Indoor positionning

Louis MENNRATH, Maxence ZHUANG

PHY7509

1 Introduction

1.1 Abstract

The aim of this project is to develop a system of indoor positioning based on Wi-Fi signal power levels. We will first use the fingerprinting map technique, followed by enhancing the system with sensor fusion techniques.

1.2 Fingerprint map

Fingerprinting map is a technique used to measure the power level of a WiFi access point relative to the user's position.

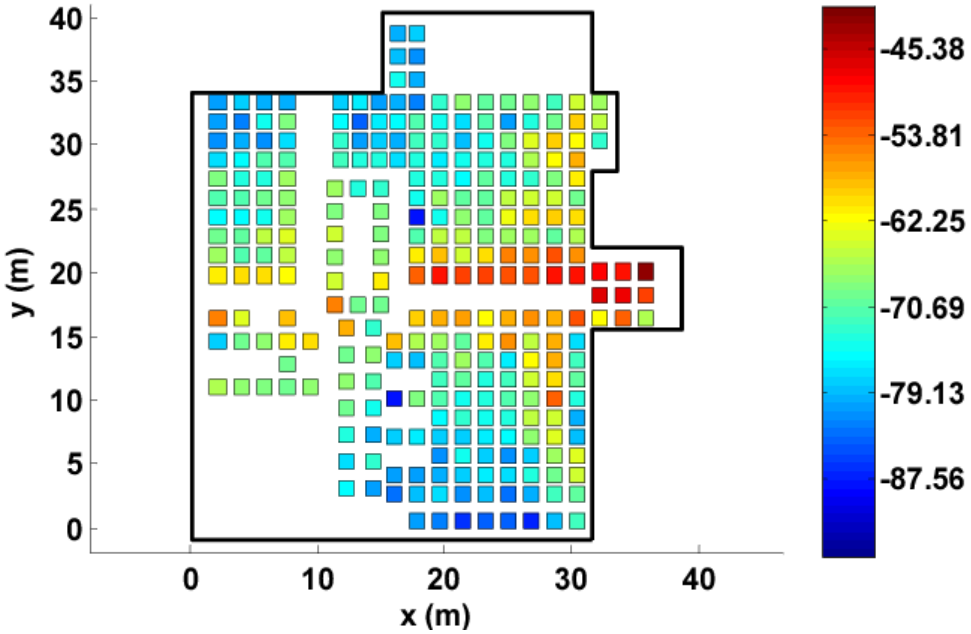

Figure 1: exemple of a fingerprinting map

The different point of the map will display the **Relevant Signal Strength Indicator (RSSI)** that is a measure of the power level of Wi-Fi signals.

1.3 Tools

To measure the RSSI of different Wi-Fi signals, we will use an ESP-32 which is a low power SoC widely use fot IoT applications because of its capabilities to receive and send Wi-Fi/Bluetooth signals.

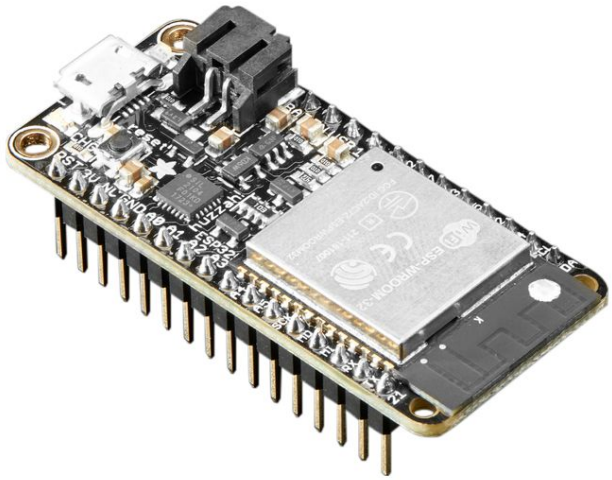

Figure 2: ESP-32 wroom Adafruit

The data are collected using an Arduino program and exported as a .csv file to be further exploited with Python programs.

The data from the accelerometer will be gathered with the application *Physics Toolbox Sensor Suite* on an iPhone SE 3.

2 Measurement

2.1 Methodology

Only the most recent version of the ESP-32 can detect 2.4 GHz and 5 GHz WiFi signals (ESP32-C5). Therefore, we chose to limit our study on 2.4 GHz signals because of the lower wall attenuation and the further range that let us detect more WiFi access points.

