A

Dissertation on

"Indoor Positioning using MEMS Sensor"

Submitted in Partial Fulfillment of the Requirement
For the Award of Degree of

Master of Technology

In

Software Technology

By

Aniroop Mathur
University Roll No. 2K14/SWT/502

Under the Esteemed Guidance of

Mr. Manoj Kumar
Associate Professor, Computer Science & Engineering

2014-2017

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
DELHI – 110042, INDIA

CERTIFICATE

Delhi Technological University
(Government of Delhi NCR)
Bawana Road, New Delhi-42

This is to certify that the thesis entitled **"Indoor Positioning Using Mems Sensor"** done by **Aniroop Mathur** (Roll Number: **2K14/SWT/502**) for the partial fulfillment of the requirements for the award of degree of **Master of Technology** in **Software Technology** in the **Department of Computer Engineering**, Delhi Technological University, New Delhi is an authentic work carried out by him under my guidance.

PROJECT GUIDE:
Mr. Manoj Kumar
Associate Professor
Department of Computer Science
Delhi Technological University, Delhi

DECLARATION

Delhi Technological University
(Government of Delhi NCR)
Bawana Road, New Delhi-42

I hereby certify that the work which is presented in the Major Project entitled **"Indoor Positioning Using Mems Sensor"** in fulfilment of the requirement for the award of the Degree of Master of Technology and submitted in the Department of Computer Science and Engineering, Delhi Technological University, New Delhi is an authentic record of my own, carried out during a period from August 2017 to April 2018, under the supervision of Mr. Manoj Kumar, Associate Professor, Delhi Technological University, New Delhi.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University.

Aniroop Mathur
2K14/SWT/502
M.Tech, Software Technology
Department of Computer Science
Delhi Technological University, Delhi

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The satisfaction of successful completion of any task would be incomplete without the mention of people and university whose ceaseless cooperation, constant guidance and encouragement made the task possible. I would like to extend my sincere thanks to all of them.

I am highly grateful to **Prof. Manoj Kumar, Associate Professor, Department of Computer Science and Engineering, Delhi Technological University** who had been a source of inspiration and for having permitted me to work on this project. I would like to express my deep sense of gratitude for his able guidance, generosity and useful suggestions which helped me in completing the first part of this big project. In this respect, I find myself lucky to have him as my guide.

Also, I would like to express my heartfelt thanks to the entire teaching and non-teaching staff in the Department of Computer Science and Engineering, DTU for such attention and time. Kudos to all my friends for thought provoking discussion and making stay very pleasant.

Aniroop Mathur
2K14/SWT/502
M.Tech, Software Technology
Department of Computer Science
Delhi Technological University, Delhi

ABSTRACT

In the outdoors having open air, a GNSS (GPS, Glonass) based system is able to estimate the position with a very high precision. However, in the inside of the building where the GNSS signals are very bad or almost not available, the accuracy from GNSS receivers are fallacious and are of no use in reality. The main motive of this project is the determine the position of the person using only the embedded mems sensors present in android mobile phones. Examples of such navigation are large hospitals, large office campus, big college campus, big shopping malls, big museums, large exhibition grounds, big train stations, etc.

The device used for this project is Samsung Galaxy S7 edge smartphone. This mobile has all the required sensor present so this device was chosen for this project. The Micro Electro Mechanical System (MEMS) based sensors present in these current mobile phones such as accelerometer sensor, gyroscope sensor, magnetic field sensor and barometer sensor allow navigation in GNSS shadowed areas. This project uses accelerometer sensor for calculating steps by an individual and distance travelled is determined using step counts, a magnetic sensor is used for detecting turns (90, 180, 270, 360 degrees) an individual makes using azimuth value derived from magnetic field generated by earth and a barometer sensor is used for detecting height changes for floor detection. Further, a tabular based map is used for pointing individual location which will also avoid false locations. The advantages of using sensors for estimating location are that they are very low cost, there is no installation time required and there is no need to make any major change with change in infrastructure of the building.

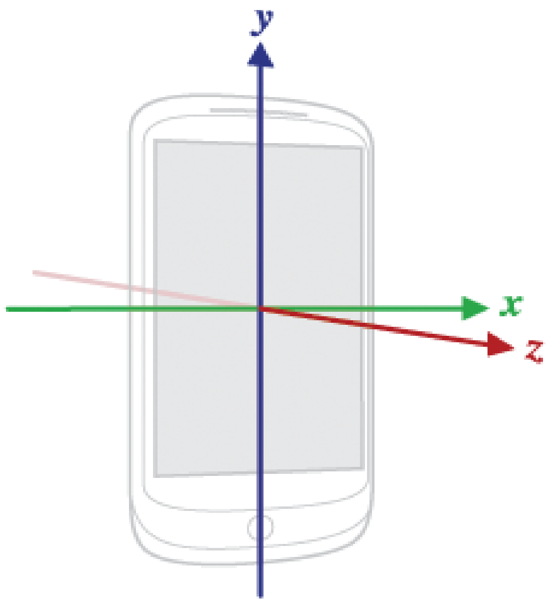
As this is a very big project so it was divided into sub projects. Floor Detection is already done in minor project 1. In major project I, position estimation is done using built-in step counter sensor and heading estimation is done. In this major project, user location is determined using an own-designed step counter algorithm based on accelerometer sensor as built-in step counter sensor was found to be not suitable and a full-fledged map of 10th floor and ground floor of Samsung R&D Institute is created using a grid based approach. A person is placed initially at fixed location and then as per person’s direction and number of steps, position is changed accordingly. Considering correct motion occurs with steps of the person, it turns out that with this approach we are able to achieve 98 % accuracy

Chapter 1. INTRODUCTION

This section describes basic description of sensors axis and various sensor used in this project

1.1 Sensor Axis:

In the case of sensors, the co-ordinate system axis is shown in below figure. The X-axis is the axis passing through home key button and Y-axis is perpendicular to it in the same horizontal plane. The Z-axis lies perpendicular to x-y plane of the mobile phone. In the case of vehicle co-ordinate axis system, the a-axis point in the direction of car travel and y-axis is perpendicular to it in the same horizontal plane. So this sensor co-ordinate system's x and y axis are swapped than normal co-ordinate systems such as used in other vehicles like car. Also there are positive and negative sides of them. For example, for the case of x-axis, the direction to the right of home key is positive x-axis and the direction to the left of home key is negative x-axis. So if the force is applied is positive direction, it is positive and if the force is applied in negative x direction, it is negative as well. It is important to note that axis for sensor are not changed upon changing the orientation of the phone. The sensor axis coordinate is static in nature. So if we hold the phone in vertical position or horizontal position or downward position, the axis system still remains same as before.



Co-ordinate axis system used by the smartphones

1.2 Accelerometer Sensor:

In this project, we are using accelerometer sensor to detect step ad calculate step length and thus distance travelled by an individual.

Accelerometer sensor is a sensor which measures the acceleration applied on the device. The accelerometer chip measures the total acceleration including the gravity but the chip is attached to the PCB which is holding the accelerometer IC so there is always a 9.8 m/s^2 acceleration acting in upward direction. So when the phone is lying still on the table then also we can see that there is acceleration of 9.8 m/s^2 although the device is at rest so its acceleration is zero. Therefore, zero is the acceleration of the device but we get the acceleration acting upon the accelerometer IC. If we drop the phone as free fall, then total acceleration of the device will be zero. When the mobile phone is pushed towards the right side, the acceleration along x-axis is positive and when it is pushed towards the left side, the acceleration values are negative. Similarly, if the mobile phone is moved towards the sky, then the acceleration value along z-axis is positive and when it is moved downwards, then acceleration value is negative.

The accelerometer sensor is a capacitive based sensor. Basically, there are 3 pairs of capacitive plates are placed in each of the 3 axis. In each pair, 1 capacitive plate is flexible to move while the other plate is fixed. So when there is movement in the device, only 1 plate is moved. And when 1 plate is moved, then distance between two capacitive plates changes. And when distance between two capacitive plate changes, the value of total capacitance changes. This change in capacitance value is mapped to change in voltage. So basically, the mechanical energy is converted into electrical energy. For example, if the right plate is flexible and when device is moved in right direction, the plates will come near to each other so this can be interpreted as increase in acceleration value in positive x direction.

The accelerometer sensor is a continuous sensor which means it senses the change in acceleration continuously and hence generate the new value continuously and store in the registers. The frequency of the accelerometer sensor used in the smartphone are 200Hz, 100Hz, 50 Hz, 15 Hz, 5 Hz. 200Hz means accelerometer IC could generate a new event in every 5 milliseconds. 100Hz, accelerometer IC could generate events every 10 milliseconds. Now a days, there are even accelerometers which run at 500 HZ. When the accelerometer runs at faster rate, there is good performance but more current consumption so the rate should be chosen by the developer very carefully as per his requirement.

The accelerometer chips are calibrated in the factory by keeping the device flat on the table and without any movement which removes bias from it. For example, if accelerometer output 1 m/s^2 in flat position in x-axis then after calibration it will becomes zero. Accelerometers are also calibrated by IC vendor to remove noise due to temperature variations. All the measurements are done by the IC in SI units which is m/s^2.

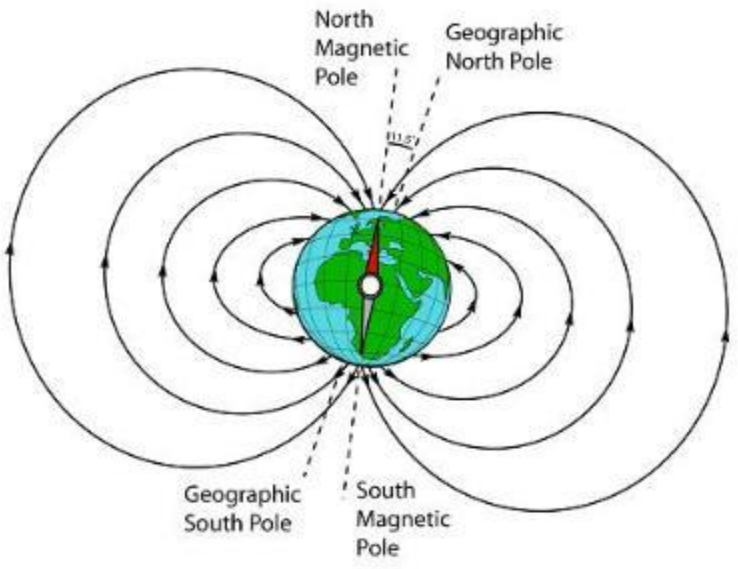
1.3 Magnetic and Compass Sensor

In this project, we are using magnetic field sensor to detect heading of an individual i.e. whether individual is pointing in north/east/west/south direction and to detect turns.

Magnetic sensor is a sensor which measures the magnetic field intensity in the environment. It senses the internal magnetic field, the external magnetic field and the magnetic field generated by the earth itself. Internal magnetic field is the field generated due to various components of the smartphones on the PCB, the external magnetic field is the field due to laptop, earphones, etc. The magnetic sensor measures the magnetic field in uT (Micro Teslas) and it measures the magnetic field in all of 3 axis. The earth has maximum magnetic field in the north direction so when any of the axis points to north, then that axis has highest value of magnetic field considering there are no external magnetic field present.

The compass sensor is derived from magnetic sensor and accelerometer sensors. Compass sensor outputs azimuth, pitch and roll of a device. Azimuth is derived from magnetic sensor while pitch and roll are derived from accelerometer sensor. Azimuth denotes the number of degrees required to rotate the device over z-axis such that its y-axis point in the north direction. So when the y-axis of the phone point to earth's north direction, azimuth value is 0. And when phone y-axis point to earth's south direction, its value is 180 degrees. When it points to east, it is 90 degrees, when pointing to west it is 270 degrees. Azimuth is always a positive value. Therefore, using the azimuth value, we can know that phone y-axis is pointing to which earth's direction whether it is east/west/north/south and therefore can tell a person's heading. Pitch value denotes the rotation performed along x-axis while Roll value denotes the rotation performed along y-axis. Pitch has a range from +90 degrees to -90 degrees and Roll has a range from +180 degrees to -180 degrees. For example, if Pitch is 90 degrees, we can tell phone is standing upright with y-axis pointing to the sky. And if Roll is 180 degrees, the phone is in the landscape position. So using all 3 values, azimuth, pitch and roll, we get the complete orientation of the device. So this sensor is also called as Orientation Sensor as it gives exact orientation of the device.

The compass sensor can be used only after calibration. Because if we need to know the north direction we need to remove the magnetic fields generated from outside device and from within the device. The magnetic field present within the device is fixed so the vendor itself removes it before proving the IC to the OEM. But for external magnetic fields, calibration need to be performed by the user. To perform the calibration, the person need to move the device in 3D direction and making an 8 number shape. By doing this motion, accuracy level gets changed to 1/2/3. 3 is the highest accuracy level and mostly this sensor is used at accuracy level of 3. At level of 2/3, north direction is rightfully know. However, one should know that there is true north and magnetic north which are separated by around 15 degrees. This sensor gives magnetic north direction using azimuth value.



Earth’s magnetic field

1.4 Gyroscope Sensor

In this project, we propose to using gyroscope sensor to detect quick turns taken by an individual during his/her navigation. However, for normal turns we are using compass sensor derived from magnetic sensor.

A gyroscope sensor is a sensor which measures the change in angular velocity of the device. In other words, it measures how many degrees the phone is rotated along x-axis, y-axis and z-axis. It measures the angular velocity in SI units which is radians per second which can then be converted to degrees per seconds for quick understanding of the output.

When the phone is placed still on the table i.e. when there is no movement, then x/y/z axis values are zero because it measures the change in angular velocity and not the absolute velocity. So its value is zero in flat position, landscape positive, upside position, etc. So this sensor is not to detect orientation of the device as most people assume about this sensor. Usually, when the phone is at rest, gyroscope value along the three axis is not exactly zero so a calibration is required to fix it which is to make it to zero along all three axis. Calibration is performed simply by taking many samples a rest and then the the average of those values are calculated and that average value is considered as noise or bias so it is subtracted from every value generated. With this, there are calibrated gyroscope sensor and there are uncalibrated gyroscope sensors. Uncalibrated gyroscope sensors outputs six values which are uncalibrated angular velocity in x/y/z axis and bias values in x/y/x axis.

When the phone is rotated in the anti-clockwise direction, the change is positive and when the phone is rotated in clockwise direction, the values are positive. Gyroscope sensors are continuous type of sensors so it senses change in velocity continuously and generate events. They usually have capability to run at 200Hz, 100Hz, 15Hz, 5 Hz. It is important to note that gyroscope sensors are totally different from accelerometer and compass sensor. Accelerometer and compass sensors outputs absolute value while gyroscope reports relative values so gyroscopes cannot be used to detect absolute orientation of the change but it could be very well used to detect the change in orientation of the device especially when the device is moved with a high speed as it has a high responsiveness quality.

1.5 Barometer Sensor:

In this project, we are using barometer sensor to detect the floor of a building.

A barometer sensor is a sensor which measures the atmospheric or ambient pressure. It measures the pressure in SI units which hPa (HectoPascals). Using the atmospheric pressure value, height value is derived using the fixed formula. This height value is the height above sea level and the formula is based on value of sea level pressure. But the pressure at sea is different in different regions of the world. For example, there is a different sea level pressure in USA and different sea level pressure in India. So it Is important to use the correct sea level pressure value to get correct value of atmospheric pressure.

Even in the same region, atmospheric pressure could be different at different times of the day. Atmospheric pressure could be different in morning, day and night and even different after some hours or even after some minutes. It is because atmospheric pressure depends on climate and on temperature which changes the density of air and hence pressure of air changes. However, we need to detect changes in height of floor and not the absolute height

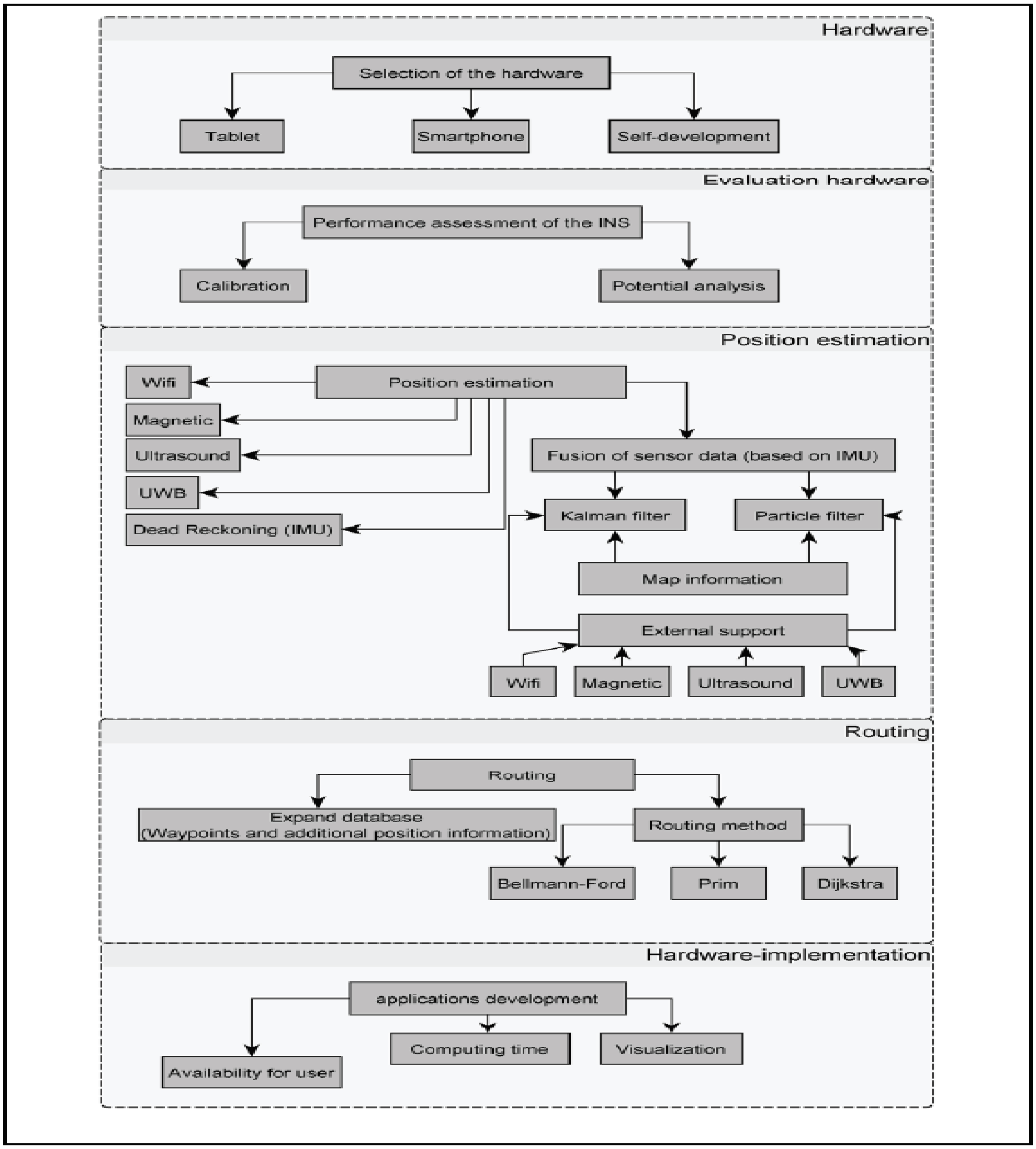
1.6 Concept of building indoor navigation system

Currently, navigation systems based on GNSS technology has become a necessary part of our life. However, GNSS technology does not cover the all the areas of the world because GNSS signals are not received on those areas by the smartphones, so they are failing in those areas. Examples of such areas are big hospitals, big offices, big college campus, big museums, big metro train stations, big exhibition grounds, etc. A lot of research is ongoing in indoor positioning to have an acceptable working of navigation in GNSS unavailable or bad quality signal areas. Amazon, Google, Microsoft, Samsung all such big companies are trying to develop an indoor navigation system but currently it is not a success yet. So this research work is to determine the accuracy using MEMS sensor which are low cost devices.

Previous technologies for indoor positioning determine distances to reference points wirelessly. They use technologies, such as Radio Frequency Identification (RFID), wireless, ultrasonic, optical methods and also the magnetic field. There are already dedicated solutions which are often implemented at great expense. Due to high demands on the infrastructure and the clients' implementations can often be realized in a single room / building or a factory only. To minimize the effort of the infrastructure measures inertial sensor is increasingly used, which is for example also implemented as MEMS design (Micro Electro Mechanical System) technology in smartphones.

Due to the efficiency and the quantities of these sensors in a smartphone, a navigation solution with this sensor is obvious. The heavily widespread use of smartphones in society, for example by the GNSS-based navigation is another decision criterion. The goal is to develop an algorithm which is based solely on smartphone sensors. Therefore, only minor interventions are necessary in the infrastructure of the building intended for navigation. The expectation of the accuracy should be between 1-3 meters.

Next, let's look at the figure which depicts the overall idea of building an indoor navigation system. Our project is developed in a manner confirming to this concept. However, in this project work, we will not be using any external sources to determine position of the person as we want to make the system as low cost as possible and not to be impacted with change in infrastructure and not to have any installation time.



Concept of building an indoor navigation system

1.7 Tools Used

Samsung Galaxy S7 edge, K6DS3TR accelerometer sensor chip, YAS537 magnetic sensor chip, LPS25H barometer sensor, Android OS, Java, Android Studio.

Chapter 2. RELATED WORK

This chapter discusses in brief the various technologies which can be used to determine the position of an individual, their functionalities and their error characteristics

2.1 Wi-Fi

Communication across a wireless network is like two-way radio communication. A device’s wireless adapter converts data into radio signals and transmits it via an antenna. A wireless router receives the signal and decodes it, and then it sends the information to the Internet using a physically wired, Ethernet connection. The process works the same way in reverse, with the router receiving data from the internet converting it into radio signals, and sending it to the device’s wireless adapter. Each wireless router broadcasts a signal that is received by devices in the area. These devices have the capability to measure the strength of the signal. This strength is then converted to a number known as received signal strength indicator (RSSI). Wi-Fi devices, such as smartphones, typically perform this conversion automatically in order to provide signal strength information to applications running on it.

