SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 7, 2017

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Karsten Andersen
Matt Gibson

ENTITLED

On the Relative Positioning of Mobile Devices Through Bluetooth Analysis

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

_____
Thesis Advisor

_____
Thesis Advisor

_____
Department Chair

_____
Department Chair

On the Relative Positioning of Mobile Devices Through Bluetooth Analysis

by

Karsten Andersen
Matt Gibson

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 7, 2017

On the Relative Positioning of Mobile Devices Through Bluetooth Analysis

Karsten Andersen
Matt Gibson

Department of Computer Engineering
Santa Clara University
June 7, 2017

ABSTRACT

Currently, there is no way to accurately track a mobile devices position in a given space without the use of GPS or WiFi. This means that people who are interested in understanding where they currently are in relationship to other people or landmarks and do not have access to these technologies are not able to get accurate location information. In many cases, GPS and WiFi fail to provide the accuracy required to provide useful information as well.

Our solution is a software layer that utilizes existing bluetooth hardware to create a highly accurate relative positioning system. The software layer will utilize a variety of algorithms to interpret the relative signal strength of known bluetooth transmitters and will be able to return a useful, three dimensional, representation of where a smartphone or other bluetooth device is. The framework offers a software solution that will be cheap and highly distributable due to the fact that it utilizes existing technologies.

Chapter 1

Acknowledgements

Our team would like to acknowledge Doctor Ahmed Amer for all of the support and guidance he gave us during the duration of this project.

Table of Contents

1 Acknowledgements **iv**

2 Introduction **1**
2.1 Problem Statement . . . 1
2.2 Objectives . . . 2
2.2.1 Functional Requirements . . . 2
2.2.2 Non-Functional Requirements . . . 2
2.2.3 Design Constraints . . . 3

3 Use Cases **4**

4 Design and Implementation **6**
4.1 Technologies Used . . . 6
4.2 Architectural Diagram . . . 7
4.3 Admin . . . 8
4.3.1 Front End . . . 8
4.3.2 Back End . . . 9
4.4 Framework . . . 10
4.4.1 Trilateration . . . 10
4.4.2 BLE Location Framework for Developers . . . 11

5 Testing and Results **14**

6 Project Management **15**
6.1 Risk Analysis . . . 15
6.2 Development Timeline . . . 15

7 Ethical Analysis **16**
7.1 Ethical . . . 16
7.2 Social . . . 16
7.3 Political . . . 16
7.4 Economic . . . 16
7.5 Health and Safety . . . 16
7.6 Manufacturability . . . 17
7.7 Sustainability . . . 17
7.8 Environmental Impact . . . 17
7.9 Usability . . . 17
7.10 Lifelong learning . . . 17
7.11 Compassion . . . 17

8 Conclusion **18**
8.1 Summary . . . 18
8.2 Lessons Learned . . . 18
8.3 Future Improvements . . . 19

A Design and Implementation Appendix **20**
A.1 Admin Portal . . . 20
A.1.1 Example JSON Code . . . 23
A.2 Framework . . . 24

B Project Management Appendix **25**
B.1 Risk Analysis . . . 25
B.2 Gantt Chart . . . 26

List of Figures

3.1 Use Case Diagram 4

4.1 Architectural Diagram 7
4.2 Home Page of Admin Portal 8
4.3 Trilateration In Two Demensions 10
4.4 Simple Trilateration Algorithm 11
4.5 Example location driven app using GPS (left) and BLE Location (right) 12
4.6 An example of how to add the BLE Location Framework to an empty iOS project . 13

A.1 Edit page of Admin Portal 20
A.2 View page of Admin Portal 21
A.3 Geo Calibration page of Admin Portal 22

B.1 Gantt Chart 26

List of Tables

B.1 Risk Analysis Table 25

Chapter 2

Introduction

2.1 Problem Statement

Currently, there is no way to accurately track a mobile devices position in a given space without the use of GPS or WiFi. This means that people who are interested in understanding where they currently are in relationship to other people or landmarks and do not have access to these technologies are not able to get accurate location information. While most people with modern smartphones do have nearly unlimited access to GPS and WiFi, this does not necessarily exclude them from the group of people who are affected by this issue. In many cases, GPS and WiFi fail to provide the fine grain accuracy required to provide useful information. Within a building GPS and WiFi can be unreliable because their physical signature is fundamentally not designed to address the need for indoor or finely tuned positioning. This can cause the distance and elevation that GPS reports back to lose accuracy. It cannot determine what floor youre on so if youre on the second floor or the ninth floor, you would show up in the same location. In terms of distance, it could show you at least several meters off from your real world location. Best case scenario, GPS will get someone to the door of a building they are looking for, but after that the person is on their own once inside of that building. Due to their design, GPS and WiFi cannot help you determine your location once inside a building in many cases, especially if the building has multiple floors. Often times stores and campuses provide physical maps for visitors. For a physical map however, there is no map for finding the map. To use the map, a person physically has to move to it to view where they are. Also, if there is a single map for all visitors, then a person cant carry it with them once they head for their destination. This means people can get lost on the way to their destination after leaving the map. Other current solutions try to solve this issue with proximity detection technology. The problem here is that proximity based systems can only tell a person how close they are relative to something; it doesnt give information on the direction or elevation of whatever the person may be

looking for. This can cause people to waste time figuring out what direction they need to head in or even put them a floor above or below their destination.

