



# *OCCUPANCY MONITORING SYSTEM USING INDOOR LOCATION*

*Cardiff University School of Computer  
Science & Informatics*

*One Semester Individual Project  
CM3202  
40 Credits*

Author: Rohan Sachdev  
1446200

Supervised By: Dr. Matthew J. W. Morgan

Moderated By: Dr. Richard Booth

## ABSTRACT

This project aims to build a system through which users are able to determine where individuals within a location of interest are currently placed. The system is based on Bluetooth Low Energy (BLE) Beacons mimicking the role of satellites in GPS but at a smaller scale to pinpoint an individual's location.

The project's approach takes into consideration beacon orientation and beacon properties in order to achieve the optimum system. The project further explores two fundamental approaches to building the ideal user-tracking system, namely foreground scanning and background monitoring. The foreground scanning approach involves trilateration, which is further broken down into 3 different approaches. In contrast, the background monitoring approach aims to build a more pervasive system.

The results from each approach are then compared and analyzed along with providing background as to why the results were achieved. Finally, the project explores future enhancements to the system with respect to exploring different technologies and different algorithms.

## ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Matthew Morgan, for guiding me throughout the duration of this project. It was a pleasure working with someone who consistently motivated me to get the best out of this project despite all the hindrances.

I would also like to thank Dr. Stuart Allen, Ms. Catherine Teehan, and Ms. Tracey Lavis for extending their constant support throughout the last 2 months of my course when unexpected hindrances occurred in my life.

Finally, I would like to thank my mother, Poonam, my father, Naresh, my brother, Gaurav, and my friends who have dedicated their time to not only helping me achieve the best I can throughout the duration of my course but also provided me with unparalleled support and belief in my abilities.

## TABLE OF CONTENTS

<b>Abstract .....</b>	<b>2</b>
<b>Acknowledgements .....</b>	<b>3</b>
<b>Table of Figures .....</b>	<b>6</b>
<b>1. Introduction.....</b>	<b>7</b>
<b>2. Background.....</b>	<b>9</b>
2.1 Indoor Location and GPS .....	9
2.2 Bluetooth Technology .....	10
2.3 iBeacon .....	11
2.4 The problem and proposed solution.....	12
2.5 Process towards achieving Solution .....	12
2.6 Choice of Technologies.....	14
<b>3. Possible Approaches .....</b>	<b>15</b>
3.1 Foreground Scanning/Ranging .....	15
3.2 Background Monitoring .....	16
<b>4. Specification and Design .....</b>	<b>17</b>
4.1 Beacon Plan .....	17
4.2 Android Application .....	18
4.2.1 Android Application for Ranging.....	18
4.2.2 The Application for Background Monitoring .....	20
4.3 Sending and Retrieving Data to and from the Database .....	20
4.4 Displaying the Position of Users .....	21
4.5 Flowchart.....	21
4.6 UML Class Diagram .....	23
<b>5. Implementation .....</b>	<b>25</b>
5.1 Beacon Plan Design .....	25
5.1.1 Beacon Plan for Ranging (Trilateration).....	27
5.1.2 Beacon Plans for Monitoring .....	29
5.2 Android Application .....	30
5.2.1 Android Application for Ranging/Trilateration .....	30
5.2.2 Android Application for Background Monitoring .....	38
5.3 Sending Data to the Database .....	40
5.3.1 Sending Trilateration Data to the Database .....	40
5.3.2 Sending Monitoring Data to the Database .....	41
5.4 Retrieving data from the Database and Displaying positions on the Front-End .....	42
5.5 Problems throughout Implementation .....	44
<b>6. Unit Testing .....</b>	<b>47</b>
6.1 Testing the Trilateration Algorithm .....	47
6.2 Testing the combinations for 6-beacon trilateration.....	48
6.3 Testing the Accurate Average Distance Comparator .....	51
6.4 Testing DistanceQueue and the accurate averages.....	52
<b>7. Results and Evaluation .....</b>	<b>56</b>
7.1 Testing of Ranging/Trilateration.....	56
7.1.1 Results of Trilateration Tests .....	58
7.1.2 Tests conducted for determining reliability of BLE Beacons .....	63
7.2 Analysis of Trilateration Results .....	65

7.3	Testing for Background Monitoring .....	65
8.	Future Works .....	71
8.1	Improvements on the current system .....	71
8.1.1	Improvements to the overall application .....	71
8.1.2	Improvements to receiving accurate distance values .....	71
8.2	Location Beacons .....	72
8.2.1	Location Beacons for Trilateration .....	72
8.2.2	Location Beacons for Background Monitoring .....	73
8.3	Other Technologies .....	74
8.3.1	Ultra-Wide Band Radio Technology.....	74
8.3.2	Magnetic Field .....	75
9.	Conclusions.....	76
10.	Reflection on Learning .....	79
	References .....	82
	Appendices .....	84
	Appendix A: Results for Trilateration.....	84
	Appendix A.1: Results after running 3-beacon trilateration according to Figure 5 .....	84
	Appendix A.2: Results after running 3-beacon trilateration according to Figure 6 .....	84
	Appendix A.3: Results after running 6-beacon trilateration.....	84
	Appendix A.4: Results after running 3-closest beacon trilateration.....	85
	Appendix B: Euclidean Distance Equation.....	86
	Appendix C: Log File displaying loss of signal in Beacons.....	87
	Appendix D: Numerical Results for Distance Values of Beacons at 1m .....	91

## TABLE OF FIGURES

Figure 1: Flowchart for trilateration .....	22
Figure 2: Flowchart for background monitoring .....	23
Figure 3: UML class diagram for the trilateration system .....	24
Figure 4: UML class diagram for the monitoring system.....	24
Figure 5: 3-beacon plan for trilateration .....	27
Figure 6: Alternate 3-beacon plan for trilateration .....	28
Figure 7: 6-beacon plan for trilateration .....	28
Figure 8: Beacon monitoring plan .....	30
Figure 9: Interaction of all components in system .....	46
Figure 10: Results of 3-beacon trilateration according to fig. 5 .....	58
Figure 11: Results of 3-beacon trilateration according to Fig. 6 .....	59
Figure 12: Distance between Expected Values and Values On Screen for 3-Beacon Trilateration .....	59
Figure 13: Results for 6-beacon trilateration .....	60
Figure 14: Distance between Expected Values and Values On Screen for 6-Beacon Trilateration .....	61
Figure 15: 3-closest beacon trilateration results.....	62
Figure 16: Distances between Expected Values and Values on Screen for 3-Closest Beacon Trilateration.....	62
Figure 17: Estimated distances at 1m for Estimote.....	63
Figure 18: Estimated distances at 1m for Kontakt.io .....	64
Figure 19: Beacon Monitoring Results .....	67
Figure 20: Beacon notificaions displayed within the span of 5 minutes .....	68
Figure 21: False Enter Notification .....	69
Figure 22: HTC One M10 Background Monitoring .....	69
Figure 23: Samsung Galaxy S7 Background Monitoring .....	70

## 1. INTRODUCTION

This report outlines the stages and procedures to develop a system in which users can track individuals in an indoor location of interest, which for the purposes of this project would be the Cardiff University's School of Computer Science Windows Lab 2. The intended audience for this application would primarily be the students and staff of Cardiff University's School of Computer Science. Furthermore, this project can potentially be extended to the rest of the University especially to the Trevithick Library (and all other libraries) to allow students to plan their visits to the libraries when studying. The system can also potentially be extended to professional environments for managers who wish to track their employees' positions within a location of interest such as an office. Throughout developing the system mentioned, the fundamentals of the project would be the choice of technology used in order to achieve the system, the choice of algorithms used to achieve a working system, and the orientation and configurations of the technologies used.

The primary aim of this project would be to produce a solution that is able to determine the location of an individual indoors along with showing how many individuals are currently within the Windows Lab 2. Furthermore, optimally, the solution presented would allow users to view the location of other users within the lab. Alternatively, the solution presented would involve allowing users of the Windows Lab to see how many other individuals are currently located within a particular section of the labs. In terms of privacy and security, the application simply assigns a random user ID to the phone when the application is installed; this user ID is not linked to any accounts the user has, and thus, no personal information of the user is revealed. This thus allows users to plan their visits to the labs. This would be shown in the form of a heat-map. To achieve the solution mentioned, a proof of concept would be presented in the form of an Android application, which is the fundamental constituent of the system for user-tracking. While this is the central aim of the project allowing the project to be geared towards this solution, the project will take into account many other sub-aims mentioned in Section 2.5, page 14.

In essence, this report examines 2 approaches to achieving an indoor location tracking system: foreground scanning/ranging and background monitoring. The foreground scanning/ranging approach can be further broken into different algorithms due to the fact that this approach achieves granular detail. These algorithms are then further tested and

analyzed in order to determine which algorithm produces the optimal results. In contrast, the background monitoring approach aims to produce a more pervasive application. The technologies used in the project will also be evaluated in order to provide more insight into the project. The assumptions for each approach are: each user will have the application installed on his/her phone, each user will have an active internet connection on his/her phone – in order to send information from the phone to the central server and each user will have an active Bluetooth connection on his/her phone – in order to receive more information for positioning the individual in the lab.



## 2. BACKGROUND

Initially, this section of the report provides the motivation to developing this system, which is comparable to the already established Global Positioning System (GPS). More detail is then provided along with insight into the technologies being used to achieve the system. Finally, the topic of research is then derived from the aim along with outlining each step to achieving the research.

### 2.1 INDOOR LOCATION AND GPS

The Global Positioning System (GPS) is essentially a system which is able to determine the location of an individual on Earth. The system makes use of signals from satellites orbiting Earth by determining the distance from at least three satellites to determine a 2-D position. The system then uses an algorithm known as trilateration, which makes use of the satellites' position and the distance from each satellite in order to determine the position of the individual in the form of latitude and longitude <sup>(1)</sup>.

GPS, however, cannot determine the precise location of an individual indoors. This is due to the fact that GPS lacks accuracy given that the signals received from the satellites are microwaves that are also prone to interference. The current accuracy of GPS is said to be within the range of 4.9m in an open sky. However, the signal can be disrupted due to hindrances such as bridges, buildings, and trees. For example, buildings can block the signal from reaching the phone or signals can be reflected onto the phone, which causes an inaccuracy in the signal received. The values received from the GPS are not precise enough to determine a location indoors especially when dealing with dimensions of rooms that are potentially 20m x 20m (length and width) along with interference – or blocked signals – disrupting the signal received indoors causing a further inaccuracy.

Indoor location has been a growing industry in technology and engineering for a number of years, and there have been many different methods in which indoor location can be achieved such as the use of Bluetooth, Wi-Fi, or radio waves and Earth's magnetic fields. Essentially, to map an individual indoors, a number of variables must be known. For example, the area of the indoor location must be known with respect to length and width - and if a 3-D model is necessary, then height must also be known. Primarily, the most common mathematical modelling method of mapping an individual's position indoors is

trilateration. Trilateration attempts to produce an individual's location in terms of X and Y-coordinates by using the distances to three known points<sup>(2)</sup>. The system in this project will be developed as a 2-D model given that height is not an essential factor in determining the position of an individual within the Lab 2.

As mentioned, three known points with the distances to each point are essential to achieve trilateration. The known points would essentially have to replicate the function of satellites in GPS. For the purposes of this project, the Bluetooth beacons emitting signals to the phone are used as satellites for determining the position of an individual indoors. The fundamentals of the Bluetooth technology are elaborated on in the following sections.

## 2.2 BLUETOOTH TECHNOLOGY

Bluetooth technology is a wireless technology that is used primarily for exchanging data from one device to another. Initially, Bluetooth was proposed and started in 1994 by Nils Rydbeck, CTO of Ericsson Mobile and Johan Ullman in Lund, Sweden. The original purpose of Bluetooth was to allow a mobile phone to communicate with accessories such as a wireless headset<sup>(3)</sup>. Since then, Bluetooth has gone through many iterations of development in terms of expanding the technology from Bluetooth 1.0 to Bluetooth 5.0.

Bluetooth 4.0 introduced a new protocol known as Bluetooth Low Energy (BLE). Bluetooth differs from BLE in the sense that BLE is both lower in power consumption and monetary cost than traditional Bluetooth. BLE achieves this by sending signals periodically whereas traditional Bluetooth signals are continuous. This essentially allows applications to exchange minimal amounts of data when sending Bluetooth packets.

In simple terms, BLE's architecture has two major components, namely the physical layer and the link layer. The physical layer is responsible for controlling radio signal transmission and receiving, and the link layer defines the packet structure, which includes the state machine and radio control along with providing link layer level encryption.

The physical layer operates in the 2.4 GHz Industrial Scientific Medical (ISM) band (2402 MHz – 2480 MHz). BLE specification selects 40 pre-defined radio frequency (RF)

channels with 2 MHz channel spacing. There are 3 reserved channels for discovery, connection establishment, and broadcast.

The fundamental link layer operations are advertising, scanning, and connection establishment. Advertising essentially allows BLE devices to send packets to other devices with Bluetooth receivers to determine the presence of BLE devices in the immediate vicinity. Scanning, in contrast, is an operation where the devices with Bluetooth receivers are listening for incoming signals from the BLE devices in order to – for the purposes of this project – receive the data broadcast by the BLE devices. There are 2 scanning modes: active and passive. Passive scanning involves devices simply listening for incoming packets from BLE devices via advertisement. In contrast, active scanning involves listening for an advertising packet from a BLE device and sending an additional scan request to receive more information about the BLE device<sup>(4)</sup>. Passive scanning can be done in the background due to the fact that passive scanning simply involves receiving an advertising packet. Active scanning, however, needs to be done in the foreground in order for the device to send an additional request for more information from the BLE device.

With respect to BLE beacons used in this project, active scanning can be comparable to foreground scanning/ranging whilst passive scanning can be comparable to background monitoring, which are covered more in detail in Section 3, page 14.

### 2.3 iBEACON

In 2013, Apple announced the integration of the iBeacon standard into iOS 7<sup>(5)</sup>. Essentially, the iBeacon standard is a general term given to the technology used by both Android and Apple devices receiving Bluetooth signals from Bluetooth beacons to produce an appropriate response. The underlying communication technology is BLE<sup>(6)</sup>. The iBeacon technology is also used by companies such as Estimote and Kontakt.io that produces Bluetooth beacons. The Estimote beacons consist of a 32-bit ARM Cortex CPU, an accelerometer, a temperature sensor, a motion detector, and a 2.4GHz radio using Bluetooth 4.0 Smart (BLE). Kontakt.io beacons also consist of the same CPU. Estimote and Kontakt.io beacons broadcast packets of data consisting of the beacons' iBeacon ID, signal strength, and other metadata such as temperature<sup>(7)</sup>. This information can then be used to

develop context-aware applications such as an application for the indoor location monitoring of an area, which will be the central solution for this project.

## 2.4 THE PROBLEM AND PROPOSED SOLUTION

As mentioned in the introduction, GPS does not translate to the precise location of an individual indoors. Thus, primarily, the problem would be to determine the position of an individual within a building to the point where we can position that individual within the room. Additionally, this can be used to determine the number of individuals currently occupying the Windows Lab 2.

The question that is left to be asked is that can Bluetooth beacons be used in simulating trilateration at a much more granular scale than GPS to determine an individual's location within the room?

The solution presented in this project would be a proof of concept of an Android application that attempts to use the information received from BLE Estimote and Kontakt.io beacons along with a trilateration algorithm to produce a relative position of the individual's location within Windows Lab 2 of Cardiff School of Computer Science. Where possible, it will be attempted to make the solution as pervasive as possible. Thus, minimizing the user's interaction with the phone in order to transmit location data. The solution also incorporates the ideal positioning of BLE beacons with respect to the dimensions of the Windows Lab 2.

To achieve this proof of concept, the Estimote Software Development Kit (SDK) will be integrated into the Android application in order to ensure that a solution is developed<sup>(8)</sup>. This API will be used as the basis of development for the proof of concept along with the development of the trilateration algorithm and the ideal beacon positions for the lab.

## 2.5 PROCESS TOWARDS ACHIEVING SOLUTION

The process of determining a location of an individual indoors – using BLE beacons – will be broken down into the following segments in no particular order:

- Determining the dimensions of the Windows Lab 2: This will be the initial step to determining the ideal position of the BLE beacons. Furthermore, any individual that receives a signal from the beacons outside of these dimensions can also be considered outside of the Windows lab.
- Develop a trilateration algorithm: Trilateration is the fundamental algorithm in determining the position of an individual indoors. The algorithm produces a pair of x and y-coordinates from three known points (also in the form of x and y-coordinates) and the distances from each point.
- Determine the ideal advertising interval for the BLE beacons – this is necessary given that BLE (Bluetooth 4.0) has gotten rid of a constant signal: As mentioned in Section 2.3, page 11, the beacons transmit their signal periodically. Thus, the ideal advertising interval is necessary to determine whether the individual is within the location of the Windows lab or not. A significant point to note is that a lower advertising interval severely depletes the battery life of the BLE beacons, and therefore, this must be taken into account when opting for the ideal advertising interval.
- Determine the ideal range of coverage for each Beacon: For the purposes of trilateration, each beacon should have a range large enough to overlap with at least two more beacons in order for the area covered by three beacons to be large enough to cover the entire lab.
- Determine the ideal positions of the BLE beacons: Involves positioning in a manner which covers the entire room. In addition, the Windows lab hosts several computers, which consist of both metal and glass parts. These parts can cause interference that results in an inaccurate received signal, which thus results in an inaccurate position indoors. Thus, the ideal position would cover both the entire lab and avoid as much interference from the computers as possible.
- Develop a web client to send and receive data to and from a central database: As mentioned above, the project involves allowing other users to determine the number of individuals within the lab along with the position of those individuals within the lab.

Thus, a central database is essential in allowing all users in determining the positions of those individuals within the labs.

The primary limitation for this project from being a complete system is the interference on Bluetooth signals. The project takes this into consideration and statistical techniques are applied to reduce the noise in the signals received to produce a more accurate distance estimate. Furthermore, the project does not take different Bluetooth receivers into account. Thus, certain phones may be harboring a higher quality of hardware producing more accurate results while other phones may not. The testing on the system as a whole is done on a variety of phones to produce a more realistic scenario for the system. Another major limitation is that the beacons used are proximity beacons manufactured by Estimote and Kontakt.io. Thus, a wide variety of hardware is not tested for the purposes of simulating the role of satellites in GPS.

## 2.6 CHOICE OF TECHNOLOGIES

Over time, this project has changed focus from being a simple occupancy application to a more sophisticated positioning application. Thus, initially, the choice of Bluetooth beacons was used for simply determining whether an individual is currently within the Lab 2 or not. However, as the focus shifted to positioning the individual within the lab, the underlying BLE beacon technology remained the same given that BLE beacons are capable of replicating the role of satellites in GPS.

The choice of building an Android application remained as well due to the fact that most, if not all, phones harbor a Bluetooth receiver. Thus, phones are the most common devices capable of receiving Bluetooth signals or packets from the BLE beacons in order to place that individual within the lab as opposed to using a laptop which may not necessarily consist of a Bluetooth receiver.

### 3. POSSIBLE APPROACHES

As mentioned in the Introduction, the report analyzes two alternate approaches to achieving the system. The approaches are described in further detail in this section. Each approach essentially produces a different Android application. The ranging/foreground scanning approach theoretically should produce a more precise position of an individual within the Lab 2. In contrast, the background monitoring, despite not precisely positioning an individual within the room, should be more pervasive in nature which essentially does not rely on the user completing any actions for the application to determine the user's position. These approaches will be implemented and compared in Section 7, page 52.

#### 3.1 FOREGROUND SCANNING/RANGING

Typically, phones with applications that integrate Beacon SDKs – for example, the Estimote SDK – can undergo a process known as ranging. Ranging involves the phone actively scanning for beacons in the foreground constantly until the application is not running in the foreground anymore. The reason for the active scanning is for the phone to send a request to receive more information from the BLE beacons, as mentioned in Section 2.2, page 10. Ranging provides the application with much more information from the BLE packets received by the phone with respect to a list of all beacons nearby along with their respective UUIDs, major and minor values. A distinct feature of ranging is that ranging provides enough information to make proximity estimations. The list of beacons received from ranging is also in order of closest to the furthest beacon. A significant point to note is that ranging is quite draining on the phone's batteries, and thus, it is not recommended for the application to undergo ranging for long periods of time, which is another limitation of the project<sup>(9)</sup>.

Given that ranging provides beacon data in much more detail, it can be used to determine the distances from each beacon. As mentioned in Section 2.5, page 12/13, trilateration involves determining the position of an individual based on the distances from three known points. Thus, the three known points in this scenario would be the coordinates of the BLE beacons. Furthermore, there are three possible strategies to using trilateration to determine an individual's position in the lab: using 3 beacons, using 3 of the closest beacons out of 6 beacons, and using all 6 beacons.

The 3-beacon approach essentially involves placing 3 beacons after having determined their ideal positions for avoiding interference due to the metal, glass, and water from other human bodies. The 6-beacon approach involves using all 6 beacons to determine the position of the individual. Essentially, the idea is to use the distances from all 6 beacons and divide them into subsets of three. The mean of the positions determined from all of the subsets would then be taken as the position of the individual. The 3-closest beacon approach involves placing all 6 beacons in the lab. The application will receive distance values from all 6 beacons, and based on the distance, the list of beacons is sorted in descending order with the 3 closest beacons. Using the 3 closest beacons, the trilateration algorithm will be applied to determine the individual's position.