At start a uniform wave front leaves the transmitting antenna (AP). Obstacles encountered by the wave front may create new RF signals. Some components will travel straight to the receiving antenna while other components could diffract, scatter or be reflected by obstructions. Diffraction, the radio waves are bent around sharp objects creating a new wave front. Scattering, RF energy is reflected by a non-uniform surface in multiple directions. Reflection, the wave contacts a uniform surface and is reflected with a predictable angle. Multipath fading, occurs when receiving the superposition of multiple copies of the transmitted signal, each traversing a different path. Two problems with the LF approach are identified. Signal strength variation and changing environment i.e. movement of people, varying arrangement of furniture, etc. Consequences of the above will result in different values obtained during the (offline) recording of the signal strength values compared to those observed during real-time running of the application. Averaged signal strength values can be used reduce the problem.

2.2 Bluetooth

Bluetooth is a wireless technology standard used by two (or more) devices for communication over short distances. The big advantages of Bluetooth are that it is wireless, inexpensive and automatic. It uses a technique called spread-spectrum frequency hopping, which allows connection between several devices simultaneously, without them interference

The Bluetooth scan does not include any information about the signal strength, there is no possibility to estimate an exact position in the building. Although the information can be used to detect and locate a user within a larger part of the building, because of the relative long scan period, that information will only be valid during a short time. The Bluetooth scan process is very costly (in CPU terms) and it takes in average 12 seconds per scan. As the distance the user can move in 12 seconds can be relatively high, even the obtained area location could be inaccurate. The facts above verified that Bluetooth did not add any value to the project.

2.3 Geo-Magnetism

This technique measures location using disturbances of the Earth's magnetic field caused by structural steel elements in a building. The presence of these large steel members warps the geomagnetic field in a way that is spatially varying but temporally stable. To localize, the magnetic field is measured using an array of e-compasses and compare the measurement with a previously obtained magnetic map. Accuracy is found to be within 1 meter 88% of the time in experiments in two buildings and across multiple floors within the buildings.

The system consists of a server that contains a magnetic fingerprint map and a client device that measures the magnetic signature of its current location. When the client collects the signature, it sends the data to the server for localization. The server compares the measurement with all the fingerprints stored in its map and reports back the position associated with the closest matching fingerprint.

2.4 Computer Vision

A collection of successive snapshots from a mobile device's camera can build a database of images that is suitable for estimating location in a venue. Once the database is built, a mobile device moving through the venue can take snapshots that can be interpolated into the venue's database, yielding location coordinates. These coordinates can be used in conjunction with other location techniques for higher accuracy.

Overall as a conclusion, the above technologies are good of indoor positioning but there are some disadvantages in using them like very high cost, high maintenance, long installation time, infrastructure intervention and high efforts involved with change in infrastructure.

Chapter 3. PROPOSED WORK

In this thesis, we propose to estimate position of an individual in indoor environment using mems sensors found in smartphones as now a days smartphones are very popular and every individual carries smartphone with him for variety of purposes. We propose to use accelerometer sensor, compass sensor and barometer sensor for indoor positioning which could detect position with high accuracy with a low cost investment. In this project, accelerometer sensor is used to detect steps taken by an individual, compass sensor is used to determine the change in direction of an individual and barometer sensor is used to detect change in floor of a building. To detect the very first position, we propose two ways for this project. First, install NFC tags with unique IDs at various required places in the building over which an individual need to tap his mobile after which software knows the first location. Second, make the individual to start from the same location which is manually entered by him while choosing source and destination in the application software before starting the navigation. Let's discuss the techniques used in the project in more detail.

3.1 Step Detection

To detect a step, accelerometer sensor can be used. In our project, we have implemented our own step detector algorithm. The motion can be tracked in any of x/y/z-axis depending upon how the person has held the phone in his hand. For the case when phone is held such that LCD screen is facing updwards, z-axis motion need to be monitored. In this case, the acceleration in z-direction is 9.8 m/s^2 when the mobile phone is kept still. We have designed the algorithm to detect the step when the z-axis data keep on increasing for at least 10 continuous sample and crosses threshold of 11 m/s^2 and then z-axis data starts to decrease for at least 5 continuous sample and crosses threshold of 9 m/s^2. This kind of motion will be determined as a single step. In current smartphones, we also have a step detector sensor available but this sensor is not suitable for our project because it detects the first step only when it sees the pattern for 6-7 steps and then after that it detect the first step otherwise it ignores the motion for 1-6 steps. After 6-7 steps detection, it starts to detect each step. This kind of motion is suitable for usual step counting but in our case, we need to detect the step as soon as it is made by the person and update the location for that step taken. In our case, we cannot wait to update the location after 6-7 steps as this may be interpreted as if the software is not able to detect the steps because location change did not occur. Therefore, it is important to use self-designed algorithm for indoor positioning. Source code implementation in brief and data analysis could be seen in Section 5 [Implementation and Results] and Section 4 [Data Acquisition and Analysis] respectively.

3.2 Calculate Distance

To calculate distance, we need to calculate the length of every step taken by analyzing the changes in accelerometer data more deeply. Basically, we need to check how high and how low accelerometer value gone for every step and also how much time did it took to complete 1 step pattern. Based on these, we can determine step length. For example, if step pattern has high accelerometer value and has a larger time period then it is a longer step than an average step. Source code implementation in brief and data analysis could be seen in Section 5 [Implementation and Results] and Section 4 [Data Acquisition and Analysis] respectively.

3.3 Direction Detection

To detect direction, compass sensor is used which is derived from magnetic sensor. Using azimuth value, north/east/west/south direction is known using which initial direction of the map and the person could be determined. For change in direction, change in azimuth value need to be monitored. For example, if there is a change of +90/-270 degrees, it denotes that the person turned in right direction. Change of -90/+270 denotes left turn & +/- 180 denotes turn back. Source code implementation in brief and data analysis could be seen in Section 5 [Implementation and Results] and Section 4 [Data Acquisition and Analysis] respectively.

3.4 Floor Detection

To detect floor, barometer sensor is used. Barometer sensor gives pressure value which could be converted to absolute height value above sea level. However, absolute height cannot be used for detecting a floor because pressure keeps varying with temperature so height of the same floor could be different at 11 AM, 2 PM and 9 PM. Change in height is what is required here to detect a floor. First, difference in height of each floor need to be recorded. Then simply change in height value will denote change in floor. However, initial floor need to be entered by person manually by entering source and destination before starting navigation or by tapping on NFC tag installed on each floor. Source code implementation in brief and data analysis could be seen in Section 5 [Implementation and Results] and Section 4 [Data Acquisition and Analysis] respectively.

3.5 Map Creation and Location Mapping

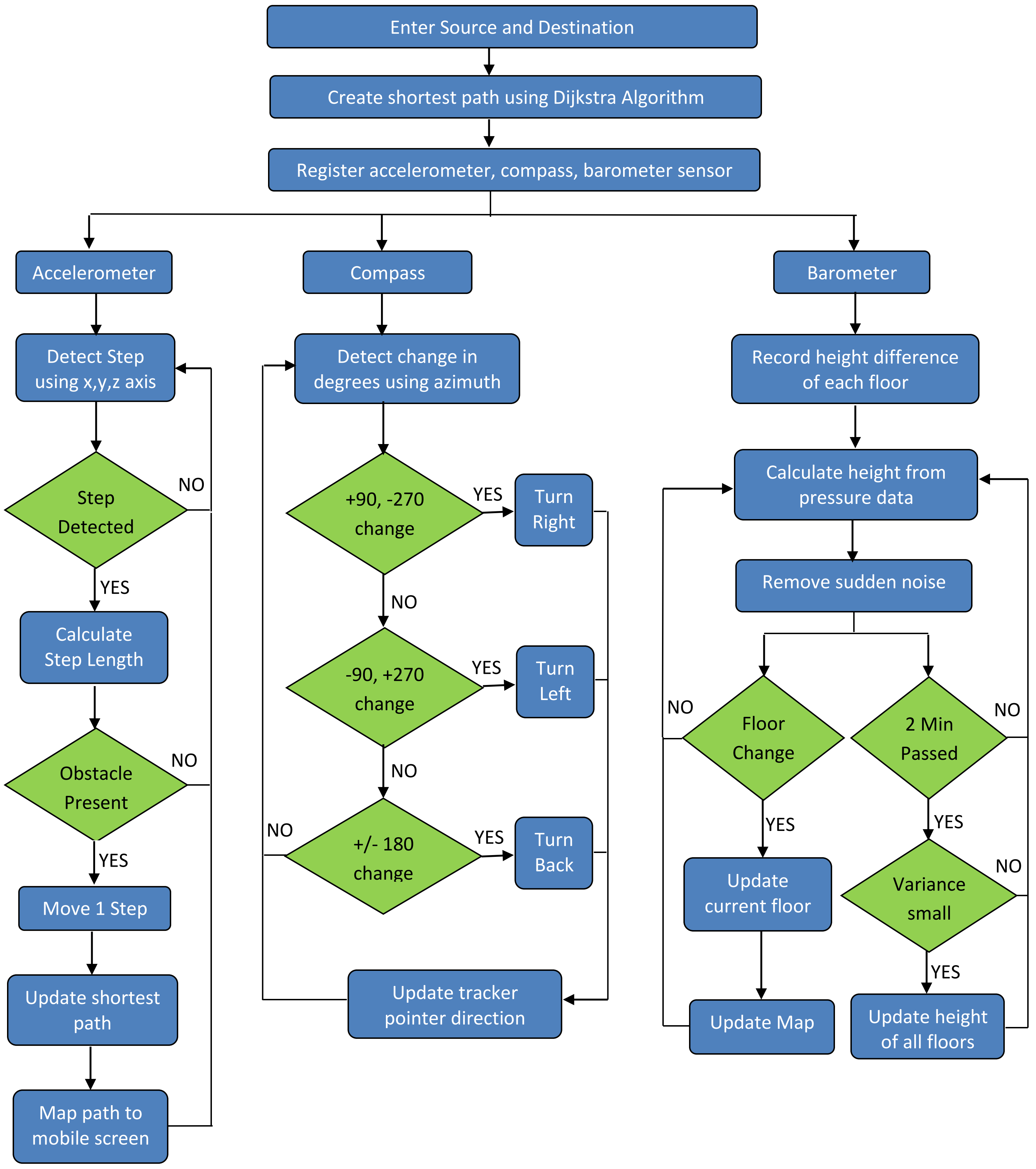
Creating an indoor map is a quite a big challenge as it is very huge in size. In case of outdoor locations, map is created using satellite images but this method does not work for indoors as satellites could not capture images inside of a building. In this thesis, we created the map using tabular approach. We divided the area of a floor into rows and columns creating a very big table. We used excel sheet for this purpose. Each cell of a table denotes 1/3 of a step so a block of 3*3 denotes one single step. So 1/3 of a step is the smallest unit used in our map. To draw the map, each cell is filled with some ID which denotes to some infrastructure component of the floor. For example, WL is used to mark the wall, FN is used to mark the fence, LO is used to mark lobby, ENT/EXT are used to mark floor entry and exit, 1/2/3 are used to mark seat numbers in a cubicle, DS is used to mark the desk and so on. So each of the cell is mapped to some location of the floor area. To fill these IDs on the whole map area at correct row/column number, measurement was taken for various infrastructure components in terms of step unit. For example, a cubicle is of 6 step * 5 steps in size, a conference room is of 12 steps * 6 steps in size, a lift is of 4 steps by 4 steps in size and so on. When each of the cell of the big table is filled with some ID, whole map is created and each cell of the map is mapped to some defined location so that a person can be tracked in an easy way. But this map is created in the excel sheet and we need to put this data on the mobile screen. To do this, each cell is considered to be of 20 pixel * 20 pixels, so a single step is 60 pixel * 60 pixels. So when a step is made, a person is moved by 60 pixels. To draw the map on the screen, excel sheet data is put into the text file with comma separated format, then file is read line by line which further splitted using comma format into IDs is. Each ID represent to a color. For example, WL denotes yellow color, LO denotes grey color, TBL denotes brown color and so on. When all the line of the map file is executed, whole map is drawn on the mobile screen.

3.6 Shortest Path Creation

To create shortest path from source to destination, Dijkstra Algorithm is used. First, a graph need to be created using map data and then a new shortest path can be calculated with each step by applying Dijkstra Algorithm again with the new source but same destination. The size of list which includes the nodes contributing to the shortest path will denote the minimum number of steps required to reach to the destination. More details are there in Section 5.

3.7 Flowchart

Next, let’s take an overview of the flowchart diagram representing proposed work.


Enter Source and Destination
Create shortest path using Dijkstra Algorithm
Register accelerometer, compass, barometer sensor
Accelerometer
Compass
Barometer
Detect Step using x,y,z axis
Step Detected
NO
YES
Calculate Step Length
Obstacle Present
NO
YES
Move 1 Step
Update shortest path
Map path to mobile screen
Detect change in degrees using azimuth
+90, -270 change
YES
Turn Right
NO
-90, +270 change
YES
Turn Left
NO
+/- 180 change
YES
Turn Back
NO
Update tracker pointer direction
Record height difference of each floor
Calculate height from pressure data
Remove sudden noise
Floor Change
NO
YES
Update current floor
Update Map
2 Min Passed
NO
YES
Variance small
NO
YES
Update height of all floors

Chapter 4. DATA ACQUISITION & ANALYSIS

For data acquisition, Samsung Research and Development Institute located in Sector 62, Noida, Uttar Pradesh is used. It has 10 floors, 74 cubicles on each floor, 5 conference rooms on each floor, two washrooms on each floor and two HOD cubicles on each floor.

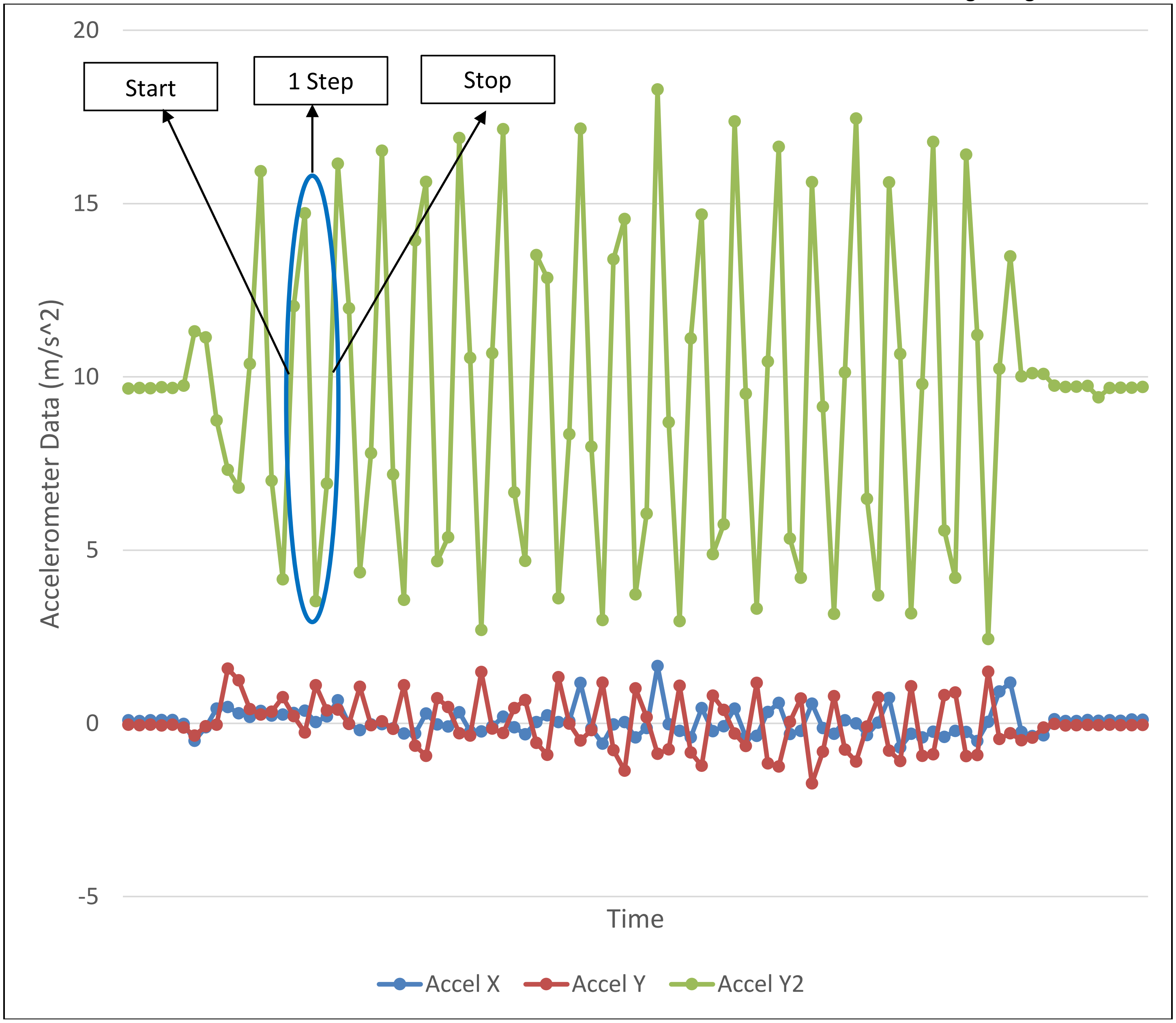
4.1 Walking Test

4.1.1 Accelerometer data while walking

To determine person's walking, accelerometer sensor is used. When the person walks, it makes a periodic pattern of peaks and valleys. A peak and its corresponding valley makes a single step complete as shown in below collected values and graph.

Sample	X axis (m/s^2)	Y axis (m/s^2)	Z axis (m/s^2)
1	0.089783	-0.03711	9.664195
2	0.068235	-0.05148	9.676167
3	0.093374	-0.03711	9.671378
4	0.100556	-0.05746	9.700109
5	0.096965	-0.03711	9.683349
6	-0.01077	-0.12091	9.745598
8	-0.11372	-0.0814	11.14261
9	0.430956	-0.03711	8.746018
10	0.472855	1.584962	7.319075
11	0.292093	1.244985	6.80073
12	0.189142	0.410606	10.38006
13	0.361524	0.257377	15.93341
14	0.226252	0.337582	7.006631
15	0.25618	0.754174	4.158728
16	0.301669	0.208296	12.03924
17	0.366313	-0.25857	14.72075
18	0.039504	1.100136	3.529053

19	0.204704	0.378284	6.92044
20	0.664391	0.399832	16.15129
21	-0.01556	-0.01796	11.98178
22	-0.19632	1.055843	4.359841
23	-0.02753	-0.05148	7.800309
24	-0.02394	0.058658	16.52837
25	-0.14006	-0.15802	7.188591
26	-0.2909	1.103727	3.563769
27	-0.29449	-0.64643	13.93665
28	0.282516	-0.93374	15.62815
29	-0.03232	0.724246	4.683059
30	-0.08739	0.471658	5.368997
31	0.32202	-0.28611	16.89468
32	-0.22865	-0.35314	10.55005
33	-0.23343	1.480814	2.699462
34	-0.09218	-0.15083	10.68413
35	0.196325	-0.27892	17.14608
36	-0.10894	0.442927	6.665458
37	-0.31125	0.671574	4.691438
38	0.039504	-0.56024	13.51886
39	0.233435	-0.90621	12.85447
40	0.039504	1.335965	3.610456
41	0.070629	-0.00838	8.346188
42	1.170765	-0.49081	17.16044
43	-0.14246	-0.19154	7.983466
44	-0.58179	1.180342	2.981978
45	-0.02873	-0.76854	13.39317
46	0.0383072	-1.365892	14.554352
47	-0.402226	1.0139444	3.7241807
48	-0.131681	0.1795651	6.0525417
49	1.6579847	-0.872687	18.2917



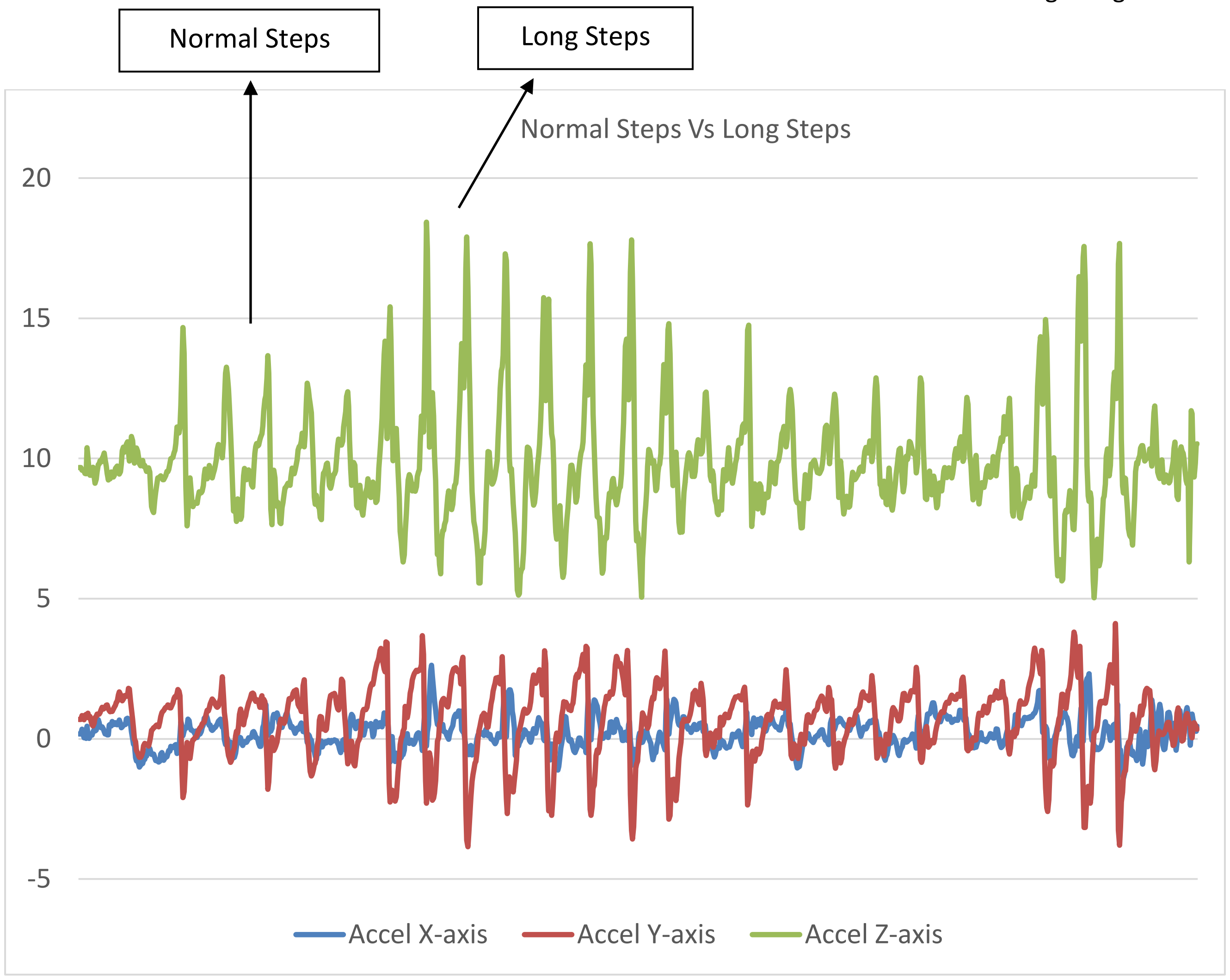
As can be seen from above graph, 1 Peak + 1 Valley combines to create 1 single step. This data is collected when the person is walking by keeping a mobile phone in in flat position such that phone's LCD Screen or z-axis is pointing upwards and x-axis and y-axis of the phone lie parallel to palm of the hand. In this position of the phone, when the person takes a step, maximum force is applied in z-axis and hence there is maximum acceleration in z-axis. In comparison to z-axis, there is no much movement in x-axis and y-axis. Since z-axis is pointing upwards, so there is always a base acceleration of 9.8 m/s^2 due to gravity which is keeping the phone still in hand. When person walks, there is acceleration of at least 2 m/s^2 which results in the value of 9.8 + 2 which 11.8 m/s^2 or more.