2.2 Objectives

Our solution is a software layer that utilizes existing bluetooth hardware to create a highly accurate relative positioning system. The software layer utilizes a variety of algorithms to interpret the relative signal strength of known bluetooth transmitters and will be able to return a useful, three dimensional, representation of where a smartphone or other bluetooth device is. The ability to understand the location of a device in two or three dimensions as opposed to the proximity of a device in one dimension allows for an exponential increase in the possible applications for such a positioning framework. Additionally, the ability to navigate using a smartphone indoors vastly improves the currently nondigital solution of central physical maps that are currently in use in many large buildings. The proposed framework offers a software solution that will be cheap and highly distributable due to the fact that is utilizes existing technologies. This means the framework will be able to support a wide variety of applications without needing to wait around for the next breakthrough spectrum standard in positioning technology.

2.2.1 Functional Requirements

Below are the initial requirements that were decided when the project was proposed.

Critical:

- The system will return a location when 3+ beacons are available
- The system will be able to interpret BLE beacon noise levels to produce a precise location
- The system will give more accurate location data than GPS or WiFi in specific situations

Recommended:

- Offline installation of system does not undermine its usability for devs and admins

2.2.2 Non-Functional Requirements

Critical:

- The framework is easy to use for developers
- The framework will return location coordinates with an security between 0m and 5m

- The system will scale to large applications

Recommended:

- Admin interface is easy to use
- Framework is optimized for battery conservations

2.2.3 Design Constraints

- System must work on iOS
- Must use low cost BLE beacons
- System conforms to iBeacon protocol

Chapter 3

Use Cases

Displays the use case diagram and the description of each. The purpose is to define tasks an actor will perform to achieve a certain goal.

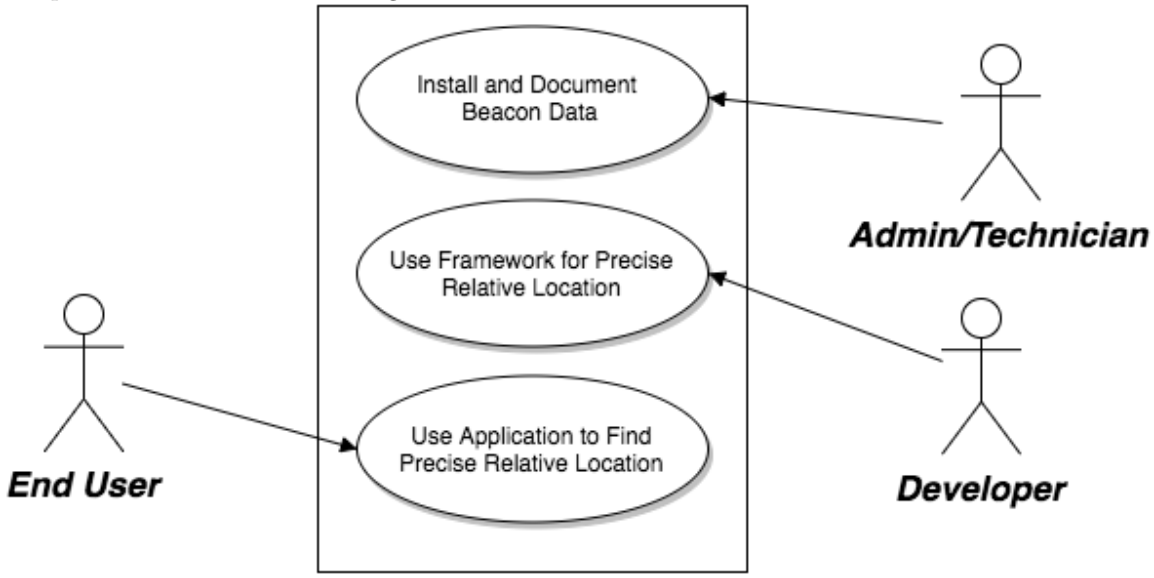

Figure 3.1: Use Case Diagram

1.

- Name: Install and Document Beacon Data
- Goal: To install and load location and other info about bluetooth beacons
- Actors: Admin/Technician
- Preconditions:Bluetooth beacons must be placed and initialized
- Steps: Location and other information is stored into a database
- Post-conditions: Database will have bluetooth beacon data loaded
- Exceptions: Information uploaded is incorrect

2.

- Name: Use Framework for Precise Relative Location
- Goal: To use framework to help application to discover the device's precise relative location
- Actors: Developer
- Preconditions:Framework must be loaded
- Steps: Developer writes code to utilize framework
- Post-conditions: Application will be able to discover the device's precise relative location
- Exceptions: Framework isn't loaded or used properly

3.

- Name: Use Application to Find Precise Relative Location
- Goal: To use application to discover the device's precise relative location
- Actors: End User
- Preconditions:Application must be downloaded
- Steps: User opens and uses application
- Post-conditions: End User will be able to discover the device's precise relative location
- Exceptions: Application isn't downloaded or used properly

Chapter 4

Design and Implementation

4.1 Technologies Used

Framework:

- Swift - Swift is the most modern language used for producing native iOS apps.It is more readable and flexible than Obj-C, making it a better choice for this project. It is also backwards compatible with Obj-C and C++ libraries so it does not limit the usage of open source material in the system. It is used for the framework as well as the trilateration calculations.