### 3.2 BACKGROUND MONITORING

Background monitoring, in contrast to ranging, is a much simpler process as it does not provide as much detail as ranging. In simple terms, background monitoring involves the application checking whether the phone has received a Bluetooth advertisement packet from a predefined list of beacons. The application checks for whether the phone has “entered” or “exited” a particular region – each beacon has its own region, which is essentially a circular space in which the beacon's packets are able to be received by another Bluetooth device<sup>(10)</sup>. Background monitoring is essentially passive scanning for beacons in the vicinity of the phone.

Despite background monitoring not providing as much detail as beacon ranging, it can potentially be used to determine the user's location – not with exact precision as trilateration – but a relative precision based on the coverage of each beacon's region, and if a user has entered the coverage of a particular beacon – which is determined by whether the user's phone has received the beacon's advertisement packet, then it can be said with certainty that the user is in a position relatively near to that particular beacon. A significant point to note is that background monitoring, as its name suggests, can occur in the background. Thus, the user does not have to have the application running on his/her phone to determine his/her location.



## 4. SPECIFICATION AND DESIGN

This project can be split up into four fundamental constituents:

1. The Beacon plan: the plan essentially positions each beacon within the lab for trilateration, and for background monitoring, it shows both the beacons and coverage in order to depict the theoretical distinctions in the sections of the lab covered by each beacon.
2. Android applications: these applications interact with the BLE beacons to determine the position and send the position to a central database. Furthermore, this application distinguishes all individuals by assigning each individual a random Universally Unique Identifier (UUID). The UUID is then permanent with that phone. Thus, a user's status of "in the lab" or "out of the lab" gets updated on the database based on the feedback from the beacons while the UUID remains the same. With respect to ranging, the application performs trilateration to determine the user's position, and this defines the user's status. Background monitoring, in contrast, defines a user's status according to the zone that the user is currently based in. Both applications would then be implemented and tested to determine the best approach.
3. Sending and Retrieving data: Prior to sending data to the database, the data is preprocessed according to its specified requirements. In addition, when required, the data will also be retrieved from the database.
4. Displaying the positions of the individuals within the labs: A scalar representation of the lab is created, and the individual will be mapped to a specific point within the lab.

These constituents are elaborated on in this section.

### 4.1 BEACON PLAN

The beacon plan is essential to the system due to the fact that the orientation of the beacon yields different values in positions. Hence, research would need to be conducted to determine the ideal beacon plan when it comes to avoiding interference. Furthermore, the number of beacons used for the purposes of determining the individual's position would

also produce different values. Therefore, this would also be taken into consideration with the beacon plan.

When it comes to ranging, the trilateration algorithm is one that involves 3 known points with distances to each point to determine the position of an individual. Therefore, research based on whether 3 beacons or more are necessary to determine the accurate position.

With respect to monitoring, however, given that granularity is essential in pinpointing users to the labs, all 6 beacons would definitely be required. Thus, assigning each beacon to an area of the lab is essential to finding out the location of an individual.

## 4.2 ANDROID APPLICATION

The Android application is the fundamental component linking the values received from the Beacons and sending them to the central database. The BLE beacons have no capabilities other than simply sending a Bluetooth signal periodically. Hence, the Android application is required to process the Bluetooth signals received by the beacons and based on the signals received, if the application is ranging, then the application must determine the position in the form of x and y-coordinates. The coordinates will then be sent to the central database, where they can be retrieved by other users to see where in the labs the individual is. If the application is monitoring in the background, then the application must determine which beacon's region the individual is currently located in and send this to the central database. The location would then be retrieved from the database and displayed on the front-end for users to then view which section of the lab the individual is in.

### 4.2.1 ANDROID APPLICATION FOR RANGING

As mentioned in Section 3.1, page 15, ranging provides the application with much more detail by providing a list of Beacons with their respective properties. The application uses the Estimote SDK to start ranging, and once a list of Beacons has been received from ranging, the application can then produce a distance estimation from each of the beacons. Once the distance estimation has been made based on the Received Signal Strength

Indicator (RSSI) and Measured Power – this will be elaborated on in Section 5.1, page 24 – the application will initially store the distances estimated from all beacons. Afterwards, the mean of the distances will be calculated; based on this initial mean, the variance and standard deviation will be calculated. The standard error of the mean is then determined by dividing the standard deviation by the square root of the sample size. The upper bound and lower bound are set using the following equations:

$$(1) \text{ Upper bound} = \text{mean} + \text{standard error} * 1.645$$

$$(2) \text{ Lower bound} = \text{mean} - \text{standard error} * 1.645$$

Any value greater than the upper bound or lower than the lower bound is considered to be an outlier and is discarded from calculations in order to determine an accurate average. The value of 1.645 is used to remove the top 5% and bottom 5% of values as these are considered to be outliers in the data and are skewing the data.

With respect to ranging for the 3-beacon trilateration, the application will retrieve the accurate average of the distances from each of the three beacons as mentioned previously, and based on this, the trilateration algorithm will be performed to determine the position of the individual. The positions are further stored and the same procedure as the one for the distances will be applied to retrieve a more accurate average for the positions. The accurate average is then sent to the central database along with the UUID.

Regarding 6-beacon trilateration for determining the position, the procedure for storing distances from each of the six beacons and determining the accurate average is the same as the one described above. The difference in this method of determining the position is that all 6 beacons along with their accurate distance values are divided into subsets of three. The positions of each of the subsets are determined using the trilateration algorithm and stored. Finally, the average of the positions is calculated and sent to the central database along with the UUID.

Similar to the procedure above, the 3-closest beacon trilateration will retrieve the accurate mean of the distances from all six of the beacons. The application will then determine the top three closest beacons based on the accurate averages of the distances determined from

each of the six beacons. Afterwards, the application will use the accurate average of the distances from the three closest beacons to execute the trilateration algorithm. Once again, the positions yielded from the execution of the trilateration algorithm were stored and the accurate mean of the positions were calculated and sent to the central database along with the UUID.

#### 4.2.2 THE APPLICATION FOR BACKGROUND MONITORING

As mentioned in the introduction of this section, the application will determine in which beacon's region the user has triggered an "onEnteredRegion" event. Based on that, the application then sends the beacon zone – which is a different pre-set value for each beacon in order to distinguish each section of the lab – to the central server along with the UUID. Similarly, when the user has triggered an "onExitedRegion" event, the application sends the beacon zone "0" to indicate the user is not in any region within the lab.

#### 4.3 SENDING AND RETRIEVING DATA TO AND FROM THE DATABASE

The central database in this project is essential in providing feedback to all users who wish to use the application and determine the number of users currently in the labs. Thus, the application must send an individual's information to the database anonymously for privacy purposes – which is why a random UUID is assigned to each individual when the application is initially installed. The application must be connected to Wi-Fi at all times in order send data as post variables to the database and retrieve data from the database.

Sending data to the database involves checking whether a few criteria is matched prior to storing the data within the database. For the purposes of the system, a relevant point is that Bluetooth is prone to interference. Thus, the estimated x and y-coordinates would constantly be changing. To store the data in the database, the criteria is set to determine whether the previous x and y-coordinates, when compared to the current x and y-coordinates, have a Euclidean distance of greater than 1m. If it does, then the user has moved a significant amount, and the new coordinates should be updated onto the database.

#### 4.4 DISPLAYING THE POSITION OF USERS

With respect to ranging, the position of users will be displayed as a circle within a scalar representation of Lab displaying each individual user in his/her particular position of the lab.

With respect to monitoring, however, the area the beacon covers would be displayed on the scalar representation of Lab 2. Furthermore, the count of the number of people within that particular area of the lab would also be displayed, and the color of the beacon area would also change based on the number of users currently occupying an area to provide visual feedback.

#### 4.5 FLOWCHART

The following diagram indicates a simple flowchart of the system with respect to trilateration. The flowchart does not describe in detail each trilateration algorithm, it simply indicates that a trilateration algorithm is being performed to determine an x and y-coordinate.

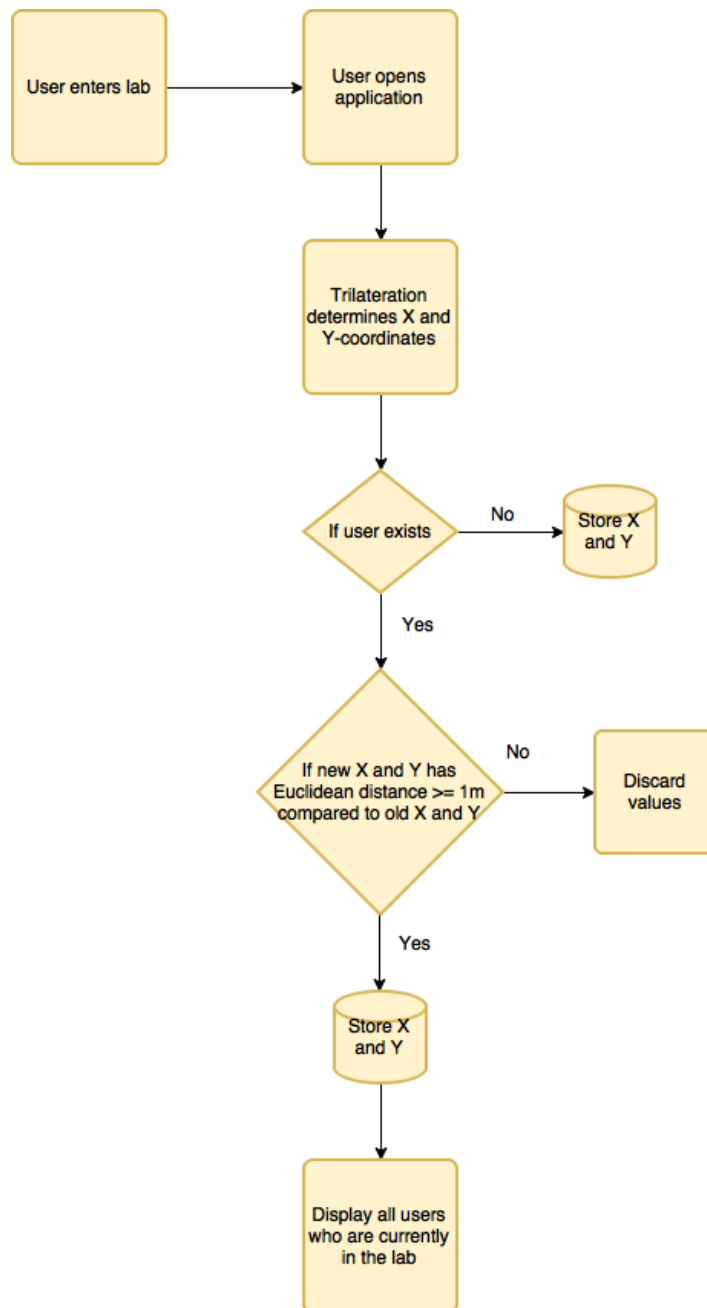


FIGURE 1: FLOWCHART FOR TRILATERATION

The flowchart for background monitoring is quite similar to that of trilateration. However, the key difference is that the flowchart for monitoring does not involve the process of the user opening the application, and it does not involve the process of checking for the previous coordinates stored in the central database. It simply checks if the user ID exists in the database, if it does, then the zone is updated and stored, and if the user ID does not exist, then the values are simply inserted into the database.

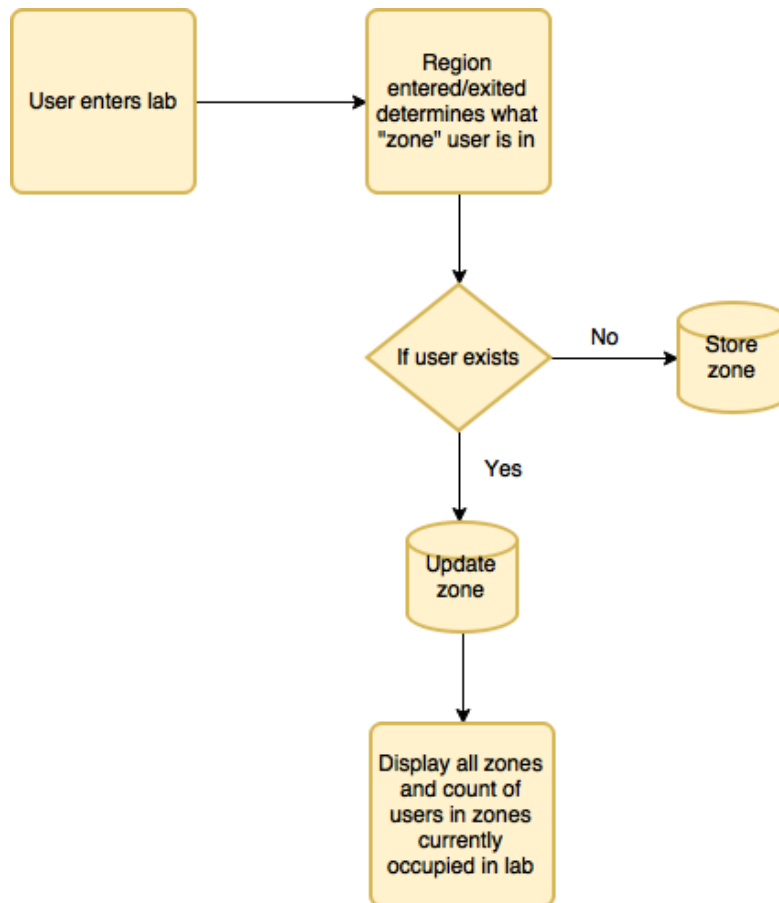


FIGURE 2: FLOWCHART FOR BACKGROUND MONITORING

#### 4.6 UML CLASS DIAGRAM

The class diagram below depicts the interactions of each entity with the other entities within the system along with each attribute that each entity consists of. The methods and attributes are expounded on throughout Section 5, starting at page 24 with the implementation describing in detail what each method aims to achieve. The class diagram also includes classes such as “BeaconAttributes” which are essentially made for defining preset coordinates that each beacon is placed at within the lab; this will also be expounded on in Section 5.

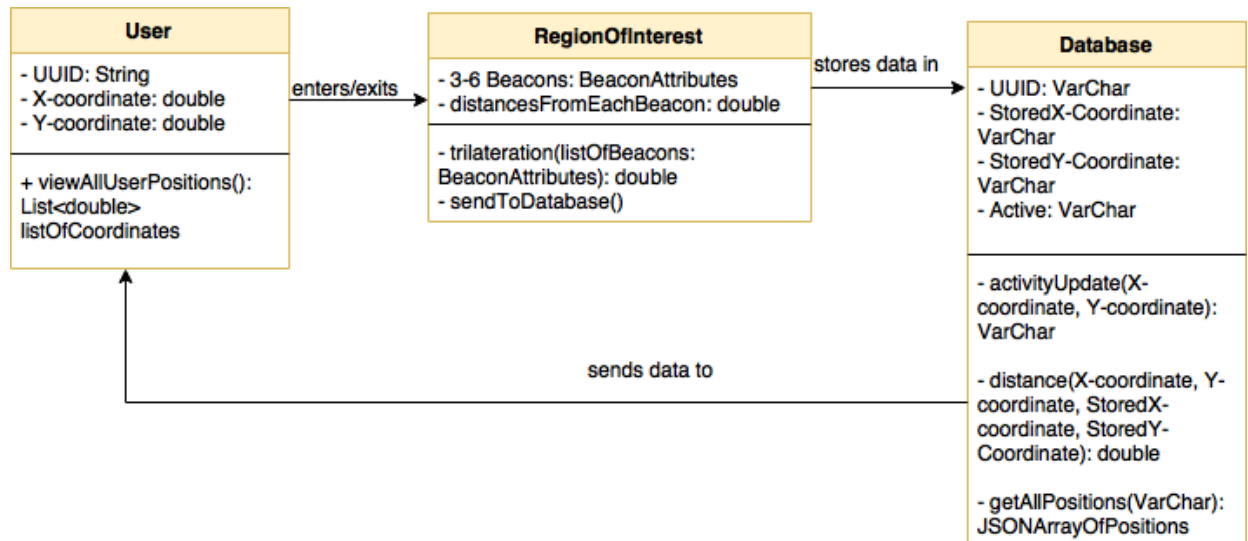


FIGURE 3: UML CLASS DIAGRAM FOR THE TRILATERATION SYSTEM

The entities in this class diagram are similar to those in the one above. However, the entities have different attributes and different methods according to the system. The methods and the attributes will also be expounded on in Section 5.

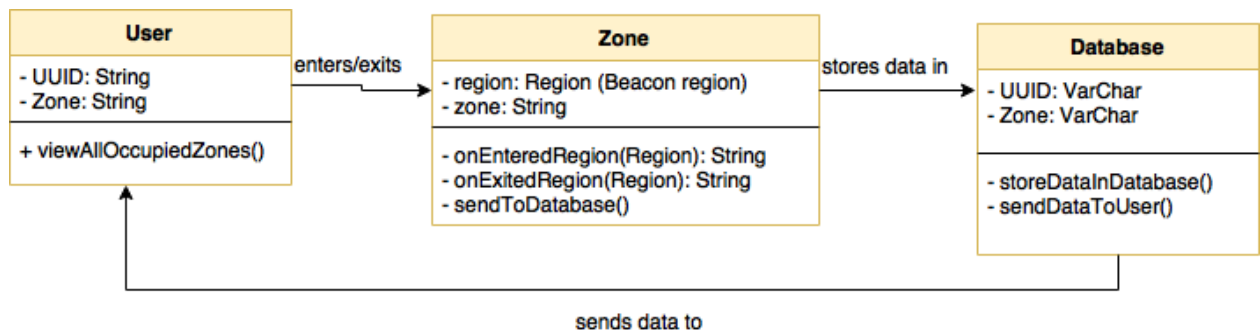


FIGURE 4: UML CLASS DIAGRAM FOR THE MONITORING SYSTEM



## 5. IMPLEMENTATION

As mentioned in the previous section, the 4 main components of the system are:

1. Beacon Plan Design
2. Android Application
3. Sending and retrieving data to and from the database
4. Displaying the positions on the front-end

This section will describe the implementation of the 4 main components along with a detailed description as to why these choices were made and the problems that were faced during this project during the implementation phase.

### 5.1 BEACON PLAN DESIGN

As mentioned in the Introduction, the location of interest for this project is the Cardiff University School of Computer Science Windows Lab 2. The dimensions of this lab were measured using a tape measure and were determined to be 12m in length and 6.2m in width. Thus, from the door to its opposite end by the windows measures 12m, and from the wall behind the computers to the opposite end where there are barricades preventing access to Lab 1 measures 6.2m.

In total, there are 6 beacons used for this project; 3 of the beacons are known as Estimote Proximity Beacons, and the other 3 beacons are known as Kontakt.io beacons. These beacons essentially host the same hardware and for the purposes of this project, they run the iBeacon protocol mentioned in Section 2.3, page 11. Each beacon can support a coverage of up to 140m. Thus, with respect to determining the ideal Beacon plan, the optimum coverages for the beacons would also be necessary. The beacons are only capable of having a coverage of 3m, 7m, 14m, 30m, 60m, 80m, 50m, and 140m in diameter. The coverage of these beacons are based on Broadcast (or Transmit) Power, which as its name suggests is the power at which each BLE beacon transmits its signal. The Broadcast Power ranges from -40 dBm to +4 dBm (dBm refers to the power ratio in decibels of the measure power referenced to 1 milliwatt - mW) for Proximity beacons; where the relationship between the Broadcast Power and the coverage of the beacon is directly proportional. Thus, a higher Broadcast Power results in a higher range<sup>(11)</sup>.

The distance from each beacon is estimated using a Received Signal Strength Indicator (RSSI). RSSI, as its name suggests, is an estimate of the strength of the signal at a certain distance away from the beacon. It is based on both Broadcast Power and Measured Power<sup>(12)</sup>. Measured Power is an indicator of the expected RSSI at a distance of 1m, which has been pre-calibrated and used to stabilize the RSSI received by the phone at any distance away from the beacon.

As mentioned in Section 3.1, page 15, there are three approaches to ranging. Hence, the plan for the approaches are different. The 3-beacon ranging approach has two plans to determine which positions are ideal for the three beacons. The 3-closest beacon approach and the 6-beacon approach follow the same plan. Each of these approaches for ranging and trilateration will be implemented, and their results would be analyzed to determine the optimum approach for foreground scanning/ranging. Furthermore, the implementation for each plan is also unique, and this would also be expounded on Section 5.1 from pages 28 – 37.

Initially, in the experimentation phase, 3 beacons' ranges were set to 14m to have them cover the entire lab. However, the values received from the beacons in the lab were far too inaccurate. For example, when standing 3m away from a beacon, the value received was approximately 5m. The cause of this has been determined to be the Broadcast Power. The relationship between the distance and its signal strength is an inverse-square relationship. Thus, if the distance away from the beacon is doubled, its signal strength decreases 4 times<sup>(9)</sup>. To solve this issue, the Broadcast Power was set to its highest at +4 dBm. Although this power covers past the range of the lab, a trade-off was made where if the user's position was determined to be within -1m to 7m for the x-coordinate and -1m to 13m for the y-coordinate, then the user can be considered to be within the lab. Once again, the system was given flexibility with respect of 2m above the ranges of values for each coordinate due to the fact that it is expected that interference of Bluetooth will affect the distance estimates between the phone and each beacon.

There are 3 different scenarios which have produced different Beacon plans:

- Beacon plans for just 3 beacons for trilateration
- Beacon plans for 6 beacons for trilateration
- Beacon plans for 6 beacons for background monitoring

Each plan has a different purpose, and a significant point to note is that the plans for trilateration are 2-dimensional; thus, the beacons are placed at a height where the difference between the height of the beacon and the height of the phone can be considered negligible. Therefore, placing the beacons on the roof cannot be considered given that the height will undoubtedly skew the distance values estimated by the Android phone.

