**Team 1: Precision Beacon Navigation**

**Final Project Report**

**Tahir Aziz**

**Adeel Khan**

**Sabur Khan**

**Alejandro Guzman**

**Casey Boyle**

**04/28/17**

**Executive Summary**

In this senior design project, we were tasked with discovering the feasibility of doing indoor navigation with proximity Bluetooth beacons. The long term goal of this project is to be able to allow people who are visually impaired to navigate inside a building using an application.

We set out and put together a project management plan that would allow us to put together a strategy for the rest of the semester and establish basic guidelines for ourselves. After that, we put together the requirements based on the end goal of the product, and also decided on an architecture (MVC) to use for that. We wrote a detailed design document as well as a testing plan. We conducted some real world tests using the application and logged the data for that, so that will be another section of the report. Additionally, at the end of the report we included a conclusion as well as suggestions for the future directions of the project.

We inherited the codebases of past teams and started with understanding the general logic and workings of the code. We merged the codebase, and after that, started to work on a few tweaks such as adding adding in an image processing algorithm that detected walls based on the image map that came with the codebase, as well as adding a GUI for real time configuration of various settings. Additionally, we did some basic research on Bluetooth and beacons so we could better understand the area that we were working in. We also started working to understand the trilateration and tried to improve it as well attempt to improve the RSSI smoothing algorithm that was in place.

We began trying to implement the end goal of the application and realized a few big things:

1. The beacons were proximity, not location based and so that means they are better used for general placement rather than high accuracy location.
2. The beacons emit just 2 RSSI values per second
3. The current implementation of the RSSI smoothing (to account for signal bouncing) took in a queue of values from each beacon, returned the average, which then passed that smoothed value to the trilateration algorithm which estimates where the user is relative to the beacons.
4. The bottleneck thus occurs in 2 areas
   1. The frequency of emission
   2. High bouncing (which leads to inaccurate RSSI values) due to the hallways + Bluetooth being at the same frequency as WIFI
5. The RSSI readings are bouncy and inaccurate even while standing still.

We decided to run some tests to get more concrete numbers and analyze the data. We ran different simulations with different independent variables such as varying walking vs standing, different beacon placement distances, and queue lengths used for the RSSI algorithms. Those findings are explained more in the Research Tests section.

Overall, this semester was more about consolidating the previous groups’ work, setting a strategy for this semester, and determining the feasibility of the proposed application.

**Table of Contents**

1. **Introduction**
2. **Project Management Plan**
3. **Requirements Specification**
4. **Architecture**
5. **Design**
6. **Test Plan**
7. **Research Tests**
8. **Conclusions & Future Directions**
9. **Acknowledgement References**

**1. Introduction**

**1.1. Purpose and Scope**

The purpose of this document is to report the various activities and tasks we accomplished relative to our senior design project this semester. While going through the process, we initially made a plan for the high level strategy as well as the various technical strategies for the project. This report will outline those initial plans in addition to the additional work we did, while also providing an update for what we accomplished as well as what we weren’t able to get to and the reasons for that. The scope of this report includes an executive summary, an introduction, the same reports we submitted through the semester which include the project management plan, requirements, architecture, design and testing. Within the test plan section we also included a section for experimental style tests we ran since a major part of our project was to analyze feasibility of beacon navigation. We also have a future directions section since this is a long term project and we want to outline basic conclusions from our findings this semester and how that will relate to the future strategy of this project.

**1.2. Product Overview (including capabilities, scenarios for using the product, etc.)**

This is a long term project so the end goal functionality is not yet in place. We will outline the product’s status before we inherited it, what we did, what needs to happen in the future, as well as the end goal of the product.

Previously:

There were two codebases we inherited. One was a Walking Navigator, which had a image map of the ECSS 3rd floor, and was able to display a route between a default starting location and a user selected destination. It processed the image so that it was navigable in terms of X/Y coordinates. Another codebase we inherited focused on the beacon portion of the project. The user was able to add nearby beacons and through their emitted RSSI signals, it would estimate where the user would be relative to the beacons.

Currently:

During the course of our project we merged the two codebases so the functionality of the route calculation on the image map and the beacon placement. Additionally some image processing was added so that walls are detected. Another piece of functionality that was built for helping us test the application was a panel for configurable settings that could be edited in real time rather than setting them in the code and recompiling the entire application. We attempted to implement another trilateration algorithm but it didn’t solve the original problems of low emission frequency and RSSI bouncing (mentioned in the executive summary). Additionally, for testing purposes logging functionality was built in to output the RSSI values that corresponded to each beacon so that we could analyze the RSSI values. Essentially, at this point, the user can open the application, select the destination room, see the route to get there, and place beacons on the map that will be used to estimate the user’s position. Currently, this product is not ready to be used by the public and the target use case of visually impaired people due to the identified bottlenecks in the RSSI values as well as the fact that due to these limitations, we have tried looking into solutions to fix these problems rather than develop the features needed for the end product. Without proper beacon trilateration, the application cannot fulfill its main purpose.

Future:

In the future, a solution needs to be attempted to be found in order to reduce signal bouncing of RSSI values in narrow and high signal areas such as hallways. After that, more attempts can be made to implement better RSSI smoothing and trilateration algorithms that can more accurately place a user in a location if needed. The user interface currently is mainly to visualize the beacons and route on the map so the developers can see its functionality, so once the beacon limitations are overcome, then the way the user interacts with the application will need to change so visually impaired people can interact with it. For example, providing different audio or haptic feedback depending on when the user is on or off target to their destination.

**1.3. Structure of the Document**

This document is structured by each section outlined in the table of contents. Within each section the relevant content is placed along with status updates on certain sections. Each section may have multiple sub sections so that the content can be more clearly organized and detailed. The updates mainly occur in sections such as the project management plan, design, architecture, and testing where the actual implementation probably differed from what was originally proposed in the original reports on these topics.

**1.4. Terms, Acronyms, and Abbreviations**

RSSI smoothing: This refers to an algorithm that attempts to correct the RSSI values that are read in from the beacons. This is needed to account for the signals being bounced and therefore the true values being altered before reaching the device. RSSI values correspond to how near or far away the transmitting device is (the beacon) from the receiving device (the phone).

Trilateration: This is a concept that comes from geometry where the general idea is to calculate a point based on the known positions of other points. An algorithm that does trilateration to estimate the user’s position is used in the code, which takes in the smoothed RSSI values in order to perform the trilateration.

**2. Project Management Plan**

**2.1. Project Organization**

The project has been organized so that the required work is split up according to the amount of work needed in addition to assigning it to team members who have the skills and desire to work on that part of the project. The goal of this rationale is to make sure more people are assigned to work that is more intensive, and the team members are happy with their role. This project will have multiple people occupying multiple roles because there are many tasks that need to be accomplished as well as the fact that having multiple people checking each other’s work is good to have to ensure a high quality project.

Team Members:

Tahir Aziz

Casey Boyle

Sabur Khan

Adeel Khan

Alex Guzman

Team Roles:

Project Manager: Tahir Aziz

Requirements Engineer: Adeel Khan

Designer: Tahir Aziz

Lead developers: Casey Boyle, Alex Guzman

Developers: Sabur Khan, Tahir Aziz, Adeel Khan

Testing: Adeel Khan

Quality Assurance: Casey Boyle, Alex Guzman, Sabur Khan, Tahir Aziz, Adeel Khan

Rationale:

Project Manager: Tahir is the project manager because he has leadership and management experience through leading the Association for Computing Machinery at UTDallas, and everyone else on the team agreed on Tahir being the project manager.

Understanding the requirements is extremely important since without a proper knowledge of the requirements, we will not know what to build, and as a result we decided to have a requirements engineer role. This will be Adeel since he has experience documenting requirements in projects from other classes.

Designer: Since the end product is going to be used by a specific user demographic, it is important to understand what that is and design the product accordingly. Tahir will work on the interface and experience design due to his background in that field.

Lead Developers: Alex and Casey will be the lead developers on this project simply because they have the most experience in Android development. This is a good role to have because they can guide the rest of the team and explain and make technical decisions that will impact the product, while having the skillset to implement them.

Developers: Sabur, Tahir, and Adeel will be the other developers on the team. Of course, this project is a development project, so everyone will need to contribute to the development of this product.

Testing: Testing the software that is produced is necessary to ensure that there are no bugs and errors in the code. Writing unit tests, and using testing tools are some ways we could test the software. Adeel will be the Tester because of his experience in testing at an internship.

Quality Assurance: While the product is being developed, the team needs to ensure that the product is meeting the functionality and requirements specified by the stakeholders and the requirements document. Since everyone on the team is responsible for the success of the project as defined by the primary stakeholder (the industry sponsor), everyone will have the responsibility of ensuring that the entire product meets quality standards throughout the development process and after the product is complete or ready for submission.

**2.2. Lifecycle Model Used**

We are planning to stick with a regular waterfall model, we have been given the project and requirements will use the SCRUM methodology to help break down work tasks. We will have daily “stand ups” to get a good picture on where everyone is in the project daily, we will also work in small deliverables to make sure there is nothing broken as we try to move forward. Additionally, we will meet on Wednesdays and Fridays where we hope to have a product review to demonstrate progress.

**2.3. Risk Analysis**

Likelihood:

High (H): Very likely to occur

Medium (M): May or may not occur  
Low (L): Low likelihood of occurrence

Possible Project Risks:

Fixing Difficulty :  
  
High (H): Very difficult to fix

Medium (M): May or may not be be difficult to fix  
Low (L): Easy to fix

Possible Project Risks:

Risk (Likelihood, Fixing difficulty)  
1) Old code being unusable in our project and having to start over. (L, H)

2) A team member dropping the class due to X reasons. (L, L)

3) Struggling to figure out proper algorithms needed for project (H, M)

4) Poor team communication, or lack of motivation from team (M, M)

5) Hardware issues. (missing or broken parts) (M, H)

6) Requirements for project change. (L, H)

7) Unable to meet deadlines. (M,L)

8) Hallway is not suitable for our project (H,H)

Risk Reduction Strategies:

1) For the first issue we will try our best to work around the issue and if it does occur we will try our best to scrape whatever code is usable for our project

2) Our team currently has 5 people which leaves a good amount of breathing room, if someone were to drop the class, or become unable to show up due to whatever reason, we should have enough manpower to fill in their place.

3) This is going to be a pretty common problem being unable to figure out exactly what our algorithms are and what they need to do. We will need to work as a team in order to figure out any issues and if needed we can turn to our mentor, or the PHD candidate who is working on this project at a higher level.

