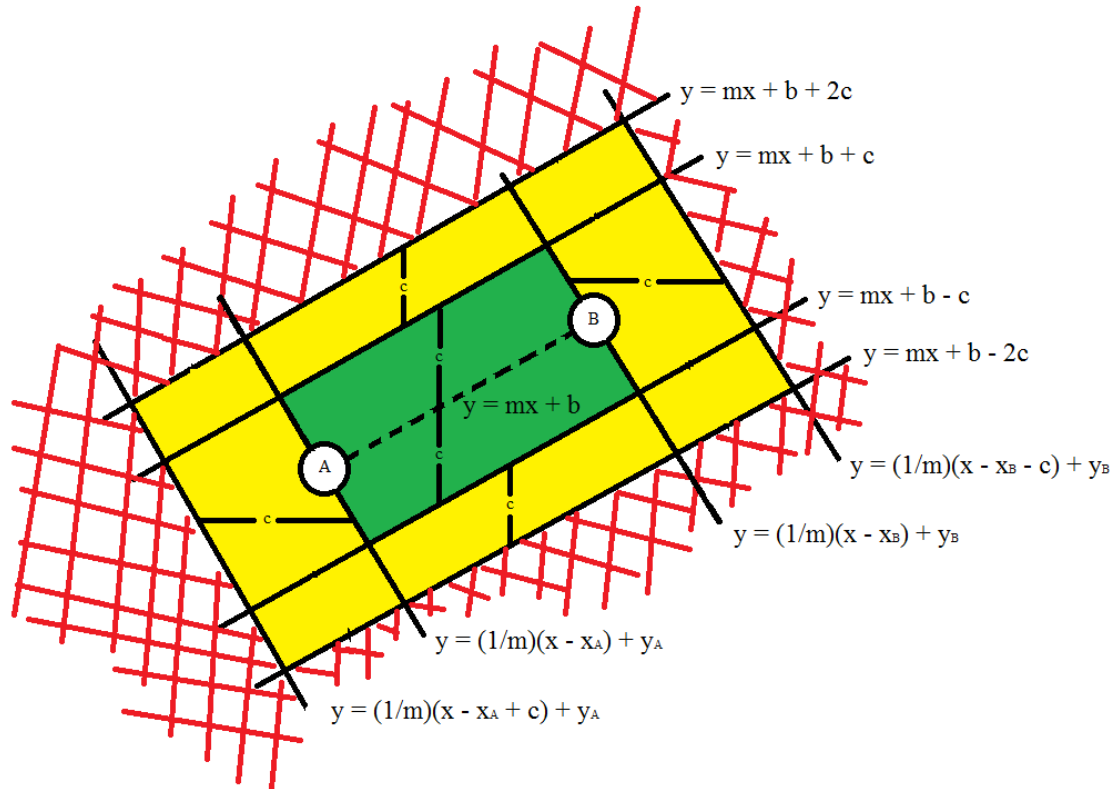


Figure 6.4.4-1 Rectangular Perimeters for User's On-Course (Green), Warning (Yellow), and Off-Course (Red) states

## 6.4.5 User Direction Protocol

The last aspect needed within the pathfinding module is the protocol to use for giving the user directions while the user walks through the building. The pathfinding module will be responsible for knowing the user's location relative to the set of nodes in the system. Thus, it would be best fitting for the pathfinding module to handle the computations behind giving the user the next node's location they need to get to. Once the next node in the path is determined, the node's position will be given to the Google Glass specific software to finish calculating the directions using the Android API, then it will give the user auditory directions or visual directions depending on the user's choice. Because of the multiple modules involved in giving directions to the user, each module must fulfil a logical, modular function. The pathfinding module is just responsible for tracking the user's position within the set of nodes and calculating the next node the user needs to reach. There will be no specific Android libraries involved allowing for more portability if the system were ever to be ported to another operating system.

The process behind determining the next node needed in the path for the user to reach is actually not a difficult one. The off-course protocol is already closely associated with directing the user. Thus, these two submodules, the user direction protocol and the off-course protocol, can be seen as one user tracker module, because they track the user and determine where exactly the user needs to go to and whether or not they are staying on

target. In order for the user direction protocol to work correctly all that is needed is the user's current position and a trigger that is fired when the user reaches the next node in the path. Once the trigger is fired the system must then determine the next node for the user to reach in the list of nodes that make up the path. This process will continue until the user reaches the end of the path, which is their destination node. The user's location is already being tracked within the off-course protocol, so that same value can be used by the direction protocol.

If  $\Theta^*$  is used within the system, the design for the trigger point that moves the user's local goal node to the next node in the path will utilize the off-course protocol's perimeter structure from Figure 6.4.4-1. Once the user passes the line that goes through node B in the figure, node B will be set as the previous node and the list that holds the nodes that make up the path will update by removing the previous node so that only the current node that needs to be reached will be at the head of the list of nodes making up the path. The process of calculating the off-course perimeter and waiting until the user reaches the next node while monitoring that they do not go off of the path will be repeated until they reach their destination. An inequality will be generated based on the line's equation that will check the user's coordinates to see if they are greater or less than the line's equation depending on the orientations of the previous and next nodes to one another. Once the inequality becomes true, the user is considered to have reached the node even if they do not directly come within the node's radius. This design allows the user to follow an approximate path and not have to reach every node exactly before moving onto the next node, which would be very impractical. The off-course protocol handles whether the user gets too far off path, so the worry that the system will move onto the next node before the user is close to the current node breaking the system can be disregarded, because the off-course protocol will always make sure the user stays within vicinity of the path. If not, the path would be recalculated.

If  $A^*$  is used or there is no off-course protocol due to the system recalculating the path at a specified rate, the off-course perimeter will not be present and thus, cannot be used to trigger the system to move to the next node. In this case, to keep the direction protocol simple, the distance between the user and the next node can be used. This is not possible with  $\Theta^*$ , because  $\Theta^*$  finds the best line-of-sight path, which could result in two nodes being a large distance apart from one another. It is possible with  $A^*$ , because the path holds nodes that are direct neighbors of one another, so that each node will be close to the next node in the path. If the user comes within a certain distance of the current node needing to be reached, then direct the user to the next node along the path.

With the design completion of the user direction protocol, the pathfinding module's design is complete fulfilling its specific responsibility of creating a collection of nodes that are used to guide the user to their desired location, while at the same time monitoring their position making sure that they follow the given path and are guided smoothly from node to node until the destination node is reached. Maintaining modularity, the pathfinding module will fit nicely into the indoor navigation system fulfilling its specific responsibilities.



## 6.4.6 Floor Sequencer

The floor sequencer is responsible for stringing together the paths from the pathfinding algorithm into a multi-floor path that the user can follow. Because the pathfinding algorithm only calculates the path of one floor at a time and the system needs to be able to calculate the shortest path across multiple floors, an extra step must take place to connect the different paths from each floor into one long path. The link between the floors will be the staircase and elevator nodes. The user will start on a particular floor. That starting floor will go through the sequencer, along with the floor where their destination is located on. Then, the graph holding the connections between the elevators and staircases held in the database module will be used to find the appropriate start and destination nodes for each floor to be passed to the pathfinding algorithm.

Figure 6.4.6-1 provides a good visualization of the structure of a building along with elevator and staircase connections that will be stored in an adjacency list. The user's start node upon entering the building is on the first floor in blue, while their desired destination node is located on the back right corner of the third floor in green. These two nodes are important to the sequencer, because the user start node will be used as the start node in the search for the first floor path, while the user destination node will be used as the destination node of the last floor's path. Any floors that are just in between the start and end floors will just use the elevator and staircase nodes as start and destination nodes for that floor.

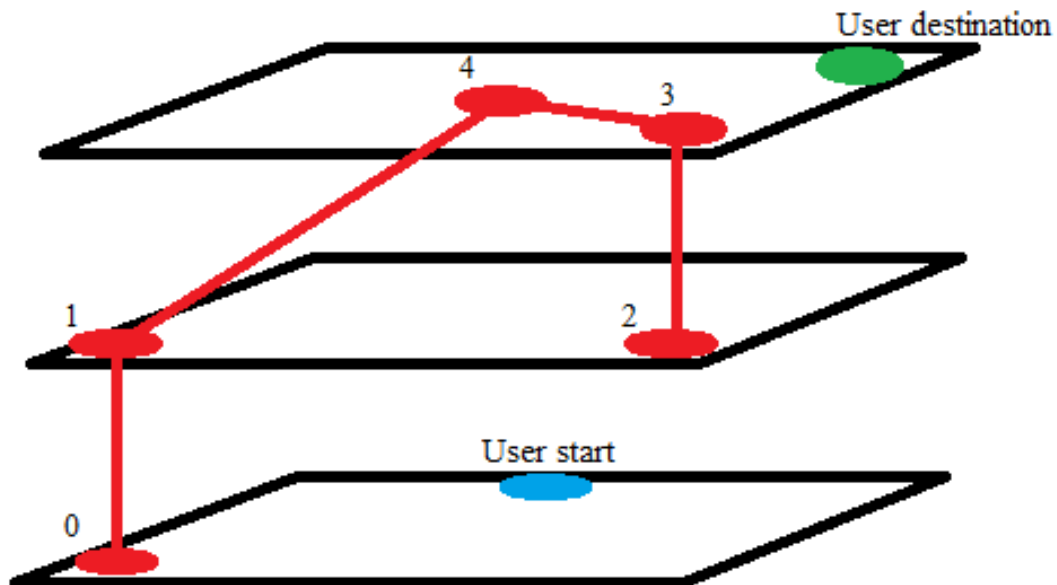


Figure 6.4.6-1 A building with three floors with the user's starting node, destination node, and the elevators and stairs (red).

The floor sequencer will be responsible for traversing through the elevator and staircase graph to find all of the possible elevators and stairs the user can take to get to the desired

floor and will provide them in the right order to the pathfinding algorithm that will use them as start and destination nodes for each floor. Once the pathfinding algorithm returns a floor's path, it will be appended to the proper multifloor path. Due to the possibility of having multiple paths to the same floor, the floor sequencer will have multiple multifloor paths. In order to determine which path is the shortest, one cannot simply find the path with the fewest nodes, especially if Theta\* is used, because Theta\* only uses the nodes that are in line-of-sight with one another, which means one path may involve only two nodes, but be very long, while another may involve four nodes and be much shorter. For this reason, the calculation time for each path must be recorded to determine the shortest path. Assuming that a shorter path will take a shorter amount of time to calculate than a longer path due to the visiting of fewer nodes when calculating the path, allows for calculation time to be used.

One case that the floor sequencer must also be able to handle is the possibility of having multiple destination nodes. There may be two entrances to the same room. Thus, the user must be guided to the entrance that is closest to them. In order to handle this, the same path search can be run as many times as there are destination nodes. Another more efficient method is to only rerun the pathfinding algorithm on the floor that the destination node is on, because the other floors' paths will not change.