We concentrate our data gathering on the third floor of the A building but we could eventually extend the fingerprinting map to the whole campus if the results are satisfying. The map is separated in different zones:

- **The rooms** : Each room is separate into 4 zones, except for the A306 where the materiel inside impose only two zones.
- **The corridor** : The corridor is divided into seven positions.
- **The offices** : We don't have access to professors office and to the meeting room so we didn't include them in the map.

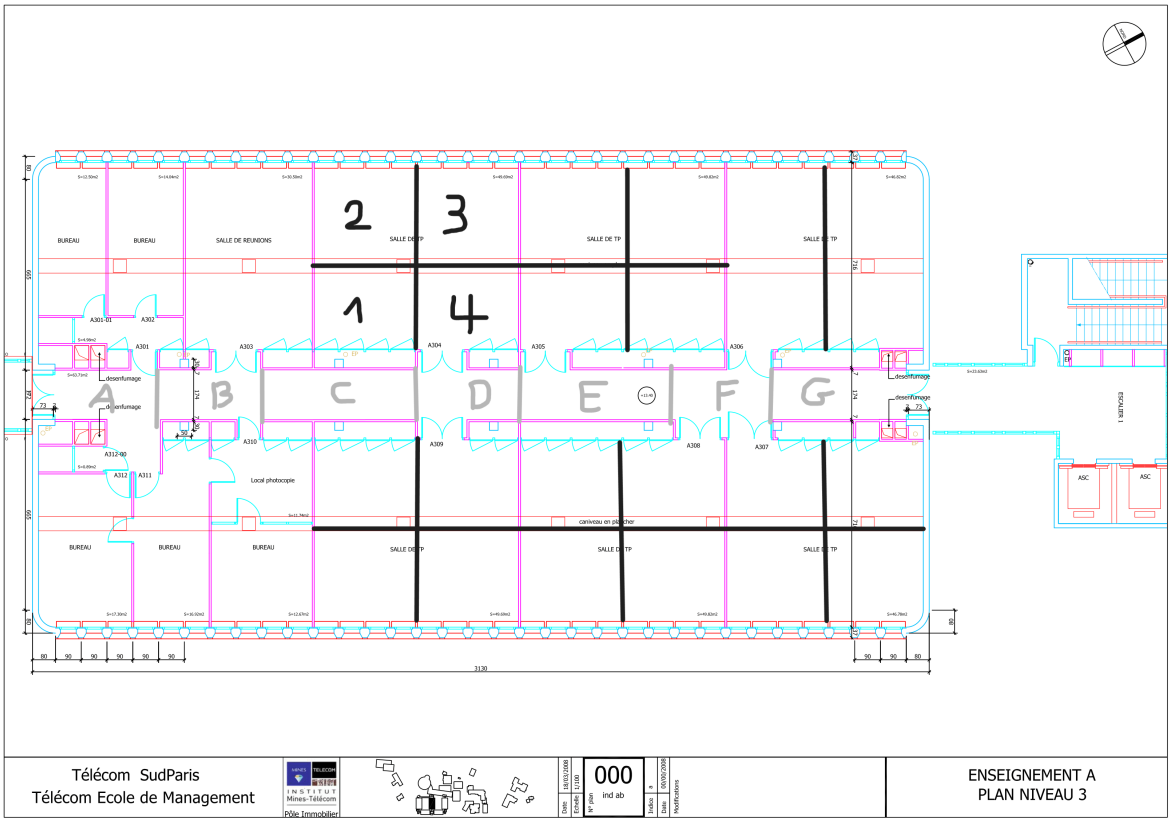

Figure 3: 3rd floor zone distribution

2.2 Data gathering

In each zone we measure all the signal we can find for a duration of 30s second and then we need to do the average of these measure to have a mean value for each access point represented by its own MAC address.

The room door is closed during the measurements for the map, but the door's position is not critical in our case. We conducted tests and found that, due to the distribution of access points and our choice to work with 2.4 GHz signals, which has lower attenuation, we have many different access points that penetrate the walls/doors. The door only affects one or two signals by 2 to 3 dBm, while the others remain undisturbed, allowing us to accurately identify the room. Moreover, since the final results are average values, the door's position (open/closed) has already been accounted for, as people were entering and exiting the room during the measurements.