4) This happens a lot for many projects some people just become busy with life, other classes pile up and things just happen. Over here this is where the standups come into play, and this is why we are mandating them so people continuously working on the project and we are left accountable for the work we need to get done.

5) There is a chance that something might not be working, for example a beacon might not work, or a phone might be dysfunctional in that case Dr. Tom has offered to get things we need, but hopefully we will be able to work with whatever we need without having to buy any more.

6) This is unlikely but we are trying to work in smaller increments so that if something did end up changing hopefully it won't be too difficult to fix

7) This goes back to good team management, and good communication, we need to work together to make sure things are done on time, and if possible get done early. Also we need good team communication in terms of making sure that if someone is unable to get something done on time, that someone covers for the individual.

8) This seems to be a problem in our project we seem to be doing our project in a hallway where the connection is sometimes faulty and that may be altering our results. We have not fully tried things out yet, but it may be a necessity for us to switch which room/hallway we use to test our beacon/app functionality.

Risk Update

1) Old code being unusable in our project and having to start over. (L, H)

This did not occur.

2) A team member dropping the class due to X reasons. (L, L)

This did not occur, as we had mentioned earlier this was low risk and we did not expect it to occur.

3) Struggling to figure out proper algorithms needed for project (H, M)  
  
This occurred as we tried to figure out algorithms in regards to trilateration, also at other points such as figuring out algorithms for how fast humans walk, and how many advertisements would be necessary for our program to keep up with a moving person. We dealt with this using outside research and reaching out to people who might have had the answer we were looking for.

4) Poor team communication, or lack of motivation from team (M, M)

This occurred to a certain extent, but we kept each other in line and got our tasks done on time. As expected our own lives got in the way and we were all forced to miss meetings, but we stayed on task and got to a good end point.

5) Hardware issues. (missing or broken parts) (M, H)

This was the biggest issue that we faced. Initially we had no idea how much of an impact that the hardware was going to have on our project. The issue of our beacons not receiving, and the issue of the advertisement rates being much lower than the required amount made our project impossible to complete to the extent it is hoping to be used for. We were not able to get new beacons, but did research on other beacon options for when another team will continue this project.

6) Requirements for project change. (L, H)  
  
The requirements for our project did change, we went from moving forward and having a working application that can be used as an indoor position locator, to solidifying the fact that the beacons we were trying to use are not built to be used for the scope of this project.

7) Unable to meet deadlines. (M,L)

This did not occur, we were able to meet our deadlines.

8) Hallway is not suitable for our project (H,H)  
  
This was an expected incident and it occurred. We tried combating the hallway and all the bouncing by using a trilateration algorithm but soon figured out that the hardware was not going to be suitable regardless of the hallway we were in.

**2.4. Hardware and Software Resource Requirements (Do not forget to describe what new software or hardware each team member learned during the project)**

Software:   
Android Studio: every individual member should have android studio downloaded on their laptop/desktop

Hardware:

An android phone for at least the lead developers to have to test the application

Beacons: provided by the customer for navigation use

**2.5. Deliverables and Schedule**

Task List:

1. Team communication and work tools set up.
   1. Attain everyone’s Github usernames for version control
      1. Ensure everyone has accepted their Github invite
   2. Send everyone Trello invites for task management
      1. Ensure everyone has accepted their Trello invite
   3. Send everyone invite invites for Slack invite for communication
2. Understand the existing code bases
   1. Understand the Aruba Beacon Codebase
      1. Share knowledge with the rest of the team
   2. Understand the Walking Navigation Codebase
      1. Share knowledge with the rest of the team
3. Set up local development environment
4. Get both codebases to run on personal device
5. Combine both codebases into 1 new one
6. Understand how the beacons work
   1. Understand how to interface with the codebase created in 5.
   2. Understand how to register the beacons
   3. Understand how the beacons will be used to route paths
7. Code the functionality of the beacons interfacing with 6.1
   1. Register beacons according to 6.2
8. Understand the navigation functionality
   1. Figure out how to store the internal floor layout
   2. Understand how to use RSSI values for beacon precision
   3. Figure out how to display current location of the user
   4. Figure out how to display beacon placement
   5. Figure out how to display step count
   6. Figure out how to give audio feedback to the user
   7. Figure out how to allow the user to store paths
   8. Figure out how to display paths
9. Develop the navigation functionality
   1. Implement 8.1-8.8
10. Design the user interface
    1. Understand the end user demographic and their needs
    2. Design the user interface according to interface guidelines
11. Develop the user interface
    1. Implement 10.2 with the functionality in 7 and 9
12. Quality Assurance
    1. Ensure the product meets all requirements set forth by the customer
13. Participate in daily updates and weekly meetings
14. Participate in useage of team tools: Slack, Github, Trello, Google Drive
15. Complete required documents
    1. Project Management Plan
    2. Requirements Documentation
    3. Architecture Documentation
    4. Testing Plan
    5. Final Project Report
16. Research alternative beacons
17. Logging code
    1. Create code that will log data based on future tests
18. Running Manual tests
    1. Test Beacons at different heights
    2. Test Beacons at different walking speeds
    3. Test Beacons with different clustering
19. Analyzing Manual tests
    1. Creating spreadsheets with data from tests

Task Details:

1. Task 1
   1. Assignees
      1. Tahir
   2. Time
      1. 2 days
   3. Dependencies
      1. Team members accepting the invites
   4. Rationale
      1. This task is needed to ensure that the entire team is using proper tools for development and communication throughout the project
2. Task 2
   1. 2.1
      1. Assignees: Casey
      2. Time: 3 days
      3. Dependencies: None
      4. Rationale: Casey wanted to take a look at this codebase
   2. 2.2
      1. Assignees: Tahir, Sabur, Adeel, Alex
      2. Time: 4 days
      3. Dependencies: Each of the assignees learning the codebase so that the knowledge can be shared
      4. Rationale: The rest of the team took the responsibility of the this task because this is the more complicated codebase.
3. Task 3
   1. Assignees: Tahir, Adeel, Sabur, Alex, Casey
   2. Time: 1 day
   3. Dependencies: Possible issues affecting installation of Android Studio
   4. Rationale: Everyone needs to have Android Studio on their computer so they can code and work on the project.
4. Task 4
   1. Assignees: Tahir, Adeel, Sabur, Alex, Casey
   2. Time: 1 day
   3. Dependencies: Possible bugs in the existing codebases, Android Studio configuration issues
   4. Rationale: Everyone needs to have the codebases running on their devices to make sure the code works and they can see the result of it.
5. Task 5
   1. Assignees: Lead Developers (Casey, Alex)
   2. Time: 1.5 weeks
   3. Dependencies: Understanding the existing codebases, potential errors in the code,
   4. Rationale: Since merging the codebases is going to be tricky, the lead developers will handle this task
6. Task 6
   1. Assignees: Tahir, Adeel, Sabur, Alex, Casey
   2. Time: 1 week
   3. Dependencies: Beacons not working
   4. Rationale: Everyone needs to know how the beacons work because that is the main part of the project
7. Task 7
   1. Assignees: Tahir, Adeel, Sabur, Alex, Casey
   2. Time: 2 weeks
   3. Dependencies: Beacons not working
   4. Rationale: This is important because this is how the team will make the navigation work. Everyone will have a hand in coding this part so that everyone has good knowledge of how to integrate this with navigation
8. Task 8
   1. Assignees: Tahir, Adeel, Sabur, Alex, Casey
   2. Time: 2 weeks
   3. Dependencies: Beacons not working, issues getting floor plans into the app
   4. Rationale: The navigation is another major part of the project, so everyone should be involved in understanding what needs to be done.
9. Task 9
   1. Assignees: Tahir, Adeel, Sabur, Alex, Casey
   2. Time: 3 weeks
   3. Dependencies: Beacons not working, issues with floor plans in the app
   4. Rationale: This is important because this is the navigation part which is very important to the main goal of the project. Everyone will have a hand in coding this part because it will be time consuming and at the same time everyone will need to know how this functionality works.
10. Task 10
    1. Assignee: Tahir
    2. Time: 1 week
    3. Dependencies: User demographic information, interface constraints from the customer
    4. Rationale: The product must have an interface, and so it must be designed. Tahir will be designated it since he is the designer on the team.
11. Task 11
    1. Assignees: Sabur, Adeel, Alex, Casey, Tahir
    2. Time: 1 week
    3. Dependencies: The user interface design (10).
    4. Rationale: The interface needs to be connected to the functionality and logic of the beacons and navigation. Everyone will work on since this will be closer to the end of the project schedule and tasks will need to be wrapped up quickly.
12. Tasks 12-14
    1. Assignees: Tahir, Alex, Casey, Sabur, Adeel
    2. Time: Consistently throughout the project
    3. Dependencies: Project requirements, and team member participation
13. Task 15
    1. Assignees: Tahir, Adeel
    2. Time: In respect to due dates found on the professor’s website
    3. Dependencies: None
14. Task 16
    1. Assignees: Tahir, Adeel
    2. Time: 1 Week
    3. Dependencies: None
15. Task 17
    1. Assignees: Casey, Sabur
    2. Time: 1-2 weeks
    3. Dependencies: None
16. Task 18
    1. Assignees: Tahir, Adeel, Casey, Sabur, Alex
    2. Time: 1 week
    3. Dependencies: Task 17
17. Task 19
    1. Assignees: Tahir, Adeel, Casey, Sabur, Alex
    2. Time: 1 week
    3. Dependencies: Task 18

Schedule:

1/21- 1/24: Tasks 1, 3

1/25-1/30: Tasks 2, 4

1/31-2/10: Task 5

2/11 - 2/17: Task 6

2/18-3/1: Task 7

3/2-3/15: Task 8

3/16- 3/29: Task 9

3/30 - 4/13: Task 10

4/14-4/20: Task 11

Tasks 12-14 will be continuously done throughout the course of the project

Tasks Update

We were able to get through tasks 1-8.2, and 12-14 at task 8.3 we realized that we would be unable to move forward because the advertisement rates were going to be too low

Schedule Update:

1/21- 1/24: Tasks 1, 3

1/25-1/30: Tasks 2, 4

1/31-2/10: Task 5

2/11 - 2/17: Task 6

2/18-3/1: Task 7

3/2-3/15: Task 8

3/16- 3/29: Task 15,16,17

3/30 - 4/13: Task 17, 18

4/14-4/20: Task 19

Tasks 12-15 will be continuously done throughout the course of the project

**2.6. Monitoring, Reporting, and Controlling Mechanisms**

1. Project Management Plan
   1. An outline of the project plan
   2. Due on 1/27
2. Requirements Documentation
   1. An in-depth documentation of the requirements the project will have to meet
   2. Due on 2/10
3. Architecture Documentation
   1. A document describing the architecture of the software the team is building
   2. Due on 2/24
4. Detailed Design Documentation
   1. A detailed document going in-depth on the design of the software, modules, interactions, components, etc.
   2. Due on 3/17
5. Testing Plan
   1. A plan that outlines what the team will be doing for testing the project’s code
   2. Due on 4/7
6. Final Project Report
   1. A detailed report on the entire project & presentation
   2. Due on 4/28

Reporting Mechanisms: The reports will be written using Google Docs and will be submitted via email to the professor.

