FINAL PROJECT REPORT

Car Navigation System in Indoor Parking Space

Group Members:

Nikhil Ghate
Sagar Kangutkar
Shreyas Rokade
Aditi Banarse

Table of Contents

List of Figures

1. INTRODUCTION

With ever increasing usage of smartphones and mobile Internet, calculating precise location of a user even within indoor spaces where there is very little or no GPS availability, has become possible. Uses of this advances are numerous such as enhancing building safety, tackle cellular dead zones in building which will be discussed later.

Even though now a days various parking technologies are used in parking to facilitate ease of parking, yet it is difficult for drivers to find timely vacant parking spaces within multi-storied parking lots where navigation assistance is not available when Global Positioning System (GPS) does not work well. Such difficulties result in unnecessary driving around parking lot to just find vacant parking space. Consequently it causes extra carbon dioxide emissions and deteriorates the environment. This problem aggravates when many people are simultaneously looking for parking places in crowded areas such as downtown area at rush hours. On the other hand, it also increases the risk of traffic accidents when drivers have to search for parking spaces while driving. In addition, unpredictable parking situations make it difficult for people to plan their mobility. All of these degrade the modern city ecosystem experience

The software proposed in this project is a light-weight android application running on a user's smartphone, which essentially captures users location based on Wireless Local Area Network (WLAN) signals being captured by the device at the same time allows users to select a destination to a particular destination which could be an empty parking slot available in the lot.

The whole project could be divided into several phases which were carried out either simultaneously or sequentially. We will describe steps in detail below.

2. TASKS

2.1 SELECTING A PARKING LOT

Selecting a suitable parking lot was initial hurdle as the selected indoor location should have sufficient number of Wi-Fi signals. So number of wifi signals available was considered as primary measure for parking lot selection as well as availability of GPS signal was also taken into consideration.

Based on above measures 'Medical District Apartment's' parking lot was decided. One could get at least 40 unique wifi signals spread throughout parking space. It's a two story parking structure located at Taylor-Ashland intersection where second level is open i.e. outdoor.

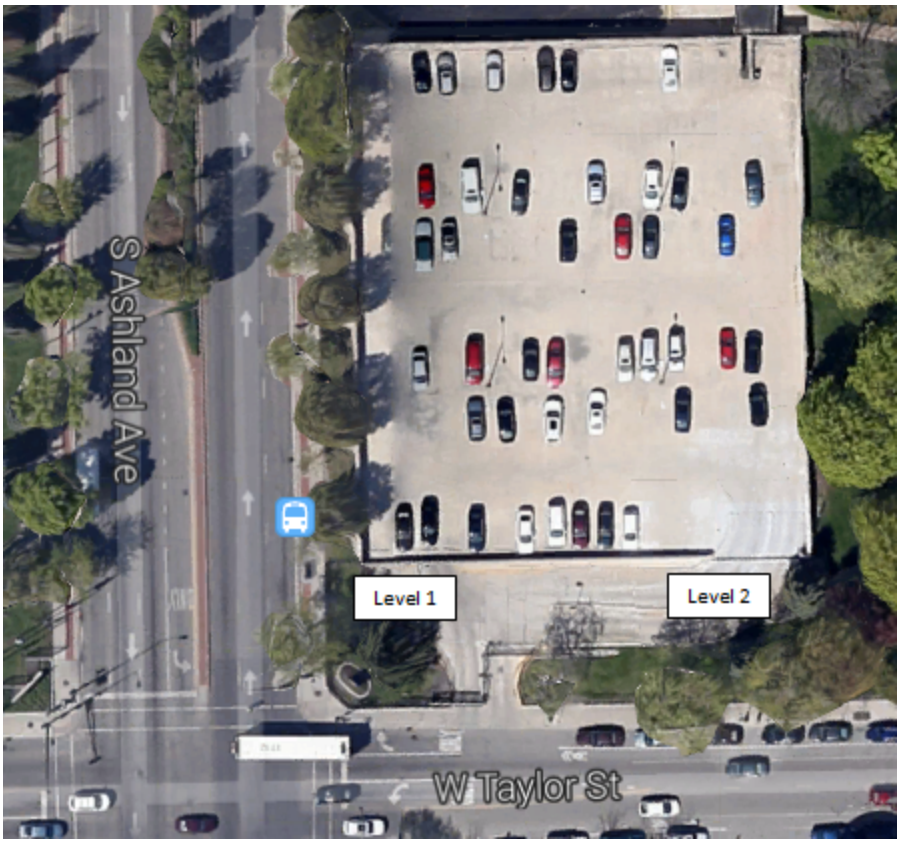

Figure 1: Google Map image of the parking lot.

More about parking lot:

This parking could accommodate approx. 180 cars. Access to level 1 and 2 could be seen in the image. Area of the parking lot is 2930 square meters and every parking lot of size 13 square meters.

