



**Cairo university**

**Faculty of Engineering**

**Department of Electronics and Electrical**

**Communication Engineering**



**Smart City**  
**Smart Indoor Navigation and Context Based**  
**Messaging over BLE**

# **Blue-tooth Low Energy Indoor Navigation System**

---

**By**

Hesham Mostafa Nouredin

Maryam Osama Omar

Mayada Samir Mohamed

Nahed Sayed Shaaban

Noha Nabih Ahmed

Reham Sami Hussein

Samar Magdy Elhadidy

Under supervision of

**Dr. Yasmine Fahmy**

**Dr. Hassan Mostafa**

A Graduation Project Report Submitted to the Faculty of  
Engineering at Cairo University in Partial Fulfillment of the  
Requirements for the Degree of Bachelor of Science

in Electronics and Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

July 2017

## Table of Contents

Table of Contents.....	III
List of Tables.....	VI
List of Figures.....	VII
List of Symbols and Abbreviations.....	IX
Acknowledgments.....	X
Abstract.....	XI
Chapter 1: Introduction.....	1
1.1 Motivation.....	1
1.2 Objectives.....	2
1.2.1 High accuracy.....	2
1.2.2 Low-cost.....	2
1.2.3 Intuitive user interface (UI).....	2
1.2.4 Maps.....	2
Chapter 2: System Design.....	3
2.1 Over view system.....	3
2.2 specification of the system.....	4
2.2.1 Beacons range , distance between them.....	4
2.2.2 Special cases regarding number of received packets from Beacons.....	4
2.2.3 Mobile application.....	4
2.2.4 Obstacle detection.....	4
Chapter 3: The Concept of Indoor-Navigation.....	5
Chapter 4: Technologies Used For Indoor-Navigation.....	8
4.1 Wi-Fi.....	8
4.1.1 Accuracy of Wi-Fi for indoor-navigation.....	8

4.1.2	Setting up an indoor-navigation system using Wi-Fi.....	8
4.1.3	Advantages and disadvantages of indoor-navigation using Wi-Fi.....	9
4.2	Blue-tooth Low Energy.....	10
4.2.1	Advantages and disadvantages of indoor-navigation using BLE.....	10
Chapter 5:	Components Used.....	12
5.1	Comparison between Beacons types.....	12
5.2	Estimote iBeacons.....	12
5.2.1	How does it work.....	13
5.2.2	Types of iBeacons.....	14
5.2.3	Sensors that iBeacons have.....	15
5.3	Raspberry-pi.....	15
5.3.1	Definition of Raspberry-pi.....	15
5.4	the connection between iBeacons and Raspberry-Pi.....	16
Chapter 6:	Indoor-Navigation Methods.....	20
6.1	Trilateration.....	20
6.2	Particle Filter.....	23
6.2.1	Particle Filter Nomenclature.....	23
6.2.2	Particle Filter Sequential Importance Sampling.....	23
6.3	A* Algorithm.....	28
6.3.1	Searching area.....	28
6.3.2	Starting the Search.....	29
6.3.3	Path Scoring.....	30
6.3.4	Continuing the Search.....	33
6.3.5	Summary of the A* Method.....	37
6.3.6	Map Constructions.....	38
Chapter 7:	Web Server Development.....	40

7.1 Back End.....	40
7.1.1 LAMP.....	41
7.2 Front End.....	47
7.2.1 User Interface.....	47
7.2.2 Asset management.....	49
7.2.3 Get statistics.....	50
Chapter 8: Mobile Application.....	51
8.1 Sign up page.....	52
8.2 Navigation page.....	53
Chapter 9: Conclusion and Future work.....	55
9.1 Conclusion.....	55
9.2 Future work:.....	55
References.....	57

## List of Tables

Table 5.1: Comparison of Beacons.....	12
Table 5.2: iBeacon ID.....	16
Table 5.3: Names of Beacons in Database.....	18

## List of Figures

Figure 2.1: System overview.....	3
Figure 3.1: GPS Application.....	5
Figure 5.1: Estimote Beacons.....	13
Figure 5.2: Code to filter packets of Beacons.....	19
Figure 6.1: Trilateration Estimation.....	20
Figure 6.2 Trilateration flow chart.....	22
Figure 6.3: Particle filter flow chart.....	24
Figure 6.4: Diagram that illustrates the concept of particle filter.....	27
Figure 6.5 :A star step 1.....	28
Figure 6.6 :A star step 2.....	30
Figure 6.7 :A star step3.....	32
Figure 6.8 :A star step 4.....	34
Figure 6.9 :A star step 5.....	35
Figure 6.10 :A star step 6.....	36
Figure 6.11:A star step 7.....	37
Figure 6.12:Scaled map.....	39
Figure 7.1: Server side vs Client side.....	40
Figure 7.2:LAMP.....	41
Figure 7.3: Direction of connection between server and client.....	42
Figure 7.4:LAMP modules.....	42

Figure 7.5:Admin’s interface .....	43
Figure 7.6:Admin’s interface for beacons locations.....	44
Figure 7.7:Admin’s data stored.....	44
Figure 7.8:Admin’s data stored of beacons locations.....	45
Figure 7.9:Registration and log in flow chart.....	45
Figure 7.10:Selection from drop down list flow chart.....	46
Figure 7.11:Map flow chart.....	47
Figure 7.12:Log in User interface.....	48
Figure 7.13:Main page at user interface.....	48
Figure 7.14:Navigating Map at user interface.....	49
Figure 7.15:Asset Management.....	50
Figure 7.16:Table of Statistics.....	50
Figure 8.1:Mobile app log in interface.....	53
Figure 8.2:Mobile app main interface.....	53
Figure 8.3:Mobile app navigation interface.....	53
Figure 8.4:Overview for android App.....	54
Figure 9.1:Anomoly detection 1.....	56
Figure 9.2:Anomoly detection 2.....	56



## List of Symbols and Abbreviations

BLE	Blue-tooth Low Energy
RSSI	Received signal strength indicator
GPS	Global Positioning System
NS	Number of particles
UI	User Interface

## **Acknowledgments**

We would like to thank everyone who helped us during the graduation project, starting with endless thanks for all professors in our college, Also we would like to express our gratitude for our supervisors **Dr. Yasmine Fahmy, Dr. Hassan Mostafa , Eng. Ayman Hendawy , Eng. Ahmed Rashad** and **our colleague**, from Faculty of Engineering Ain Shams University, Ahmed Ayman who didn't keep any effort in encouraging us to do a great job, providing our group with valuable information and advises to be better each time. Thanks for the continuous support and kind communication which had a great effect regarding to feel interesting about what we are working on.

It is difficult to acknowledge everyone who was involved in the preparation of this project by name. Nevertheless, we appreciate their help no matter how simple it might have been.

Finally, we owe our colleagues great thanks for supporting us through our college years and for making those years the best years of our life.

## **Abstract**

Navigation entails the continuous tracking of the user's position and his surroundings for the purpose of dynamically planning and following a route to the user's intended destination. The Global Positioning System (GPS) made the task of navigating outdoors relatively straightforward, but due to the lack of signal reception inside buildings, navigating indoors has become a very challenging task. However, a new application will be introduced to solve this problem which is based on Bluetooth Low Energy "BLE".

# **Chapter 1: Introduction**

Navigation is the process of accurately establishing the user's position and then displaying directions to guide them through feasible paths to their desired destination. The Global Positioning System (GPS) is the most common and the most utilized satellite navigation system. Almost every aircraft and ship in the world employs some form of GPS technology. In the past few years, smart phones have evolved to contain a GPS unit, and this has given rise to location-based mobile applications such as geofencing and automotive navigation for the common user. However, GPS has its limitations. In particular we are concerned with the lack of GPS signal reception in indoor environments. GPS satellites fail to deliver a signal to a device if there is a direct obstruction on its path. Therefore we have to consider alternate methods of achieving indoor navigation on a smart phone.

## **1.1 Motivation**

Our motivation for this project stems from the fact that people are increasingly relying upon their smart phones to solve some of their common daily problems. One such problem that smart phones have not yet completely solved is indoor navigation.

An indoor navigation application would certainly benefit users who are unfamiliar with a place. Tourists, for instance, would have a better experience if they could navigate confidently inside a tourist attraction without any assistance. In places such as museums and art galleries, the application could be extended to plan for the most optimal or 'popular' routes. Such a system could also be integrated at airports to navigate passengers to their boarding gates. Similarly an indoor navigation system could also benefit local users who have previously visited the location but are still unaware of the whereabouts of some of the desired items. These include supermarkets, libraries and shopping malls. The application could also benefit clients who install the system by learning user behaviors and targeting advertisements at specific locations.

## **1.2 Objectives**

The objective of this project was to build a robust and flexible BLE based indoor navigation system that meets the following criteria:

### **1.2.1 High accuracy**

The application should locate users' position indoor and guide them to desired destination.

### **1.2.2 Low-cost**

The application should not require any expensive infrastructural changes to obtain accurate positioning data. Clients will not be interested in large investments unless they financially benefit from it. Beacons are low cost and can be replaced after few years. There is also some Beacons that have long life time and batteries can be changed easily.

### **1.2.3 Intuitive user interface (UI)**

The application should have an easy-to-use

UI that displays navigation hints correctly based on the user's current state. The application should also take into account the obstacles surrounding the user to avoid displaying any incorrect hints. For instance, it should not tell users to go straight if there is an obstacle immediately ahead of them.

### **1.2.4 Maps**

The application should be able to navigate the user while loading the map of the specific floor from Database .

## Chapter 2: System Design

### 2.1 Over view system

Firstly the scanner ( the Mobile phone ) reads packets from beacons then send it to the server to pass through the following steps:

Step 1: Trilateration to know the current location of the user but not accurate .

Step 2: Particle Filter to increase the accuracy of the current location using resembling technique.

Step 3: store the point (current location) in database ,all Beacons location and destination names.

Step 4: A\* algorithm reads from database the destination of the user and detects the shortest path.

Step 5: Web server reads the current location from database and draw it ,display Maps and shortest path .

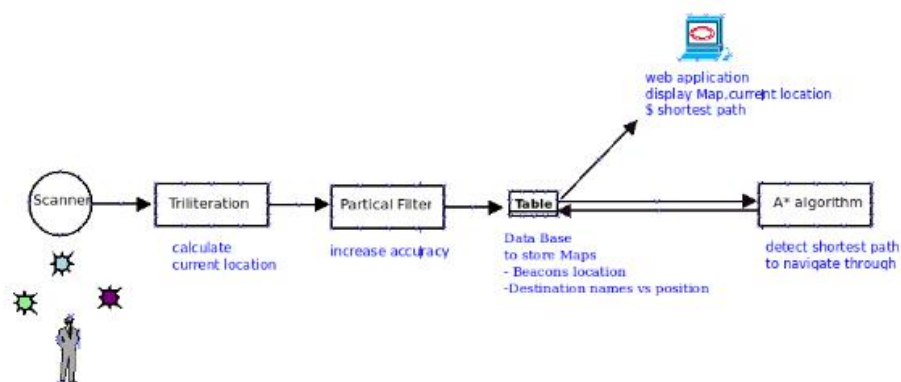


Figure 2.1

## **2.2 specification of the system**

A system specification describes the operational and performance requirements of a system. It uses a combination of AI techniques with Bluetooth and Web server to accurately guide users to their desired destination. Our indoor navigation solution required the study and development of five individual components prior to their integration:

1-Resolution and Scale

2-Beacons Range ,Distance between them

3-Special cases regarding number of received packets from Beacons

4-Mobile Application

5-Obstacle Detection

### **2.2.1 Beacons range , distance between them**

Proximity 60 m and Location 180 m

### **2.2.2 Special cases regarding number of received packets from Beacons**

There is an critical error we handled it which is If the scanner received less than 3 points trilateration will ignore these readings and take the previous three readings of packets

### **2.2.3 Mobile application**

This application is used on smart phones or tablets,to be used as user interface from which user can open the map of location he is present at and navigate.It will also work as scanner.

### **2.2.4 Obstacle detection**

Our application detects obstacles in the environment in real-time using the A\* algorithm The purpose of this task was to avoid giving users directions towards a non-feasible path.

## Chapter 3: The Concept of Indoor-Navigation

During the last decade, the concept of location and navigation in outdoor areas has become fairly common. An example of this is the use of Global Positioning System (GPS) signals to navigate the streets of a city. With the large increase in mobile device production, more people have access to mobile applications developed for this purpose like figure (2).



Figure 3.1

On the other hand, micro-location, also referred as indoor navigation, is still considered to be in its early development stages. One might think that by now, with all the advances in geolocation, such a simple concept as navigating inside a building would have been already addressed. This does not mean there has not been extensive research going on for several years, but no definite method



for such purpose has been agreed up on yet.