Monitoring and Controlling Mechanisms: Each phase of the project roughly corresponds to a management report. Thus, the content and information needed for each report will be gathered as that part of the project is being implemented. As a result, the report content will be monitored by the team members who will already understand what the project details relevant to the upcoming report. The project manager will control the reports and ensure that the reports are satisfactory and that they are turned in on time.

**2.7. Professional Standards**

The team is expected to behave in a professional manner. This project is an introduction to how our life will be in the real world and we are going be acting with that in mind.   
  
We have shared our personal numbers, emails, and have created a slack for our project, we have explicitly arranged meeting times around everyone's schedule so that everyone can attend. We have a policy where a meeting needs at least three people to begin and we will stick to that. As a team we are holding ourselves accountable and are hoping that any problem can be solved internally as a team. We will also keep in mind the management references in regards to the IEEE standards.

We have our “customer” which is ultimately Dr. Tom we understand that he has requirements that need to be met and a certain standard those requirements need to be at as well. We will work together to make a code that not only meets the customer's requirements but also meets his standards as well

With that being said, there is also a knowledge that the end does not justify the meaning that we will not resort to scholastic dishonesty in any way. We will work hard at keeping ourselves and each other in check and make sure that we know exactly where the code is coming from, and if needed we will cite any of the sources we use to further our project.

**2.8. Evidence all the artifacts have been placed under configuration management**

We are using Google docs to make sure all our documents are being typed out in a way that everyone can contribute. At the same time we are using Trello to keep track of activities and Slack for communication. All our google documents and their links are sent on to our slack and can be sent to the professor very easily.

**2.9. Impact of the project on individuals and organizations (Include a description of what impact your project will have on individuals and society)**

The reason our group chose this project was because of how excited we were for the end goal for this long term project. The end goal of this project is to assist the visually impaired down hallways in real time without the use of a cane to an extent, allowing them to walk more freely without worrying about hitting walls. On an individual level we will be helping the visually impaired who are apart of our society and we can not just leave them alone. In accordance to our own morals we knew that no matter how difficult this project is, if it had a positive effect on even one person’s life then we would have done our job. At a societal level this project is kind of an obligation that needed to be fulfilled. Although our group was not able to continue this project due to the hardware issues we came across, the idea is still the same we need to help those people who have a disability. Hopefully the next group can use a better beacon that will allow them to take this project further.

**3. Requirement Specifications**

**3.1. Stakeholders for the system**

* Dr. Tom Hill - industry sponsor
* The team
* Professor Wong & TAs
* Visually impaired people (end goal customer)
* Kirthy - PhD student who is taking this project further

**3.2. Use case models**

1. Navigation: Refer to figure 1
   1. Actors:
      1. Users
   2. Entry Condition(s)
      1. User Opens Application
   3. Normal Flow of Events
      1. User Opens App
      2. User selects start point
      3. User selects destination
      4. User follows directions
      5. User reaches destination
   4. Exit Condition(s)
      1. User reaches destination
   5. Exceptions (Alternate Flow of Events)
      1. User quits app mid-navigation
      2. User changes mind and re-routes
      3. User gets lost
   6. Special Requirements
      1. None
   7. Rationale:
      1. Navigation is a key function of this product so it should be a use case
2. Navigation Feedback: Refer to Figure 2
   1. Actors:
      1. Users
   2. Entry Condition(s)
      1. User enters navigation
   3. Normal Flow of Events
      1. User navigates
      2. While user is on path, constant haptic/audio feedback (beeping) is given
   4. Exit Condition(s):
      1. User completes navigation
   5. Exceptions:
      1. If user deviates, more alerted beeping is done
   6. Special requirements:
      1. None
   7. Rationale:
      1. Since we want to display feedback to users who are blind, giving auditory or haptic feedback based on their navigation must also be a use case
3. Image Processing: Refer to Figure 3
   1. Actors:
      1. Users
   2. Entry Condition(s):
      1. Map has not been loaded into the application or has not been selected
   3. Normal Flow of Events
      1. User Opens App
      2. User adds map to the app
         1. User loads in an existing map
      3. Map is displayed on screen
   4. Exit Conditions
      1. User completes adding of map to app
   5. Exceptions:
      1. User takes a picture of a map to load it in
   6. Special Requirements:
      1. None
   7. Rationale: The primary way to see a map through which one can navigate is through loading one in. Thus, having a use case for that is necessary.
4. Location Display: Refer to Figure 4
   1. Actors:
      1. Users
   2. Entry Conditions(s):
      1. User hasn’t begun navigating yet
   3. Normal Flow of Events
      1. User opens app
      2. Device asks for user location
      3. User grants device location access
   4. Exit Conditions(s):
      1. User gives device location access
   5. Exceptions:
      1. Quit app or display error if location access is not granted
   6. Special Requirements:
      1. None
   7. Rationale:
      1. Knowing the user location needs to be a use case because that will help with trilateration with the beacons, which is the key step in order to navigate with the beacons.
5. GUI: Refer to Figure 5
   1. Actors:
      1. Users
   2. Entry Condition(s):
      1. No beacons have been placed on the map
   3. Normal Flow of Events:
      1. User detects no beacons are connected on the app
      2. User clicks add button to see available beacons
      3. User selects beacons to add
      4. Map displays beacons
   4. Exit Conditions:
      1. User completes selection of beacons to add to map
   5. Exceptions
      1. User deletes beacons off the map
   6. Special Requirements:
      1. None
   7. Rationale: Since the app requires an initial setup to have beacons loaded on the map in order to navigate, having that step as a use case is important.

**3.2.1. Graphic use case models**

Figure 1: Navigation

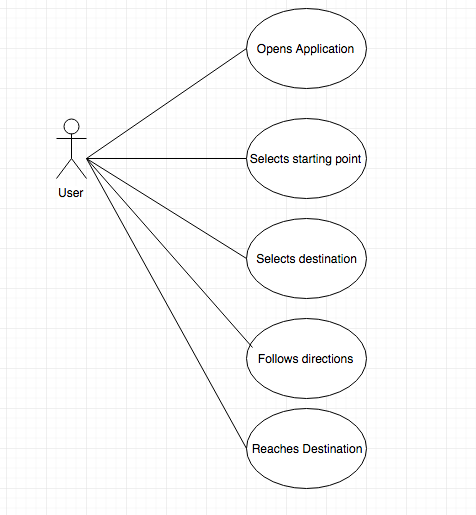


Figure 2: Navigation Feedback

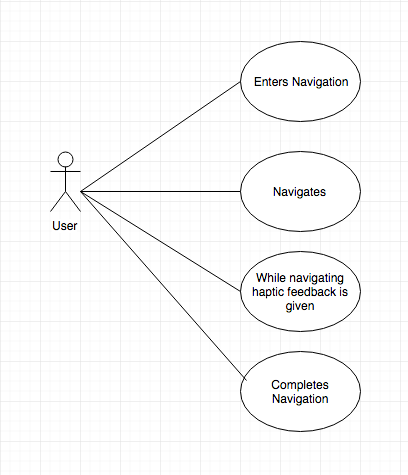


Figure 3: Map

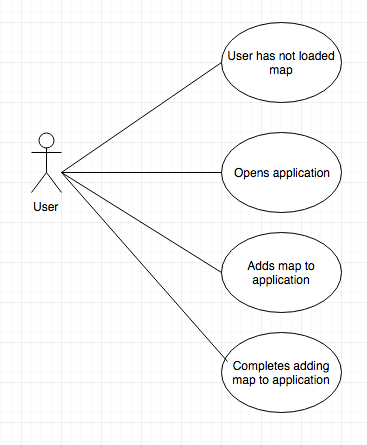


Figure 4:

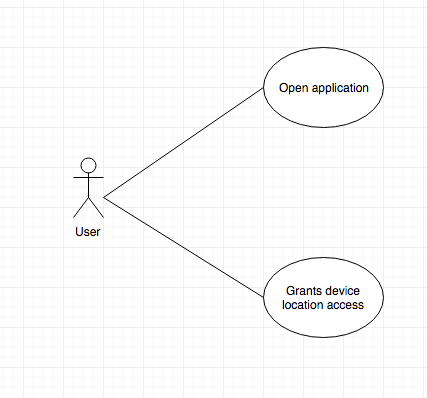
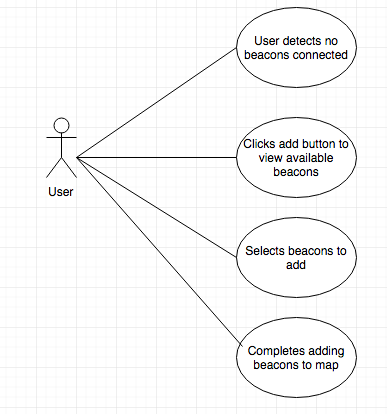


Figure 5:



**3.3. Rationale for use case models**

The rationale for each use case is outlined in each use case model in Section 3.2

**3.4. Non-functional requirements**

1. Usability
   1. The application shall be usable for people who are blind
   2. The application shall start navigation in 2 steps - selecting origin and destination
   3. The application shall be simple to use
   4. Initial configuration of beacons shall be simply adding broadcasting beacons to the GUI
   5. The user should not have to reconfigure beacon location after initial setup
   6. The user interface should follow proper UI design guidelines
2. Performance
   1. The application should react to a user getting a TBD distance away from a wall and react according to Functional Requirement 2.2
   2. The beacons should be on at all times in case a user is using the application
   3. The application should not take longer than 3 seconds to load navigation between 2 points
3. Modifiability
   1. The application should be able to evolve from being a visual application to one that can actually be used by people with visual impairments
4. Reliability
   1. The application should not crash when choosing new maps
   2. The application shall not crash in the middle of navigation
   3. The application shall not have delayed haptic feedback
   4. The application should not provide haptic feedback at the wrong times
   5. The application should be able to function regardless of what kind of building they're in.