Below is the algorithm that will be used to find all of the different paths to different floors according to the graph of the connected elevators and staircases. The paths generated will be kept in a list and will be used to sequence the right floors to be used to find the entire path. The algorithm uses an iterative depth-first search algorithm with modifications so that a list of all of the paths between the start and end floors can be returned.

```
List<List<Node>> GetFloorPaths(startFloor, destFloor)
  let L1 be a list
  let L2 be a list
  let S be a stack
  for all nodes v on startFloor
    S.push(v)
  while S is not empty
    v S.pop()
    if v is not labeled as discovered:
      label v as discovered
      add v to the end of L2
      if v is on destFloor:
        insert L2 at the end of L1
        remove v from the end of L2
        label v as undiscovered
      else:
        for all edges from v to w on G.adjacentEdges(v) do
          S.push(w)
  return L1
```

If the example in Figure 6.4.6-1 was placed through the algorithm above, L1 would contain two paths. The first path would start at node 0, then go to node 1, then node 4. The second path would start at node 0, then go to node 1, then node 2, then node 3. Once these paths are found, the floor sequencer can then call the pathfinding algorithm with the right sequence of floors to use to calculate the list of entire, multi-floor paths, while tracking the time for each multifloor path to be calculated so that the shortest is used as the final path that the user must follow.

## 6.5 Application UX Design

The Google Glass application is designed to be usable by both visually impaired and non-visually impaired users. Text and images will be displayed on the screen, but the device will also communicate with the user by speaking any text currently being displayed. In most cases, the user will communicate with the application by simply using their voice to either speak a key phrase or to choose the destination. The only times the user will need to use the touchpad on the device is to wake the device in order to start the application and in order to exit the application. Visually, the application will mostly consist of brightly colored text or images on a black background. This is due to the way the display on the device works. The black sections will actually look transparent while the other sections will be visible. Having a brightly colored background can will direct too much light into the users eyes making it uncomfortable, especially in a dark environment.

Figure 6.5.1-1 shows a mockup of the application. The first and second images show how the user can access the application. These are both part of the operating system running on the device and cannot be changed. Whenever the device is on the home screen, as indicated by the clock and the “ok, glass” prompt, the user can simple speak the key phrase “ok, glass” in order to access the list of application-specific key phrases. This list includes many different key phrases specified by applications which correspond to an action which the device will carry out. In order to start the directions application, the user will simply say “get directions to.” This will start the application which will then prompt the user to say their desired destination. The application will then check that the location is valid by cross-referencing the database. If the destination is invalid, the application will prompt the user and shortly after the application will be closed. In order to specify a new destination, the user will need to start the application again from the home screen. If the destination is valid, the application start generating directions from the users current location to their specified destination.

Once the application has generated the directions for the user, it will start prompting the user to follow said directions. The application will display an arrow which will continuously point to the next point of interest in the directions and underneath the next step in the directions will be display. The application will also periodically speak to the user by dictating the next step of the directions. The application will also tell the user when they reached a point of interest and what they should do next. If at any time the user starts getting off-course, the application will prompt the user, both by displaying a message and speaking to the user, to get back on course. Once the user gets back on course, the application will continue to display and dictate the directions as it was doing before. If the

user spends too much time going off-course, the application will automatically generate new directions using the user's current location. The application will then start displaying and dictating the newly generated directions. Once the user reaches the destination the application will let the user know which direction the destination is in relation to the direction the user is facing. If at any time the user wants to exit the application, they can simply swipe down on the touchpad located on the side of the device.

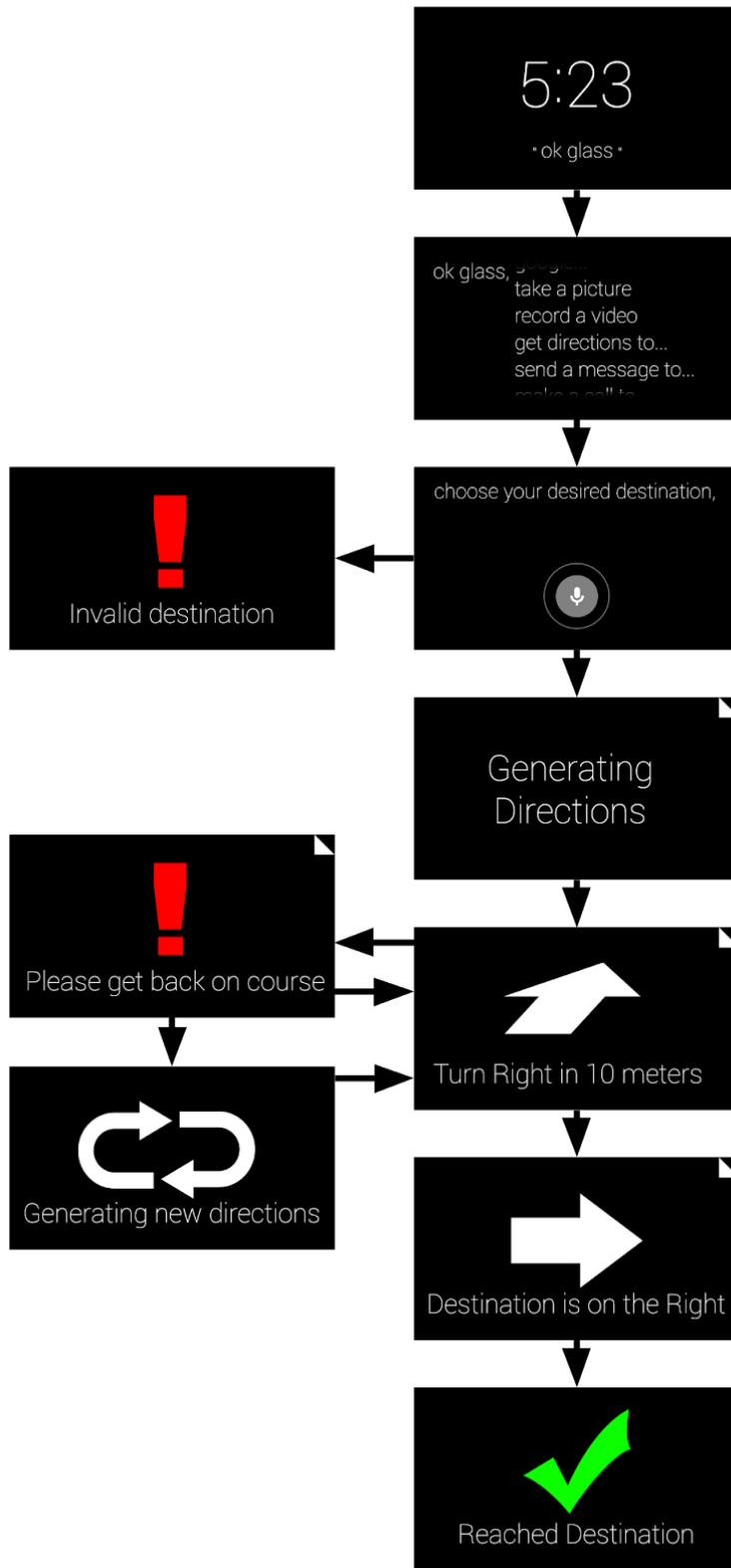


Figure 6.5.1-1: Mockup of the Google Glass Application

## 6.6 Concurrency

This section explores the possibility of implementing a simple non-concurrent call pattern in comparison to a call pattern that utilizes concurrency including some of the challenges that arise with concurrency.

### 6.6.1 Comparisons for different call patterns

The most simple call pattern is shown in Figure 6.6.1-1. It doesn't use any concurrency and therefore is the least efficient call pattern. The call pattern shown in Figure 6.6.1-2 uses concurrency to scan for Beacons. Both call patterns have the same start-up process. The User chooses a destination using the Application using voice commands. The Application then gets more information on the destination from the database such as the it's location. With this information, the Application starts finding the users location by scanning for Beacons with the Beacon Scanner. Once the Beacon Scanner is done scanning for Beacons, the distance for each Beacon is calculated. With the distance of Beacon, the Trilateration algorithm can be used to find the Users starting location. With the starting location and the destinations location, the pathfinding algorithm is used to generate a path from the User to the destination. Step-by-step Directions are then generated from the path and the Main Application Loop begins.

In the non-concurrent call pattern shown in Figure 6.6.1-1, the main application loop starts by scanning for Beacons. Once the Beacons are done being scanned, the distance for each found Beacon is calculated. With the distance for each Beacon calculated, the users current location is calculated using the Trilateration algorithm. This location is then passed into the Off-course Protocol, where it is determined whether the user is still on-course. The results of this protocol are passed into the Audio Protocol where it determined what should be communicated to the user. This call pattern is not very efficient since nothing is being done while the Application is scanning for Beacons.

In the concurrent call pattern shown in Figure 6.6.1-2, the main application loop is similar to the loop shown in Figure 6.6.1-1, but instead of scanning for Beacons in the main thread, it scans for Beacons in the background as the rest of the process takes place. In the first iteration, the loop only scans for Beacons that will be used in the next iteration of the main application loop. In subsequent iterations, the main loop will use the Beacons scanned in the preceding iteration to carry out many of the same steps taken in the non-concurrent call pattern.

Using the non-concurrent call pattern allows for a less complex operation and avoids having to deal with thread safety and race conditions. On the other hand, the concurrent call pattern performs much better allowing for each iteration to last less time. This allows the application to check the users location and determine whether they have gone off course more often.

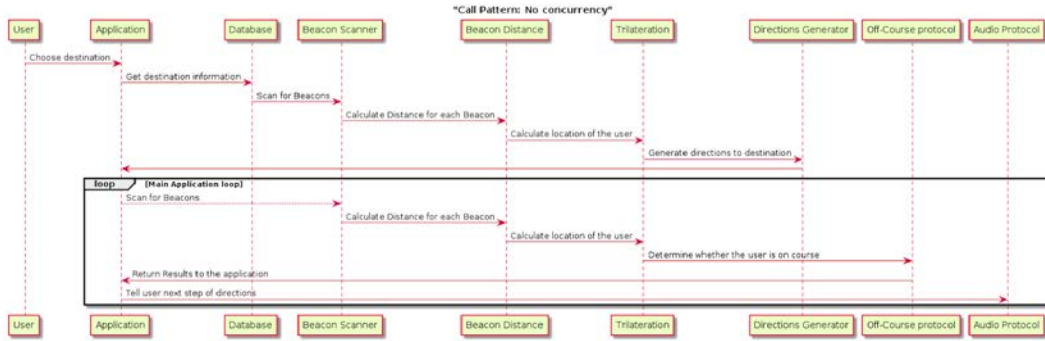


Figure 6.6.1-1: Sequence diagram for non-concurrent call pattern

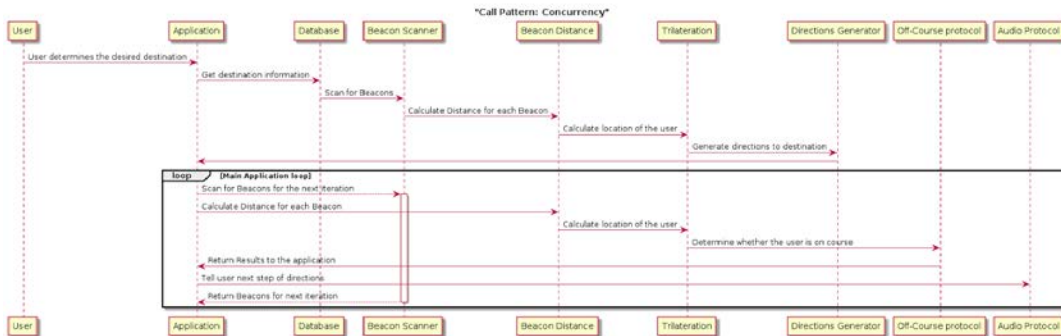


Figure 6.6.1-2: Sequence diagram for concurrent call pattern

## 6.6.2 Ensuring thread safety

If the concurrent call pattern is implemented, it is important to maintain thread safety. This means one thread should not be able to change the value of a variable that is being used in another thread. This becomes necessary while scanning for Beacons. If the application adds more Beacons to the set of Beacons currently being used to calculate the users location, the resulting location would be inaccurate. In order to ensure thread safety, it is important to create a new empty Set of Beacons every time the scanning is started. This will ensure that the instance of the set in which the Beacons are being added is different that the instance of the set being used to calculate the users location.