In order to find out one's position inside an enclosed space, some kind of reference is needed. GPS satellite signals cannot be received indoors, which means that indoor positioning systems must rely on other kinds of technologies to function. Usually, reference data in the form of radio signals, infrared, ultrasound or magnetic field readings is used for this purpose. Once collected, this data can be interpreted and used in positioning algorithms, providing different levels of accuracy.

For the purpose of developing an indoor navigation system that is easy to implement, as well as widely available to the public, one must consider using common and simple technologies. Positioning methods such as those using magnetic field readings offer great accuracy but their implementation is difficult and pricey, not to mention, they cannot be easily made available to the public. On the other hand, Radio Frequency (RF) signals, such as such as those emitted by Wi-Fi Access Points (AP) and Bluetooth devices, are fairly common and may be also used as reference for indoor navigation systems. Nowadays most mobile devices are equipped with both Wi-Fi and Bluetooth adapters which enable them to pick up these signals and connect, if necessary, to the transmitting devices. By taking advantage of this reality, mobile applications aimed to indoor navigation can be developed and made available to everyone through mobile marketplaces.

Just as those relaying on GPS, indoor navigation applications can be also designed to provide additional information, either on request or automatically, related the user's location. A company building or university, for example, can

offer guidance to visitors about the location of a classroom or conference room, including schedules, features, etc. Another good example is an airport that chooses to provide information about flights as soon as the clients walk through the door, as well as ways to check-in and find their luggage. The list goes on and on and the possibilities become more interesting. Being able to provide contextual information can be highly valuable not just for the user but also for companies and services that choose to implement this system. Stores, for instance, can provide information about certain products on sale or available promotions. They can also track, not intrusively, customers to learn about their shopping habits and identify popular products in order to provide a better service. Whether they are for commercial purposes or not, the mobile applications should always ask permission from the user and inform them about any information being collected.

The next chapter provides an introduction to the two most used technologies in indoor navigation, Wi-Fi and Bluetooth. Understanding these technologies is essential in order to choose which one to implement into a mobile application, or any other kind of micro-location system.

## **Chapter 4: Technologies Used For Indoor-Navigation**

### **4.1 Wi-Fi**

Inside buildings Wi-Fi is a good alternative to GPS, which is not available indoors. In most cases it is easy to install a Wi-Fi positioning system (WPS), since Wi-Fi access points already exist in many buildings. The advantage is that for example existing cash register systems, public hot-spots and access points of shops or exhibitors can be used. The user doesn't necessarily have to connect with the Wi-Fi, it is sufficient to have Wi-Fi enabled.

For positioning, the so-called fingerprinting method is used. The strength of the Wi-Fi signals (received signal strength indication, RSSI) and the MAC address (media access control) are significant. There must be an appropriate app installed on the smart phone which calculates the current position based on these data.

#### **4.1.1 Accuracy of Wi-Fi for indoor-navigation**

Accuracy depends on multiple factors, for example the number of available networks, reflections for example in corridors and last but not least shielding through walls, ceilings and your own body. The accuracy of Wi-Fi used for indoor positioning varies from five to 15 meters – depending on the preconditions. Sensor fusion – this means the use of smart phone sensors – can even improve accuracy. A big advantage compared to GPS is that it is possible to determine the current floor level.

#### **4.1.2 Setting up an indoor-navigation system using Wi-Fi**

The calibration of position determination works with a one-time reference measurement in the run-up, where the signal strength of the Wi-Fi networks is determined. For this purpose infsoft offers an app with which the client can easily execute the measurement on his own. This solution is client based, which means an app is necessary.

If a server based solution is more suitable for the project, infsoft's self-developed locator nodes can be used. In this case no app is required, all Wi-Fi capable devices are detected and Wi-Fi tracking (asset tracking of, for example, mobile goods) is possible. Locator nodes also support Bluetooth low energy.

#### **4.1.3 Advantages and disadvantages of indoor-navigation using Wi-Fi**

##### **Advantages :**

- existing Wi-Fi infrastructure can be used
- enabled Wi-Fi is sufficient
- there is a back channel to the client
- large range (up to 150m)
- detects floor level

##### **Disadvantages:**

- relatively inaccurate (5-15m) compared to BLE/RFID
- Wi-Fi client based positioning is not possible with iOS devices – but BLE can be used as an alternative
- application required

## **4.2 Blue-tooth Low Energy**

During the last couple of years, there was a lot going on on the Bluetooth market. The technology itself is not new; the functionality of Bluetooth has been well-known since the 1990s. But it was only in recent years that whole new application scenarios have occurred, originating from the energy saving Bluetooth version BLE (Bluetooth Low Energy, Bluetooth Smart). Apple with its iBeacons and just recently Google with its Eddy stone Beacons have got the market moving furthermore. Bluetooth is another good alternative for indoor positioning and indoor navigation. Bluetooth beacons are able to send out signals, but they can't receive them. They are relatively cheap, have a maximum range of 30 meters indoors. Accuracy is up to one meter. On the one hand they are used in client based solutions, that is to say, positioning via app on the smart phone itself. In this case, Bluetooth must be activated on the device. On the other hand, server based tracking solutions using beacons are possible as well. For positioning in client based applications, several beacons are required. They send out unique signals with which the app determines the position by means of fingerprinting. Based on beacons, it is possible to trigger an action, for example displaying a coupon or information on the smart phone.

### **4.2.1 Advantages and disadvantages of indoor-navigation using BLE**

#### **Advantages:**

- cost-effective, unremarkable hardware
- low energy consumption
- flexible integration into the existing infrastructure (battery-powered or power supply via lamps and the domestic electrical system)
- works where other positioning techniques do not have a signal
- compatible with iOS and Android
- high accuracy compared to Wi-Fi (up to 1m)

**Disadvantages:**

- additional hardware
- app is required for client based solutions
- relatively small range (up to 30m)

We will discuss in this report indoor-navigation using BLE in details .

## Chapter 5: Components Used

### 5.1 Comparison between Beacons types

This comparison shows that the Estimote Beacons (That we used ) is the best choice for this application as high range and low cost and higher battery capacity










Name of beacon	Price for min. pack	Form factor/ size, lw xh	Battery / Capacity, mAh	Transmission power, dBm	Transmission range, m	Advertising interval, ms	USB version	Sensors
AprilBeacon 202 	Upon request	40 x40 x18	Coin L	-30/+4	to 30	600	Available in another model	no
BKON A1 	\$ 30.00 / 1	60 x36.5 x8.75	AAA x 2	-30/+4	100+	100-1285	no	no
Blue-beacon Mini 	3 / € 59.00	70 x50 x29	Coin M	-40/+4 with 4 dmb step	30-150	100 to 5000 with 100 ms step	USB dongle, € 63.00	no
Bluecats 	3 / € 59.99	Round with cat's ears Ø76 x27.80	AA x 2		30/60 (high/low RF)		USB dongle (20/40 m trans. range)	no
BlueBar beacons 	1 / £ 20.99	100 x35 x18	Coin M	-24/0	30-100		Both power adapter and USB dongle	no
Dragonfly v. 2.0 	1 / \$ 29.99	39 x19 x5.8	Coin S		to 100		Direct power via micro-USB	no
EasiBeacon 	3 / € 59.99	Round2.7 x2.7 x0.7	Coin S	-40/4			USB adapter	no
EMBC01 	Upon request starting from \$ 18.25	round30 x10	Coin S		15	100	n/a	EMBC02 comes with accelerometer
					30	500		
					75	1000		
Estimote 	3 / \$ 99	Irregular form	Coin L	-20/+4		5-50	100-2000	Accelerometer and temperature available as options

Table 5.1

### 5.2 Estimote iBeacons

They are essentially tiny, low power computers attached to walls or objects in the physical world. Using proximity technologies they detect human presence and behavior and trigger pre-programmed actions delivering contextual and personalized experiences. Beacons are small, often inexpensive devices that enable more accurate location within a narrow range than GPS, cell tower triangulation and

Wi-Fi proximity. Beacons transmit small amounts of data via Bluetooth Low Energy (BLE) up to 50 meters, and as a result are often used for indoor location technology, although beacons can be used outside as well.

Beacons are typically powered by small batteries, but they can be plugged into an outlet or USB port instead to maintain consistent power. Figure(2) shows the estimate beacons

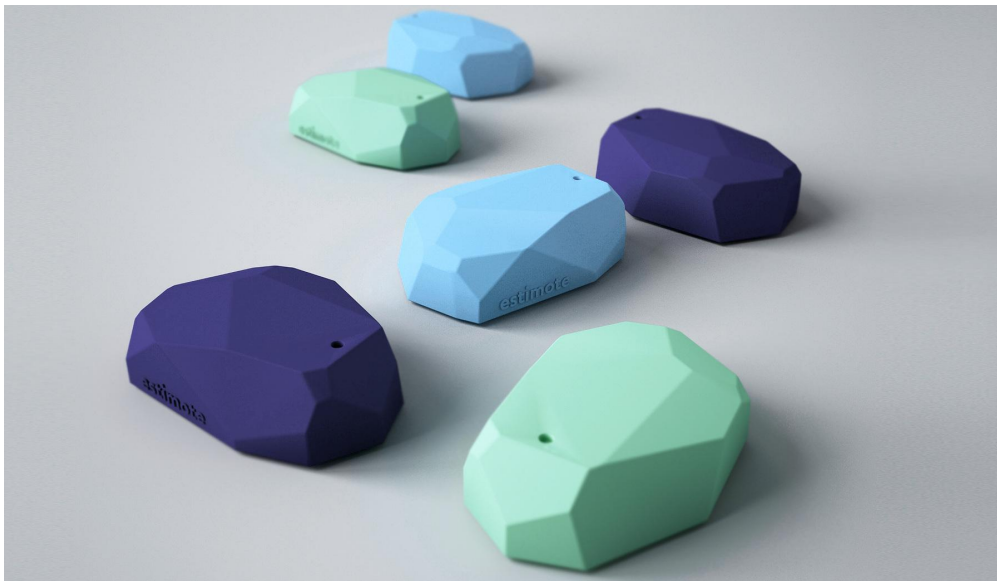


Figure 5.1

### 5.2.1 How does it work

Each beacon has a Bluetooth Low Energy transmitter. It broadcasts tiny radio signals over the air containing unique, location-specific data.

Modern smart phones constantly scan for these signals. If they enter their range an associated app responds with the desired action.

For example, it can fetch content that is tied to a user's profile or micro-location. Apps can also push to the cloud user data or statistics.



Personalized content is displayed as a notification or directly in the app. Nearby screens can also respond with relevant information.

## **5.2.2 Types of iBeacons**

**Estimote Proximity Beacons** and **Estimote Location Beacons** are two separate product lines. This split represents two different approaches, different types of use cases. Let's dig into the difference a little.

### **5.2.2.1 Estimote Proximity Beacons**

The Proximity approach involves “tagging” certain areas (a hotel lobby, a restaurant table, an airport gate, etc.) with a “marker”—a beacon—recognizable by smart phones. You can tune a beacon's broadcasting power to adjust the range of this marker. Your app can simply detect it in the proximity, read its identifier to figure out which marker it is, and act accordingly.

Estimote Proximity Beacons offer long battery life and great range and are usually put on walls, tables, shelves and other static objects. Mobile apps with beacon support can then detect the smart phone coming in and out of range of a beacon, and act on that information. Estimote Proximity Beacons support iBeacon and Eddy stone protocols, offer over three years of battery life on default settings, and have a range of up to 70 meters. They were designed to attach to fixed points in venues, to provide apps with location context.

### **5.2.2.2 Estimote UWB Location Beacons**

are the new generation of our core product coming to you with several improvements. They are tailored for Indoor Location, becoming the first version of hardware dedicated to the location approach.

Both Long-range Location Beacons and Location Beacons with UWB support multi-packet advertising, mesh networking, have built-in sensors, and a range of 200m.

For other use cases, IoT prototyping, or enterprise-grade deployments choose Location Beacons—the most robust beacons on the market.

Estimote UWB Location Beacon, as well as Long-range Location Beacons, can broadcast multiple packets simultaneously. Besides iBeacon and Eddy stone, there are new packets available:

- Estimote Connectivity
- Estimote Location
- Estimote Telemetry

### **5.2.3 Sensors that iBeacons have**

- Motion Sensor
- Temperature Sensor
- Ambient Light Sensor
- Real Light Sensor

## **5.3 Raspberry-pi**

### **5.3.1 Definition of Raspberry-pi**

Raspberry Pi is a low-cost, basic computer that was originally intended to help spur interest in computing among school-aged children. The Raspberry Pi is contained on a single circuit board and features ports for:

- HDMI
- USB 2.0
- Composite video
- Analog audio
- Power
- Internet
- SD Card

The computer runs entirely on open-source software and gives students the ability to mix and match software according to the work they wish to do.