To calculate the average for 30 second we need to convert the dBm into mW following the formula:

$$P_{(mW)} = 10^{\frac{P_{(dBm)}}{10}}.mW$$

Then, we reconvert the value back into dBm. All the value are sent in a .csv file:

MAC	P(dBm)	Position
50:06:04:66:71:F1	-56,46	A304(1)
20:BB:C0:4B:94:01	-64,06	A304(1)
50:06:04:66:91:D1	-69,84	A304(1)
50:06:04:5B:45:E1	-90,97	A304(1)
CC:16:7E:15:F2:B0	-90	A304(1)
50:06:04:5B:44:E1	-91	A304(1)
20:BB:C0:4B:9F:81	-89,55	A304(1)
20:BB:C0:4B:D7:21	-95	A304(1)
20:BB:C0:4B:E5:C1	-90	A304(1)

Table 1: Exemple of the data for the position A301(1)

3 The algorithm

To process the data and determine in which room we are, we developed a python program using **Matplotlib** and **Pandas** libraries.

3.1 Creation of the Dataframes

To proceed to the positioning calculation, we need to convert our data in a mathematical form. We use Pandas to create **Dataframes** that are tabular objects to work with.

We have 2 types of Dataframe, the fingerprinting map and some measurement points. The rooms fingerprinting map is a Dataframe of **25 rows** and **24 columns** where each column correspond to one zone and each row to a WiFi access point represented by its MAC address.

$$\mathbf{map} = \begin{bmatrix} MAC & A301(1) & A201(2) & \cdots & A309(4) \\ 20:BB:C0:4B:94:01 & -64.06 & -58.51 & \cdots & -70.45 \\ 50:06:04:5B:45:E1 & -90.97 & -92.36 & \cdots & -100.00 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CC:16:7E:15:F2:B0 & -90.00 & -100.00 & \cdots & -100.00 \end{bmatrix}$$

The MAC addresses are ordered in alphanumerical order and acts as the coordinate of our vectors. For all the zones where the access point were out of range, we assign the value of the coordinate to **-100 dBm** which is the minimum threshold detection value of the ESP-32.

Further in the algorithm, once all the MAC addresses of the map and the measured point are in the same order we drop the MAC address column of the Dataframe and we identify each column by its name, the **map** Dataframe look like that:

$$\mathbf{map} = \begin{bmatrix} -64.06 & -58.51 & \cdots & -70.45 \\ -90.97 & -92.36 & \cdots & -100.00 \\ \vdots & \vdots & \vdots & \vdots \\ -90.00 & -100.00 & \cdots & -100.00 \end{bmatrix}$$

and have a dimension of 24 rows and 23 columns.

For the measured vectors, it is the same process, we have 24 rows but only 2 columns the MAC and the power level. After filling the empty cells with the -100 dBm value and dropping the MAC columns the samples measured are of the following shape:

$$\mathbf{X} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{23} \end{bmatrix}$$

3.2 Distance measurement

We have now a map matrix: $map = (m_{ij})$ with $(i, j) \in [0; 22] * [0; 23]$ and the X vector and we can calculate the distance between the measured vector X and each position vector of the map by using the Euclidean distance formula:

$$\sqrt{(m_{i1} - x_1)^2 + (m_{i2} - x_2)^2 + \cdots + (m_{in} - x_n)^2}$$

The algorithm return the 3 nearest zone from the sample vector in a tuple.

3.3 Result interpretation

The algorithm returns the three closest zones to the sample vector, and we need to determine which room we are in. Since there are four zones (or two for A306), the algorithm is more precise in detecting a room, but it only returns zone names. Therefore, we need to convert the result into a prediction of the correct room. To achieve this, we developed a decision table based on the algorithm's output.

There are 3 possible value for the output tuple: A, B, C

Output	Decision	Trust Value
(A, A, A)	A	1
(A, A, B)	A	2
(A, B, A)	A	2
(A, B, C)	A	3
(A, B, B)	A	3

Table 2: Table decision of room decision

The first index of the output tuple have more weight because it is the closest position from the measured vector.

During tests only the 2 first outputs appears.

3.4 Testing

We conducted a series of tests, and the room was correctly identified in **100% of cases**. This accuracy is due to dividing the rooms into subsections. While the identification of

these subsections is only **80% accurate**, any misidentifications still remain within the same room. Thus, our system is highly precise in detecting which room the measurement is taken in.