**4. Architecture**

**4.1. Architectural style(s) used**

We chose to consider the MVC framework (Model, View, Controller) for our project. The application is modularized into three main components: model, view, and controller. The model is where the data is stored, the view is where the user sees and provides inputs to the interface, and the controller handles the logic coming from user inputs in the view to any changes in the model that occur as a result of those inputs. The main feature of our application will be navigating a user along a path and checking if they are on track by trilaterating their position via bluetooth beacons. Thus, the main functionality involves some sort of logic updating the view and the data when new signals are read in. As such, MVC supports the main functionalities of our application. Other features involved providing feedback based on the user being on or off the path, which also falls in with the MVC framework since the view can be updated with the feedback based on the user input (the trilaterated user location in this case). The main category of functionality involves displaying information to the user based on what the user is doing, so the MVC makes sense because we can have the view, a controller to handle the input, and the model for the data.

**4.2. Architectural model**

System: Android application navigating a person between 2 points in a building where the navigation is checked with bluetooth beacons to locate the user.

Subsystems:

* Beacons
* Navigation
* User View
* Database
* Controller

**4.3. Technology, software, and hardware used**

Technology: Beacon navigation, wayfinding

Software: Android Studio, Git/Github, Java, SQLite

Hardware: Aruba Bluetooth Beacons

Communication between application server and database server: Since this is an Android application, the app will be run on the user’s device via the OS, and the database will be stored within the application. Since we are doing the MVC pattern, the controller/logic will handle interactions with the database, and those updates will be reflected in the application view.

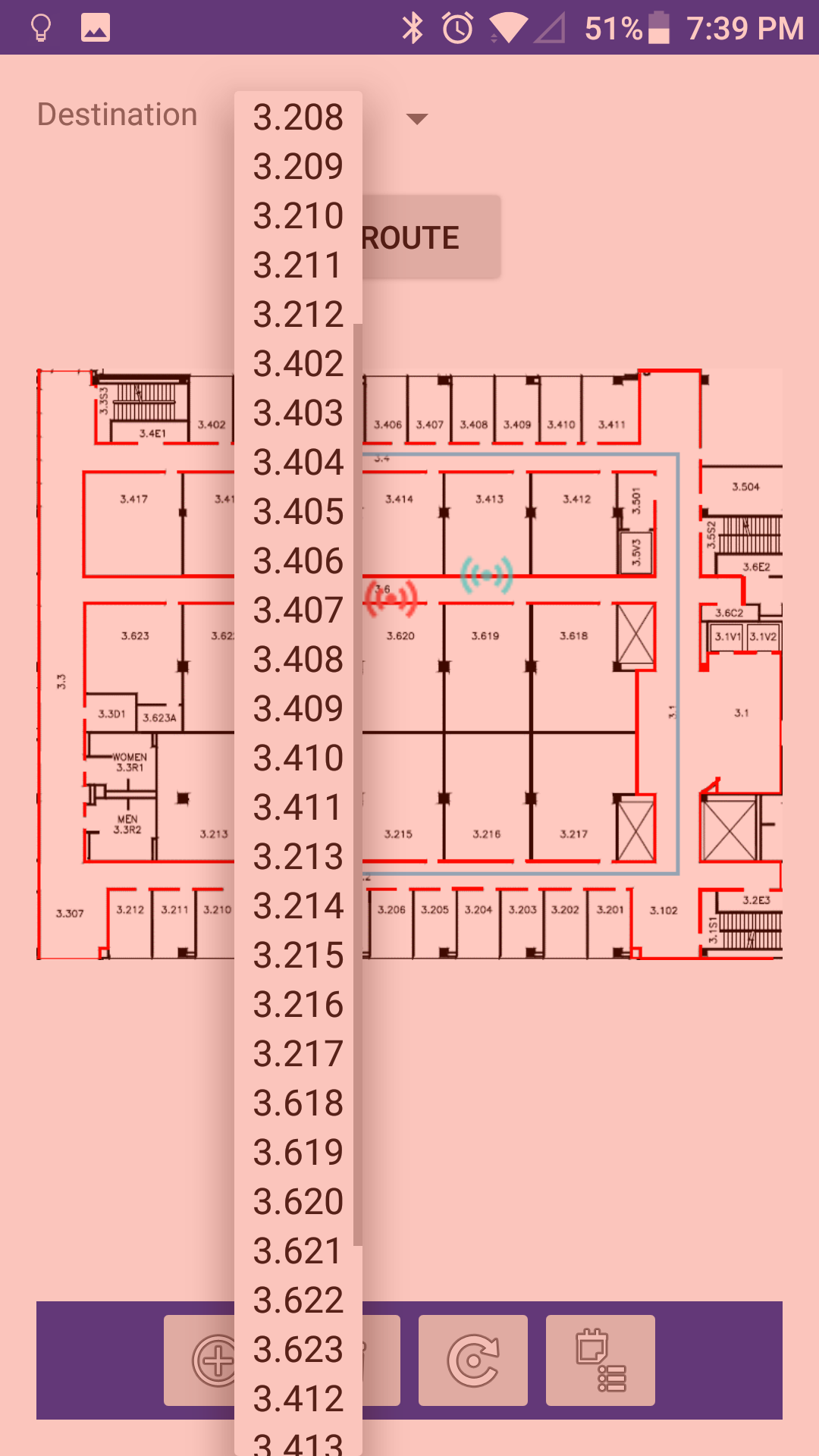
**4.4. Rationale for your architectural style and model**

There are many different types of architectures, we had the opportunity to look through the different aspects of our application and choose an appropriate architecture, we first looked at client/server architectures and understood our app doesn't truly require a lot of security as there won't be a login system, or too much sensitive data being shared, so we crossed that one out, then we then looked at component based architectural styles this model is better for games that will constantly need reusable features, and are not context specific. Our application will not need to reuse a lot of features as each map will have a different path for example. So in the end we ended up choosing kind of a mixed model using the MVC model gives us the range that we need to show how the data is stored, a view to help indicate how the software will be viewed and the controller which will be the main connection of how the database interacts with the user view.

**5. Design**

**5.1. GUI (Graphical User Interface) design**



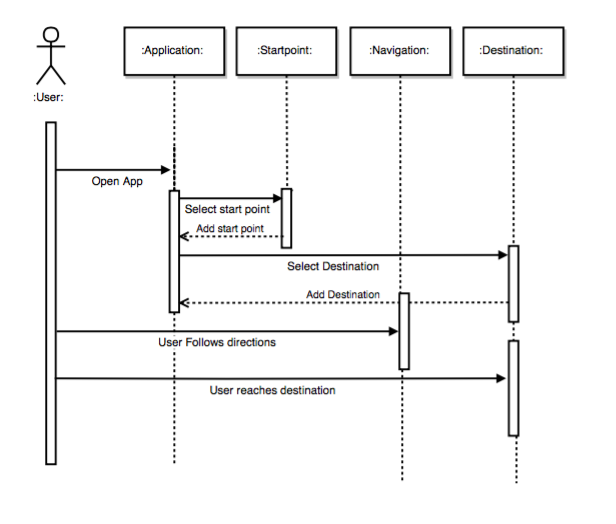


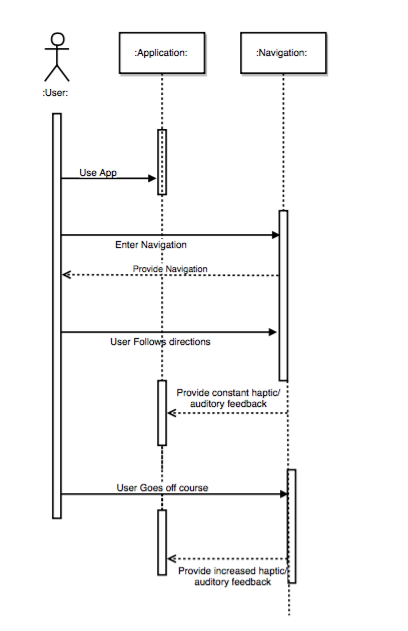
One thing to note about the GUI design is that this an intermediary interface used by the developers (us) to be able to visualize the route, where the beacons are, where the trilateration places us, etc. Because of the aforementioned bottlenecks with regards to the positioning and updating user location, the interface caters towards those building the application for testing purposes. In the future, when these problems are resolved, the interface will need to be redesigned so that visually impaired people can input their start and endpoints and receive non-visual form of feedback and input for the application.

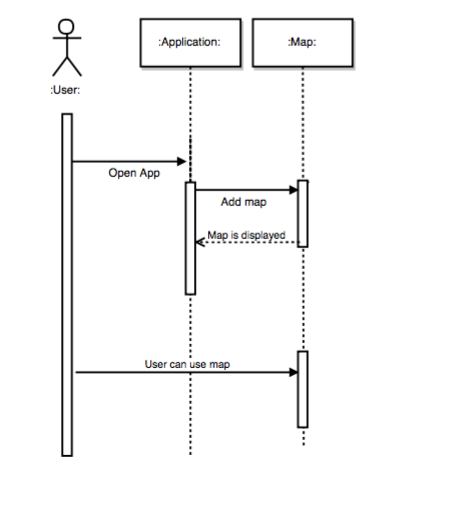
**5.2. Static model – class diagrams**

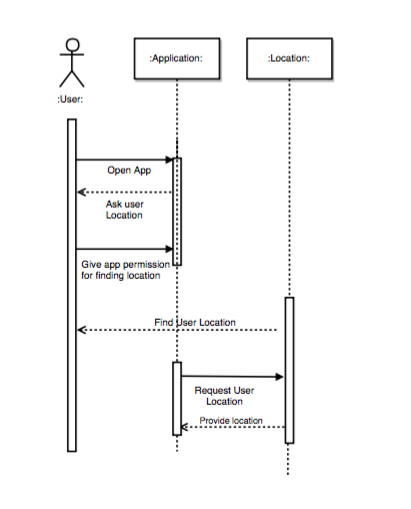
DetailedDesignUML.png

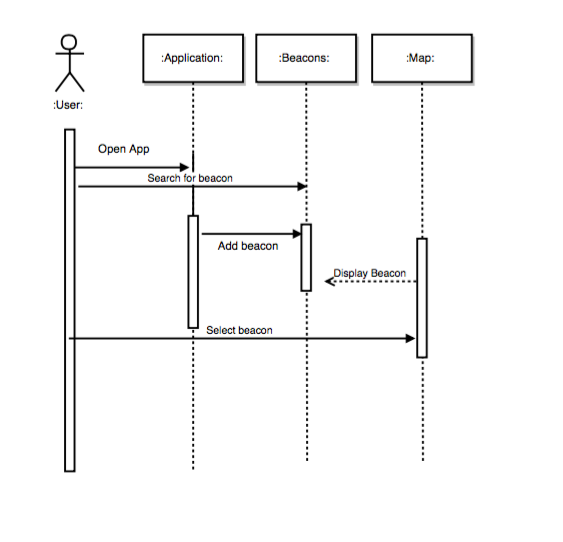
**5.3. Dynamic model – sequence diagrams**











**5.4. Rationale for your detailed design model**

The UML outlines the rooms in the image map and how those are mapped to the application for use and then how that is connected to the Main Activity of the Android application. This is important because it helps us identify rooms and places that can be navigated to. The sequence diagrams outline the interactions between the user and the application and shows the steps the application takes.