4.1.2 Normal Steps Vs Long Steps

To detect whether the person made a long step or a normal step as before, accelerometer sensor is used. This is necessary to determine the step length of a person based on which distance travelled by a person is calculation. As we analyzed the data, it was found that when person takes a normal step, its acceleration highest value is a lower value but when the person takes a long step, the acceleration highest value is a higher value than usual normal step highest acceleration value. Interpreting this data can help to estimate the step length. Here is the data collected and graph obtained.

Sample	X axis (m/s^2)	Y axis (m/s^2)	Z axis (m/s^2)
1	0.107742	-2.09739	14.67212
2	0.282524	-1.88669	13.7575
3	0.399844	-1.23545	11.26746
4	0.373507	-0.55068	8.973742
5	0.421392	-0.12929	7.59464
6	0.701523	-0.35914	8.011244
7	0.646454	-0.63927	9.014444
8	0.421392	-0.70392	9.311335
9	0.308861	-0.37111	8.796566
10	0.301679	-0.11732	8.411087
11	0.28013	-0.03352	8.286585
12	0.234639	-0.00239	8.394327
13	0.162811	0.05028	8.430242
14	0.277736	0.081405	8.423059
15	0.047885	0.148445	8.396722
16	0.136474	0.256187	8.609812
17	0.162811	0.344776	8.810931
18	0.177176	0.395055	8.758257
19	0.548289	2.22907	9.744699
20	0.756591	2.351178	10.1206
21	0.684763	2.461314	10.4965
22	0.588992	2.26259	11.36802
23	0.550683	1.994431	12.28024
24	0.411815	1.570644	14.11664
25	-0.01915	2.011191	15.73517
26	-0.06943	3.146077	14.36804

27	0.502798	2.686376	12.06715
28	0.600963	0.462095	12.46699
29	-0.05028	-1.82444	14.74155
30	-0.70152	-2.56666	15.68011
31	-0.7518	-2.54272	13.79581
32	-0.73983	-2.4278	11.71279
33	-0.18436	-2.73426	10.87719
34	-0.23464	3.016786	10.5803
35	-0.28731	2.760599	11.28661
36	-0.18915	2.40864	12.43586
37	-0.48843	3.304099	13.33611
38	0.691945	3.23706	12.40474
39	1.395862	1.32164	12.26108
40	0.457306	-0.78053	15.83334
41	0.031126	-2.51638	17.66496
42	-0.03831	-2.73187	16.94668
43	0.584203	-2.40625	13.79581
44	1.101366	-1.75261	11.57632
45	1.383891	-1.65205	10.30257
46	1.362342	-1.1732	9.404712
47	1.266571	-0.59378	8.595447
48	1.161223	-0.39745	7.927444
49	0.938556	-0.5028	7.89153
50	0.672791	-0.33999	7.87477
51	0.521952	0.023943	7.467743
52	-0.23225	0.830813	9.447808
53	-0.2921	0.883487	9.198804
54	-0.3328	0.845179	9.239507
55	-0.29928	0.758985	9.905115
56	-0.2945	0.756591	10.38876
57	-0.28492	0.866727	10.03919
58	-0.18197	1.00799	9.574705
59	-0.05028	1.108549	9.277815
60	0.160416	1.223474	9.119793
61	0.158022	1.35516	9.390346
62	0.146051	1.417411	9.586677
63	0.071828	1.489239	9.830893

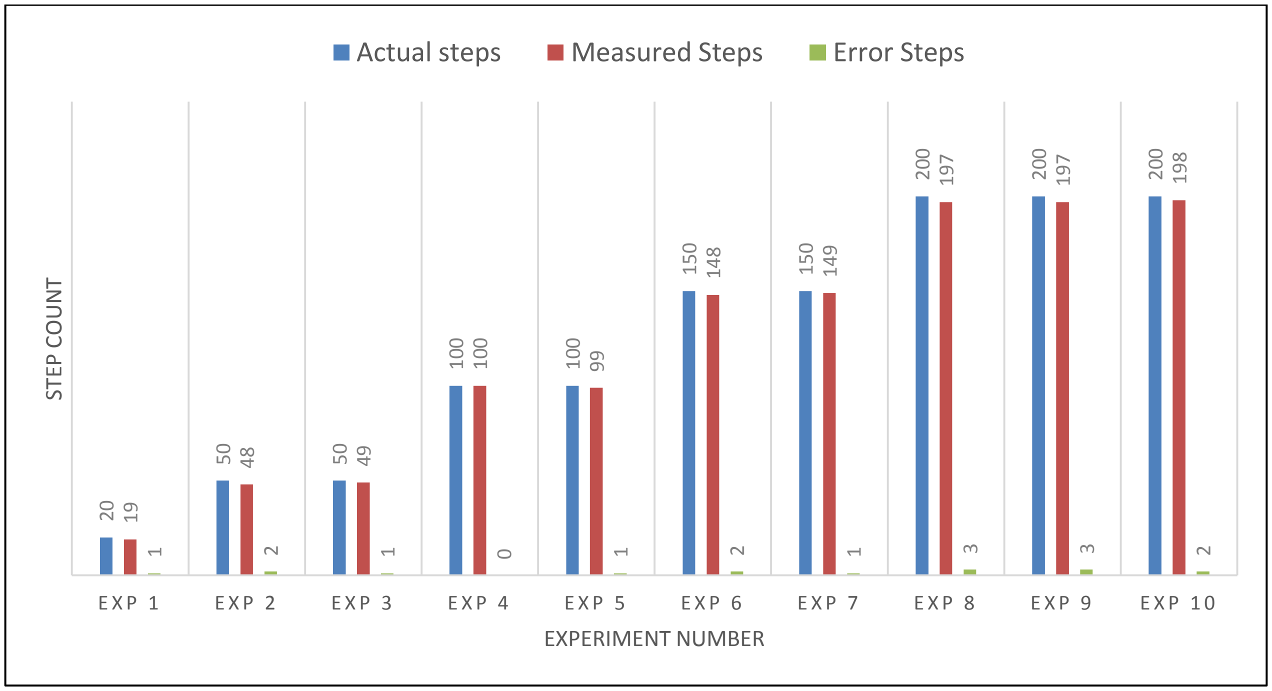


As can be seen from above graph, for the case when person is walking by carrying phone such that its LCD screen is pointing upwards so that there is more acceleration is z-axis direction, the highest value of acceleration is a much higher value in comparison to the highest value of normal steps. So based on this logic, if acceleration value is higher, it means the person has taken a long step. However, for this, we first need to store the highest value for normal step length of a person using calibration and based on this step length value, we need to estimate step length for other highest acceleration value. For example, from the graph we can see that for normal step case, values are near to 11 m/s^2 while from longer step length it is around 17-18 m/s^2. Along with this information, we can also notice that for longer length, the time period to get the acceleration from high value to a lower value is quicker but for shorter lengths, the time period of high value to low value is more. So based on this data, step length can be estimated which could lead to reduce error even further.

4.1.3 Measured Steps Vs Actual Steps

This is the data which compares measured steps with actual steps and determine the error in step count value for 10 experiments as shown in below table

Experiment No.	Actual Steps	Measured Steps	Error Steps
1	20	19	1
2	50	48	2
3	50	49	1
4	100	100	0
5	100	99	1
6	150	148	2
7	150	149	1
8	200	197	3
9	200	197	3
10	200	198	2



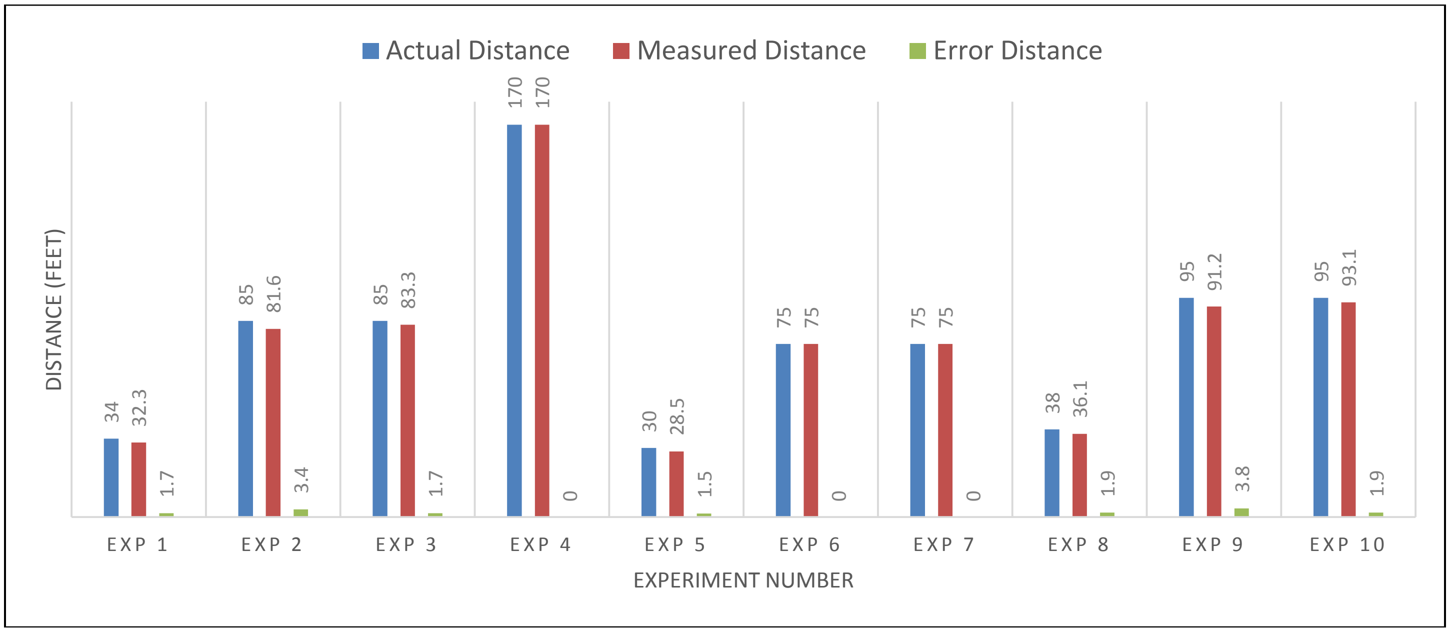
In this experiment, it was found that step count accuracy is very good. The maximum error found is of 3 steps. As observed, step detection is based on person's motion. If the motion is not performed well i.e. motion data does not match to step data pattern, then there will be error and step will be go undetected. This is usually a human error in taking a step as humans may not be able to make the similar step motion every time. But this step data pattern is very easy formed upon a usual step so there is very less chance of a step to be missed. As observed, the step is usually missed at the beginning of a walk i.e. first step. Once the walk starts and the person keep walking, then no steps are missed. When the person stops and walks again, then there is a chance of 1 step to go undetected. This is because the first step pattern is different from subsequent step data pattern. In first step, there is a less force applied and so there is less acceleration than second step onwards acceleration value. So sometimes the acceleration threshold to detect a step is not reached and so step is missed. To avoid this error, one solution is to change the starting threshold to a lower value than subsequent step threshold values. But with this solution, it was observed that even a small movement in the phone like for typing, vibration or making a turn, there is a good chance that a step is detected. Therefore, we did not change the first step threshold and kept it same for all steps. Second solution is to guide the user to take a first step little bit heavily so that it is detected well. With this second approach, the first step is not missed. Third solution is to mark the first step as a valid step only after second step is detected as well so that we know that first step motion pattern was indeed a step as there is a second motion data pattern detected as well. With this approach, first two steps are detected together and we can move the person ahead by two steps directly.

This approach is okay but as observed during navigation that there are cases when a person only need to make a single step to follow his destination path way. For example, to take a right turn immediately after making a single step, a person has to make a single step and then stop and make a right turn and then continue walking in that direction. So in this case, we will not be able to detect that single step before right turn was made as we are bounded to detect first two steps together. So in order to detect the first step in all cases, we did not used this approach. Hence, one way is to keep first step threshold low and person has to ensure to not make much movement after walking process started and another way is to take the first step with little more force than usual to make sure first step is detected well. In this project, we followed the second approach.

4.1.4 Measured Distance Vs Actual Distance

This is the data which compares measured distance with actual distance and determine the error for 10 experiments for different persons with different heights as shown in below table

S.No	Height (feet)	Step Length (feet)	Actual Distance (feet)	Measured Distance (feet)	Error (feet)
1	5.7	1.7	34	32.3	1.7
2	5.7	1.7	85	81.6	3.4
3	5.7	1.7	85	83.3	1.7
4	5.7	1.7	170	170	0
5	5.4	1.5	30	28.5	1.5
6	5.4	1.5	75	75	0
7	5.4	1.5	75	75	0
8	6.0	1.9	38	36.1	1.9
9	6.0	1.9	95	91.2	3.8
10	6.0	1.9	95	93.1	1.9



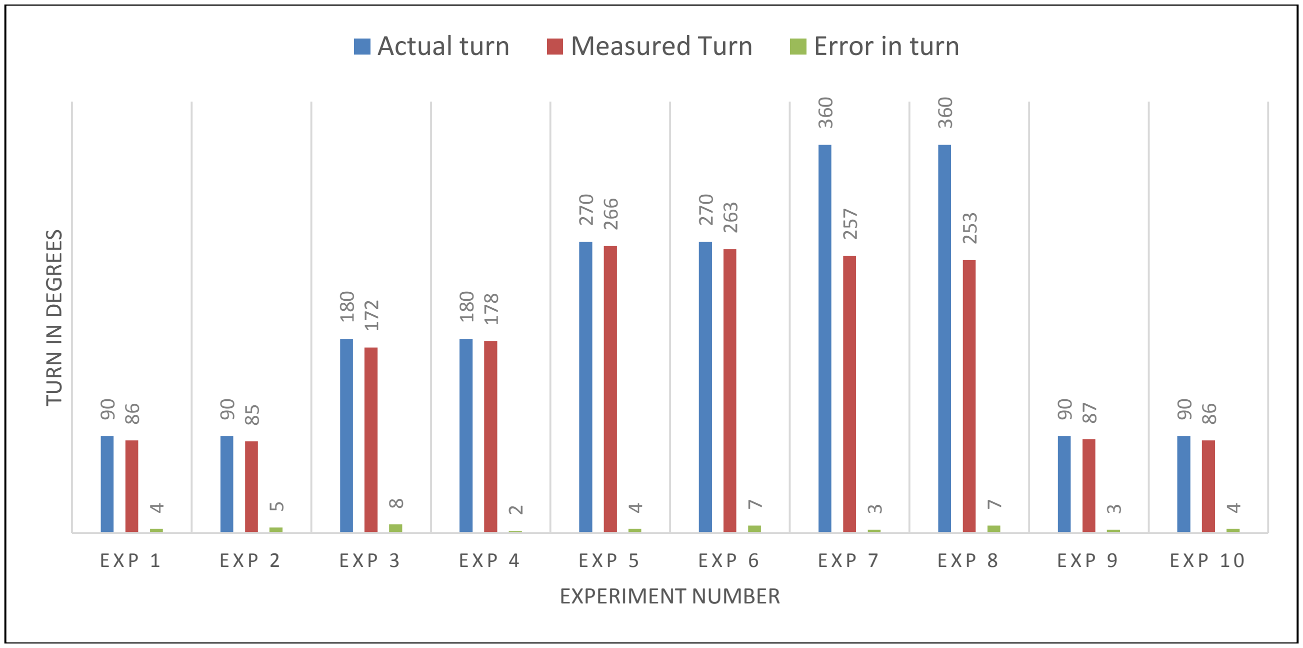
In this experiment, it was found that there is a maximum error of 3.84 feet i.e. 1.17 meters over a distance of around 100 meters. Every individual could have a different step length. If two persons have same height they may have an approximately same step length but there is a good chance they still have different step length. If two persons have a different height, then also there is good chance that their step lengths are different. So we conducted tests on different individual with different heights and different step length to check accuracy of distance travelled. For the case of same individual, it was found error of 3.4 feet in 4 experiments performed and with other individuals, it was found error of 3.84 feet. In this experiment, it was found that the error does not depend on person's height, but it depends on step length of individual for every step because it is not possible for an individual to make every step with same step length. If a person usual step length is 1.7 feet then it is very likely possible that some steps are of 1.6 feet, 1.5 feet or 1.8 feet. Due to this, there is error in measured distance with the actual distance. With error of 0.1 or 0.2 feet with every step, it could accumulate to an error of 3.84 feet. This error could be problematic for case when individual walks straight without making any turn for a long distance. But if he makes turn then he need to stop in between and then make a turn and then start walking again. With this, the error is again set to 0 because the person location is known at the turn. If the person is making a turn, then we can fix the error to shift the position to the location where nearest turn is present. Since error is only of approximately 1 meters and there are usually no 2 turn under a 2 meters range so it is easy to fix the individual location with every turn he makes. Also, as observed in indoor locations like offices, hospitals there is a good chance that person will make a turn after around 50 meters at which we could have an error of 2 meters which is fixable. In this project, we have used a static approach and used the step length same for all steps so this problem is occurring. If this approach is used on a robot which makes the same step pattern on every step, there will be 0 error rate. So the case of robots, this approach work absolutely fine. But for case of humans, other approach could be used to fix this.

Other approach is to determine the step length of an individual with every step he makes. As we know that a step is determined from a peak and its subsequent valley. Currently, we had set a single threshold to determine peak and valley. So even if acceleration value goes beyond the threshold, it is just considered a step with fixed length. But we could use maximum acceleration value and minimum acceleration for every step and determine step length based on that. So when acceleration reaches a value more than usual highest acceleration value, we could increase the step length depending upon how large the value is in comparison to its usual highest acceleration value. So if highest acceleration value is larger than usual, step length could be increased for that particular step and if highest acceleration value is smaller than usual, then step length could be decreased for that particular step. This way the error could be reduced to even a smaller value that 1-meter error value. Here, the usual highest acceleration value need to be determined using calibration as every person has different step pattern and hence different highest acceleration value with usual step.

4.2 Direction Test

This is the data which compares degrees of turn made by a person which is used to determine whether the person turned right, left and turned back.