We run the same tests for the corridor but with only 7 zones as described in the Figure 3. The algorithm detect the correct zone only with a **90% precision**. We could divide these spaces into smaller sections, but we don't want to take additional measurements to record a complete and precise trajectory, as it might generate too much data for the ESP-32 to process, potentially affecting accuracy. We can certainly find an alternative solution.

4 Sensor Fusion

To improve our system, we chose to add an accelerometer to detect when we are moving and when we are stationary. When moving, more access points can be detected than in a static position, which can lead to confusion between two zones. Therefore, we need to ensure we remain in one zone before interpreting the data.

At first we run a few calibration tests when we were moving and stading still to determine a threshold in the accelerometer z axis to determine when we are walking between 2 different rooms.

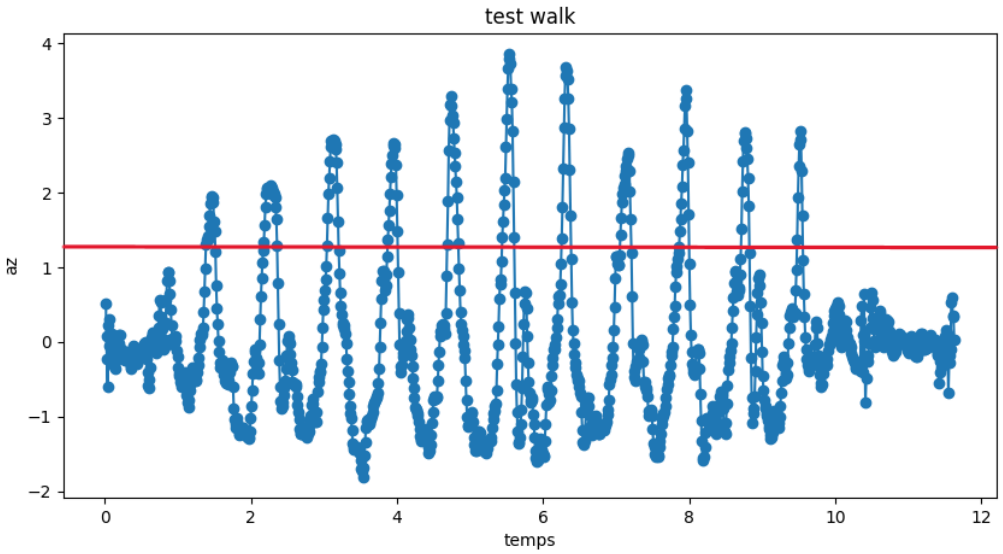

Figure 4: Threshold determination

When the accelerometer give a value under the threshold during a period longer than 5 sec we can consider that we stopped and we will take into account the values sent by the ESP-32.

It is important that the measures of the RSSI and the accelerometer are **synchronised** to be able to take into account the RSSI only when we are stationnary.

The ESP-32 took measures every second so we need to stay for about 5 sec in a zone to correctly detect it.

Supposing that the accelerometer and the ESP-32 data are synchronised, we can launch our algorithm of room detection when the accelerometer indicates that we are not moving. If no room is detected, we compare the point to the corridor map. We can obtain the detail of the path on the area.