**5.5. Traceability from requirements to detailed design model**

The sequence diagrams that were drawn in this document were taken directly from the Use cases that we came up with in the requirements document. We see a lot of the features that we had first thought of as just requirements “come to life” in this document.

Specifically sequence diagram one correlates with the first use case “Navigation”, sequence diagram two correlates with the second use case “Navigation feedback”, sequence diagram three correlates with the third use case “Image Processing”, sequence diagram four correlates with the fourth use case “Location Display”,  
sequence diagram five correlates with the fifth use case “GUI”.

**6. Test Plan**

**6.1. Requirements/specifications-based system level test cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Steps | Test Data | Expected Results |
| TestUC1 | The user wants to Navigate from point A to point B. | 1. User opens App  2. User selects Start Point  3. User selects destination  4. User follows directions  5. User reaches destination | User chooses the start point.  User chooses the destination | User should reach destination based on application Map |
| TestUC2 | Navigation feedback is given to the user | 1. User navigates  2. Haptic or audio feedback is given while user is on path |  | User should feel haptic feedback |
| TestUC3 | User adds new map to the application | 1. User opens App  2. User adds map to the application | User chooses new map to add to the application | User should be able to see the new map in the application. |
| TestUC4 | User gives the application access to location | 1. User opens app  2. User grants device location access |  | The device should display the user’s current location |
| TestUC5 | User adds beacons to the application | 1.User sees no beacons connected  2.User selects a beacon to add | User selects beacon to add | The device should display added beacon. |

**6.2. Traceability of test cases to use cases**

The test cases that we are including in this document relate directly to the use cases in the requirements document. To be specific the Test Case ID’s correlate with the Use cases with them being: TestUC1 goes with use case 1, TestUC2 goes with use case 2, TestUC3 goes with use case 3, TestUC4 goes with use case 4, TestUC5 goes with use case 5

**6.3. Techniques used for test generation**

In order to generate the test cases, we plan to use JUnit for helping us to write the unit tests. It will allow us to test these basic functions and return the pass/fail values from the tests, while allowing us to configure various settings such as creating test objects to run the tests on and customizing what we want to do with the test using the various built in JUnit fixtures. We will also need to map the steps of our use cases to the appropriate test so that the tests are as realistic as possible.

Since we are doing unit testing to determine the outcomes of various steps per the use cases, we will be using black box testing. Black box testing looks at testing the input and output of the test rathe the inner workings of the code, which is what white box testing is. Since our applications is built off existing code and various native Android classes, it isn’t feasible to test the inner workings of the code. We will just look at the results of our test cases to ensure the application is functioning.

**6.4. Assessment of the goodness of your test suite (Which metrics were used for such assessment?)**

In order to measure the quality of our tests, we need to determine criteria that measures the success of those tests. Since our tests are supposed to be traceable to our use cases, the expected outcome of the use case is going to be a huge factor that determines the quality of our tests. Additionally, in order to have the inner workings of the code do what they need to, we need to provide proper inputs to the test cases. Also, as mentioned earlier, the correct mapping of the use case steps to the tests is another aspect that is important to having good quality and realistic test cases.

Summarized, the criteria for measuring the quality of the test cases are:

* Correct mapping of use case steps to the test cases
* Proper and realistic inputs for expected success, bad/fake input for expected failures
* Correct expected outcome given the input

Regarding the research tests mentioned in the previous section, here are the criteria for those:

* All outlined tests are conducted
* The independent variables are incorporated into the experiment
* The controls of the experiment are not interfered with
* Logging of data is accurate per the expected logging functionality.

**7. Research Tests**

**7.1 Rationale and Background**

Because the main goal of the project was determining the effectiveness and feasibility of the indoor navigation using the beacons, we decided to run some manual tests on the application and log data for each test so we could better understand what was going on and try to come away with insights that would help the development of the application.

**7.2 Experimental Setup**

Here is our testing approach:

What we will keep note of for each test:

* How many beacons
* Queue length,
* Beacon Placement height
* Where each beacon was placed (spacing)
* Activity performed
* Record configuration settings (the settings on the application)
* Distance between each beacon

For all tests:

We used 3 beacons

Beacons were placed at waist level (around 3 feet above the ground)

For beacons, the first beacon placed on the left side of the hallway, the second one placed on the right side, and the third one placed on the left side.

For stationary tests, we stood in the middle of the hallway next to the first beacon

Each test had the same configuration settings (except the queue length when that was changed for each test)

Variations:

1. Queue length 6, 7ft spacing, stationary
2. Queue length 12, 7ft spacing, stationary
3. Queue length 6, 7ft spacing, walking normally
4. Queue length 12, 7ft spacing, walking normally
5. Queue length 6, 14ft spacing, stationary
6. Queue length 12, 14ft spacing, stationary
7. Queue length 6, 14ft spacing, walking normally
8. Queue length 12, 14ft spacing, walking normally

Explanation and Other Details:

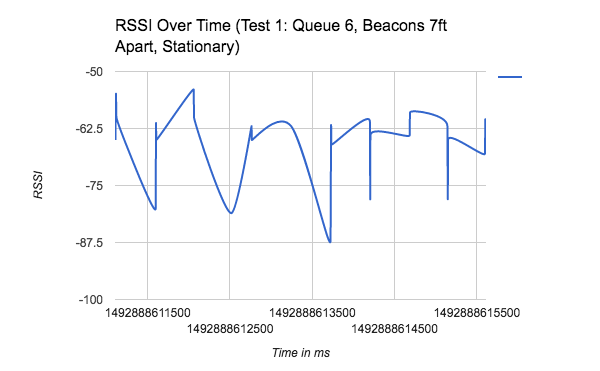
* Queue length refers to the amount of RSSI values a queue in the code receives before passing them off to the RSSI smoothing algorithm. Each beacon’s RSSI values go into a separate queue. Each set of values then is passed off to the the RSSI smoothing, which takes an average of those values (per beacon), and then the smoothed values for each beacons are passed to the trilateration algorithm which estimates position. The queue length was a variable in our test because we wanted to see the tradeoff of having more values to work with in a bigger queue, but having to wait slightly longer for a position update.
* Walking normally refers to walking back and forth between the beacons at a normal walking pace.
* Stationary tests were standing next to the first beacon in the middle of the hallway for 5 seconds.
* The number of feet apart in the test titles refers to how far away the beacons were placed. For 7ft, the beacons were 7 floor tiles away from each other (~7ft), and for 14ft, each beacon was 14 floor tiles away from each other.
* Regarding the configuration settings, these are tweakable settings that can be adjusted in real time. They affect different things and here is what they are and what they were set to:
  + Map width constant (scaling factor for the map so the calculated route fits on a phone, this has to be adjusted per device): 204
  + Map height constant (same as the width constant except for the height): 233
  + Bluetooth advance DB filter (this filters out RSSI values below the set value): -120
  + Trilateration solver max iterations: 10,000
  + Max Beacon Silence (this drops a signal if isn’t received from the beacon within the set time in ms): 1300ms
  + Minimum position update delay (can have a position update at the slowest of set time amount): 200ms
  + Maximum position update delay (can have a position update at the fastest of the set time amount): 100ms

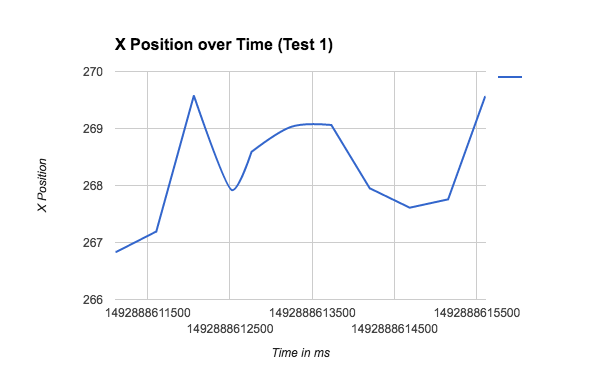
Procedure:

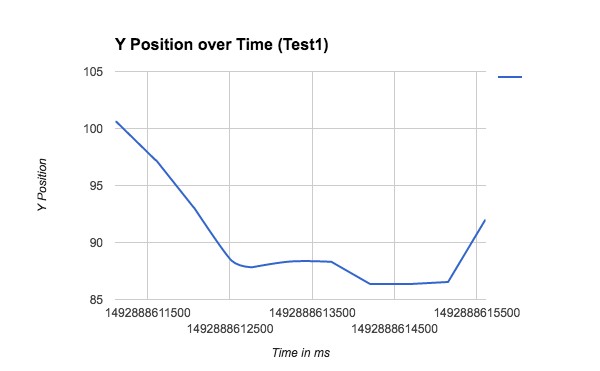
1. Open app
2. Set up beacons
3. For each test:
   1. Start the logging
   2. Run a test
   3. Stop logging
   4. Verify log file exists
   5. Ensure contents within log file reflect the number of tests performed
   6. Prepare for next test