Thread safety is also important when editing UI elements within the application. If any changes are done to UI elements outside of the main thread, also known as the UI thread within Android, it is not guaranteed that these changes will take place. This means that whenever an UI element need to be edited outside main thread, it is necessary to use the *runOnUiThread* method to execute the changes.

## 6.7 Security

This section explores some of the security concerts regarding the project and how they are will be address in the implementation of the systems.

## 6.7.1 User Data

One of the advantages of using iBeacons, is the fact that the hardware is relatively “dumb.” Its only responsibility is to advertise the iBeacon protocol. The users device reads the incoming signals being advertised by the iBeacons and instead of using a mesh network that relates information through the grid of devices back to a server in order to process the required data, all of the data processing is done on the users device. This means all of the users data will remain on their device, eliminating the possibility of the users data being compromised if the hardware becomes compromised. This not only protects the users, but also protects the company/business using the system from the backlash of compromising their customer’s data.

Another security advantage of the iBeacon system is that it works in a similar fashion to an opt-in service. Users are able to remain invisible to the system by simply not installing the application on their device. Other systems track users either without their knowledge or without their explicit consent. This becomes especially important if the system collects data for analytics. Certain users, who are conscious of their privacy, would not feel comfortable having their actions being tracked and used by companies without their consent.

## 6.7.2 iBeacon spoofing

Due to the simplicity of the iBeacon protocol, it is relatively simple to program a Bluetooth capable device (such as a smartphone) to advertise the same signals as the Beacons installed within a building. If the system scans one of these spoofed Beacons, it could confuse it and possibly cause it to calculate an incorrect location of the user.

The simplest case to filter out spoofed Beacons is by ignoring Beacons which are in a different floor than the majority of the other detected Beacon. It is safe to assume that if the majority of the Beacons scanned are located on the first floor of the building, any Beacon scanned that is located on other floors are either spoofed Beacons or simply Beacons that should not have been scanned. An example of this can be seen in Figure 6.7.2-1. As the figure shows, four Beacons are detected in the first floor, 2 Beacon in the second floor and one Beacon in the third floor. The application will automatically assume that any Beacon that is not located in the first floor,



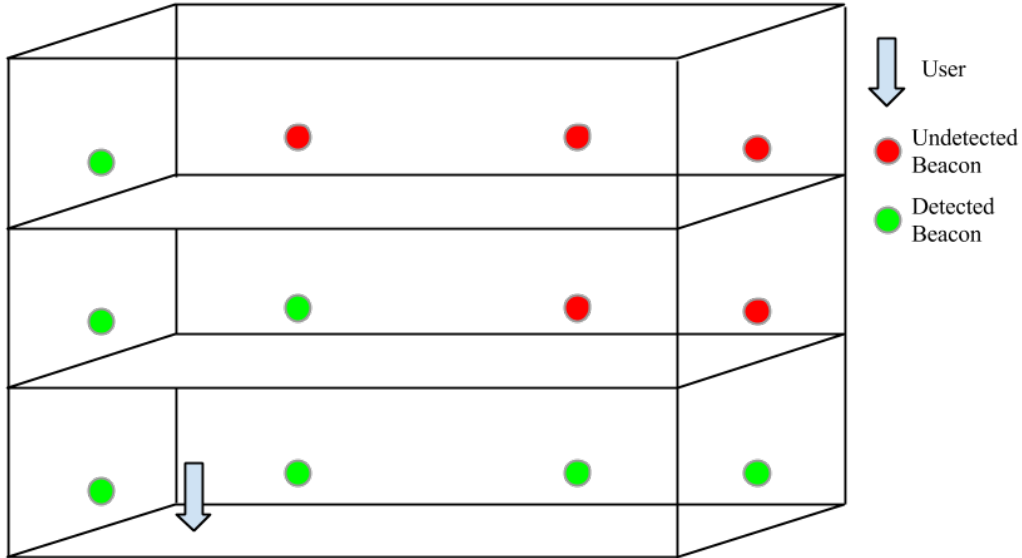


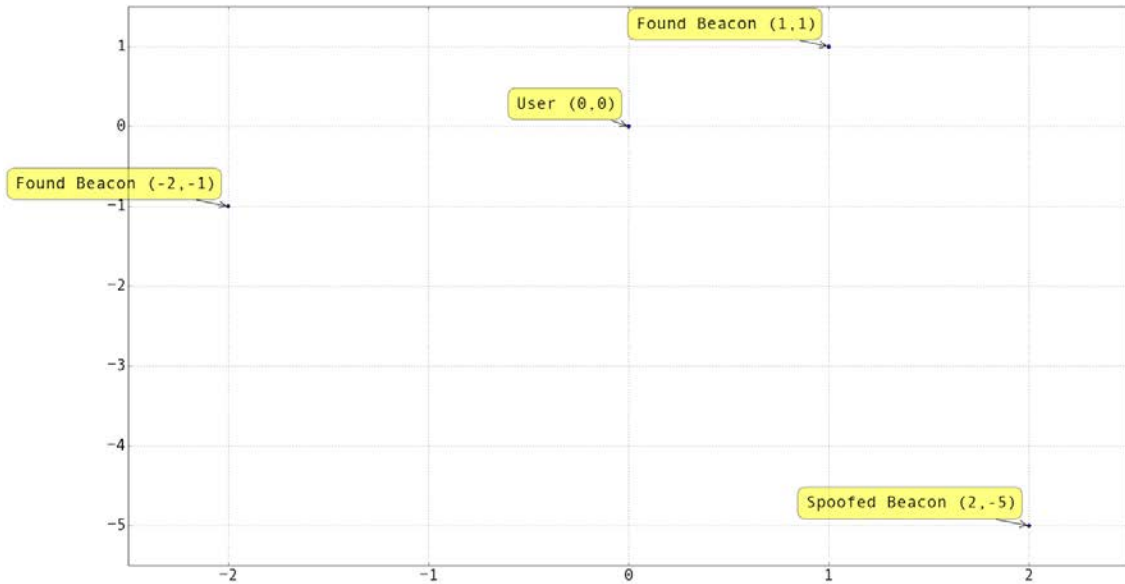
Figure 6.7.2-1: Beacons detected from multiple floors.

A more complex scenario is when a spoofed Beacon is detected in the correct floor. This problem can be solved by cross-referencing the distance from one Beacon to another to the calculated distance from the Beacon to the user. This is done by getting the measured distance from the user to one Beacon, the measured distance from the other Beacon to the user, and the theoretical distance from one Beacon to the other. These three values have to make up the sides of a valid triangle. The conditions to check whether the values were valid are shown in Figure 6.7.2-2. Once this is carried out for all the Beacons, the Beacons that didn't pass any test cases can be assumed to be spoofed Beacons and can therefore be ignored.

With length A, B, and C  
 $A+B>C$   
 $A+C>B$   
 $B+C>A$

Figure 6.7.2-2: Conditions for the sides of a triangle.

An example of this method is shown in Figure 6.7.2-3. Three Beacons are scanned by the application. By matching their minor and major values, their coordinates are extracted from the JSON data of the building. First, Beacons 1 and 2 are checked to be real. The distance between them is calculated from their points. This distance along with the measured distance from the user pass all the conditions given in Figure 6.7.2-2. Next the spoofed Beacon, who JSON data says is in Point (2, -5), is checked alongside Beacon 1, which is already confirmed to be real. The distance between them is calculated from their points. When checking these distances, they automatically fail the first condition therefore confirming that the spoofed Beacon is fake.



*User Coordinates: (0,0)*  
*Found Beacon 1 Coordinates: (1,1) Measured Distance = 1.414*  
*Found Beacon 2 Coordinates: (-2,-1) Measured Distance = 2.236*  
*Spoofed Beacon 3 Coordinates: (2,-5) Measured Distance = 2.5*

*Testing Beacon 1 and the Spoofed Beacon*

$$\text{Distance between Beacons} = [(1 - 2)^2 + (1 + 5)^2]^{\frac{1}{2}} = 6.083$$

$$\text{Assume } A = 1.414, B = 2.5, C = 6.083$$

$$A + B = 3.914 > C$$

$$A + C = 5.02 > B$$

$$B + C = 5.842 > A$$

*Testing Beacon 1 and 2*

$$\text{Distance between Beacons} = [(1 + 2)^2 + (1 + 1)^2]^{\frac{1}{2}} = 3.606$$

$$\text{Assume } A = 1.414, B = 2.236, C = 3.606$$

$$A + B = 3.65 < C$$

*First condition is false, therefore the spoofed Beacon is confirmed to be fake*

Figure 6.7.2-3: Example of finding spoofed Beacons

One problem with this method arises when both Beacons being compared and the user are all collinear. If the distances from this use case are tested against the conditions shown in Figure 6.7.2-2, a false negative can arise. This is explored in Figure 6.7.2-4 with a modified version of the example in Figure 6.7.2-3. Here the position of Beacon 2 is changed in order to make Beacon 1 and Beacon 2 collinear. The measured distance is also changed accordingly. When testing Beacon 1 and Beacon 2 using the conditions from Figure 6.7.2-2, one of the conditions fails due to the sum of the distances being equal to the third distance. This would cause Beacon 1 and Beacon 2 to be marked as possibly fake. If all other test

fail when testing the other Beacons, the algorithm will not be able to confirm any of the Beacons scanned as being real.

