

INDOOR LOCALIZATION USING WIFI FINGERPRINTING

INTRODUCTION

In recent years, with the advent of commodity sector and construction of complex indoor structures a demand has been created for locating physical objects inside a building. The GPS signals received from a satellite inside a building are scattered and are not strong enough for localization. There are multiple use cases for indoor localization. I would like to list down at least a couple of them. For one such case, it is difficult for a consumer to find something specific in an indoor space. For example, 'I enter a mall, and I want to know where I am right now and where is Abercrombie and Fitch store and eventually get the directions to get there.' Second use case relates to evolving trends, their marketing and customer experience. If we could capture the exact location or even an approximate location of a person with small errors, we could use this information to provide the best deals and offers right in front of the person. Based on the accuracy of localization, there is large range of applications, which can be brought into effect.

FINAL GOAL:

I will be using the Wi-Fi fingerprinting technique to localize an individual in an indoor space. I will be using 3rd floor of GCCIS building as the indoor space of interest. The final goal is to be able to locate an individual on the map with accuracy less than 5 meters. To achieve this the only external input is the 2D map of the area of interest with accurate measurements and the coordinates of the access points.

WiFi LOCALIZATION ALGORITHM

*/ This function fetches the current wifi scan results at the spot to be localized. Compare these results with the calibrated database and receives the localized coordinates. These localized coordinates are set on the map. */

```
function localize:
  rssiDB <- fetch rssi values from calibrated database;
  current <- fetch current wifi scan results;
  filteredDB <- checkNoSignalMatch(current, rssiDB);
  */ check if filtered dataset is empty or not */
  if filteredDB.size() < 0:
    filteredDB = rssiDB;
  */find the coordinate of the most similar strength values to current */
  foreach coordinate in filteredDB:
    difference = findMatch(current, filteredDB);
    if difference < minimum:
      minimum = difference;
      localizePoints = coordinates;
  */set the coordinates on screen */
  setLocation(localizePoints)
```

```
*/ This function checks if the current strength signal values
with records in the wifi database. If both the vectors receive no
signal from the same set of access points then it returns true for that
record in the database. It returns these set of records which match as
the filtered dataset of probable matching values */
```

```
function checkNoSignalMatch(currentWiFiResults, rssiDB):
    foreach coordinate in rssiDB:
        boolean match=checkZeroAlignment(currentWiFiResults,
rssiDB.get(coordinate));
        if match:
            populate filteredDB
    return filteredDB;
```

```
function checkZeroAlignment(currentWiFiResults, rssiValues):
    for i <- 0 to currentWiFiResults.length:
        x <- currentWiFiResults.get(i)
        y <- rssiValues.get(i)
    if x == 0 || y ==0:
        if x != y:
            return false;
    return true;
```

```
function setLocation(localizePoints):
    x <- scaleX(localizePoint);
    y <- scaleY(localizePoint);
    dot.setX(x);
    dot.setY(y);
    screenLayout.addView(dot)
```

Edge Case:

If the filteredDB after No Signal Match returns an empty database, the localize function matches similarity of the current scan result with entire database.

LOCALIZATION PROCESS:

This project will be using an Android tablet to localize an individual on GCCIS 3rd floor. I have created an android application to carry out the process of calibration. The image below shows the interface of the localization client.