Admin Webservice:

- HTML/CSS - Will be used for formatting and styling of admin webpage.
- Javascript and JQuery - Javascript was used to add responsiveness and to provide underlying front end logic. JQuery was used to simplify DOM manipulation of admin webpage and AJAX calls.
- PHP - Used on the back end of the admin web service to sanitize inputs and interface with the database.
- MySQL - Used for the datastore of the admin web service.
- Google Maps API - Used for abstract mapping of floor plan to real life latitude and longitude.

Network:

- iBeacon - the iBeacon protocol will be used in all broadcasting beacons

4.2 Architectural Diagram

The following architectural diagram shows that the framework conceptually is built on top of the iOS CoreLocation service.The admin web app will only have to be reachable by the framework during the initial phase of the system installation or when major changes are made to the system.

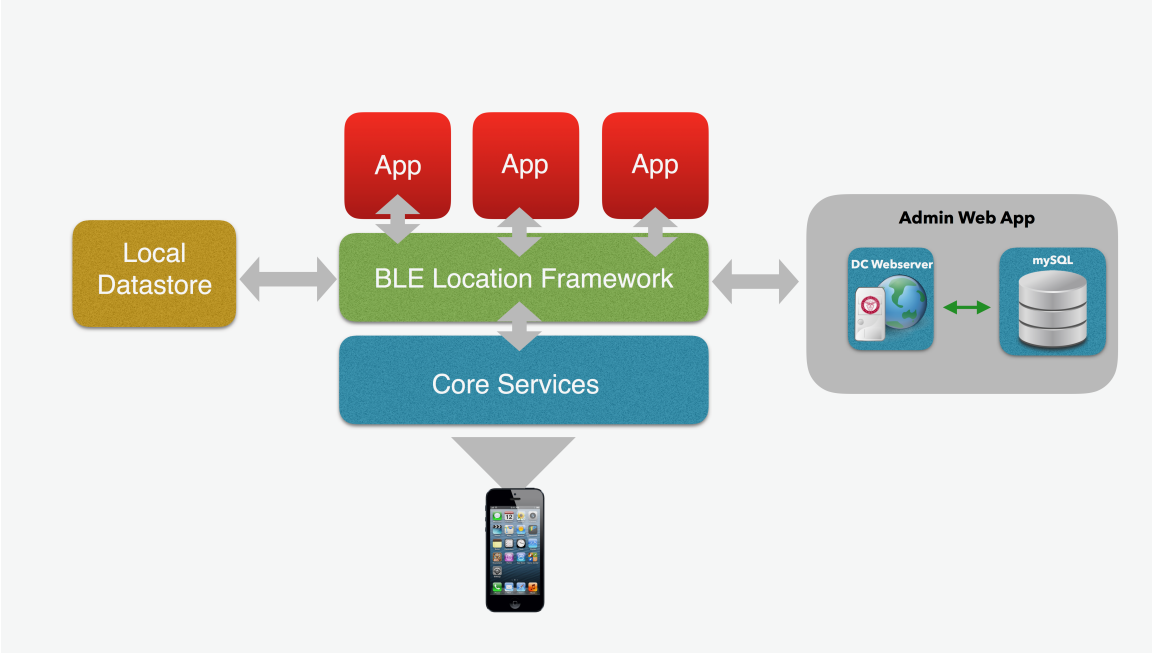

Figure 4.1: Architectural Diagram

4.3 Admin

The Admin Portal was built for documenting the locations of iBeacons as well as any other pertinent information. The overall structure was based on the LAMP (Linux, Apache web server, MySQL, and PHP) model. The user of the portal has the ability to add beacon data, edit beacon data, delete beacon data, view the beacon data, and geo calibrate the location service. The Geo Calibrate feature allowed for the user to drag a shape over the building that the BLE location system will be implemented in. See Appendix A for screenshots of some of these features. The anchor points in latitude and longitude were tracked and then used to calculate a ratio that can be used by the system to convert abstract units to real world latitude and longitude. The home page can be seen in figure ***.