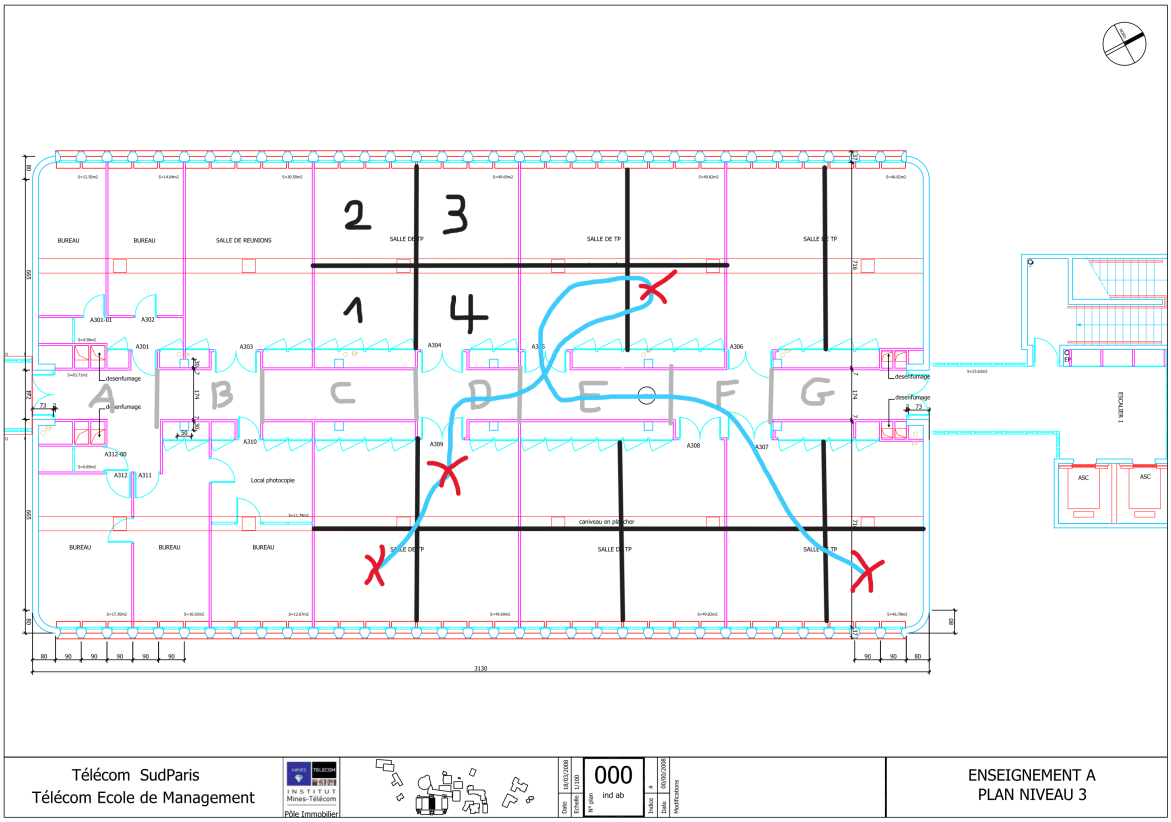

Figure 5: Progression path

5 Conclusion

5.1 Feedback

The fingerprint map allows us to detect a room with a 100% accuracy and the result in the corridor are also very satisfying, we are able to identify the position of the ESP-32 in the 3rd floor of the A building.

With the help of sensor fusion, we were able to detect precisely the path of a moving target which travels between rooms.

5.2 Improvements

We can extend the fingerprinting mapping to the entire building and even to the campus. To do this, we will need to improve the speed of measurement. With our current technique and no optimization, we can perfectly identify a room. However, our mapping grid might be overdimensioned, so we can potentially reduce the number of zones in each room.

Some steps of the algorithm are currently done manually because we haven't had time to fully develop it. However, in the future, we could aim to make the tracking process completely autonomous.

The fingerprinting maps were created using an ESP-32, so we need to continue using it for location tracking. We are only able to locate the ESP-32 itself. We need to conduct tests to determine if the same map works with a mobile device or if we need to create an offset to compensate for the differences between antennas.

6 Annexes

A Code

```
1  import pandas as pd
2  import math
3
4  data = 'map.csv'
5
6  map = pd.read_csv(data)
7  print(map)
8  macs = list(map['MAC'])
9  print(macs)
10 map = map.drop(map.columns[0], axis=1) # We take only the data columns,
       we drop the MAC adresses
11 map = map.astype(float) # data are originally Strings and we convert
       them to floats
12 columns = list(map.columns)
13
14 def nearestPosition (test, map):
15     dist = []
16     for pos in list(map.columns):
17         diff = []
18         for i in range(map.shape[0]):
19             diff.append((float(test[i]) - float(map[pos][i]))**2)
20             i=i+1
21         dist.append(math.sqrt(sum(diff)))
22     p1 = dist.index(min(dist))
23     dist.pop(p1)
24     p2 = dist.index(min(dist))
```

```
25          dist.pop(p2)
26          p3 = dist.index(min(dist))
27          return (p1, p2, p3)
28
29
30
31
32
33  randomTest = '2' # change the value to test an other random point
        between 1 and 6
34
35  pos = (columns[nearestPosition(samples[randomTest], map)[0]], columns[
        nearestPosition(samples[randomTest], map)[1]], columns[
        nearestPosition(samples[randomTest], map)[2]])
36
37  print(pos)
```

B Test Data

Test Vector	Position
1	A308(1)
2	A308(3)
3	A307(1)
4	A307(3)
5	A304(3)
6	A304(1)

Table 3: Position of tests data