S.No	Actual Turn (degrees)	Measured Turn (degrees)	Error in Turn (degrees)	Direction
1	90	86	4	Turn Right
2	90	85	5	Turn Right
3	180	172	8	Turn Back
4	180	178	2	Turn Back
5	270	266	4	Turn Left
6	270	263	7	Turn Left
7	360	257	3	No change
8	360	253	7	No change
9	90	87	3	Turn Right
10	90	86	4	Turn Right

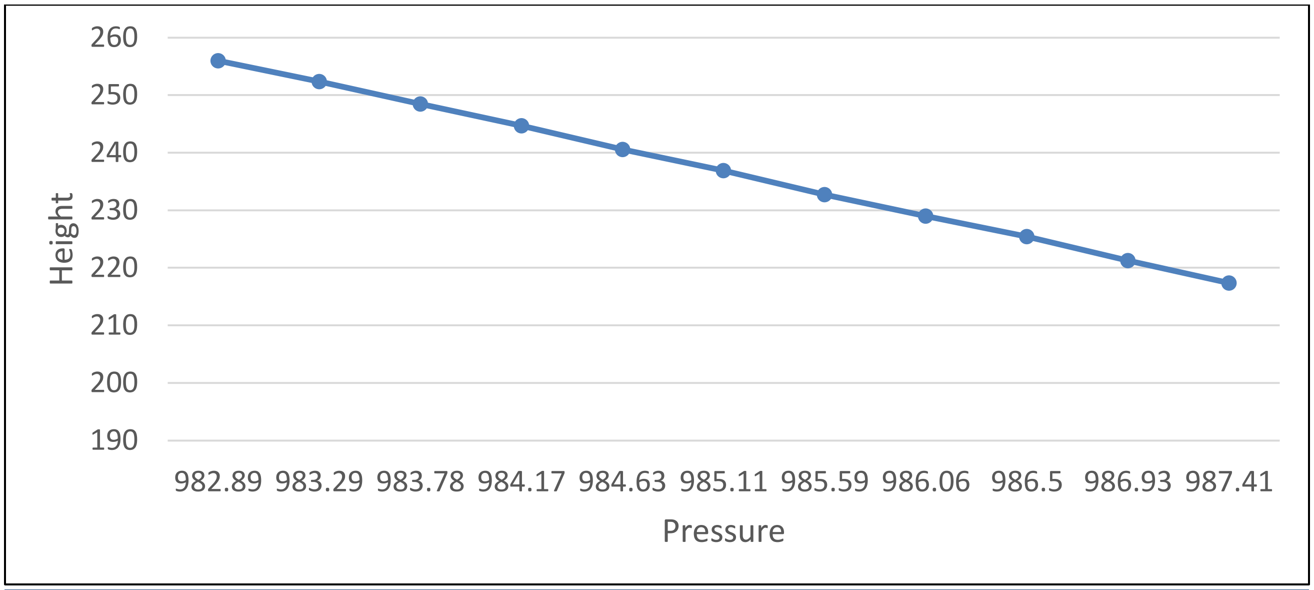


Direction is calculated based on degrees of turn a person makes. If a person makes a turn of 90 degrees, then it is a right turn. Here, there is allowed error of +/- 20 degrees because we only need to know whether person made a right turn, left turn or turned back. We do not have a requirement to know whether person turned 25 degrees because usually, there are turns of 90 degrees. Even if there are some turns in a indoor location which is of degrees other than 90 degrees, it can be determined with very good accuracy as we got the maximum error of around 10 degrees. So with turns of 90 degrees, direction is never determined wrong and is correct 100%.

4.3 Floor Test

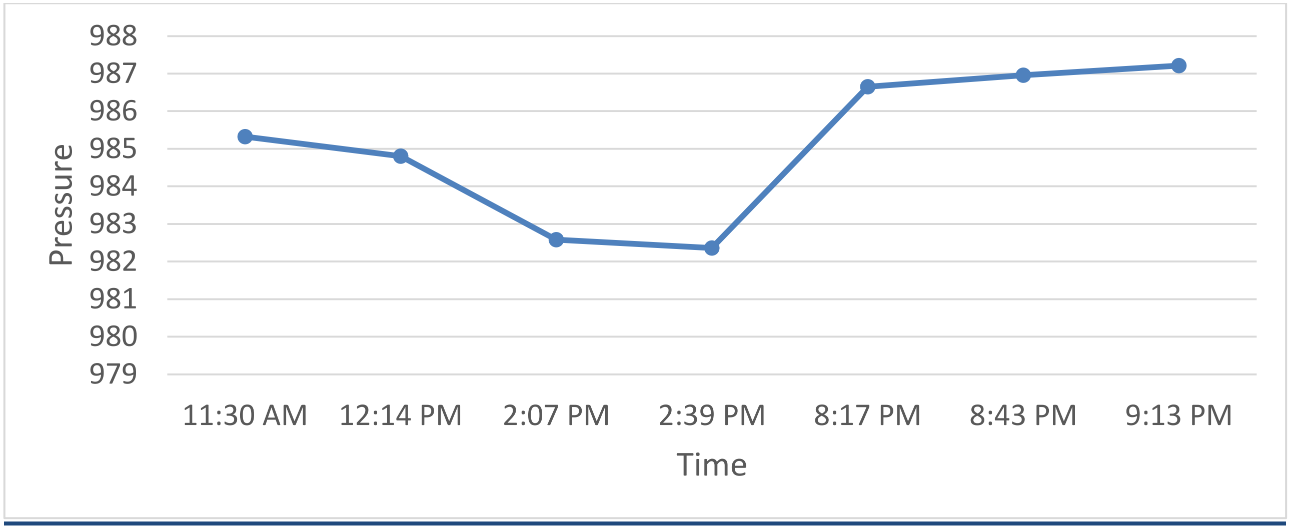
This test is performed to determine floor based on barometer sensor. Barometer sensor provides atmospheric pressure using which height above sea level is calculated. Height difference between ground to 10 floor is captured. Next, pressure and height is measured on different times of the day as pressure changes with time i.e. it is different is morning, afternoon, evening, night as it climate changes.

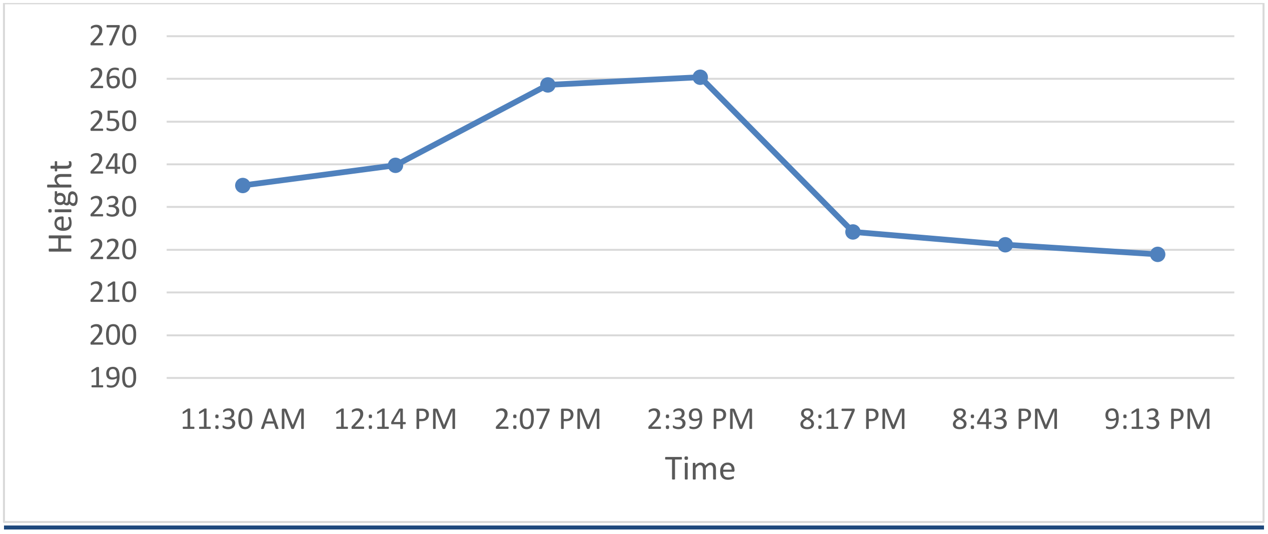
Floor	Atmospheric Pressure (hecto Pascal)	Height (meters)	Height Difference (meters)
Ground	987.41	217.35	0
First	986.93	221.23	3.88
Second	986.50	225.42	4.19
Third	986.06	228.98	3.56
Fourth	985.59	232.70	3.72
Fifth	985.11	236.89	4.19
Sixth	984.63	240.57	3.68
Seventh	984.17	244.67	4.10
Eight	983.78	248.46	3.79
Ninth	983.29	252.38	3.92
Tenth	982.89	255.97	3.59



As found from the experiment that pressure decreases with increment in height. And as pressure decreases, height increases. Simply, previous height data and new height data is subtracted to calculate the height difference between two consecutive floors. It is observed that height difference between each floor is not exactly same with a max difference of around 0.6 meters. This could be due to error in construction or error is pressure readings.

Time	Floor	Pressure	Height	Floor	Pressure	Height	Difference
11:30 AM	Tenth	985.32	235.02	Ninth	985.79	231.13	3.89
12:14 PM	Tenth	984.80	239.80	Ninth	985.28	235.85	3.95
02:07 PM	Tenth	982.58	258.61	Ninth	983.04	254.68	3.93
02:39 PM	Tenth	982.36	260.37	Ninth	982.86	256.50	3.87
08:17 PM	Tenth	986.65	224.21	Ninth	987.13	220.31	3.90
08:43 PM	Tenth	986.96	221.16	Ninth	987.43	217.28	3.88
09:13 PM	Tenth	987.21	218.96	Ninth	987.69	215.05	3.91
11:01 PM	Tenth	987.25	218.79	Ninth	987.73	214.84	3.95
11:36 PM	Tenth	986.93	221.45	Ninth	987.37	217.56	3.89
11:40 PM	Tenth	987.02	220.47	Ninth	987.50	216.57	3.90





As can be seen from above data, pressure does not remain constant with time at a fixed location. It could increase and it could decrease depending on climate. As pressure is not constant at a fixed so height also changes so at first glance it seems that we cannot used this approach to determine floor. But actually, we are measuring change in floor and not absolute floor. So we are only interested in change of pressure value or change in height value upon going from one floor to another floor. As can be seen from above data, although pressure and height changes with time at a fixed location but the difference between them remain fixed. There is minor difference between the height difference but that is acceptable because the difference is of 0.5 meters but difference between heights of two floors is around 3 meters. In our case it is 3.5 meters so this difference is completely acceptable. Since height difference is fixed, so we are able to detect change in floor with 100% accuracy.

Chapter 5. IMPLEMENTATION & RESULTS

5.1 Map Creation

In this project, we have used map reference of Samsung R&D Institute Building located in Sector 62, Noida having floors from ground to 10. Floors 1 to 10 are employee working floors having same design of cubicles and conference rooms. Ground floor has a cafeteria so it has a different design. So, a floor map of 10th floor and floor map of cafeteria is created for this project.

To create the map, floor area is divided into squares of 1.9 feet by 1.9 feet. 1.9 feet is usually a length of a single step of person having height inches and this project uses the same step length. Upon doing the measurement, it is found that floor map is 581.4 * 381.9 feet which is same as 306 * 201 steps i.e. 306 steps in vertical (length) and 201 steps in horizontal (breadth). This measurement is including the wall area. So total area comes out to be 222036.66 square feet or 61506 steps.

Next, to distinguish between an obstacle such a wall, a gate, a cubicle, a table and pathway, the map is filled with 0's and 1's, where 0 denotes a pathway and 1 denotes an obstacle. And each square of 1 step by 1 step is further divided into 9 small squares where each square denotes 20 pixels by 20 pixels on mobile display screen. The squares are further divided because on obstacle could be smaller than 1 single step such as a gate or a fence of a cubicle. Therefore, our small unit comes out to be 20 pixels or 1/3 of a step or 0.63 feet and overall pixels used by this floor map are 20 * 3 * 201 by 20 * 3 * 306 pixels or 12060 by 18360 pixels. However, mobile screen is not that big. In our project we used S7 edge, which has 720 by 1080 pixels. Therefore, at once we could only see some portion of the map and the rest of the map we could see simply by scrolling on the screen both vertically and horizontally covering the whole floor map.

Upon more progress, it turned out that 1's and 0's could not solve our purpose as we need to mark every position in the map to some location so that when the person takes a next step we could not know its new live location. So instead of 0's, we used IDs to fill the map pathways. Along with this, we also changed 1's to ID's of obstacles. Hence live location update is made easy. Another benefit of using ID's over 0's and 1's is that it helps in applying colors to the map so we can apply the color based on ID. Overall, some ID's just denote specific area, some ID's denotes destination/source and some ID's denotes obstacles

Here is the list of Source and Destination ID List used in the map

ID	Description		ID	Description
LF1	Lift 1		LOB	Lobby
LF2	Lift 2		ENT	Entrance
LF3	Lift 3		EXT	Exit
LF4	Lift 4		CFE	Cafeteria Entry
MPE	Maple Room Entry		WRBE	Washroom Boys Entry
TPE	Tulip Room Entry		WRGE	Washroom Girls Entry
IRE	Iris Room Entry		1	Akshay Goyal
LLE	Lily Room Entry		7	Prateek Sharma
RSE	Rose Room Entry		21	Dhruv Goyal
GM1	Sachin Gupta		22	Akash Agarwal
GM2	Mohan Jangir		54	Abhay Sharma
GM3	Pradeep Bohra		55	Prachi Gupta
GM4	Nirja Sinha		242	Aniroop Mathur
GM5	Ashish Kalra		243	Kuldeep Kumar
GM6	Sachin Papneja		247	Kapil Gautam
HOD2	Changhwan Seo		260	Pankaj Sharma
HLE	Hardware Lab Entry		262	Ravindra Jain

These are ID's which will be selected when the person enters the source and destination name in the application. Each of these ID's represent a row index and column index number in the tabular map. And each of row and column number can be converted to represent the location on the mobile screen at appropriate pixel by pixel location. For example, row index 120 and column index 200 represent 120 * 3 * 20 by 200 * 3 * 20 i.e. 7200 by 12000 pixels' location on the mobile display screen.

Next is the list of ID's which represent the usual walking area. So when the person is in the lobby, index in the tabular data denotes LO and our application will come to know that the person is in the lobby currently. Similarly, as the person walks and if ID changes to ENT, then we know the person is at the entrance of the floor. Similarly, if index value changes to SW after taking a step, it means the person is on the side walkway. This way, we know all locations of the person as he moves.

After this list, list of obstacle ID's are mentioned. If a person makes a move and the ID is one of the obstacle listed ID, then that movement will not be a legal movement and the person position does not change in the map. So we never have an illegal position in the map.

Here is the list of usual walking area ID List used in the map

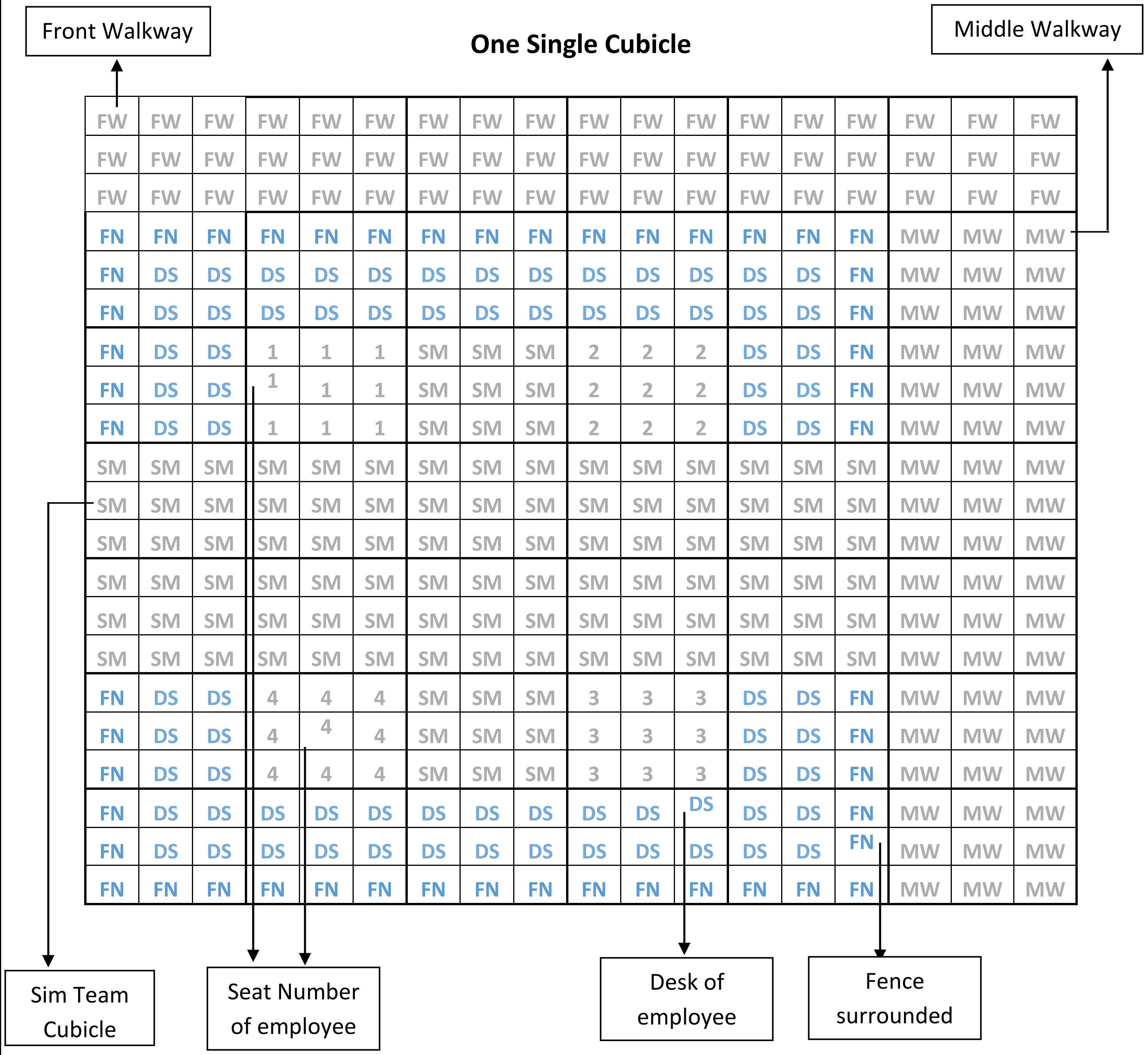
ID	Description		ID	Description
LFA	Lift Area		MP	Maple Conference Room
LFT	Lift		TP	Tulip Conference Room
LO	Lobby		IR	Iris Conference Room
FW	Front Walkway		LL	Lily Conference Room
SW	Side Walkway		RS	Rose Conference Room
MW	Middle Walkway		CF	Cafeteria
BW	Back Walkway		WR	Washroom
CW	Cafeteria Walkway		WRA	Washroom Area
HW	Hidden Walkway		HL	Hardware Lab
BLB	Back Lobby		SM	SIM Team Cubicle
SMPL	SIM Part Leader Cubicle		RL	RIL Team Cubicle
RLPL	RIL Part Leader Cubicle		SGL	System Group Leader Cubicle
CP	CP Team Cubicle		CPPL	CP Part Leader Cubicle
BSP	BSP Team Cubicle		S1PL	System1 Part Leader Cubicle
LCD	LCD Team Cubicle		EXST	Expat System Team Cubicle
TSP	TSP Team Cubicle		FS	File System Team Cubicle
USB	USB Team Cubicle		HD	Hardware Team Cubicle
BL	BL Team Cubicle		FAC	Factory Team Cubicle
SNS	Sensor Team Cubicle		MM	Memory Team Cubicle
PF	Performance Team Cubicle		CC	Current Consumption Cubicle
BTR	Battery Team Cubicle		S2PL	System2 Part Leader Cubicle

Here is the list of obstacle ID List used in the map

ID	Description		ID	Description
WL	Wall		DS	Desk
FN	Fence		TBL	Table

Using all of the above ID's, whole floor map is created in the excel sheet. The main problem is that this needs to created manually so one had to manually measure the length and breadth of a cubicle, a lift, a lobby, a conference, a path way and fill the IDs accordingly. If some location is lobby, then need to fill LO ID 9 times in a single square box of 1 step as smallest unit is of 1/3 of a step which is required to mark obstacles like Fence which are of very short breadth. If nearby area is also lobby, then need to mark LO in nearby squares as well. Same logic goes for side walkways, front walkways, desk, fence, team cubicle area, etc

As the map is very large, let’s take a look of some major parts drawn in the map.


Front Walkway
One Single Cubicle
Middle Walkway
Sim Team Cubicle
Seat Number of employee
Desk of employee
Fence surrounded

FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW
FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW
FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW
FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	MW	MW	MW
FN	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	FN	MW	MW	MW
FN	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	FN	MW	MW	MW
FN	DS	DS	1	1	1	SM	SM	SM	2	2	2	DS	DS	FN	MW	MW	MW
FN	DS	DS	1	1	1	SM	SM	SM	2	2	2	DS	DS	FN	MW	MW	MW
FN	DS	DS	1	1	1	SM	SM	SM	2	2	2	DS	DS	FN	MW	MW	MW
SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	MW	MW	MW
SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	MW	MW	MW
SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	MW	MW	MW
SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	MW	MW	MW
SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	MW	MW	MW
SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	SM	MW	MW	MW
FN	DS	DS	4	4	4	SM	SM	SM	3	3	3	DS	DS	FN	MW	MW	MW
FN	DS	DS	4	4	4	SM	SM	SM	3	3	3	DS	DS	FN	MW	MW	MW
FN	DS	DS	4	4	4	SM	SM	SM	3	3	3	DS	DS	FN	MW	MW	MW
FN	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	FN	MW	MW	MW
FN	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	DS	FN	MW	MW	MW
FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	MW	MW	MW

This is just one cubicle and similar to this logic all the other cubicles are also drawn with different ID’s as required. In this cubicle, 1-2-3-4 are seat numbers and are destination ID’s as well. SM indicates that it is a cubicle of SIM team. DS is desk of the cubicle and FN is fence of the cubicle. Person is not allowed to walk over FN and DS. FW is front walk way as it occurs right after entering the floor and MS is middle walkway as it is in between the floor.