A. DATABASE DESIGN:

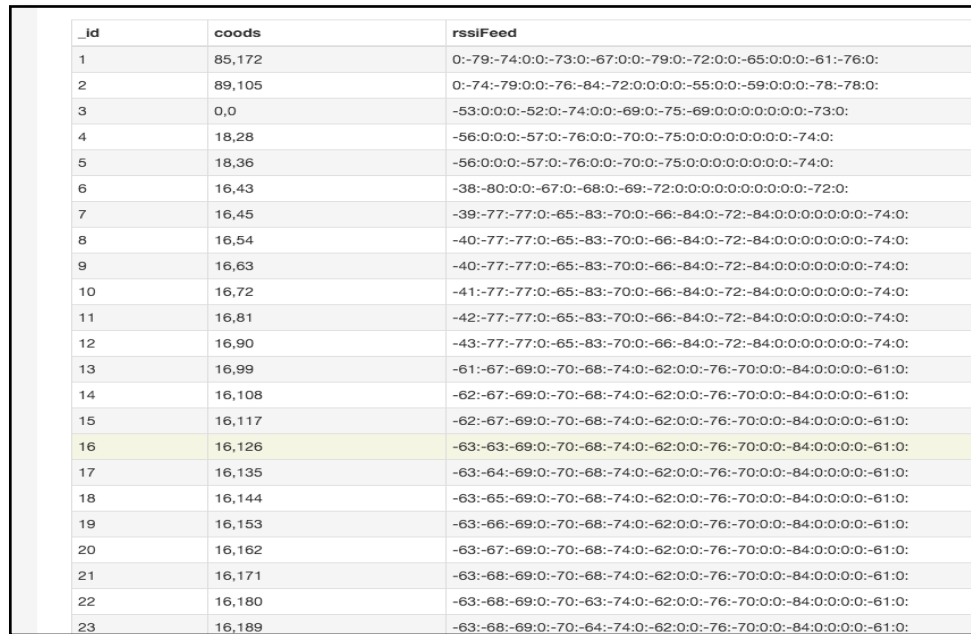
Each record is represented by the coordinates of the location and the signal strength from different access points at that location.

The database schema:

_id	Coordinates	Receiving Signal Strength Values
1	16,108	0:-79:-74:0:0:-73:0:-67:0:0:-79:0
2	85,172	-62:-67:-69:0:-70:-68:-74:0:-62:0:0:-76

The receiving signal strength values at a particular coordinate are not stored in random order. Each index represents the strength from a particular access point. Maintaining the order of strength values forms the basis of my implementation. This database is stored locally on the device.

Screenshot of the actual SQLite database:



_id	coords	rssiFeed
1	85,172	0:-79:-74:0:0:-73:0:-67:0:0:-79:0:-72:0:0:-65:0:0:0:-61:-76:0:
2	89,105	0:-74:-79:0:0:-76:-84:-72:0:0:0:-55:0:0:-59:0:0:0:-78:-78:0:
3	0,0	-53:0:0:0:-52:0:-74:0:0:-69:0:-75:-69:0:0:0:0:0:0:-73:0:
4	18,28	-56:0:0:0:-57:0:-76:0:0:-70:0:-75:0:0:0:0:0:0:-74:0:
5	18,36	-56:0:0:0:-57:0:-76:0:0:-70:0:-75:0:0:0:0:0:0:-74:0:
6	16,43	-38:-80:0:0:-67:0:-68:0:-69:-72:0:0:0:0:0:0:0:0:-72:0:
7	16,45	-39:-77:-77:0:-65:-83:-70:0:-66:-84:0:-72:-84:0:0:0:0:0:-74:0:
8	16,54	-40:-77:-77:0:-65:-83:-70:0:-66:-84:0:-72:-84:0:0:0:0:0:-74:0:
9	16,63	-40:-77:-77:0:-65:-83:-70:0:-66:-84:0:-72:-84:0:0:0:0:0:-74:0:
10	16,72	-41:-77:-77:0:-65:-83:-70:0:-66:-84:0:-72:-84:0:0:0:0:0:-74:0:
11	16,81	-42:-77:-77:0:-65:-83:-70:0:-66:-84:0:-72:-84:0:0:0:0:0:-74:0:
12	16,90	-43:-77:-77:0:-65:-83:-70:0:-66:-84:0:-72:-84:0:0:0:0:0:-74:0:
13	16,99	-61:-67:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
14	16,108	-62:-67:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
15	16,117	-62:-67:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
16	16,126	-63:-63:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
17	16,135	-63:-64:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
18	16,144	-63:-65:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
19	16,153	-63:-66:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
20	16,162	-63:-67:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
21	16,171	-63:-68:-69:0:-70:-68:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
22	16,180	-63:-68:-69:0:-70:-63:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:
23	16,189	-63:-68:-69:0:-70:-64:-74:0:-62:0:0:-76:-70:0:0:-84:0:0:0:-61:0:

Fig. 1. SQLite Database View of wifi database calibration

My second choice of database is Firebase platform. It allows me to push the calibrated data over the network using REST API to external online database. Firebase is powerful platform to build mobile application.

