**Report of TD 3 FingerPrint**

Positioning Systems techniques and Applications Master 1,

International Master 1 -Internet of Things

The University of Franche-Comté

### By: Uendi MUCA and Omar KHALIL

### Submit To: Dr. Phillipe CANALDA

### Date:27.03.2025

Colorful lines on a black background

AI-generated content may be incorrect.A logo for a school

AI-generated content may be incorrect.

A black and red text on a white background

AI-generated content may be incorrect.

Table of Contents

1.Introduction to FingerPrint

2.Introduction to K-nearest

3.Problem Introduction

4.Mathematical problem

5.Conceptualization of the Problem

6.Architecture of program

7.Class Diagram

8.Data Structure

9.Pseudo-Code

10.Algorithm

11.Mathematical Calculations

12.Comparison of code results and theoretically computed results

13.Conclusions

### **Introduction to Fingerprint-Based Positioning**

Fingerprint-based positioning is a **localization method** used to estimate the coordinates of an unknown point (mobile device) based on **pre-collected signal measurements** (RSSI values) from known reference locations. This technique is widely used in **indoor positioning** where traditional GPS signals are weak or unavailable, such as in malls, airports, or underground facilities.

Unlike lateration methods (such as **N-Lateration**), which rely on **geometric distance calculations**, fingerprinting is based on **pattern matching** by comparing real-time received signals to a pre-recorded database of signal strengths from different access points.

Fingerprint-based localization operates in two main phases:

#### **1. Offline Phase (Fingerprint Collection)**

* 1. A **radio map** is created by measuring **RSSI (Received Signal Strength Indicator)** from multiple known reference locations (cells).
  2. The fingerprint database is stored, associating each reference position with the **RSSI vector** from multiple transmitters (WiFi APs, Bluetooth beacons, etc.).
  3. The recorded dataset consists of: Reference Location(xi,yi)→RSSI Vector

**2. Online Phase (Position Estimation)**

* The mobile device measures real-time **RSSI values** from available reference signals.
* The system compares this new measurement with the stored fingerprint database and selects **k-nearest reference points** based on RSSI similarity.
* The estimated position is computed using a **weighted averaging method**, where reference points with closer RSSI values contribute more to the final position.x

# 

# **Introduction to K-Nearest Neighboring**

The **K-Nearest Neighbor (KNN) approach** in fingerprint-based localization is a method used to estimate the position of a mobile terminal by comparing its **Received Signal Strength Indicator (RSSI)** values with a predefined database of reference points. In this method, a set of locations with known coordinates (reference points) and their associated RSSI values are stored. When a mobile device measures its RSSI values in real-time, the algorithm **computes the difference** between the measured RSSI and the stored RSSI values for each reference point.

The algorithm then **sorts the reference points** based on their similarity to the measured RSSI and selects the **k-nearest** reference points that have the smallest RSSI difference. These selected points contribute to estimating the mobile device's position. The final estimated position is obtained using a **weighted average**, where closer reference points (with lower RSSI differences) have higher weights, ensuring a more precise localization.

By using **KNN with RSSI-based weighting**, the algorithm provides a robust way to estimate the mobile terminal’s position while considering multiple signal sources, minimizing errors caused by fluctuations in signal strength.

# 

# **Problem Introduction**

The goal of this code is to **estimate the position** of a **mobile terminal (MT)** within a predefined **square area** by utilizing a set of **reference points (cells)** with known locations and their recorded **Received Signal Strength Indicator (RSSI) values** from multiple transmitters.

### 

### 

### 

### 

### **Mathematical Model**

Instead of solving distance-based nonlinear equations like **N-Lateration**, fingerprinting uses an **RSSI-based similarity approach**:

1. **Find k-Nearest Neighbors:**
   * Compute the absolute RSSI difference for each reference position: di​=∑∣ri,j​−rtm​∣
   * where:
     + ri,j​ is the stored RSSI from reference point i for transmitter j.
     + rtm is the RSSI data of terminal mobile.
     + di​ represents the RSSI difference between the mobile device and reference point i.
2. **Weight Calculation (Inverse Distance Weighting - IDW):**
   * Assign weights based on inverse RSSI difference:wi​=d1/di​, for di​ != 0
   * Normalize weights: wi′​=wi/∑wi​
3. **Estimate the Mobile Position:**
   * Compute the weighted average of k-nearest reference points: x^=∑wi′​xi​ ,y^​=∑wi′​yi​

* Where the coordinates of the reference positions.

## **Conceptualization of the Problem**

A drawing of a building

AI-generated content may be incorrect.The **Fingerprinting-based positioning algorithm** is used to estimate the position of a receiver (**mobile terminal**) based on **RSSI (Received Signal Strength Indicator) values** from multiple **reference points (WiFi APs, Bluetooth Beacons, or known locations)**.

Unlike **N-Lateration**, which relies on **distance measurements**, **fingerprinting** compares the **RSSI patterns** of a mobile device to a **pre-recorded fingerprint database** and estimates the **most likely position** using RSSI similarity.

Process of RSSI-based Fingerprint Positioning:

1. **Offline Phase:** A database of **RSSI values** is collected at known reference positions (cells).
2. **Online Phase:** The mobile device records **live RSSI values** and compares them to the **pre-recorded database** to estimate the position.
3. **Position Estimation:** The system finds the **k-nearest reference points** (RSSI similarity) and calculates the estimated position using **weighted averaging**.

Data set input:

|  |  |  |
| --- | --- | --- |
| Cell i | Reference Location (x, y) (meters) | RSSI Vector [r1​,r2​,r3​,r4​] |
| 0 | (0,0) | [-38, -27, -54, -13] |
| 1 | (4,0) | [-74, -62, -48, -33] |
| 2 | (8,0) | [-13, -28, -12, -40] |
| 3 | (0,4) | [-34, -27, -38, -41] |
| 4 | (4,4) | [-46, -48, -72, -35] |
| 5 | (8,4) | [-45, -37, -20, -15] |
| 6 | (0,8) | [-17, -50, -44, -33] |
| 7 | (4,8) | [-27, -28, -32, -45] |
| 8 | (8,8) | [-30, -20, -60, -40] |

The **mobile terminal (MT)** is located at an **unknown position** and measures the following **RSSI values**:

TM RSSI=[−26,−42,−13,−46]

**Architecture and program flow**

The purpose of this program is to estimate the position of a mobile terminal (receiver) based on the **Received Signal Strength Indicator (RSSI)** values from multiple reference points using **RSSI-based fingerprinting**. The program is implemented in Python and follows a structured approach for data processing, selection of k-nearest reference points, and weighted positioning estimation.

### **Requirements:**

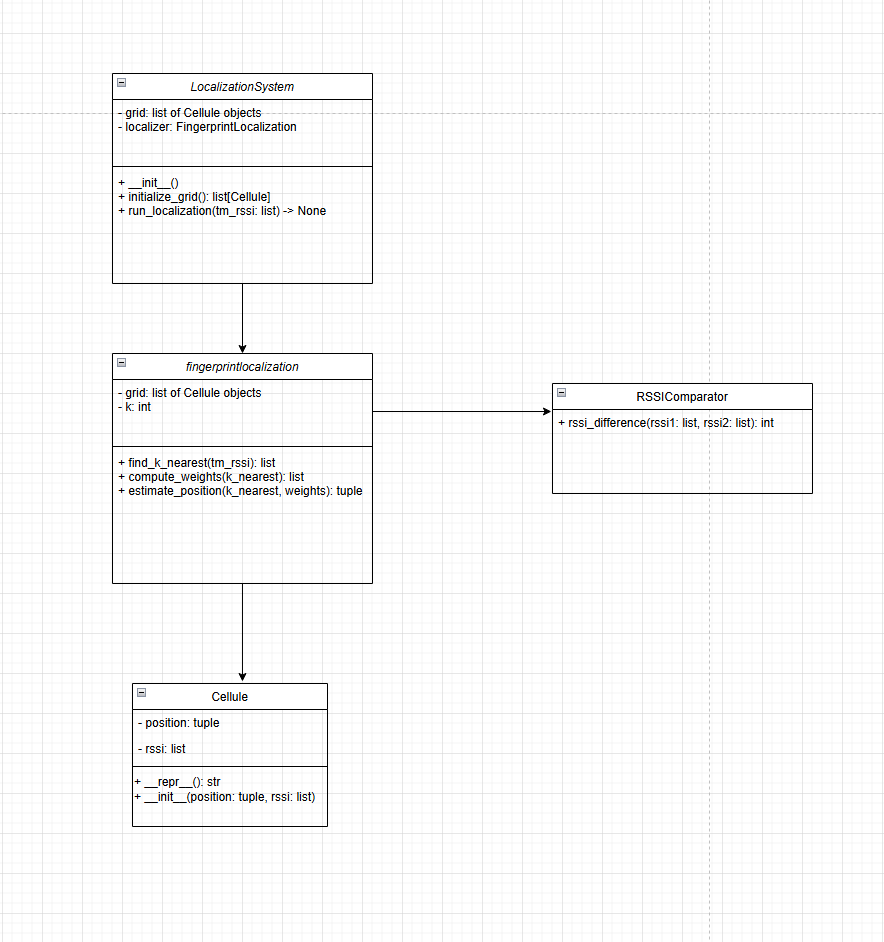
* **Programming Language:** Python 3 or above.
* **Libraries:** NumPy for numerical computations and data handling.
* **IDEs:** VS Code is used as the primary development environment, but any Python-compatible IDE can be used.

### **Design of the Program**

The program consists of three main components:

1. **Cellule Class (Reference Points Storage)**
   * Stores the **(x, y) coordinates** of each reference point.
   * Stores the associated **RSSI values** for that location.
   * Provides a method to access RSSI values for comparison.
2. **RSSI-Based Fingerprinting Algorithm**
   * The **fingerprint\_localization function** compares the real-time RSSI values from the mobile terminal with stored RSSI values in the reference database.
   * It computes the **absolute difference** between the RSSI vectors of the mobile terminal and each reference point.
   * The reference points are **sorted by similarity**, and the **k-nearest points** with the lowest RSSI difference are selected.
3. **Weighted Position Estimation**
   * The selected **k-nearest points** contribute to estimating the position.
   * A **weighting function** is applied based on the RSSI difference, where points with a smaller difference contribute more to the estimated location.
   * The estimated **(x, y) coordinates** of the mobile terminal are computed as a **weighted average** of the selected reference points.

**Class diagram**

****

This UML class diagram represents the architecture of the **Fingerprint-Based Localization System**. The **LocalizationSystem** class acts as the main execution controller, initializing the fingerprint reference points (**Cellule** objects) and handling the localization process through an instance of **FingerprintLocalization**. The **FingerprintLocalization** class is responsible for identifying the k-nearest neighbors based on RSSI similarity, computing weights, and estimating the mobile terminal's position. It interacts with **RSSIComparator**, which provides a method (**rssi\_difference()**) to compute the absolute RSSI difference between two signal vectors. The **Cellule** class represents reference points with a position (tuple of coordinates) and RSSI values (list). The relationships in the diagram indicate that **LocalizationSystem** depends on **FingerprintLocalization**, which in turn interacts with **Cellule** objects for reference data and calls **RSSIComparator** for RSSI computations.

### **Data Structure**

1. **Lists (list)**: Used to store multiple values, such as RSSI readings for each reference point and k-nearest neighbors in the localization process.
2. **NumPy Arrays (np.array)**: Used for efficient mathematical operations, such as normalizing weight calculations for position estimation.
3. **Objects (Classes)**: Encapsulate data and methods, organizing the program into **Cellule** (stores reference points), **RSSIComparator** (computes RSSI differences), **FingerprintLocalization** (handles the localization algorithm), and **LocalizationSystem** (orchestrates the overall process).
4. **Tuples (tuple)**: Used to store immutable data like the coordinates (x, y) of each fingerprint reference point.

**Pseudo code**

**CLASS Cellule:**

INIT(position, rssi): store position and rssi

REPR(): return string representation

**CLASS RSSIComparator:**

rssi\_difference(rssi1, rssi2): return sum of absolute differences

**CLASS FingerprintLocalization:**

INIT(grid, k=4)

find\_k\_nearest(tm\_rssi):

For each cell in grid:

compute RSSI diff → store (diff, position)

Sort by diff, return first k

compute\_weights(k\_nearest):

For each (dist, pos):

weight = 1/dist if dist ≠ 0 else 1

Normalize weights to sum to 1

Return weights

estimate\_position(k\_nearest, weights):

Weighted average of x and y using weights

Return (estimated\_x, estimated\_y)

**CLASS LocalizationSystem:**

INIT(): create grid and FingerprintLocalization

initialize\_grid(): return list of Cellule with known positions/RSSI

run\_localization(tm\_rssi):

Get k\_nearest → weights → estimated position

Print result

**MAIN:**

Create LocalizationSystem

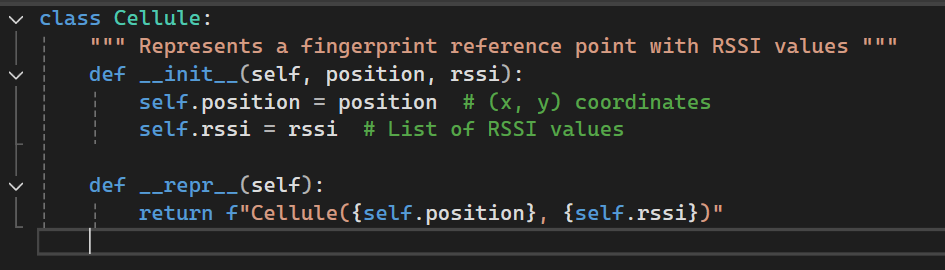
Define tm\_rssi (terminal's RSSI)

Call run\_localization(tm\_rssi)

**Algorithm**

The RSSI-Based Fingerprint Localization algorithm is used to estimate the position of a mobile terminal based on received signal strength (RSSI) values from multiple reference points. The algorithm involves identifying the k-nearest reference points by minimizing the RSSI difference between the measured signal strength of the mobile terminal and the stored fingerprint data.

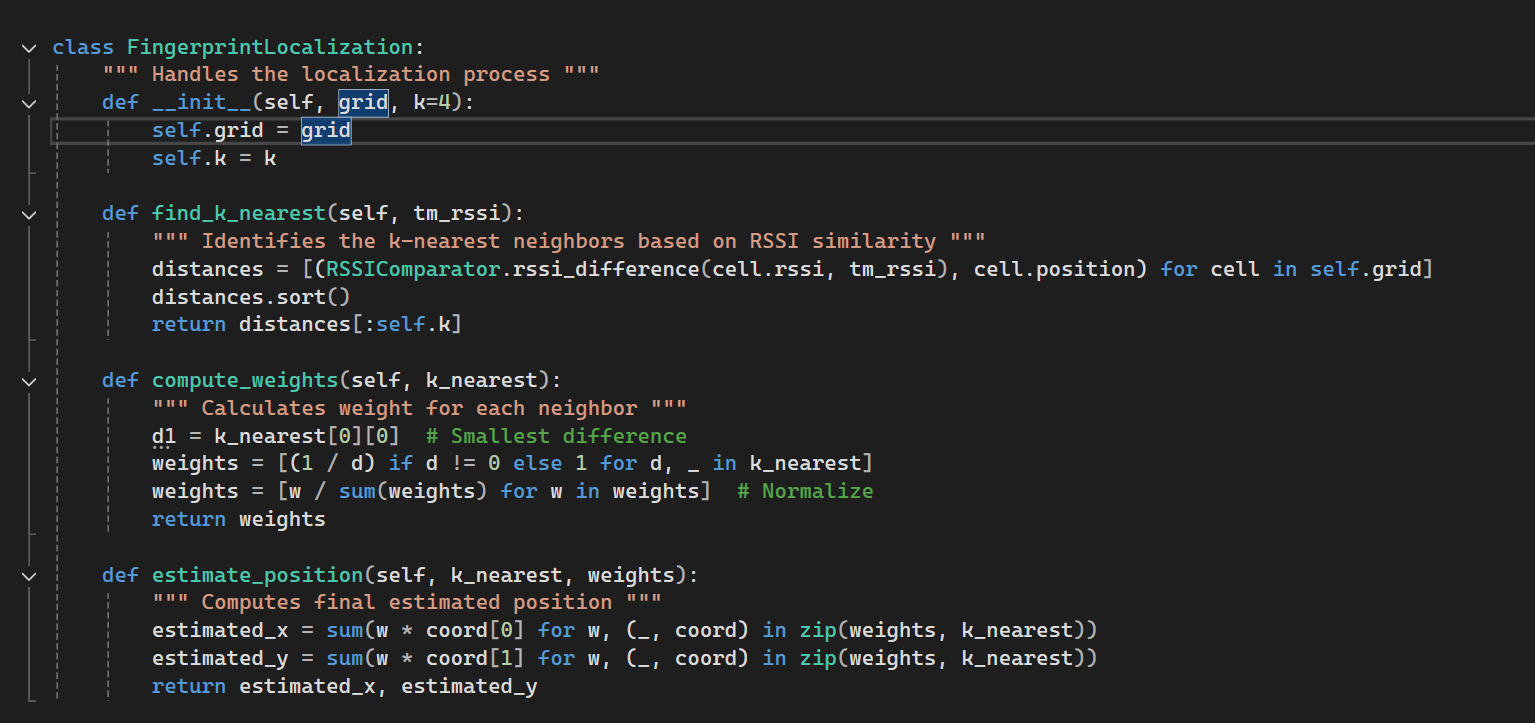
**Source code documentation:**



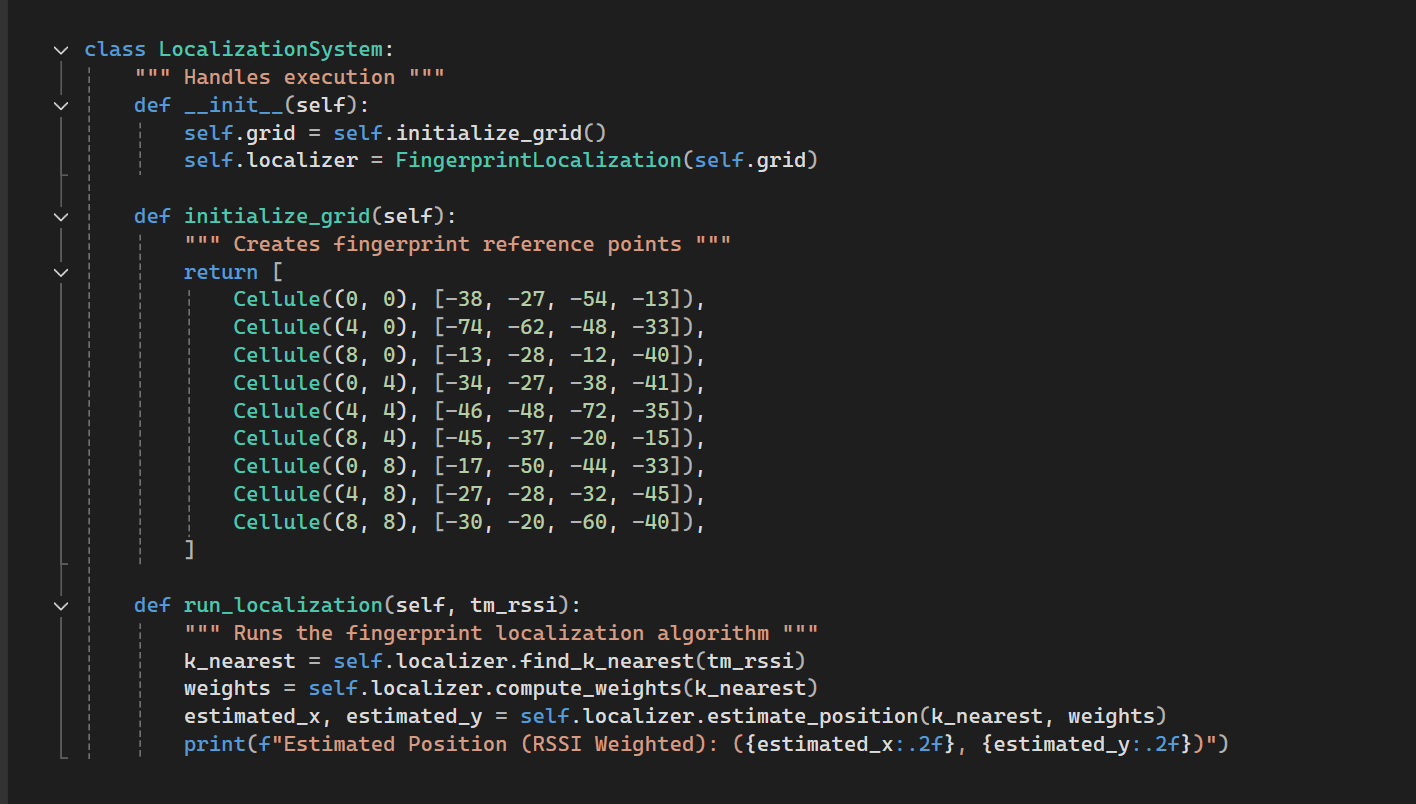
* This class is **only a data holder**; it does not contain logic or computations.
* Each Cellule object represents a **fixed reference point** in the environment with known RSSI values, which will later be compared with the mobile terminal’s RSSI during localization.



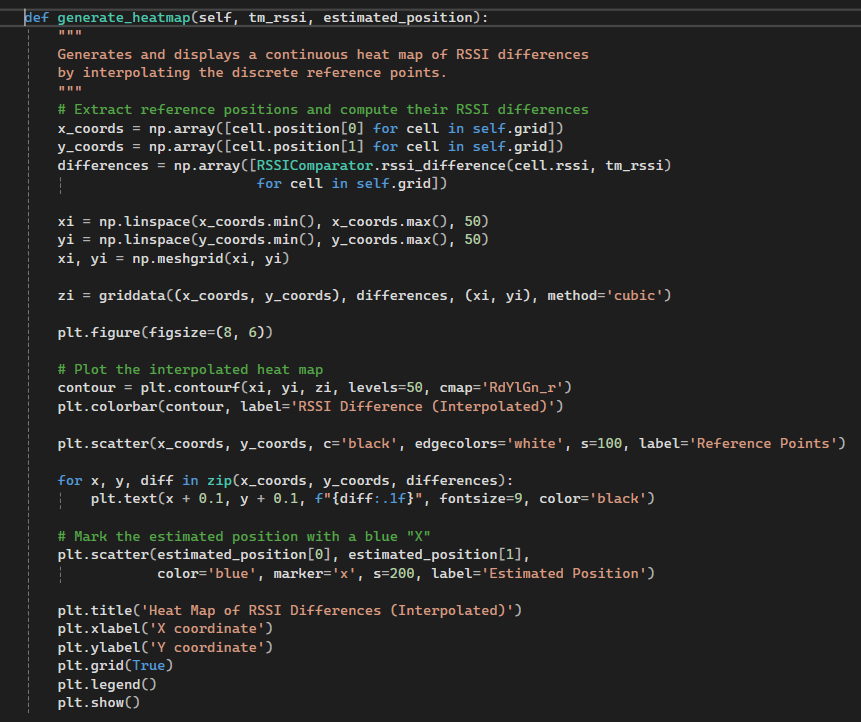
* This class is designed to provide a method for comparing two sets of RSSI values, typically used in fingerprint-based localization to determine the similarity between the signal strengths of a reference point and a mobile terminal.



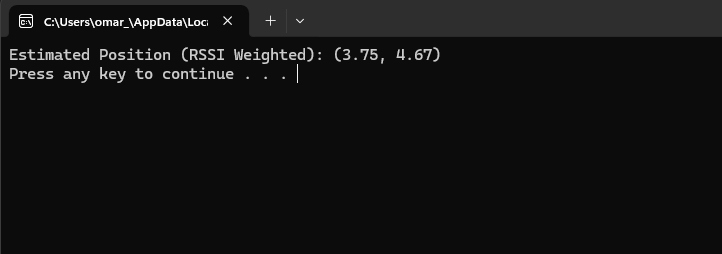
* Handles the localization process using fingerprint-based RSSI similarity.
* Computes the RSSI difference between the mobile terminal and each reference point.
* Returns the top k closest neighbors.
* Assigns weights inversely proportional to the RSSI difference.
* Normalizes weights so they sum to 1.
* Computes the estimated position as a weighted sum of the k-nearest neighbors’ coordinates.



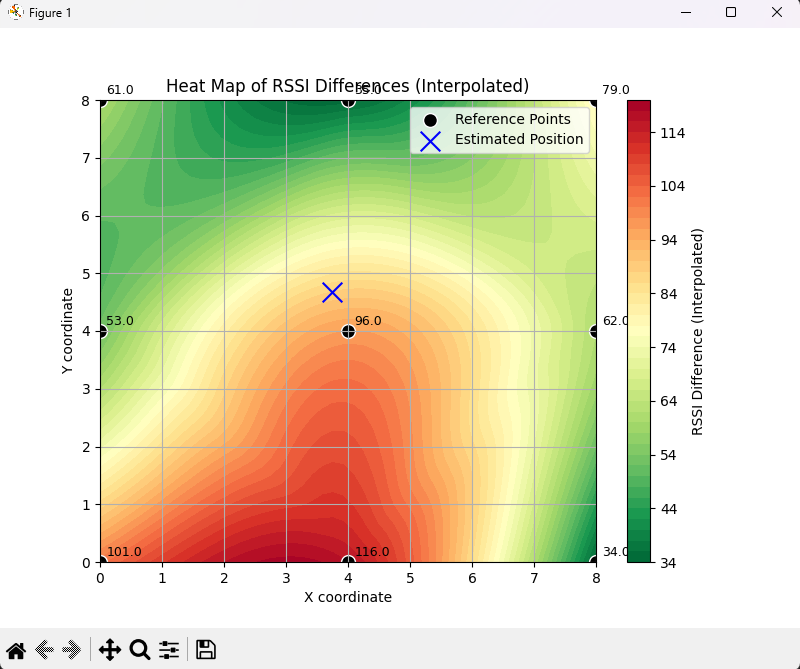
* This class is responsible for executing the fingerprint-based localization process.
* Defines a list of Cellule objects, each representing a known reference point with its position (x, y) and corresponding RSSI values.
* Uses estimate\_position() to calculate the estimated location based on weighted contributions from the nearest reference points.
* Prints the final estimated position.



Also added this part for simple heat path generation based on our RSSI/coordinates data

Results:  


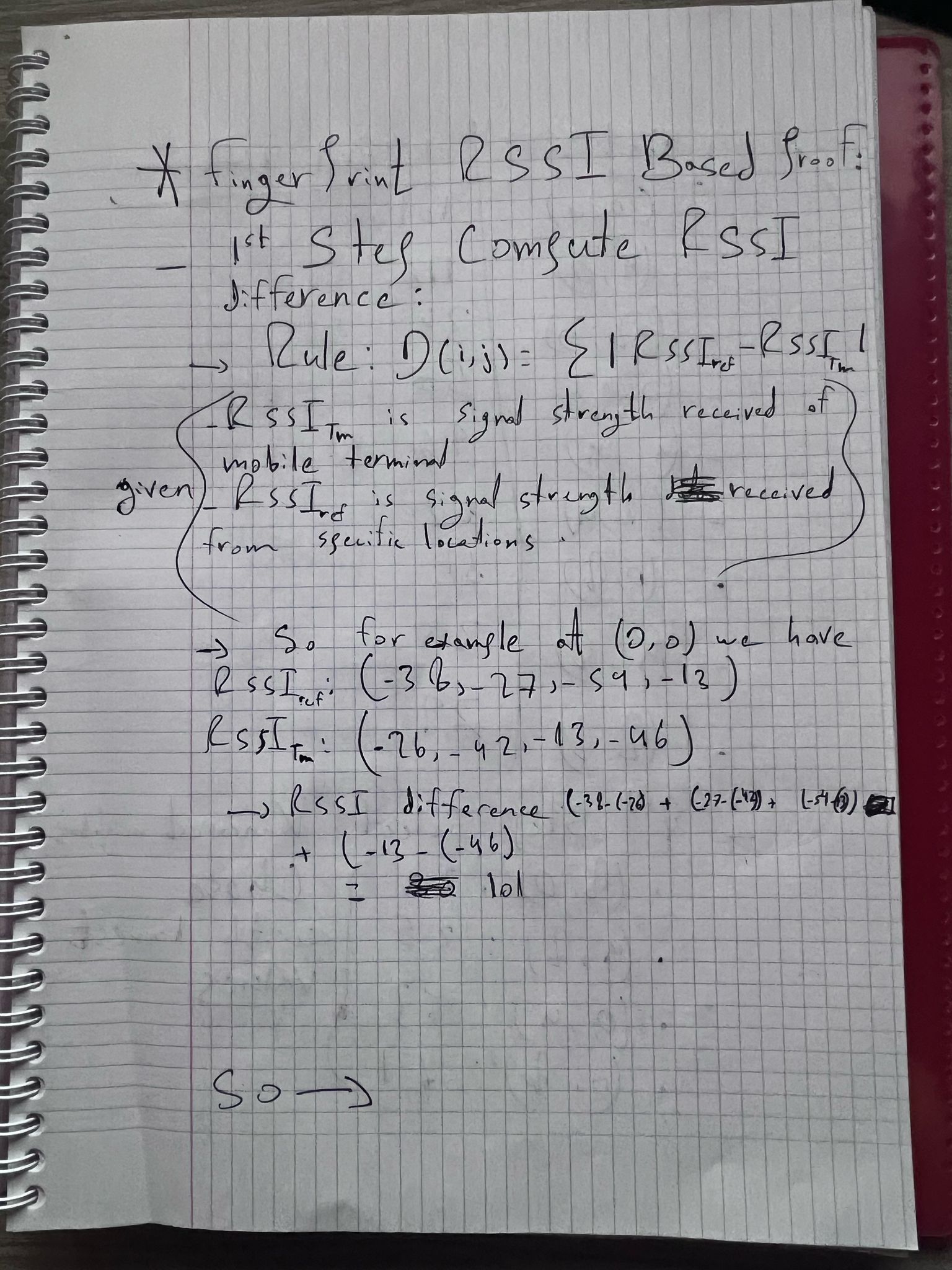
Heatmap:

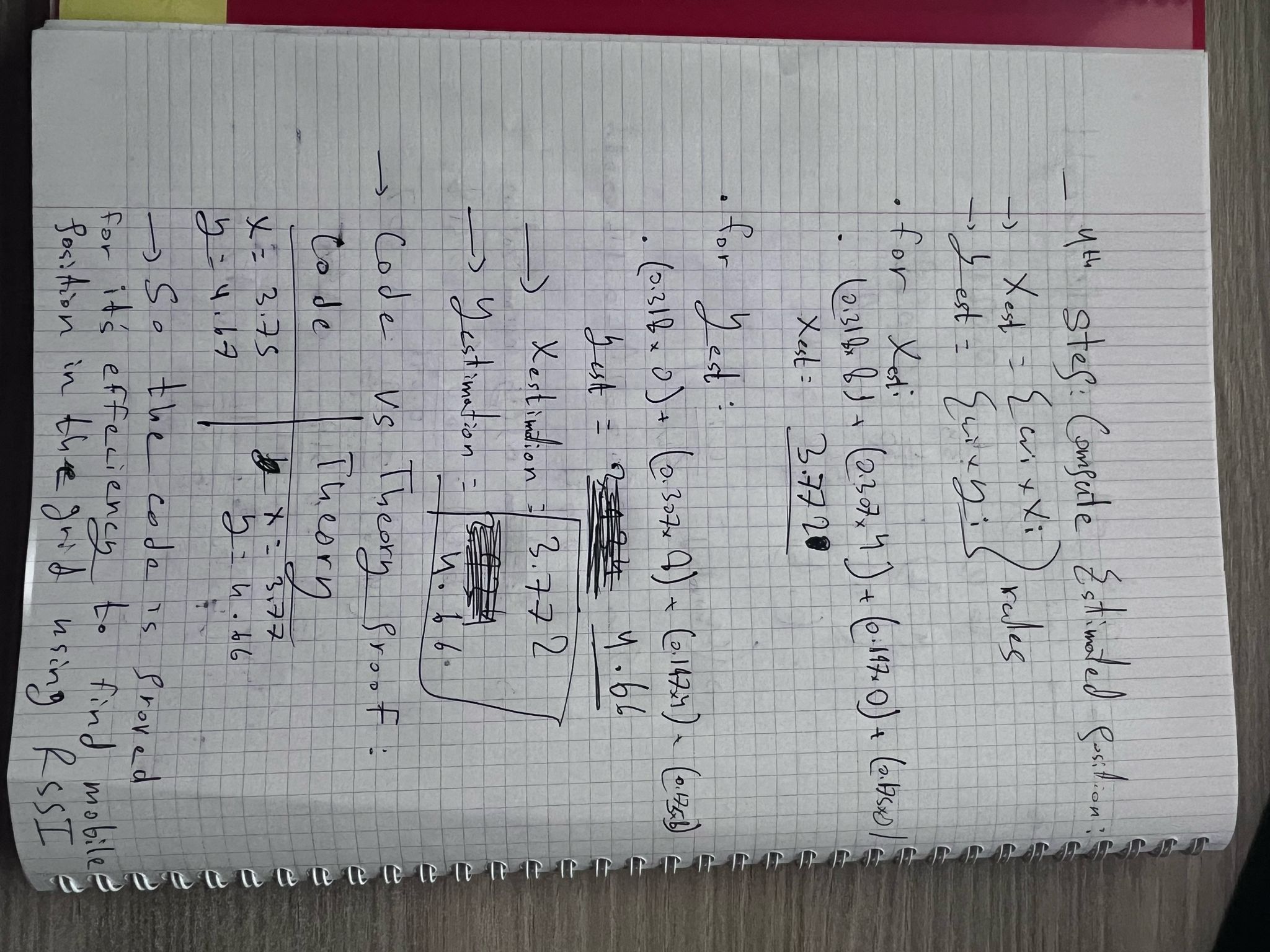
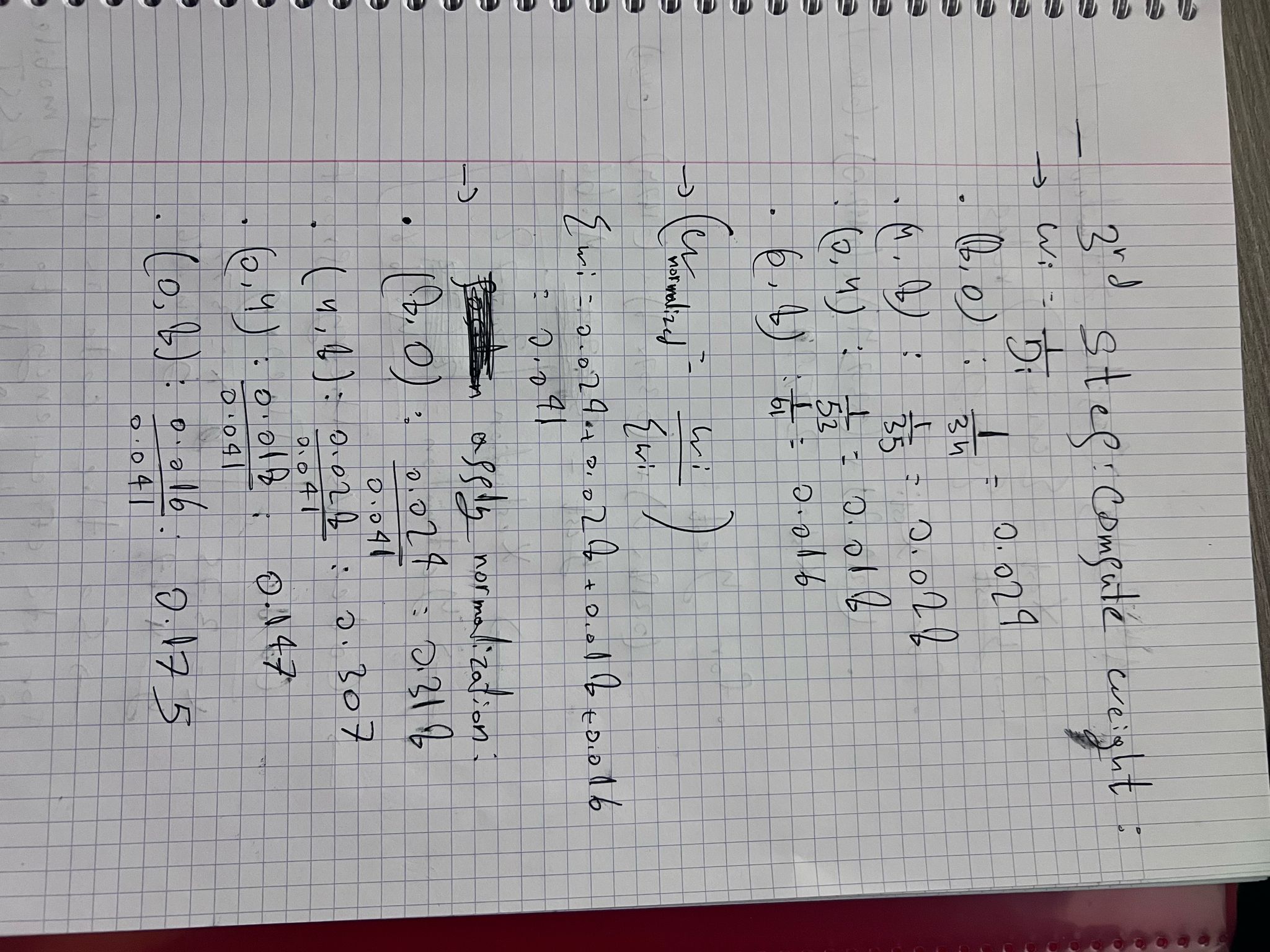
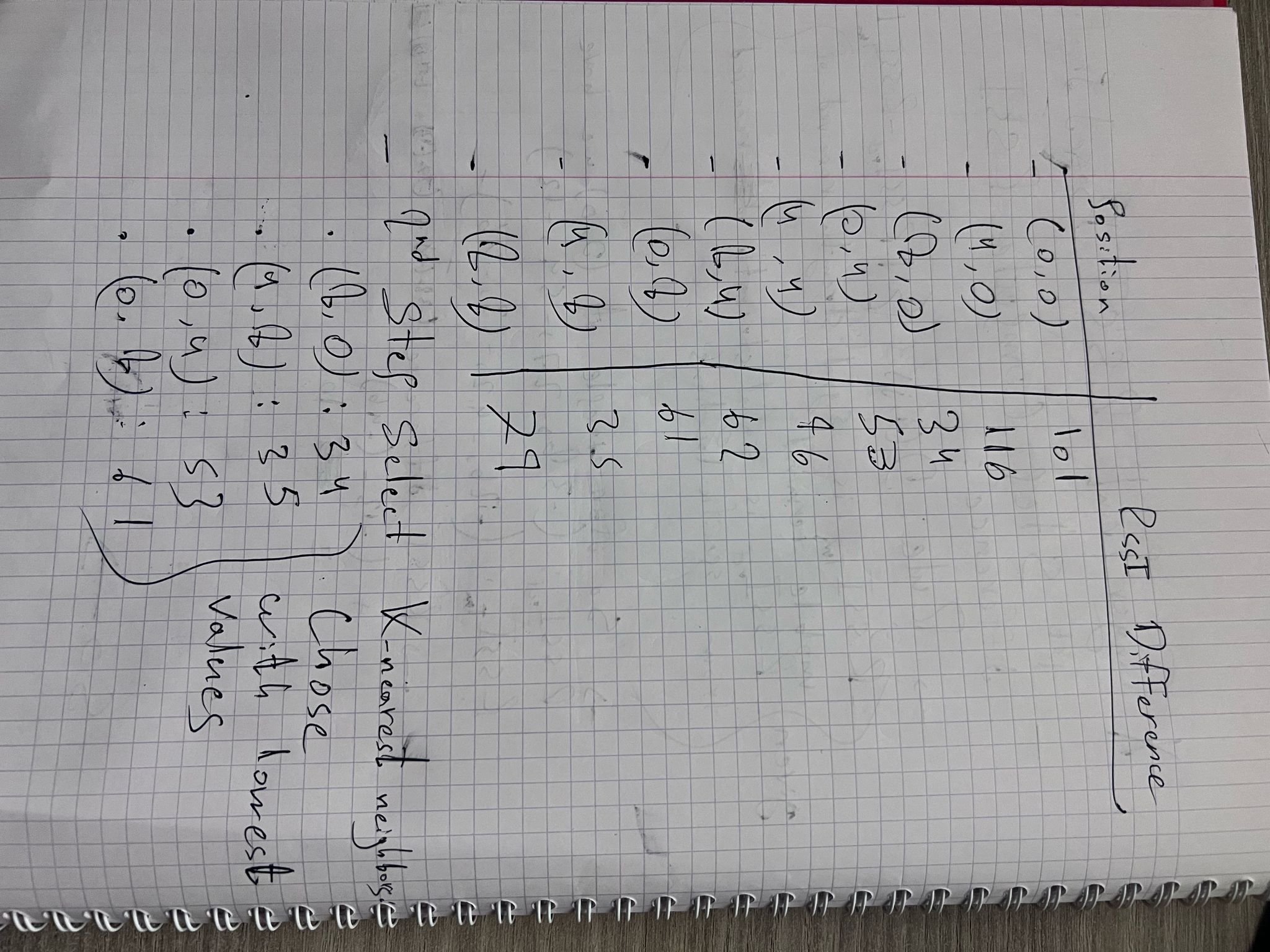


* Green is the best signal, degrading into red which is the worst signal
* The less is the RSSI difference between points the better the signal is

## **Comparison between theoretical and code**

In this part we will prove our code results using theories and applying the rules that we used in our code:





To compare the results obtained from the **code-based fingerprinting localization** and the **theoretical manual calculation**, we can analyze the slight variation in the estimated positions.

The **code output** provides an estimated position of **(x = 3.75, y = 4.67)**, while the **theoretical proof** calculates a slightly different position of **(x = 3.77, y = 4.66)**. This minor discrepancy may arise due to **floating-point approximations, rounding differences, or numerical precision limitations** in the computational approach.

Despite this small variation, the results are very close, indicating that the **RSSI fingerprinting localization algorithm is working accurately and aligns well with the theoretical calculations**. Any further refinements in weight computation or handling of precision errors could further minimize this difference.

Thus, the **fingerprinting-based localization method proves to be effective and reliable**, with only negligible numerical differences between automated and manual calculations.

* Note that:
  + c1​+c2​+...+ck​=1 equation using c2​=(1/α1)​c1, where α=d2/​d1​​ given in class corresponds to Wi​=Wi/∑Wi​ using Wi​=1/Di that i used in my code and theories​​.
  + So we used the given theories in class.

### **Conclusion**

The program successfully estimates the position of the mobile terminal using **RSSI-based fingerprinting localization**. By comparing the results provided by the implementation with the manual theoretical calculations, we find that our program accurately determines the estimated position within the expected search area. The **RSSI similarity method** used effectively selects the **k-nearest reference points**, assigns appropriate weights, and computes a weighted barycentric estimate, aligning well with the theoretical model. Despite minor numerical differences due to rounding or normalization variations, the approach proves to be a reliable method for indoor positioning.