Entire parking lot was logically divided into a grid of size 9x15 (Every cell either being a parking slot or navigation path). The following are the components of the parking lot:

- Blue rectangular shapes - parking cells
- White spaces between them - navigation cells
- Blue circular shape on the left bottom - entry to the lot
- Black rectangular shape on the top right - unusable space of the lot

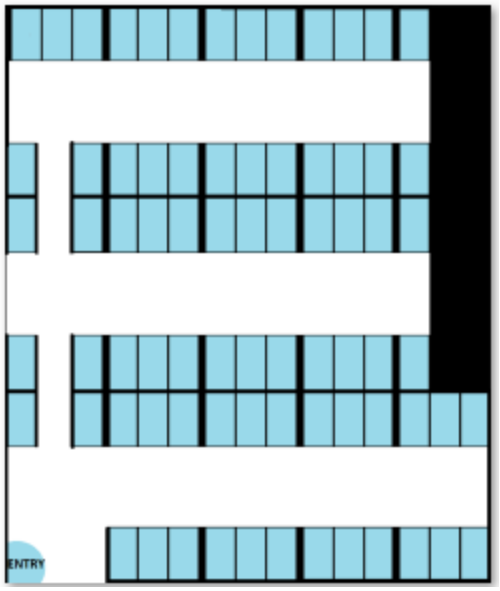

Figure 2: Blueprint of the parking lot

2.2 CREATING GEOFENCE

A geofence is a virtual barrier. Programs that incorporate geo-fencing allow an administrator to set up triggers so when a device enters (or exits) the boundaries.
'Auto-Location' and 'Tasker' were the third party applications were used to implement geo-fencing capabilities.

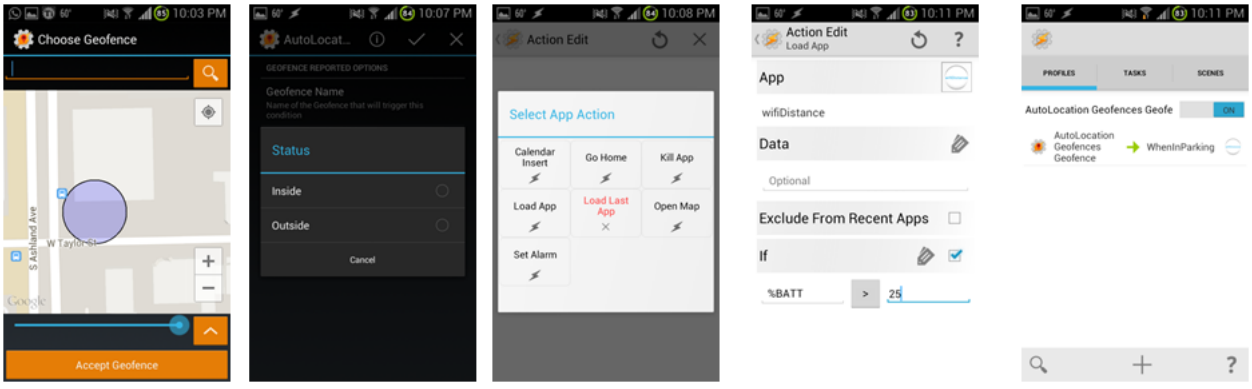

Figure 3: Geofencing application

Steps taken :

- Define geofence in AutoLocation
- Create a Tasker rule with following criteria

- When user is inside geo fence
- Battery is > 25%

Application should launch automatically and allow user to navigate.

2.3 PACKAGE STRUCTURE

In brief, we have described the package structure as given below:

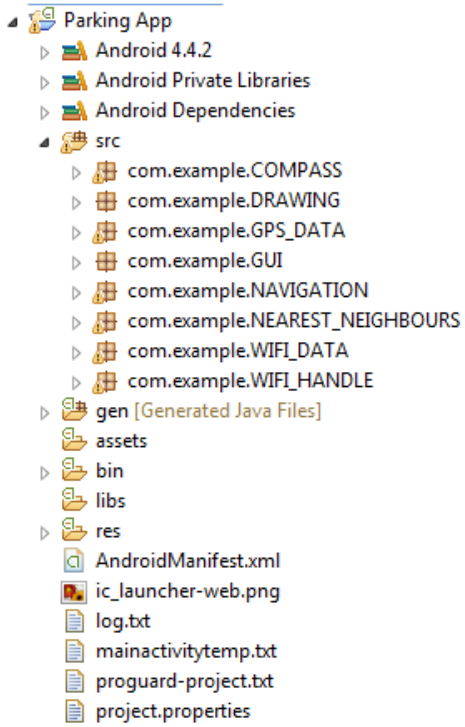

Figure 4: Package Structure

- COMPASS - This package describe the compass that we had planned to include on our driving directions page.
- DRAWING - This package deals with the drawables related to the canvas. It is largely unused.
- GPS_DATA - We have put all our code related to extraction of GPS co-ordinates and storing the values in the database in this package.
- GUI - The main execution start point of the application is here. We link the main page with the other parts of the application such as navigation, offline phase, route computation etc.
- NAVIGATION - Navigation package has all the code related to computing the shortest paths between the two nodes, placing the start/end points as either a parking cell or on the main navigation path/block. This is the main package as far as the computation for localization and navigation goes.

- NEAREST_NEIGHBORS - We calculate the route between the start and the end points. Based on the intermediate cells, we get the nearest neighbors and therefore we can keep our localization between those points. This allows our systems to have a better accuracy in terms of percentage.
- WIFI_DATA - This is concerned with the storage of Wi Fi data. The Wi Fi data is stored majorly in the offline mode and this is the package that takes care of that.
- WIFI_HANDLE - Wi Fi handle package is concerned with the operation on Wi Fi signals of the particular phone handset.