The advantages of using Firebase:

1. Can be used with other devices without the need of exporting the local database.
2. Server could perform operations on this database without causing hindrance to the functioning of localization client.

Here is snapshot of how the data is stored in the firebase.

```

torrid-torch-2612
00229092d32: 6,1)->-57 (6,3)->-53 (6,5)->-45 (6,7)->-44 (6,...
00229092d38: "(6,1)->0 (6,3)->0 (6,5)->-88 (6,7)->-88 (6,9)->..
00229092d59: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092d9e: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092da8: "(6,1)->-52 (6,3)->-53 (6,5)->-61 (6,7)->-61 (6..
00229092df5: "(6,1)->0 (6,3)->0 (6,5)->-80 (6,7)->-80 (6,9)->..
00229092eaf: "(6,1)->-77 (6,3)->-75 (6,5)->-72 (6,7)->-72 (6..
00229092eb9: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092f0a: "(6,1)->-81 (6,3)->-79 (6,5)->-74 (6,7)->-74 (6..
00229092f40: "(6,1)->-66 (6,3)->-70 (6,5)->-70 (6,7)->-70 (6..
00229092f4d: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092fd2: "(6,1)->-79 (6,3)->-79 (6,5)->-77 (6,7)->-77 (6..
00229092fda: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092fdf: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092fe0: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
00229092fe7: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..
0022909301c: "(6,1)->0 (6,3)->0 (6,5)->0 (6,7)->0 (6,9)->0 (6..

```

Fig. 2. Firebase view of wifi calibration data

The key '00229092d32' is my way of representing the address of the access point. Each key has the strengths at the corresponding coordinates. The key and value in this database may change as I proceed with the final phase of my implementation.

B. APPLICATION INTERFACE

Indoor Localization

X coordinate: 85

Y coordinate: 172

Increment X: 0

Increment Y: 9

0, -79, -74, 0, 0, -73, 0, -67, 0, 0, -79, 0, -72, 0, 0, -65, 0, 0, 0, -61, -76, 0

POLL MAP

Fig. 3 Application Interface for calibration

C. PROCESS:

I will briefly list down the steps used to calculate the location of a person.

a. Calibration:**1. Steps:**

- In the x and y fields, enter the coordinates of the point where the user is standing.
- In the auto-increment fields, enter the values at which the user would like to record the wifi signal strength values.
- Poll button when pressed checks all the wifi signals present at that location and accordingly picks up the strength of the access points on the 3rd floor of GCCIS). This data is entered with the coordinates of that location and its wifi signal strength values. If the user records the strength for a duplicate location, the average of the current and existing values is stored as the updated value

2. Calibration Results:

This stage was the most time consuming phase of the entire project. Multiple iterations were required to achieve good calibration results of the GCCIS third floor. I divided calibration phase into three tasks:

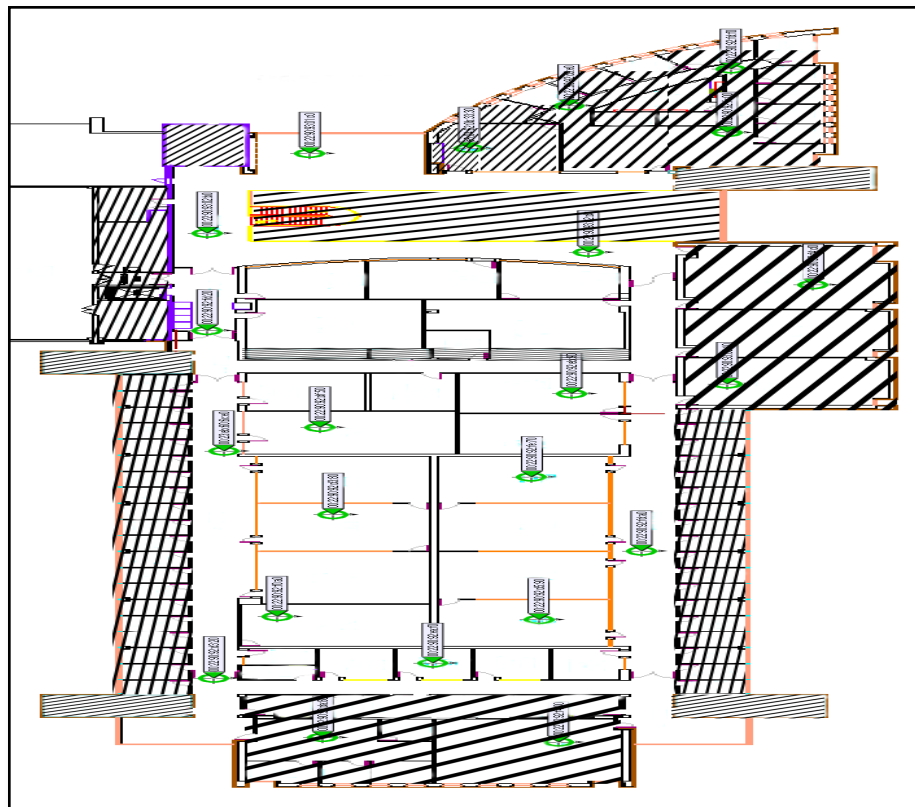
a. Marking the obstacles.