*User Coordinates: (0,0)*

*Found Beacon 1 Coordinates: (1,1) Measured Distance = 1.414*

*Found Beacon 2 Coordinates: (-2, -2) Measured Distance = 2.828*

*Spoofed Beacon 3 Coordinates: (2, -5) Measured Distance = 2.5*

*Testing Beacon 1 and 2*

$$\text{Distance between Beacons} = [(1 + 2)^2 + (1 + 2)^2]^{\frac{1}{2}} = 4.242$$

$$\text{Assume } A = 1.414, B = 2.828, C = 4.242$$

$$A + B = 4.242 > C$$

$$A + C = 5.656 > B$$

$$B + C = 7 > A$$

*First condition is false, Beacon 1 and 2 marked as possibly fake*

Figure 6.7.2-4: Example where both Beacons and the user are collinear

In order to deal with this use case, the conditions will remain the same, but the tests should still be considered a success if one of the conditions fails due to the sum of the two distances being equal to the third distance. This will allow the algorithm to handle collinear points.

When dealing with these calculations using real-world data, many of the measured values will not be exact making it hard to fulfill some of the conditions. In order to deal with this, a delta value will be assigned to each test determining the allowed margin of error. This value cannot be determined at this point in time, but will be set after some real-world testing is performed on the application.

## 6.8 DynamoDB

Up until now, the application has assumed that the building's information is present in the form of a JSON (JavaScript Object Notation) file that contains all of the information needed for the system to guide the user from their current location to their desired destination. Unless the file is stored on the user's Android device, there is no way of retrieving the building file. In order to solve this issue with a dynamic real world solution, a database will be used as a centralized point where all of the buildings that use the navigation system will hold their JSON building files. The database of choice is DynamoDB, because it provides support for JSON storage and retrieval.

DynamoDB is a NoSQL database service that fits perfectly into the processes of the navigation system. Using the service also remains simple due to the Amazon Web Services (AWS) SDK, which provides an API for creating and querying tables. Upon entering the building, the user will start their application. When the application starts up, it will listen for the Bluetooth beacons and retrieve their UUIDs. Because these UUIDs uniquely identify the building from other buildings, the beacon UUID will be used to query the correct building's JSON file. Once the file is queried, it will then be parsed within the

database module of the system, which will store the information in the file in appropriate data structures allowing the system to use the information.

The process behind storing the JSON file in DynamoDB involves converting the file made up of nested values into one long Java String literal as seen in Figure 6.8-1. The conversion process is pretty simple and just involves combining the separated keys and values in the JSON file into a Java String using escape characters wherever needed to specify where the keys end and the values begin. Once the file is converted, a few extra lines of code allow it to be inserted into the table holding all of the buildings and their respective information uniquely identified by each of their beacons UUIDs. In the previous section that discussed parsing the JSON file, the UUIDs were not mentioned as being needed within the JSON file. However, with the addition of using DynamoDB, each JSON building file will contain a UUID that distinguishes it from the other buildings. When the String literal is added to the DynamoDB table, the UUID will be used as a primary key for the table. When the application then needs to retrieve a building's information, it will connect to the DynamoDB database holding the building files and query for the building with the primary key that matches the readings of the beacon UUIDs from the Bluetooth module.

Converting the JSON file into the correct format of a string, though simple to do manually, will require software so that the process can be done automatically. Having it be converted automatically will allow for easy setup of getting the system to work within the building. The same application used to guide the user through buildings will be used by the building owner, or whoever's responsible for setting up the system, will also be used for system setup. This way, when a JSON file is created by the building technician, a touch of a button will convert the file into the right format and insert it into the DynamoDB buildings table.

The correct format can be seen in Figure 6.8-1, which shows a part of the converted JSON file with the original JSON format being from Figure 6.1.2-1. This figure also includes the UUID that uniquely identifies a building from other buildings. The UUID comprises part of the Bluetooth message that comes from the beacons making up 16 bytes of 31 bytes in the entire Bluetooth message. The algorithm behind converting the JSON file to a Java String involves reading in the entire file into a character buffer. Then, within the buffer, an escape character is added behind every quotation mark that is in the buffer, so that the quotation marks within the original JSON file do not end the string's continuation. Once all of the escape character ('\') are added, the buffer can be set to the String json as seen Figure 6.8-1.

```
String json = "{
+   \"\nUUID\n\" : \"\nB9407F30F5F8466EAFF925556B57FE6D\n\" ,
+   \"\nnodeCollection\n\" : {
+     \"\nnodeDistance\n\" : 1 ,
+     \"\nbuildingLayout\n\" : [
+       [
+         \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\",
+         \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\" ,
+         \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\" ,
+         \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\" ,
+         \"\nS\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\nE\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\" ,
+         \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\" ,
+         \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\", \"\n0\n\", \"\n1\n\", \"\n0\n\" ,
+         \"\n0\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n1\n\", \"\n0\n\"
+       ] ,
+     ] ,
+   }
+ }";
```

Figure 6.8-1 JSON building file converted to Java String Literal

The next step is to take the string and insert it into the table holding all of the other building strings taken from JSON files. The insertion code will look somewhat similar to Figure 6.8-2, which shows the a table being retrieved from the database, followed by the insertion of the string into the table. The figure shows just how simple it is to insert the string into the database due to the sophistication of the AWS SDK and its DynamoDB library. Once inserted, retrieving the information back into its original JSON form involves retrieving the string that was inserted into the table, then decoding the string back into JSON format to get the original JSON file. Once the file is in its original form, it can then be passed into the system’s database module to parse the file into appropriate data structures. The process behind retrieving the string from the table involves code very similar to how it was inserted, but instead of creating a new Item, one calls getItem to get the string.

```
// Gives an access point into the library
DynamoDB dynamo = new DynamoDB(new AmazonDynamoDBClient());

// Gives access to the table with the given table name
Table table = dynamo.getTable("buildings");

// Creates a new item where the primary key
// is the UUID and the json string is attached.
Item item = new Item().withPrimaryKey("UUID", "B9407F30F5F8466EAFF925556B57FE6D")
    .withJSON("buildingDoc", json);

table.putItem(item);
```

Figure 6.8-2 Example code of inserting a building string into the DynamoDB building information database.

With the addition of the DynamoDB service to hold the information, the application has a dynamic starting point of searching the database for the UUID of the beacons to acquire the correct JSON file, instead of having a static file on the Android device that is read on startup. It allows for numerous buildings to be used for navigation in a dynamic, real-time environment.

## 7.0 Beacon System Prototype Construction and Coding

The following section includes a list of what parts we plan to obtain.

### 7.1 Parts Acquisition and BOM

| part                                                 | quantity      | order from:     | estimated cost |
|------------------------------------------------------|---------------|-----------------|----------------|
| 1N4001 diodes                                        | 50            | amazon.com      | \$2            |
| solderless wire jumper                               | 1 set of 100  | amazon.com      | \$6            |
| breadboard                                           | 1             | amazon.com      | \$6            |
| 3v battery(pack of 5)                                | 3             | amazon.com      | \$12           |
| Signal Conditioning<br>50ohm balun<br>Trans 2G45 ISM | 3             | mouser.com      | \$2.70         |
| 50 ohm and 100 ohm resitors                          | 100 of each   | amazon.com      | \$7            |
| LEDs                                                 | 80            | amazon.com      | \$5            |
| LCD                                                  | 1             | amazon.com      | \$11           |
| pcb                                                  | 15            | pad2pad.com     | \$120          |
| Nordic nrf51822 chip                                 | 15            | parts.arrow.com | \$44           |
| push button                                          | 40            | amazon.com      | \$4.50         |
| voltage regulator<br>L7805CV                         | 15            | amazon.com      | \$15           |
| 47 nF, 100 nF, 12 pF,1 nF, 2.2 nF capacitors         | 20-25 of each | amazon.com      | \$25-\$30      |
| 1p2t switch                                          | 30            | amazon.com      | \$5            |

7.1-1 Parts list including price and quantity

As it can be seen in the table, we will be ordering the majority of our parts from amazon.com because one of the group members has an Amazon Prime account and we can benefit from free shipping.

## 7.2 PCB Vendor and Assembly

We have looked at several websites that manufacture pcb custom boards. So far we will be manufacturing our boards with pad2pad.com. This is most likely to change once we have a working prototype built on our breadboard and have more information regarding the size of each board and the parts it will be using. Almost every website offers the boards to be ready in approximately 2 weeks. This shows that we must be certain of our design before submitting to build because we will be ordering around 15 boards and cannot afford to order a new set budget-wise and time-wise.

## 7.3 Final Coding Plan for Application

Though this section is titled “Final Coding Plan”, it should be seen as an estimate to the final code within the navigation system. Figure 7.3-1 and Figure 7.3-2 show the UML class diagrams laid out, which provide a visual presentation of the classes and their relationships to each other within the application software. In this diagram, the UserTracker class acts as the main class responsible for connecting all of the modules together. It contains the pathfinding module, the trilateration module, the Bluetooth module, and the building database module. The UserTracker will be responsible for the overall flow of the system and logic behind taking the Bluetooth module’s readings from the beacons, transferring the readings to the trilateration module to get the user’s position in the building and checking to see if the user is still on the path by taking the user’s location along with the generated path perimeter, which requires knowledge of the path computed in the pathfinding module.

The pathfinding module comprises the FloorSequencer, the Pathfinder abstract class along with the ThetaStar class that subclasses Pathfinder. An abstract class is present so that if a different pathfinding algorithm, like A\*, is needed instead, it can easily be added to the system through the object-oriented principle of polymorphism. One will notice that the FloorSequencer contains a member of type Pathfinder. Within the constructor of FloorSequencer, a ThetaStar object will be created of the type Pathfinder. If A\* were to be used, then the only place in code that would change would be in the constructor of FloorSequencer by creating an AStar object. All of the other calls to where ThetaStar is used would not have to be changed, because technically they are calling Pathfinder methods, which AStar would be a subclass of. When a path is needed, the UserTracker will call the FloorSequencer’s findPath method passing in BuildingDatabase’s nodeCollection. The FloorSequencer will then calculate the different possible elevators and stairs the user can take and pass in a floor at a time to the ThetaStar’s findPath method in the right sequence. Once the entire path is found it will be returned to the UserTracker to use to guide the user through the building. The UserTracker will also keep track of the node the user is currently at as well as the node they need to get to so that a perimeter can be calculated from the two nodes to make sure the user stays on the path.

The trilateration module just consists of the Trilateration class. It only needs one method, because it is comprised of mathematical calculations taking in the known points and the estimated distances from those points as parameters and returning the estimate location, which in the navigation system's case, is the user's position within the building.

The Bluetooth module is comprised of the BluetoothModule class. It is instantiated within the constructor of the UserTracker and is responsible for reading the beacons at a given rate. Whether it has its own loop that simply updates its own list of beacon readings at a rate allowing the UserTracker to grab the updated values or whether the UserTracker determines when the beacons should be read is still undecided. However, in the UML, the UserTracker is the one responsible for calling the BluetoothModule's getRecentBeacons method to scan and read the beacons when the method is called making the BluetoothModule the slave and the UserTracker the master.

The database module can be seen in the BuildingDatabase class and is responsible for holding all of the major data structures needed within the system. At first it seemed like a good idea to make the BuildingDatabase a singleton. However, that would create more dependencies within the code. Plus, because UserTracker has access to the parts of the system that need the data from the database and holds the building database, a singleton really is not needed because the data can be passed as parameters to the different modules held in UserTracker.