#### 5.1.1 BEACON PLAN FOR RANGING (TRILATERATION)

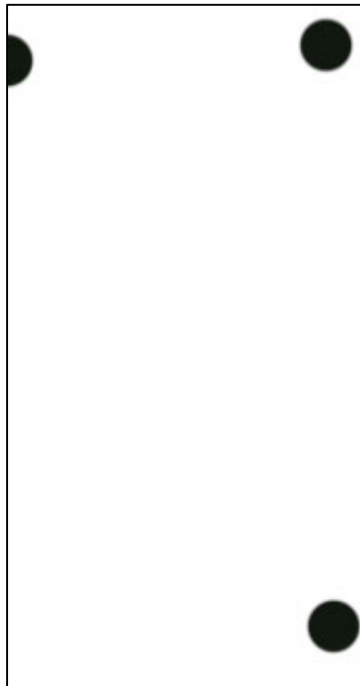


FIGURE 5: 3-BEACON PLAN FOR  
TRILATERATION

Figure 5 displays a scalar representation of the window lab, and the placements of 3 beacons within the labs used for testing the trilateration algorithm.



Figure 6: Alternate 3-beacon plan for trilateration

Figure 6 involves placing one of the beacons in a different position in order to determine whether it avoids more interference to provide a greater accuracy in results for trilateration.

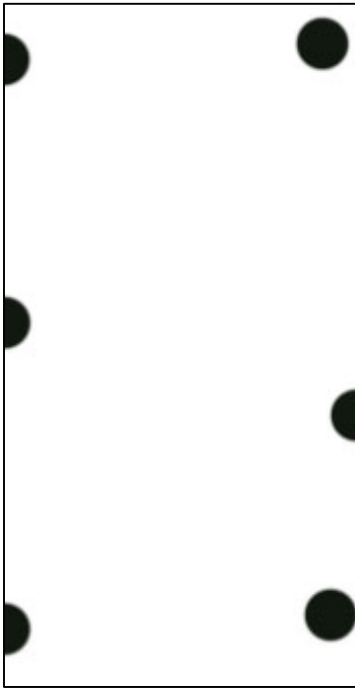


FIGURE 7: 6-BEACON PLAN FOR TRILATERATION

Figure 7 is used both for the approach where the closest 3 beacons are determined and the approach where all 6 beacons are used in determining the position of the individual in the labs.

To add more context to each design, the top of the design represents the side with the entrance to the Windows Lab 2. The right side of the design represents the side that separates the Windows Lab 2 from the Windows Lab 1.

#### 5.1.2 BEACON PLANS FOR MONITORING

The approach to this plan is slightly different to the approaches mentioned above. Initially, it can be assumed that within the lab, for practical purposes, users would like to know whether computers are occupied or not. Thus, it can be assumed that the point of interest within the labs would be near a computer. Based on this assumption, the plan made is placing three beacons on both tables. The first 3 beacons (beacon A – beacon C) covers a range of 6 computers, and the other 3 beacons (beacon D – beacon F) cover a range of 3 computers. This is because the larger of the 2 tables hosts twice the number of computers as the smaller of the two tables. All other areas within the labs that are not covered by the beacon are assumed to be not of interest.

The Broadcast Power for the beacons in this scenario were set to the lowest possible Broadcast Power, -30dBm, in order to achieve a range of 3m in diameter. This was done in order to achieve the highest precision in terms of area. Thus, the lower the area covered by the beacon, the more granular and detailed the feedback from the beacon. For example, if an individual has triggered the “onEnteredRegion” event of Beacon A, then it can be said the individual is currently in the area covered by Beacon A. Essentially, onEnteredRegion is exhibiting the event of passive scanning where a signal or advertisement packet is received from a beacon, and the application processes the packet in order to provide the appropriate response, which in this case, is placing a user into a particular pre-defined area within the lab based on the particular beacon’s placement in the lab.

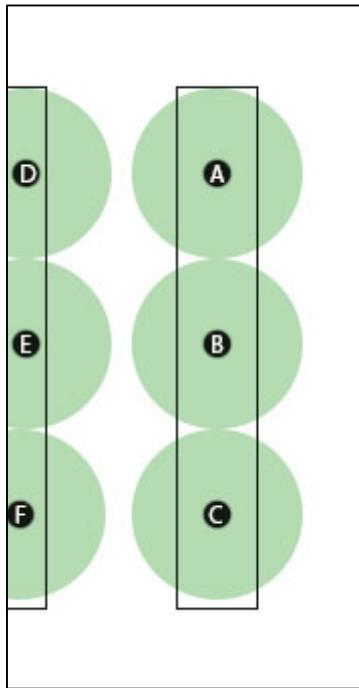


FIGURE 8: BEACON MONITORING PLAN

In this plan, the tables are displayed along with their ranges to show what area each beacon covers along with what zone each user is in. Elaborating on the assumptions mentioned above, the beacon plan assumes that if an individual is within region covered by Beacon A then a computer (along with that computer’s seat) is taken. Thus, depending on the count displayed by region A, the user knows that he/she can either sit in that region or move to another region of the lab.

## 5.2 ANDROID APPLICATION

There are two versions of the Android application, namely, the ranging/trilateration application and the monitoring application. Each application works in a different manner; however, each application interacts with the rest of the system in the same manner. The differences between each application will be expounded on in this section.

### 5.2.1 ANDROID APPLICATION FOR RANGING/TRILATERATION

The application integrates the Estimote SDK in order to interact with the Estimote and Kontakt.io beacons. The central aspect from the SDK that is used is the “BeaconManager” class. The class in this application allows the option of ranging beacons, which as mentioned in Section 3.1, page 15, scans for beacons in the foreground and filters them providing much more details about the beacons within range of the phone.

```
beaconManager = new BeaconManager(this);
beaconManager.setForegroundScanPeriod(600, 200);
beaconManager.setRangingListener(new BeaconManager.RangingListener() {
    @Override
    public void onBeaconsDiscovered(Region region, List<Beacon> list) {
        if (!list.isEmpty()) {
            for (int i = 0; i < list.size(); i++) {
                storedDistances.get(getKey(list.get(i))).offer(Utils.computeAccuracy(list.get(i)));
            }
        }
    }
});
```

Initially, in the application, a new instance of the BeaconManager class is created and declared. The attribute of ForegroundScanPeriod refers to how long to scan for (600ms) and after how long to scan for (200ms). These values were chosen to be ideal due to the fact that the beacons’ advertising interval was set to 201ms (milliseconds). Thus, the scan period is 3 times as long to ensure that the beacon’s packets are received by the phone.

The method of “setRangingListener()” is called in order to complete a set of actions based on when the phone receives signals from the BLE beacons in range. The BLE beacons in range are received in the form of a List. Thus, the first check is to determine whether the beacon list is empty; if it is not empty, then the distance from each beacon to the phone is calculated and estimated using the “computeAccuracy” method. As mentioned previously, this distance estimate is based on the RSSI of the Bluetooth signal received from the beacon. The variable “storedDistances” is an instance of a HashMap, which has the key of String and the value of “DistanceQueue”, which is a class created in this application. The

String value in storedDistances refers to the “beaconKey”, which is simply the major and minor values. This can be thought of as the BeaconID in order to distinguish each beacon from the other.

Essentially, the DistanceQueue class is a queue that has a pre-specified limit. In the case of all the ranging applications, the limit is set to 100. This is because the application assumes that the user is moving, and when the user moves from one location to another, the most recent values of the perceived distances are the most significant. Thus, with an advertising interval of 201ms, and a limit of 100, the DistanceQueue reaches this limit after 20,100ms, which is equivalent to 20.1 seconds. This is a reasonable value when determining the average of the user’s most recent position. In addition, when the queue reaches its limit, the element that was added in first is removed in order to make room for the new element to be added. This is done using a method created in the class known as “offer()” as seen in the following code snippet.

```
public boolean offer(double element) {
    if (queue.size() < limit) {
        queue.addFirst(element);
    } else {
        queue.removeLast();
        queue.addFirst(element);
    }
    return true;
}
```

The class also consists of methods such as “findAverage()”, “getStandardDeviation()”, and “findAccurateAverage()”, which are discussed further in Section 6.4, page 52. Essentially, the class uses all the values stored in the queue to determine an average, and then the standard deviation is determined based on the average value. The standard error of the average is then computed based on the standard deviation. Any values that are stored within the queue that are above or below 1.645 times the standard error are not included in the calculations of the “accurate” average. Essentially, this is done to avoid outliers in distance values estimated by the phone from each beacon, but the reasoning behind the choice of these values are also further discussed in Section 6.4; theoretically, this should result in a more stable distance value.

The “trilateration()” method is then called. This method was implemented based on the following mathematical equations <sup>(13)</sup>:



1.  $(x - x_a)^2 + (y - y_a)^2 = r_a^2$
2.  $(x - x_b)^2 + (y - y_b)^2 = r_b^2$
3.  $(x - x_c)^2 + (y - y_c)^2 = r_c^2$

Where  $(x_a, y_a)$ ,  $(x_b, y_b)$ ,  $(x_c, y_c)$  are the known coordinates of where the beacons are positioned, and  $r_a$ ,  $r_b$ , and  $r_c$  are the known distances from each beacon;  $x$  and  $y$  are the coordinates of the position of the user.

The equations can then be expanded eventually leaving the following equations:

4.  $v_a = (r_b^2 - r_c^2 + x_c^2 - x_b^2 + y_c^2 - y_b^2)/2.0$
5.  $v_b = (r_b^2 - r_a^2 + x_a^2 - x_b^2 + y_a^2 - y_b^2)/2.0$

Finally,  $x$  and  $y$  were determined using the following equations:

6.  $y = (v_b * (x_c - x_b) - v_a * (x_a - x_b)) / (y_a - y_b)(x_c - x_b) - (y_c - y_b)(x_a - x_b)$
7.  $x = (v_a - y * (y_c - y_b)) / (x_c - x_b)$

The equations were then written in Java and implemented. The Java implementation was then tested with hard-coded variables in order to verify that the implementation written will result in accurate values when receiving distance estimations from Bluetooth beacons. This is seen in Section 6.1 page 46.

#### 5.2.1.1 TRILATERATION FOR 3 BEACONS

---

Trilateration for simply 3 beacons is implemented exactly like the one implemented above, where 3 arguments from the “BeaconAttributes” class are supplied to the function in order to return the position instance of the BeaconAttributes class. Referring back to Figure 5, the BLE beacon Blueberry (names seen below) would be placed on the right corner (with the coordinates seen below), Mint would be placed on the bottom right corner, and Ice would be placed on the top left corner. According to this placement, the origin (the point at which the graph is: 0, 0) of the axes is the top right corner. The reason this was chosen was because the values for each user would always be positive in both the x-axis and the

y-axis. The origin would remain at the top right corner for each of the trilateration applications as well.

```
private BeaconAttributes finalPosition;
private static BeaconAttributes ba1 = new BeaconAttributes(0.61, 0.74); //Blueberry
private static BeaconAttributes ba2 = new BeaconAttributes(0.47, 10.91); //Mint
private static BeaconAttributes ba3 = new BeaconAttributes(6.2, 0.98); //Ice

public void initializeBeaconsDistances() {
    storedDistances.put("207:1", new DistanceQueue(100));
    storedDistances.put("207:2", new DistanceQueue(100));
    storedDistances.put("207:3", new DistanceQueue(100));
}

static {
    Map<String, BeaconAttributes> keyWithBeacons = new HashMap<>();
    keyWithBeacons.put("207:1", ba1);
    keyWithBeacons.put("207:2", ba2);
    keyWithBeacons.put("207:3", ba3);
    BEACON_ATTRIBUTES = Collections.unmodifiableMap(keyWithBeacons);
}

private String getKey(Beacon beacon) {
    String beaconKey = String.format("%d:%d", beacon.getMajor(), beacon.getMinor());
    return beaconKey;
}

beaconManager = new BeaconManager(this);
beaconManager.setForegroundScanPeriod(600, 200);
beaconManager.setRangingListener(new BeaconManager.RangingListener() {
    @Override
    public void onBeaconsDiscovered(Region region, List<Beacon> list) {
        if (!list.isEmpty()) {
            for (int i = 0; i < list.size(); i++) {
                //store the distance values from each beacon with its respective beaconKey
                storedDistances.get(getKey(list.get(i))).offer(Utils.computeAccuracy(list.get(i)));
            }

            //Trilateration for 3 beacons
            ba1.setDistance(storedDistances.get("207:1").findAccurateAverage());
            ba2.setDistance(storedDistances.get("207:2").findAccurateAverage());
            ba3.setDistance(storedDistances.get("207:3").findAccurateAverage());

            finalPosition = trilateration(ba1, ba2, ba3);
            positionX.offer(finalPosition.getX());
            positionY.offer(finalPosition.getY());
            sendData();
        }
    }
});
```

The position of the beacons are predetermined using a measuring tape to measure the distance between the beacons and the origins of each axis in the lab and are stored as instances of BeaconAttributes. The accurate average of the distances from each beacon is set as the distance between the phone and each beacon, which is then used by the trilateration function to yield the final position. The “getKey()” method is called to return the right key and right BeaconAttributes instances which are stored in a final unmodifiable Collections instance known called “BEACON\_ATTRIBUTES”.

“positionX” and “positionY” are also instances of the DistanceQueue class for the values of x and y, respectively. The position that gets sent to the server is the accurate average of positionX and positionY. The limit for these DistanceQueues are also set to 100 to allow a reasonable range of values in order to find the accurate average of these values. These DistanceQueues also reach their limit after 20.1 seconds. The UUID and the accurate averages of positionX and positionY are sent to the database using the “sendData()” method, which is essentially a method for a HTTPRequest to a PHP script which handles the data being sent over.

#### 5.2.1.2 TRILATERATION FOR 6 BEACONS

The 6-beacon trilateration approach involves placing all six beacons within the Windows Lab 2 with the maximum range of 70m as seen the plan in Figure 7. Given that trilateration is done with three known points, to complete trilateration with six beacons, it would involve dividing the values received from all six beacons into subsets of three. Trilateration would be performed on each of the subsets, and the average of the positions determined by the trilateration of each of the subsets would result in the accurate coordinates reflecting the position of the user.

```
private static final Map<String, BeaconAttributes> BEACON_ATTRIBUTES;
private BeaconAttributes finalPosition;

private BeaconManager beaconManager;
private Region region;

private static BeaconAttributes ba1 = new BeaconAttributes(0.61, 0.74); //Blueberry
private static BeaconAttributes ba2 = new BeaconAttributes(0.47, 10.91); //Mint
private static BeaconAttributes ba3 = new BeaconAttributes(6.2, 0.98); //Ice
private static BeaconAttributes ba4 = new BeaconAttributes(0, 7.22); //Glag
private static BeaconAttributes ba5 = new BeaconAttributes(6.2, 10.57); //8wHr
private static BeaconAttributes ba6 = new BeaconAttributes(6.2, 5.6); //iTTA
Map<String, DistanceQueue> storedDistances = new HashMap<String, DistanceQueue>();
private PositionQueue positionX = new PositionQueue(100);
private PositionQueue positionY = new PositionQueue(100);
private List<BeaconAttributes[]> allCombinationsOfBeacons = new ArrayList<BeaconAttributes[]>();
private BeaconAttributes[] allBeacons = new BeaconAttributes[6];
```

```

static {
    Map<String, BeaconAttributes> keyWithBeacons = new HashMap<>();
    keyWithBeacons.put("207:1", ba1);
    keyWithBeacons.put("207:2", ba2);
    keyWithBeacons.put("207:3", ba3);
    keyWithBeacons.put("207:4", ba4);
    keyWithBeacons.put("207:5", ba5);
    keyWithBeacons.put("207:6", ba6);
    BEACON_ATTRIBUTES = Collections.unmodifiableMap(keyWithBeacons);
}

public void initializeBeaconsDistances() {
    storedDistances.put("207:1", new DistanceQueue(100));
    storedDistances.put("207:2", new DistanceQueue(100));
    storedDistances.put("207:3", new DistanceQueue(100));
    storedDistances.put("207:4", new DistanceQueue(100));
    storedDistances.put("207:5", new DistanceQueue(100));
    storedDistances.put("207:6", new DistanceQueue(100));
}

beaconManager = new BeaconManager(this);
beaconManager.setForegroundScanPeriod(600, 200);
beaconManager.setRangingListener(new BeaconManager.RangingListener() {
    @Override
    public void onBeaconsDiscovered(Region region, List<Beacon> list) {
        if (!list.isEmpty()) {
            for (int i = 0; i < list.size(); i++) {
                //store the distance values from each beacon with its respective beaconKey
                storedDistances.get(getKey(list.get(i))).offer(Utils.computeAccuracy(list.get(i)));
            }

            //Trilateration for all 6 beacons
            if (list.size() == 6) {
                for (int j = 0; j < list.size(); j++) {
                    //array consisting of all 6 beacons (can't be changed unlike arrayList)
                    allBeacons[j] = getBeaconPosition(list.get(j));
                    //set distances of each element in the array to be the accurateAverage of the DistanceQueue in storedDistances
                    allBeacons[j].setDistance(storedDistances.get(getKey(list.get(j))).findAccurateAverage());
                }
                //find all combinations of 3 beacons from 6, should be 20 combinations
                findAllSubsets(3, allBeacons);
                for (BeaconAttributes[] subset : allCombinationsOfBeacons) {
                    finalPosition = trilateration(subset);
                    //don't put in Nan values into DistanceQueue
                    if (!(Double.isNaN(finalPosition.getX()))) {
                        positionX.offer(finalPosition.getX());
                    }
                    if (!(Double.isNaN(finalPosition.getY()))) {
                        positionY.offer(finalPosition.getY());
                    }
                }
            }
            sendData();
        }
    }
}

```

```

private void findAllSubsets(int r, BeaconAttributes[] beaconList) {
    int[] indices = new int[r];

    if (r <= beaconList.length) {
        for (int i = 0; (indices[i] = i) < r - 1; i++) ;
        allCombinationsOfBeacons.add(getSubset(beaconList, indices));
        while(true) {
            int i;
            for (i = r - 1; i >= 0 && indices[i] == beaconList.length - r + i; i--) ;
            if (i < 0) {
                break;
            } else {
                indices[i]++;
                for (++i; i < r; i++) {
                    indices[i] = indices[i - 1] + 1;
                }
                allCombinationsOfBeacons.add(getSubset(beaconList, indices));
            }
        }
    }
}

BeaconAttributes[] getSubset(BeaconAttributes[] input, int[] subset) {
    BeaconAttributes[] result = new BeaconAttributes[subset.length];
    for (int i = 0; i < subset.length; i++) {
        result[i] = input[subset[i]];
    }
    return result;
}

```

The 6-beacon trilateration is set up initially the same as the 3-beacon trilateration and all the distances received by the phone from all the relevant beacons are then stored in the storedDistances HashMap. The application checks if there are signals being received from all six beacons. The beacons are then stored in an array – allBeacons, an array of BeaconAttributes – which cannot be manipulated unlike an ArrayList. All values in the array are then set with their respective distances. Each subset is determined using the “findAllSubsets()” method <sup>(14)</sup>. The subsets of the array are then stored in an ArrayList, which is then iterated through to perform trilateration on each subset. Finally, the respective coordinates of trilateration on each subset is stored in the positionX and positionY DistanceQueues where the accurate average is once again sent to the database.

#### 5.2.1.3 TRILATERATION FOR 3-CLOSEST BEACONS

The idea behind this approach is to determine the three closest beacons to the user in order to perform trilateration. This is done to achieve more accurate values because as mentioned in Section 5.1, page 25, the distance and signal strength have an inverse-square relationship. Thus, the three closest beacons will perhaps provide the three best distance values in order to determine the user’s position.

The set up to this approach is identical to the one above. The only difference is the main

code, which is displayed in the following code snippet. The storedDistances values are compared by their accurate averages and sorted in ascending order. The top three values from the sorted HashMap are then returned in the “getSortedDistances()” method, which are used for trilateration to determine the positions.

```
//Closest Beacons
List<Entry<String, DistanceQueue>> sortedDistances = getSortedDistances(storedDistances, 3);
finalPosition = trilateration(sortedDistances);

if(!(Double.isNaN(finalPosition.getX())) {
    positionX.offer(finalPosition.getX());
}
if(!(Double.isNaN(finalPosition.getY())) {
    positionY.offer(finalPosition.getY());
}
sendData();

//Comparator class for comparing the accurate averages
class DistanceQueueComparator implements Comparator<Map.Entry<String, DistanceQueue>>
{
    @Override
    public int compare(Map.Entry<String, DistanceQueue> o1, Map.Entry<String, DistanceQueue> o2) {
        return o1.getValue().findAccurateAverage() > o2.getValue().findAccurateAverage() ? -1 : 1;
    }
}

//method to return the top three values of the sorted distances
public List<Entry<String, DistanceQueue>> getSortedDistances(Map<String, DistanceQueue> map, int n) {
    Set<Map.Entry<String, DistanceQueue>> entrySet = map.entrySet();
    PriorityQueue<Entry<String, DistanceQueue>> topThree = new PriorityQueue<Entry<String, DistanceQueue>>(n, new DistanceQueueComparator());
    for(Entry<String, DistanceQueue> entry: entrySet){
        topThree.offer(entry);
        while(topThree.size() > n){
            topThree.poll();
        }
    }
    List<Entry<String, DistanceQueue>> result = new ArrayList<Map.Entry<String, DistanceQueue>>();
    while(topThree.size() > 0){
        result.add(topThree.poll());
    }
    return result;
}
```

The most significant detail of the approaches for this application is that ranging cannot occur in the background. Thus, there is a major lack of pervasiveness with the application. The application assumes that the user will always take his/her phone out for the phone to receive the signals from the beacon and achieve trilateration to determine the individual’s position, which is then sent to the server.