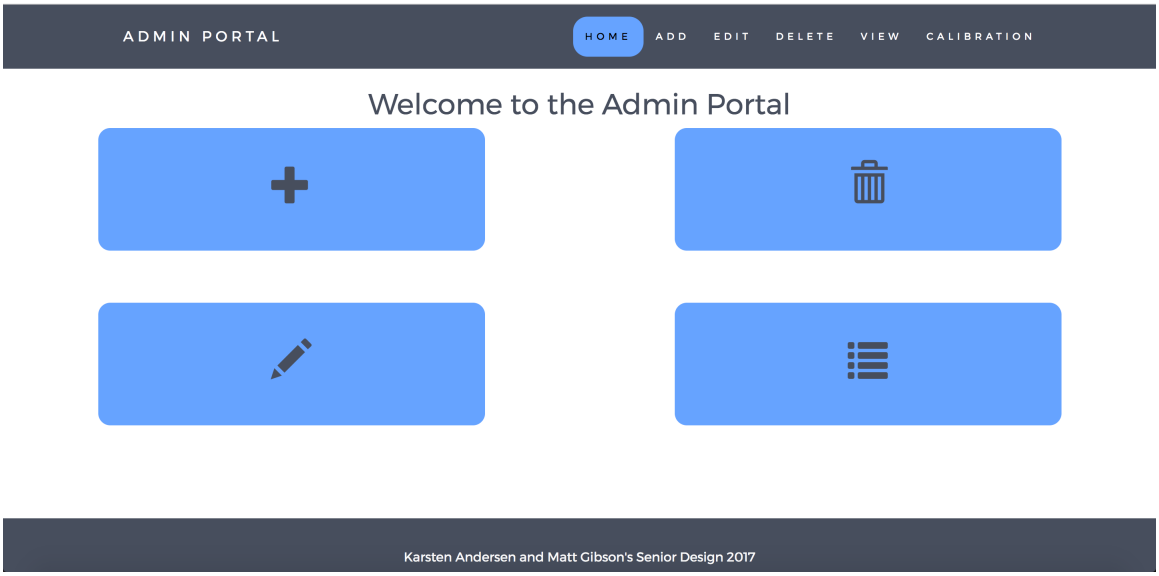

Figure 4.2: Home Page of Admin Portal

4.3.1 Front End

The front end was built upon the traditional web programming languages. HTML5 was used for the basic skeleton of the website and CSS3 was used for the aesthetics. Javascript was used for the front end functionality and jQuery was used to make the AJAX calls and to simplify the DOM manipulation. Bootstrap was used to give the site a mobile responsive design to allow the user to have the ability to enter and manipulate beacon data on the go with a smaller device. Finally the Google Maps API was used to allow the system to obtain real world latitude and longitude values.

4.3.2 Back End

Since the overall structure was based on the LAMP model, the website was hosted on an Apache Web Server with an HTTPS connection. The admin portal can be found at andersenkarsten.com/seniorDesign. The PHP programming language was used as the intermediary between the front end and the MySQL database. The MySQL database was used to store the important information for the iBeacons within two tables. The first table contained information having to do with beacon identification as well as location including: name, UUID, minor value, major value, and XYZ coordinates. The second table contained the four anchor points of the latitude and longitude, the ratio data for the abstract to real world calculation, and the unique identifier for that layout. The data supplied to and from the PHP and MySQL were done through queries and transported to the front end through JSON.

4.4 Framework

Once a system has been configured by an administrator using the admin portal discussed in the previous section, the BLE Location framework is available for developers to use in their Apps. The following two sections will discuss the underlying algorithm used by the BLE Location framework, as well as the methods exposed to developers using the framework.

4.4.1 Trilateration

Trilateration is the process of using a device's known distance from three set locations to determine the point in space where that device currently is. This algorithm is what enables the device to determine where it is relative to the set of iBeacons it currently can see. These iBeacons broadcast information about where they are located, and the device can implicitly determine its distance from the iBeacon. In the figure below, P1, P2, and P3 represent iBeacons, at set locations in their respective coordinate planes. Like the iBeacons, each point has three dimensions defining its location. In the Figure 4.2[1], all points are located on the same plane in the Z dimension for simplicity.

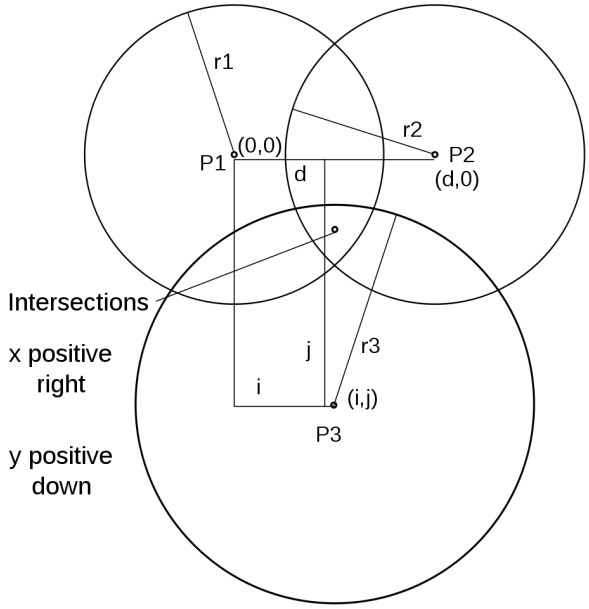

Figure 4.3: Trilateration In Two Demensions

There are a variety of different ways to perform the trilateration calculations. While Figure 4.2 shows three points sharing the same plane in the Z dimension, the underlying algorithm using the the BLE Location framework includes functionality to calculate position in three dimensions from know

[1]Source: https://en.wikipedia.org/wiki/Trilateration/media/File:3spheres.svg

distances to iBeacons in three dimensions. The algorithm and underlying code in the framework supports a Z dimension, but to keep the project within scope and to keep results accurate for proof of concept, most tests were conducted with a roughly constant plane in the Z dimension.

First consider the three spheres in Figure 4.2 on the plane Z = 0. This collapses the spheres into circles for simplicity. The equations of these spheres can be represented by:

$$r_1^2 = x^2 + y^2 + z^2$$
$$r_2^2 = (x-d)^2 + y^2 + z^2$$
$$r_3^2 = (x-i)^2 + (y-j)^2 + z^2$$

The point at X, Y, Z must satisfy all three equations. First, r1 and r2 can be used to eliminate y and z, and begin the process of solving for the X coordinate:

$$r_1^2 = x^2 + y^2 + z^2$$
$$r_2^2 = (x-d)^2 + y^2 + z^2.$$
$$r_1^2 - r_2^2 = x^2 - (x-d)^2$$
$$r_1^2 - r_2^2 = x^2 - (x^2 - 2xd + d^2)$$
$$r_1^2 - r_2^2 = 2xd - d^2$$
$$r_1^2 - r_2^2 + d^2 = 2xd$$
$$x = \frac{r_1^2 - r_2^2 + d^2}{2d}.$$