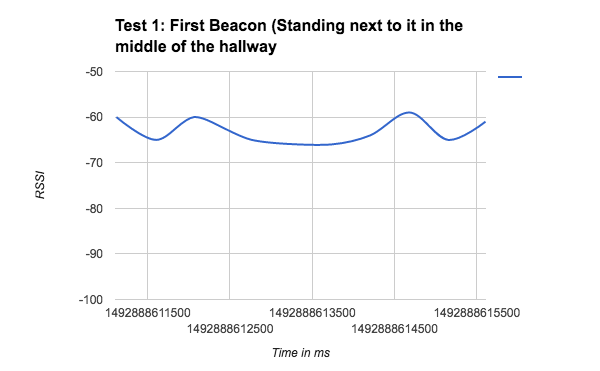
**7.3 Graphs**

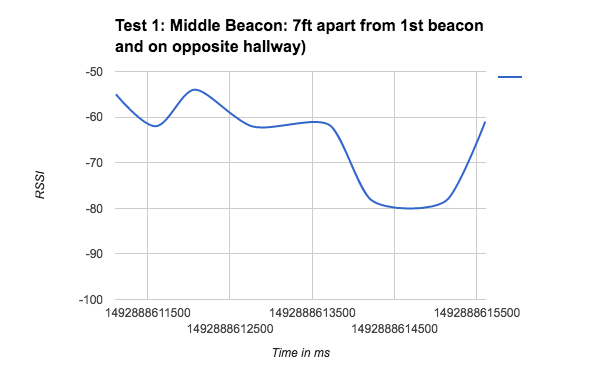
Queue length 6, Beacons 7ft spacing, Stationary

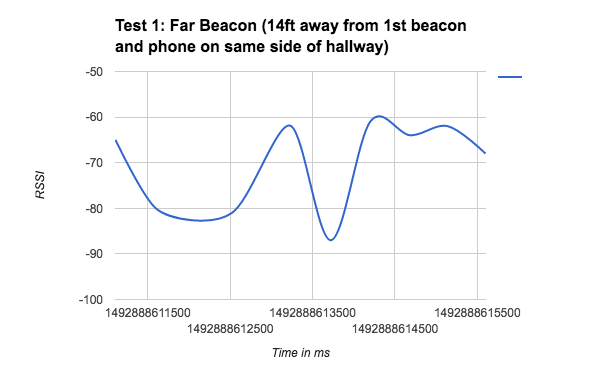




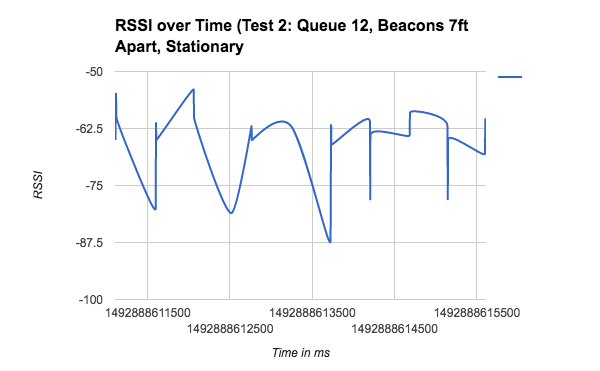


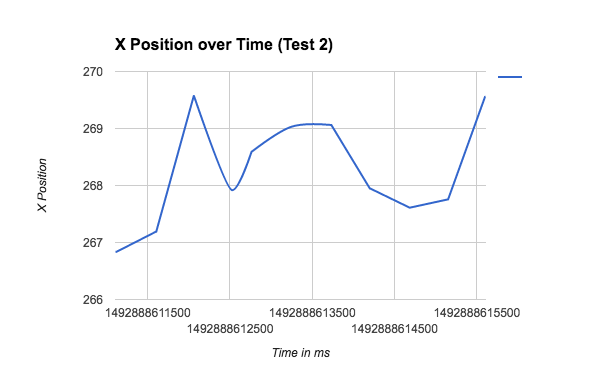


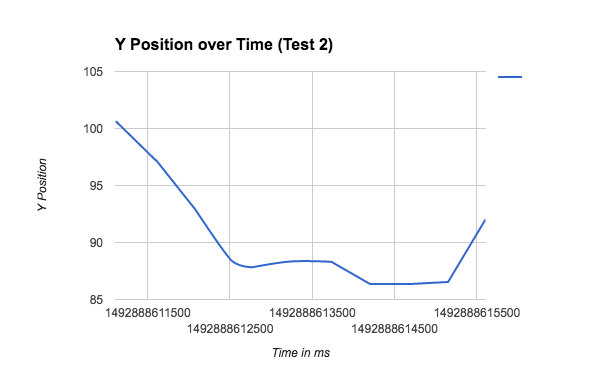


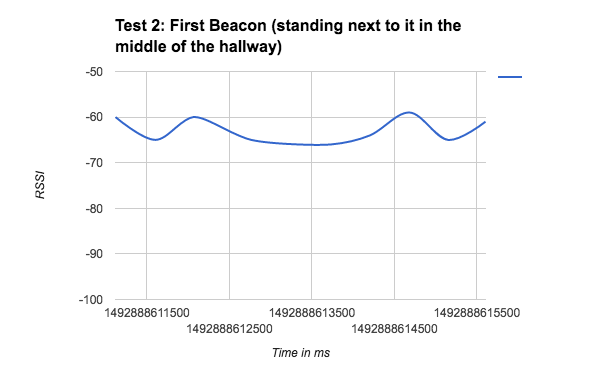


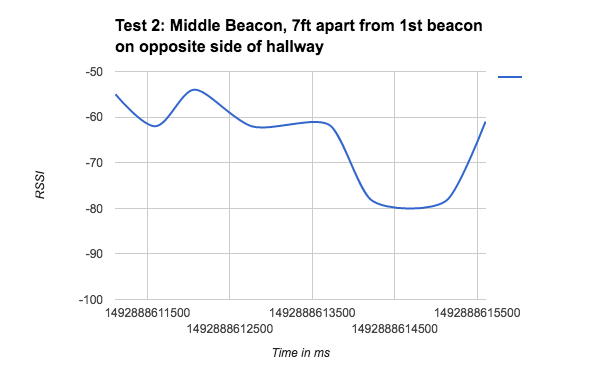
Test 2 Queue length 12, Beacons 7ft apart, Stationary

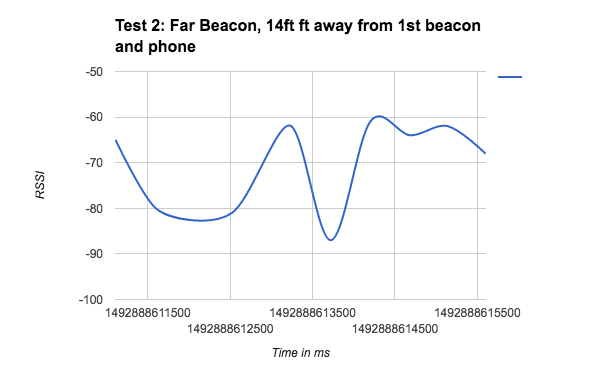




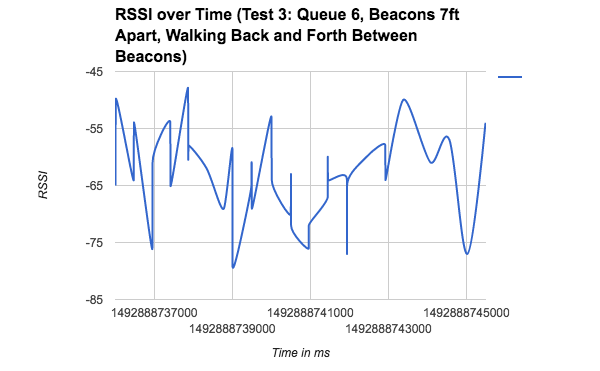


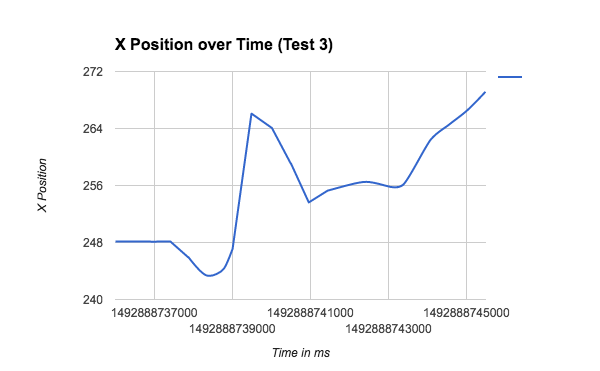


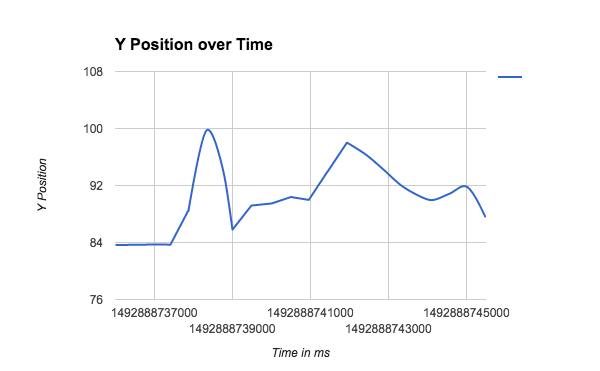




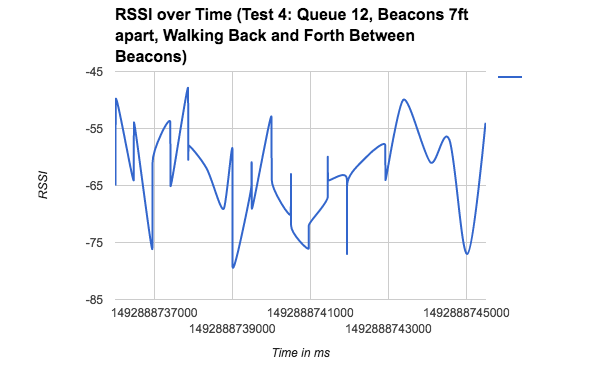
Test 3 Queue of 6, Beacons 7ft apart, Walking back and forth between beacons