### 5.2.2 ANDROID APPLICATION FOR BACKGROUND MONITORING

The application for background monitoring takes a completely different approach than the one for ranging. The application is much different in approach in that it determines which beacon is sending a signal to the application. Subsequent to determining which beacon is sending a packet, it simply associates the packet with a predefined zone. A zone is essentially a particular region in the lab that is covered by the beacon’s coverage. This is depicted in Figure 8 of Section 5.1, page 30. The most significant aspect of this application

is that it works in the background. Thus, it is pervasive despite not being able to pinpoint an individual's location, it is able to display the user's area relative to the lab's area.

```
private BeaconManager beaconManager;
private Context context;
private String zone;
private String UID;
private static final Map<Integer, String> ZONE_MAP;

private Region region1 = new Region("blueberry", UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), 207, 1);
private Region region2 = new Region("mint", UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), 207, 2);
private Region region3 = new Region("ice", UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), 207, 3);
private Region region4 = new Region("glag", UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), 207, 4);
private Region region5 = new Region("8whr", UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), 207, 5);
private Region region6 = new Region("iita", UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), 207, 6);

static{
    HashMap<Integer, String> zoneMap = new HashMap<Integer, String>();
    zoneMap.put(1, "A");
    zoneMap.put(2, "B");
    zoneMap.put(3, "C");
    zoneMap.put(4, "D");
    zoneMap.put(5, "E");
    zoneMap.put(6, "F");
    ZONE_MAP = Collections.unmodifiableMap(zoneMap);
}

public BeaconManagerProximity(Context inContext, String inUID){
    context = inContext;
    UID = inUID;
    beaconManager = new BeaconManager(inContext);
    beaconManager.setMonitoringListener(new BeaconManager.MonitoringListener() {
        @Override
        public void onEnteredRegion(Region region, List<Beacon> list) {
            //Used for debugging to see when enteredregion is triggered
            Log.d("BeaconManager", "EnteredRegion: " + region.getIdentifier());

            zone = ZONE_MAP.get(region.getMinor());
            sendData();
        }
        @Override
        public void onExitedRegion(Region region) {
            //used for debugging to see when exitedregion is triggered
            Log.d("BeaconManager", "ExitedRegion: " + region.getIdentifier());
            zone = "0";
            sendData();
        }
    });
}

public void startMonitoring(){
    beaconManager.connect(new BeaconManager.ServiceReadyCallback(){
        @Override
        public void onServiceReady(){
            beaconManager.startMonitoring(region1);
            beaconManager.startMonitoring(region2);
            beaconManager.startMonitoring(region3);
            beaconManager.startMonitoring(region4);
            beaconManager.startMonitoring(region5);
            beaconManager.startMonitoring(region6);
        }
    });
}
```

The information above is stored within a class known as BeaconManagerProximity. The regions and their major and minor values are defined along with the “ZONE\_MAP” defining which zone each minor value belongs to. The zones can then be retrieved when a



“region” is picked up by the application via a beacon signal. The zone is then sent to the central database along with the user ID (UID). The “startMonitoring()” method is also defined in this class to start listening for signals from each of the regions defined above.

```
@Override
public void onCreate() {
    super.onCreate();

    UID = UUID.randomUUID().toString();
    mPreferences = getApplicationContext().getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    String storedUID = mPreferences.getString("UID", "");
    if (storedUID.equals("")) {
        UID = UUID.randomUUID().toString();
        mPreferences.edit().putString("UID", UID).commit();
    } else {
        UID = storedUID;
    }
}

public void determinePosition(){
    BeaconManagerProximity bmp = new BeaconManagerProximity(this, UID);
    bmp.startMonitoring();
}
```

The application also defines an application class, BeaconApplication, to initialize and store key information such as the UID. Furthermore, the method of determinePosition() is defined to create an instance of the BeaconManagerProximity class and starts listening for beacon signals via the startMonitoring() method. Given that the startMonitoring method is called in the Application class, the phone continues listening for beacon signals/advertisement packets because it is not an “Activity” class, which is eventually killed off by the phone when not in use. Nonetheless, the MainActivity creates an instance of the BeaconApplication class and simply initiates the monitoring when the application is first opened. Given that the granular detail in ranging is not required, this application can continue scanning for beacons’ packets in the background without an extremely negative impact on performance and battery life of the phone.

### 5.3 SENDING DATA TO THE DATABASE

Sending data to the database is done via PHP scripts stored on a central server. The scripts receive “Post” variables from the application and connect to a MySQL database to store the data of the individual user associating the information with the user’s ID.

#### 5.3.1 SENDING TRILATERATION DATA TO THE DATABASE



```

function ActivityUPDATE($X_NEW_Coor, $Y_NEW_Coor,$UID,$con){
    #Checks if user is within the range of the labs and sets active to 1
    if (((($X_NEW_Coor >= -1) and ($X_NEW_Coor <= 7.0)) and (($Y_NEW_Coor >= -1) and ($Y_NEW_Coor <= 13.0)))){
        $query = mysqli_query($con, "UPDATE `Active_Users` SET `Active`=1 WHERE UID='$UID'");
    }
    else {
        #User not in labs, so set active to 0
        $query = mysqli_query($con, "UPDATE `Active_Users` SET `Active`=0 WHERE UID='$UID'");
    }
}

function Distance($X_NEW_Coor, $Y_NEW_Coor,$X_DB_Coor,$Y_DB_Coor,$UID,$con){
    $distance = sqrt( pow(($X_DB_Coor-$X_NEW_Coor),2) + pow(($Y_DB_Coor-$Y_NEW_Coor),2) );
    echo $distance;
    if ($distance >= 1.0){
        #User moved a big distance, so need to update that in db
        $query = mysqli_query($con, "UPDATE `Active_Users` SET `X`=$X_NEW_Coor , `Y`=$Y_NEW_Coor WHERE UID='$UID'");
    }
    else {
    }
}

if(isset($_GET['UID'])){
    $UID = mysqli_real_escape_string($con, $_GET['UID']);
    $X_NEW_Coor = mysqli_real_escape_string($con, $_GET['X']);
    $Y_NEW_Coor = mysqli_real_escape_string($con, $_GET['Y']);
    $USER_CHECK = mysqli_query($con, "SELECT * FROM Active_Users WHERE UID='$UID'");
    #Checks if user exists
    if ($USER_CHECK->num_rows){
        $ActiveUSER = $USER_CHECK->fetch_array();

        #User Exist In the DB, checks if user has moved a lot, and checks if activity needs to be updated
        Distance($X_NEW_Coor, $Y_NEW_Coor,$X_DB_Coor,$Y_DB_Coor,$UID,$con);
        ActivityUPDATE($X_NEW_Coor, $Y_NEW_Coor,$UID,$con);
    }
    else{
        #User does not exist, simply insert
        $query = mysqli_query($con, "INSERT INTO `Active_Users` (`UID`, `X`, `Y`) VALUES ('$UID','$X_NEW_Coor','$Y_NEW_Coor')");
        Distance($X_NEW_Coor, $Y_NEW_Coor,$X_DB_Coor,$Y_DB_Coor,$UID,$con);
        ActivityUPDATE($X_NEW_Coor, $Y_NEW_Coor,$UID,$con);
    }
}

```

The script initially checks if the user exists. If the user does exist, the script checks if the user's position needs to be updated. Given that it is expected that the coordinates will be constantly changing due to the fact that Bluetooth is extremely prone to interference, if the user's new coordinates result in a distance of greater than 1m (using the Euclidean distance formula and the user's previous coordinates for comparison), then the values are updated.

Furthermore, the application must be constantly ranging because if the user leaves the labs, and the phone is still receiving a signal from ranging for the beacons, then the user may not necessarily be in the lab given that the beacons' ranges are set to that of beyond the lab. Thus, the script also constantly checks if the user is in the labs by checking if the coordinates are in between the range of -1 and 7 for the x-coordinate and -1 and 13 for the y-coordinate. Once again, due to Bluetooth being prone to interference, the ranges are extended to + or - 1 meter for each axis in the labs (where the dimensions of the labs are 6.2 for width (related to the x-axis) and 12 for the length (related to the y-axis). If the user

is within this range, then the user is considered to be “active” in the labs, and thus, the variable “active” is updated to 1 in the database.

### 5.3.2 SENDING MONITORING DATA TO THE DATABASE

```
function ActivityUPDATE($ZONE,$UID,$con){
    $query = mysqli_query($con, "UPDATE `click` SET `zone`='$ZONE' WHERE id='$UID'");
    echo "UPDATED";
}

if(!$con){
    die ("Failed to connect" . mysqli_connect_error());
}
else{
    //if connection worked
    if(isset($_POST['UID'])){
        $UID = mysqli_real_escape_string($con, $_POST['UID']);
        $ZONE = mysqli_real_escape_string($con, $_POST['ZONE']);

        #Checks if user exists
        $USER_CHECK = mysqli_query($con, "SELECT * FROM click WHERE id='$UID'");
        if ($USER_CHECK->num_rows){
            $ActiveUSER = $USER_CHECK->fetch_array();

            #User exists, so need to update their zone
            ActivityUPDATE($ZONE,$UID,$con);
        }
        else{
            #User does not exist, so simply insert
            $query = mysqli_query($con, "INSERT INTO `click` (`id`, `zone`) VALUES ('$UID','$ZONE')");
            ActivityUPDATE($ZONE,$UID,$con);
        }
    }
}
```

Sending monitoring data to the database is quite simple. The script checks if the user exists via the user ID; if the user exists, the user’s “zone” is updated, and if the user does not exist, the user’s userID along with the user’s current zone is inserted.

## 5.4 RETRIEVING DATA FROM THE DATABASE AND DISPLAYING POSITIONS ON THE FRONT-END

With respect to displaying positions on the front-end, the lab’s dimensions are drawn in scale to pixels and the image is set as the background. A CSS file includes the settings for the background along with the images used to represent different areas of the lab for monitoring and the images used to represent a user’s location for foreground scanning.

```

#Called when the Ajax requests it
$query = "SELECT * FROM Active_Users WHERE Active='1'";
$results = mysqli_query($con,$query);

$x_coordinate = 0;
$y_coordinate = 0;

$array = array();

if(mysqli_num_rows($results) > 0){
    while ($row = mysqli_fetch_array($results,MYSQLI_ASSOC)) {
        $x_coordinate = $row['X'];
        $y_coordinate = $row['Y'];

        #Get coordinates in pixels
        $x_coordinate = $x_coordinate * 100;
        $y_coordinate = $y_coordinate * 100;

        array_push($array, array(
            'X' => $x_coordinate,
            'Y' => $y_coordinate
        ));
    }
    #Return results in numerical format in JSON array
    echo json_encode($array);
}

```

For the purposes of trilateration, the coordinates are retrieved from the database where the all users' "Active" attribute is set to "1". The scalar representation of the lab is 620 pixels by 1200 pixels. Thus, the coordinates stored in the database are multiplied by 100. These coordinates are then encoded in a JSON array to be processed by the JavaScript script.

```

$(document).ready(function() {
    $.displayActivity = function(){
        var request = new XMLHttpRequest();
        request.open('GET', 'http://localhost/dev/FrontEndTest/position.php');
        request.onload = function(){
            var data = JSON.parse(request.responseText);
            loadData(data);
        };
        request.onerror = function() {
            console.log("Connection error");
        };
        request.send();
    }
});

function loadData(data){
    for(i = 0; i < data.length; i++){
        //Create new div for positioning element for every "Active" user in the lab
        var element = document.createElement("div");
        element.style.position = "absolute";

        //Need to position them according to the origin, which is the top right corner
        element.style.right = data[i].X + 'px';
        element.style.top = data[i].Y + 'px';
        element.id = "heatmap-container-user-div";
        document.getElementById('container').appendChild(element);
    }
}

```

Once the coordinates are retrieved by the PHP script, the JavaScript file retrieves the coordinates by sending a XMLHttpRequest to the file to retrieve the link. The results of the

JSONArray are then parsed and a new user icon is generated for each user as seen in the “loadData()” function. The user icon is then positioned relative to the origin to depict where the user is currently located in the lab.

```
<script type="text/javascript">

    function codeAddress() {
        $.displayActivity();
    }
    //When screen loads, call function $.displayClicks in heatmap.js
    window.onload = codeAddress;

    //Window is reloaded every 5 seconds to check if new user is in
    window.setInterval(function(){
        location.reload();
    }, 5000);

</script>
```

Another PHP file includes a JavaScript function which reloads the window and calls the function every 5000ms (5 seconds). This essentially checks for any updates to the database every 5 seconds and updates the front-end accordingly. Thus, if a new user enters within the labs within the 5 seconds and the coordinates determined are within the dimensions of the lab, the JavaScript function would get called and to update the front-end with another user icon.

With respect to displaying the user positions using background monitoring, a similar sort of process is done. A PHP script retrieves all users that are in zones between A - F, then the coordinates of these zones are predetermined using the image displayed as Figure 8. These predetermined coordinates are hard-coded into the PHP script, and whenever a zone has at least one user, the count for the number of distinct users in each zone is retrieved. The JavaScript in the main page is called every 5000ms (5 seconds) in order to call the PHP script to retrieve values from the database again. Once these values have been retrieved by the PHP script, the values are sent to the body of the main page, which position the circles depicting the zones according to the hard-coded variables.

## 5.5 PROBLEMS THROUGHOUT IMPLEMENTATION

Throughout the implementation of the system, there were many problems faced. The first and foremost problem was foreground scanning/ranging vs. background monitoring. The foreground scanning/ranging method for implementing provides granular details for each beacon, which allows the application to use the RSSI of the signal that is sent as packets from each beacon to estimate a distance. However, the only way for the application to receive these packets is if the application is running in the foreground due to active scanning. Thus, this solution is not pervasive in nature and relies on the user taking out his/her phone to have the application open on the foreground. Having said that, background monitoring is not able to extract finer details given that the application exhibits passive scanning; thus, the background monitoring application cannot perform trilateration for positioning an individual. Therefore, this application trades off precision for pervasiveness. Nonetheless, each beacon can be placed in different sections of the lab as seen in Figure 8 on page 30 in order to display where in the labs the individual is currently placed in.

Another major issue in the implementation was the large inaccuracy of the distance estimates received from each BLE beacon. As mentioned, the distance estimates are determined through the signal strength received by the phone, and given that metal surfaces and bodies of water can fluctuate the signal strength, the distance estimated by the application would therefore be also be fluctuated causing inaccuracy. To combat this inaccuracy, the first iteration of the application simply involved using the averages of the estimated distances from each BLE beacon. The averages of the estimated beacons were then compared in different beacon scenarios from two different settings for three beacons along with trying values received from all six beacons and finally trying to use three of the closest out of all six beacons for trilateration. However, this was also not enough to provide an accurate position of an individual in the lab. These tests were also conducted using only one phone, a Oneplus One. Therefore, a more statistical approach was taken where the standard deviation of the average was determined, and the standard error of the mean was determined using the standard deviation. This technique is expounded on in Section 6.4, page 49. Initially, again, the second iteration of the application that incorporated the statistical techniques were tested using only the Oneplus One. However, it was theorized that potentially, the Oneplus One's Bluetooth receiver is not adequate. Thus, it was necessary to test this in a more realistic scenario with a few other phones in order to determine whether the BLE beacons or the phone's receiver were providing inaccurate

results. Each of the tests conducted are elaborated on in Section 7, page 52, and the test conditions can be found on page 55.

Another major issue was the time constraints. In developing the overall system, it was difficult to prioritize achieving accurate values for trilateration or produce an overall application that is able to accurately display the position of an individual within the lab. In the end, a compromise on both priorities was necessary, where there were consistent tests conducted to determine whether, theoretically, each approach in terms of beacon placement along with each statistical technique produced accurate results or not. This took away from the time for developing a full Android application. The focus was then shifted to developing a full system where values were sent to the database and would be displayed in some form of front-end. The front-end, chosen due to time constraints, was built on a web browser rather than a phone. This prevented development on producing an algorithm to provide accurate results. Furthermore, another issue with time constraints was the development of the background monitoring application. Given that the testing was prioritized for this application, the application is based on an Estimote template application. The essential functionality was then added to the application for the purposes of this project.

Due to the issue of the time constraint, the system currently interacts with the following components in the following manner:

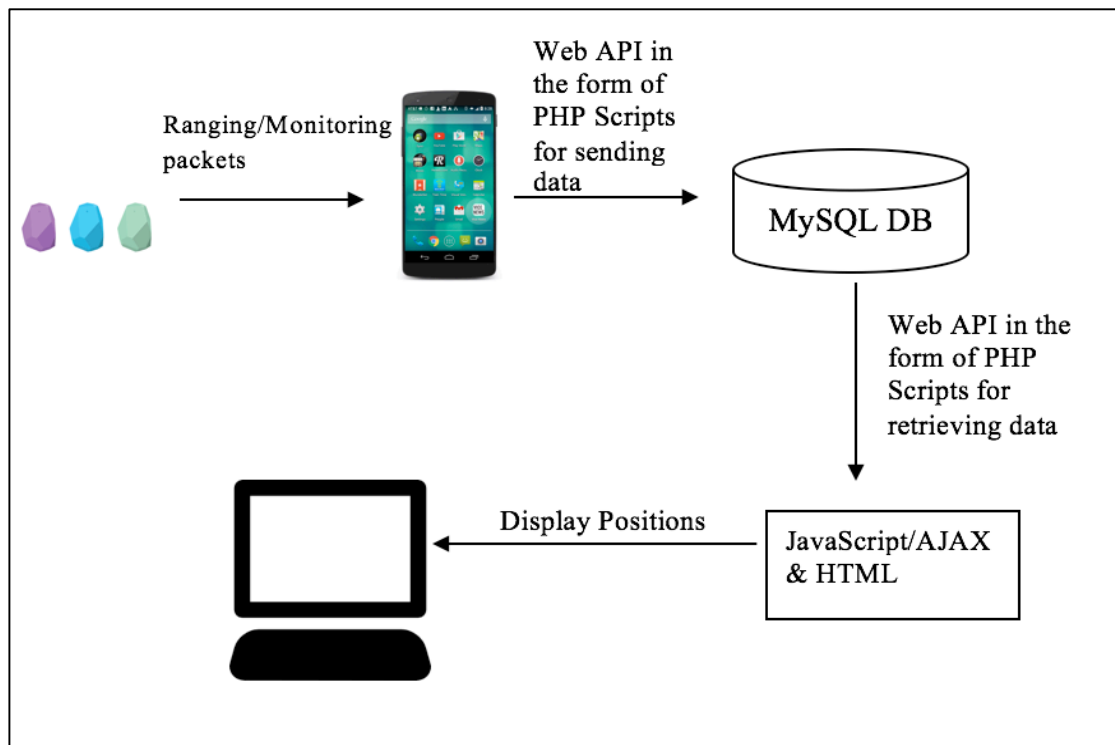


FIGURE 9: INTERACTION OF ALL COMPONENTS IN SYSTEM

Essentially, the system consists of BLE beacons which send their packets to the phone. The application on the phone then processes the packets received from the BLE beacons and sends the information processed over to a MySQL central server via PHP Scripts described in Section 5.3, pages 39 – 41. The retrieval PHP scripts, described in Section 5.3, pages 41 – 43, in combination with JavaScript/AJAX scripts constantly retrieve data stored in the MySQL central database to in order to produce HTML to display the positions of the individuals located within the labs on the PC.

## 6. UNIT TESTING

Each Java algorithm that was used in the system was tested to ensure that the system produces results as intended. The unit tests are conducted in a manner where input is fed to each method and the results from each method are then analyzed to ensure that the results produced are expected. The unit tests for the purposes of this system are done manually given that the system on a whole will be using approximations from beacons, for example. Thus, for the unit tests, each method would also be fed approximations to receive approximate values, and if the results of the test are similar to the values expected, it can be concluded that the method has been developed correctly.