Substituting the representation of x back into the first equation for r1 will provide the intersection for the first two spheres:

$$y^2 + z^2 = r_1^2 - \frac{(r_1^2 - r_2^2 + d^2)^2}{4d^2}.$$

Substituting this new equation into the equation for the third sphere will allow us to solve for y:

$$y = \frac{r_1^2 - r_3^2 - x^2 + (x-i)^2 + j^2}{2j}$$

With these representations of x and y, z can be calculated from the first sphere:

$$z = \pm\sqrt{r_1^2 - x^2 - y^2}.$$

Figure 4.4: Simple Trilateration Algorithm

4.4.2 BLE Location Framework for Developers

The goal of this project was to make the BLE Location framework as simple and familiar as possible for a developer using it. The framework follows a familiar pattern for iOS developers because it is

loosely based on the delegate pattern used by the CoreLocation framework provided by Apple. It is designed to operate on the same software layer as CoreLocation in fact, working in tandem, rather than on wrapping the whole location services package. This allows the developer to have a higher level of control over when BLE Location is used instead of GPS or WiFi location. The framework exposes relevant hooks with these sorts of switching operations in mind. Figure 4.4 illustrates this context switch between GPS and BLE Location, and shows an example of how an App utilizing the framework would be able to provide an improved user experience in some cases.

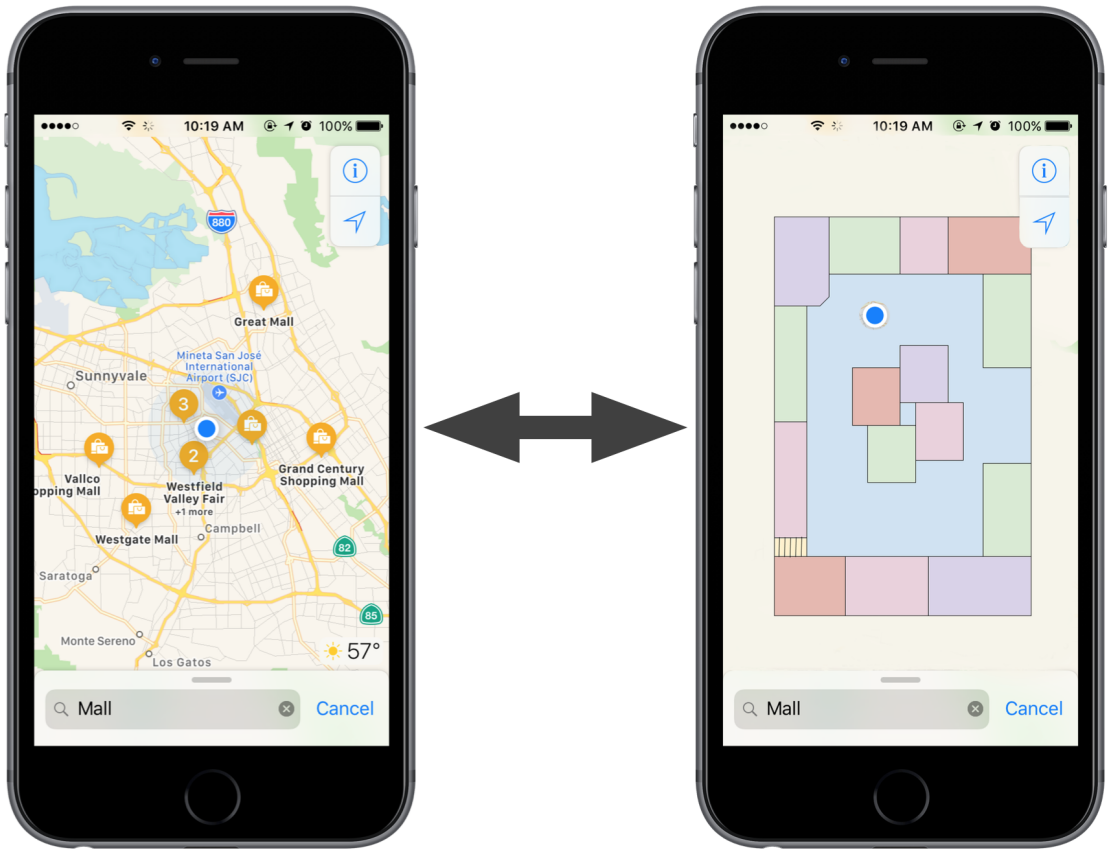

Figure 4.5: Example location driven app using GPS (left) and BLE Location (right)

Figure X.X illustrates how the BLE Location framework could be added to a project with only a few lines of code. After adding the framework source file to an iOS project, the developer must simply follow the basic setup illustrated below to begin integration with the BLE Location framework. Full documentation for the BLE Location framework is available in Appendix A, however a general understanding of how it works can be gained from examining a few select parts of Figure X.X:

- Line 4 shows how to tell a class it must conform to the BLE Delegate Protocol.