## 5.4 the connection between iBeacons and Raspberry-Pi

Estimote Beacons broadcast tiny packets of data, containing their Mac Address and iBeacon ID , Information about signal strength ( RSSI and RSSI per Meter(A) ) is defined at the scanner which are used in calculating the position, so that the phone can understand which beacon it hears and how far it is.

Every iBeacon ID is 20 bytes long and is divided into three sections:

- UUID (16 bytes)
- major number (2 bytes)
- minor number (2 bytes)

Those values are hierarchical. Apple provides a great example of how you should think about them. Imagine a chain of retail stores (defined by UUID) that deploys beacons in three cities (defined by major): San Francisco, Paris, and London.

It would look like that:

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

Table 5.2

- Each kind of beacons has the same UUID for example: all Proximity Beacons has the same UUID (**B9407F30-F5F8-466E-AFF9-25556B57FE6D**) and all stickers has the same UUID, but we need to differentiate between different beacons so we can differentiate between them through the major and minor.
- You can change major, minor or UUID from estimate application but you must be the owner of the beacons or the owner can transfer the beacons to your account so that you can change any parameter.
- For facilitation we put a constant number for the major of each kind of the beacons for example:
  - Estimote Beacons Major = 150
  - Sticker Beacons Major = 151
  - Location Beacons Major = 153
- We put a different minor for each colour of any type of beacons.
- So if we want to detect certain type of beacons for example sticker , we can detect it through major and if we want to detect a certain colour of sticker we can detect it through major and minor together .

- The following table shows each Beacon and the corresponding major and minor concatenated together.

<b>Beacon_name</b>	<b>major_minor</b>
Mint	1501
Ice	1502
Blueberryy(proximity)	1503
Icy Marshmallow car(sticker)	1511
Pink bed(stickers)	1512
Mint Refrigerator(sticker)	1513
Lemon bike(sticker)	1514
Mint chair(sticker)	1515
Mint shoes(sticker)	1516
Pink(sticker)	1517
Blueberry door(sticker)	1518
sweet beetroot(location)	1531
candy(location)	1532
lemon(location)	1533

Table 5.3

- The main function of the scanner (Raspberry PI) is to receive this packets sent from beacons  
And translate this encrypted packet to packet parameters mentioned before (Mac Address, UUID, Major, Minor, RSSI and A).
- The scanner (Raspberry PI) acts like mobile phone.
- The scanner code receives the packet from Dongle Bluetooth USB which is connected to Raspberry PI and acts like receiving antenna.
- This dongle can detect any Bluetooth device not only Beacons so we had to filter the packets and take the Beacons packets only

we did that through the major of the Beacons as shown below :

```
# filtering the estimate packets only
if (Adstring[76:79]=="150" or Adstring[76:79]=="151" or Adstring[76:79]=="153"):
    myFullList.append(Adstring)
```

Figure 5.2

## Chapter 6: Indoor-Navigation Methods

### 6.1 Trilateration

Trilateration is used to find an unknown location from at least three known locations. Trilateration uses the distances from the known locations to estimate the coordinate of the unknown location. The distances between reference locations and the unknown location can be treated as the radii of many circles with centers at every reference location. Thus, unknown location is the intersection of all the spherical surfaces as shown in figure 6.1 .

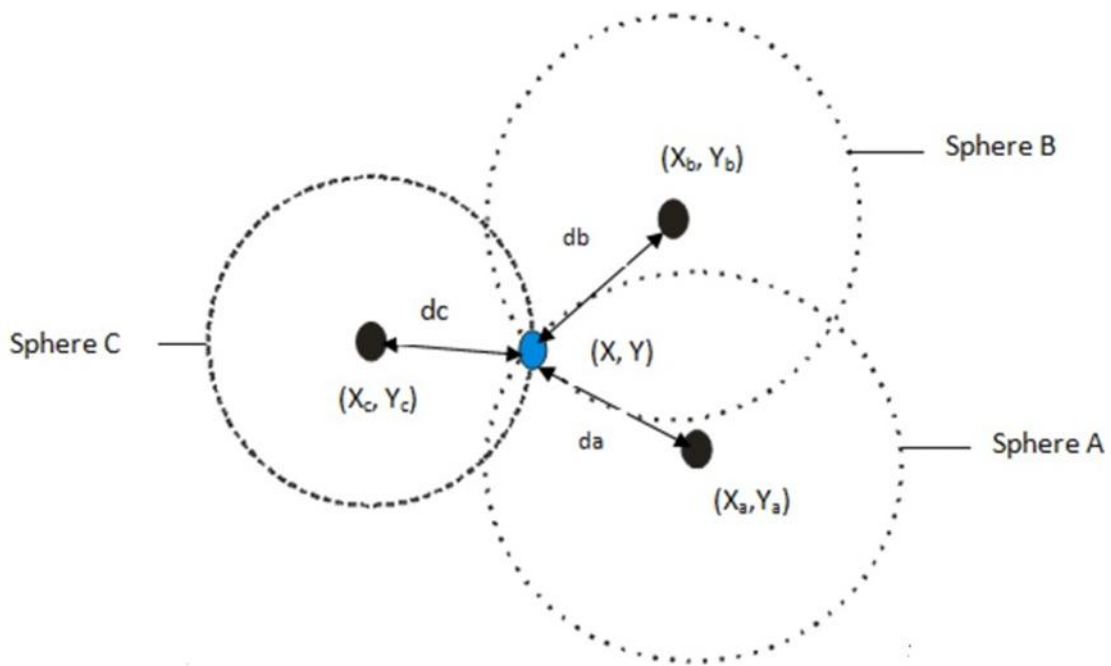


Figure 6.1 Trilateration Estimation which shows the concentric circles used by the RSSI ranging technique to calculate the position using R1, R2, R3. The mobile node, is at the center-point of all three circles of each anchor node.

There are three nodes that are randomly allocated with known location R1( $X_a, Y_a$ ), R2( $X_b, Y_b$ ), R3( $X_c, Y_c$ ). Node ( $x, y$ ) at unknown coordinates can be calculated using reference nodes R1, R2, and R3 and the distances  $d_a$ ,  $d_b$ , and  $d_c$  between the reference nodes and the unknown target node. A solution to this problem can be achieved using the Pythagorean Theorem.

$$d_a^2 = (x_a - x)^2 + (y_a - y)^2$$

$$d_b^2 = (x_b - x)^2 + (y_b - y)^2 \quad (6.1)$$

$$d_c^2 = (x_c - x)^2 + (y_c - y)^2$$

Rearranging the equations in (6.1) and solving for x and y.

$$X = \frac{AY_{cb} + BY_{ac} + CY_{ba}}{2(X_a Y_{cb} + X_b Y_{ac} + X_c Y_{ba})} \quad (6.2)$$

$$Y = \frac{Ax_{cb} + Bx_{ac} + Cx_{ba}}{2(Y_a x_{cb} + y_b x_{ac} + y_c x_{ba})} \quad (6.3)$$

Where

$$\begin{aligned} A &= x_a^2 + y_a^2 - d_a^2 \\ B &= x_b^2 + y_b^2 - d_b^2 \\ C &= x_c^2 + y_c^2 - d_c^2 \end{aligned} \quad (6.4)$$

And

$$\begin{aligned} x_{cb} &= (x_c - x_b) \\ x_{ac} &= (x_a - x_c) \\ x_{ba} &= (x_b - x_a) \\ y_{cb} &= (y_c - y_b) \\ y_{ac} &= (y_a - y_c) \\ y_{ba} &= (y_b - y_a) \end{aligned} \quad (6.5)$$

By using equations 6.2 and 6.3 in conjunction with 6.4 and 6.5, localization is easily achieved because the distances ( $d_1, d_2, d_3$ ) can be obtained from ranging. The locations of each of the reference nodes ( $x_1, y_1$ ), ( $x_2, y_2$ ), and ( $x_3, y_3$ ) can be stored at the reference node itself, or at a base station. In large scale networks,



three possible scenarios for localization might occur:

1. The node(s) is able to reach at least three reference nodes.
2. The node(s) is able to reach one or two reference nodes only.
3. The node(s) is unable to reach any reference node at all.

For any localization system, three reference nodes are required. Therefore, it is important to set up the system such that at any given point, three reference nodes can be reached.

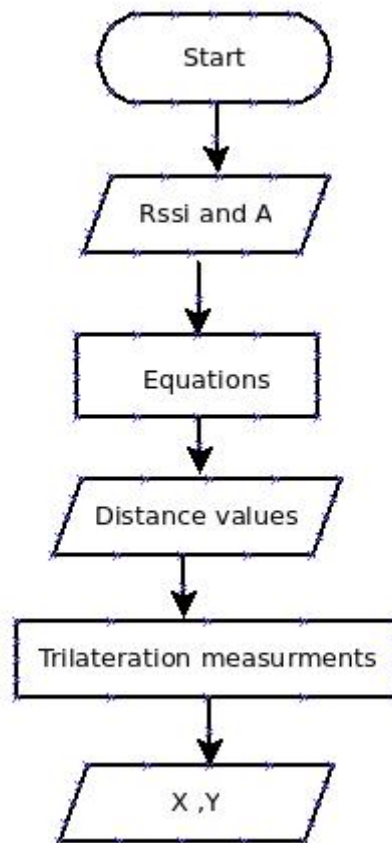


Figure 6.2

## **6.2 Particle Filter**

A Bayesian filter is used to estimate the state of any system that changes over time using a sequence of noisy measurements made on the system. The particle filter is a way to implementing the recursive Bayesian filter by Monte Carlo simulations. The key feature here is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights.

### **6.2.1 Particle Filter Nomenclature**

In probability theory, a random measure is a measure-valued random element. While a random variable is a variable whose value is real and subject to mathematical randomness, or chance, a random element is a more generalized representation of a random variable such that instead of associating the value with a real number, it can be associated with functions, vectors, processes, sets, collection of sets, transformations, etc.

In Bayesian estimation or statistics, the posterior probability is the conditional probability of a random event of an uncertain proposition that has been assigned after relevant observations have occurred and been used to calculate its probability. Related is the posterior probability distribution which is treated as a random variable conditional on (given two jointly distributed  $X$  and  $Y$  random variables, the probability of  $X$  given when  $Y$  is at some particular value) the observation obtained from the experiment or survey itself.

With statistics, the idea of probability density functions or densities is a function that describes the chance or likelihood of a random variable taking on a given value.

### **6.2.2 Particle Filter Sequential Importance Sampling**

The sequential importance sampling particle filter is used to represent the posterior density function as a random set of samples with weights and then compute estimates based on these samples and weights. With an increased number of samples,

the Monte Carlo simulation becomes a close approximation to the usual functional description of the posterior.

such, the particle filter approaches the optimal estimate of the state. Much of this section's math is thanks to the work of .

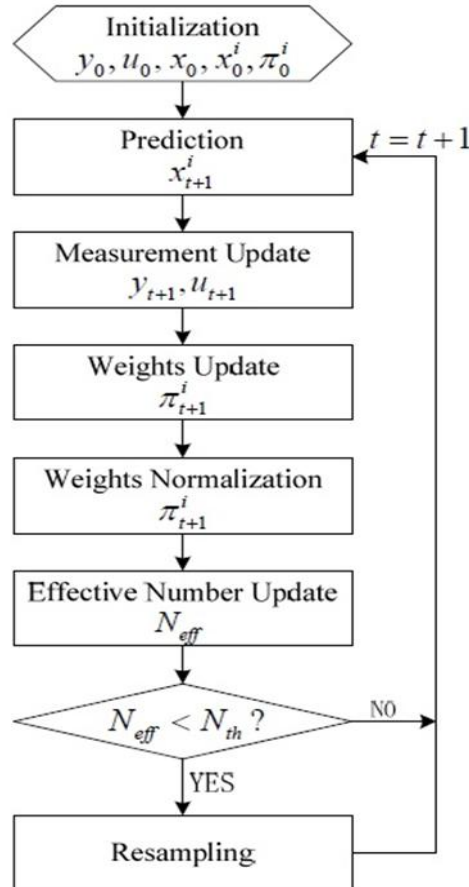


Figure 6.3

“Let us specify that the posterior density can be approximated as:

$$p(x_{0:k} | z_{1:k}) \approx \sum_{i=1}^{N_s} (w_k^i \delta(X_{0:k} - x_{0:k}^i)) \quad (6.6)$$

Since the posterior density can be approximated by  $p(x_{0:k} | z_{1:k})$ , we have a discrete weighted approximation to the true posterior. The weights are chosen using the principle of importance sampling [35], [36]. The principle relies on the following.