### 6.1 TESTING THE TRILATERATION ALGORITHM

A test, for example, would be to choose known points:

$(x, y) = (5, 3)$ ; which in this case would be the expected values - the result of the trilateration function

$$(x_a, y_a) = (1, 2)$$

$$(x_b, y_b) = (3, 8)$$

$$(x_c, y_c) = (10, 7)$$

The distance between  $(x_a, y_a)$  and  $(x, y)$  is approximately 4.12311, which is determined using the Euclidean distance formula. Thus,  $r_a = 4.12311$ . Similarly,  $r_b = 5.38516$  and  $r_c = 6.40312$ .

```
private BeaconAttributes test1 = new BeaconAttributes(1, 2, 4.12311);
private BeaconAttributes test2 = new BeaconAttributes(3, 8, 5.38516);
private BeaconAttributes test3 = new BeaconAttributes(10, 7, 6.40312);
```



```

private BeaconAttributes trilateration(BeaconAttributes ba1, BeaconAttributes ba2, BeaconAttributes ba3) {
    //Initializing all the variables
    double xa, xb, xc, ya, yb, yc, ra, rb, rc;
    xa = ba1.getX();
    xb = ba2.getX();
    xc = ba3.getX();

    ya = ba1.getY();
    yb = ba2.getY();
    yc = ba3.getY();

    ra = ba1.getDistance();
    rb = ba2.getDistance();
    rc = ba3.getDistance();

    double va = (Math.pow(rb, 2)-Math.pow(rc, 2) + Math.pow(xc, 2)-Math.pow(xb, 2) + Math.pow(yc, 2)-Math.pow(yb, 2)) / 2.0;
    double vb = (Math.pow(rb, 2)-Math.pow(ra, 2) + Math.pow(xa, 2)-Math.pow(xb, 2) + Math.pow(ya, 2)-Math.pow(yb, 2)) / 2.0;

    double y = ((vb * (xc - xb)) - (va * (xa - xb))) / (((ya - yb) * (xc - xb)) - ((yc - yb) * (xa - xb)));
    double x = (va - (y * (yc - yb))) / (xc - xb);

    BeaconAttributes position = new BeaconAttributes(x, y);
    return position;
}

public static void main(String[] args){
    Trilateration trilateration = new Trilateration();

    BeaconAttributes pos = trilateration.trilateration(trilateration.test1, trilateration.test2, trilateration.test3);

    System.out.println("X: " + String.valueOf(pos.getX()) + " Y: " + String.valueOf(pos.getY()));
}

```

The variables are initialized as an instance of the class “BeaconAttributes”, which is essentially a class written to save the positions of the beacons along with the distances from each beacon. The “trilateration()” function in this case returns an instance of “BeaconAttributes”, which saves the position of x and the position of y. The results of the test are as follows:



```

<terminated> Trilateration [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_31.jdk/Contents/Home/bin/java (15 Apr 2017 18:58:01)
X: 5.000001168110228 Y: 3.0000069311715905

```

These values are approximations of the expected values of (x, y) mentioned above. The reason the values have not been exactly 5.0 and 3.0 are perhaps because the distance values are approximations and not exact values. However, with respect to receiving distance estimations from the BLE beacons, the values of x and y displayed above can be considered accurate enough. Thus, it can be concluded that the trilateration function has been implemented correctly.

## 6.2 TESTING THE COMBINATIONS FOR 6-BEACON TRILATERATION

Given that trilateration requires three beacons, to involve all six beacons in trilateration, the average of all the coordinates from 3-beacon combinations from a set of 6 would lead to the position of the individual. The combinations,  $C(n, r)$  where “n” is the total number

of items to choose from and “r” is the number of items to place in the subset, in this scenario, would be  $C(6, 3)$ . The number of subsets generated would be 20. Thus, when a list of all combinations is generated, the size of the list can be checked as a secondary measure to ensure that the number of combinations generated is accurate.

```
private BeaconAttributes b1 = new BeaconAttributes("red", 0.61, 0.7, 2.68);
private BeaconAttributes b2 = new BeaconAttributes("blue", 6.2, 10.57, 3.98);
private BeaconAttributes b3 = new BeaconAttributes("green", 6.2, 5.6, 2.19);
private BeaconAttributes b4 = new BeaconAttributes("yellow", 5, 10, 8.544);
private BeaconAttributes b5 = new BeaconAttributes("purple", 1, 2, 3.245);
private BeaconAttributes b6 = new BeaconAttributes("black", 0.1, 8, 1.205);
private BeaconAttributes[] allBeacons = new BeaconAttributes[] {b1, b2, b3, b4, b5, b6};
private List<BeaconAttributes[]> findAllSubsets(int r, BeaconAttributes[] beaconList) {
    int[] indices = new int[r];
    List<BeaconAttributes[]> allCombinationsOfBeacons = new ArrayList<BeaconAttributes[]>();

    if (r <= beaconList.length) {
        for (int i = 0; (indices[i] = i) < r - 1; i++) ;
        allCombinationsOfBeacons.add(getSubset(beaconList, indices));
        while(true) {
            int i;
            for (i = r - 1; i >= 0 && indices[i] == beaconList.length - r + i; i--) ;
            if (i < 0) {
                break;
            } else {
                indices[i]++;
                for (++i; i < r; i++) {
                    indices[i] = indices[i - 1] + 1;
                }
                allCombinationsOfBeacons.add(getSubset(beaconList, indices));
            }
        }
    }
    return allCombinationsOfBeacons;
}

BeaconAttributes[] getSubset(BeaconAttributes[] input, int[] subset) {
    BeaconAttributes[] result = new BeaconAttributes[subset.length];
    for (int i = 0; i < subset.length; i++) {
        result[i] = input[subset[i]];
    }
    return result;
}

public static void main(String[] args){
    Trilateration trilateration = new Trilateration();

    List<BeaconAttributes[]> listOf3Beacons = new ArrayList<BeaconAttributes[]>();

    listOf3Beacons = trilateration.findAllSubsets(3, trilateration.allBeacons);
    for(BeaconAttributes[] combination: listOf3Beacons){
        System.out.println(Arrays.toString(combination));
        for(int i = 0; i < combination.length; i++){
            System.out.println(combination[i].getName());
        }
    }

    System.out.println("List size: " + listOf3Beacons.size());
}
```

The variables were defined above and an extra field of name was given to the BeaconAttributes class in order to distinguish each variable within the ArrayList of BeaconAttributes arrays, which are essentially the subsets. Given that the ArrayList is constantly being populated with BeaconAttributes arrays in the application, the variable of

allCombinationsOfBeacons was left as local within the method and returned with each method call. This ensures that there are always 20 combinations within the ArrayList as opposed to constantly adding more values in the same ArrayList had the variable been made global.

```
[BeaconAttributes@7852e922, BeaconAttributes@4e25154f, BeaconAttributes@70dea4e]
red
blue
green
[BeaconAttributes@7852e922, BeaconAttributes@4e25154f, BeaconAttributes@5c647e05]
red
blue
yellow
[BeaconAttributes@7852e922, BeaconAttributes@4e25154f, BeaconAttributes@33909752]
red
blue
purple
[BeaconAttributes@7852e922, BeaconAttributes@4e25154f, BeaconAttributes@55f96302]
red
blue
black
[BeaconAttributes@7852e922, BeaconAttributes@70dea4e, BeaconAttributes@5c647e05]
red
green
yellow
[BeaconAttributes@7852e922, BeaconAttributes@70dea4e, BeaconAttributes@33909752]
red
green
purple
[BeaconAttributes@7852e922, BeaconAttributes@70dea4e, BeaconAttributes@55f96302]
red
green
black
[BeaconAttributes@7852e922, BeaconAttributes@5c647e05, BeaconAttributes@33909752]
red
yellow
purple
[BeaconAttributes@7852e922, BeaconAttributes@5c647e05, BeaconAttributes@55f96302]
red
yellow
black
[BeaconAttributes@7852e922, BeaconAttributes@33909752, BeaconAttributes@55f96302]
red
purple
black
[BeaconAttributes@4e25154f, BeaconAttributes@70dea4e, BeaconAttributes@5c647e05]
blue
green
yellow
[BeaconAttributes@4e25154f, BeaconAttributes@70dea4e, BeaconAttributes@33909752]
blue
green
purple
[BeaconAttributes@4e25154f, BeaconAttributes@70dea4e, BeaconAttributes@55f96302]
blue
green
black
[BeaconAttributes@4e25154f, BeaconAttributes@5c647e05, BeaconAttributes@33909752]
blue
yellow
purple
[BeaconAttributes@4e25154f, BeaconAttributes@5c647e05, BeaconAttributes@55f96302]
blue
yellow
black
[BeaconAttributes@4e25154f, BeaconAttributes@33909752, BeaconAttributes@55f96302]
blue
purple
black
[BeaconAttributes@70dea4e, BeaconAttributes@5c647e05, BeaconAttributes@33909752]
green
yellow
purple
[BeaconAttributes@70dea4e, BeaconAttributes@5c647e05, BeaconAttributes@55f96302]
green
yellow
black
[BeaconAttributes@70dea4e, BeaconAttributes@33909752, BeaconAttributes@55f96302]
green
purple
black
[BeaconAttributes@5c647e05, BeaconAttributes@33909752, BeaconAttributes@55f96302]
yellow
purple
black
List size: 20
```

The results displayed above also indicate that the method produces a list of distinct combinations. Furthermore, this is also established by the final line of the results indicating that the list's size is 20.

### 6.3 TESTING THE ACCURATE AVERAGE DISTANCE COMPARATOR

To test this method, the distances of each beacon are stored in a HashMap. The HashMap is then sorted using the accurate averages of the distances stored in each DistanceQueue. Once the HashMap is sorted, the top three values are returned along with their keys. These values are then compared with all the values in the HashMap. Based on whether the top three values are the smallest accurate average distance values received from all six beacons, the method can be concluded as developed correctly or incorrectly.

```
private BeaconAttributes b1 = new BeaconAttributes("red", 0.61, 0.7, 2.68);
private BeaconAttributes b2 = new BeaconAttributes("blue", 6.2, 10.57, 3.98);
private BeaconAttributes b3 = new BeaconAttributes("green", 6.2, 5.6, 2.19);
private BeaconAttributes b4 = new BeaconAttributes("yellow", 5, 10, 8.544);
private BeaconAttributes b5 = new BeaconAttributes("purple", 1, 2, 3.245);
private BeaconAttributes b6 = new BeaconAttributes("black", 0.1, 8, 1.205);
private HashMap<String, BeaconAttributes> allBeacons = new HashMap<String, BeaconAttributes>();

public void addBeaconsToList(){
    allBeacons.put(b1.getName(), b1);
    allBeacons.put(b2.getName(), b2);
    allBeacons.put(b3.getName(), b3);
    allBeacons.put(b4.getName(), b4);
    allBeacons.put(b5.getName(), b5);
    allBeacons.put(b6.getName(), b6);
}

//Comparator class for comparing the accurate averages
class DistanceComparator implements Comparator<Map.Entry<String, BeaconAttributes>>
{
    @Override
    public int compare(Map.Entry<String, BeaconAttributes> o1, Map.Entry<String, BeaconAttributes> o2) {
        return o1.getValue().getDistance() > o2.getValue().getDistance() ? -1 : 1;
    }
}

//method to return the top three values of the sorted distances
public List<Entry<String, BeaconAttributes>> getSortedDistances(Map<String, BeaconAttributes> map) {
    Set<Map.Entry<String, BeaconAttributes>> entrySet = map.entrySet();
    PriorityQueue<Entry<String, BeaconAttributes>> topThree = new PriorityQueue<Entry<String, BeaconAttributes>>(3, new DistanceComparator());
    //order the hashmap by accurate averages in descending order
    for(Entry<String, BeaconAttributes> entry: entrySet){
        topThree.offer(entry);

        //remove the greatest value in the priorityqueue
        while(topThree.size() > 3){
            topThree.poll();
        }
    }
    List<Entry<String, BeaconAttributes>> result = new ArrayList<Map.Entry<String, BeaconAttributes>>();
    while(topThree.size() > 0){
        result.add(topThree.poll());
    }
    return result;
}

Name: red; distance: 2.68
Name: green; distance: 2.19
Name: black; distance: 1.205
List size: 3
```

Given that while testing, there is no input stream of distances, the `DistanceQueue` class is replaced with the `BeaconAttributes` class, and the hard-coded distances are compared in this test instead. The results produced are as expected. The three lowest distances from the variables above are 1.205, 2.19, and 2.68, and the results indicate that in descending order along with ensuring that only the lowest three distance values are returned.

#### 6.4 TESTING DISTANCEQUEUE AND THE ACCURATE AVERAGES

In the `DistanceQueue` class, there are several significant methods. For example, the “offer” method adds a value to the queue while removing the earliest added value from the queue. This maintains the queue’s ability to only store the most recent – and therefore, most relevant – values. To test the offer method, a simple test was conducted, the queue’s limit is set to 5, and 6 values are added to the queue.

```
public static void main(String[] args){
    DistanceQueue d = new DistanceQueue(5);
    d.offer(1);
    d.offer(7);
    d.offer(2);
    d.offer(5);
    d.offer(6);
    d.offer(9);
    System.out.println(d.queue);
```

```
[9.0, 6.0, 5.0, 2.0, 7.0]
```

As expected, the values in the queue exclude “1”, which is the first value that was added to the queue.

The next test that was conducted was to ensure that the accurate average is producing the right result. The values added to the queue are quite similar to each other with the exception of one extremely large value and one extremely small value. The accurate average is determined by first calculating the average of all the values in the queue. The average is then used to determine the variance that when square rooted yields the standard deviation. Once the standard deviation has been determined, the standard error of the mean can be produced. The standard error is significant as it allows the definition of an upper and lower bound of values. Anything between the upper and lower bound is included in the calculations of the accurate average, and any value above the upper bound and any value

below the lower bound is considered an outlier and is considered to skew the actual values of the distances<sup>(15)</sup>.

For the purposes of determining the distances, the top 5% and the bottom 5% of values. Thus, 90% of the values would be used in the calculations of the accurate average. This allows a wide enough range between the average, and as mentioned above, prevents skewed results. The upper and lower bound values are determined using the mean  $\pm$  standard error multiplied by 1.645 (+ for upper bound, - for lower bound), which is a predetermined value to cover 90% of values within the normal distributions curve<sup>(16)</sup>. This is the recommended range of values used for removing outliers. If the dataset consists of more outliers within the 90% of the data that is used for determining the trend, then the dataset can be considered unreliable.

For this test, the DistanceQueue's limit was set to 10. The following values were added to the queue: 1000.0, 120.0, 108.0, 121.0, 9.0, 106.0, 105.0, 102.0, 107.0, 111.0. The two values that are expected to be omitted from calculations of the accurate average are: 1000.0 because it is a much larger value and 9.0 because it is a much smaller value than the rest of the queue.

The implementation is as follows:

```

public double findAccurateAverage(){
    double sum = 0;
    double count = 0;
    double stdDev = getStandardDeviation();
    double mean = findAverage();
    double SEM = stdDev/Math.sqrt(queue.size());
    System.out.println("SEM: " + SEM);
    double upperbound = mean + SEM * 1.645;
    double lowerbound = mean - SEM * 1.645;
    System.out.println("lower " + lowerbound);
    System.out.println("upper " + upperbound);

    for(int i = 0; i < queue.size(); i++){
        if((double) queue.get(i) <= upperbound && (double) queue.get(i) >= lowerbound){
            System.out.println((double) queue.get(i));
            sum += (double) queue.get(i);
            count++;
        }
        else{
            System.out.println((double) queue.get(i) + " not added");
            continue;
        }
    }

    double newMean = sum/count;

    return newMean;
}

public static void main(String[] args){
    DistanceQueue d = new DistanceQueue(10);
    d.offer(111);
    d.offer(107);
    d.offer(102);
    d.offer(105);
    d.offer(106);
    d.offer(9);
    d.offer(121);
    d.offer(108);
    d.offer(120);
    d.offer(1000);
    System.out.println(d.queue);

    System.out.println("Mean: " + d.findAverage());
    System.out.println("Variance: " + d.getVariance());
    System.out.println("Standard Dev: " + d.getStandardDeviation());
    System.out.println("Accurate Average: " + d.findAccurateAverage());
}

```

```

[1000.0, 120.0, 108.0, 121.0, 9.0, 106.0, 105.0, 102.0, 107.0, 111.0]
Mean: 188.9
Variance: 74038.89000000001
Standard Dev: 272.10088202723637
SEM: 86.04585405468411
lower 47.354570080044624
upper 330.44542991995536
1000.0 not added
120.0
108.0
121.0
9.0 not added
106.0
105.0
102.0
107.0
111.0
Accurate Average: 110.0

```

As expected, the values of 1000.0 and 9.0 were not included in the calculations of the accurate average given that they are outliers. The rest of the values when added equals to  $880/8$  gives the accurate average of 110.0.



## 7. RESULTS AND EVALUATION

Primarily in this section, the application would be tested as a whole to determine which algorithm is most suited for the system. The results of each test on the application would also be analyzed.

### 7.1 TESTING OF RANGING/TRILATERATION

The tests of the ranging/trilateration application will be done in a manner in which the application can be analyzed under real-world conditions as opposed to theoretical conditions. From unit testing in the previous section, it is apparent that the algorithms developed for the application have been developed correctly. Thus, these tests are conducted in order to evaluate the technology being used for the application. The maximum capacity of Windows Lab 2 is 29 given that there are 29 computers currently placed in that lab, and ideally, only one person should occupy one computer. Given that the application is currently a proof of concept in the Android platform, it is difficult to get 29 Android users to test the application to simulate the maximum capacity. The reason for simulating maximum capacity would be to see to what extent would 29 bodies affect the Bluetooth signals – due to the fact that Bluetooth is highly prone to interference by water and the human body is approximately 45 - 70% water<sup>(17)</sup>. Nonetheless, on average, six to ten users would simulate a normal usage of the lab, and the values from the six to ten users can be used to determine the most accurate sub-approach for trilateration.

Furthermore, given that metal also causes Bluetooth interference, the values received on each user's phone can also indicate to what extent computers cause interference. Using a tape measure, each user's position in terms of x and y-coordinates would be determined relative to the origin, which was set to be the top right corner. The user would be asked to switch on the application with their phone remaining in the same spot. The phone will display a set of coordinates, and the coordinates displayed would then be compared against the coordinates that were determined using the tape measure.

The test conditions for each test conducted would be similar. Each test would consist of six users with the application installed on his/her phone. Each user currently holds a different phone which ranges from the following brands: Samsung, Google, Oneplus, and HTC.

Using different brands in conducting these tests ensure that the receivers of the phones are not the issue in the system. Each user would then be placed in different parts of the lab and the user's coordinates would be pre-determined using a tape measure as mentioned above. The beacons are set up according to the plans seen in Figure 5 – Figure 7, pages 26 and 27. The height of the beacons were, on average, at a height of: 1.53m, which is about 0.87m above where the phone would be. This height can be considered negligible given that the beacons are placed in a manner which is above the computers in order to avoid interference due to the metal in the computers. The beacon settings were set to an advertising interval of 201 milliseconds, which as established, is the ideal advertising interval for the purposes of the system given that every 20.1 seconds, the user has a fresh set of values which are used to determine the accurate average of the distance and position. This takes into consideration if the user has moved from his/her spot.

Furthermore, the transmit power was set to +4 dBm, which as mentioned in the Implementation section, was necessary in order to achieve a more accurate distance estimate. The tests were not repeated more than once due to the fact that a wide variety of phones used within the tests would ensure that the Oneplus One phone being used to test the initial iteration of the application without statistical techniques is not harboring faulty hardware – particularly with respect to the Bluetooth receiver. Instead, each test consisted of allowing the user's phone to receive a more consistent Bluetooth flow of packets from each BLE beacon by allowing the phone to continuously receive values for five minutes – thus, the application was left on the phone for five minutes for the purposes of the test. This allows enough distance and position estimates to be made for the purposes of finding the accurate average. The statistical technique of finding an accurate average without outliers is used in order to increase the accuracy of the distance estimates received from the BLE beacons. As mentioned in Section 6.4, page 52, if the distance estimates received from the BLE beacons have more outliers than 90% of the distance estimates, then the estimates from the beacons can be considered to be unreliable.

The purposes of these tests are primarily to determine which plan would be the most optimum in terms of using the application for the purposes of determining each individual's exact position in the labs. However, more information about the system can be derived from the tests such as whether the cause of inaccurate results is due to a faulty Bluetooth receiver on the initial phone used for testing or whether the BLE proximity beacons used

for the application are simply inadequate for replicating the role of satellites in GPS. Given the dimensions of the lab, in the ideal scenario, the beacons would be able to provide an accuracy of the position determined to 1 meter from the actual position. Anything above 1.5 meters from the actual position would be too inaccurate due to the relatively small dimensions of Windows Lab 2.

### 7.1.1.1 RESULTS OF TRILATERATION TESTS

As mentioned above the tests would consist of comparing the user's actual position in the labs against the user's position displayed on the application. This test consisted of placing the phone in one location of the lab for five minutes in order to allow values to be more consistent when determining distance averages, etc. It has been noticed that when the phone initially receives Bluetooth values from the phone, the values are initially inconsistent. The test conditions are the same as those described on the previous page.

The first set of the results of the tests are below along with the positions they were tested in:

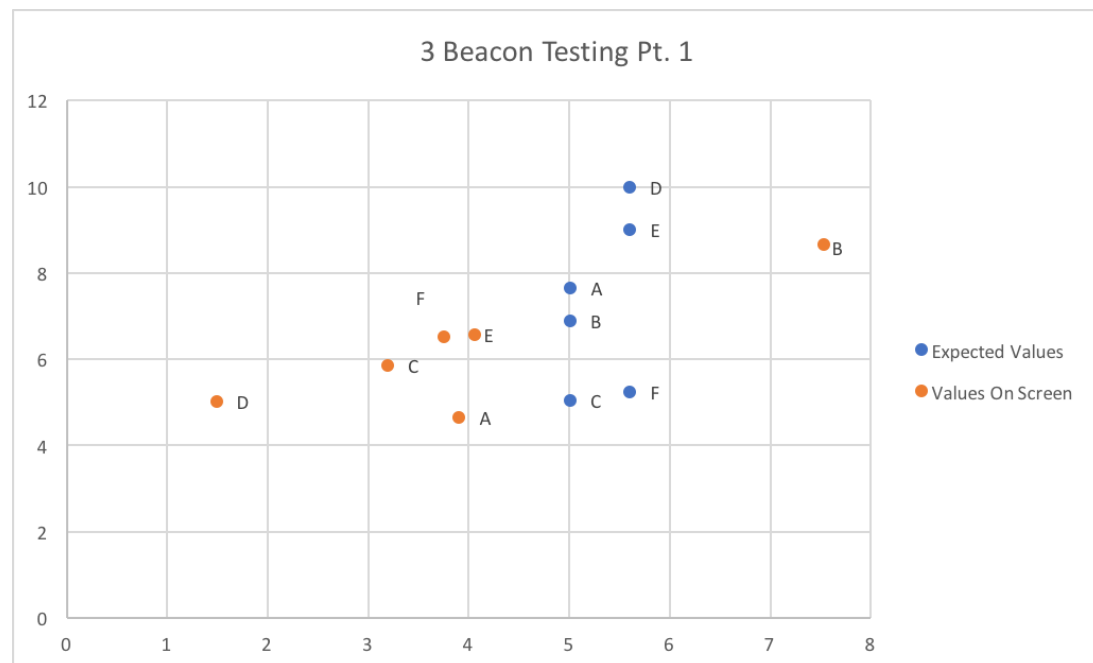


FIGURE 10: RESULTS OF 3-BEACON TRILATERATION ACCORDING TO FIG. 5

The results in Figure 10 were tested with the beacons placed in the orientation as seen in Figure 5. As seen in the legend, the “Expected Values” are the values determined by

manually placing the user within the labs and using a tape measure. The “Values On Screen” were the positions determined by the trilateration method. Each value in blue is labeled with a letter, and its corresponding value in orange is labeled with the same letter to depict the difference in the Expected Values and the Values On Screen determined by the application.