Expat System Team Cubicle

Seat of HOD

HOD Cubicle

FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN
FN	FN	FN	EXST	EXST	EXST	DS	DS	DS	DS	DS	DS	DS	DS	FN
FN	FN	FN	EXST	EXST	EXST	DS	DS	DS	DS	DS	DS	DS	DS	FN
FN	FN	FN	EXST	EXST	EXST	DS	DS	DS	HOD2	HOD2	HOD2	DS	DS	FN
FN	FN	FN	EXST	EXST	EXST	DS	DS	DS	HOD2	HOD2	HOD2	DS	DS	FN
FN	FN	FN	EXST	EXST	EXST	DS	DS	DS	HOD2	HOD2	HOD2	DS	DS	FN
EXST	EXST	EXST	EXST	EXST	EXST	DS	DS	DS	EXST	EXST	EXST	DS	DS	FN
EXST	EXST	EXST	EXST	EXST	EXST	DS	DS	DS	EXST	EXST	EXST	DS	DS	FN
EXST	EXST	EXST	EXST	EXST	EXST	DS	DS	DS	EXST	EXST	EXST	DS	DS	FN
EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
FN	FN	FN	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
FN	FN	FN	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
FN	FN	FN	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
FN	FN	FN	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
FN	FN	FN	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	EXST	FN
FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN

LIFT

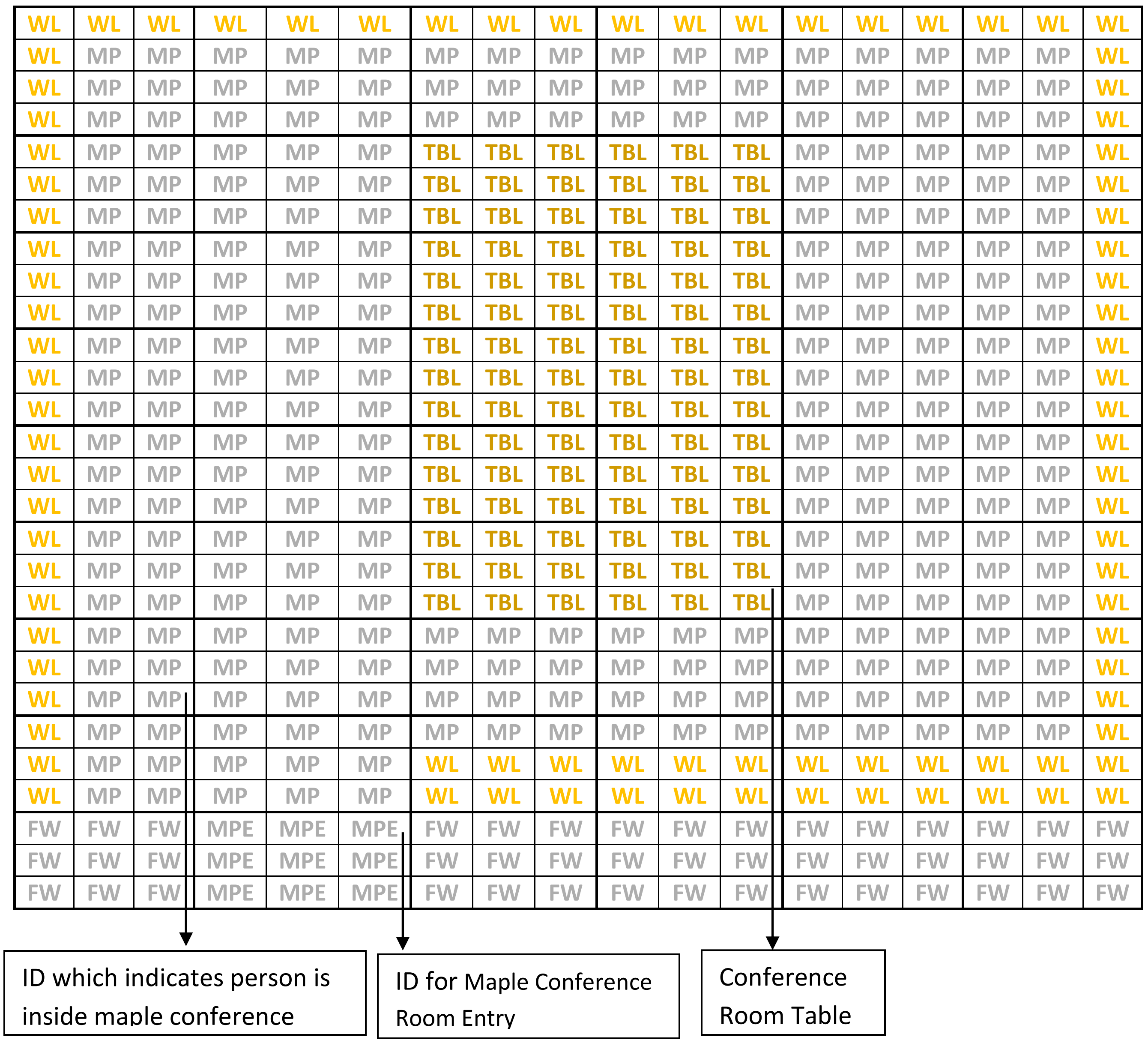
WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL
WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL
WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	LFT	WL	WL	WL	WL
WL	WL	WL	WL	WL	WL	LFA	LFA	LFA	LF3	LF3	LF3	WL	WL	WL	WL	WL	WL
WL	WL	WL	WL	WL	WL	LFA	LFA	LFA	LF3	LF3	LF3	WL	WL	WL	WL	WL	WL
WL	WL	WL	WL	WL	WL	LFA	LFA	LFA	LF3	LF3	LF3	WL	WL	WL	WL	WL	WL
LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA
LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA
LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA	LFA

Lift Area

ID which indicates person is inside the lift

ID of Lift 3

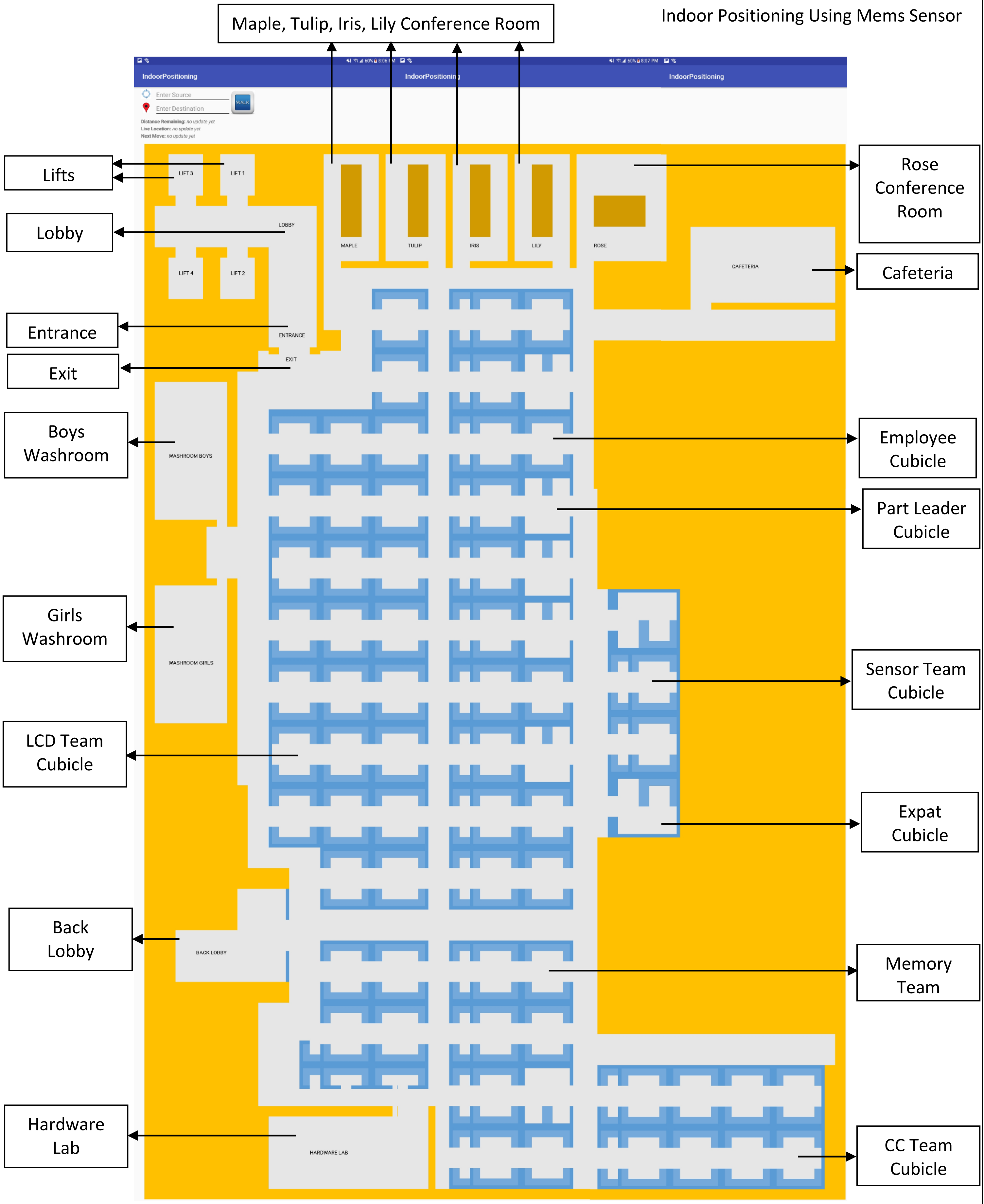
Maple Conference Room



WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	TBL	TBL	TBL	TBL	TBL	TBL	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	MP	WL
WL	MP	MP	MP	MP	MP	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL
WL	MP	MP	MP	MP	MP	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL	WL
FW	FW	FW	MPE	MPE	MPE	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW
FW	FW	FW	MPE	MPE	MPE	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW
FW	FW	FW	MPE	MPE	MPE	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW	FW

ID which indicates person is inside maple conference
ID for Maple Conference Room Entry
Conference Room Table

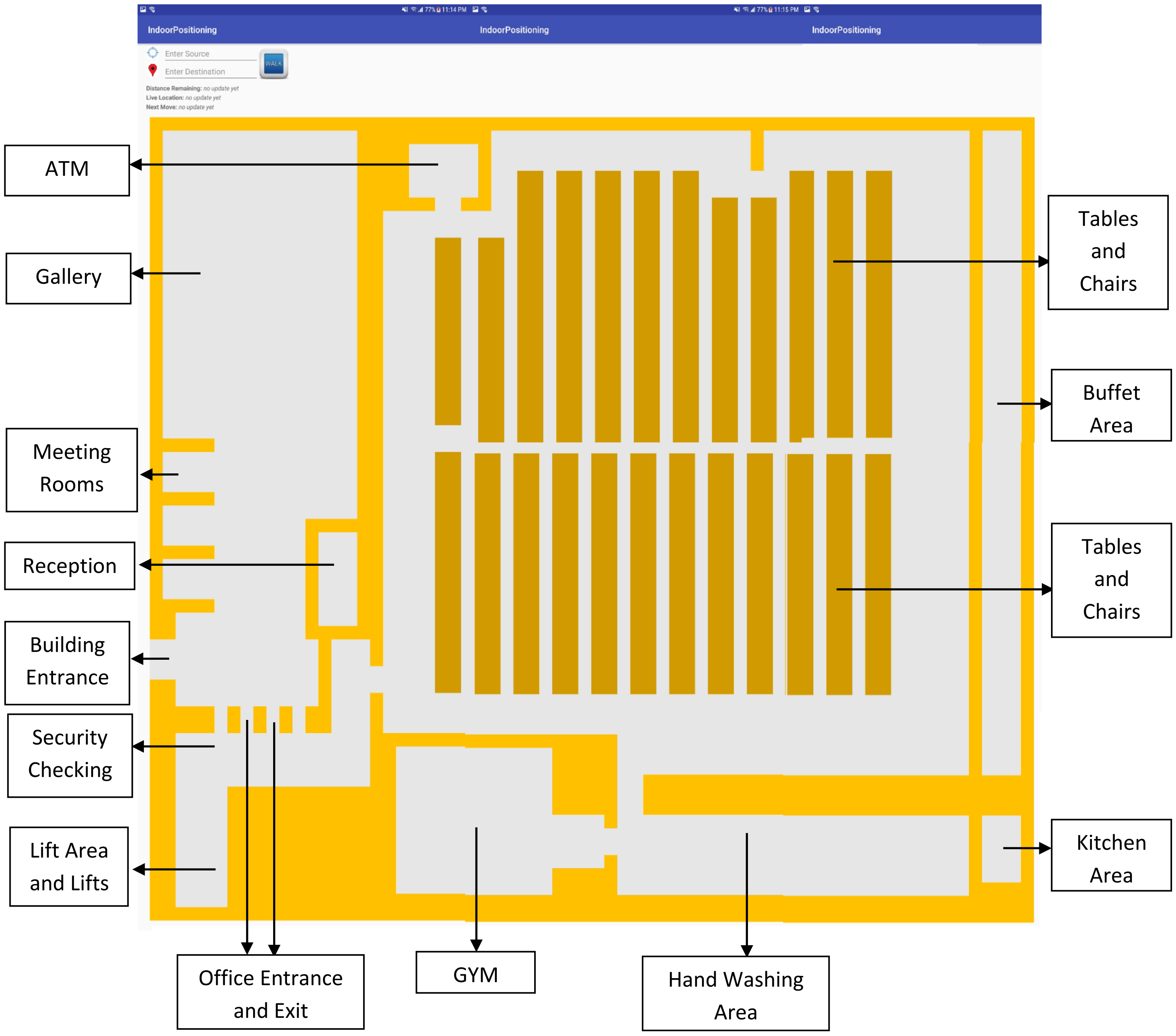
So, this is how different parts of the floor are created in the excel sheet and combining these different parts will create a complete map. But this map is created on excel sheet and we need to draw the map on the mobile screen. For this, the excel sheet data is put into text file in a comma separated format. Then at the time of opening the application, the software reads the data from this text file line by line. Then the software draws the rectangle of size 20 pixel by 20 pixel of the color based on the ID. Wall is of yellow color, Fence is of dark blue color, Desk is of light blue color, Table is of brown color, and all walking areas are of light grey color. Using this criteria, when the software finishes the drawing, it looks as below:


Maple, Tulip, Iris, Lily Conference Room
IndoorPositioning
Enter Source
Enter Destination
WALK
Distance Remaining: no update yet
Live Location: no update yet
Next Move: no update yet
Lifts
LIFT 3
LIFT 1
Lobby
LOBBY
LIFT 4
LIFT 2
MAPLE
TULIP
IRIS
LILY
ROSE
Rose Conference Room
CAFETERIA
Cafeteria
Entrance
ENTRANCE
Exit
EXIT
Boys Washroom
WASHROOM BOYS
Employee Cubicle
Part Leader Cubicle
Girls Washroom
WASHROOM GIRLS
Sensor Team Cubicle
LCD Team Cubicle
Expat Cubicle
Back Lobby
BACK LOBBY
Memory Team
Hardware Lab
HARDWARE LAB
CC Team Cubicle

FLOOR MAP

Similar to floor map of 10th floor, map of floors from 1 to 9 are also kept same. However, ground floor has cafeteria so the map is also drawn with same logic used to draw floor 10th. Cafeteria map is kept simple and is not used for navigation. Floor 10th is only used for navigation purpose. But we need to test change in floor if our software is able to detect change in floors well and able to change map according to floor. So when the person goes from floor 10th to ground floor map view changes to ground floor map and rest of the floors it remains same. Here is the look of ground floor drawn which is kept simple.

GROUND FLOOR MAP



Here is the brief source code implementation to draw the above two maps from big table containing IDs of all the map area.

```
mPaintRectWall.setStyle(Paint.Style.FILL);
mPaintRectWall.setColor(Color.rgb(255,192,0));
mPaintRectWall.setAntiAlias(true);
mPaintRectWall.setDither(true);
```

Set paint object for wall

```
mPaintRectFence.setColor(Color.rgb(91,155,213));
mPaintRectDesk.setColor(Color.rgb(115,169,219));
mPaintRectTable.setColor(Color.rgb(208,154,0));
```

Set paint object for Fence, Desk, Table

```
static final int startX = 60;
static final int startY = 0;
static final int unitSize = 20;
```

Start drawing map from (60, 0) pixel coordinate

```
public void onDraw(Canvas canvas) {
    int x1 = startX;;
    int y1 = startY;
    int x2 = x1 + unitSize;
    int y2 = y1 + unitSize;
```

Start and End point of a cell Length and breadth are of 20 pixels

```
    if (mFloor != 0) {
```

Draw map of floors from 1 to 10

```
        for (int r = 0; r < IndexDestinationMap.mRows; r++) {
            for (int c = 0; c < IndexDestinationMap.mCols; c++) {
                if (mObstacleList.contains(mIndDestIDMap[r][c])) {
                    if (mIndDestIDMap[r][c].equals("WL"))
                        canvas.drawRect(x1, y1, x2, y2, mPaintRectWall);
                    else if (mIndDestIDMap[r][c].equals("FN"))
                        canvas.drawRect(x1, y1, x2, y2, mPaintRectFence);
                    else if (mIndDestIDMap[r][c].equals("DS"))
                        canvas.drawRect(x1, y1, x2, y2, mPaintRectDesk);
                    else if (mIndDestIDMap[r][c].equals("TBL"))
                        canvas.drawRect(x1, y1, x2, y2, mPaintRectTable);
                } else {
                    canvas.drawRect(x1, y1, x2, y2, mPaintRectPath);
                }
```

Choose paint color object to draw the cell of the map based on cell ID

```
                x1 = x1 + unitSize;
                x2 = x2 + unitSize;
            }
```

Increment 20 pixels column wise

```
            x1 = startX;
            y1 = y1 + unitSize;
            x2 = x1 + unitSize;
            y2 = y1 + unitSize;
        }
    }
```

Increment 20 row wise and move to start of row

Draw map of ground floor

Choose paint color object to draw the cell of the map based on cell ID

```
} else if (mFloor == 0) {
  for (int r = 0; r < FloorDetector.mRows; r++) {
    for (int c = 0; c < FloorDetector.mCols; c++) {
      if (groundFloorMap[r][c].contains("1"))
        canvas.drawRect(x1, y1, x2, y2, mPaintRectWall);
      else if (groundFloorMap[r][c].contains("2"))
        canvas.drawRect(x1, y1, x2, y2, mPaintRectTable);
      else if (groundFloorMap[r][c].contains("0"))
        canvas.drawRect(x1, y1, x2, y2, mPaintRectPath);

      x1 = x1 + unitSize;
      x2 = x2 + unitSize;
    }
```

Increment 20 pixels column

```
    x1 = startX;
    y1 = y1 + unitSize;
    x2 = x1 + unitSize;
    y2 = y1 + unitSize;
  }
}
```

Increment 20 row wise and move to start of row

```
x1 = startX;;
y1 = startY;
x2 = x1 + unitSize;
y2 = y1 + unitSize;
```

Set back to initial position (60, 0) pixels

Write name of locations in the map of floors 1 to 10

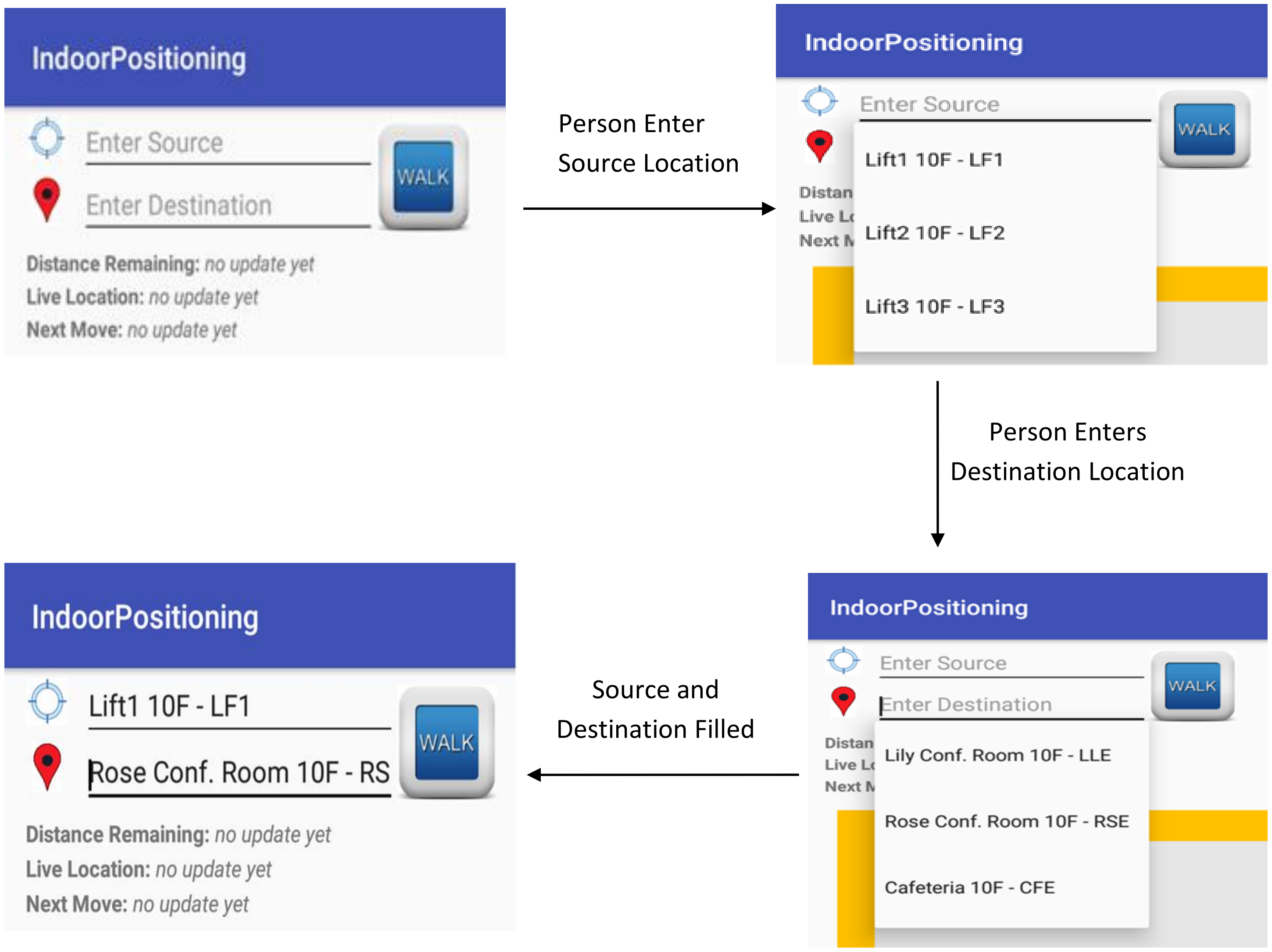
If ID contains a "-" character, get the location name after that character else skip it

```
if (mFloor != 0) {
  for (int r = 0; r < IndexDestinationMap.mRows; r++) {
    for (int c = 0; c < IndexDestinationMap.mCols; c++) {
      if (!mObstacleList.contains(mIndDestIDMap[r][c])) {
        if (mIndDestIDMap[r][c].contains("-")) {
          String text[] = mIndDestIDMap[r][c].split("-");
          String name = "";
          for (int i = 0; i < text.length - 1; i++)
            name += text[i] + " ";

          canvas.drawText(name, x1, y1, mPaintText);
        }
      }

      x1 = x1 + unitSize;
      x2 = x2 + unitSize;
    }

    x1 = startX;
    y1 = y1 + unitSize;
    x2 = x1 + unitSize;
    y2 = y1 + unitSize;
  }
}
```

5.2 Source & Destination List Creation

To start indoor navigation, person need to enter source location and destination location. This location list is shown to person in form of auto complete drop down. Either person can directly type and use suggestions or directly choose from drop down option. Every such location is given a unique ID which is used by software to determine its position on the map. Each of these IDs are mapped to map index and map index is mapped to these IDs already. Some of the location list options are shown below and user interface for this is also shown below it.