Fig. 4 Map of the 3rd Floor of GCCIS with obstacles

- b. Marking the coordinates of the floor.

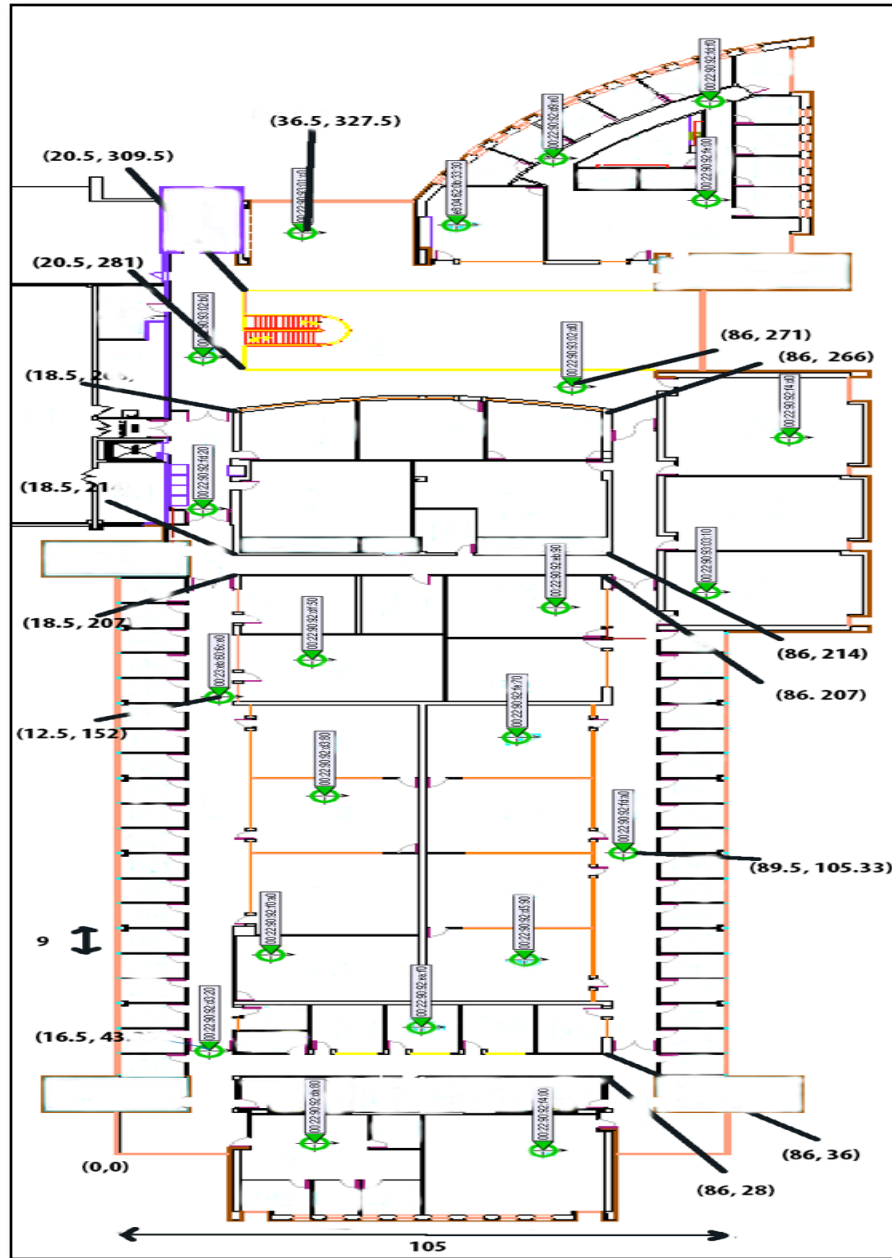


Fig. 5 Map of the 3rd Floor of GCCIS with precise measurements (in feet)

- c. Scaling it to the android device screen.