Suppose  $p(x) \propto \pi(x)$  is a probability density for which it is hard to draw samples from, but it is possible for  $\pi(x)$  to be evaluated. This also allows for  $p(x)$  to a certain extent as well due to proportionality. Also, let  $x_i \sim q(\cdot)$ ,  $i=1, \dots, N_s$ , be samples that are generated from a proposal  $q(\cdot)$  which is called an importance density. A weighted approximation to the density  $p(\cdot)$  is given by:

$$p(x) \approx \sum_{i=1}^{N_s} (w^i \delta(x_{0:k} - x_{0:k}^i)) \quad (6.7)$$

Where

$$w^i \propto \frac{\pi(x^i)}{q(x^i)} \quad (6.8)$$

Is the normalized weight of the particle and  $N_s$  is the number of samples.

Therefore, if the samples  $(x_{0:k}^i)$  where drawn from an importance density  $(x_{0:k} | z_{1:k})$ , then the weights in (6.6) are defined by (6.8) to be :

$$w_k^i \propto \frac{p(x_{0:k}^i | z_{1:k})}{q(x_{0:k}^i | z_{1:k})} \quad (6.9)$$

Returning to the sequential case, at each iteration, one could have samples constituting an approximation to  $P(x_{0:k-1} | z_{1:k-1})$  and want to approximate  $P(x_{0:k} | z_{1:k})$  with a new set of samples. If the importance density is chosen to factorize such that

$$q(x_{0:k} | z_{1:k}) = q(x_k | x_{0:k-1}, z_{1:k}) q(x_{0:k-1} | z_{1:k-1}) \quad (6.10)$$

Then one can obtain samples  $x_{0:k}^i \sim (x_{0:k} | z_{1:k})$  by augmenting each of the existing samples  $x_{0:k-1}^i \sim (x_{0:k-1} | z_{1:k-1})$  with the new state  $x_k^i \sim (x_k | x_{0:k-1}, z_{1:k})$ .

To derive the weight update equation,  $(x_{0:k} | z_{1:k})$  is first expressed in terms of  $(x_{0:k-1}, z_{1:k-1})$ ,  $(z_k | x_k)$ , and  $(x_k | x_{k-1})$ .

$$\begin{aligned}
(x_{0:k} | Z_{1:k}) &= \frac{p(z_k | x_{0:k}, z_{1:k-1}) p(x_{0:k} | z_{1:k-1})}{p(z_k | z_{1:k-1})} \\
&= \frac{p(z_k | x_{0:k}, z_{1:k-1}) p(x_k | x_{0:k-1}, z_{1:k-1})}{p(z_k | z_{1:k-1})} p(x_{0:k-1} | z_{1:k-1}) \\
&= \frac{p(z_k | x_{0:k}, z_{1:k-1}) p(x_k | x_{0:k-1}, z_{1:k-1})}{p(z_k | z_{1:k-1})} p(x_{0:k-1} | Z_{1:k-1}) \quad (6.11)
\end{aligned}$$

$$\begin{aligned}
&= \frac{p(z_k | x_k) p(x_k | x_{k-1})}{p(z_k | z_{1:k-1})} p(x_{0:k-1} | Z_{1:k-1}) \\
&\propto p(z_k | x_k) p(x_k | x_{k-1}) p(x_{0:k-1} | z_{1:k-1}) \quad (6.12)
\end{aligned}$$

By substituting (6.10) and (6.11) into (6.9), the weight update equation can be shown to be

$$\begin{aligned}
W_k^i &\propto \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i) p(x_{0:k}^i | z_{1:k-1}^i)}{q(x_k^i | x_{0:k-1}^i, z_{1:k}) q(x_{0:k-1}^i | z_{1:k-1})} \\
&= W_{k-1}^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{q(x_k^i | x_{0:k-1}^i, z_{1:k})} \quad (6.13)
\end{aligned}$$

Furthermore, if  $(x_k | x_{0:k-1}, z_{1:k}) = (x_k | x_{k-1}, z_k)$ , then the importance density becomes only dependent on  $x_{k-1}$  and  $z_k$ . This is useful in the common case when only a filtered estimate of  $(x_k | z_{1:k})$  is required at each time step. In such scenarios only

$x_k^i$  need to be stored; therefore, one can discard the path  $x_{0:k-1}^i$  and history of observations  $z_{1:k-1}$ . The modified weight is then

$$W_k \propto W_{k-1}^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{q(x_k^i | x_{0:k-1}^i, z_{1:k})} \quad (6.14)$$

and the posterior filtered density  $p(x_k | z_{1:k})$  can be approximated as

$$p(x_k | z_{1:k}) \approx \sum_{i=1}^{N_s} (w_k^i \delta(x_k - x_k^i)) \quad (6.15)$$

where the weights are defined in (6.14). It can be shown that  $N_s \rightarrow \infty$ , the approximation (6.15) approaches the true posterior density  $p(x_k | z_{1:k})$ .

The algorithm used thus consists of recursive propagation of the weights and support points as each measurement is received sequentially.” as we see in **figure 6.1**

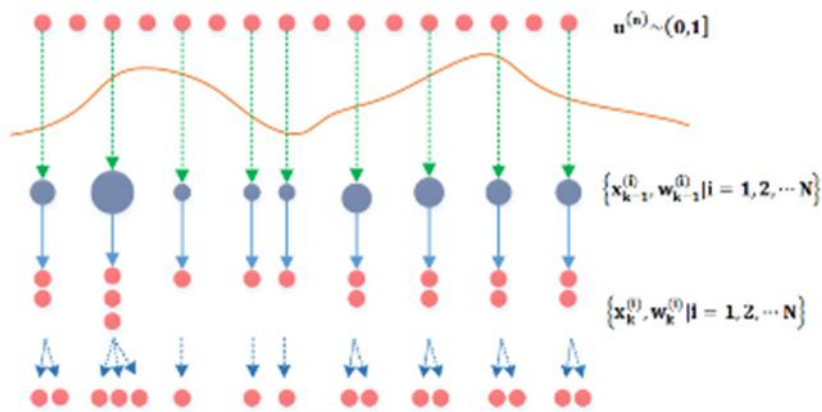


Figure 6.4

## 6.3 A\* Algorithm

In order to reach to the destination, knowing only position is not enough because there are lots of rooms inside building, for example, hotels and malls. Thus knowing the route to the destination is also very important so that user can reach the destination in lesser time easily.

A\* search algorithm is an extension of the Dijkstra's algorithm. A\* algorithm is Dijkstra's algorithm without heuristic. Both the algorithm gives the shortest path but the A\* gives more faster and efficient result. Dijkstra's algorithm is slower as compared to A\* because Dijkstra's repeatedly attempts to improve an initial approximation (cost) of each node. Due to this it takes more time to reach the target node.

### 6.3.1 Searching area

A\* Path finding algorithm The Search Area Let's assume that we have someone who wants to get from point A to point B. Let's assume that a wall separates the two points. This is illustrated below, with green being the starting point A, and red being the ending point B, and the blue filled squares being the wall in between.

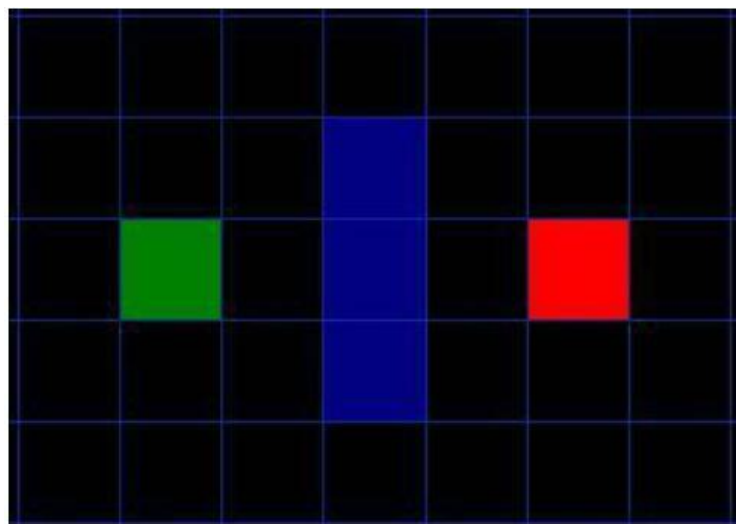


Figure 6.5

The first thing you should notice is that we have divided our search area into a square grid. Simplifying the search area, as we have done here, is the first step in

path finding. This particular method reduces our search area to a simple two dimensional array. Each item in the array represents one of the squares on the grid, and its status is recorded as walk able or unwalkable. The path is found by figuring out which squares we should take to get from A to B. Once the path is found, our person moves from the center of one square to the center of the next until the target is reached.

These center points are called “nodes”. When you read about path finding elsewhere, you will often see people discussing nodes. Why not just call them squares? Because it is possible to divide up your path finding area into something other than squares. They could be rectangles, hexagons, triangles, or any shape, really. And the nodes could be placed anywhere within the shapes – in the center or along the edges, or anywhere else. We are using this system, however, because it is the simplest.

### **6.3.2 Starting the Search**

Once we have simplified our search area into a manageable number of nodes, as we have done with the grid layout above, the next step is to conduct a search to find the shortest path. We do this by starting at point A, checking the adjacent squares, and generally searching outward until we find our target.

We begin the search by doing the following:

1. Begin at the starting point A and add it to an “open list” of squares to be considered. The open list is kind of like a shopping list. Right now there is just one item on the list, but we will have more later. It contains squares that might fall along the path you want to take, but maybe not. Basically, this is a list of squares that need to be checked out.
2. Look at all the reachable or walkable squares adjacent to the starting point, ignoring squares with walls, water, or other illegal terrain. Add them to the open list, too. For each of these squares, save point A as it’s “parent square”. This parent square stuff is important when we want to trace our path. It will be explained more later.



- Drop the starting square A from your open list, and add it to a “closed list” of squares that you don’t need to look at again for now.

At this point, you should have something like the following illustration. In this illustration, the dark green square in the center is your starting square. It is outlined in light blue to indicate that the square has been added to the closed list. All of the adjacent squares are now on the open list of squares to be checked, and they are outlined in light green. Each has a gray pointer that points back to its parent, which is the starting square.

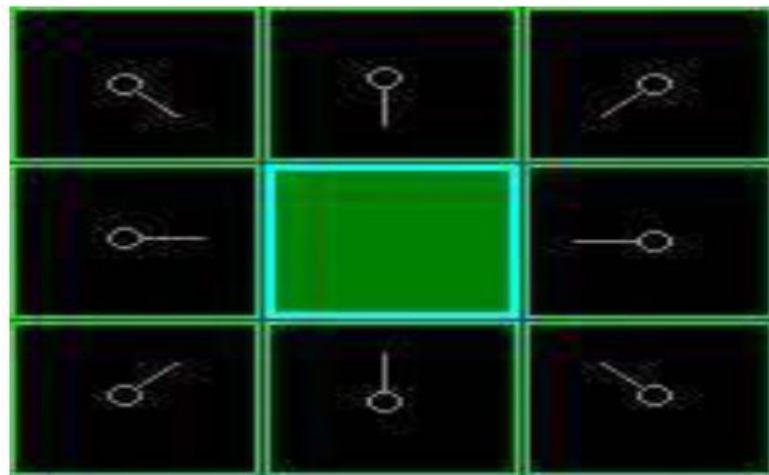


Figure 6.6

Next, we choose one of the adjacent squares on the open list and more or less *repeat* the earlier process, as described below. But which square do we choose? The one with the lowest F cost.

### 6.3.3 Path Scoring

The key to determining which squares to use when figuring out the path is the following equation:

$$F = G + H$$

Where:

- $G$  = the movement cost to move from the starting point  $A$  to a given square on the grid, following the path generated to get there.
- $H$  = the estimated movement cost to move from that given square on the grid to the final destination, point  $B$ . This is often referred to as the heuristic, which can be a bit confusing. The reason why it is called that is because it is a guess. We really don't know the actual distance until we find the path, because all sorts of things can be in the way (walls, water, etc.). You are given one way to calculate  $H$  in this tutorial, but there are many others that you can find in other articles on the web.

Our path is generated by repeatedly going through our open list and choosing the square with the lowest  $F$  score. This process will be described in more detail a bit further in the article. First let's look more closely at how we calculate the equation. As described above,  $G$  is the movement cost to move from the starting point to the given square using the path generated to get there. In this example, we will assign a cost of 10 to each horizontal or vertical square moved, and a cost of 14 for a diagonal move. We use these numbers because the actual distance to move diagonally is the square root of 2 (don't be scared), or roughly 1.414 times the cost of moving horizontally or vertically. We use 10 and 14 for simplicity's sake. The ratio is about right, and we avoid having to calculate square roots and we avoid decimals. This isn't just because we are dumb and don't like math. Using whole numbers like these is a lot faster for the computer, too. As you will soon find out, pathfinding can be very slow if you don't use short cuts like these.