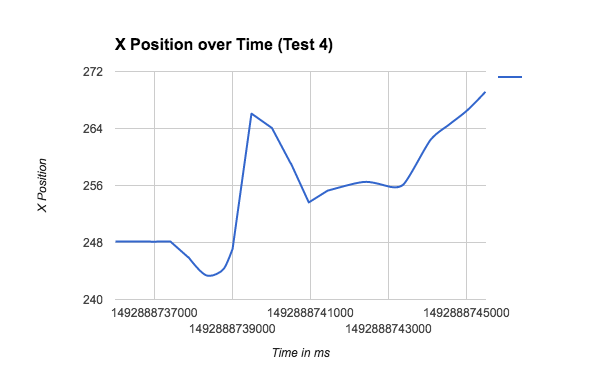


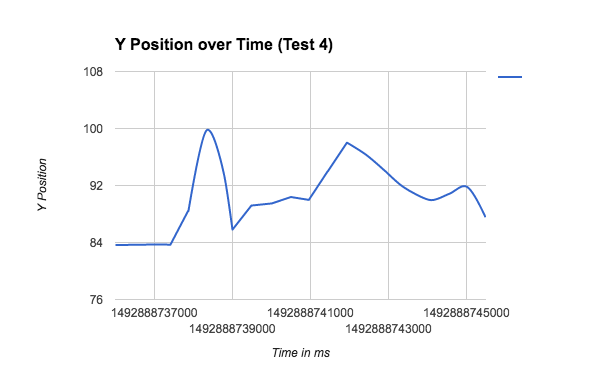




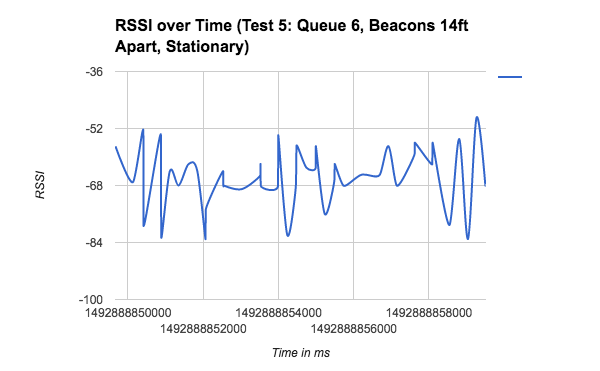
Test 4 Queue of 12, Beacons 7ft apart, Walking back and forth between beacons

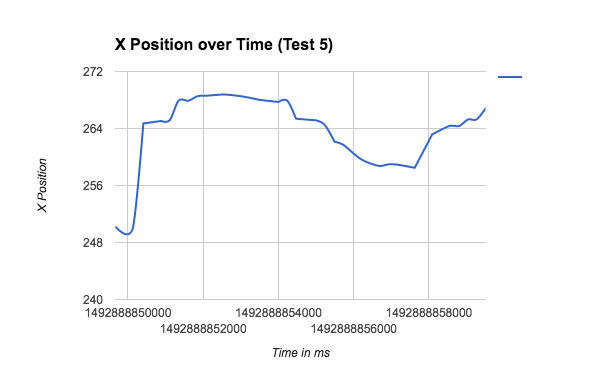


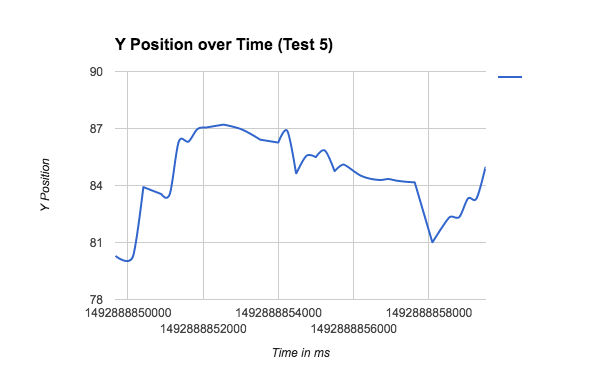


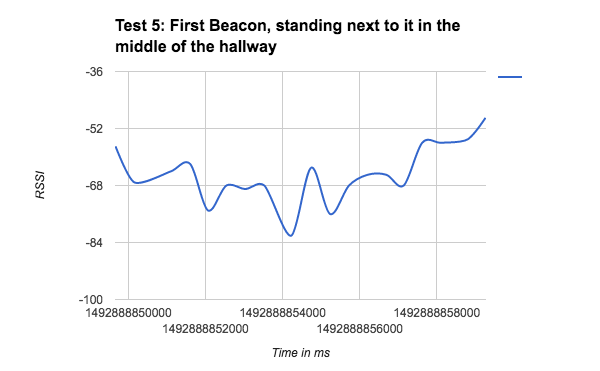


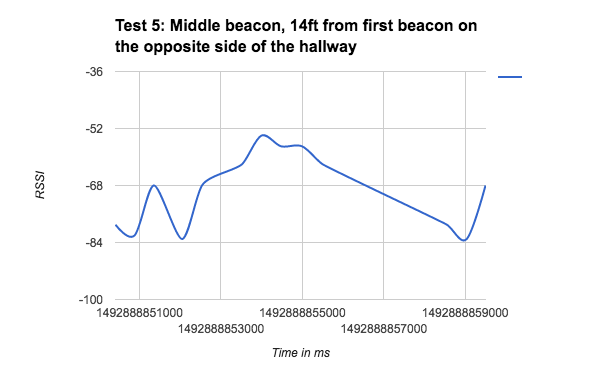
Test 5 Queue of 6, Beacons 14ft apart, stationary

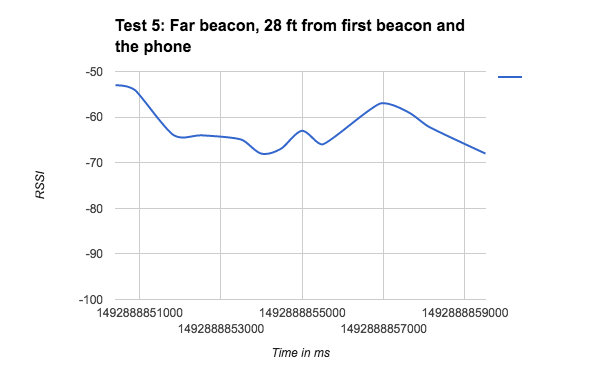




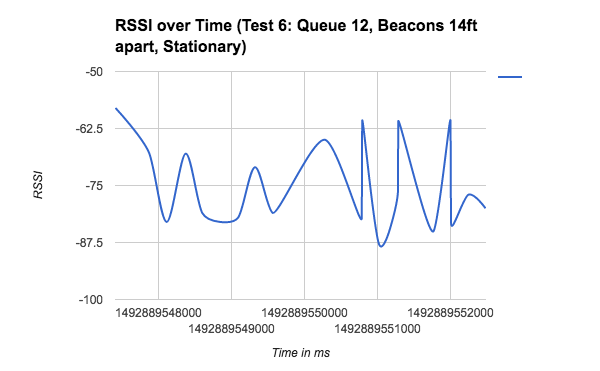


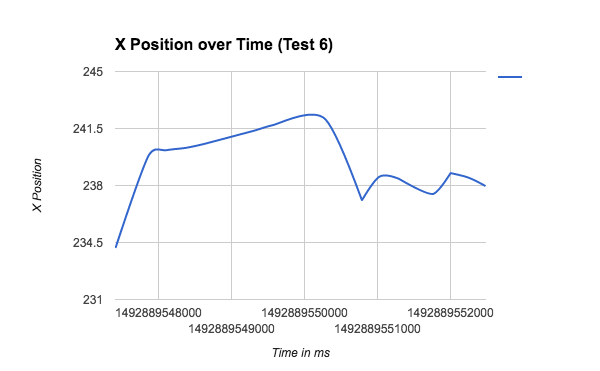


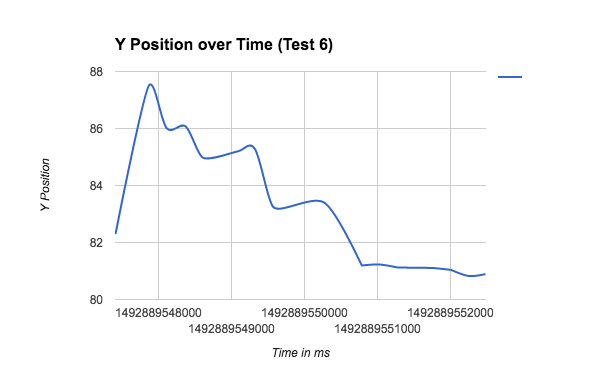


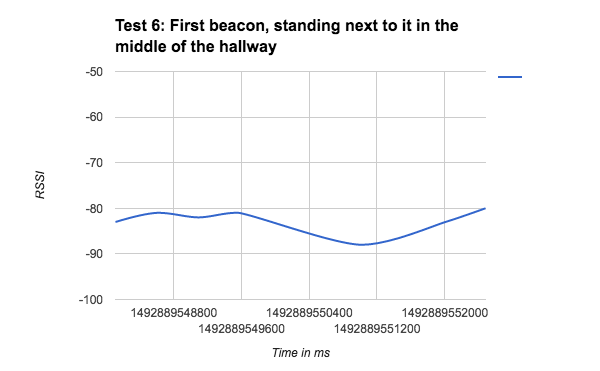


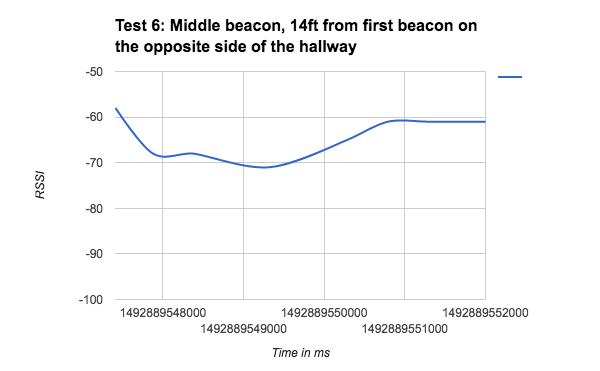
Test 6 Queue of 12, Beacons 14ft apart, Standing