- Lines 12-14 show how to initialize a new BLEManger, and tell it that the current class will act as a delegate for it.
- Line 16 shows how to begin listening for updates from the BLE Manager.
- The functions on lines 22-37 are all examples of how to implement the functions required by the BLE Manager Delegate Protocol.

```
1  import UIKit
2  import CoreLocation
3
4  class ViewController: UIViewController, BLELocationDelegate {
5
6      var locationManager: CLLocationManager?
7      var myBLEManger: BLEManager?
8
9      override func viewDidLoad() {
10         super.viewDidLoad()
11         self.locationManager = CLLocationManager()
12         self.myBLEManger = BLEManager(locationManager: locationManager!,
13                                       apiURL: "https://www.myapi.com/api")
14         myBLEManger?.delegate = self
15         myBLEManger?.getItemsFromAPI(withURI: "/myItems")
16         if (myBLEManger?.startMonitoring())!{
17             /*congrats, your App is now ready
18             to provide BLE Location updates when available*/
19         }
20     }
21
22     func didRecieveBLELocation(withCoords: [Double]) {
23         //successfully recieved BLE Location with Coords
24     }
25
26     func BLELocationDidStall() {
27         //defer to GPS or wait
28     }
29
30     func BLEAvailabilityDidChange(isAvailable: Bool) {
31         if isAvailable {
32             //switch to BLE Location Handler
33         }
34         else{
35             //BLE no longer available - defer to GPS
36         }
37     }
38 }
```

Figure 4.6: An example of how to add the BLE Location Framework to an empty iOS project

Chapter 5

Testing and Results

- Unit Testing: Most testing will be done by our developers using manual white box testing. Automated testing will not be used because it would not give us the desired results and will take as much time setting up as it would do to just test manually.

  We will start off with General Feature Testing wherein we will code a particular feature and test it for it's functionality. As the Unit Testing model recommends, every time we write a new feature, we will test it for its functionality ensuring minimization of errors. This method will help us break our system into smaller, more manageable parts, the testing of which will be relatively simpler. It will also help us make these small parts ready to be used as a part of the bigger system.

- Regression Testing: As we will progress with the development of the system, we will begin implementing Regression Tests on top of the general feature testing. Unlike what we will do in Unit Testing, in Regression Testing we will write and test a particular feature and test the system. Once everything works fine, we will add the next feature and then retest the system from the beginning, instead of just testing the new feature.

  Simply put, regression testing is testing the system in its entirety every time a change or addition is made to it to make sure the already implemented features are not be affected by the new features in an undesirable way.

  Regression testing will help us efficiently manage our system whenever a change will be made.

- Ad hoc Testing: Ad hoc testing is performed without a plan of action. It will be performed by improvisation where the testers will find bugs by any means that seemed appropriate. Ad hoc tests will help us spot out any strange bugs that could arise in the operation of the system.

Chapter 6

Project Management

6.1 Risk Analysis

The two most potentially impactful risks are if the planned implementation of the system failed to provide good results, and if the project runs over budget.By coding in a modular style that provides low coupling, the system can be modified if necessary without having to start from scratch.Additionally, by buying most line items in the budget up front, we can be sure that the costs do not balloon as the project progresses.

6.2 Development Timeline

The table columns in the Gantt Chart lists the set time periods over which the project is distributed in. For our purpose, we have divided the project across ten weeks for each quarter. The rows of the table in the chart lists the various tasks that need to completed. The intersection of the rows and columns in the Gantt Chart would provide data about the team member who will be working on a respective task and when a particular task is due for submission or presentation. The Gantt Chart proves to be an efficient visual method of tracking the different tasks of the project and helps staying on schedule.

Chapter 7

Ethical Analysis

7.1 Ethical

There are no real ethical questions because there is no outcome that could have an immoral aspect.

7.2 Social

This could have an impact on society because it could decrease the amount of time people spend at malls or other places due to more efficient navigation of a building. People might spend less time in public, will less likely run into people at the mall or other places, and overall reduce person-to-person contact. It could also reduce the amount of impromptu purchases from businesses because people will not have to walk by unnecessary stores.

7.3 Political

There are no real political questions because this will not affect government at any level besides maybe easier navigation in their buildings.

7.4 Economic

The cost of development will be relatively low because iBeacons are inexpensive, easy to maintain, and the framework can be reproduced as many times as you want for no extra cost.

7.5 Health and Safety

There are no real risks to Health and Safety questions because bluetooth is safe for public use.
A potential benefit is that it could allows Emergency Medical Services to quickly navigate buildings to have a better response to a critical situation.

7.6 Manufacturability

iBeacons are an existing product and the framework requires no manufacturing.

7.7 Sustainability

iBeacons' batteries can be sustained for several years on average so they won't need to be replaced very often.

7.8 Environmental Impact

There will be minimal waste but once an iBeacon dies it will need to be recycled.

7.9 Usability

Our framework will be easy to use for developers to help them design excellent applications.

7.10 Lifelong learning

This project will help our group to continually learn how to gain knowledge outside a traditional lecture.

7.11 Compassion

This project has a compassionate side because it will help reduce the stress for developers creating location based apps and reduce stress for users when navigating inside buildings.

Chapter 8

Conclusion

8.1 Summary

People who are interested in understanding where they currently are in relationship to other people or landmarks and do not have access to GPS or Wifi are not able to get accurate location information. In many cases, GPS and WiFi fail to provide the accuracy required to provide useful information as well.

Our solution is a software layer that utilizes existing bluetooth hardware to create an accurate relative positioning system. The software layer utilized a variety of algorithms to interpret the relative signal strength of known bluetooth transmitters and try to return a useful, three dimensional, representation of where an iOS device is. The framework offered a software solution that is cheap and highly distributable due to the fact that it utilizes existing technologies.

8.2 Lessons Learned

With Bluetooth Low Energy Beacons, there is no one size fits all solution for filling a building. Buildings with more open floor plans will more easily accommodate these beacons and our algorithms because the signal will propagate off walls less. In a narrow room, a lot of noice will be created making it much harder for the real signal to be distinguished from the rest. Another problem is how easily the signal is absorbed by the human body. The system can return a different value for location if there are too many people between the iBeacons and the device. Finally, there is no definite number for the amount of iBeacons needed to fill a floor plan. Although there is the general rule of more being better, there is no definite answer to how many beacons the system really needs.