2.4 CREATING UNDERLYING DATABASE

As project aims at creating an android app which will run on a device with limited specs and since it is highly likely that the user won't have 'data network' to communicate with database residing on internet (e.g. Open Street Maps data or Skyhook Data) it was decided to create database which will remain internal for the application. SQLite was decided to use as dbms due to several reasons such as less memory footprint, ease of use and most important fact that it is an open source database.

2.4.1. Selecting suitable wifi signals.

Before we could generate underlying database schema it was necessary to identify which wifi signals would be used to localize user within the indoor space. 3rd party application of 'wifi Analyzer' was used to identify these signals. As it is well knows that different cellphone's (of different phone manufacturers) give different wifi signal reception at any location we decided to use multiple smartphones; LG Nexus, Samsung Galaxy S2, Sony Xperia, Apple Iphone were used to take the readings. These readings were taken at different day and time.

Readings were compared in order to find out most prominent wifi SSID's which would allow us to cover the whole parking lot. Wifi SSID's were chosen based on number of cells that particular Wifi is available in. At same time we also had to take into account a possibility that a wifi access point could go out of service at any moment hence we had to select enough number of access points to tolerate such loss of signals. Out of 40 available wifi signals 25 signals were selected.

Wifi SSID	# of Cells
2WIRE912	47
ATT848	79
ATT384	71
Westeros	71
ATT912	57
TullNet	45
ATT424	44
belkin.17b	41
ATT704	38
Sunshine	38
Sameer	34
HOME-D024	32

Wifi SSID	# of Cells
ATT072	30
HOME-6A72	29
allen	24
ATT528	22
HOME-0182	22
HOME-0972	22
NETGEAR94	22
HOME-DB88	21
ATT168	18
HOME-1E92	18
NETGEAR26	18
Vitaaksi	18
Biranavi	17

Figure 5: Available Wi Fi SSIDs and number of cells they cover.

IMPROVEMENT :

Wi Fi SSIDs are prone to changes i.e. a user could easily change Wifi network name(SSID). Hence using SSID is less reliable; instead one should use MAC id of the access point as it is always unique and never changes.

2.4.2. Creating schema for the database.

Considering limited ram and computational capacity available in handheld devices un-normalized tables were maintained.

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	id	INTEGER	1	null	1
1	x	INTEGER	1	null	0
2	y	INTEGER	1	null	0
3	2WIRE912	INTEGER	0	0	0
4	ATT848	INTEGER	0	0	0
5	ATT384	INTEGER	0	0	0
6	Westeros	INTEGER	0	0	0
7	ATT912	INTEGER	0	0	0
8	TullNet	INTEGER	0	0	0
9	ATT424	INTEGER	0	0	0
10	belkin.17b	INTEGER	0	0	0
11	ATT704	INTEGER	0	0	0
12	Sunshine	INTEGER	0	0	0
13	Sameer	INTEGER	0	0	0
14	HOME-D024	INTEGER	0	0	0
15	ATT072	INTEGER	0	0	0
16	HOME-6A72	INTEGER	0	0	0
17	allen	INTEGER	0	0	0
18	ATT528	INTEGER	0	0	0
19	HOME-0182	INTEGER	0	0	0
20	HOME-0972	INTEGER	0	0	0
21	HOME-DB88	INTEGER	0	0	0
22	ATT168	INTEGER	0	0	0
23	HOME-1E92	INTEGER	0	0	0
24	NETGEAR26	INTEGER	0	0	0
25	Vitaaksi	INTEGER	0	0	0
26	Biranavi	INTEGER	0	0	0

Figure 6: Database Schema for Wi-Fi

Id field is Primary Key,
x, y fields correspond to cell x and y coordinates
All other columns are nothing but the SSIDs identified earlier and responsible for storing signal strengths for each x, y cell.

Another table was created to store GPS coordinates at every cell.

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	ID	INTEGER	1	null	1
1	X	INTEGER	0	null	0
2	Y	INTEGER	0	null	0
3	Lat Top	NUMERIC	0	null	0
4	Lat Bottom	NUMERIC	0	null	0
5	Long Left	NUMERIC	0	null	0
6	Long Right	NUMERIC	0	null	0

Figure 7: Database Schema for GPS

Same as above the id is primary key; x, y indicates cell
Lat Top indicates Latitude at top of the particular cell ; similarly Lat Bottom indicates Latitude at bottom.
Long Left, Long Right indicates longitude at cells left and right edge respectively.

2.4.3. Creating and populating actual database files.

In order to create and populate the sqlite database an app was created (DB_TEST1)