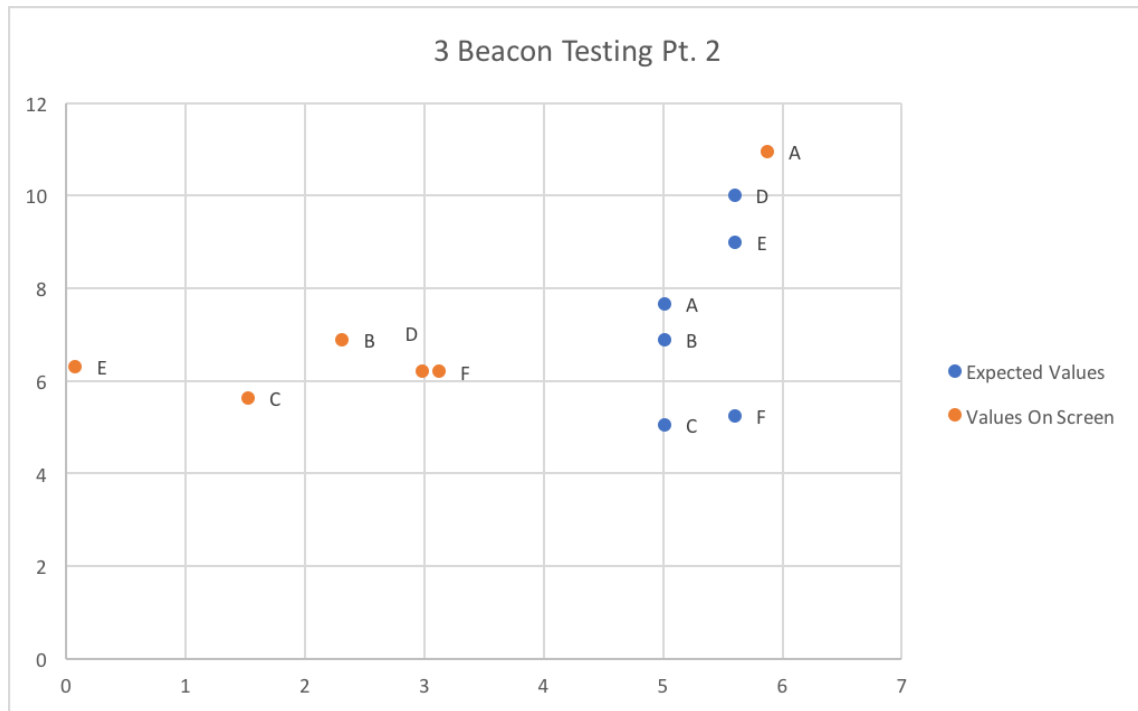


FIGURE 11: RESULTS OF 3-BEACON TRILATERATION ACCORDING TO FIG. 6

Comparing the values in Figure 10 and Figure 11, it is apparent that simply using three beacons does not yield an accurate position of the user. The Values On Screen are much more displaced as opposed to their corresponding Expected Values in both scenarios.

	Figure 10	Figure 11
Person	Distance (m)	
A	3.208145882	3.390870095
B	3.085255257	2.700018518
C	1.992109435	3.537866589
D	6.464688701	4.615668966
E	2.868449058	6.145185107
F	2.243969697	2.659323222
Average	3.310436338	3.84148875

FIGURE 12: DISTANCE BETWEEN EXPECTED VALUES AND VALUES ON SCREEN FOR 3-BEACON TRILATERATION

Figure 12 displays the values showing the Euclidean distances between the Expected Values and Values On Screen for Figure 10 on the left column and Figure 11 on the right column. These are a numerical representation of what was mentioned in before. The positions determined from 3-beacon trilateration are on average 3.3m off for the left column and 3.8m off for the right column. Thus, these values can be concluded to be far too inaccurate.

Similar to the test above, the phone remained still for a duration of five minutes and the values displayed on screen were taken as the values that were determined by the phone under the same test conditions as those described on page 55. There were six beacons set up in the lab for this test, and they were placed in the positions as seen in Figure 7. The implementation of the tests were that of 6-beacon trilateration where subsets of three of the six beacons were used, and the position from each subset was determined and averaged.

The results are displayed below:

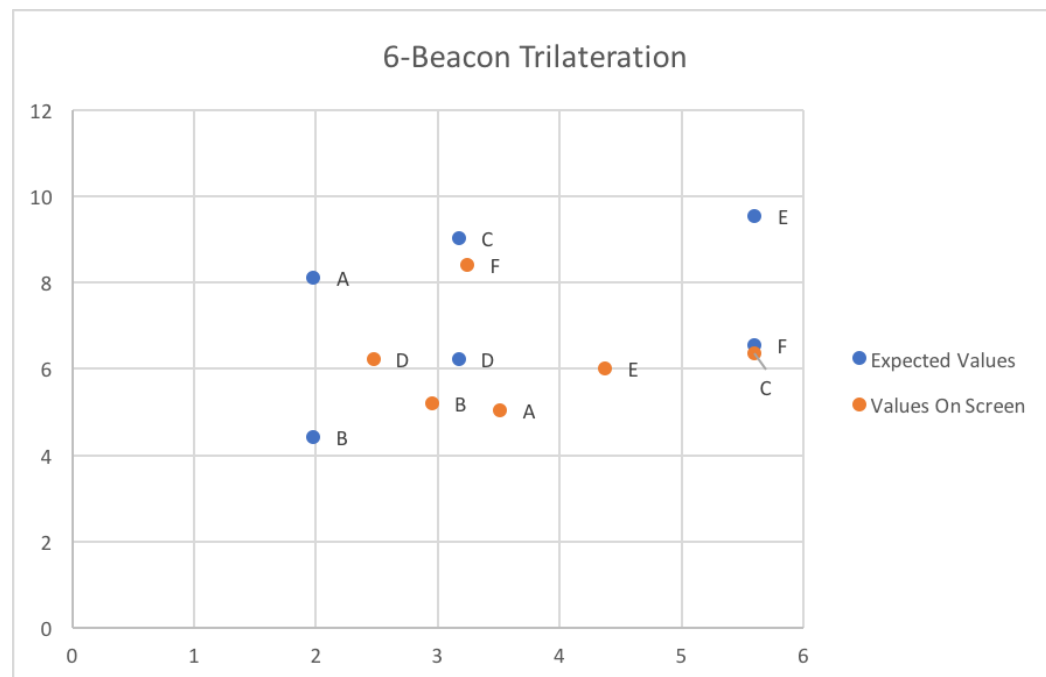


FIGURE 13: RESULTS FOR 6-BEACON TRILATERATION

Person	Distance (m)
A	3.448042923
B	1.244708801
C	3.595469371
D	0.7
E	3.738154625
F	2.998683044
<b>Average</b>	<b>2.620843127</b>

FIGURE 14: DISTANCE BETWEEN EXPECTED VALUES AND VALUES ON SCREEN FOR 6-BEACON TRILATERATION

In Figure 13 and 14 above, with the exception of person B and D that received values relatively near to their actual position, the 6-beacon trilateration also was not able to place an individual near to their respective Expected Values. On average, the distance between the Expected Values and the Values On Screen is: 2.62m, which is still quite inaccurate.

The next test was conducted using the 3-closest beacon approach with the beacons remaining in the same setup as seen in Figure 7. Once again, the position of the phone remained still for five minutes under the same test conditions as described on page 55, and the values displayed on the screen of the application were compared against the predetermined positions.

The results are displayed on the next page:

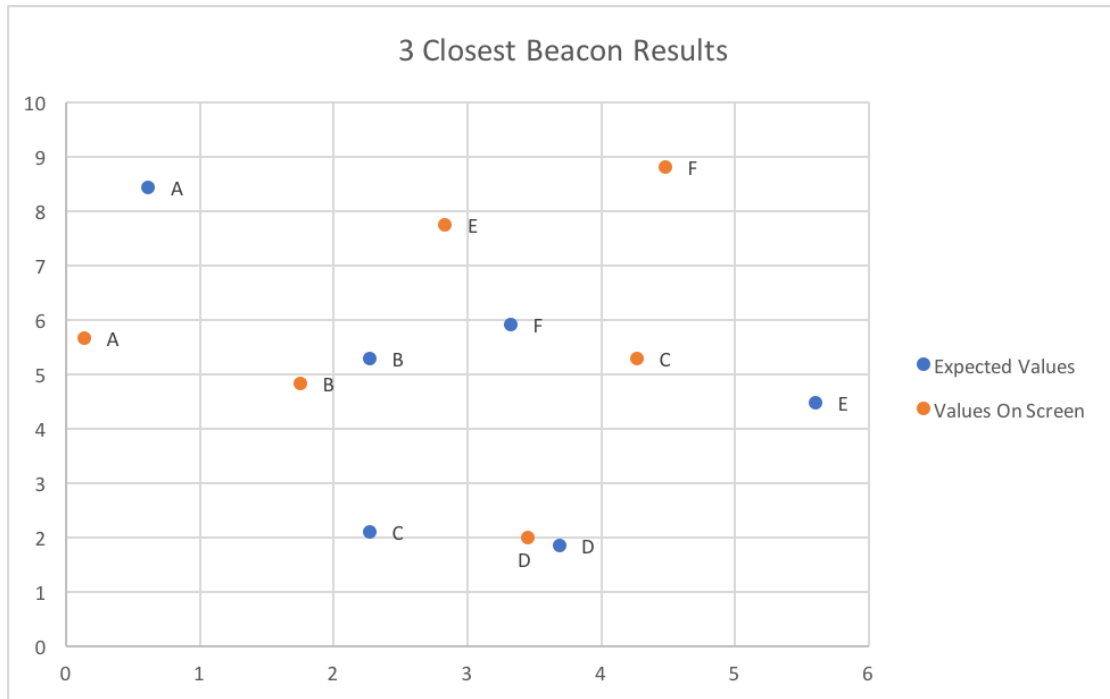


FIGURE 15: 3-CLOSEST BEACON TRILATERATION RESULTS

Person	Distance (m)
A	2.811280847
B	0.694262198
C	3.7651162
D	0.27784888
E	4.285533806
F	3.123395588
<b>Average</b>	<b>2.492906253</b>

FIGURE 16: DISTANCES BETWEEN EXPECTED VALUES AND VALUES ON SCREEN FOR 3-CLOSEST BEACON TRILATERATION

The results for the 3-closest beacons are more promising given that the values for A, B, and D on screen are relatively near the expected values. However, C, E, and F are still much farther than their expected values. On average, the distance between the Expected Values and the Values On Screen is: 2.49m, which is still highly inaccurate.

All three approaches yielded inaccurate values with the best being 2.49m, on average, away from the actual position. Thus, statistically, this is inaccurate which requires a further

investigation as to why there are inaccuracies in distance estimates. Further tests are conducted in the following subsection.

### 7.1.2 TESTS CONDUCTED FOR DETERMINING RELIABILITY OF BLE BEACONS

Following the experiments conducted above, it can be deduced that the initial phone used for testing the application does not contain a faulty Bluetooth receiver given that the range of phones used in the testing of each scenario still produced inaccurate results. Furthermore, given that the unit tests were conducted in Section 6 along with the phones having functioning Bluetooth receivers which was concluded in Section 7.1.1, the reliability of the BLE beacons are tested. For the purposes of this test, one beacon from each manufacturer (Estimote and Kontakt.io) is tested. The beacon is placed under three object conditions: (1) no object is placed in between the beacon and the phone, (2) a human body is placed in between the beacon and the phone, and (3) a computer is placed in between the beacon and the phone. The settings of the beacons were set to 201ms (0.201 seconds) for the advertising interval – thus, packets are sent to the phone from the beacons every 201ms, and the transmit power was set to +4 dBm, the same as the beacon settings for testing the application in Section 7.1.1, page 55. The test consisted of receiving 100 distance estimations from the beacon from the phone under all three object conditions at a distance of 1 meter between the phone and the beacon.

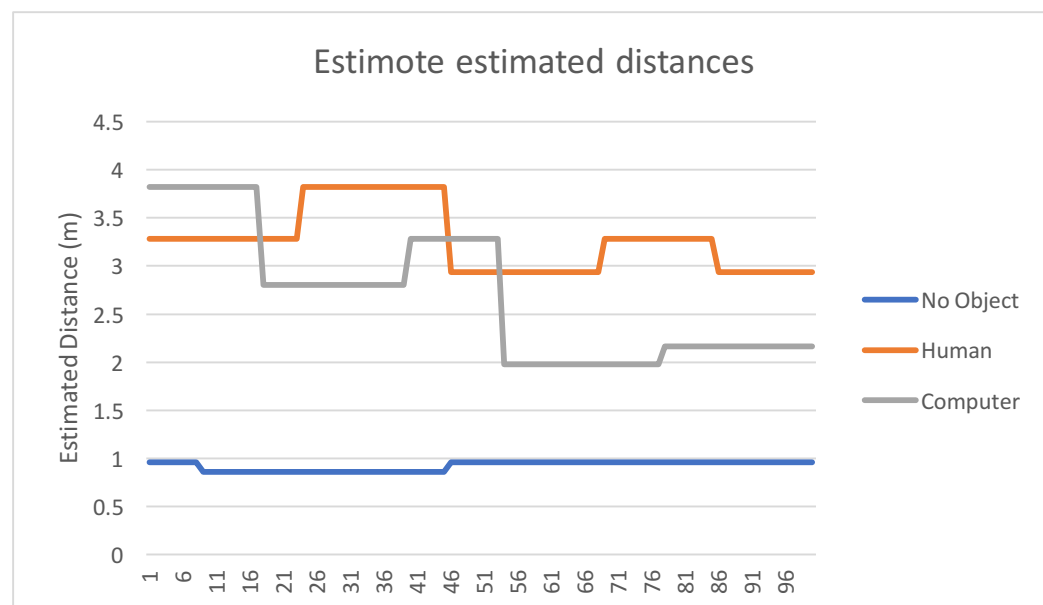


FIGURE 17: ESTIMATED DISTANCES AT 1M FOR ESTIMOTE



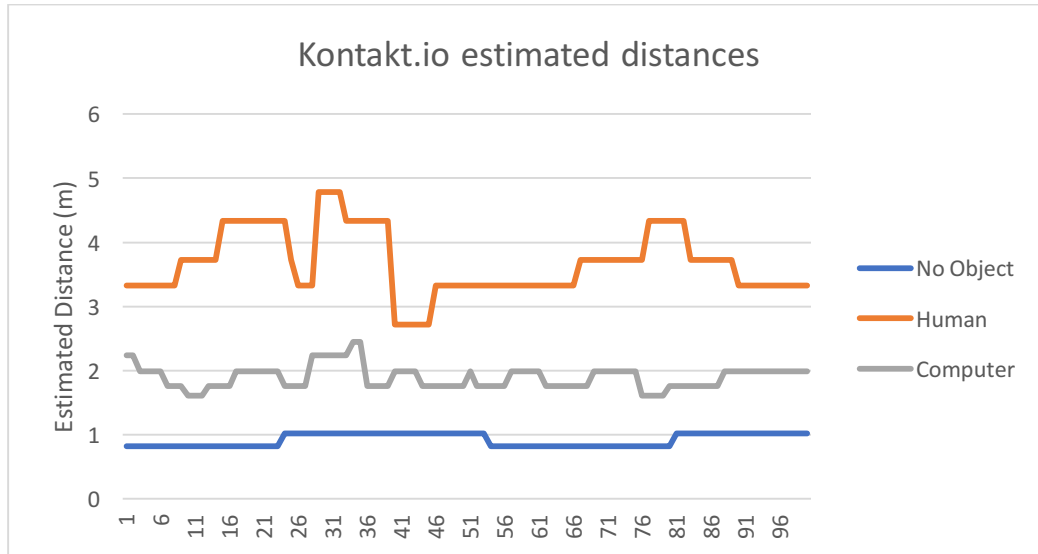


FIGURE 18: ESTIMATED DISTANCES AT 1M FOR KONTAKT.IO

The estimated distances by the Estimote and Kontakt.io beacons at 1 meter away from the phone without any objects in between the beacon and the phone is quite accurate. The average estimated distance by the application is 0.92m from Estimote and Kontakt.io. However, when comparing the estimated distances to an object placed in between the beacon and the phone, there is quite a large inaccuracy. On average, when a human is between the beacon and the phone at 1 meter, the phone estimates the distance is 3.27m from the Estimote beacon 3.68m from the Kontakt.io beacon, and the phone estimates the distance between the phone and the beacon when a computer is placed in between is 2.70m from the Estimote beacon and 1.90m from the Kontakt.io beacon. When looking at the dimensions of the lab, which are 6.2 x 12 meters, the estimated distances between the phone and the beacon with objects in between are quite inaccurate. The numerical results for the beacons are attached as Appendix D. It can therefore be concluded that despite attempting to avoid interference by trying different approaches for trilateration, it is difficult to achieve an accurate distance estimation even at 1 meter away from the beacon if objects are in between the beacon and the phone.

A significant point to note is that during the second phase of testing with respect to placing the beacons under three different object conditions to determine the accuracy of the distance estimates is that Estimote beacon often lost signal while sending packets over to the phone. This is also a significant point to note for Section 7.3, page 65 due to the fact

that a signal loss would also render the proximity beacons unreliable for placing an individual in a particular section of the lab as well. The loss in signal is attached determined via a “log” file in Android which is attached as Appendix C in this report. It can be seen in the text from 17:20:14.339 to 17:21:14.661 there was no advertising packets being received from the beacon.

## 7.2 ANALYSIS OF TRILATERATION RESULTS

Judging from the results above in Figure 10 - 18, BLE beacons are incapable of accurately positioning an individual within a room. The 3-closest beacon approach provided the most promising results for certain phones with an average of 2.49m of distance between the values displayed from the application and the expected value. Different models of phones have different quality Bluetooth receivers, and thus, as seen with the results for Figure 16, this is still not enough to produce an actual application for monitoring the occupancy of the labs along with the position of the individual.

The inaccuracy in results is fundamentally Bluetooth being incapable of providing accurate distance estimates. This can be seen with the extra tests conducted with respect to receiving distance estimates from each beacon manufacturer. Furthermore, the environment the BLE beacons are placed in is much more prone to interference due to the PCs of the Windows Lab 2 as opposed to an open environment. Viewing the results in Figure 17 and Figure 18, the theory that if more than 10% of the data is outliers, then the data is unreliable is confirmed. Therefore, despite statistical techniques being used to remove outliers in the distance estimates, the estimates itself are still unreliable; for example, when the phone is placed 1m away from the beacon with a human in between the phone and the beacon, the phone estimates the distance between itself and the beacon is, on average, 3.27m for Estimote, which is undoubtedly unreliable. Thus, it can be concluded that the proximity beacons are not the most appropriate technology for the purposes of trilateration and

## 7.3 TESTING FOR BACKGROUND MONITORING

The tests conducted for this approach are extensive in the fact that it tests whether the system works together as a whole. These tests cannot be conducted numerically given that there are no numerical values for each phone in terms of whether the phone is within the

proximity region of a respective beacon. Each phone is placed into a beacon's particular zone, and the values sent to the server are viewed online.

In total, there were six phones used for this test, and two phones each were placed into zones A, B, and C. The test was conducted with the phones left in place for a total of five minutes with the screen switched off and the application running in the background. This is done in order to see how consistent the beacons are with respect to sending advertising packets to the phone with keeping in mind the pervasive element of the application. The beacons advertising interval remained at 201 milliseconds for this test, and the transmit power was changed to -30 dBm in order to achieve a radius of 3 meters for each beacon. This allows each beacon to be placed in different sections of the lab. The beacons in this scenario are placed on the table in the positions seen in Figure 8. This is done due to the fact that users are likely to place their phones on the table of the desk, and placing the beacons on the desk as well gives the phones the best chance of receiving the advertising packet coming from the beacons.