Source and Destination List			
Name	**Floor**	**ID**	**Team**
Lift1	10F	LF1	System R&D
Lift2	10F	LF2	System R&D
Lobby	10F	LOB	System R&D
Entrance	10F	ENT	System R&D
Exit	10F	EXT	System R&D
Maple Conf. Room	10F	MPE	System R&D
Tulip Conf. Room	10F	TPE	System R&D
Rose Conf. Room	10F	RSE	System R&D
Cafeteria	10F	CFE	System R&D
Washroom Boys	10F	WRBE	System R&D
Washroom Girls	10F	WRGE	System R&D
Sachin Gupta	10F	GM1	SIM Team
Pradeep Bohra	10F	GM3	System R&D
Ashish Kalra	10F	GM5	System1 Team
Changhwan Seo	10F	HOD2	System R&D
Akshay Goyal	10F	1	SIM Team
Aniroop Mathur	10F	242	Sensor Team
Kapil Gautam	10F	247	Sensor Team
Ankit Sharma	10F	203	Memory Team
Pankaj Sharma	10F	260	Current Consumption Team
Abhay Sharma	10F	54	RIL Team
Akash Agarwal	10F	22	RIL Team
Gunjan Gupta	10F	258	Sensor Team
Kuldeep Kumar	10F	243	Memory Team
Neetesh Singh	10F	211	Memory Team
Hardware Lab	10F	HLE	System R&D



5.3 Graph Creation and Mapping

Using the data of map, a graph is created. This graph includes only the pathways and not the obstacles. This graph is required to track movement of the person. With this, it will not be possible to have the person moved inside some obstacle because the graph contains only the valid pathways. This graph will also be used to create a shortest path from source to destination as mentioned in next section. To create the graph, whole map is traversed row by row and if there is no obstacle for 3 consecutive small unit or for 1 step, then that next node is chosen to be part of the graph. For example, if there LO, LO, LO consecutively, then end index is chosen to be a node in the graph. This checking is done for consecutive indexes increasing in both forward and down direction. Checking in this way for each row covers all the indexes of the map.

Below is the quick source code of creating the graph which also creates mapping of map index to graph vertex and graph vertex to map index

```
public void createGraph() {

    g = new Graph(maxVertices);

    for (int i = 0; i < maxRows; i++)
        for (int j = 0; j < maxCols; j++)
            indexVertexMap[i][j] = -1;

    for (int i = 0; i < maxVertices; i++)
        vertexIndexMap[i] = null;

    for (int i = 0; i < IndexDestinationMap.mRows; i = i + 3) {

       for (int j = 0; j < IndexDestinationMap.mCols; j = j + 3) {

            int r = i / 3;
            int c = j / 3;

            if (j + 3 < IndexDestinationMap.mCols &&
```

Check if 3 consecutive map indexes are valid

```
                    !mObstacleList.contains(mIndDestIDMap[i][j]) &&
                    !mObstacleList.contains(mIndDestIDMap[i][j+1]) &&
                    !mObstacleList.contains(mIndDestIDMap[i][j+2]) &&
                    !mObstacleList.contains(mIndDestIDMap[i][j+3])) {
```

Choose start vertex Map to map index

```
                     if (indexVertexMap[r][c] == -1) {
                            vertexIndexMap[vertexCount] = new Index(r, c);
                            indexVertexMap[r][c] = vertexCount++;
                     }
```

Choose end vertex Map to map index

```
                     if (indexVertexMap[r][c + 1] == -1) {
                            vertexIndexMap[vertexCount] = new Index(r, c + 1);
                            indexVertexMap[r][c + 1] = vertexCount++;
                     }
```

Add Edge back and forth

```
                  g.addEdge(indexVertexMap[r][c], indexVertexMap[r][c + 1], 1);
                  g.addEdge(indexVertexMap[r][c + 1], indexVertexMap[r][c], 1);
              }
         }
    }
}

public void addEdge(int src, int dest, int weight){
    Vertex s = vertices.get(src);
    Edge new_edge = new Edge(vertices.get(dest),weight);
    s.neighbours.add(new_edge);
    edgeCount++;
}
```

Add edge to the vertex

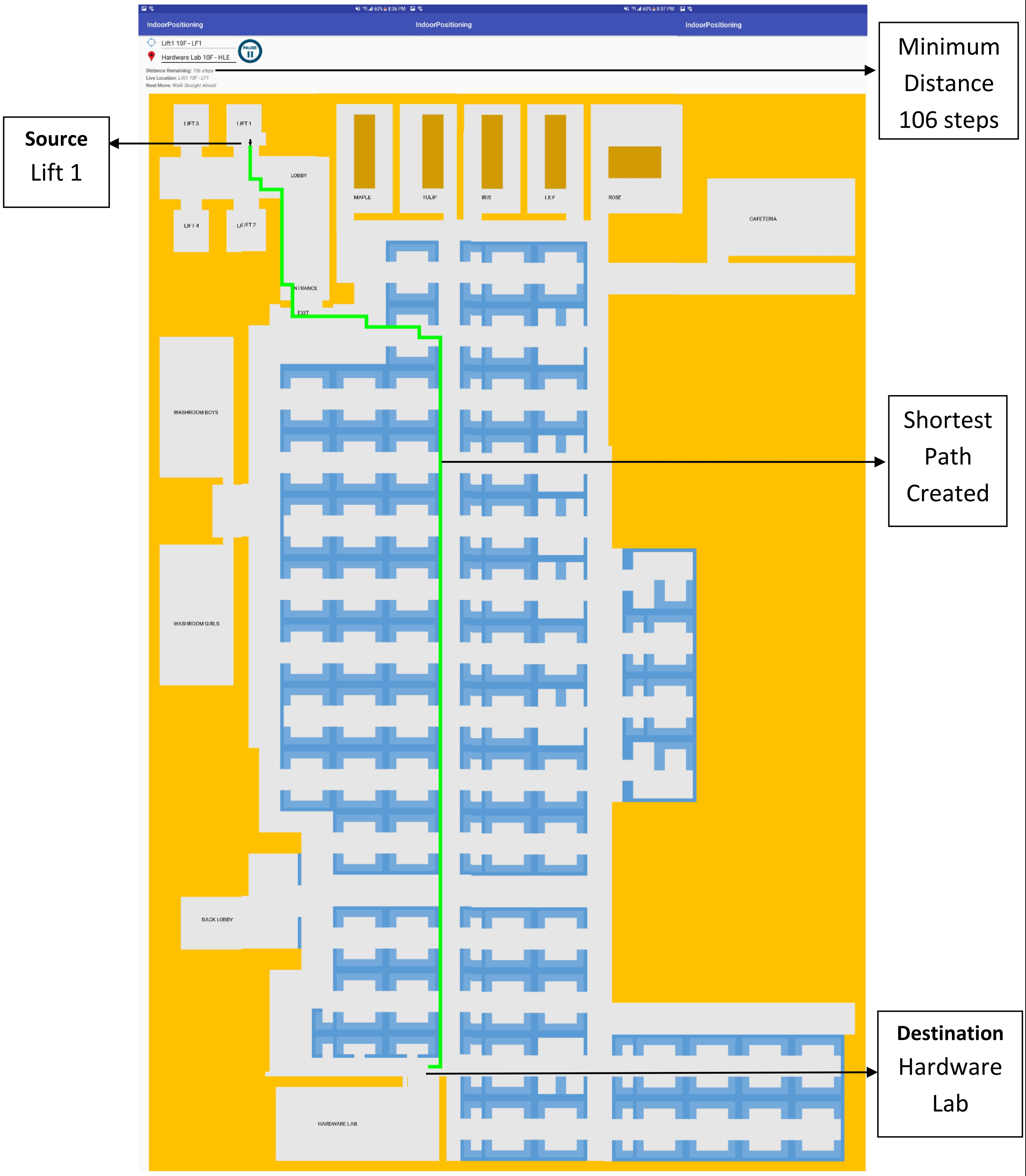
5.4 Shortest Path Creation

In our project, Dijkstra Algorithm is used for calculating shortest path. After the person enters source and destination and press walk button then shortest path need to be created and shown to the person which he can follow to reach to his destination. Firstly, from the source text and destination text, ID is extracted. This ID is mapped to the index of the map. This map index is then converted to graph vertex just by multiplying both row and column value by 3. Using this new index, source vertex of the graph is found. Using this source vertex, Dijkstra Algorithm is applied on the graph created. When the algorithm finishes, shortest path to the destination vertex is found. This path consists of vertexes and to draw it on the map, these vertex list is converted to index of the map using the mapping done while creating the graph. In the shortest path, we can see zig zag in between because for this algorithm, there is no difference between forward vertex and downward vertex as both constitute 1 step so algorithm can chose any one of them whichever comes in queue first. Minimum distance required is simply the size of node path list as distance from one node to another is 1 step so size of list is total minimum distance or steps required to reach to destination. Below is the source code used and snapshot of the shortest path that is created.

Dijkstra Algorithm

```
public void calculateMinimumDistance(Vertex source) {

    source.minDistance = 0;
    PriorityQueue<Vertex> queue = new PriorityQueue<Vertex>();

    queue.add(source);

    while(!queue.isEmpty()){

        Vertex u = queue.poll();  ──────►  Pick unvisited node with minimum weight

        for(Edge neighbour:u.neighbours){  ──────►  Visit all neighbors
            Double newDist = u.minDistance + neighbour.weight;

            if(neighbour.target.minDistance > newDist){

                queue.remove(neighbour.target);
                neighbour.target.minDistance = newDist;  ──►  Update the distance if it is less

                neighbour.target.path = new LinkedList<Vertex>(u.path);
                neighbour.target.path.add(u);  ──────►  Add node to the shortest path list
                queue.add(neighbour.target);
            }
        }
    }
}
```

Shortest path from Lift 1 to Hardware Lab marked in green color



Lift1 10F - LF1

Hardware Lab 10F - HLE

Distance Remaining: *106 steps*
Live Location: *Lift1 10F - LF1*
Next Move: *Walk Straight Ahead*

5.5 Location Tracking

In this project, we are using a filled arrow to denote the current location of a person. If the person takes a step, then the arrow is moved a step ahead. If the person takes a turn, then the arrow is turned accordingly as well. If a step is detected then it is first checked whether it is possible to move the tracker or not because it is possible there is an obstacle after 1 step so that is not considered a valid move and the tracker remain at the same position. Upon successful move, we check for next move as well to guide the person for his next movement. Here is the quick source code of implementing location tracking.

```
int lengthRect = 15; int widthRect = 10;
int baseTriangle = 20;
```

Dimensions of the tracker created using a rectangle and a triangle

```
public void drawRightArrow(Canvas canvas) {
    int x = PosX;
    int y = PosY;
```

New position of the person

```
    x = x - lengthRect - baseTriangle;
```

Starting position for the triangle

```
    mPaintPath.reset();
    mPaintPath.moveTo(x + lengthRect, y);
    mPaintPath.lineTo(x + baseTriangle*2, y + baseTriangle/2);
    mPaintPath.lineTo(x + lengthRect, y + baseTriangle);
    mPaintPath.close();
    canvas.drawPath(mPaintPath, mPaintPointer);
```

Drawing triangle part of the tracker

```
    y = y + widthRect/2;
    canvas.drawRect(x, y, x + lengthRect, y + widthRect, mPaintPointer);
}
```

Drawing rectangle part of the tracker

```
public void drawDownArrow(Canvas canvas) {
    int x = PosX;
    int y = PosY;

    y = y - lengthRect - baseTriangle;
    mPaintPath.reset();
    mPaintPath.moveTo(x, y + lengthRect);
    mPaintPath.lineTo(x + baseTriangle/2, y + baseTriangle*2);
    mPaintPath.lineTo(x + baseTriangle , y + lengthRect);
    mPaintPath.close();
    canvas.drawPath(mPaintPath, mPaintPointer);

    x = x + widthRect/2;
    canvas.drawRect(x, y, x + widthRect, y + lengthRect, mPaintPointer);
}
```

Similar to right pointing tracker, down pointing arrow is drawn but with different parameters

```
public void changePosition(int steps) {
      boolean posChanged = false;

     if (mArrow == rightArrow) {
         if (mapIndY + 3 < IndexDestinationMap.mCols &&
             !mObstacleList.contains(mIndDestIDMap[mapIndX][mapIndY + 1])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX][mapIndY + 2])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX][mapIndY + 3])) {
             PosX = PosX + singleStep * steps;
             mapIndY = mapIndY + stepUnit;
             posChanged = true;
         }

     } else if (mArrow == downArrow) {
         if (mapIndX + 3 < IndexDestinationMap.mRows &&
             !mObstacleList.contains(mIndDestIDMap[mapIndX + 1][mapIndY])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX + 2][mapIndY])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX + 3][mapIndY])) {
             PosY = PosY + singleStep * steps;
             mapIndX = mapIndX + stepUnit;
             posChanged = true;
         }
     } else if (mArrow == leftArrow) {
         if (mapIndY - 3 >= 0 &&
             !mObstacleList.contains(mIndDestIDMap[mapIndX][mapIndY - 1])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX][mapIndY - 2])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX][mapIndY - 3])) {
             PosX = PosX - singleStep * steps;
             mapIndY = mapIndY - stepUnit;
             posChanged = true;
         }
    } else if (mArrow == upArrow) {
         if (mapIndX - 3 >= 0 &&
             !mObstacleList.contains(mIndDestIDMap[mapIndX - 1][mapIndY])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX - 2][mapIndY])
             && !mObstacleList.contains(mIndDestIDMap[mapIndX - 3][mapIndY])) {
                 PosY = PosY - singleStep * steps;
                 mapIndX = mapIndX - stepUnit;
                 posChanged = true;
         }
     }
}


public void resetPath() {
     for(Vertex v : g.getVertices()) {
         v.path = new LinkedList<>();
         v.minDistance = Double.POSITIVE_INFINITY;
     }
}
```

Tracker is moved 1 step right only if 3 consecutive cells of the map do not contain IDs of obstacles

Tracker is moved 1 step down only if 3 consecutive cells of the map do not contain IDs of obstacles

For left direction movement, x is decreased by 60 pixels and y remains same

For up direction movement, y is decreased by 60 pixels and x remains same

After changing the position, the shortest path is reset because shortest path need to be determined again with new source index of the map

5.6 Determining Next Move

Since we have created the whole map in tabular format so determining the next move is not tough as it looks. First, we need to determine the source row and source column index from shortest path list and then next row and next column index. In our project, the person can move only in one of the axis at a time, so if next row is same as source row, it means there is movement in y-axis direction and if next column is greater than source column, it means the person is moving in downward direction. Similar logic for other cases. Next, to determine the next move, if new and previous tracker are separated by -90 degrees, it means the person need to make a left turn. Let's look at brief source code implementation of it.

```java
public void checkNextMove() {

Index srcIndex = mPathView.getPathObject().vertexIndexMap[srcVertex];
int srcRow = srcIndex.getRow() * 3;
int srcCol = srcIndex.getCol() * 3;

Index nextIndex = mPathView.getPathObject().vertexIndexMap[nextVertex];
int nextRow = nextIndex.getRow() * 3;
int nextCol = nextIndex.getCol() * 3;


if (nextCol == srcCol) {
    if (nextRow > srcRow)
        nextMove = downArrow;
    else
        nextMove = upArrow;
} else if (nextRow == srcRow) {
    if (nextCol > srcCol)
        nextMove = rightArrow;
    else
        nextMove = leftArrow;
}

if ((mArrow == 0 && nextMove == 1) || (mArrow == 1 && nextMove == 2)
    || (mArrow == 2 && nextMove == 3) || (mArrow == 3 && nextMove == 0)) {
    nextMoveDescription = "Turn Right";
} else if ((mArrow == 0 && nextMove == 3) || (mArrow == 3 && nextMove == 2)
        || (mArrow == 2 && nextMove == 1) || (mArrow == 1 && nextMove == 0)) {
    nextMoveDescription = "Turn Left";
} else if ((mArrow == 0 && nextMove == 2) || (mArrow == 2 && nextMove == 0)
        || (mArrow == 3 && nextMove == 1) || (mArrow == 1 && nextMove == 3)) {
    nextMoveDescription = "Turn Back";
} else if (mArrow == nextMove) {
    nextMoveDescription = "Walk Straight Ahead";
}}
```

Get the source row, col from the shorted path list

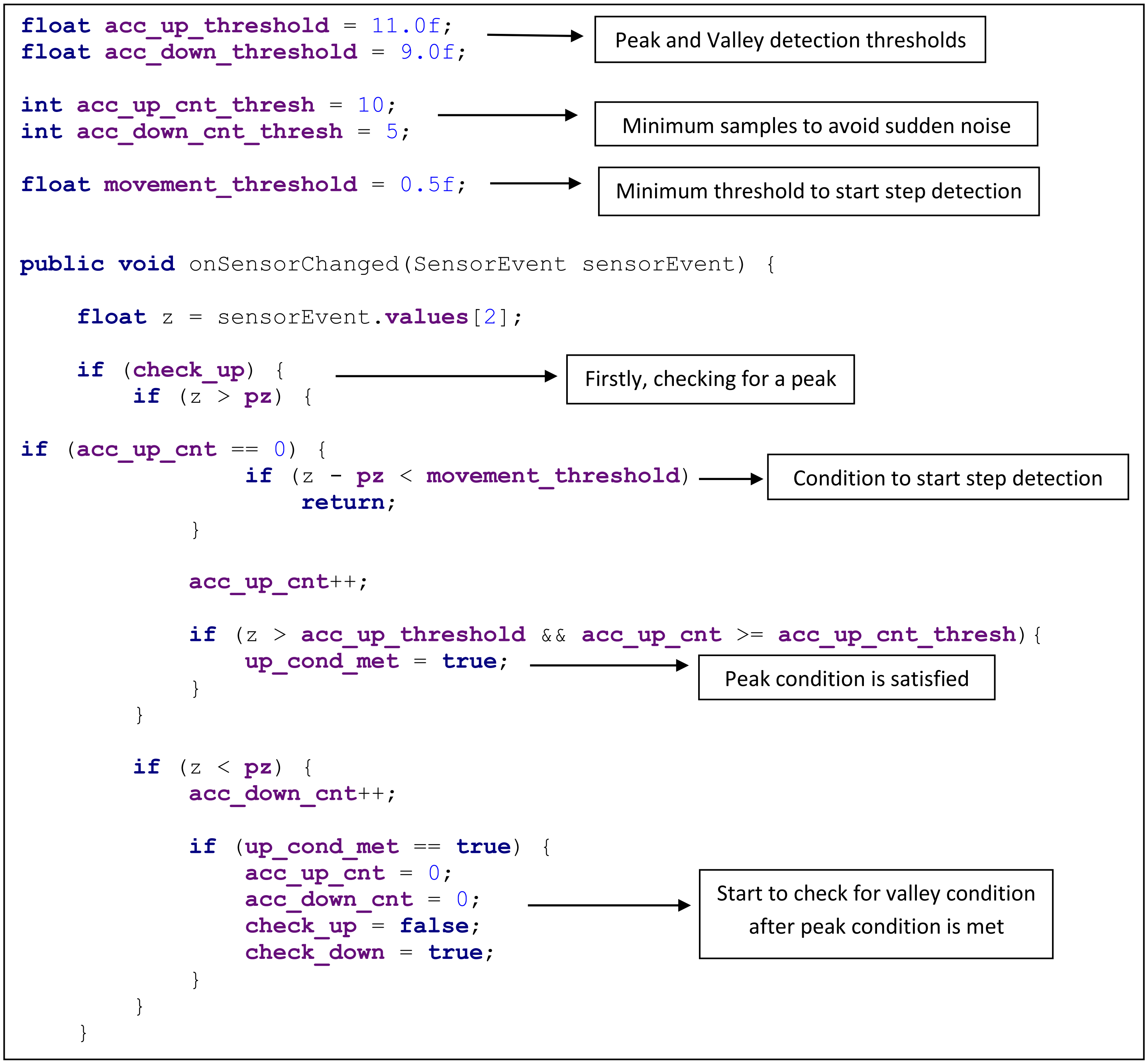
Get the next row, col from the shorted path list