Since we are calculating the  $G$  cost along a specific path to a given square, the way to figure out the  $G$  cost of that square is to take the  $G$  cost of its parent, and then add 10 or 14 depending on whether it is diagonal or orthogonal (non-diagonal) from that parent square. The need for this method will become apparent a little further on in this example, as we get more than one square away from the starting square.

$H$  can be estimated in a variety of ways. The method we use here is called the Manhattan method, where you calculate the total number of squares moved horizontally and vertically to reach the target square from the current square, ignoring

diagonal movement, and ignoring any obstacles that may be in the way. We then multiply the total by 10, our cost for moving one square horizontally or vertically. This is (probably) called the Manhattan method because it is like calculating the number of city blocks from one place to another, where you can't cut across the block diagonally.

Reading this description, you might guess that the heuristic is merely a rough estimate of the remaining distance between the current square and the target "as the crow flies." This isn't the case. We are actually trying to estimate the remaining distance along the path (which is usually farther). The closer our estimate is to the actual remaining distance, the faster the algorithm will be. If we overestimate this distance, however, it is not guaranteed to give us the shortest path. In such cases, we have what is called an "inadmissible heuristic."

Technically, in this example, the Manhattan method is inadmissible because it slightly overestimates the remaining distance. But we will use it anyway because it is a lot easier to understand for our purposes, and because it is only a slight overestimation. On the rare occasion when the resulting path is not the shortest possible, it will be nearly as short.

F is calculated by adding G and H. The results of the first step in our search can be seen in the illustration below. The F, G, and H scores are written in each square. As is indicated in the square to the immediate right of the starting square, F is printed in the top left, G is printed in the bottom left, and H is printed in the bottom right.

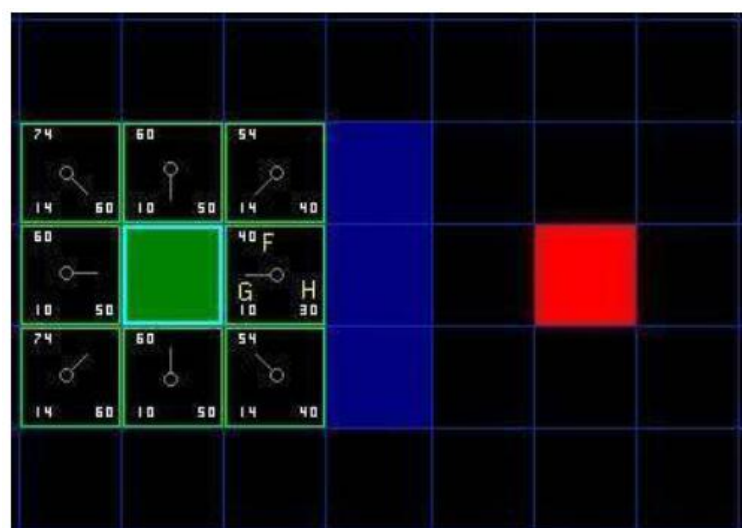


Figure 6.7

So let's look at some of these squares. In the square with the letters in it,  $G = 10$ . This is because it is just one square from the starting square in a horizontal direction.

The squares immediately above, below, and to the left of the starting square all have the same  $G$  score of 10. The diagonal squares have  $G$  scores of 14.

The  $H$  scores are calculated by estimating the Manhattan distance to the red target square, moving only horizontally and vertically and ignoring the wall that is in the way. Using this method, the square to the immediate right of the start is 3 squares from the red square, for a  $H$  score of 30. The square just above this square is 4 squares away (remember, only move horizontally and vertically) for an  $H$  score of 40. You can probably see how the  $H$  scores are calculated for the other squares.

The  $F$  score for each square, again, is simply calculated by adding  $G$  and  $H$  together.

#### **6.3.4 Continuing the Search**

To continue the search, we simply choose the lowest  $F$  score square from all those that are on the open list. We then do the following with the selected square:

4. Drop it from the open list and add it to the closed list.
5. Check all of the adjacent squares. Ignoring those that are on the closed list or unwalkable (terrain with walls, water, or other illegal terrain), add squares to the open list if they are not on the open list already. Make the selected square the "parent" of the new squares.
6. If an adjacent square is already on the open list, check to see if this path to that square is a better one. In other words, check to see if the  $G$  score for that square is lower if we use the current square to get there. If not, don't do anything.

On the other hand, if the  $G$  cost of the new path is lower, change the parent of the adjacent square to the selected square (in the diagram above, change the direction of the pointer to point at the selected square). Finally, recalculate both the  $F$  and  $G$  scores of that square. If this seems confusing, you will see it illustrated below.

Okay, so let's see how this works. Of our initial 9 squares, we have 8 left on the open list after the starting square was switched to the closed list. Of these, the one with the lowest F cost is the one to the immediate right of the starting square, with an F score of 40. So we select this square as our next square. It is highlighted in blue in the following illustration.



Figure 6.8

First, we drop it from our open list and add it to our closed list (that's why it's now highlighted in blue). Then we check the adjacent squares. Well, the ones to the immediate right of this square are wall squares, so we ignore those. The one to the immediate left is the starting square. That's on the closed list, so we ignore that, too. The other four squares are already on the open list, so we need to check if the paths to those squares are any better using this square to get there, using G scores as our point of reference. Let's look at the square right above our selected square. Its current G score is 14. If we instead went through the current square to get there, the G score would be equal to 20 (10, which is the G score to get to the current square, plus 10 more to go vertically to the one just above it). A G score of 20 is higher than 14, so this is not a better path. That should make sense if you look at the diagram. It's more direct to get to that square from the starting square by simply moving one square diagonally to get there, rather than moving horizontally one square, and then vertically one square.

When we repeat this process for all 4 of the adjacent squares already on the open list, we find that none of the paths are improved by going through the current square,

so we don't change anything. So now that we looked at all of the adjacent squares, we are done with this square, and ready to move to the next square.

So we go through the list of squares on our open list, which is now down to 7 squares, and we pick the one with the lowest F cost. Interestingly, in this case, there are two squares with a score of 54. So which do we choose? It doesn't really matter. For the purposes of speed, it can be faster to choose the last one you added to the open list. This biases the search in favor of squares that get found later on in the search, when you have gotten closer to the target. But it doesn't really matter. (Differing treatment of ties is why two versions of A\* may find different paths of equal length.) So let's choose the one just below, and to the right of the starting square, as is shown in the following illustration.

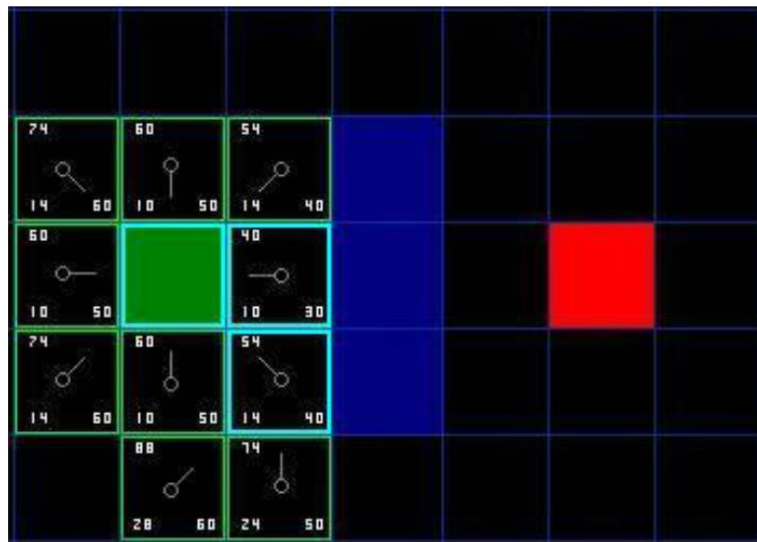


Figure 6.9

This time, when we check the adjacent squares we find that the one to the immediate right is a wall square, so we ignore that. The same goes for the one just above that. We also ignore the square just below the wall. Why? Because you can't get to that square directly from the current square without cutting across the corner of the nearby wall. You really need to go down first and then move over to that square, moving around the corner in the process. (Note: This rule on cutting corners is optional. Its use depends on how your nodes are placed.)

That leaves five other squares. The other two squares below the current square aren't already on the open list, so we add them and the current square becomes their parent. Of the other three squares, two are already on the closed list (the starting square, and the one just above the current square, both highlighted in blue in the diagram), so we ignore them. And the last square, to the immediate left of the current square, is checked to see if the G score is any lower if you go through the current square to get there. No dice. So we're done and ready to check the next square on our open list.

We repeat this process until we add the target square to the closed list, at which point it looks something like the illustration below.

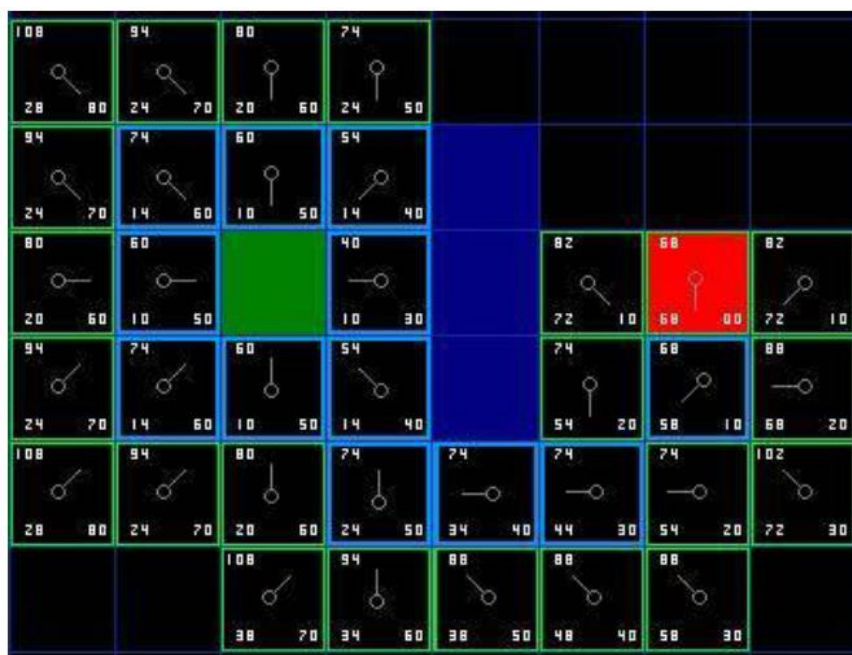


Figure 6.10

Note that the parent square for the square two squares below the starting square has changed from the previous illustration. Before it had a G score of 28 and pointed back to the square above it and to the right. Now it has a score of 20 and points to the square just above it. This happened somewhere along the way on our search, where the G score was checked and it turned out to be lower using a new path – so the parent was switched and the G and F scores were recalculated. While this change doesn't seem too important in this example, there are plenty of possible situations where this

constant checking will make all the difference in determining the best path to your target.

So how do we determine the path? Simple, just start at the red target square, and work backwards moving from one square to its parent, following the arrows. This will eventually take you back to the starting square, and that's your path. It should look like the following illustration. Moving from the starting square A to the destination square B is simply a matter of moving from the center of each square (the node) to the center of the next square on the path, until you reach the target.

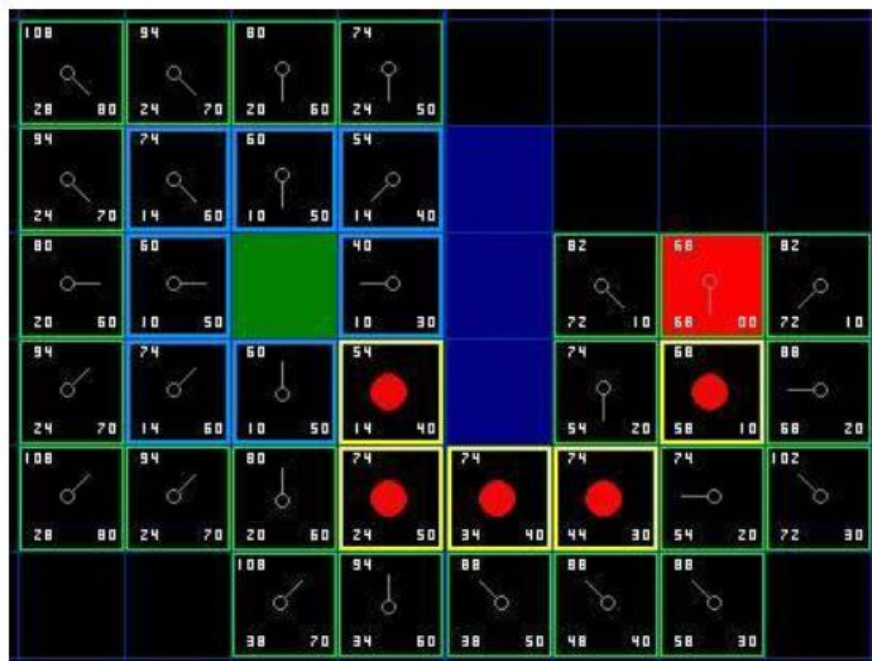


Figure 6.11

### 6.3.5 Summary of the A\* Method

now that you have gone through the explanation, let's lay out the step-by-step method all in one place:

- 1) Add the starting square (or node) to the open list.
- 2) Repeat the following:
  - a) Look for the lowest F cost square on the open list. We refer to this as the current square.
  - b) Switch it to the closed list.