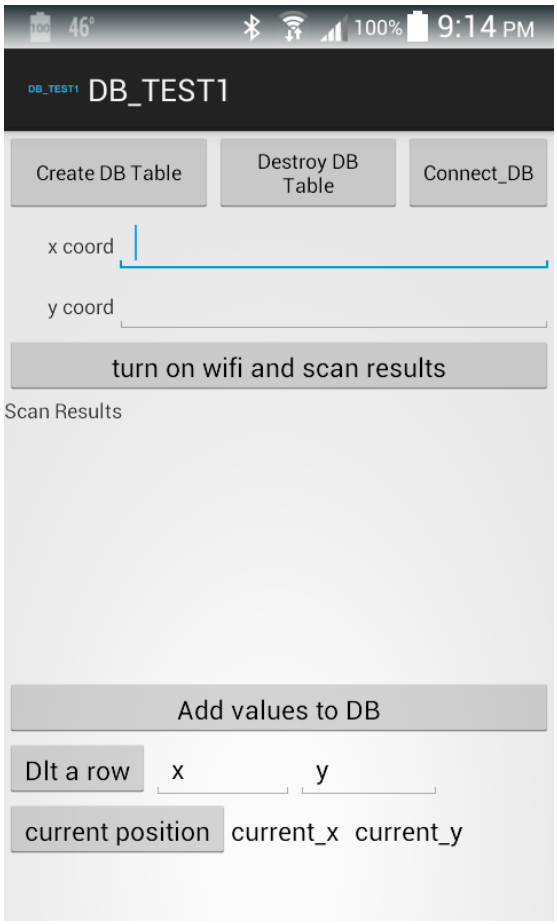

Figure 8: Database creation

Overall process was to create a .sqlite file which will be stored on device itself; and it will hold the tables mentioned above; final application should communicate this database on device storage for localization.

Functionality :
- Connect DB : This will allow app to connect to underlying database .sqlite file.
- Destroy DB Table : Responsible for deleting tables present within the database.
- Create DB Table : Create necessary tables.

- Turn on wifi and scan : This button will ensure that wifi is turned on and it will also scan the available wifi signals at the particular location i.e. individual cell in which user is currently standing.
- Scan Result TextArea will display these results.
- Add Values to DB will add wifi signal strengths to the underlying wifi data table. X coord and y coord are provided by user.
- Dlt a row will allow you to delete rows from database
- current position – Responsible for localizing users position. (Discussed Later)

Usage :
Application was installed on separate devices and each device was responsible for creating and populating the database. These offline readings were taken multiple times on separate days.
Once the database was populated within each device it was pulled and merged to create one single database having multiple scan readings for every cell.
For each cell approximately 47 readings were recorded.

x	y	Number of readings
1	1	45
1	2	48
1	3	48
1	4	45
1	5	45
1	6	45
1	7	45
1	8	45
1	9	45
1	10	45
1	11	45
1	12	48
1	13	45
1	14	45
1	15	45

x	y	Number of readings
3	1	45
3	2	44
3	3	46
3	4	47
3	5	48
3	6	44
3	7	44
3	8	43
3	9	47
3	10	46
3	11	49
3	12	46
3	13	42
3	14	45
3	15	49

Figure 9: Sample screenshot of final database

2.4.4. My Position

LOCALIZATION:

Current_Position function written would take as an input currently available wifi signals and their strengths store both in an array and compare this with underlying wifi data database table and will give the matching row's x and y coordinates i.e. current cell. This function also

considers Nearest Neighbor cells. The system localizes all by itself on the first try. For all the consequent attempts, the app uses the last known position for its localization work. Whatever is the previous known location of the application is stored outside the application. When the user tries to 'reset' the map or tries to 'get current position' any time after that, the app simply makes use of the reference of the last known position that is stored outside the file. Although the app was very robust and designed to prevent crashes at as many places as we could detect, we did this external storage of the last known location so as to be able to handle this scenario where in the app had to be restarted. In case the last know location were saved within the app, that location would have been lost along with the crash itself. Saving the value to an external file helps us retrieve it even after a crash or closing the app where ever need be. Please refer to code for further details.

The image given below represents how the application displays the localized position of where the person is standing. The cell highlighted with dark-blue rectangle is the place where the app has determined that we currently are.

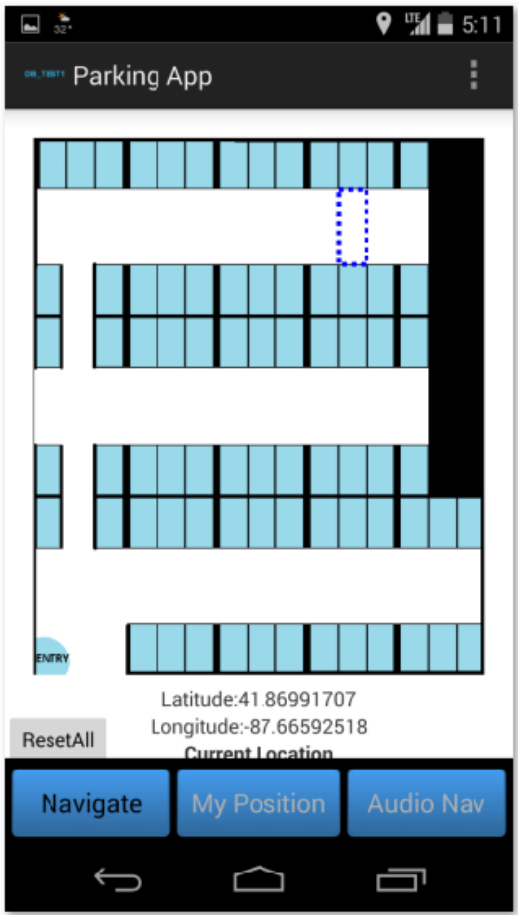

Figure 10: UI for My Position

EXPERIMENTAL RESULTS:

After multiple readings it was found that app could localize the user an average of 8.5 out of 10 trials.

The localization was affected only in cells where signal strengths would vary a lot over short period of time. Please see the rest of the report to know the finer details of this particular occurrence.