Determining the pointing direction of the next location using previous location

If new and previous tracker are separated by +90 degrees, next move is right turn. If difference of 180 degrees, then next move is to turn back. And if there is no difference, it means the person need to walk straight in same direction

5.7 Step Detector Algorithm

In this project, we have developed own step detector algorithm. The inbuilt step detector algorithm is not feasible to use here because it detects the first step only after detection of a pattern of 6-7 steps in sequence. However, this project demands immediate response with each step, so different step detector algorithm is required. A brief implementation of algorithm is shown below.



```
float acc_up_threshold = 11.0f;
float acc_down_threshold = 9.0f;

int acc_up_cnt_thresh = 10;
int acc_down_cnt_thresh = 5;

float movement_threshold = 0.5f;


public void onSensorChanged(SensorEvent sensorEvent) {

    float z = sensorEvent.values[2];

    if (check_up) {
        if (z > pz) {

if (acc_up_cnt == 0) {
                if (z - pz < movement_threshold)
                    return;
            }

            acc_up_cnt++;

            if (z > acc_up_threshold && acc_up_cnt >= acc_up_cnt_thresh){
                up_cond_met = true;
            }
        }

        if (z < pz) {
            acc_down_cnt++;

            if (up_cond_met == true) {
                acc_up_cnt = 0;
                acc_down_cnt = 0;
                check_up = false;
                check_down = true;
            }
        }
    }
```

Peak and Valley detection thresholds
Minimum samples to avoid sudden noise
Minimum threshold to start step detection
Firstly, checking for a peak
Condition to start step detection
Peak condition is satisfied
Start to check for valley condition after peak condition is met

```
        if (check_down) {
            if (z < pz) {
                acc_down_cnt++;


                if (z < acc_down_threshold && acc_down_cnt >=
                    acc_down_cnt_thresh) {
                    down_cond_met = true;

                    if (up_cond_met == true &&  down_cond_met == true) {

                        numSteps++;
                        up_cond_met = false;

                        mPointerView.changePosition(1);
                        mPointerView.invalidate();
                        mPointerView.checkNextMove();
                    }
                }
            }

            if (z > pz) {
                acc_up_cnt++;

                if (down_cond_met == true) {
                    acc_down_cnt = 0;
                    acc_up_cnt = 0;

                    check_up = true;
                    check_down = false;

                    up_cond_met = false;
                    down_cond_met = false;
                }

                if (acc_up_cnt > start_again_up_check_cnt) {
                    acc_down_cnt = 0;
                    acc_up_cnt = 0;

                    check_up = true;
                    check_down = false;

                    up_cond_met = false;
                    down_cond_met = false;
                }
            }
        }

        pz = z;
    }
```

Next, checking for a valley

Valley condition is satisfied

Move 1 step ahead as both conditions are met

Start checking for peak condition again

If valley condition is not met for an unusual long time, then start to check peak condition

5.8 Turn Detection Algorithm

To detect whether the person has turned right, turned left or turned back, compass sensor is used in this project work. For this, there is a need to check the difference in azimuth value for every turn made. Here is the quick implementation of the algorithm developed.

```
int orientCountLimit = 30;
int stableOrientCountLimit = 100;
int turn_margin = 20;
```

Minimum samples to detect a turn
Minimum sample to start detecting turns
Allowed margin to detect a turn

```
int turn_right1_s = 90 - turn_margin;
int turn_right1_e = 90 + turn_margin;
int turn_right2_s = turn_right1_s - 360;
int turn_right2_e = turn_right1_e - 360;
```

Start and End condition to detect right turn

```
int turn_left1_s = 270 - turn_margin;
int turn_left1_e = 270 + turn_margin;
int turn_left2_s = turn_left1_s - 360;
int turn_left2_e = turn_left1_e - 360;
```

Start and End condition to detect left turn

```
int turn_back1_s = 180 - turn_margin;
int turn_back1_e = 180 + turn_margin;
int turn_back2_s = turn_back1_s - 360;
int turn_back2_e = turn_back1_e - 360;
```

Start and End condition to detect a back turn

```
public void onSensorChanged(SensorEvent sensorEvent) {
    float orientValue = getAvgOrientValue(sensorEvent.values[0]);
    float rotation = 0;
    String direction = "same";
```

Take average to remove sudden noise

```
    if (orientValueStablized == true) {
        rotation = orientValue - baseOrientValue;
        if ((rotation > turn_right1_s && rotation < turn_right1_e)
                || (rotation > turn_right2_s && rotation < turn_right2_e)) {
            rightOrientCount++;
            leftOrientCount = 0;
            backOrientCount = 0;
```

Turn Right condition satisfied

```
        } else if ((rotation > turn_left1_s && rotation < turn_left1_e)
                || (rotation > turn_left2_s && rotation < turn_left2_e)) {
            leftOrientCount++;
            rightOrientCount = 0;
            backOrientCount = 0;
```

Turn Left condition satisfied

```
        } else if ((rotation > turn_back1_s && rotation < turn_back1_e)
                || (rotation > turn_back2_s && rotation < turn_back2_e)) {
            backOrientCount++;
            rightOrientCount = 0;
            leftOrientCount = 0;
        }
```

Turn Back condition satisfied

```
        if (rightOrientCount > orientCountLimit) {
           direction = "right";
           rightOrientCount = 0;
           leftOrientCount = 0;
           backOrientCount = 0;

           baseOrientValue = baseOrientValue + 90;
           if (baseOrientValue > 360)
               baseOrientValue = baseOrientValue - 360;

            mPointerView.changeArrowDirection(direction);
            mPointerView.invalidate();
            mPointerView.checkNextMove();

        } else if (leftOrientCount > orientCountLimit) {
            direction = "left";
            rightOrientCount = 0;
            leftOrientCount = 0;
            backOrientCount = 0;

            baseOrientValue = baseOrientValue - 90;
            if (baseOrientValue < 0)
                baseOrientValue = 360 + baseOrientValue;

            mPointerView.changeArrowDirection(direction);
            mPointerView.invalidate();
            mPointerView.checkNextMove();

        } else if(backOrientCount > orientCountLimit) {
            direction = "back";
            rightOrientCount = 0;
            leftOrientCount = 0;
            backOrientCount = 0;

            baseOrientValue = baseOrientValue + 180;
            if (baseOrientValue > 360)
                baseOrientValue = baseOrientValue - 360;

            mPointerView.changeArrowDirection(direction);
            mPointerView.invalidate();
            mPointerView.checkNextMove();

        }

    } else {
        stableOrientCount++;
        if (stableOrientCount >= stableOrientCountLimit) {
            orientValueStablized = true;
            baseOrientValue = orientValue;
        }
    }
}
```

Increase base value by 90 for right turn

Point tracker pointer to right

Decrease base value by 90 for left turn

Increase base value by 180 for back turn

Check whether values are stabilized after sensor enable

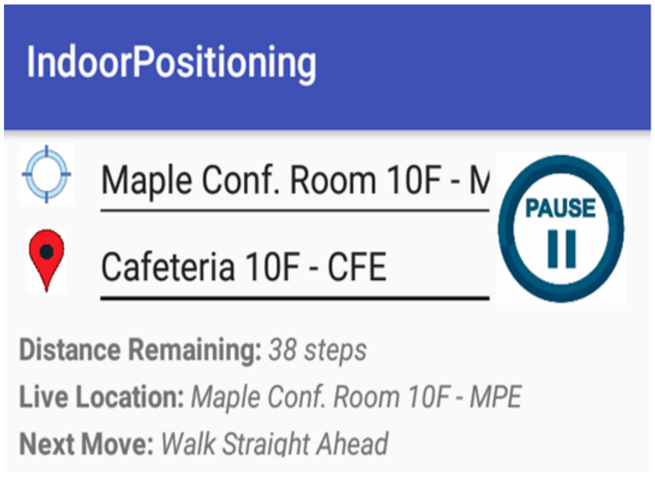
5.9 Floor Detection Algorithm

To detect change in floor, barometer sensor is used as mentioned in above sections. Here is the quick implementation of the algorithm developed.

```
→ Calculate height from pressure data
Height = 44330.0f*(1.0f-(float)Math.pow(pressure/SEA_LEVEL_PRESSURE, 1.0f/5.255f

→ Update absolute floor height
floorHeight[currentFloor] = height;
for (int i = currentFloor - 1; i >= 0; i--) {
   floorHeight[i] = floorHeight[i + 1] - floorHeightDiff[i + 1];
}

→ Update current floor
for (int i = 0; i < MAX_FLOORS; i++) {
   if ((height < floorHeight[i] + floorPadding) && (height >
        floorHeight[i] - floorPadding)) {
        if (currentFloor != i) {
            currentFloor = i;
            updateFloorHeightTime = FIVE_MINUTES;      →  Wait for 5 minutes after a floor is changed
            newTime = prevTime = SystemClock.elapsedRealtimeNanos();
            break;
        }
    }
}

→ Update floor height every two minutes if variance is small
newTime = SystemClock.elapsedRealtimeNanos();

if (checkDataDiff == false) {
    if ((prevTime == 0) || ((newTime - prevTime) > updateFloorHeightTime))
        checkDataDiff = true;
                                      →  Two minutes passed, start checking variance
    else
        baseHeight = height;
} else {
    if ((Math.abs(height - baseHeight) < HEIGHT_VARIANCE)) {
        checkDataDiffCount++;             →  Check if variance is small for 10 samples
    } else {
        checkDataDiffCount = 0;
        baseHeight = height;
    }

    if (checkDataDiffCount > checkDataDiffCountMax) {
        updateFloorHeight(height);        →  Update floor height again as variance is small
        checkDataDiff = false;
        checkDataDiffCount = 0;
        baseHeight = height;
        prevTime = newTime;
        updateFloorHeightTime = TWO_MINUTES;
    }
}
```

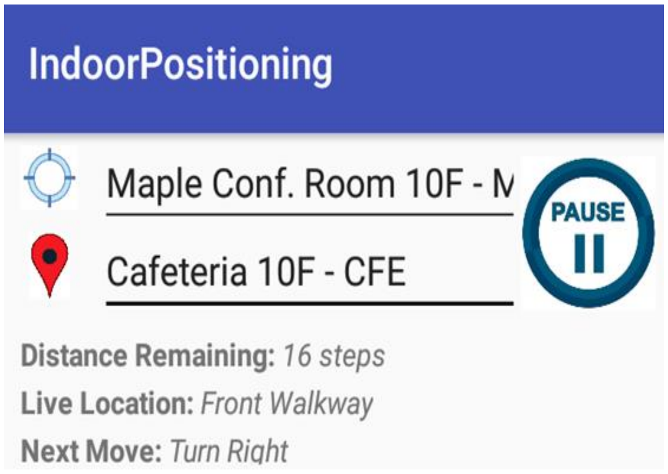
5.10 Live Updates

This project also supports showing live updates as the person makes a new step. We implemented 3 live update feature namely Distance Remaining, Live Location and Next Move. Distance Remaining indicates that how many more steps a person need to make to reach to the destination. If the person makes a correct move, then distance remaining decreases and if the person makes a wrong step then distance remaining increases. Distance remaining is calculated again on every move by applying the Dijkstra algorithm again with new source vertex and hence the new path list size is the new minimum step remaining to reach the destination. Live Location indicates the current location as the person makes a new step. As the map is fully filled using ID's so every map index indicates some location which can be displayed to the person. Next move guides the person for his next step. If some turn need to be made, then next move will indicate to turn right or turn left or turn back. If there is no need to make any turn, then next move indicates walk straight ahead. This is really required because currently the map does not rotate as the person but only the pointer rotates. So direction is with respect to the pointer. So if the person cannot make decision based on pointer's direction, next move section can guide him/her. Below is the snapshot of live update section and how it updates as person makes some steps.



Distance Remaining: *38 steps*
Live Location: *Maple Conf. Room 10F - MPE*
Next Move: *Walk Straight Ahead*

After 22 Steps ➡



Distance Remaining: *16 steps*
Live Location: *Front Walkway*
Next Move: *Turn Right*

As can be seen from above pictures, initially the person need to make 38 steps, live location is maple and next move is walk straight. After the person make 22 step in straight direction, the person reaches a point where he need to make a turn right so next move is updated from Walk Straight Ahead to Turn Right. Distance Remaining is changed from 38 step to 16 steps. And the new location where person has reached is just a front walkway so it is changed from Maple Conference Room to Front Walkway. And the new shortest path is also drawn starting from current location. Similarly, live update is made for every single step a person makes while walking towards the destination.

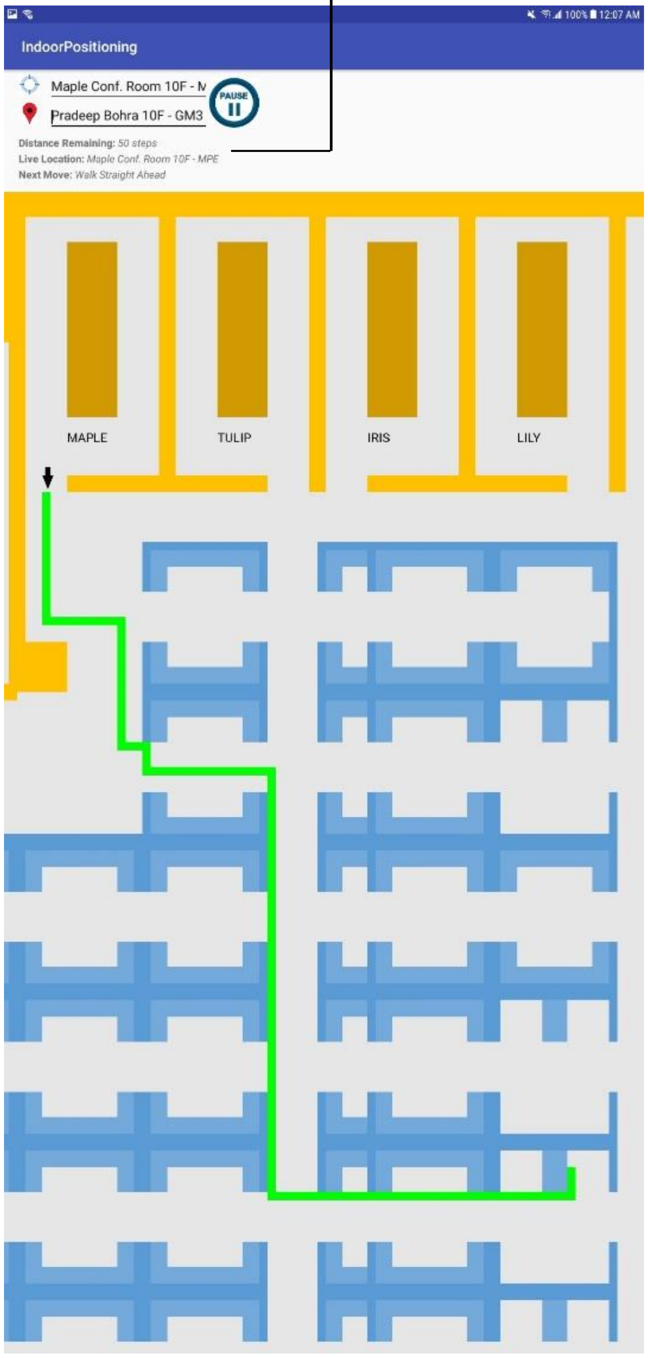
5.11 Project Live Working Snapshots

Let’s consider a case where person need to go from Maple Conference Room to Group Leader of System R&D Team. Here are the snapshots of the walk performed by the person

Distance Remaining	:	50 Steps
Live Location	:	Maple Conference Room
Arrow Direction	:	Down
Next Move	:	Walk Straight

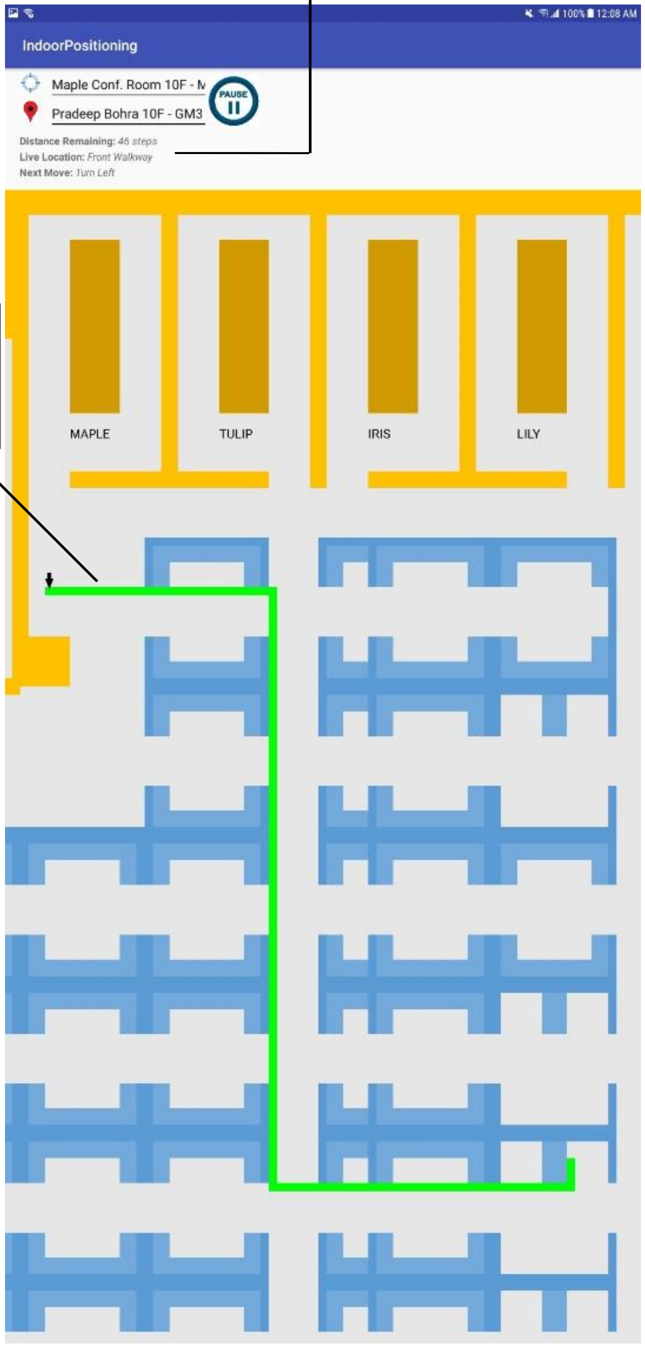
After 4 steps

Distance Remaining	:	**46 Steps**
Live Location	:	**Front Walkway**
Arrow Direction	:	Down
Next Move	:	**Turn Left**



Distance Remaining: *50 steps*
Live Location: *Maple Conf. Room 10F - MPE*
Next Move: *Walk Straight Ahead*

New Shortest Path



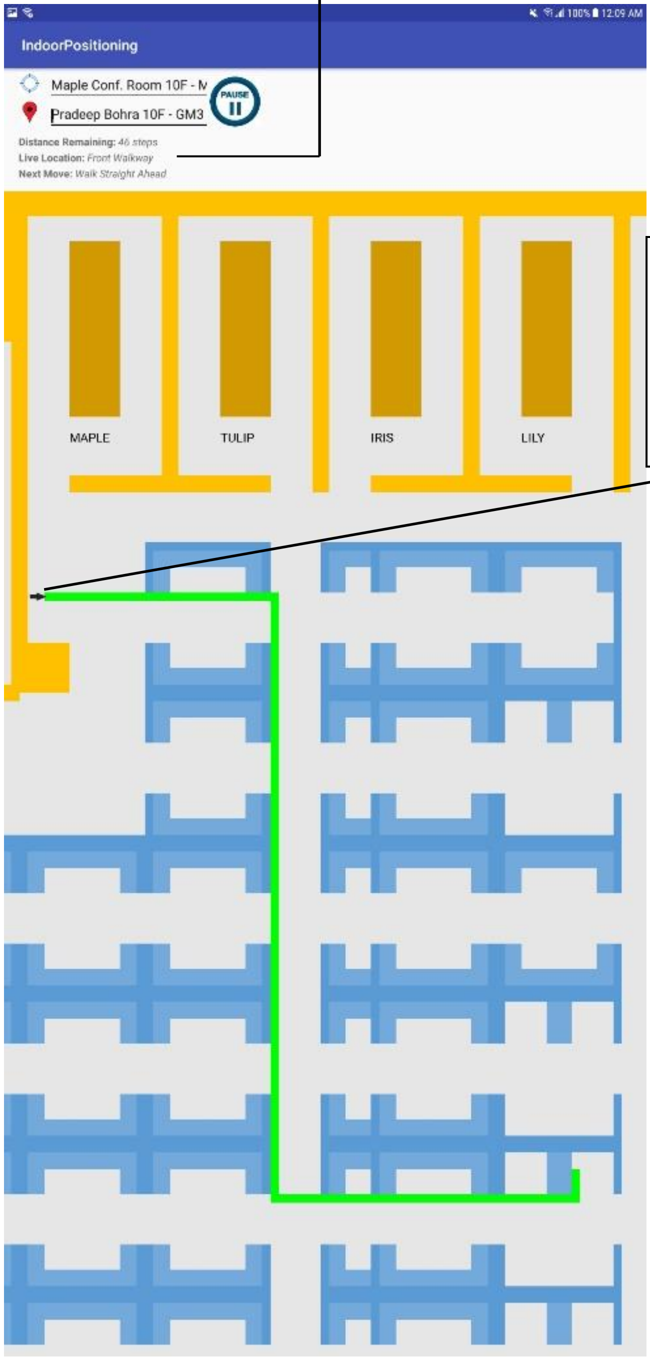
Distance Remaining: *46 steps*
Live Location: *Front Walkway*
Next Move: *Turn Left*

After turning left

Distance Remaining	:	46 Steps
Live Location	:	Front Walkway
Arrow Direction	:	**Right**
Next Move	:	**Walk Straight Ahead**