The results are displayed below:

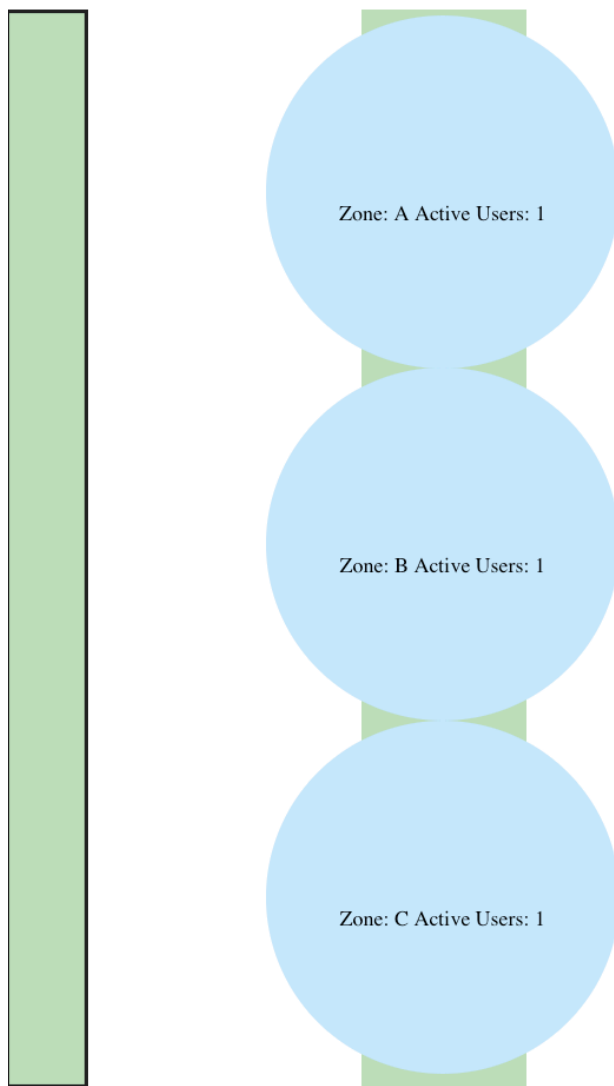


FIGURE 19: BEACON MONITORING RESULTS

The zones of each user were depicted as displayed in the image above. No phones were placed in zone E, F, and D. The results after five minutes were that each beacon registered only one phone at the time of five minutes being up as opposed to registering the two phones that were actually placed in each zone.

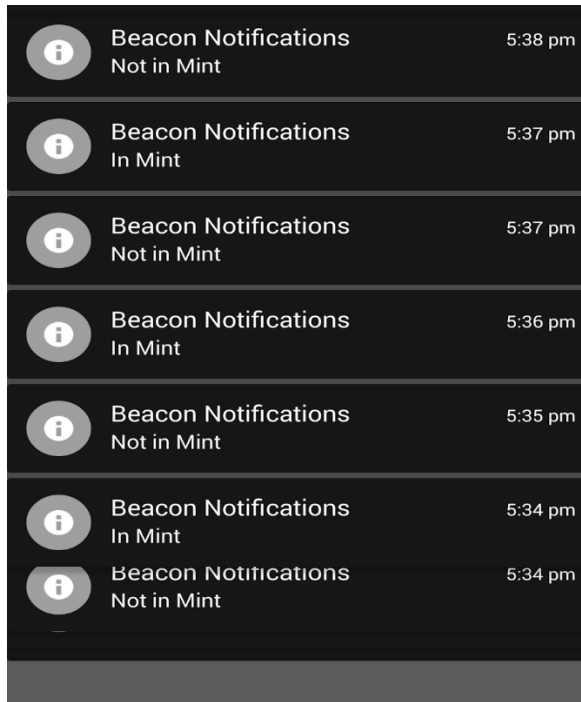


FIGURE 20: BEACON NOTIFICAIONS DISPLAYED WITHIN THE SPAN OF 5 MINUTES

Furthermore, it was noticed that the phones consistently received “onEnteredRegion” and “onExitedRegion” notifications as seen in Figure 20. This indicates that proximity beacons’ technologies are constantly receiving false “onExitedRegion” events and the hardware is not good enough to provide accurate values of the user to the server from the Estimote beacons. The notifications occur every minute; thus, the values are sent to the server often as seen in Figure 20. However, this means that the Estimote BLE beacons are unreliable at providing a consistent set of values for determining the position of the individual. Therefore, the proximity beacons can once again be said to not be reliable with respect to providing consistent data on the individuals within the lab.

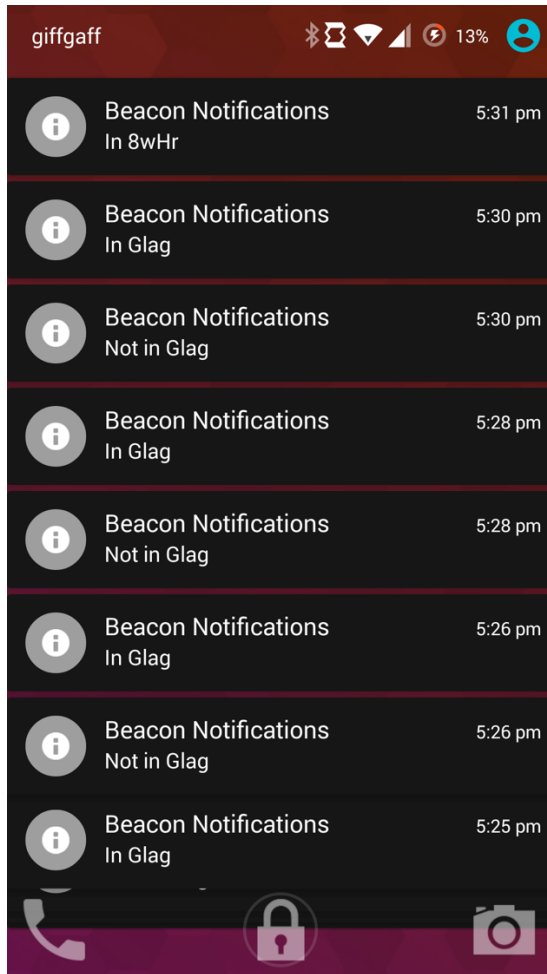


FIGURE 21: FALSE ENTER NOTIFICATION

In addition, in certain cases, the monitoring application triggered false “onEnteredRegion” events for a particular beacon’s region which the phone is not in. This also further adds to the unreliability of background monitoring given that the algorithm would essentially position the user in the wrong location of the lab. This issue was mentioned in the Introduction as a limitation given that the receiver within each Android phone is not taken into consideration when testing. This is seen in Figure 21 as there is a false enter event at 5:31PM in the beacon known as “8wHr” when the phone was placed in the region of the beacon known as “Glag”. These are the names of the Kontakt.io beacons, but the false enter event also occurs with Estimote beacons.

Despite this conclusion, however, another interesting fact noticed in the testing of background monitoring is that more expensive phones with better hardware are capable of receiving more consistent Bluetooth values when it comes to background monitoring. For example, the HTC One M10 managed to have only one enter event throughout the duration

of the testing while the phone was left in the same spot. A similar occurrence was noticed with the Samsung Galaxy S7 as seen in Figures 22 and 23. Thus, it can be concluded that this application has a much greater potential with more expensive phones. A significant point to add is that this was achieved with Kontakt.io beacons and not Estimote beacons. Thus, Estimote beacons are more unreliable than Kontakt.io beacons.

Furthermore, when testing for trilateration in Section 7.1.1, starting from page 57, there were certain results that proved to have a greater accuracy than the others. For example, person D in Figure 16 on page 61 had a much closer position to his/her expected value than person E. For the purposes of privacy, the person's phones were not recorded when testing for trilateration. However, judging from the results achieved in Figure 22 and 23, it can be concluded that person D had a more expensive phone with better hardware than person E. Thus, the trilateration application may also produce more accurate results with better Bluetooth receivers.

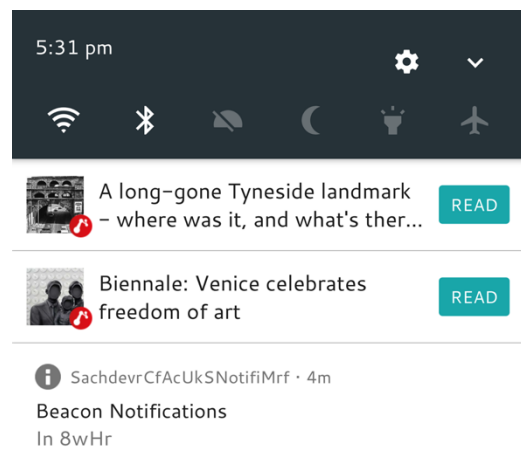


FIGURE 22: HTC ONE M10 BACKGROUND MONITORING

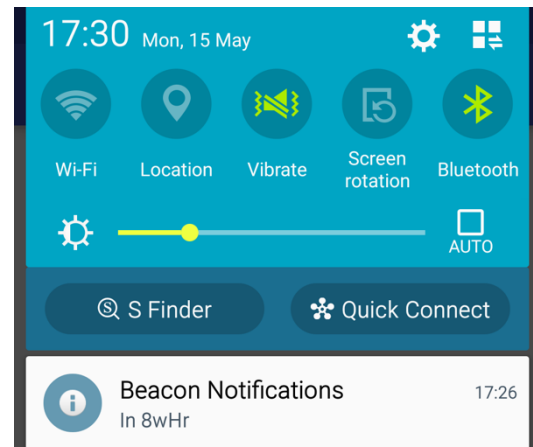


FIGURE 23: SAMSUNG GALAXY S7 BACKGROUND MONITORING

## 8. FUTURE WORKS

This section will explore future improvements on the current system along with provide alternative technologies – instead of BLE beacons – which can be incorporated into the current system for the purposes of increasing the accuracy of the position of an individual.

### 8.1 IMPROVEMENTS ON THE CURRENT SYSTEM

This subsection describes algorithmic changes to the current system, which could have possibly generated more accurate results along with producing a more complete system.

#### 8.1.1 IMPROVEMENTS TO THE OVERALL APPLICATION

Improvements on the current application can be made with respect to providing the user with visual feedback on the application itself. However, due to time constraints, the visual feedback was displayed on a web interface simply due to the ease at which the room was able to scale on the web. The web's interface would have to be programmed on the phone which use a different scalar reference of pixels, which due to time constraints were not achieved. This is the case for both the trilateration application and the background monitoring application. Regardless of the background monitoring application being perhaps more complete and pervasive than the trilateration application, it was still not possible to integrate the front-end into the application due to time constraints.

#### 8.1.2 IMPROVEMENTS TO RECEIVING ACCURATE DISTANCE VALUES

With respect to improvements on the current system, perhaps an implementation of a Kalman filter can be adopted into the code. The Kalman filter is essentially a recursive algorithm which uses historical data to produce a linear model of either RSSI value or the distance estimations based on the RSSI value. However, one of the greatest drawbacks of using a Kalman filter is that it assumes that the phone is not in motion<sup>(18)</sup>. This is essentially what allows the filter to use historical data to produce a prediction and an adjustment to develop a linear model.

To implement this into the application, perhaps the best approach would be to incorporate an accelerometer function into the application. Thus, if a user is with his/her phone and



his/her phone is in motion, then the filter can be reset until the user is not in motion again. Once the user is deemed to be stationary, the application can restart the filter and filter out distance estimates from the beacon again.

A significant point to note is that the filter is heavily reliant on the initial values received by the phone, which is why it is necessary to assume that the user is not in motion. Thus, if the phone is actually 1m away from the user, but the phone initially receives values of 4m, for example, then this is what the filter uses to base its model on. Hence, it would be difficult to truly incorporate such a filter into the application.

Other filters such as the particle filter can be applied to the application as this filter takes motion into account when determining distance estimates<sup>(19)</sup>. However, a significant point to note is that the particle filter is more computationally intensive than the Kalman filter given that the particle filter uses a simulation to predict where the individual would be using probabilities of each particle representing the true position of the individual. Thus, it would perhaps be difficult to incorporate this into the application which uses a phone's processor that may not necessarily be equipped to handle the computational intensity of the particle filter or perhaps the results may be quite delayed. Nonetheless, to incorporate this into the application for the purposes of increasing accuracy would perhaps provide more accurate results than those achieved with trying to determine an accurate average.

## 8.2 LOCATION BEACONS

This subsection explores the possibility of adding location beacons to the system as opposed to the use of proximity beacons in order for achieving more accurate results.

### 8.2.1 LOCATION BEACONS FOR TRILATERATION

For the purposes of this project, proximity beacons were used to develop the system. However, perhaps a greater accuracy would be achieved using location beacons. The differences between the location beacons and the proximity beacons are that the location beacons incorporate Bluetooth 4.0 into the hardware increasing the maximum range to 200 meters with a power of +10dBm<sup>(20)</sup>. This allows for a much more accurate distance estimation from the beacons. However, a significant point to note is that for the purposes

of the lab, it may still not necessarily be ideal due to the large amount interference due to the computers and other individuals. The accuracy is as good as 1 meter in small rooms - Lab 2 can be considered a relatively small room - but due to interference from the computers and a greater crowd density, the accuracy may decrease to 4 meters <sup>(21)</sup>. An accuracy of up to 1 meter may still not be good enough for the relatively small dimensions of Windows Lab 2, but it is certainly a step up from the accuracy received by proximity beacons.

Furthermore, Estimote have developed a new API for the location beacons known as “Indoor Location”. Perhaps the most significant aspect of this SDK is the fact that it is able to transmit locations in the background <sup>(22)</sup>. This essentially makes a much more pervasive application than with ranging for proximity beacons in the foreground. Thus, the application does not depend on the user opening his/her phone when in the labs to transmit coordinates to the central database. However, a significant point to note is that this SDK is only available for iOS and as mentioned above, if the application is producing results accurate to 1m to 4m, the application still would not be good enough for the dimensions of the lab. More tests such as the ones conducted in Section 7 for trilateration would have to be conducted to evaluate the effect of interference from the environment of the Windows Lab 2.

#### 8.2.2 LOCATION BEACONS FOR BACKGROUND MONITORING

With respect to the background monitoring application, Estimote’s location beacons provide granular data in the background via “location” packets. The packets allow events to occur for three different categories of distances, namely 1.5m, 7m, and 15m from the beacon. Thus, this eliminates the issue of falsely placing an individual in another beacon. This also allows the user to be placed more accurately in the lab <sup>(23)</sup>.

Furthermore, the reliability of “onExitedRegion” events has been significantly increased and is triggered as soon as the user has exited the zone as opposed to falsely triggering exit events due to false spikes in transmit power. This is due to the fact that the granular data received from location packets involve determining the distance between the user and the beacon. Theoretically, this should be ideal. However, in practice, there is no certainty to due to interference <sup>(23)</sup>.

### 8.3 OTHER TECHNOLOGIES

This subsection describes the possibility of using technologies other than Bluetooth for achieving a more complete and accurate system.

#### 8.3.1 ULTRA-WIDE BAND RADIO TECHNOLOGY

In order to accurately position an individual in the room, it has been noticed that Bluetooth is very prone to interference which causes an inaccuracy in distance estimations based on the RSSI values received by the phone. Thus, other technologies that avoid interference can perhaps be used to correctly position the individual within the room such as Ultra-Wideband (UWB) based radio technology.

Essentially, UWB makes use of short impulse transmissions with extremely high peaks and valleys. Thus, when listening for a signal, it is easy to determine when the signal has been sent and received. Instead of using signal strength to estimate the distance between two devices, UWB uses the time taken for the signal to reach from one device to another to estimate how far the device receiving the transmission is from the device sending the transmission. Given that UWB uses an extremely high amplitude when sending its signal, the signal would not be disrupted by a noise environment given that the signal's amplitude would surpass that of the noise allowing the device receiving the signal to determine when the signal has been received <sup>(24)</sup>.

The major drawback with UWB, however, is that UWB is not as widely renowned as Bluetooth. Many devices lack UWB capability, and due to strict regulations conducted by the National Telecommunications and Information Administration (NTIA), many countries prohibit UWB frequency allocation unless it adheres to the NTIA's regulations or the respective country's requirements <sup>(25)</sup>.

Furthermore, Estimote have also developed Location Beacons that use UWB technology. The technology essentially works by sending UWB signals from one beacon to another to determine how far each beacon is from one another. The beacons then automatically develop a floorplan which can be used with Estimote's Indoor Location API on iOS. The beacons are still currently on pre-order and are yet to be released <sup>(26)</sup>. This could be

incorporated into the application in order to accurately determine the position of users within the lab.

### 8.3.2 MAGNETIC FIELD

With respect to magnetic field technology, smartphones harbor technology that can sense and record changes in the magnetic field of the earth. Thus, the approach to map an individual in an indoor location would be to use the smartphone's built-in compass to detect the magnetic field within the building, which is currently being achieved by IndoorAtlas<sup>(27)</sup>. The accuracy of this approach is nearest to 1 - 2m. Therefore, once again, for the dimensions of Windows Lab 2, this is not necessarily optimal, but it is still an enhancement as opposed to the proximity beacons.

## 9. CONCLUSIONS

In conclusion, for the purposes of trilateration, BLE beacons are still not technologically capable of producing accurate values for positioning an individual. This is mostly due to the fact that the distance estimates of the beacons determined by the application are inaccurate, which can be attributed to interference. The unit tests have indicated that each algorithm used for the system has provided accurate and expected results. Testing the system in a real-life scenarios with the use of beacons, however, has produced discrepancies between the expected values and the values displayed by the application. This is seen in the second set of tests done in Section 7.1.2, page 57. Thus, it can be concluded that BLE proximity beacons are not optimally suited for indoor location and trilateration.

Furthermore, when placing the beacons in different sections of a location of interest, the number of individuals situated in a different section of a location of interest can still be determined based on which beacon's "onEnteredRegion" has been triggered. The proximity beacons, however, used for the purposes of testing still have not provided reliable results due to the fact that false "onExitedRegion" events are triggered, and when the application is in the background of the phone, there is a much larger delayed "onEnteredRegion" trigger event as opposed to when the phone is running in the foreground. However, it has been noticed that Kontakt.io beacons and newer, more expensive phones such as the HTC One M10 and the Samsung Galaxy S7, there is potential with the background monitoring application as there are more reliable and consistent zone values received from these beacons on these phones due to the higher quality in Bluetooth receivers.

The aims of the project will now be assessed and will be compared to that mentioned in the initial report. With respect to pinpointing an individual's location within the lab, the application has not been able to achieve that as mentioned above. However, an alternative application that displays relevant location within a section of the labs has been proposed and is reliable when using Kontakt.io beacons with newer phones.

The process towards achieving the solution involved the following: determining the dimensions of the Windows Lab 2, developing a trilateration algorithm, determining the ideal advertising interval for the beacons, determining the ideal range of coverage for the beacons, determine the ideal positions of the beacons, develop a web client to send and receive data to and from a central database.

The dimensions of the Windows Lab 2 were easily achieved via a tape measure. This served as a fundamental platform towards achieving a functional beacon plan. The trilateration algorithm was also researched and tested to display values as expected.

With respect to determining the ideal advertising interval, this was set to 201ms (0.201 seconds) in order to retrieve a lot of values in a short amount of time. This is due to the fact that a wider range of values can be used to determine the accurate average in hopes that the accurate average is able to remove outliers and produce a more accurate distance estimate. Furthermore, battery life was also taken into account. According to Estimote's battery life estimator, with an advertising interval of below 201ms, the battery life gets severely reduced to less than 100 days, which is not cost-effective. Furthermore, given that the system must take into account the user's movement, having an advertising interval of greater 201ms would lead to a longer time in the DistanceQueue replacing previous values. Thus, essentially, the estimated distances would not be relevant until the user has stopped moving and the distances received are more consistent. Having said that, however, the fact is proximity beacons are still incapable of providing accurate distance estimates.

The ideal range of coverage for the beacons was initially thought to be 14m, which is enough to have three beacons intersect the area of the labs in order to accomplish trilateration. However, given that the beacon's coverage is estimated based on the beacon's transmit power, which is used in distance estimations, it was necessary to maximize this value in order to get a stronger signal on the phone to achieve relatively more accurate distance estimates. Furthermore, given that the coverage was maximized - to a value of about 70m - the positions of the beacons were a bit irrelevant when taking into consideration of avoiding individuals in the hallway. Thus, the check of whether the user is within a set coordinates are made when sending the data to the database which deems whether a user is an "active" user or not.

With respect to determining the ideal beacon positions, the major issue was avoiding interference. However, theoretically, the beacons could have been placed on the roof, but given that the roof has a significant height difference which would also negatively impact the distance estimates, the beacons were placed on the walls at a more negligible height. The positions of the beacons determined in Figure 5 - Figure 7 would work fine if distance estimates from the beacons had been more accurate or if the signals from the beacons had not been interfered with by the computers within the lab. The plan in Figure 8 is also theoretically fine given that the range extends to only 3m in diameter of the lab. However, certain phones with more advanced Bluetooth receivers are still capable of receiving signals from beacons that are further than 3m from the phone.

Reflecting on the initial plans, the initial aims have been quite consistent with the current aims of the project which have also changed throughout the duration of the project. The aim of disallowing duplicates in the system was not necessary and was combatted by adding a UUID for each user, which gets updated with values. Thus, duplicates could never be added. A JavaScript function was called every 5 seconds, which retrieved data from the database consistently. Thus, the aim of constantly retrieving values from the server was achieved. A significant point to note was that a desirable aim, namely the aim of displaying the occupancy of the labs graphically became the main focus of the project in order to research the ideal method to determine the position of the user of the labs.

All in all, as mentioned above, BLE proximity beacons are not optimal for determining the position of the user via trilateration. The proximity beacons, however, are more useful when placing them in different sections of an indoor location in order to determine where in the indoor location a user is situated in. Having said that, however, the proximity beacons are still not reliable enough to determine the occupancy of a location given that the background monitoring is still not reliable due to the fact that there are still false exit events triggered. Perhaps these beacons are more useful for targeted advertisement. For example, if the user walked in a particular section of a store and there is a promotion within that particular section of the store, the user would be able to receive a notification on the phone from the beacon. In addition, other technologies such as location beacons or UWB beacons would provide a more interesting and potentially a more accurate approach to determining the position of a user indoors.