## 8.0 Beacon Location System Architecture

The beacon location system architecture describes the way in which the Google Glass interfaces with the other aspects in the system including the beacons, the database, and the user.

### 8.1.1 Architecture Summary

As shown in Figure 8.1.2-1, the architecture of the Beacon Location System centers around the Google Glass. The device, which is running an application designed to interact with the system, is responsible for interacting with all the different systems to seamlessly deliver accurate and efficient directions to the user. This architecture abstracts out the complexity from the User and only exposes them to the application running on the Google Glass.

Figure 8.2-2 depicts the interaction between the Google Glass and the Beacons. The Beacons are purposefully simple devices whose sole job is to blindly transmit the iBeacon protocol designed by Apple. As explained in Section 3.2.3, this protocol transmits enough information to calculate the User's distance from the Beacon. By using multiple Beacons and providing the necessary context, the device is also able to derive a close estimate of the User's location within a building. Combined with its simplicity, the use of Bluetooth Low Energy technology allows these Beacons to be extremely power efficient making it



capable of being powered off a coin cell battery for over a year. Its use of BLE also means it is compatible with most devices available in the market such as Android, iOS, and Windows Phone.

Figure 8.2-3 depicts the interactions between the Database and the Google Glass. As previously mentioned, extra context is required in order to derive the User's location from the Beacons. This context is stored on Database, allowing the Google Glass easy access to any information it needs. Once a user enters a building using this system, the application running on Google Glass is able to request information on the building by simply making a request to the Database using the UUID of Beacons it detects. The database will respond to this request with a JSON (JavaScript Object Notation) representation of the building. This response will contain all the information about the building including possible destinations, floor layouts, and elevators/stairs. The application will then use this information to carry out the commands given to the Google Glass by the User.

Figure 8.2-4 depicts the interactions between the Google Glass and the User. The Google Glass, running the systems application, is responsible for accurately directing the user to their chosen destination. The user is able to specify where they want to be guided by simply speaking the Glass. The Glass takes the users speech and feeds it into the application where it can be analyzed and matched to a specific destination found in the building JSON object. With the users chosen destination, the Glass is now able to feed directions to User by using visual and audible cues. The User is also able to interact with the Glass by administering touch gestures to the touchpad located on the side of the device. These gestures serve to wake the Glass before starting the application and to stop the application while it is giving directions.

## 8.2 Architecture Diagrams

The following diagrams illustrate the architecture of the project. Figure 8.2-1 shows a complete overview of the systems and how they interact with each other. Figure 8.2-2 shows the communication between the Beacons and the Google Glass in more detail. The Beacons simply send the iBeacon protocol which is then consumed by the Google Glass. No communication is done from the Google Glass to the Beacons. Figure 8.2-3 shows the communication between the Google Glass and the Database. The Google Glass sends a request to the database containing the UUID for a Building. The database then responds with the data for the Building. Lastly, Figure 8.2-4 shows the communication between the User and the Google Glass. The User uses their voice in order to launch the application and specify a destination. In return the Google Glass communicates turn-by-turn directions and general information to the user (both visually and vocally).

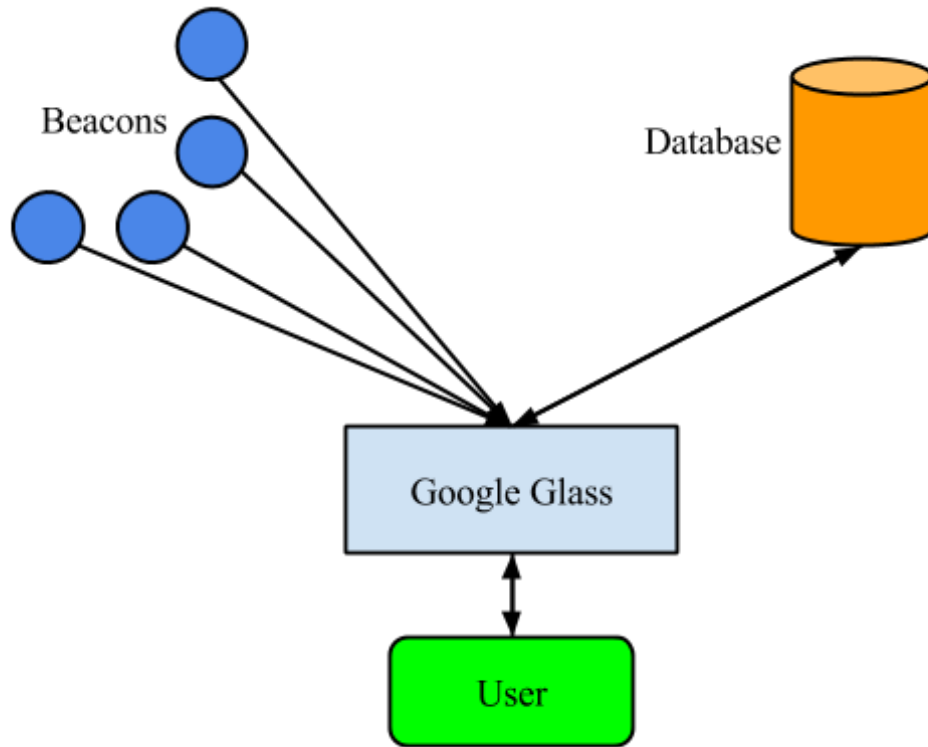


Figure 8.2-1: High level view of all system interactions

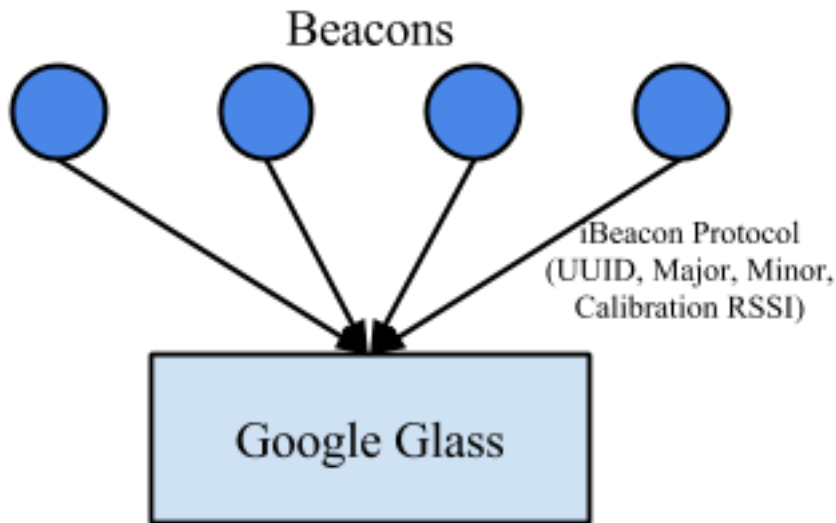


Figure 8.2-2: High level view of the interactions between the Beacons and the Google Glass

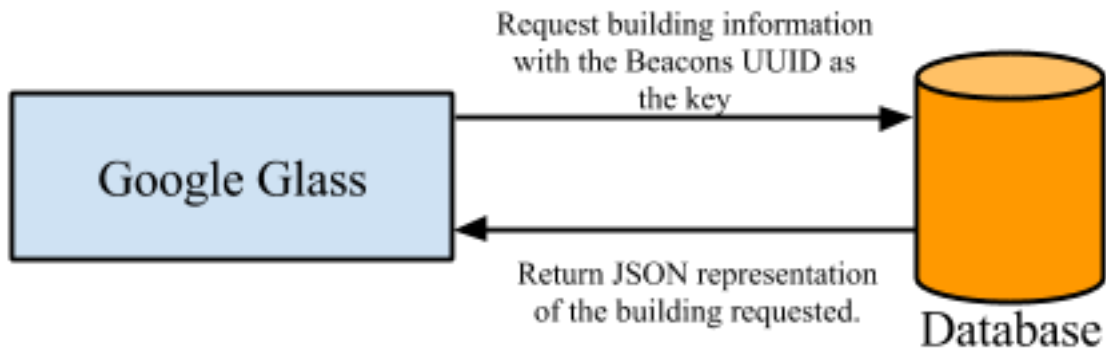


Figure 8.2-3: High level view of the interactions between the Google Glass and the Database

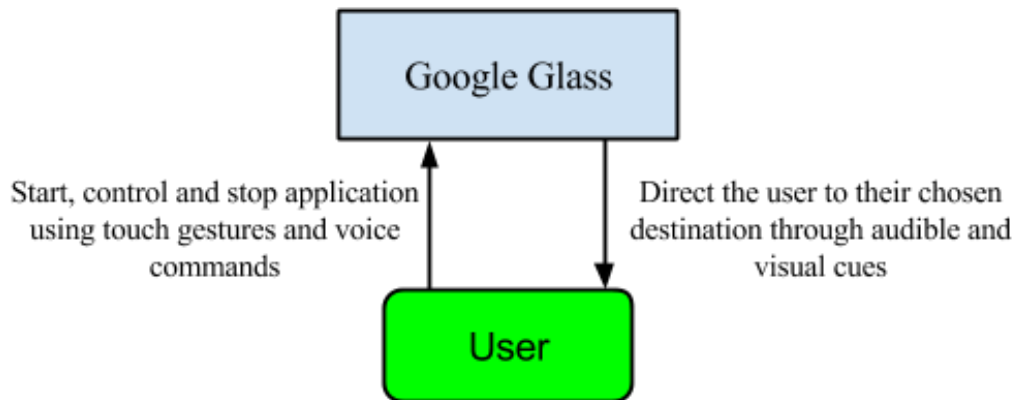


Figure 8.2-4: High level view of the interactions between the Google Glass and the User

## 9.0 Beacon Prototype Testing

This section goes over how we plan to test our project.

### 9.1 System Testing

Once all the different systems making up the Beacon Location System are deployed for real-world testing, the system as a whole will finally be able to undergo some tests. Previously to this, each component could only be tested individually with mocked data. These real-world tests will demonstrate how each of the separate systems behaves when they are finally interacting with one another. Therefore, it is important to keep track of how each component of the system behaves when they are not longer being tested in isolation of each other and how their performance compared to the isolated testing previously done.

In order to test the system, the fourth floor of the Engineering 1 will be mapped out. The mapped out floor along with the locations of different possible destinations are bundled together to create a JSON representation of the floor. This JSON object will then be added

to the database so the Application can have access to the information it requires. Next, the fleet of Beacons will be deployed to their appropriate locations on the fourth floor.

With the environment set up, tests can be carried out. Many of the tests, which were mocked out during the integration tests for the Application, will have to be carried out again, but with real data instead of mocked data. It will be important to note how the application performs in these tests in comparison to the integration tests. It will also be important to test out the actual accuracy of the location being calculated by the Application to ensure that it satisfies the expected level of accuracy. These tests will also allow for viewing the application's impact on battery life and component temperatures. It is especially pertinent to ensure that the device does not overheat due to all the computations being done. If it does overheat, the device will shut off until its temperature is safe enough to operate. Another test, which will need to be carried out, is whether the application will be able to correctly identify which building it's in and pull down the correct data from the database. Lastly, it is important to test that the measures taken against Beacon spoofing correctly filter out the fake Beacons.