To display the location on the android screen we have to set the x and y coordinates of location inferred from my calculation. Since the coordinates of location received are in feet and the units of display on the screen is in pixels, we need to scale the map according to calibration. The white space on the

borders of the map needs to be added as margins to the feet to pixel conversion.

Specification	Width	Length
Android Screen	1440	1840
GCCIS 3 rd Floor	105	342

b. Localization:

The map does have a localize button to calculate the location of an individual. When pressed,

1. records the current wifi signal strengths and perform a no signal checking initially. This returns a set of coordinates which have no signal strength from same set of access points. This narrows down the possibility of the location of the user.
2. From this set of results, the system performs similarity check on them. Currently the system picks up the closest match and returns the corresponding coordinates.
3. These coordinates are scaled to pixels and displayed on the screen as a blue dot.

Localization Results:

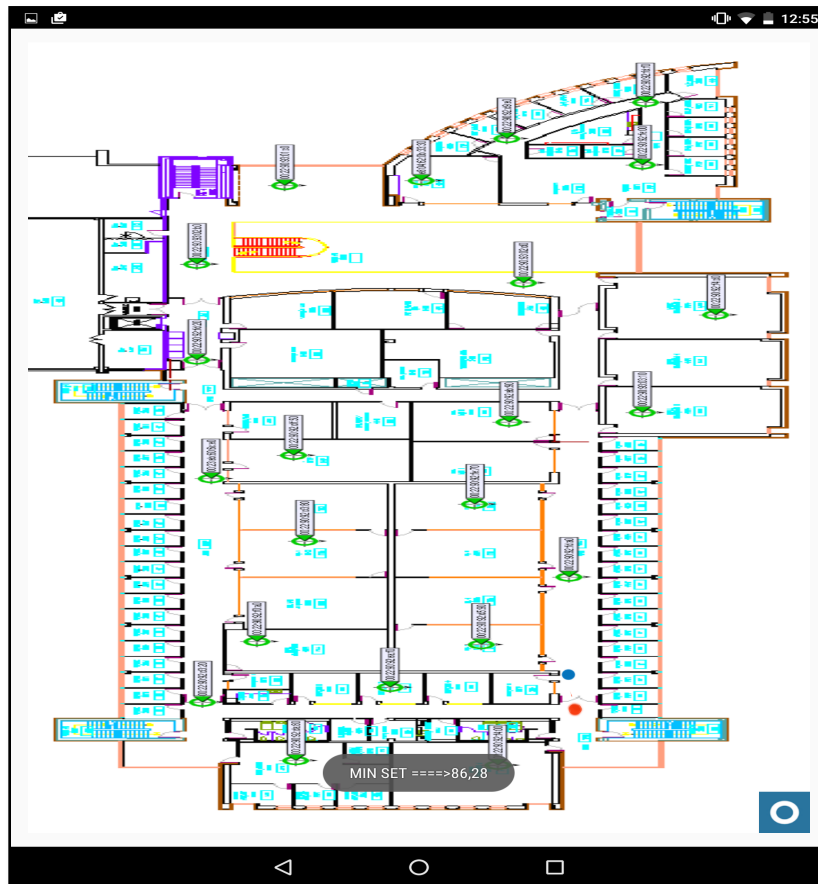


Fig. 6 Accuracy of 4 meters

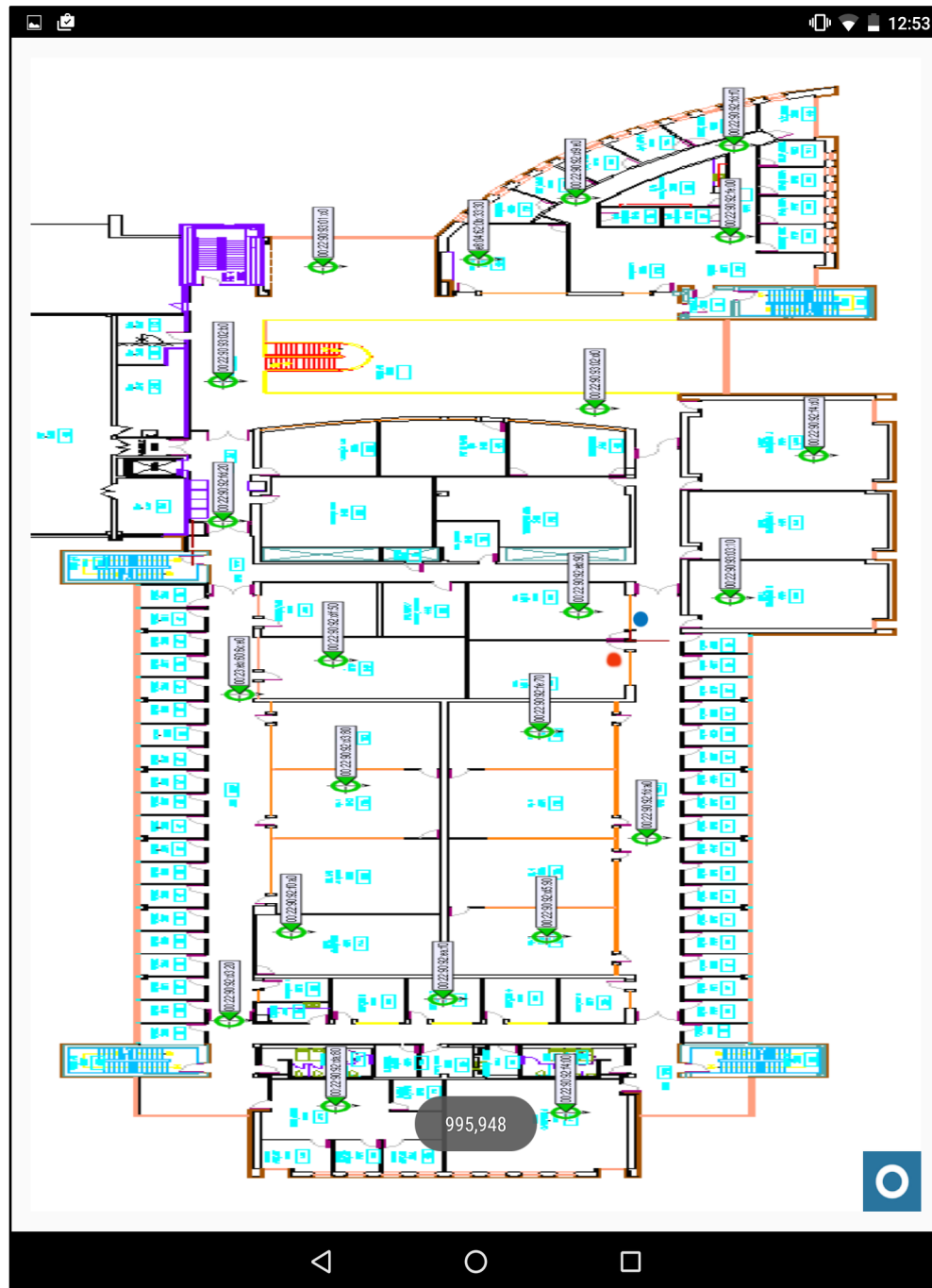


Fig. 7 Accuracy of 3 meters

The red dot on the map is the actual location of person with the device. The blue dot is the calculated location using WiFi signal strength. I have included two screenshots of the working of my application.

D. ISSUES:

1. The biggest assumption is that the MAC addresses received from ITS are precise.
2. Scaling the coordinates to pixels and moving the blue dot on the android screen is still an issue as it is not exactly precise.
3. During testing I have noticed, sometimes the wifi needs to be disconnected and connected back to reflect the changes. I relate this issue to the wifi driver of the device.
4. Due to changes in database, I had to re-create the database and calibrate it multiple times. The previous calibrations were rendered useless.