2.5 GPS IMPLEMENTATION FOR BETTER LOCALIZATION:

GPS is deemed as an outstanding navigational system due to its capability to attain high positioning accuracy (ranging from tens of meters down to millimeters) and its signal availability to users anywhere on the globe (air, ground, and sea applications). We experimented with the idea of GPS along with Wi-Fi for better localization. GPS, as we all know, is a power hungry and generally outdoor source of global positioning. Still, because of the nature of our parking lot we decided to go ahead with the GPS experiment for localization.

2.5.1. Experiment

The GPS technology is based on time of arrival TOA, which is the time interval for the signal to travel from the satellite to the receiver. We wrote a program for calculating the user's exact location using latitude and longitude co-ordinates. The program consisted of the following

- Use of the LOCATION_SERVICE of Android's native library Location Manager.
- getBestProvider which used a combination of GPS satellites and mobile networks to get an accurate position of the user at that instant.
- getLastKnownLocation in case the satellites are not reachable at that instant. In this case the receptor kept on searching for satellites and provided an update as soon as a connection was established.
- requestLocationUpdates which kept on providing updates at specified intervals (milliseconds) and/or when the user moved more than a specified distance (meters).
- getLatitude and getLongitude returned the latitude and longitude coordinates respectively whenever the function was called and these were used to determined the accurate position of the user within the parking lot.

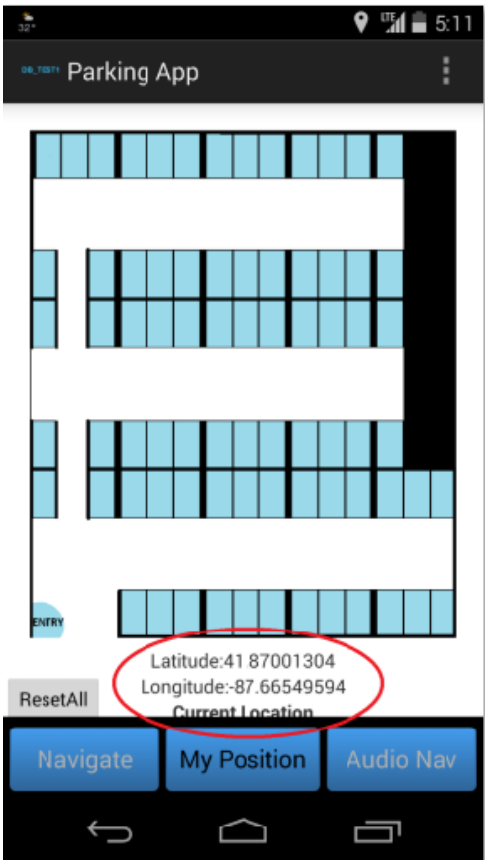

Figure 11: GPS coordinates

2.5.2. Result

GPS worked well when we were at the entrance of the parking lot with direct line of sight to the satellites. Latitude and longitude updates were happening quickly in this case. But when we entered inside the parking lot, the time for updates dropped drastically and location updates happened only when we were at the edges of the parking lot or stood for a long time at the same position anywhere inside the parking lot. It required a lot of time to contact the satellite and get a fix on the position of the user. The accuracy for localization dropped considerably and results were poorer than those achieved just using Wi-Fi.

Attached are two screenshots which were taken at position 1,2 (which is the top left in our case) and position 1,10 (which is the 6th cell from right in the top row).

The latitude and longitude readings remained the same for both locations and changed only when we went to cell 1,15 (top right) which had an opening on the right side.

For this reason, we were forced to consider only Wi-Fi for localization.