It will also be important to carry out tests on the Beacons themselves. It is important to test that they are transmitting the correct data and that antenna is propagating the signal in the expected manner. This can be done by measuring the strength of the signal at different distances and angles from the Beacon. It is also important to test all the power systems being used in the Beacons. First it is important to make sure that the battery is being drained or overused, as this would greatly impact the great battery life expected from the device. It is also important to check that the power source switching is working correctly only causes a minimal (if any) interruption to the transmission of the system.

Last it is important to check for heat being output by the modules. In theory, the Beacons should remain fairly cool when in use so unexpected heat being generated could mean errors in the hardware.

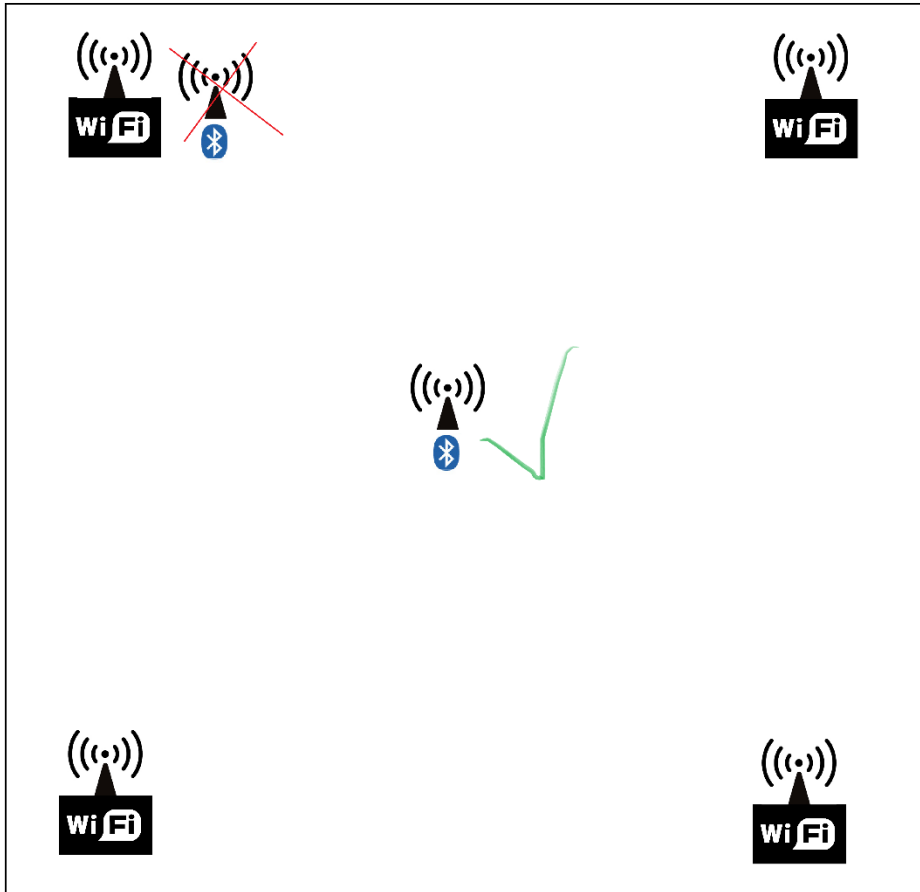
## 9.2 Hardware Test Environment

The following section aims to discuss the method and processes of planning and prototyping the hardware beacon design in the test environment – i.e. indoor serving area in which our indoor positioning system will operating in.

### 9.2.1 Site Survey

The installation of any wireless infrastructure requires crucial planning in order to deliver optimal performance with upmost efficiency. Several factors should be actively sought out for when designing a system which operates in an indoor setting. This indoor positioning system aims to be installed in an indoor college/ university setting so it is expected to have other interfering radio frequency emitters as well as other obstructions from the environment.

A survey of the site in which these beacons will be installed will be required in order to avoid placing the beacons in troublesome areas. The area in which the beacons will be installed is most likely laden with interfering 2.4 GHz wireless protocols such as WiFi or even other Bluetooth wireless protocol transmissions which will definitely affect Bluetooth Smart beacon performance even with its frequency hopping schemes. However, since WiFi transmission nodes are most likely operational 24/7, a site survey can easily locate the areas in which there are a high amount of interference.



The above diagram shows incorrect beacon placement in crossed-out red due to its close proximity to a WiFi router. Bluetooth Smart based beacon for indoor positioning system would function best away from strong sources of similar frequency interference.

Luckily, a site survey for our indoor positioning system is much more simplistic than a site survey for a wireless protocol such as WiFi since there is no concern about traffic congestion in each beacon node – i.e. beacons transmit only and can still serve multiple users using our indoor positioning system much like how global positioning satellite systems can serve an infinite number of users since all calculations for the user's position are done on the receiving device.

In order to save time and money, a quick and cheap method of scouting out the indoor serving area is to be devised. There is extensive literature available online for WiFi site survey methods which can be applied to the Bluetooth Smart wireless protocol having a similar working radio frequency band of around 2.4 GHz. Often, wireless infrastructure designers utilize expensive but thorough survey kits with specialized, portable equipment in order to achieve upmost wireless performance. However, as starving college students, site surveys will be done in an inexpensive manner using software which can monitor a site for heavy 2.4 GHz traffic - i.e. Bluetooth and WiFi. The companies “ekahau” and “NetSpot” offer free WiFi site survey software which could help find heavy traffic areas that might cause interference with 2.4 GHz devices. This free software requires the usage of a WiFi compatible laptop and time to scan and survey the indoor serving area. This will definitely not be as thorough as using specialized site survey equipment or spectrum analyzers.

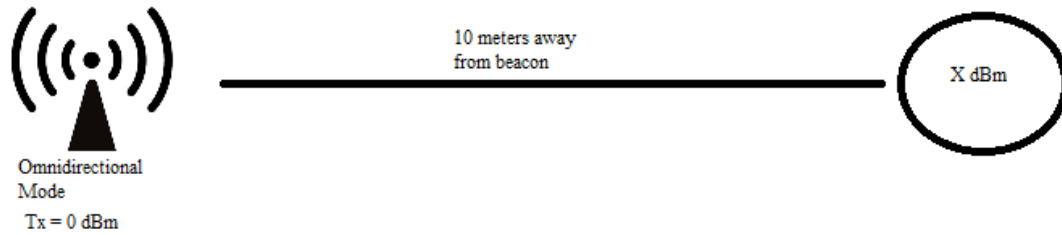
## 9.2.2 Signal Strength Testing

The indoor positioning system relies on measurements of received signal strength indication (RSSI) in order to perform complex trilateration algorithms which derives the user’s position. The indoor site will very likely attenuate beacon signals due to obstructions, distance limits, and interference from other. The aforementioned site survey method can only attempt to avoid obvious major sites of interference and cannot alone ensure ideal Bluetooth wireless performance. Also, each device will have its own signal strength associated with each device antenna configuration. Actual device prototype testing of the performance of each device module is necessary to ensure optimum operation of our indoor positioning system.

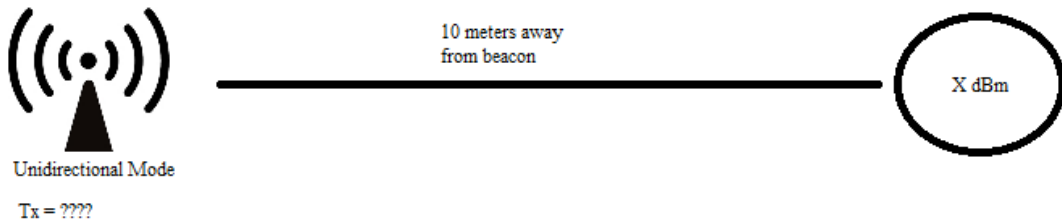
The RSSI can be most easily measured using any Bluetooth Smart compatible device such as the latest Android or Windows 8 OS with an installed Bluetooth Smart hardware. Free software is available out there which can effectively measure the RSSI of Bluetooth Smart transmitters – e.g. “Bluetooth Signal” is an app available in Google Play created by user nakaborigawa. All it requires is that the Bluetooth device is in discoverable mode. The app can display the perceived signal strength of the user device in units of decibel-milliwatts.

It is expected that, under the same transmission power, the unidirectional antenna design should have a greater signal strength (using signal strength synonymous to RSSI) than the omnidirectional antenna design in a given distance (i.e. given distance in the operating direction of the Yagi unidirectional antenna). The unidirectional antenna design transmission power should be calibrated in order to find the experimental value in which both signal strengths are equal. Thus, use the RSSI measurement software to find the point at which the unidirectional antenna reaches the same gain as the omnidirectional antenna at the same distance which would give us the transmission power value at which the unidirectional antenna can be configured at in order to reduce power consumption and provide roughly the same gain and, by extension, range. This provides a unique power

saving configuration for module devices set in a unidirectional configuration.

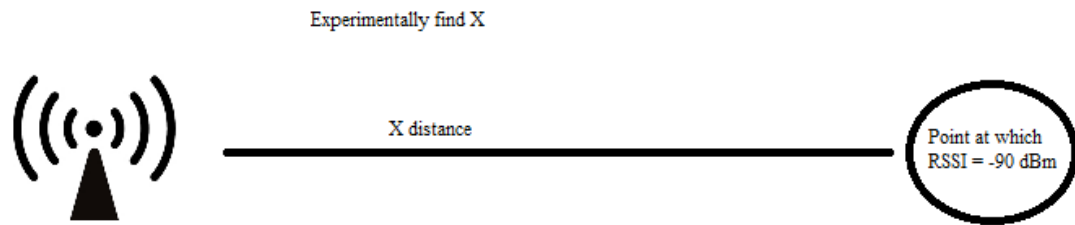


Solve for Tx of unidirectional beacon configuration



The above diagram shows a summary of the calibration procedure to create a much more power efficient directional antenna which can serve an area with the same operation range and performance as the omnidirectional configuration (albeit, only in one direction).

The aforementioned RSSI measurement software should also be used in a simple test procedure to obtain a general gauge on the operating range of the beacon modules in both antenna configurations. It should be noted that signal strength measurements less than -90 dBm is perceived as unsatisfactory quality for Bluetooth communications (note, perceived signal strength is measured in negative decibels; 0 dBm is greatest quality and -90 dBm is terrible quality). Each powered on beacon device (this is done on both antenna configurations) should be placed in a static location with relatively few obstructions and obstacles – e.g. outdoor field. The RSSI is to be measured continuously moving away from the source transmitter. Once the RSSI measures to be below -90 dBm, measure the distance from the point at which the RSSI measured -90 dBm and the transmitting beacon device. This will be the maximum experimental operating range that will be used in reference to further beacon placement. It should be emphasized that this procedure should be performed with both unidirectional and omnidirectional antenna configuration modes.