8.3 Future Improvements

Although Bluetooth Low Energy beacons have been shown to be able to return a position for an indoor positioning system, it does not do it well enough to be a complete solution. Instead of forcing iBeacons to be the answer and using the software side to correct their shortcomings, a new hardware solution would be a much better answer to this problem. The hardware could use a better frequency that allows for better frequency analysis but still possibly use a similar protocol to the iBeacon protocol to allow for ease of use. Instead of trying to force hardware to work with software solutions, this problem should be conquered by a hardware solution.

Appendix A

Design and Implementation Appendix

A.1 Admin Portal

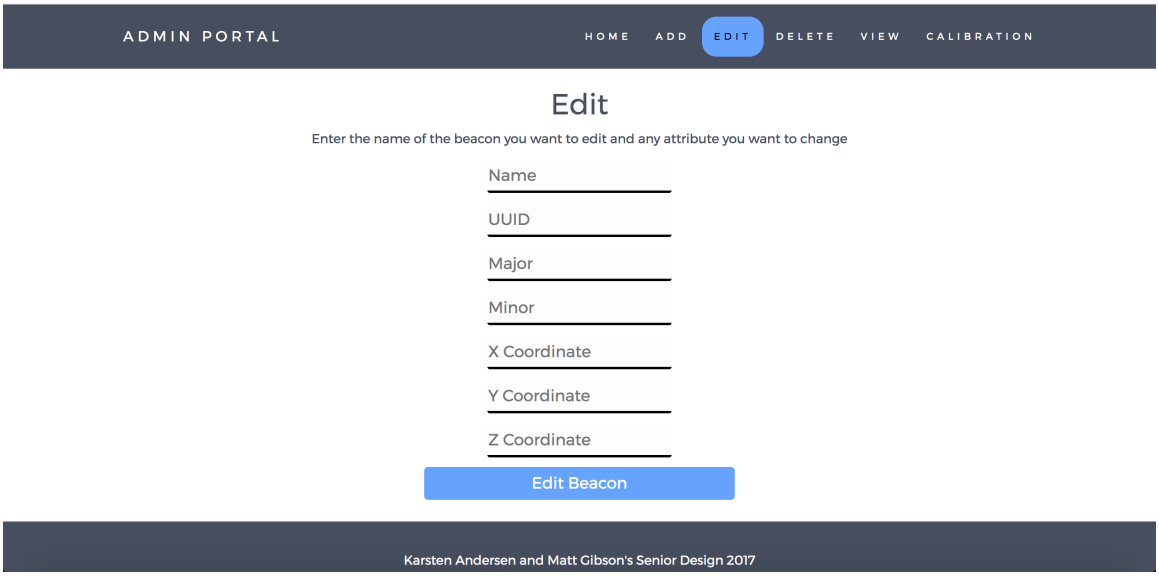

Figure A.1: Edit page of Admin Portal

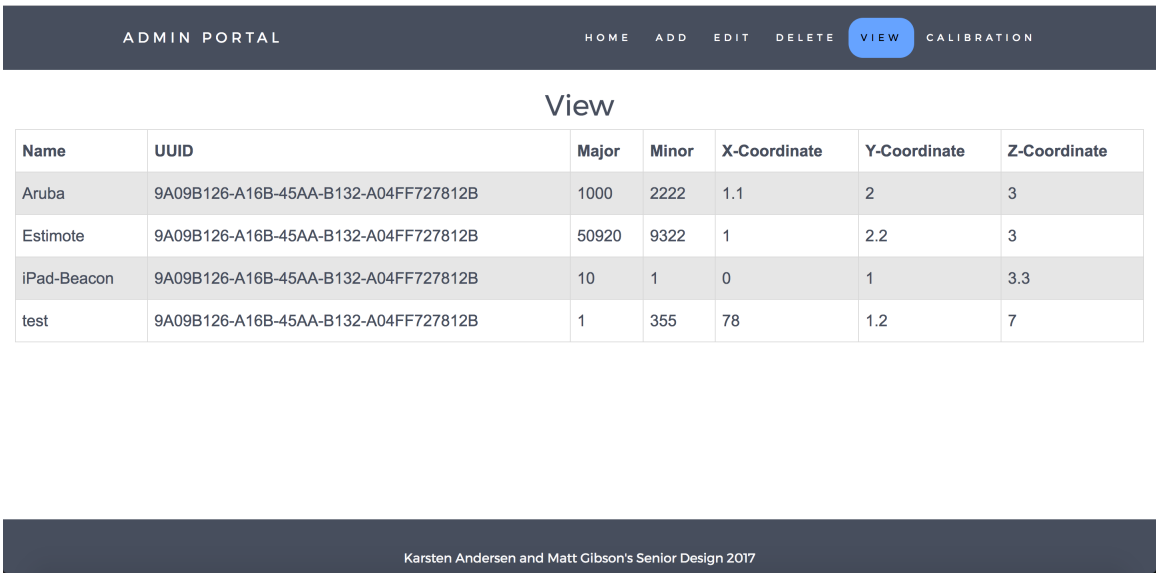

Figure A.2: View page of Admin Portal

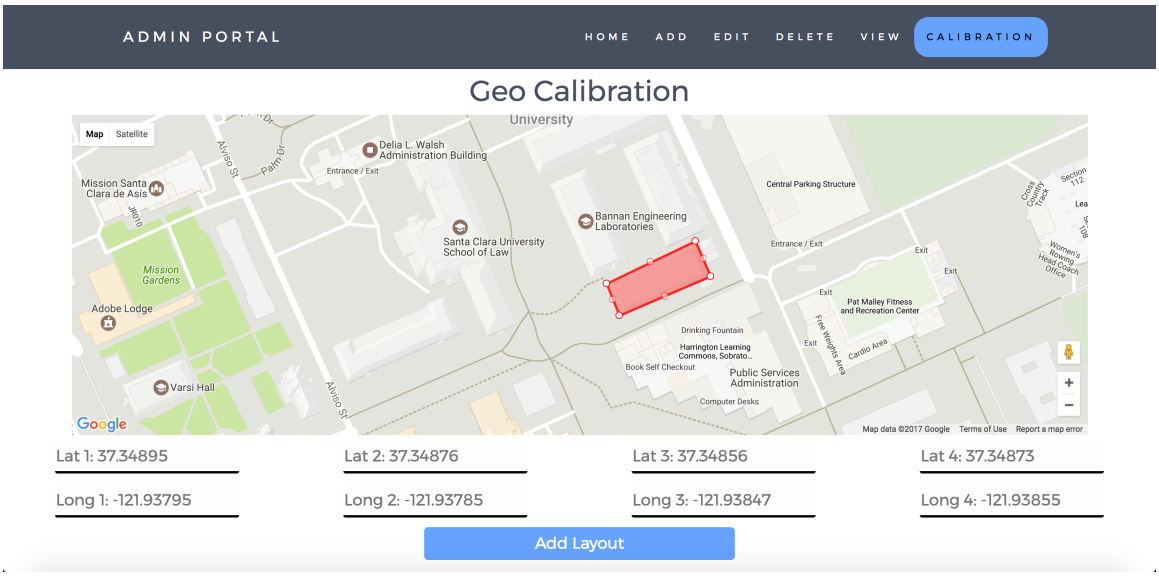

Figure A.3: Geo Calibration page of Admin Portal

A.1.1 Example JSON Code

```
[
    {
        "name": "Oltica_0",
        "UUID": "E2C56DB5-DFFB-48D2-B060-D0F5A71096E0",
        "Major": 1,
        "Minor": 0,
        "Coords": {
            "X": 0,
            "Y": 0,
            "Z": 0
        }
    },