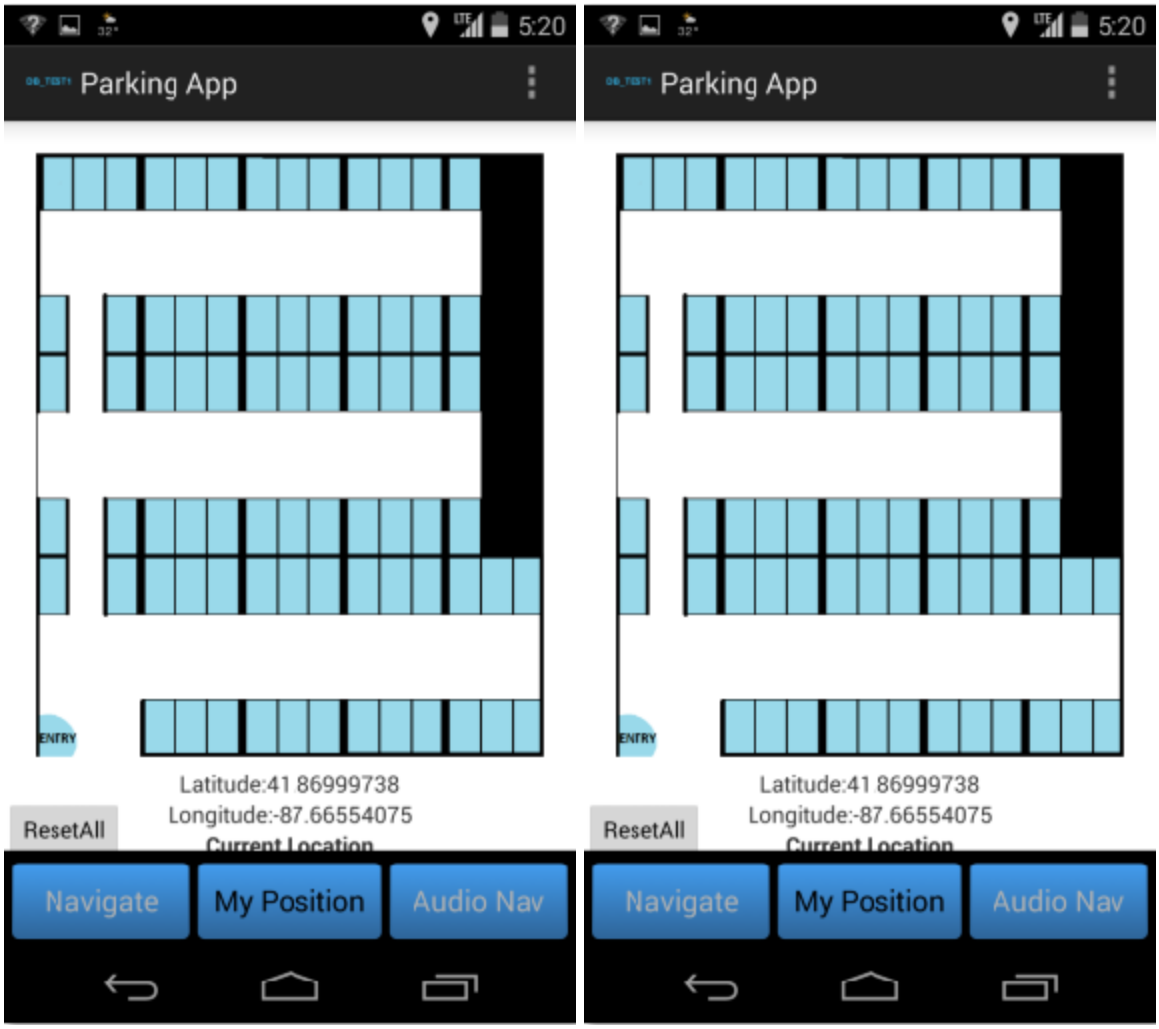

Figure 12: GPS coordinates not changing

2.6 USER INTERFACE:

2.6.1. Main Screen

The main screen consists of a map and three buttons for the user's benefit. The map is long enough and wide enough to occupy a significant portion of the screen and the buttons are placed right at the bottom so as to keep the thumb movement of the user to a minimal level. There is also a 'reset all' button on the left, just above the Navigate button. The user, when he opens up the app will be shown an empty map to which he will be able to make an entry by either:

- Using the ‘My Position button - It localizes him and displays the location on the map
- Touching anywhere on the map
- The two buttons - Navigate & Audio Nav - are kept unusable at the beginning. Until the user gives a start and a destination address using the above methods, there is no point in giving the user the option to navigate.
- The second page is the one which is only called for when the user hits ‘Audio Nav’ button.
- This page displays the text (and audio) instructions to the user once he chooses a start and an end point on the map.

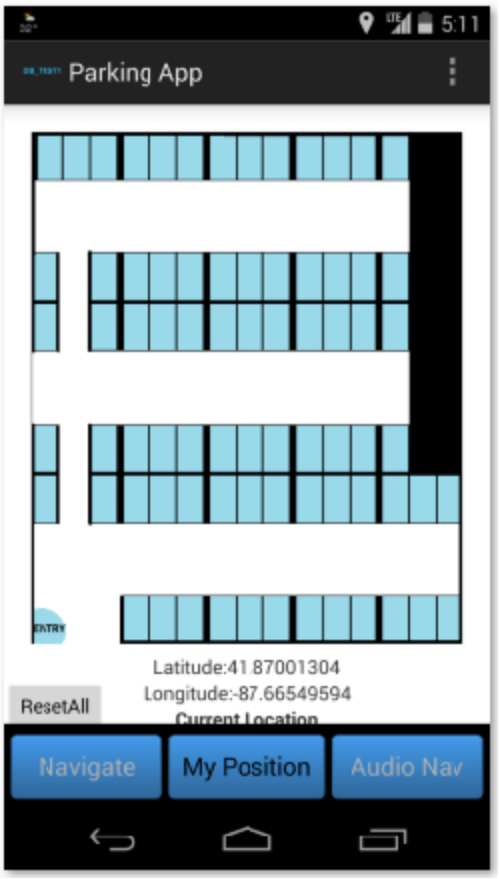

Figure 13: UI at Application start

2.6.2. Directions Screen

The directions screen consists of the textual directions of the route step-wise and also their audio versions. The screen below shows how the directions screen looks like. The screen is divided into the following components:

- Moving panel for step-by-step instructions from our start to the destination point.
- The complete directions list for the entirety of our journey.
- Reset Button - Takes us to the map view and reset the map
- DB Settings - Gives us the offline mode view of the app.
- Next Step - The user can manually go and see the next step whatever it may lead him to.

Figure 14: UI for voice navigation

2.7 NAVIGATION:

2.7.1. Route Computation

The algorithm in the app computes the path from the source to the destination by means of a vector between the two points, and then it just displays the route, which also happens to be the shortest means of getting there. Since the structure of the parking lot allows one central line serving as a backbone of all the routes, we use it to calculate most of the routes. Some of the routes that are very short and constrained to just one section of the parking lot do not require to use this central navigation plot. Two of the following images show the routes that are calculated for a different set of start point **(S)** and destination point **(D)**.