E. IMPROVEMENTS NEEDED:

I would like to list down the steps I would be undertaking to make the localization readings obtained more consistent:

1. Currently the database has 250 entries. I would like to increase the number of records to at least 1000.
2. Currently I choose coordinate of the closest matching strength values from the database. I will be working towards collecting a set of closely similar signal strength values and calculate the location from those set of signal strength values.

EXPERIMENT RESULTS

To check the accuracy of the localization results, I calibrated a specific corridor on the floor at intervals of 3 feet and 1 foot along x-axis and y-axis. The zoomed in map shows the corridor in red intersecting lines with their coordinates. I have included the readings and the localization errors taken at different points in that stretch.

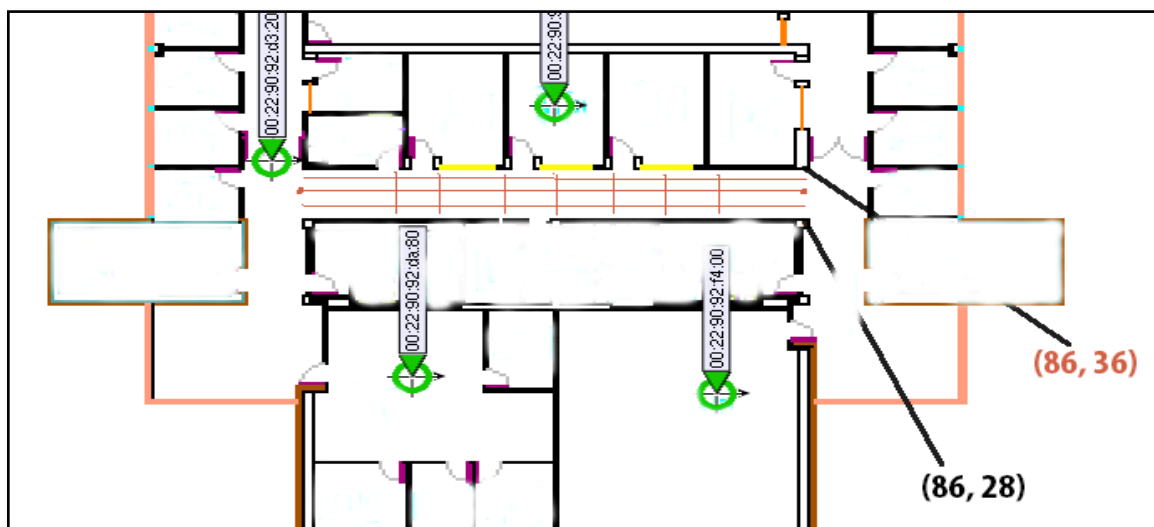


Fig. 8 Localization Experiment in a corridor.

Reading No.	Actual Location (x, y)	Localization Result (x, y)	Error
1	(86, 28)	(85, 32)	4.123105626
2	(80, 28)	(85, 36)	9.433981132
3	(74, 28)	(77, 32)	5
4	(68, 28)	(68, 30)	2
5	(62, 28)	(65, 32)	5
6	(56, 28)	(59, 33)	5.830951895
7	(50, 28)	(42, 32)	8.94427191
8	(44, 28)	(39, 36)	9.433981132
9	(38, 28)	(39, 36)	8.062257748
10	(32, 28)	(39, 32)	8.062257748
11	(26, 28)	(18, 32)	8.94427191
12	(20, 28)	(18, 43)	15.13274595

Average error (in meters) = 2.285

Future Work:

- Refine the calibration process for further refinement
- Test localization and reduce the errors.
- Additional scenarios can be thought of for narrowing down to accurate location incase of a very large dataset.

PERSONAL ASSESSMENT:

This phase included considerable implementation, lot of calibration and testing. I have gained significant insight on indoor localization using WiFi. To achieve consistent results with high accuracy, WiFi signal strength alone is not a great candidate. Combining gyroscope, accelerometer and stride length with WiFi signal strength could help achieve much better solution.

CODE BASE:

<https://github.com/chiransgit/localize>
<https://gitlab.com/kwonm71/indoorlocal>

REFERENCES:

Locating in fingerprint space: wireless indoor localization with little human intervention.

<http://www.cse.ust.hk/~liu/LIFS.pdf>