- c) For each of the 8 squares adjacent to this current square ...
- If it is not walk able or if it is on the closed list, ignore it. Otherwise do the following.
  - If it isn't on the open list, add it to the open list. Make the current square the parent of this square. Record the F, G, and H costs of the square.
  - If it is on the open list already, check to see if this path to that square is better, using G cost as the measure. A lower G cost means that this is a better path. If so, change the parent of the square to the current square, and recalculate the G and F scores of the square. If you are keeping your open list sorted by F score, you may need to resort the list to account for the change.
- d) Stop when you:
- Add the target square to the closed list, in which case the path has been found (see note below), or
  - Fail to find the target square, and the open list is empty. In this case, there is no path.
- 3) Save the path. Working backwards from the target square, go from each square to its parent square until you reach the starting square. That is your path.

### **6.3.6 Map Constructions**

As known that any image consist of number of pixels, but A star deals with the image or the map as number of grids so we had to construct the map as number of grids so that we can navigate using A star on a real map.

So, first we divide the map into grids by considering every 41x41 pixel as a grid as showing below,

**this number is not constant it depends on map and the accuracy you want.**

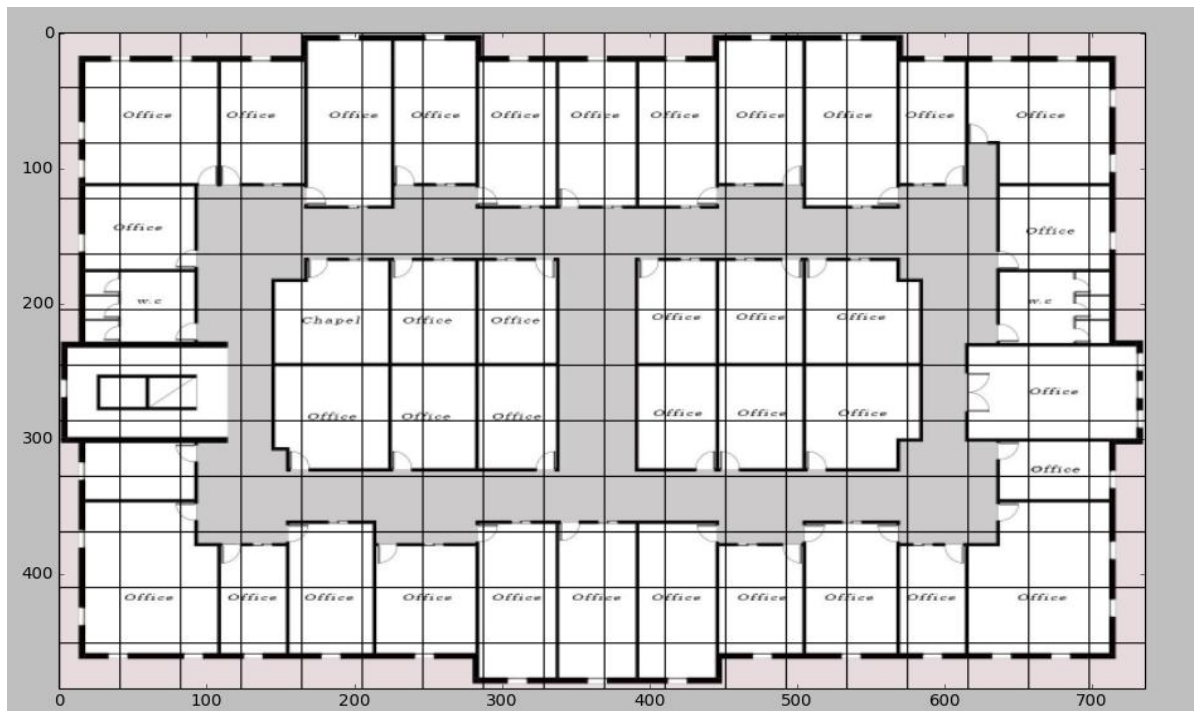


Figure 6.12

After that we collect the index of every obstacle grid in an array to be an input to A star, we did the same for space grids but in this case we made an array from the corresponding value of every space grid in pixel so that we can draw resulting path on this map.

## Chapter 7: Web Server Development

### 7.1 Back End

The back-end or the server side, is basically how the site works, updates and changes. This refers to everything the user can't see in the browser, like database and servers. Usually people who work on the back-end are called programmers or developers. The warring things at the back-end are security, structure and content management. They usually know and can use languages like HTML and CSS. but not our focus. Back end development is required to create a dynamic site. A dynamic site is a site that's constantly changing and updated in real time. Most sites are dynamic sites, as opposed to static sites. Facebook, Google Maps and their blog are all considered dynamic sites. Blogs are dynamic sites since their content is constantly changing and updating. A dynamic site requires a database work properly. All information like user profiles or images they've uploaded or blog posts are stored in the database. php is the main language used in web developing. It is the server side programming language. since the database understands this language, Php helps us to send and get request from and to database like MySQL. The code written in php communicates with the server and then tells the browser what to use from the database. PHP can communicate with client side languages like javascript and HTML. Using Ajax request can be sent to server using php and get results into html page with no need to refresh the page.

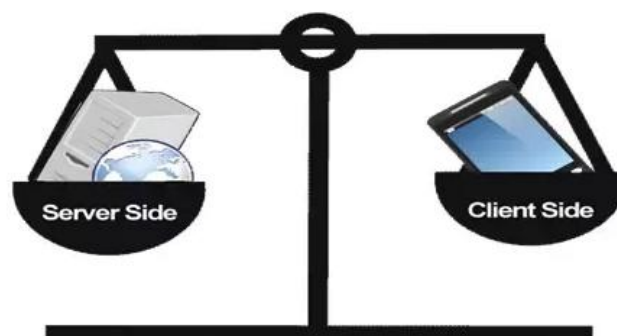


Figure 7.1

### 7.1.1 LAMP

The LAMP platform consists of four components that are structured in a layered way. Each layer provides a critical part of the entire software stack:

Linux: is the lowest-level layer and provides the operating system. Linux actually runs each of the other components.

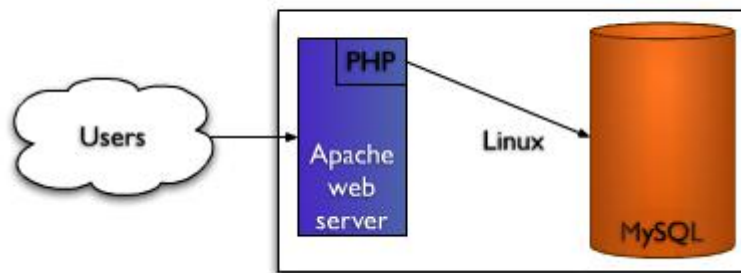


Figure 7.2

Apache: The next layer is Apache, the web server. Apache provides the mechanics for getting a web page to a user. Apache is a stable, mission-critical-capable server, and it runs more than 65 percent of all web sites on the internet. The PHP component actually sits inside Apache, the Apache and PHP are used together to create dynamic pages.

MySQL: MySQL provides the data-storage side of the LAMP system. With MySQL, you have access to a very capable database suitable for running large and complex sites. Within web applications, all data, products, accounts, and other types of information will reside in this database in a format that you can easily query with the SQL languages.

PHP: is a simple and efficient programming language that provides the glue for all the other parts of the LAMP system. PHP is used to write dynamic content capable of accessing the data in the MySQL database and some of the features that Linux provides.

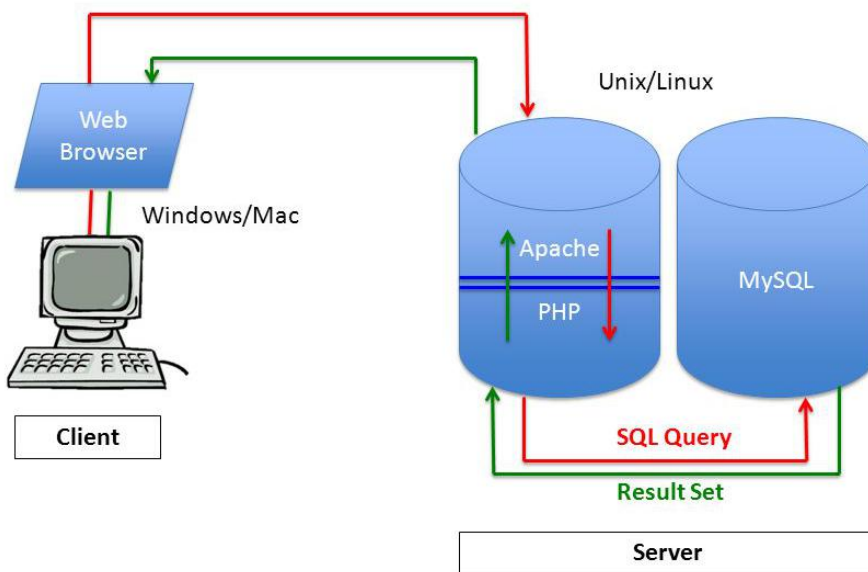


Figure 7.3

Some modules need to be downloaded in order to get started with LAMP. Apache PHP module connects Apache with PHP engine. PHP MySQL module connects PHP engine with database.

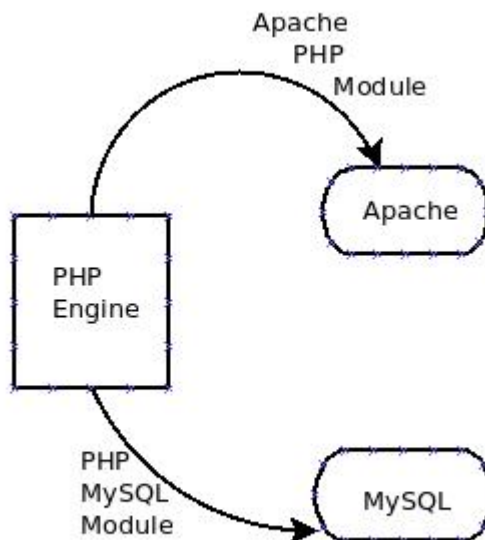


Figure 7.4

### 7.1.1.1 CRUD operation

The CRUD cycle describes the elemental functions of a persistent database. CRUD stands for Create, Read, Update and Delete. Retrieve may be occasionally be substituted for Read.

Crud operation is created using PHP, MySQL and CSS. Different administrators may have different CRUD cycles based upon the requirements of the location. In the location required to build indoor navigation system in it the administrator through Crud operation can enter the database of the building like name of building, number of floors, name of shops and shop coordinates. The data he entered will be stored in data base. These data is needed for indoor navigation system in order to know the place of beacons, the shop coordinates and all data of the location so that the algorithms can connect on database and read from it the data needed to detect the place of the user when he passed by beacons and can draw on map the path for his required destination.

The figure below shows the admin's interface where he enters the input data that will be stored in database.



The screenshot displays a web interface titled "BLE indoornavigation". At the top, there is a blue header bar with the title. Below the header, there is a light gray container with a "back to main page" link in blue text. The main content area contains a form with five input fields, each with a light blue border and a dashed outline. The fields are labeled "name", "address", "description", "coordinates", and "image\_path". At the bottom of the form, there is a blue button with the text "SAVE" in white capital letters.

Figure 7.5

The screenshot shows the BLENS application interface. At the top, there is a blue header with the text "BLENS". Below the header, there is a light gray box containing a form. At the top of the form, there is a link that says "back to main page". The form has three input fields: "Beacon\_Name", "Major\_Minor", and "Coordinates". At the bottom of the form, there is a blue button labeled "SAVE".

Figure 7.6

The figures below shows the data the administrator stored in data base. From this interface he can update ,delete or add the row he want to change. He can also add new data.