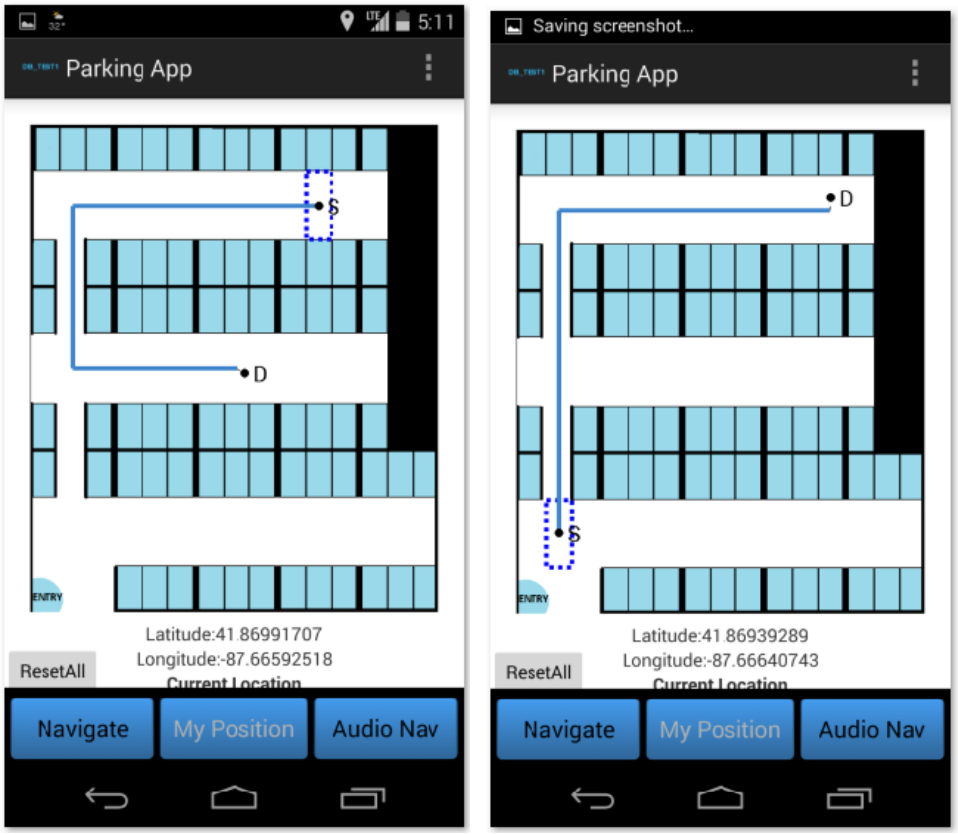

Figure 15: UI for path generation

The navigation part has been given to the user in two different forms. One where the user can listen to audio instructions and not worry about looking at the map and the other is where the user can see his/her current location in the form of a trail on the map.

2.7.2. Visual Navigation.

The visual navigation is the first of the two types of navigation offered to the user. This, as the name suggests helps the user locate himself on the map visually. Once the user hits the 'Navigate' button on the bottom left of the screen, he is able to then see a trail of his movements on the map. As his location changes from the original (source) toward the destination, so does his trail advance on the map.

The first image below shows the starting moments of the trail as we walk from our source to our destination. The wifi signal picked up along the way by phone help the application in detection the current position as we go along. This is one area which caused a few problems as the wifi signal for our parking lot varied to a large degree.

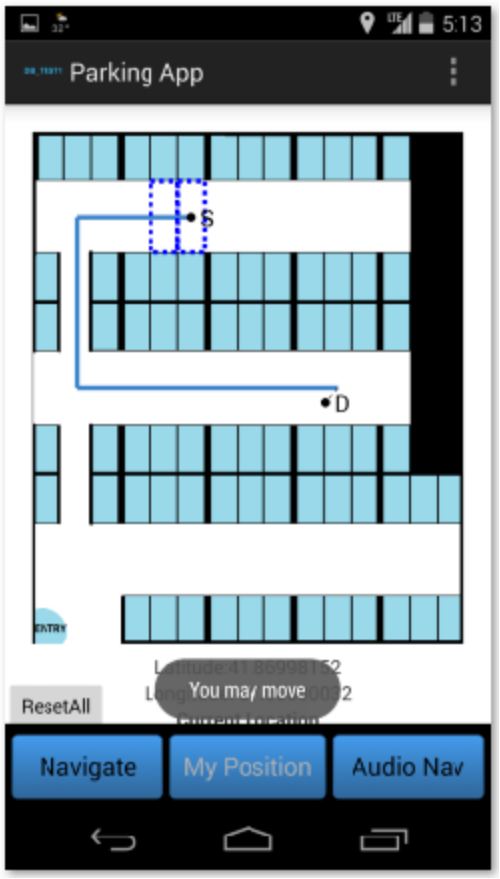

Figure 16: UI for visual navigation

The second image below describes the same journey as above, but at a later stage in time. You can clearly see that the trail has advanced in this case. The trail position tells us where we are currently and how far we have travelled on our way. The trail will terminate once we reach our destination. Given below, the trail has us placed at the destination.