## 10. REFLECTION ON LEARNING

Throughout the duration of this project, I was able to develop myself personally, both academically and mentally. There are several important principles that I have learnt when undertaking a project with a longer duration such as this, and I will expound on this in this section.

Ultimately, the primary solution to this project was to solve the problem of indoor location and increase the precision of GPS. To solve this problem, after choosing the technology (BLE beacons) to represent satellites in GPS but at a much smaller scale, there were steps into achieving this solution. I believe an approach to breaking each task into smaller tasks and solving smaller tasks together to achieve a solution was an essential process to completing the project. However, first and foremost, the issue was to understand the fundamentals of the problem. This allowed me to understand the constraints, and allowed me to approach the problem in a more realistic manner.

After initially conducting a literature analysis, it was quite prominent that the prevalent constraint to this problem was undoubtedly the distance estimates received from each beacon. Once I realized this constraint, I opted to solve this issue with 3 different approaches, namely trying different configurations, trying different algorithms, and increasing the beacons' transmit power. Ultimately, after trying three different approaches to increase the accuracy of the distance estimates received from beacons, I sought to take a different approach, and use a more realistic approach to solving the problem by using proximity and background monitoring as opposed to ranging and trilateration. There are other approaches I would have liked to experiment with, which I have mentioned in Section 8.1.2, page 71. However, due to time constraints and constraints in receiving individuals to test the platform, it was difficult to achieve.

Furthermore, another essential skill that I believe I have developed through this project is adaptation. As mentioned in Section 9, page 71, my initial plan had a different focus for the project, which I decided to adapt and develop a more interesting solution by using graphical and more precise feedback for monitoring occupancy. Thus, the fundamental focus of the project changed from simply occupancy monitoring to developing an indoor



location solution. I believe this corresponds to the scrum methodology of approaching a project given that my project was constantly changing<sup>(28)</sup>, namely moving from occupancy to indoor location. In addition, when developing a solution for indoor location, the project took different approaches to achieving an ideal solution by moving on from ranging/trilateration to background monitoring.

Expounding within the spectrum of professionalism, I was always transparent with the state of the project with my supervisor. In each weekly meeting, I had discussed with my supervisor what I had achieved along with what I plan to achieve. Moreover, in order to allow my supervisor to achieve a stronger grasp of the system, I had constantly prepared documents with details regarding the achievements and the progression of the project. This also enhanced my communication skills due to the fact that initially I had a difficult time explaining to my supervisor what I had achieved along with what I plan to achieve without a clearly written document. Consistency is another key principle throughout the duration of the project. This confirms directly with having a minor goals to achieve a major goal. Although I dedicated between 30 - 50 hours each week to the project, it was still not enough to try other approaches such as incorporating filters into the system to achieve a higher accuracy. However, consistency allowed me to build 3 different approaches and test whether they produce optimal results in a real-life scenario. Theoretically, the approaches work as seen with unit testing in Section 6, starting from page 47.

Although the project had not achieved what I would have liked to achieve, and I have not provided a solution to determine the precise location of an individual indoors, I strongly believe that I have learnt a lot from completing this project. There is still much I would have liked to achieve, but I am satisfied with the fact that I was able to research alternate approaches in order to achieve a more realistic solution with the BLE technology. Personally, I have developed a much stronger understanding of the Android system via doing this project in addition to keeping things in mind such as anonymity which is also essential to this project. Other fundamental constituents of every Android application such as Activities and protected methods such as “onCreate()” and “onPause()” are also things I had to learn throughout the project. For example, when an Activity is not in the foreground, the onPause() method is called, and saving significant data such as the UUID. This allowed me to keep the UUID consistent throughout each activity in the application.

Furthermore, with respect to background monitoring, an Application class was needed to be developed to make the system more pervasive.

All in all, I have achieved a lot from this project and developed myself, and although there is still much to be achieved with this project, I believe I was still able to provide interesting results.

## REFERENCES

1. **Lee, Desmond.** What is GPS? *Garmin*. [Online] 2017. [Cited: March 30, 2017.] <http://www8.garmin.com/aboutGPS/>.
2. **Harris, Tom and Brain, Marshall.** How GPS Receivers Work. *How Stuff Works*. [Online] September 25, 2006. [Cited: March 30, 2017.] <http://electronics.howstuffworks.com/gadgets/travel/gps1.htm>.
3. **Tonner, Dominic.** Information Age. *Information Age*. [Online] 2007. [Cited: March 30, 2017.] [https://web.archive.org/web/20071222231740/http://www.information-age.com/article/2001/may/the\\_bluetooth\\_blues](https://web.archive.org/web/20071222231740/http://www.information-age.com/article/2001/may/the_bluetooth_blues).
4. **Silicon Labs.** An Overview of Bluetooth with Low Energy Functionality. *Silicon Labs*. [Online] 23 February 2016. [Cited: 2 May 2017.] <http://www.silabs.com/whitepapers/designing-for-bluetooth-low-energy-applications>.
5. **iBeaconInsider.** 2014: The Year of the Beacon. *iBeaconInsider*. [Online] 2014. [Cited: March 30, 2017.] <http://www.ibeacon.com/2014-the-year-of-the-beacon/>.
6. —. What is iBeacon? A guide to iBeacons. *iBeaconInsider*. [Online] 2014. [Cited: March 30, 2017.] <http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/>.
7. **Estimote.** Beacon Tech Overview. *Estimote Developer Docs*. [Online] 2015. [Cited: March 30, 2017.] <http://developer.estimote.com/ibeacon/>.
8. —. Estimote Android SDK. *Estimote Android SDK*. [Online] 2017. [Cited: February 16, 2017.] <https://github.com/Estimote/Android-SDK>.
9. —. Estimote Developer Docs: Ranging Beacons. *Estimote Developer Docs*. [Online] 2015. [Cited: February 20, 2017.] <http://developer.estimote.com/android/tutorial/part-3-ranging-beacons/>.
10. —. Estimote Developer Docs: Background Monitoring. *Estimote Developer Docs*. [Online] 2015. [Cited: February 4, 2017.] <http://developer.estimote.com/android/tutorial/part-2-background-monitoring/>.
11. **Puchta, Ola.** Beacons. *Estimote Community Portal*. [Online] 2017. [Cited: March 18, 2017.] <https://community.estimote.com/hc/en-us/articles/201636913-What-are-Broadcasting-Power-RSSI-and-other-characteristics-of-beacon-s-signal->.
12. **Estimote.** Class Utils. *Estimote Java Documentation*. [Online] [Cited: February 23, 2017.] <https://estimote.github.io/Android-SDK/JavaDocs/com/estimote/sdk/Utils.html#computeAccuracy-com.estimote.sdk.Beacon->.
13. **Trilateration Based localization Algorithm for Wireless Sensor Network. Oguejiofor, O.S, et al.** 10, September 2013, International Journal of Science and Modern Engineering (IJISME), Vol. 1, p. 23.
14. **Salauiyou, Alex.** Stack Overflow. *Stack Overflow*. [Online] 2015. [Cited: March 15, 2017.] <http://stackoverflow.com/questions/29910312/algorithm-to-get-all-the-combinations-of-size-n-from-an-array-java>.

15. **InvestopediaStaff**. Standard Error. *Investopedia*. [Online] 2016. [Cited: 18 3, 2017.] <http://www.investopedia.com/terms/s/standard-error.asp>.
16. **Neal, David**. Calculating z-scores. *people.wku.edu*. [Online] 2003. [Cited: March 18, 2017.] <http://people.wku.edu/david.neal/statistics/misc/zscore.html>.
17. **Lee, Matthew**. Healthy Body Water Percentage. *Livestrong*. [Online] 2010. [Cited: April 18, 2017.] <http://www.livestrong.com/article/340756-healthy-body-water-percentage/>.
18. **Bulten, Wouter**. Kalman filters explained: Removing noise signals from RSSI. *wouterbulten*. [Online] 2015. [Cited: April 16, 2017.] <https://wouterbulten.nl/blog/tech/kalman-filters-explained-removing-noise-from-rssi-signals/>.
19. *Particle Filters for Positioning, Navigation and Tracking*. **Gustafsson, Fredrik, et al.** 2001, [https://pdfs.semanticscholar.org/3bb4/1e674eb39ad04d060409f75947b25e1f958c.pdf?\\_ga=1.52967265.893013915.1493394676](https://pdfs.semanticscholar.org/3bb4/1e674eb39ad04d060409f75947b25e1f958c.pdf?_ga=1.52967265.893013915.1493394676). LiTH-ISY-R-2333.
20. **Estimote Blog Team**. Updated Location Beacons: 200 m range, NFC, new APIs, and more. *Reality Matters: Estimote Team Blog*. [Online] 2016. [Cited: April 19, 2017.] <http://blog.estimote.com/post/149362004575/updated-location-beacons-200-m-range-nfc-new>.
21. **Estimote**. What is Estimote Indoor Location SDK. *Estimote Developer Docs*. [Online] 2015. [Cited: April 18, 2017.] <http://developer.estimote.com/indoor/>.
22. **Matsuk, Liliya and Krawiec, Piotr**. Physical-world context 101: Proximity vs Location. *Reality Matters: Estimote Team Blog*. [Online] 2016. [Cited: April 18, 2017.] <http://blog.estimote.com/post/148730623715/physical-world-context-101-proximity-vs-location>.
23. **Alex, Bartek &**. Estimote Monitoring - more reliable notifications for beacon-based apps. *Reality Matters: Estimote Team Blog*. [Online] 2016. [Cited: April 18, 2017.] <http://blog.estimote.com/post/149409589655/estimote-monitoring-more-reliable-notifications>.
24. **Connell, Ciaran**. What's The Difference Between Measuring Location By UWB, Wi-Fi, and Bluetooth? *Electronic Design*. [Online] 2016. [Cited: April 19, 2017.] <http://www.electronicdesign.com/communications/what-s-difference-between-measuring-location-uw-b-wi-fi-and-bluetooth>.
25. *Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances*. **Alarifi, Abdulrahman, et al.** May 2016, *Sensors — Open Access Journal*.
26. **Estimote Blog Team**. Estimote Beacons with UWB can now automatically create floor plans. *Reality Matters: Estimote Team Blog*. [Online] 2017. [Cited: April 19, 2017.] [http://blog.estimote.com/post/154460651570/estimote-beacons-with-uw-b-can-now-automatically?gclid=Cj0KEQjwxPbHBRCdxJLF3qen3dYBEiQAMRyxS\\_b8TdAr4IBEeEhDAqL8-W5dC59M2sjrxA1OKgiOmtQaAtl-8P8HAQ](http://blog.estimote.com/post/154460651570/estimote-beacons-with-uw-b-can-now-automatically?gclid=Cj0KEQjwxPbHBRCdxJLF3qen3dYBEiQAMRyxS_b8TdAr4IBEeEhDAqL8-W5dC59M2sjrxA1OKgiOmtQaAtl-8P8HAQ).
27. **Indoor Atlas**. Indoor Atlas: How it works. *Indoor Atlas*. [Online] 2016. [Cited: April 20, 2017.] <http://www.indooratlas.com/how-it-works/>.
28. **Unknown**. What is scrum? *Version One*. [Online] 2014. [Cited: April 25, 2017.] <https://www.versionone.com/agile-101/what-is-scrum/>.

## APPENDICES

### APPENDIX A: RESULTS FOR TRILATERATION

#### APPENDIX A.1: RESULTS AFTER RUNNING 3-BEACON TRILATERATION ACCORDING TO FIGURE 5

User	Expected x and y-coordinates		On screen x and y-coordinates	
A	5.01	7.66	3.9	4.65
B	5.01	6.89	7.53	8.67
C	5.01	5.05	3.19	5.86
D	5.6	10	1.49	5.01
E	5.6	9	4.06	6.58
F	5.6	5.24	3.75	6.51

#### APPENDIX A.2: RESULTS AFTER RUNNING 3-BEACON TRILATERATION ACCORDING TO FIGURE 6

User	Expected x and y-coordinates		On screen x and y-coordinates	
A	5.01	7.66	5.87	10.94
B	5.01	6.89	2.31	6.9
C	5.01	5.05	1.52	5.63
D	5.6	10	2.98	6.2
E	5.6	9	0.07	6.32
F	5.6	5.24	3.12	6.2

#### APPENDIX A.3: RESULTS AFTER RUNNING 6-BEACON TRILATERATION

User	Expected x and y-coordinates		On screen x and y-coordinates	
A	1.98	8.12	3.51	5.03
B	1.98	4.43	2.95	5.21
C	3.17	9.02	5.6	6.37
D	3.17	6.24	2.47	6.24
E	5.6	9.54	4.37	6.01
F	5.6	6.56	3.24	8.41

## APPENDIX A.4: RESULTS AFTER RUNNING 3-CLOSEST BEACON TRILATERATION

User	Expected x and y-coordinates		On screen x and y-coordinates	
A	0.61	8.44	0.13	5.67
B	2.27	5.3	1.75	4.84
C	2.27	2.11	4.27	5.3
D	3.69	1.86	3.45	2
E	5.6	4.48	2.83	7.75
F	3.32	5.92	4.48	8.82

## APPENDIX B: EUCLIDEAN DISTANCE EQUATION

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = \text{Distance}$$

## APPENDIX C: LOG FILE DISPLAYING LOSS OF SIGNAL IN BEACONS

```

05-10 17:20:10.931 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:11.568 12624-12624/com.example.rohan.finalyearproject I/System.out: Key:
207:1 Distance: 0.8624894056112784
05-10 17:20:11.823 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:12.500 12624-12624/com.example.rohan.finalyearproject I/System.out: Key:
207:1 Distance: 0.8624894056112784
05-10 17:20:12.728 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:13.405 12624-12624/com.example.rohan.finalyearproject I/System.out: Key:
207:1 Distance: 0.8624894056112784
05-10 17:20:13.666 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:14.339 12624-12624/com.example.rohan.finalyearproject I/System.out: Key:
207:1 Distance: 0.8624894056112784
05-10 17:20:14.584 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:15.545 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:16.431 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:17.343 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:18.255 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:19.170 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:20.099 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:21.003 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:21.913 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:22.809 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:23.741 12624-12640/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:24.647 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:25.547 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:26.244 12624-12634/com.example.rohan.finalyearproject W/art: Suspending all
threads took: 5.793ms
05-10 17:20:26.457 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:20:27.381 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5

```



[illegible]

05-10 17:20:51.077 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:51.984 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:52.900 12624-12640/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:53.811 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:54.682 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:55.607 12624-12640/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:56.519 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:57.446 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:58.360 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:20:59.310 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:00.249 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:01.150 12624-12640/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:02.057 12624-12640/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:02.981 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:03.860 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:04.832 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:05.763 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:06.666 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:07.586 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:08.518 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:09.440 12624-12640/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:10.335 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:11.242 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:12.185 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5  
05-10 17:21:13.083 12624-12641/com.example.rohan.finalyearproject  
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5

```
05-10 17:21:13.990 12624-12641/com.example.rohan.finalyearproject
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
05-10 17:21:14.661 12624-12624/com.example.rohan.finalyearproject I/System.out: Key:
207:1 Distance: 0.96
```

## APPENDIX D: NUMERICAL RESULTS FOR DISTANCE VALUES OF BEACONS AT 1M

	Estimote at 1m			Kontakt.io at 1m		
Run #	No Object	Human in between	Computer in between	No Object	Human in between	Computer in between
1	0.96	3.284268713	3.822944277	0.820672529	3.326217969	2.238753215
2	0.96	3.284268713	3.822944277	0.820672529	3.326217969	2.238753215
3	0.96	3.284268713	3.822944277	0.820672529	3.326217969	1.988237887
4	0.96	3.284268713	3.822944277	0.820672529	3.326217969	1.988237887
5	0.96	3.284268713	3.822944277	0.820672529	3.326217969	1.988237887
6	0.96	3.284268713	3.822944277	0.820672529	3.326217969	1.988237887
7	0.96	3.284268713	3.822944277	0.820672529	3.326217969	1.763834718
8	0.96	3.284268713	3.822944277	0.820672529	3.326217969	1.763834718
9	0.862489406	3.284268713	3.822944277	0.820672529	3.72488325	1.763834718
10	0.862489406	3.284268713	3.822944277	0.820672529	3.72488325	1.605682232
11	0.862489406	3.284268713	3.822944277	0.820672529	3.72488325	1.605682232
12	0.862489406	3.284268713	3.822944277	0.820672529	3.72488325	1.605682232
13	0.862489406	3.284268713	3.822944277	0.820672529	3.72488325	1.763834718
14	0.862489406	3.284268713	3.822944277	0.820672529	3.72488325	1.763834718
15	0.862489406	3.284268713	3.822944277	0.820672529	4.337421422	1.763834718
16	0.862489406	3.284268713	3.822944277	0.820672529	4.337421422	1.763834718
17	0.862489406	3.284268713	3.822944277	0.820672529	4.337421422	1.988237887
18	0.862489406	3.284268713	2.801041182	0.820672529	4.337421422	1.988237887
19	0.862489406	3.284268713	2.801041182	0.820672529	4.337421422	1.988237887
20	0.862489406	3.284268713	2.801041182	0.820672529	4.337421422	1.988237887
21	0.862489406	3.284268713	2.801041182	0.820672529	4.337421422	1.988237887
22	0.862489406	3.284268713	2.801041182	0.820672529	4.337421422	1.988237887
23	0.862489406	3.284268713	2.801041182	0.820672529	4.337421422	1.988237887
24	0.862489406	3.822944277	2.801041182	1.02	4.337421422	1.763834718
25	0.862489406	3.822944277	2.801041182	1.02	3.72488325	1.763834718
26	0.862489406	3.822944277	2.801041182	1.02	3.326217969	1.763834718
27	0.862489406	3.822944277	2.801041182	1.02	3.326217969	1.763834718
28	0.862489406	3.822944277	2.801041182	1.02	3.326217969	2.238753215
29	0.862489406	3.822944277	2.801041182	1.02	4.78073939	2.238753215
30	0.862489406	3.822944277	2.801041182	1.02	4.78073939	2.238753215
31	0.862489406	3.822944277	2.801041182	1.02	4.78073939	2.238753215
32	0.862489406	3.822944277	2.801041182	1.02	4.78073939	2.238753215
33	0.862489406	3.822944277	2.801041182	1.02	4.337421422	2.238753215
34	0.862489406	3.822944277	2.801041182	1.02	4.337421422	2.451276961
35	0.862489406	3.822944277	2.801041182	1.02	4.337421422	2.451276961
36	0.862489406	3.822944277	2.801041182	1.02	4.337421422	1.763834718
37	0.862489406	3.822944277	2.801041182	1.02	4.337421422	1.763834718

38	0.862489406	3.822944277	2.801041182	1.02	4.337421422	1.763834718
39	0.862489406	3.822944277	2.801041182	1.02	4.337421422	1.763834718
40	0.862489406	3.822944277	3.284268713	1.02	2.717066037	1.988237887
41	0.862489406	3.822944277	3.284268713	1.02	2.717066037	1.988237887
42	0.862489406	3.822944277	3.284268713	1.02	2.717066037	1.988237887
43	0.862489406	3.822944277	3.284268713	1.02	2.717066037	1.988237887
44	0.862489406	3.822944277	3.284268713	1.02	2.717066037	1.763834718
45	0.862489406	3.822944277	3.284268713	1.02	2.717066037	1.763834718
46	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
47	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
48	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
49	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
50	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
51	0.96	2.933973046	3.284268713	1.02	3.326217969	1.988237887
52	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
53	0.96	2.933973046	3.284268713	1.02	3.326217969	1.763834718
54	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
55	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
56	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
57	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.988237887
58	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.988237887
59	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.988237887
60	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.988237887
61	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.988237887
62	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
63	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
64	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
65	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
66	0.96	2.933973046	1.979177564	0.820672529	3.326217969	1.763834718
67	0.96	2.933973046	1.979177564	0.820672529	3.72488325	1.763834718
68	0.96	2.933973046	1.979177564	0.820672529	3.72488325	1.763834718
69	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
70	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
71	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
72	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
73	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
74	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
75	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.988237887
76	0.96	3.284268713	1.979177564	0.820672529	3.72488325	1.605682232
77	0.96	3.284268713	1.979177564	0.820672529	4.337421422	1.605682232
78	0.96	3.284268713	2.165621183	0.820672529	4.337421422	1.605682232
79	0.96	3.284268713	2.165621183	0.820672529	4.337421422	1.605682232

80	0.96	3.284268713	2.165621183	0.820672529	4.337421422	1.763834718
81	0.96	3.284268713	2.165621183	1.02	4.337421422	1.763834718
82	0.96	3.284268713	2.165621183	1.02	4.337421422	1.763834718
83	0.96	3.284268713	2.165621183	1.02	3.72488325	1.763834718
84	0.96	3.284268713	2.165621183	1.02	3.72488325	1.763834718
85	0.96	3.284268713	2.165621183	1.02	3.72488325	1.763834718
86	0.96	2.933973046	2.165621183	1.02	3.72488325	1.763834718
87	0.96	2.933973046	2.165621183	1.02	3.72488325	1.763834718
88	0.96	2.933973046	2.165621183	1.02	3.72488325	1.988237887
89	0.96	2.933973046	2.165621183	1.02	3.72488325	1.988237887
90	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
91	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
92	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
93	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
94	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
95	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
96	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
97	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
98	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
99	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
100	0.96	2.933973046	2.165621183	1.02	3.326217969	1.988237887
<b>Average</b>	0.92392108	3.269664984	2.699022694	0.920336265	3.676106171	1.896511668