The screenshot shows the BLE Indoor Navigation application interface. At the top, there is a blue header with the text "BLE Indoor Navigation". Below the header, there is a light gray box containing a table. At the top of the table, there is a link that says "add data here.". The table has six columns: "name", "address", "description", "coordinats", "image\_path", and "Operations". The "Operations" column contains two sub-columns, each with a pencil icon and a red 'X' icon. The table contains two rows of data.

add data here.					
name	address	description	coordinats	image_path	Operations
Cairo Festival City	5th settlement	Shopping Mall	3,4	c/Newfolder	
Mall of Arabia	6 October	Shopping Mall	1,2	D/pictures	

Figure 7.7

# BLENS

add data here.				
Beacon_Name	Major_Minor	Coordinates	Operations	
Mint	1501	7,7		
Ice	1502	11,8		
Blueberyy(proximity)	1503	12,11		
Icy Marshmallow car(sticker)	1511	16,11		
Pink bed(stickers)	1512	15,12		
Mint Refrigerator(sticker)	1513	13,20		
Lemon bike(sticker)	1514	12,5		

Figure 7.8

## 7.1.1.2 Code sequence

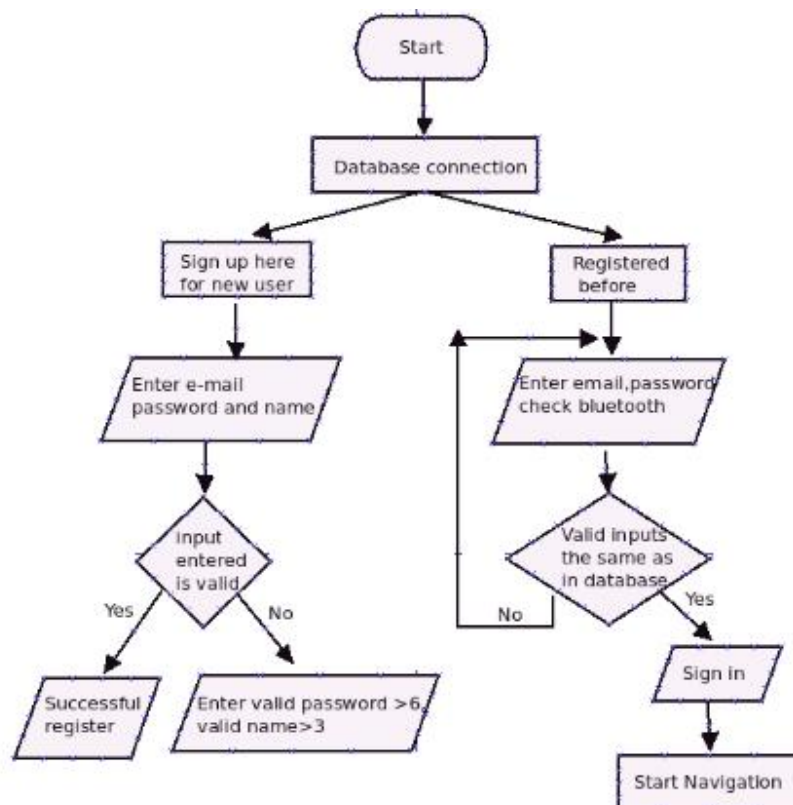


Figure 7.9



Flow chart above clarify the sequence of codes that used in **registration** process. First the connection between code and data base is established. If the user is newer to the application he will sign up enter his data then the code takes the input the user entered in the interface and store it in database. If he used the application before he will enter email and password only then sign in, The code will compares the input with the data stored in database then if they are the same then a new page will be reloaded.

The flow charts below clarify the sequence of the code. The code stores the destination the user choose from the interface and store it in database the A\* takes the destination from database then the algorithm executes and outputs the points of path then code reads these points and output them on the map on web page. also takes the values(x,y) of output of trilateration that are stored in database and make them the coordinates of point that detects the place of user on map.

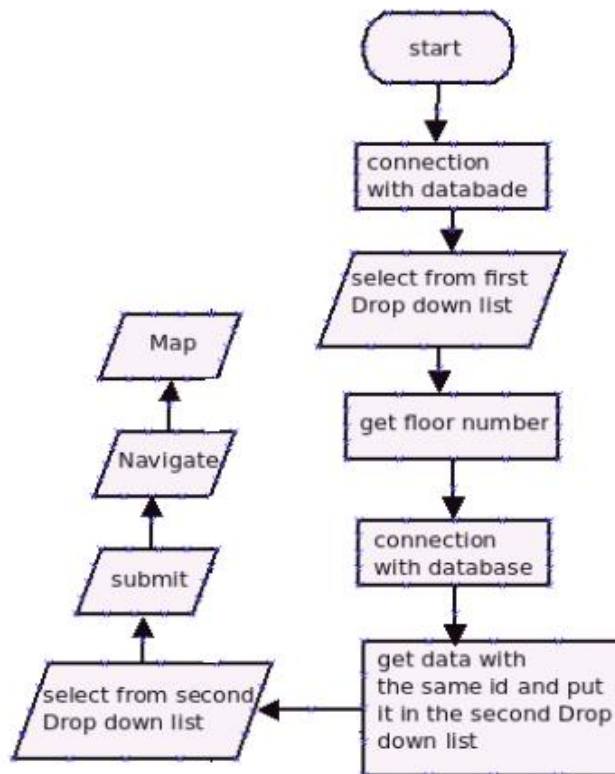


Figure 7.10

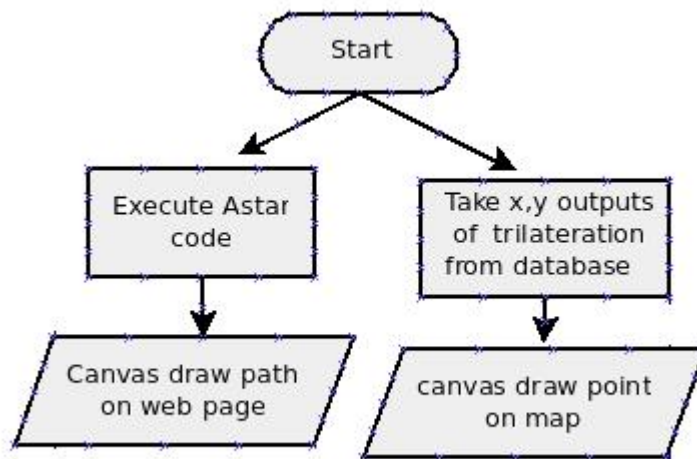


Figure 7.11

## 7.2 Front End

The front end is every thing involved with what the user sees,including design and some languages as HTML and CSS.HTML is the standard markup language for creating web pages .CSS is a core functionality of front end development,the styles that lay out the page and give it both its unique visual flair and a clear user friendly view.

### 7.2.1 User Interface

The interface of the web page which user will open to use BLE indoor navigation system to detect his place on map and choose the place he want to go, so the system will show the path he will go through to reach his destination easily.AT the start,the web server looks as shown in the picture below.

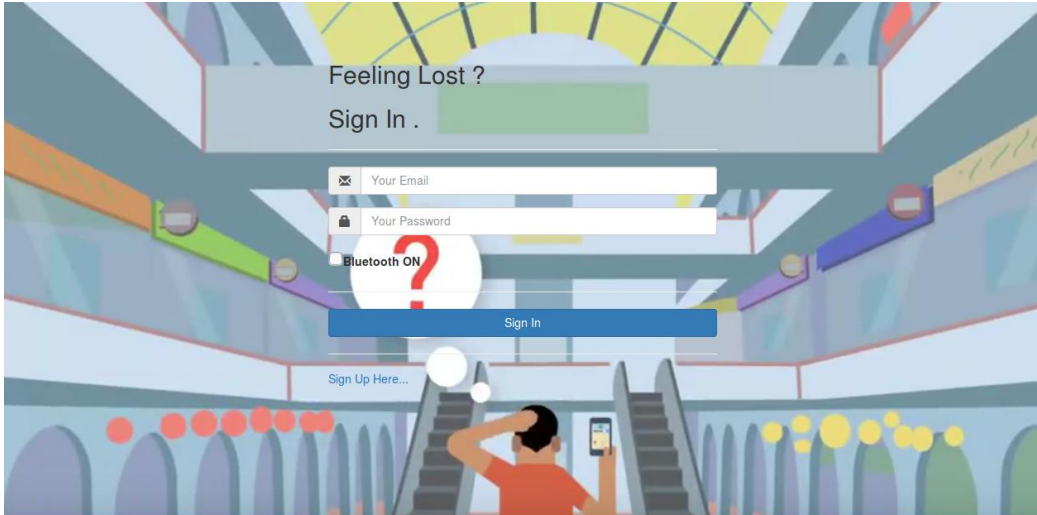


Figure 7.12

A starting page is displayed asking user to sign up entering his name, new registration email and password. Then he will sign into the page. Also asking permission from the user to turn on the Bluetooth adapter. If he enters wrong name or password a message will be displayed to check his inputs. When he signed in a new page will be displayed as shown below.

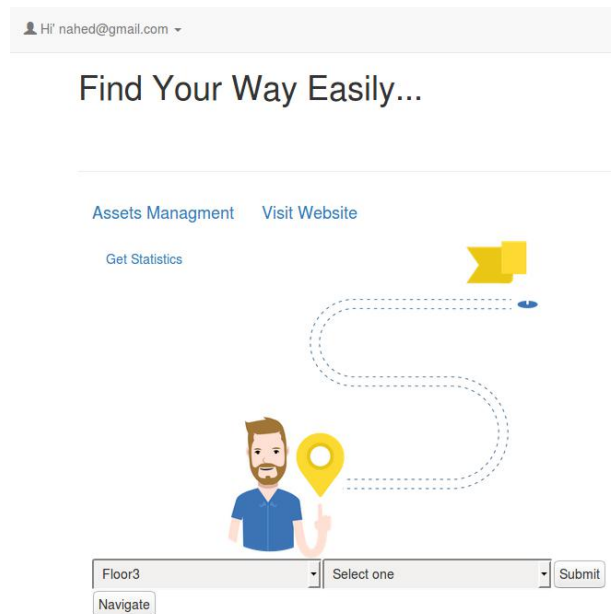


Figure 7.13

At the above figure the user choose the floor he is present at and Then the destination which he wants from the shown two drop down lists,Then submit ,Then he will navigate.

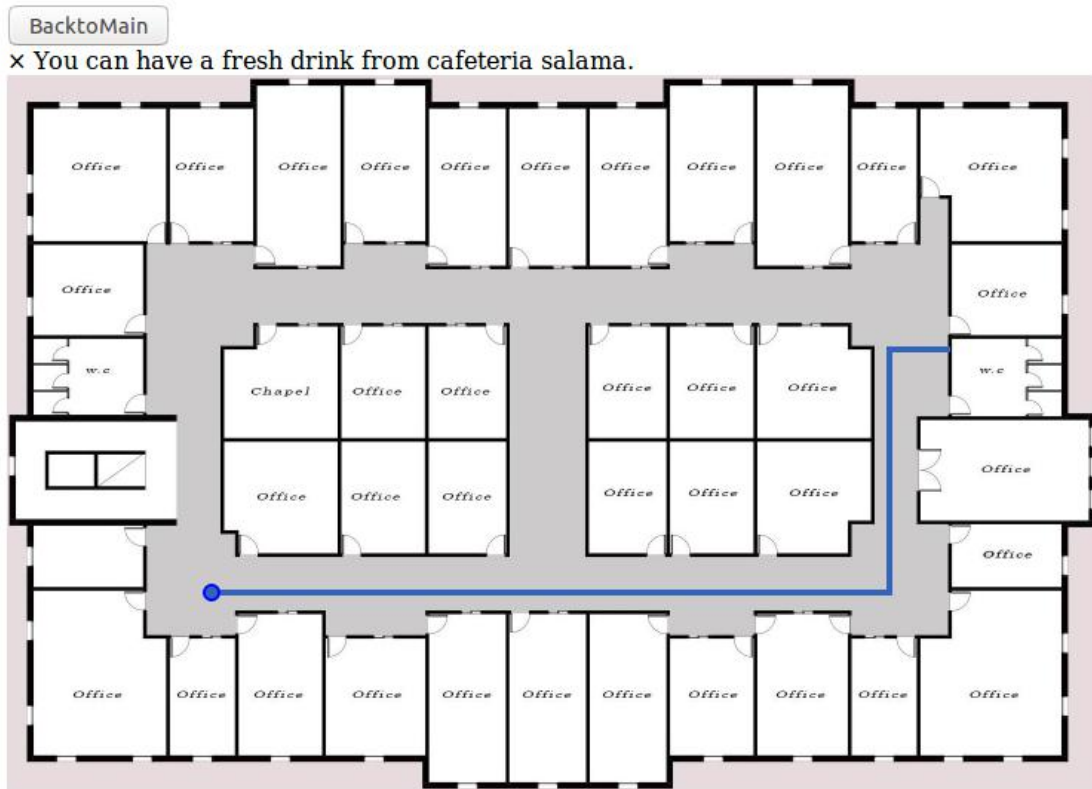


Figure 7.14

A new page containing map of floor that user chose will be displayed showed on the map the point which detects the user place and a path is drawn to the destination he wants to go to. When he follows this path he will reach his destination easily.