After 10 steps And Right Turn

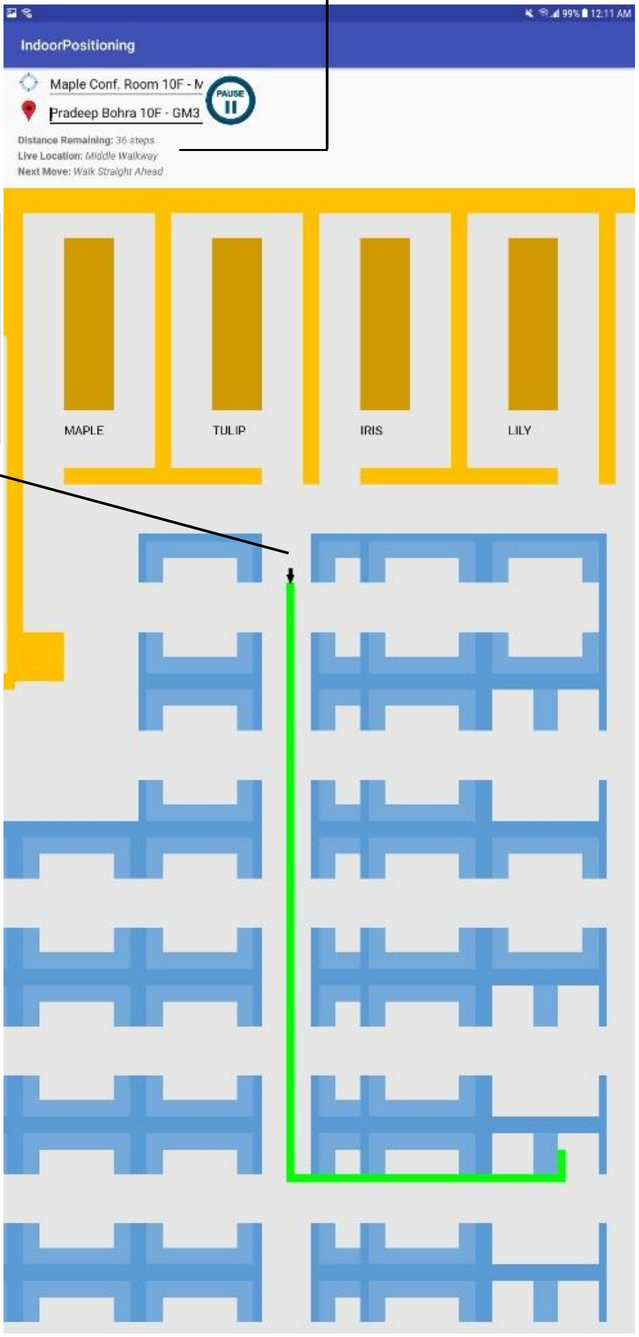
Distance Remaining	:	**36 Steps**
Live Location	:	**Middle Walkway**
Arrow Direction	:	**Down**
Next Move	:	**Walk Straight Ahead**


IndoorPositioning

Maple Conf. Room 10F - M
Pradeep Bohra 10F - GM3

Distance Remaining: *46 steps*
Live Location: *Front Walkway*
Next Move: *Walk Straight Ahead*

Arrow Changed Direction After Turns


IndoorPositioning

Maple Conf. Room 10F - M
Pradeep Bohra 10F - GM3

Distance Remaining: *36 steps*
Live Location: *Middle Walkway*
Next Move: *Walk Straight Ahead*

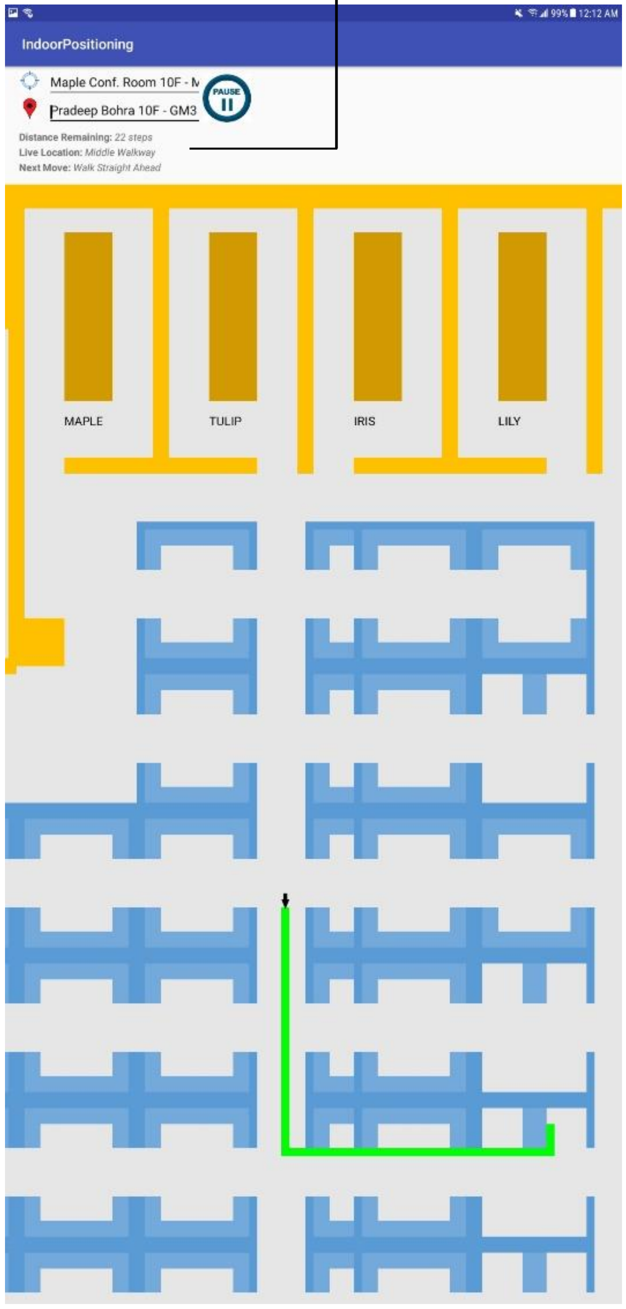
After taking
14 steps

Distance Remaining	:	**22 Steps**
Live Location	:	Middle Walkway
Arrow Direction	:	Down
Next Move	:	Walk Straight Ahead

After 9
steps

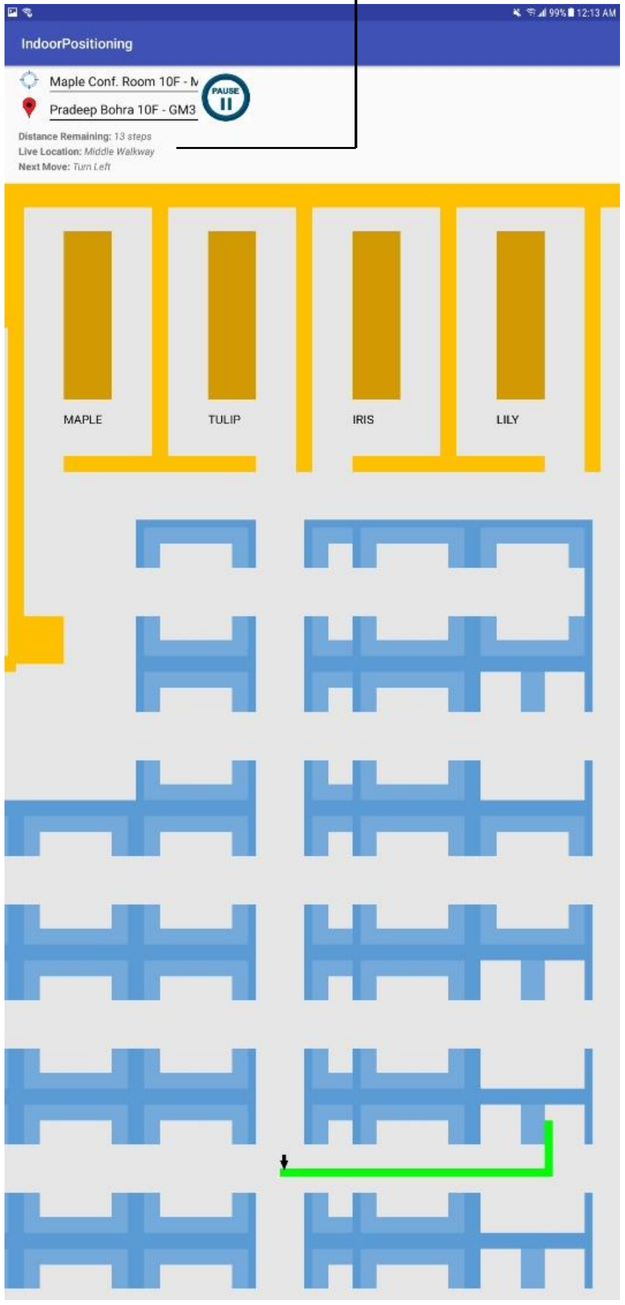
And No
Turn

Distance Remaining	:	**13 Steps**
Live Location	:	Middle Walkway
Arrow Direction	:	Down
Next Move	:	**Turn Left**


IndoorPositioning

Maple Conf. Room 10F - M
Pradeep Bohra 10F - GM3

Distance Remaining: 22 steps
Live Location: Middle Walkway
Next Move: Walk Straight Ahead


IndoorPositioning

Maple Conf. Room 10F - M
Pradeep Bohra 10F - GM3

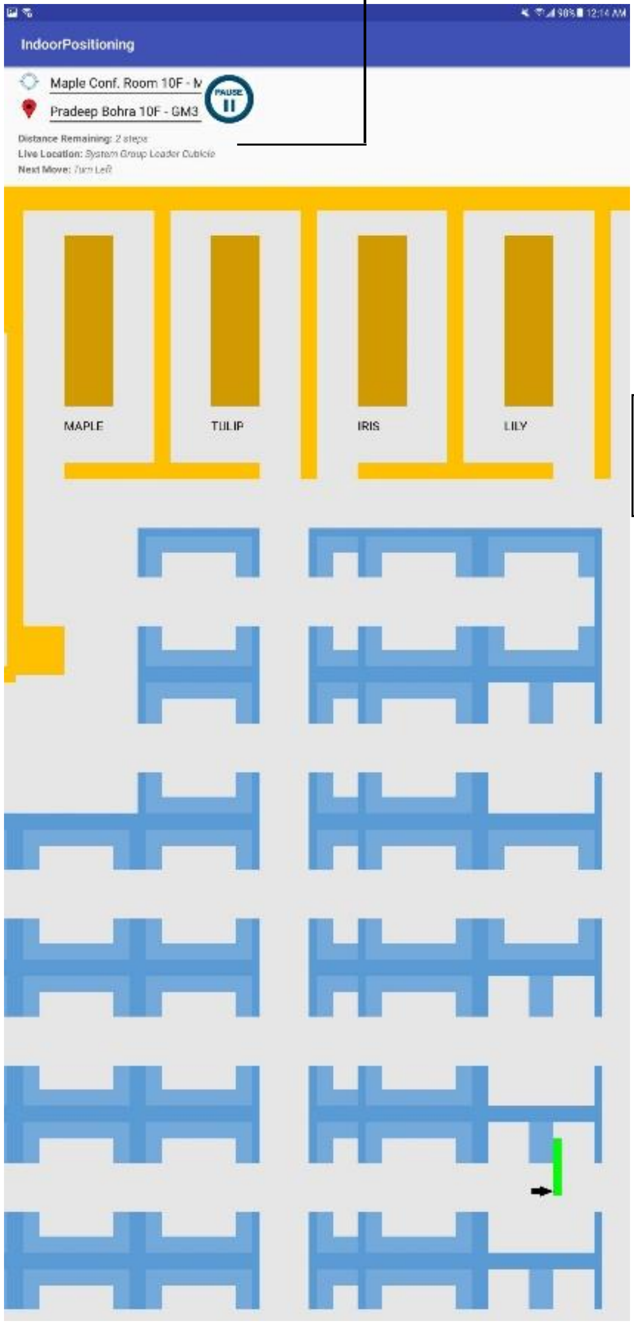
Distance Remaining: 13 steps
Live Location: Middle Walkway
Next Move: Turn Left

After Turning Left
and taking 11 steps

Distance Remaining	:	2 Steps
Live Location	:	System Group Leader Cubicle
Arrow Direction	:	Right
Next Move	:	Turn Left

After Turning Left and taking 2 steps

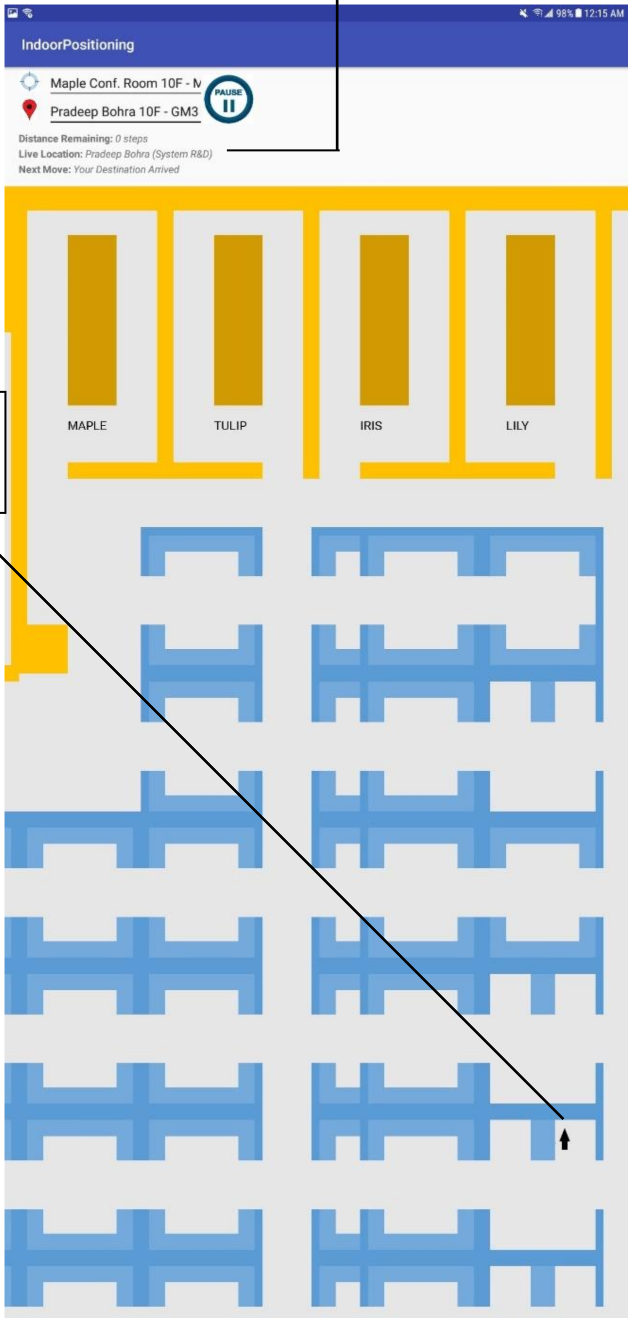
Distance Remaining	:	0 Steps
Live Location	:	Pradeep Bohra (System R&D)
Arrow Direction	:	Up
Next Move	:	Destination Arrived



Maple Conf. Room 10F - M
Pradeep Bohra 10F - GM3

Distance Remaining: *2 steps*
Live Location: *System Group Leader Cubicle*
Next Move: *Turn Left*

Destination Arrived



Maple Conf. Room 10F - M
Pradeep Bohra 10F - GM3

Distance Remaining: *0 steps*
Live Location: *Pradeep Bohra (System R&D)*
Next Move: *Your Destination Arrived*

Chapter 6. CONCLUSION

Due to the wide spread of smartphones now a days and the strong use of GNSS based position estimation applications, the extension to navigation in GNSS shadowed areas is a very interesting application.

The main objective of this theis is to check the feasibility of indoor navigation using mems sensor available in the smart phones and to determine its accuracy. In this project, accelerometer sensor, compass sensor and barometer sensor is used to determine the position of person. Accelerometer sensor was used to determine the step taken by a person by developing a self-designed algorithm for step detection, compass sensor is used to determine the change in direction of a person and barometer sensor is used to determine the change in floor of a building. Change in direction of person was found to be giving 100% accurate results. Change in floor using barometer sensor was also found to be giving 100% accurate results. Although pressure changes with climate but with dynamic calibration approach and need to detect change and not absolute pressure, there was no problem found using barometer sensor for detecting change in height. But the distance travelled was found to be given 98% accurate results. It was found that for distance estimation, the length of a single step is a very important factor. Since length of each step is little bit different, the small errors accumulate with each step and makes up to 2 meters of error for each 100 meters. In this project, fixed length of a step is used as complete testing was performed on a single individual. To determine step length of different individuals, training data or calibration technique can be used. But practically, a person cannot make each step of same length so more manipulation of accelerometer data is required to determine step length dynamically with each step. As it was found during experiments, there is a noticeable difference in accelerometer data pattern of long step length and short step length. Considering usual walking of a person, it was found the person usually makes step of approximately same length so accuracy achieved was very good as mentioned it is just of 2 meters for 100 meters of distance travelled.

As the accuracy achieved using mems sensors to detect position of a person is found to be very good, so this approach is highly feasible to be used for indoor navigation and can very strongly support in other techniques used for indoor navigation like Wi-Fi, Bluetooth, computer vision. So combining mems sensor with Wi-Fi can output 100% accurate results. This kind of navigation technique has various prospective application areas such as big hospitals, big shopping malls, large hotels, large offices, museums, etc. It is highly likely that in the future a similar system is developed into a commercial product soon.

Chapter 7. FUTURE WORK

As this is a very big project and after doing research and experiments during this thesis, this thesis could be extended to achieve better results and resolve problems found during this research work.

Firstly, a dynamic algorithm could be developed to determine the length of step for every step taken by an individual to achieve better accuracy and even remove error of 2 meters found in this project implementation. Secondly, although the shortest distance is correct but shortest path algorithm could be tuned more to give straight paths at all instances rather that zig zag paths at certain places as Dijkstra algorithm cannot distinguish between a forward vertex and the downward vertex because both has same weight edge connected and practically, a person travels in straight and not zig-zag. Thirdly, in this project, as the person changes his direction, the pointer indicating person location rotates and provides next turn guidance as well but the person has to refer to the pointer for next move so a map rotation feature could be added to this work in future with which the map will rotate itself as the person change his direction so to have a better user experience and highly ease the person to determine his next move that whether he need to take a left turn or right turn as now the map orientation will match to the direction the person currently pointing towards. Fourthly. In this project, a compass sensor is used to determine direction which give very good performance but it is not the best choice for quick turns. So a gyroscope sensor could be used in this project in future to detect quick turns taken by a person and therefore there will be a better user experience as now the new direction will get calculated in just 200 ms from 1 second.

Next, in this project a statically and manually designed map is used and for creating larger maps, a lot of manual effort is required. Also, if the map changes due to some renovation work and new requirement then manually then map need to be updated with new measurement manually taken. So fifthly, a new technique need to be implemented which could draw the map by itself by sending and receiving some signals from obstacles and then only the map need to be marked with places. Lastly, this project could not detect initial or first position of person as it is not possible to detect it through mems sensors used in mobile phones so our project requires the person to start from the same location which the person has self-entered in the source text box. Therefore, this problem could be solved in the future by determining the first location using WIFI. Therefore, this project could be combined with Wi-Fi based positioning so that Wi-Fi + mobile sensors combined could output 100% accurate results.

REFERERENCES

[1] Thomas Willemsen, Friedrich Keller, Harald Sternberg, "Concept for building a MEMS based indoor localization system", IEEE International Conference on Indoor Positioning & Navigation, 27-30 October 2014, Busan, S. Korea

[2] Celalettin Uyar, İsmail Dervişoğlu, Metehan Güzel, "Multi sensor based indoor positioning", IEEE International Conference on Computer Science and Engineering (UBMK), 5-8 Oct 2017, Antalya, Turkey

[3] Sangjae Lee, Namkyoung Lee, Jeonghee Ahn, "Construction of an indoor positioning system for home IoT applications" IEEE International Conference on Communications (ICC)n, 21-25 May 2017, Paris, France

[4] Qian J, Ma J, Ying R, Liu P, Pei L, "An Improved Indoor Localization Method UsingSmartphone Inertial Sensors", International Conference on Indoor Positioning and Indoor Navigation, 28-31 October 2013, Montbiliard, France

[4] Roie Melamed, "Indoor Localization: Challenges and Opportunities", IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft), 16-17 May 2016, Austin, TX, USA

[5] Keller. F, Sternberg H, Willemsen T, "Calibration of Smartphones for the use in indoor navigation", IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN), 13-15 November 2012, Sydney, Australia.

[6] Goyal P, Ribeiro V, Saran H and Kumar A, "Strap-Down Pedestrian Dead-Reckoning System", IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN), 21-23 September 2011, Guimara, Portugal

[7] Android Developer, https://source.android.com/devices/sensors/sensor-types, 2017

[8] Biljana Risteska Stojskoska1, Ivana Nižetić Kosović, Tomislav Jagu, "A Survey of Indoor Localization Techniques for Smartphones", 8th International Conference ICT Innovations, 2016, Ohrid, Macedonia

[9] Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Indoor_positioning_system, 2017

[10] Zuwei Yin, Chenshu Wu, Zheng Yang, "Peer-to-Peer Indoor Navigation Using Smartphones", IEEE Journal on Selected Areas in Communications, 09 March 2017

[11] Jó Ágila Bitsch Link, Paul Smith, Nicolai Viol, "Footpath: Accurate map-based indoor navigation using smartphones", IEEE International Conference on Indoor Positioning and Indoor Navigation, 21-23 Sept 2011, Guimaraes, Portugal

[12] You Li, Peng Zhang, Xiaoji Niu, "Real-time indoor navigation using smartphone sensors", IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN), 13-16 Oct 2015, Banff, AB, Canada