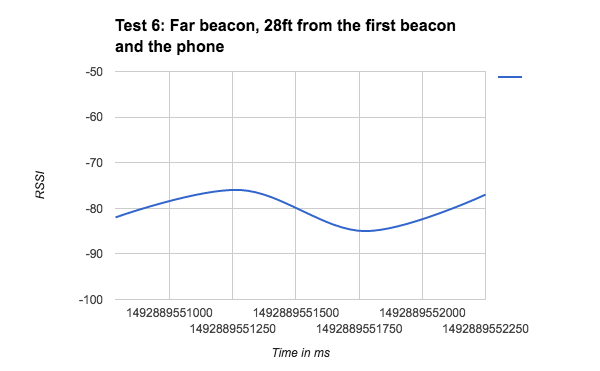




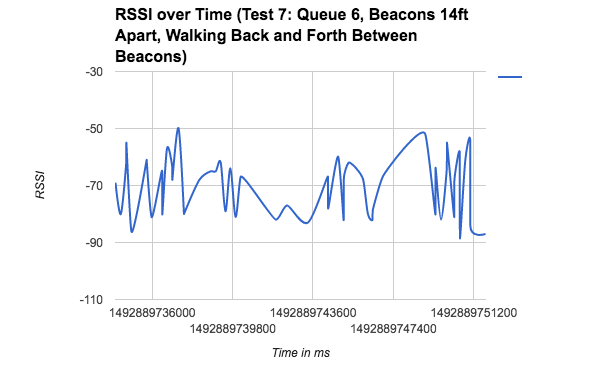


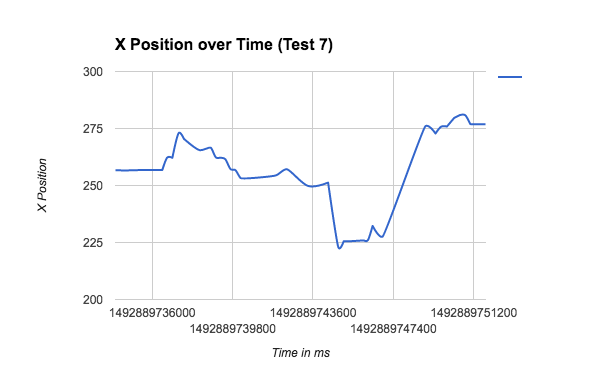


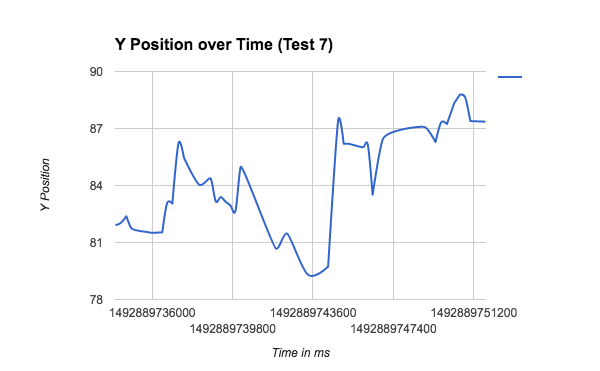




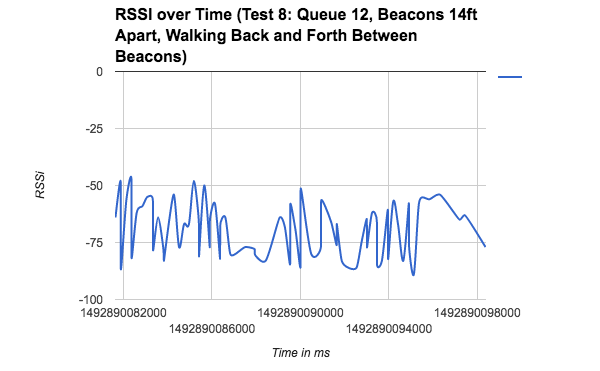
Test 7 Queue of 6, Beacons 14ft apart, Walking back and forth between beacons

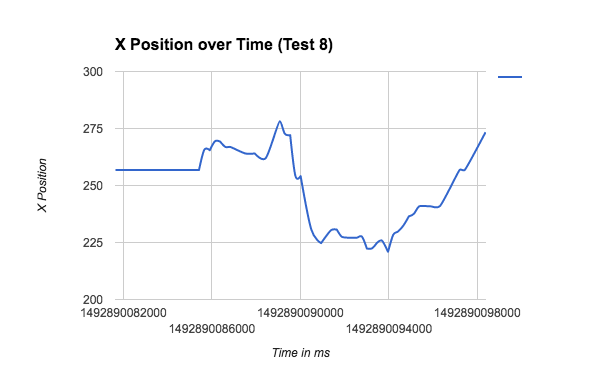


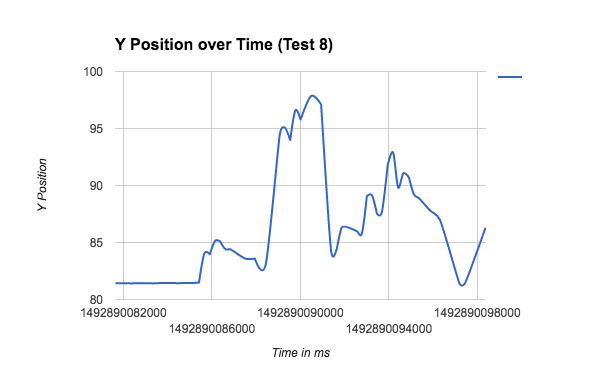




Test 8 Queue of 12, Beacons 14ft apart, Walking back and forth between beacons







**7.4 Findings From Tests**

In the graph section we have multiple tests in regards to the below:

RSSI over Time: shows how the RSSI values vary over time from all 3 beacons. On the stationary tests, this is very useful because it shows the bouncing and accuracy of the signals.

Each Beacon over Time: For stationary tests, we showed each beacon’s RSSI values over time and recorded its distance from the phone. We found that the closer the beacon to the phone, the less varying the RSSI values were.

X Position over Time: Shows how we are moving down the hall over time.

Y Position over Time: Shows how accurate the app is at placing us in the hallway itself.

(are we in the middle of the hallway, closer to one side etc..)

While looking at the logs we noticed a few things. One is that the the trilateration function keeps running even if the received RSSI values are the same. Solving this problem could help the application run more efficiently. Another thing we saw from the logs is that since the trilateration algorithm needs at least 2 beacons to estimate a position, the position updates get really wacky as soon as you start reaching the boundaries of where the beacons are set up because it has less to work with. Additionally, due to RSSI bouncing or signal loss, we did not always have the correct number of RSSI readings in the log. For example, we received a different amount of RSSI readings from each beacon is some tests.

From the graphs we can come to a safe conclusion of two things, the application works to the extent of the hardware, meaning that had the scope of this project been “ Show your approximate location in a building” this application would have been better suited. As shown in the graphs and what we witnessed from the experiments was that sooner or later it updated and we had an approximate location as to where we were (minus the bouncing, which we’ll discuss later) The scope of the project was not approximation though, the scope was to have the user’s location to the extent that it could catch them before they hit a wall. The hardware itself just was not cut out for the job of location detection and should only be used as a proximity beacon.

The second conclusion we also see is that the hallway we were in also causes a lot of bouncing of signals which ruins our location. So imagine the scope was approximation. Even in approximation and while we were standing in one place we would randomly see our location on the application bounce around and then settle back in place.

Attached above we have extensive tests showing the application work in different circumstances and from them we come to the above conclusion. These conclusions are provided to prove our original hypotheses that these beacons will not work, even if the hallway had no bouncing the hardware was found to be far to subpar for the scope. Additionally, there were bouncing and varying RSSI values even while standing stationary.

**7.5 Constraints and Other Findings**

* Typically there is a new log file created every time the app is closed. However, rotating the orientation of the device will also cause a new log file to be created.
* The trilateration algorithm runs even if the received RSSI values are not new so that slows down position update because it has to finish running and then update position before new values can be processed. This was determined by looking through the logs.

**8. Conclusions and Future Directions**

In this senior design project, we were tasked with discovering the feasibility of doing indoor navigation with proximity Bluetooth beacons. The long term goal of this project is to be able to allow people who are visually impaired to navigate inside a building using an application.

We inherited the codebases of past teams and started with understanding the general logic and workings of the code. We merged the codebase, and after that, started to work on a few tweaks such as adding adding in an image processing algorithm that detected walls based on the image map that came with the codebase, as well as adding a GUI for real time configuration of various settings. Additionally, we did some basic research on Bluetooth and beacons so we could better understand the area that we were working in. We also started working to understand the trilateration and tried to improve it as well attempt to improve the RSSI smoothing algorithm that was in place.

1. The beacons were proximity, not location based and so that means they are better used for general placement rather than high accuracy location.
2. The beacons emit just 2 RSSI values per second
3. The current implementation of the RSSI smoothing (to account for signal bouncing) took in a queue of values from each beacon, returned the average, which then passed that smoothed value to the trilateration algorithm which estimates where the user is relative to the beacons.
4. The bottleneck thus occurs in 2 areas
   1. The frequency of emission
   2. High bouncing (which leads to inaccurate RSSI values) due to the hallways + Bluetooth being at the same frequency as WIFI
5. The RSSI readings are bouncy and inaccurate even while standing still.

In the future, there are a few things that we want to outline, since this is a long term project. We believe the next steps in terms of strategy include getting location and not proximity beacons that emit RSSI at a higher frequency and are configurable. After that, the application should be tested and modified based on having the application receiving more signals because now the RSSI smoothing and therefore trilateration will be impacted since the RSSI values will be more accurate and more frequent. Additionally, it could be worth it to determine if there are better smoothing and trilateration algorithms out there because that could help the location accuracy. Regarding logging data like was done for the tests, we think it would be useful to log the x and y coordinates of the beacons as they are placed on the map, so data analysis can be done on what happens as the phone moves to and from the beacons. After the bottleneck of the Bluetooth signals is solved, then the end user functionality and software development can begin. With regards to our report that would include implementing the MVC architecture, adding functionality to load in various maps, redesigning the interface to be useable by people who are visually impaired, and implementing software testing. Some of the aspects we reported on earlier in the semester such as architecture and unit testing were not able to be implemented due to the nature of our project becoming more research and feasibility analysis based rather than software development.

Regarding the overall process, we have a few recommendations to make. It would be ideal if the industry sponsor had the codebases rather than us getting the code from Github so that the code is one central location. We also think it would be beneficial to operate under the agile methodology since it’s a more robust way to allow teams to develop software in a rapid and organized pace with proper accountability and team communication.

**8. Acknowledgment References**

“How to Write Test Cases: Step by Step Guide with Examples,” *Guru99*. [Online]. Available: http://www.guru99.com/test-case.html.

“Indoor Navigation Wayfinding Mobile App,” *SPREO*. [Online]. Available: http://spreo.co/products/wayfinding-indoor-navigation-and-digital-displays/.

“Making indoor worlds discoverable,” *IndoorAtlas*. [Online]. Available: https://www.indooratlas.com/.

“SmartIndoor®,” *SmartIndoor®*. [Online]. Available: http://smartindoor.com/#sdk.

“Estimote,” *Map any space and visualize your location within it in real-time*. [Online]. Available: http://estimote.com/indoor/.