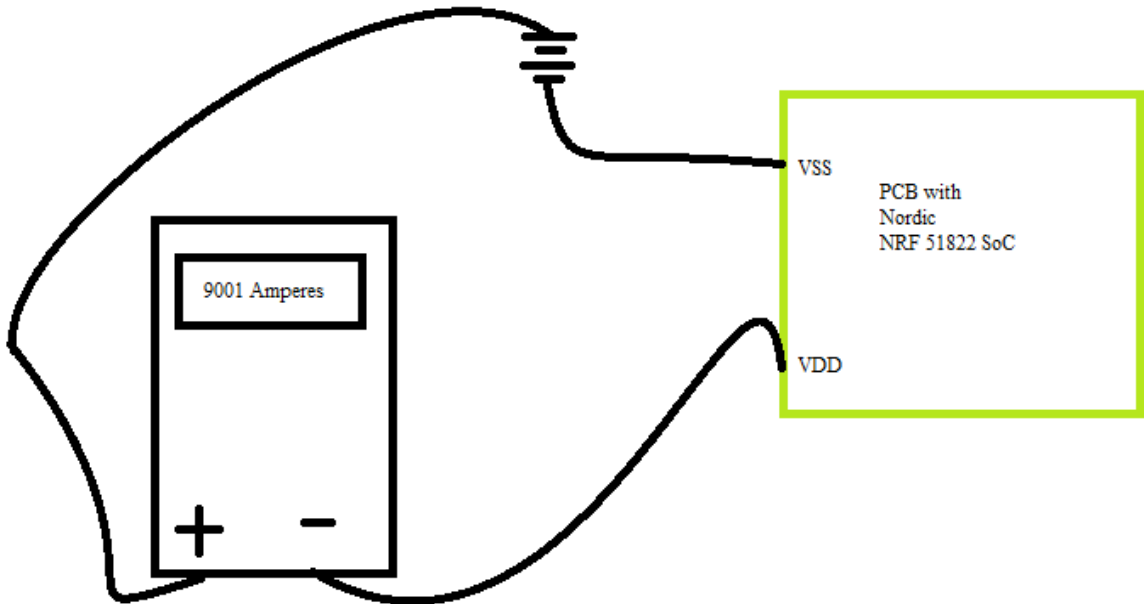
Above shows summary of experimental procedure to find the max operational range of the constructed beacon devices.

Previously acquired experimental values for the beacon maximum operating range is a best case scenario and the indoor RSSI will very likely underperform this operating range. Thus, after the placement of the indoor positioning system beacons, sweeps of the service area should be performed measuring the RSSI at every relevant point to insure that at least three beacons are able to make a satisfactory connection.

### 9.2.3 Battery Tests

A key design feature of this indoor positioning system is high-efficiency, low power consumption. The system's current draw during normal operations must be tested in order to ensure proper power efficiency. This test can be simply performed using a multimeter in a laboratory setting. Configure the meter in series configuration between the voltage source and the Nordic system on a chip and measure the current flow between the two. Assuming that the voltage source is a constant, the power usage of the device can be measured by multiplying the voltage times the averaged current flow squared.





The above figure shows a sample test bed for measuring the power consumption of the assembled beacon device.

The battery longevity on the receiving user device must also be taken into account in order to test the power efficiency of the software application side of the indoor positioning system. Our key users will be using Android OS based devices such as the Google Glass or Android Smart Phones. These devices have built in battery diagnostics and analysis that can check how much of a battery hog the indoor positioning system software is and can decide whether or not further software optimization is required before releasing the product for public utilization.

## 9.2.4 Antenna Impedance Testing

After the PCB antennas are assembled, they should be tested for impedance so that additional capacitive elements can be added to match the antenna system to the proper impedance and thus retain optimum efficiency for transmission power. Network analyzers capable of working in the UHF range can be used to check for properly matched impedances of antennas. However, network analyzers are very expensive, can be easily disabled without the proper training, and thus have limited access in the UCF EECS department.

After properly utilizing the network analyzer on the PCB antennas, the resulting display should give an adequate transmission range in the Bluetooth Smart frequency range of 2.4 GHz to 2.48 GHz.

## 9.3 Software Test Environment

This section explores the different measures put in place to both test the code and ensure its quality.

### 9.3.1 Application-specific Testing

In order to assure that the code for the application has the least amount of bugs as possible, the application will have to go through four different test environments. First, in order to ensure the integrity of the basic functionality of the codebase, unit tests will be used. These tests will mainly deal with testing code which accepts an input, carries out some actions with the input and generates an output (parses data or calculates values) and not with UI-related code. These tests will test modules individually to ensure that each part of the codebase works independently of each other. In order to write and execute these tests, Junit will be used in conjunction with Gradle. The tests will be run whenever the application is built. If any of these tests fail, the build will also fail in order to grab the developer's attention.

One of the tests within the trilateration module will involve passing known points and distances to those points to see if the result matches with the known values. Also, because the algorithm used can take in multiple known points along with their respective measurements, tests will be done to see the best number of known points to use to have the most efficient and accurate estimation of the user's position. Too many points used and the calculations may take too long, but too few points used and the estimated calculate position may be too inaccurate, so a right balance must be achieved and determined in this unit test.

The pathfinding module will undergo two main unit tests. The first test that will be conducted will involve the class that holds the Theta\* algorithm and will involve passing through test nodes along with a two-dimensional test search area to see if the path calculated is the shortest possible path. Also, the nodes in the path will be checked to see if the neighbors are within line of sight of each other, because Theta\* calculates the path that requires the fewest turns. The other main test will incorporate the floor sequencer and calculating the path between nodes that are on separate floors. Just like the first pathfinding test, a variety of test nodes will be passed in on the same floor and on different floors along with the three-dimensional search area that represents the entire building's node mappings.

The second test stage will be code reviews. Since the codebase is being stored on Github, it is possible for developers to review each other's code before it is submitted to the main branch. Each developer will make changes inside their own branch. Once the developer is done with their changes, they can push their code to their remote branch. With the new changes in the developer's branch, the other developer can review and comment on the

changes before it is merged into the main branch. This will ensure that mistakes that were not caught by author work make it into the main branch of the repository. Using Github also allows for the use of git which will help keep track of the codebase's history and will allow for unwanted or breaking changes to be reverted.

The next testing stage is integration testing. These tests would finally put together all the different modules and test how they interact. Artificial data would be generated and injected into the application. The tests would ensure that the application is behaving in an expected manner for the given data. The application would be run on either an emulator or on the actual device.

The final testing stage is manual testing in a real-world environment. With an area mapped out and with Beacons installed, the application would be tested by the developers or by volunteers to ensure that the application is ready for release. In order to pass, the application has to be able to efficiently direct the user to their desired location with little to no mistakes. At this stage, the user interface has to work as expected and the application must be easy to use and intuitive. At this stage, the applications impact on the hardware will also be tested to ensure that it doesn't cause the device to overheat or that it doesn't drain the battery.

In order for the application to be considered done, it has to pass all the test stages. Whenever issues arise in any of the stages, they are added to the corresponding repository on Github. This will make it easier for the developers to keep track of existing and past issues.

## 9.3.2 Beacon Firmware-specific Testing

The Beacon firmware will have to go through a similar range of testing environments in order to assure the quality of the codebase. It goes through three different testing stages. The first stage will be unit tests. These tests will be written using the MinUnit testing framework and will be automatically carried out during the build process using GNU Make. These tests will mainly test logic within the code will test modules individually. All these tests should pass before a developer pushes code to the repository.

Much like with the Application, the second testing stage is code reviews. All the code regarding the Beacon firmware will be stored on Github. Developers will make changes to their own local branch and when those changes are ready, they will push their changes to their remote branch. Before their changes are merged into the master branch, the other developer must review it.

The final testing stage is manual testing in a real-world environment. The firmware will be built using the latest changes from the repositories and will then be flashed onto an actual Beacon. The Beacon will then be tested to ensure it is correctly transmitting the iBeacon signal. If possible, longer term testing will be done to ensure other aspects such as battery life are not being negatively affected by the current version of the firmware. Once this stage of testing is successfully completed, the firmware can be considered complete.

## 10.0 Administrative Content

This section will go over our schedule to finish this project on time and our budget considerations.

### 10.1 Milestone Discussion

We have learned in senior design that not everything works out as planned, but we have been able to manage our time efficiently and meet at least once every two weeks in order to discuss each other's progress and make decisions. Our team has been good in cooperating and making decision that benefit the outcome of our final product. We have our designs, but will use the winter break in order to run simulations and improve our designs, that way when the get back next semester we can start building the devices and be ready for testing within a month. Besides working on improving our designs, we will also work on finishing our software code during the winter break.

The following is our proposed schedule for next semester:

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| 12-10-14 to 1-7-15 | improve designs, run computer simulation, and work on software code |
| 1-7-15             | order necessary parts                                               |
| 1-12-15 to 1-19-14 | analyze best location for beacons in Engineering building 4th floor |
| 1-20-15 to 2-3-15  | build first beacon on breadboard and test it functions properly     |
| 2-4-15 to 2-11-15  | have the rest of the beacons made                                   |
| 2-11-15            | software code is finished                                           |
| 2-12-15 to 2-19-15 | run first set of tests and gather results to improve the software   |
| 2-20-15 to 2-25-15 | fix problems that came up during first testing                      |
| 2-26-15 to 3-1-15  | run second round of tests                                           |
| 3-2-15 to 3-7-15   | spring break- fix problems that have occurred during testing        |
| 3-10-15            | have final product working                                          |

Figure 10.1-1 Milestone Schedule for next semester

This proposed schedule will leave us with several weeks left in the semester for problems that might arise with the software, building the circuit, or testing the final product. We will also be meeting multiple times a week as a group, but since we will all be having senior design 2 at noon on Fridays, our official weekly meeting will take place on Fridays at 9am.

During our meetings we will discuss our individual progress to the project and if we have come up with a better solution or approach for our software.

## 10.2 Budget and Finance Discussion

The group decided to seek sponsorship from a local small company that is currently working in creating beacon devices and implementing a software to use the devices for indoor navigation. Even though the company is working on a similar project as ours, they decided to not to sponsor us because it was not the right time for them, we believe because they would feel forced to help us when we run into a problem.

We decided to fund our own project because it will not be too much money since half of our project will be writing the software in order for the indoor navigation system to function properly. We expect to spend around \$500 total on this project. We already spent \$130 on getting equipment to test and make sure what we want to achieve is possible. The rest of our budget we plan to spend on acquiring parts and making the pcb for our devices.

We felt that it was too much trouble to go through to acquire a sponsorship just so the company can take our project and never use it again. By funding our own project we plan to make all our information open source. We believe that by making our software available to the internet, it will be more beneficial for future senior design teams to improve our project and one day be a product that can help the community as we intended. We hope that a future senior design team can make our project a product that can help visually impaired people navigate more easily indoors, like the UCF library for example.

Since there are four team members, we will be splitting up the cost of all parts and equipment we buy. One group member will keep the beacon devices after we are done with the project because he has an interest in working with the software in the future to make this a better product, but the other team members have other interests to seek and learn about. We want to use this senior design project as an experience to become familiar with the industry of engineering.