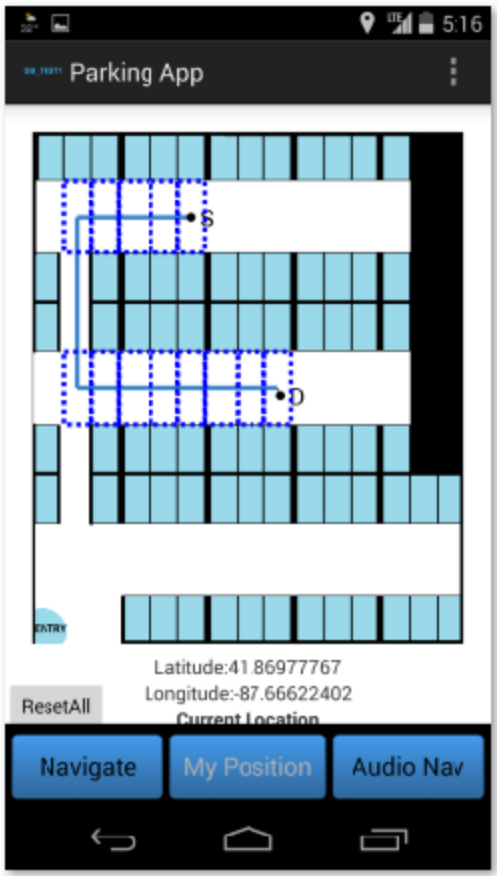

Figure 17: Path trace during navigation

2.7.3. Audio Navigation.

In this kind of navigation, we made use of Android's "TextToSpeech" feature. The system had an algorithm designed to fetch the driving instructions for the user based upon his current or starting position and the destination specified by him. This kind of hands-free navigation is what most of the current navigation apps on the market. The instructions are to be move ahead at every step and the app will inform the user when he reaches the destination.

The following image describe a particular part of a route that is displayed on the screen as well as read out by google's text to speech commands. That way the user does not have to read what the line says and can simply focus more on his driving.

Figure 18: Audio Navigation

3. STATISTICS

Paths were created accurately from source to destination at all times and there were no mapping errors.

When current position was requested, accurate cells were detected on an average 8 out of 10 times giving a localization accuracy of around 85%.

Due to the constantly changing signal strengths of available Wi-Fi due to interference, accuracy during navigation went down. We tried many different approaches in order to increase this accuracy but none of the approach was successful in getting a higher accuracy. Based on our experiments, an accuracy of 60% was observed during navigation.

4. CHALLENGES

4.1 WiFi scan interval:

Beacon frames are transmitted periodically to announce the presence of a Wireless LAN. Such Beacon frames are transmitted by the each and every Access Point (AP) at fixed interval of time and contain information such as BSSID, SSID etc.
Also every device has manufacturer specific scan interval set in 'build.prop' file (wifi.supplicant_scan_interval parameter) which can't be changed as it is set in system folder inaccessible to normal user.
The best solution would be to use the ScanResult.timestamp to determine if you should use this result or not. For example, if you're trying to get the signal strength for each access point each second, you can compare the current BSSID to previous BSSIDs. If the current BSSID was included in a scan result from the last second, you can simply ignore it.
In our experiment we found out that the wifi signals and signal strength changes only after interval of 2 seconds or so.

4.2 Error prone wifi readings:

One of the problems we faced during localization was often when at one location, with seemingly nothing changing, signal strengths would vary erratically. Wifi networks appear from nowhere with strong signals and then disappear at very next scan. There could have been several reasons for such behavior –

1. We considered 25 distinct signals where signal sources were operating at the same frequency i.e. either 2.4 GHz or 5GHz hence it is very much possible that these signals would interfere with each other.
2. Electromagnetic interference (e.g. signal source i.e. access point is located near to a bundle of unshielded wires or electric appliances such as microwave oven)
3. As this was indoor closed space it is more likely that wifi signals were reflected and diffracted from various objects such as cement walls.
4. Another reason could be moving wifi signals such as mobile hotspots. (This possibility was taken care of by taking multiple wifi readings on different days and selecting only those signals which were common across multiple readings were considered)
5. Last reason could be environmental factors as well, such as humidity, barometric pressure etc.

For instance a HAM radio transmission signal can travel across the world bouncing off the clouds, and are a relatively low frequency. In order to overcome this problem of precision GPS fingerprint of cellphone was also considered, which was demonstrated earlier.

5. CONCLUSION

The proposed indoor positioning solution allows to use already available wifi infrastructure in order to determine users location. Such solution could enhance the efficiency at a well-managed parking structure and also help in reduction of unnecessary fuel consumption. Other than parking structure there could be several applications for this method such as navigating traveler to a particular terminal so as to avoid possible flight delays.

6. BIBLIOGRAPHY & REFERENCES

The following material was helpful and a reference point for a variety of tasks that we performed:

- Android Wi Fi developer tools
  - http://developer.android.com/reference/android/net/wifi/WifiManager.html
- Android GPS documentation
  - http://developer.android.com/reference/android/location/GpsStatus.html
- Android GUI
  - http://developer.android.com/guide/topics/ui/index.html
- Android Buttons
  - http://developer.android.com/guide/topics/ui/controls/button.html
- Various Navigation Algorithms
- Youtube tutorials - Android