    {
        "name": "Oltica_1",
        "UUID": "E2C56DB5-DFFB-48D2-B060-D0F5A71096E0",
        "Major": 1,
        "Minor": 1,
        "Coords": {
            "X": 1.0,
            "Y": 0,
            "Z": 0
        }
    }
]
```

Listing A.1: Example JSON code generated by Admin Portal and used by Framework

A.2 Framework

Appendix B

Project Management Appendix

B.1 Risk Analysis

Risk	Consequences	Probability	Severity	Impact	Mitigation
Planned Implementation failed to work	Mid-cycle decisions/changes are required	0.4	7	2.8	Research and careful planning.
Loss of Code	Significant setback in timeline and extra work	0.1	10	1	Use Git and Githut for everything. Don't forget to push code.
Scheduling Conflicts	Pace of project completion slowed/delayed. Communication is not fluid.	0.25	4	1	Clean communication. Schedule meetings far in advance.
Illness	Work distribution shifted unevenly. Project pace slowed	0.3	6	1.8	Get plenty of sleep. Don't cram for schoolwork.
Over Budget	Forced to cut features and and cut automation of test processes, slowing down the project	0.4	6	2.4	Research all system components and their individual costs.

Table B.1: Risk Analysis Table

B.2 Gantt Chart

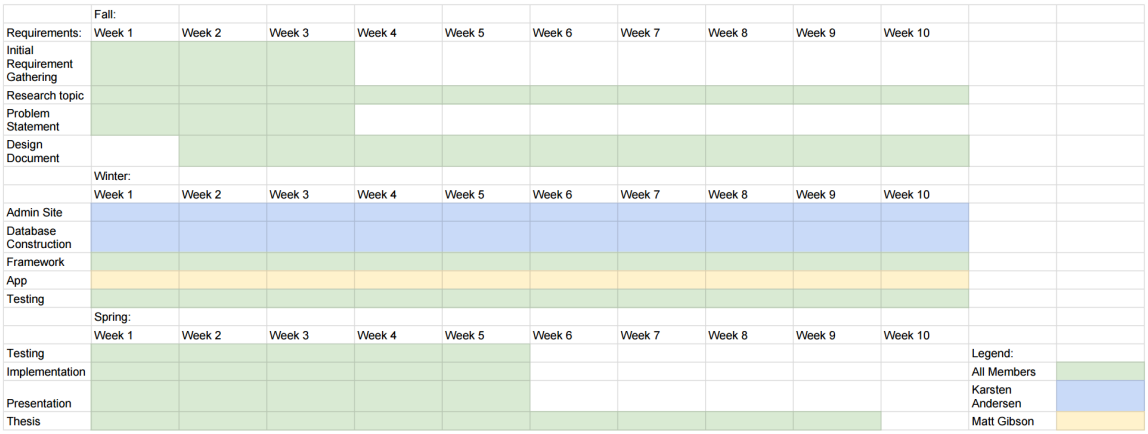

Figure B.1: Gantt Chart