Many job postings say they seek a candidate that can manage time efficiently and stay within budget, this project will help us learn to maintain a budget, especially since we will be funding our own project and will have to make decisions that save money and time. During presentations from engineers that work in the field they always mention how changing one small detail in a design can save a few cents in cost production and still maintain the same performance because millions of the same devices will be made in mass production.

To a small scale we will be doing something similar with our beacon devices, we will have our designs and from that one design we will create a prototype to test and will then create at least one dozen beacon. So if we can improve our design in a way that we can save one dollar per beacon built without affecting the performance, which will be an advantage to us. When we are sitting in an interview and the hiring manager asks us if we are capable of stating within budget, we elaborate on our experience with this senior design project.

Compared to other senior design teams in our class we do have a low budget, and this is because they are spending more money on hardware than us, a big portion of our project will consist on writing the software code to implement the indoor navigation as compared to other projects that might require a small amount of code to make the project work. This is another reason we decided to do this project, it is a low budget project on a new technology that big companies such as Microsoft are spending a lot of money and research on, and ten years from now when there will be beacons set up in every mall and supermarkets, we will already have a good understanding of how the system works.

## 11.0 Specifications and Standards

The following section talks about different standards and specifications taken into consideration while completing the project. These were critical to creating a viable product that could be used in real world applications.

### 11.1 Software

The application running on Google Glass had many specifications to follow. First, much of the functionality of the application required special permissions such as internet access, control over Bluetooth, and access to the wake lock. According to Android specifications, these permissions must be defined in the application's `AndroidManifest.xml` file. If the permissions are not properly defined, the application will simply not be able to use much of the required functionality of the device. Another Android specification was to keep the method count under 65,000. Any methods defined in libraries used by the application count towards this total. Special consideration had to be taken in order to use only light-weight Java libraries or Android-specific libraries as some larger libraries could cause the application to reach this limit. Lastly, the some Google Glass-specific specification had to be taken into consideration. Google strictly specifies that application written for Google Glass must be simple, predictable, and unobtrusive. Due to the nature of Google Glass, it provides a very intimate experience between the user and the device. Therefore, the application must behave in a predictable fashion as to not disturb the user. The application should also follow the style guide stating that application should use a dark background with light text in order to save battery and to not create an image that could fatigue the user's vision.

### 11.2 Firmware

The firmware running on the beacon's nrf51822 system-on-chip uses Nordic's SoftDevice to handle much of the low-level communication and flash storage processes. S110 SoftDevice v7.0 is the specific version used within the beacons. The SoftDevice allows for a clean separation between the beacon application and the bare metal hardware of the ARM Cortex-M0 processor. It provides libraries that allow for simple flash storage management, Bluetooth stack initialization, and control of other modules within nrf51822 chip.

Compilation of the beacon application source code utilizes the Keil compiler. Flashing the nrf51822 SoC involves connecting the development machine to the chip through

SEGGER's J-Link Lite ARM, which is a fully-functioning version of SEGGER J-Link. J-Link utilizes a standard 20 pin 0.1 inch male JTAG/SWD connector and requires a 3.3 V target interface voltage. It also supports a number of different ARM CPUs including the ARM7/9/11, Cortex-A5/A8/A9, as well as the Cortex-M family of processors.

## 11.3 Hardware

Our hardware implementations of beacon transmitters rely on several standards. Primarily, the communications protocol we are utilizing is Bluetooth 4.0 LE which is standardized by IEEE as IEEE 802.15.1. This standard consists of WPAN and Bluetooth. It is defined by the physical layer (PHY) and media access control (MAC). Due to specifications of this standard, our design had limited transmission power as Bluetooth is specified to be a short-range RF standard with limited transmission power. IEEE has listed many restrictions for using Bluetooth as it is meant to be used in a congested RF spectrum for only short range communications. The Bluetooth low-energy 4.0 updates more specifications to the standard restricting data throughput even more. Luckily, these restrictions did not greatly impede the performance of our indoor navigation system.

Consumer wireless transmitter products must also be FCC compliant. If we were to release a commercial iteration of our Beacon Indoor Navigation System, we would have to receive FCC compliance and certification in order to commercially sell and use our product. FCC compliance is necessary in order to limit the instances of commercial wireless products sending out spurious this would require potential redesign and optimization of the entire PCB and electronic components as FCC compliance testing for RF often fails the first test. This redesigning would hinder the deployment of our product as time and money would be required to create an additional iteration of our beacon products.

The batteries we use on this project follow IEC 60086 international standards. The IEC 60086 standard provides physical dimensions, we use 2032 batteries because of the compact size. The compact size of the battery allows us to build a compact device. The IEC 60086 standard also provides discharge test conditions and discharge performance requirement, these standards were taken into consideration and did not affect our project. IEC 60086-4 is the international standard for safety, we did not encounter any safety issues regarding batteries during this project as all safety protocols were followed.

We also took into consideration storing our pcb components. The ICL 7673 chip has small pins that can easily be broken. The nrf 51822 is a small chip that can easily be lost because it is as small as 6mm by 6mm. All components must be stored at ambient room temperature, we did not leave any components such as chips, capacitors, or inductors in a car all day long that could have impacted the performance after use.

When creating our prototype, we soldered a nrf51822 chip to a test board using solder paste type 3, which is considered the industry standard. Throughout this process we learned of many considerations that must be taken into account when soldering a qfn chip.

# Appendices

Copyright – Texas Instruments (<http://www.ti.com/corp/docs/legal/copyright.shtml>) :

“Texas Instruments is pleased to provide the information on these pages of the World Wide Web. We encourage you to read and use this information in developing new products.

TI grants permission to download, print copies, store downloaded files on a computer and reference this information in your documents only for your personal and non-commercial use. But remember, TI retains its copyright in all of this information. This means that you may not further display, reproduce, or distribute this information without permission from Texas Instruments. This also means you may not, without our permission, "mirror" this information on your own server, or modify or re-use this information on another system.

TI further grants permission to non-profit, educational institutions (specifically K-12, universities and community colleges) to download, reproduce, display and distribute the information on these pages solely for use in the classroom. This permission is conditioned on not modifying the information, retaining all copyright notices and including on all reproduced information the following credit line: "Courtesy of Texas Instruments". Please send us a note describing your use of this information under the permission granted in this paragraph. Send the note and describe the use according to the request for permission explained below.”



## Works Cited

- [1] Estimote, [Online]. Available: <http://estimote.com>. [Accessed November 2014].
- [2] A. Allan and S. Mistry, "Makezine," 3 January 2014. [Online]. Available: <http://makezine.com/2014/01/03/reverse-engineering-the-estimote/>. [Accessed November 2014].
- [3] SPREO, [Online]. Available: <http://spreo.co>. [Accessed November 2014].
- [4] SPREO, [Online]. Available: <http://spreo.co/technology/indoor-navigation-positioning-technology/>. [Accessed November 2014].
- [5] Motorola Solution, [Online]. Available: <http://mpact.motorolasolutions.com>. [Accessed November 2014].
- [6] Indoors Solutions, [Online]. Available: <http://indoo.rs>.
- [7] Florian, 21 July 2014. [Online]. Available: <http://indoo.rs/indoo-rs-and-san-francisco-international-airport-unveil-app-for-visually-impaired-passengers/>. [Accessed November 2014].
- [8] P. Sawyer, "Microsoft pilots 3D audio technology to help blind people navigate," The Next Web, 06 11 2014. [Online]. Available: <http://thenextweb.com/microsoft/2014/11/06/microsoft-pilots-3d-audio-technology-help-blind-people-navigate/>. [Accessed 11 2014].
- [9] E. R. Council, "A brighter future? Anti-ageing treatment for solar panels," [Online]. Available: <http://erc.europa.eu/erc-stories/brighter-future-anti-ageing-treatment-solar-panels>. [Accessed October 2014].
- [10] Apple Inc, 2 June 2014. [Online]. Available: <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>. [Accessed November 2014].
- [11] D. Thompson, "Beekn," 19 July 2014. [Online]. Available: <http://beekn.net/2014/07/ibeacon-for-android/>. [Accessed November 2014].
- [12] M. Missfeldt, February 2013. [Online]. Available: <http://www.brillen-sehhilfen.de/en/googleglass/>. [Accessed November 2014].
- [13] J. Donovan, "Energy Harvesting," [Online]. Available: <http://www.mouser.com/applications/energy-harvesting-new-applications/>. [Accessed November 2014].

- [14] Alternative energy news, "Battery Technology," [Online]. Available: <http://www.alternative-energy-news.info/technology/battery-power/>. [Accessed November 2014].
- [15] R. Whitman, "Researchers cannibalize old car tires to avert the lithium-ion battery crisis," *Extreme Tech*, 01 September 2014. [Online]. Available: <http://www.extremetech.com/extreme/188963-researchers-cannibalize-old-car-tires-to-avert-the-lithium-ion-battery-crisis>. [Accessed November 2014].
- [16] S. H. ,. S. T. a. J. S. Artem Dementyev, "Power Consumption Analysis of Bluetooth Low Energy, ZigBee and ANT Sensor Nodes in a Cyclic Sleep Scenario," *Proceedings of IEEE International Wireless Symposium (IWS)*, 2013.
- [17] S. L. ,. A. M. ,. K. S. P. Thomas Watteyne, "Mitigating Multipath Fading Through Channel Hopping in Wireless Sensor Networks," *Communications (ICC), 2010 IEEE International Conference on*, pp. 1 - 5, 2010.
- [18] Alex West, "Smartphone, the key for Bluetooth® low energy technology," [Online]. Available: <http://www.bluetooth.com/Pages/Smartphones.aspx>. [Accessed 5 11 2014].
- [19] R. Wallace, "Antenna Selection Guide," [Online]. Available: <http://www.ti.com/lit/an/swra161b/swra161b.pdf>. [Accessed 2014].
- [20] A. Andersen, "2.4 GHz Inverted F Antenna," [Online]. Available: <http://www.ti.com/lit/an/swru120b/swru120b.pdf>. [Accessed 2014].
- [21] R. W. &. S. Dunbar, "2.4 GHz YAGI PCB Antenna," [Online]. Available: <http://www.ti.com/lit/an/swra350/swra350.pdf>. [Accessed 2014].
- [22] H. D. A. C. P. A. R. a. D. H. P. Joan E, "Age-related changes in speed of walking," *MEDICINE AND SCIENCE IN SPORTS AND EXERCISE*, vol. 20, pp. 161-166, 1988.
- [23] S. K. West, G. S. Rubin, A. T. Broman, B. Muñoz, K. Bandeen-Roche and K. Turano, "How Does Visual Impairment Affect Performance on Tasks of Everyday Life?: The SEE Project.," *Arch Ophthalmol.*, pp. 774-780, 2002.
- [24] Y. Zhou, "An Efficient Least-Squares Trilateration Algorithm for Mobile Robot Localization," *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3474 - 3479, 2009.