We can also send on map advertising messages for nearby customers.

### 7.2.2 Asset management:

Asset management can be used to monitor valuable products at stores, kids at kindergarten, etc. It works as follows: beacons move and scanner remains at its place, for example a scanner will be placed in each room and it will scan the beacons in each room the names of beacons present in each room will appear at web page.



Figure 7.15

### 7.2.3 Get statistics:

It also will help producers to know the best products that customers need. In malls for example they will know the most visited places.

# Visits	Office
1	Arheticture
1	Civil Control
1	Hanafi Hassan
2	Hanan Ahmed
1	Hany Amin
8	Hassan Mostafa




Figure 7.16

## **Chapter 8:                    Mobile Application**

In this chapter we will have a brief discussion about the mobile interface and how it works.

A mobile app is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers.

There are essentially two ways to deliver an application on Android: as a client-side application (developed using the Android SDK and installed on user devices in an APK) or as a web application (developed using web standards and accessed through a web browser).

Our app targetting android mobile phones so,we use android studio program to implement our app. code written in javascript then send the application to the desired mobile through it's corresponding driver.

First, we work on create client-side application which use the wifi to access the database and codes uploaded on the server to get or store the required data for navigation on the application which done as follows.

Mobile Interface:

Java files:

- 1.Login activity
- 2.Sign\_up activity
- 3.Navigation activity
- 4.Map activity

First login if the user already have an account in the database on server also, the user must be sure that the Bluetooth button is checked then click on "sign in or register" button to sign in or click on "sign up here" text view to create a new account first then sign in.

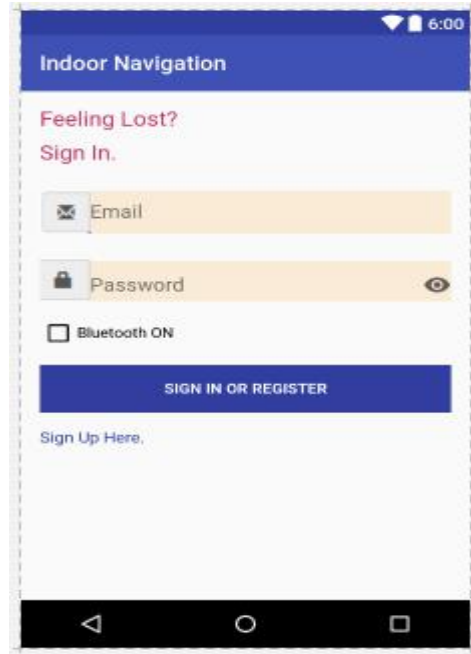


Figure 8.1

## 8.1 Sign up page

Here the user create a new account then click on "sign up" button to connect with server and store the new user as a new row in user table in the database then click on "sign in here" text view to return to the "sign in" page.

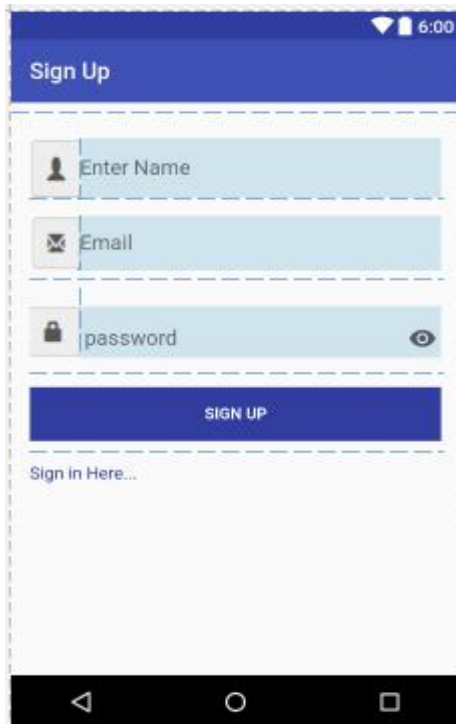


Figure 8.2

## 8.2 Navigation page

here the user select the required floor in communication building from the first popup menu then select the required destination from the second menu and click "submit" button which store the target in the destination table in the database and by click on "navigate" button the action occurred is fetching the start and destination point then locate them on the map and execute the A\* algorithm to draw the shortest path.

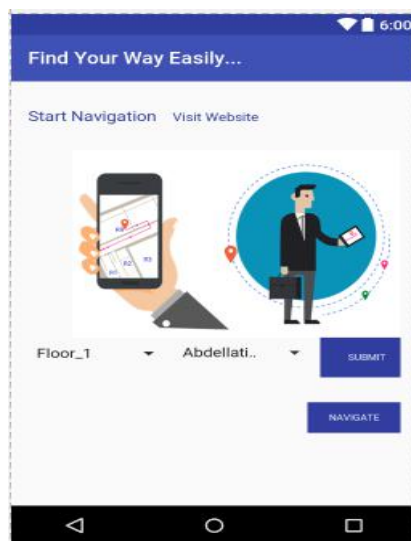


Figure 8.3



But this way is not completed as it's requires AJAX calling function which is a JQUERY function that it is used to send a request to the web server and also needs php codes to receive or store data on the server database so, as we work on android studio which handles javascript codes and due to time limitations we use the second kind of mobile app's which is the web based app.

Here web based app acts like a web viewer which access the website and online server created before for our project through the URL and by using webview framework on android studio.

Webview is not just a web viewer it having two main advantages:

First, allows us to specify viewport and style properties that make our web pages appear at the proper size and scale on all screen configurations.

Second, we can even define an interface between our Android application and our web pages that allows JavaScript in the web pages to call upon APIs in our Android application—providing Android APIs to our web-based application.

So, we develop our code on android studio then by using the appropriate driver (installed on laptop to transfer the app. to mobile) such as mobilego for Huawei GR5.

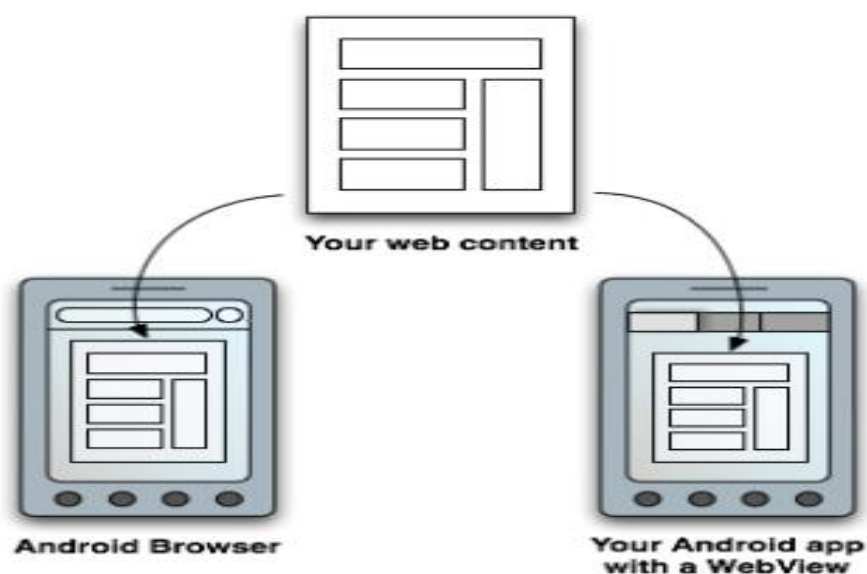


Figure 8.4

## **Chapter 9: Conclusion and Future work**

### **9.1 Conclusion**

Therefore, From this system we can:

1) Customers and visitors could easily navigate through indoor buildings. They can select destination they want to reach and a path to this destination will be drawn on map. But the point detects user's place with some errors in reading, so it is not accurate detection.

2) Monitoring valuable assets in stores, kids at kindergartens, etc.

3) Retailers could send relevant advertising for promotions other thing in malls for example.

4) In banks, airports, hospitals and any public places could send awarenesses to their customers and visitors.

### **9.2 Future work:**

1) Mobile application that will be installed on mobile that will scan the beacons instead of raspberry pi and the navigation will be from it.

2) Trilateration is not accurate so we can increase accuracy by using fingerprint. In order to increase the accuracy of the detection point.

3) Anomaly detection is the problem of identifying data points that don't conform to expected (normal) behavior. We implement a python code that take two features of group of devices to determine anomaly devices.

**Results:**

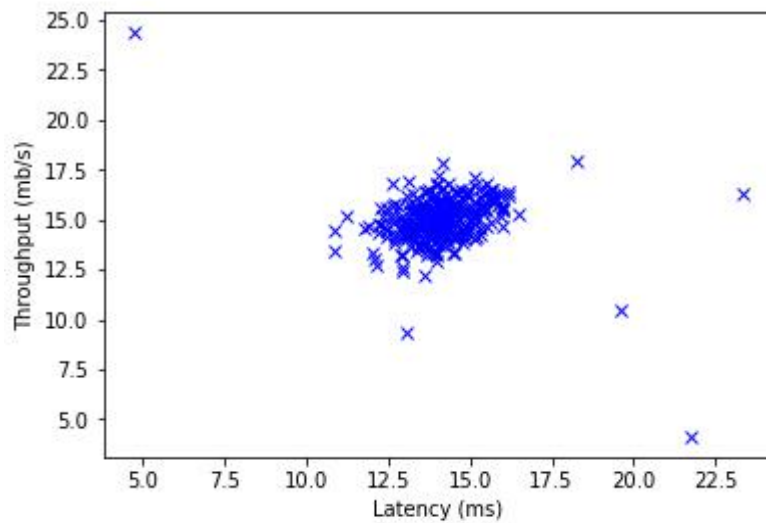


Figure 9.1

Here each device is implemented as a point in the 2D plane .

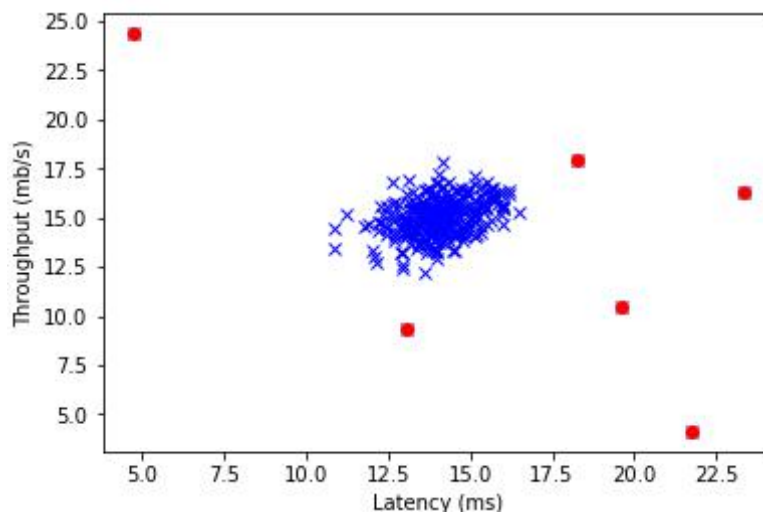


Figure 9.2

Anomaly devices marked as a red points. We can use the same code to detect anomaly beacons by read telemetry packet (which contains sensors readings of beacons) using

scanner and extract two features like temperature and vibration but as this code require three types of files for beacons features: training, validation and test set and these files must contain large number of readings about 400 which requires a lot of experiments so, due to that and also time limitation we can't complete this part.

## References

- [1] Chuan-Chin Pu, Chuan-Hsian Pu, and Hoon-Jae Lee (2009).
- [2] Front end and back end from <<https://www.quora.com/Is-PHP-a-Backend-or-Front-end>>
- [3] Getting started with Beacons from <[estimote.com](http://estimote.com)>
- [4] IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 2, FEBRUARY 2002 Page 55
- [5] Lamp <<https://www.ibm.com/developerworks/web/tutorials/wa-lamp/wa-lamp.html>>
- [6] Milan Herrera vargas, Indoor navigation using bluetooth low energy (BLE) beacon, Turku: 2016.
- [7] N. Bergman, "Recursive Bayesian estimation: Navigation and tracking applications," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1999.
- [8] Pu, C.-C. (2009). Development of a New Collaborative Ranging Algorithm for RSSI Indoor Location Tracking in WSN, PhD Thesis, Dongseo University, South Korea. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking, M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp.