INDOOR POSITIONING USING RASPBERRY PI

Taner Eşme, 2018

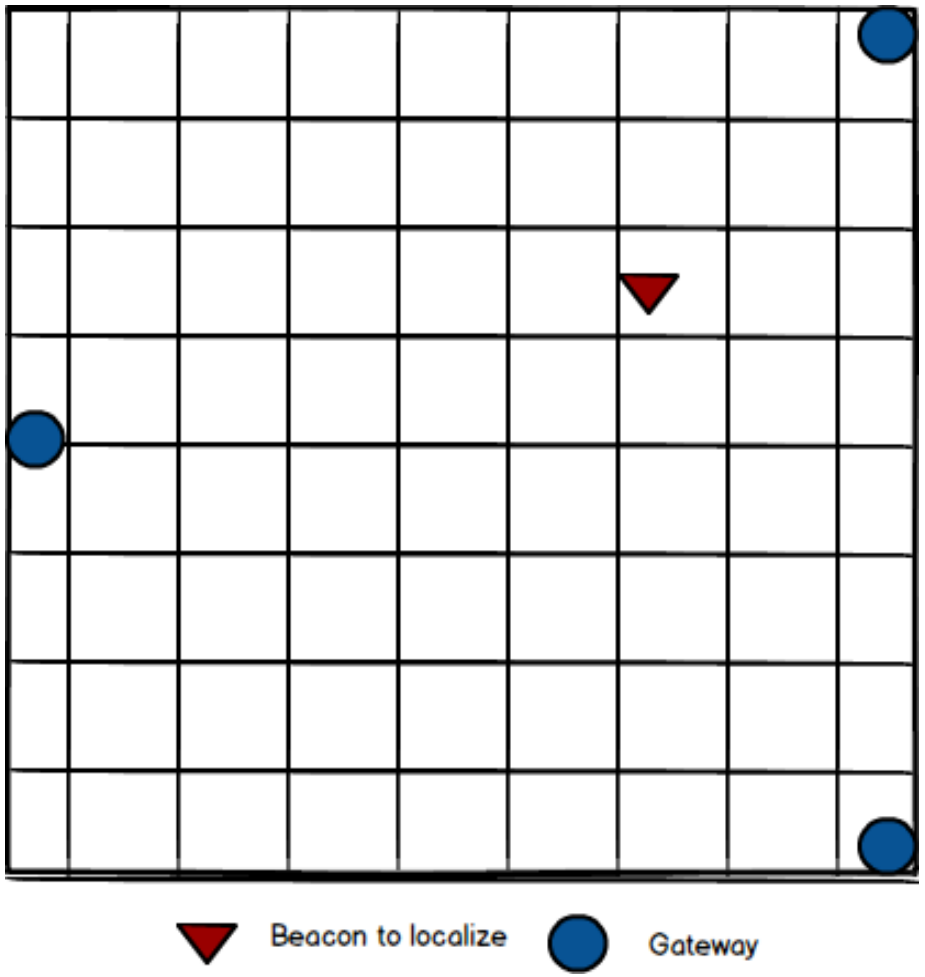
Project Description

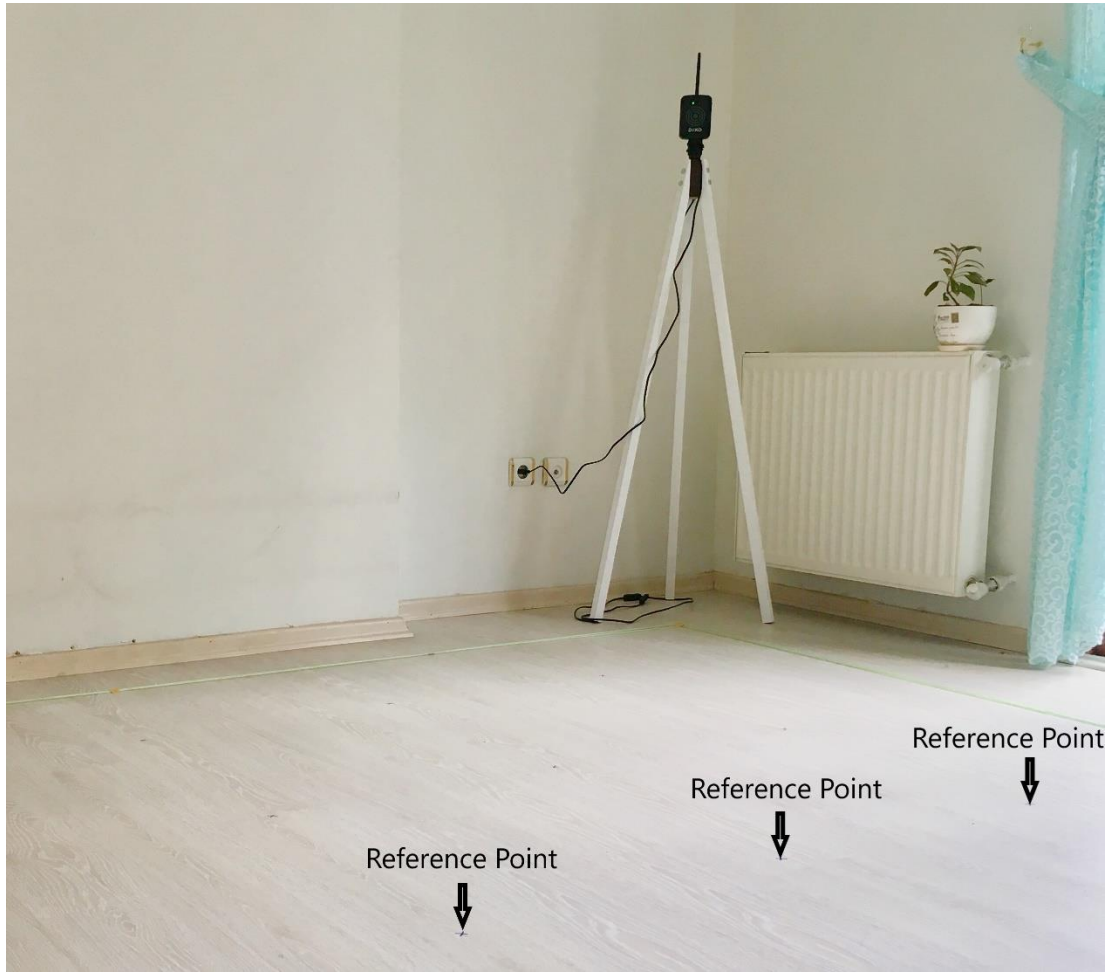
- Estimating the location of a Bluetooth capable device in an indoor environment has many practical purposes. Either for a smart warehouse with autonomous robots carrying the goods around, or for a classic multi-storey office with employees moving from one place to another.
- Indoor positioning can be performed using a set of stationary Bluetooth access points.
- In this project, we are going to implement a hybrid solution of fingerprinting and triangulation (not triangulation, trilateration).

Requirements of the Project

- Raspberry Pi 3
- A cheap version of beacon or a smartphone
- .NET Core 2.1
- Windows 10 IoT Core
- A server or a commodity computer
- Visual Studio 2017
- A PC installed Windows 10

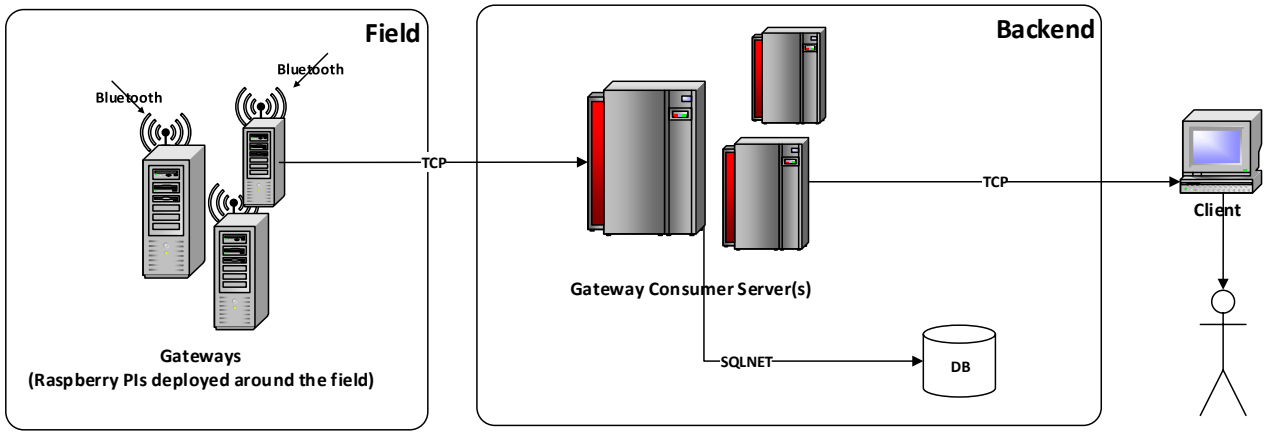
Design of the Project - Environment



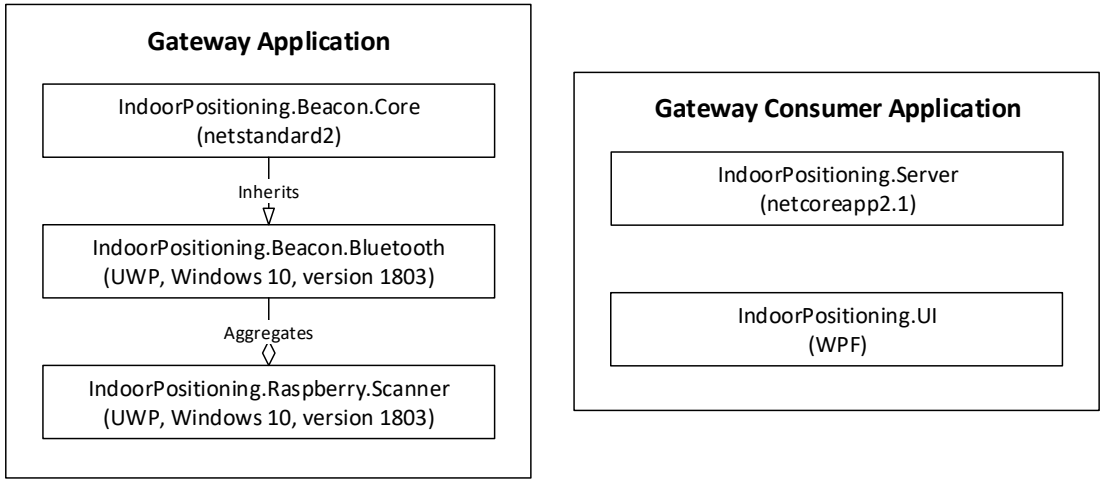


Design of the Project

- Logical Design



- Software Modules



Implementation

Three different application implemented.

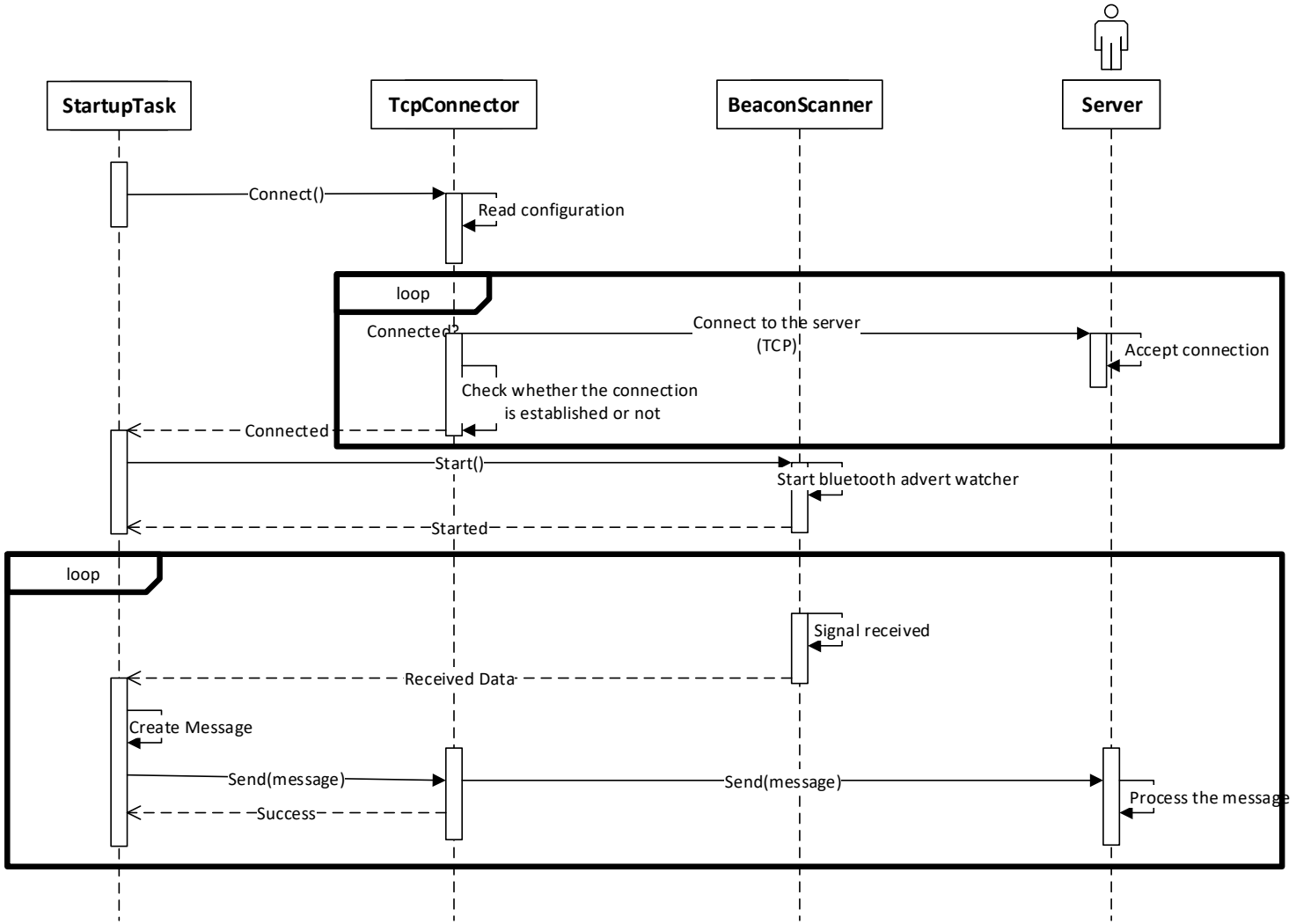
- **IndoorPositioning.Raspberry.Scanner:** The application running on Raspberry PIs
- **IndoorPositioning.Server:** The application that Raspberry PIs connect and transmit the RSSI values they collect
- **IndoorPositioning.UI:** The user interface to manage all of the functions of the system

Scanner Application

- It is a straightforward, lightweight background application developed based on UWP in .NET Core and runs on Raspberry PIs.
- There is a predefined IP address and port for the server in the application to connect.
- The application listens to the Bluetooth signals, reads the data radiated in the Bluetooth packages such as mac address of the device, RSSI (received signal strength indication), and transmits them to the server through TCP/IP protocol.
- Sample frame:

Gateway Type	Beacon Mac Address	Gateway Mac Address	RSSI
RASPBERRY,	AC233F23BB8C,	B827EB4BACAA,	-50,

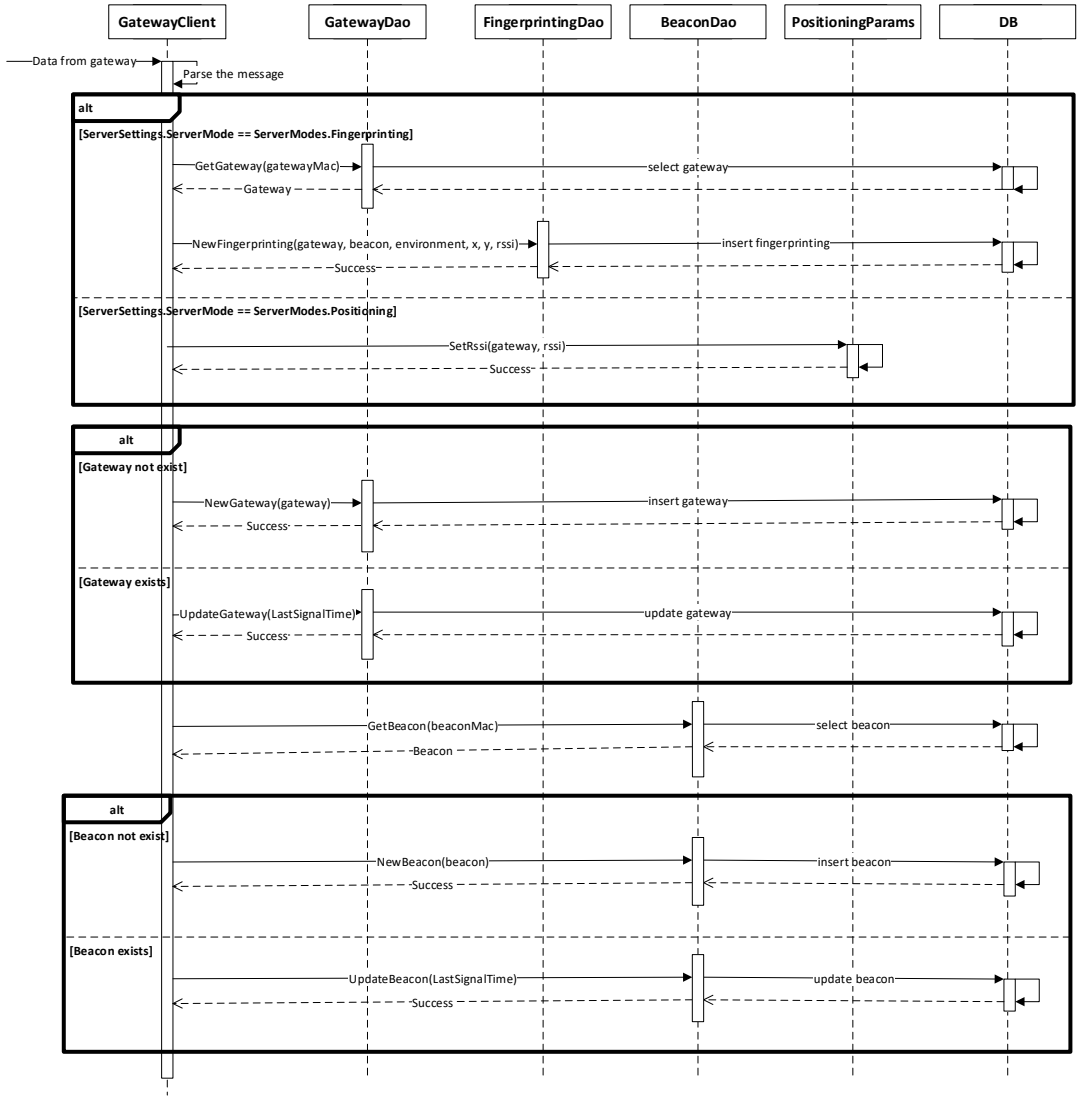
Scanner Application – How it works



Server Application

- The application that all Raspberry PIs deployed to connect and communicate. It listens to a specific port known by the gateways and a different port that is used for the service and managerial purposes.
- A tailored command based integration protocol upon TCP/IP is developed to allow the user interface to fetch and post the necessary information for the purposes such as management, fingerprinting and positioning.
- Sample commands:
  - `get beacons` returns all the beacons stored on DB as JSON.
  - `get fingerprinting –env <digit>` returns all the fingerprint values corresponding environment id of `<digit>` stored on DB.

Server Application – How it works



User Interface

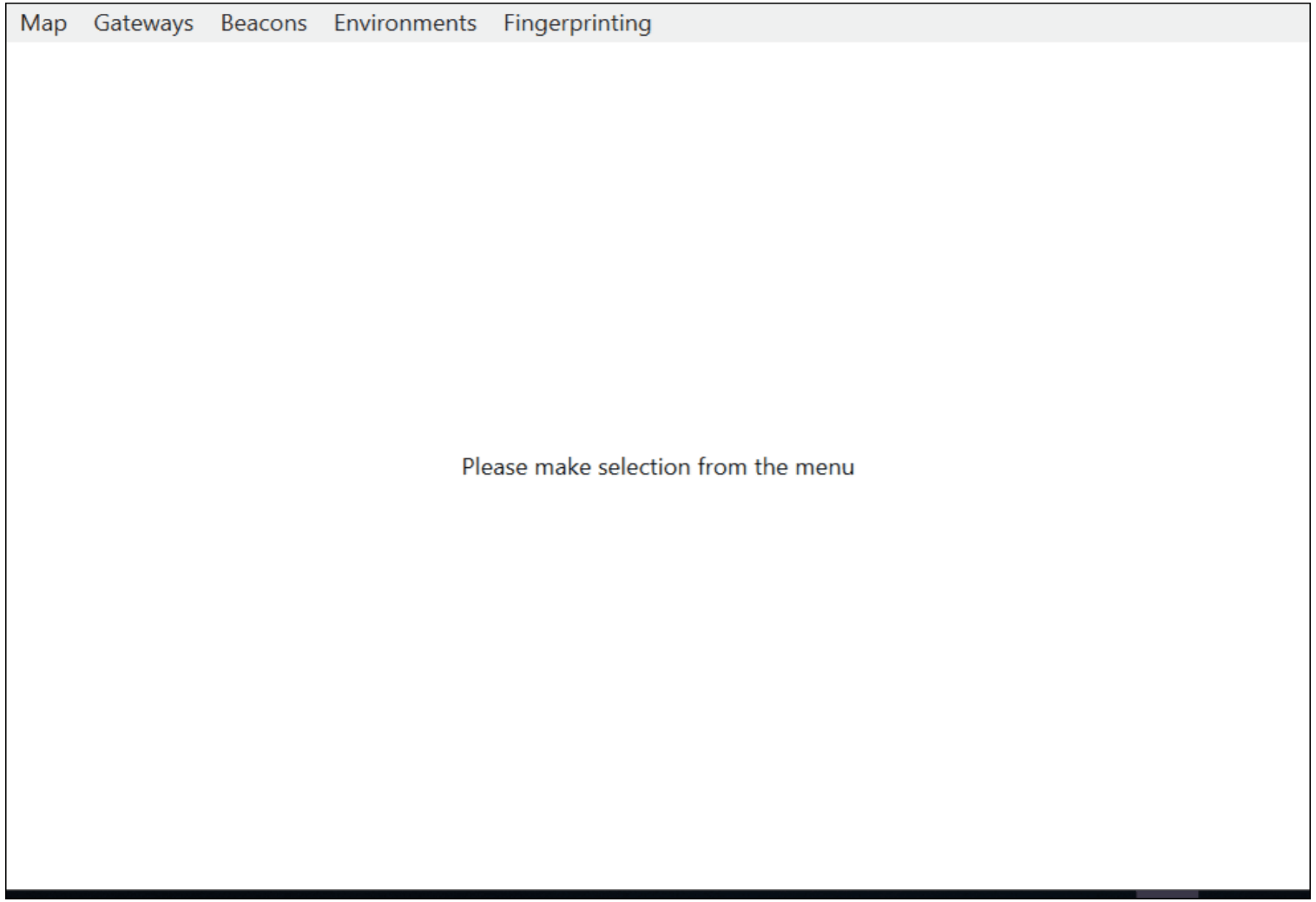
- The user interface has 2 major roles in the project.
  1. To use the server commands in an easy manner.
  2. To run the positioning algorithms.
- The server collects lots and lots of Bluetooth data from the gateways, but it cannot determine which data will be consumed for the positioning purposes. That’s why this decision left to the user interface and turned into a demand-triggered approach.

User Interface - Screens

- **Environment Screen:** The application designed to manage more than one environment. Therefore, the first thing to do before starting positioning is to define a new environment with the necessary information needed by the system.
- **Gateways Screen:** All of the gateways connected to a server through TCP saved in DB automatically. On this screen, all the gateways connected to the server in any time were displayed and it will let you delete and update the name and the position of the gateways.
- **Beacons Screen:** The server collects all beacons' data from and stores them on DB. If it receives a data for an unknown beacon, it will store them as "unknown" and on this screen, these "unknown" beacons can be turned into the known ones.

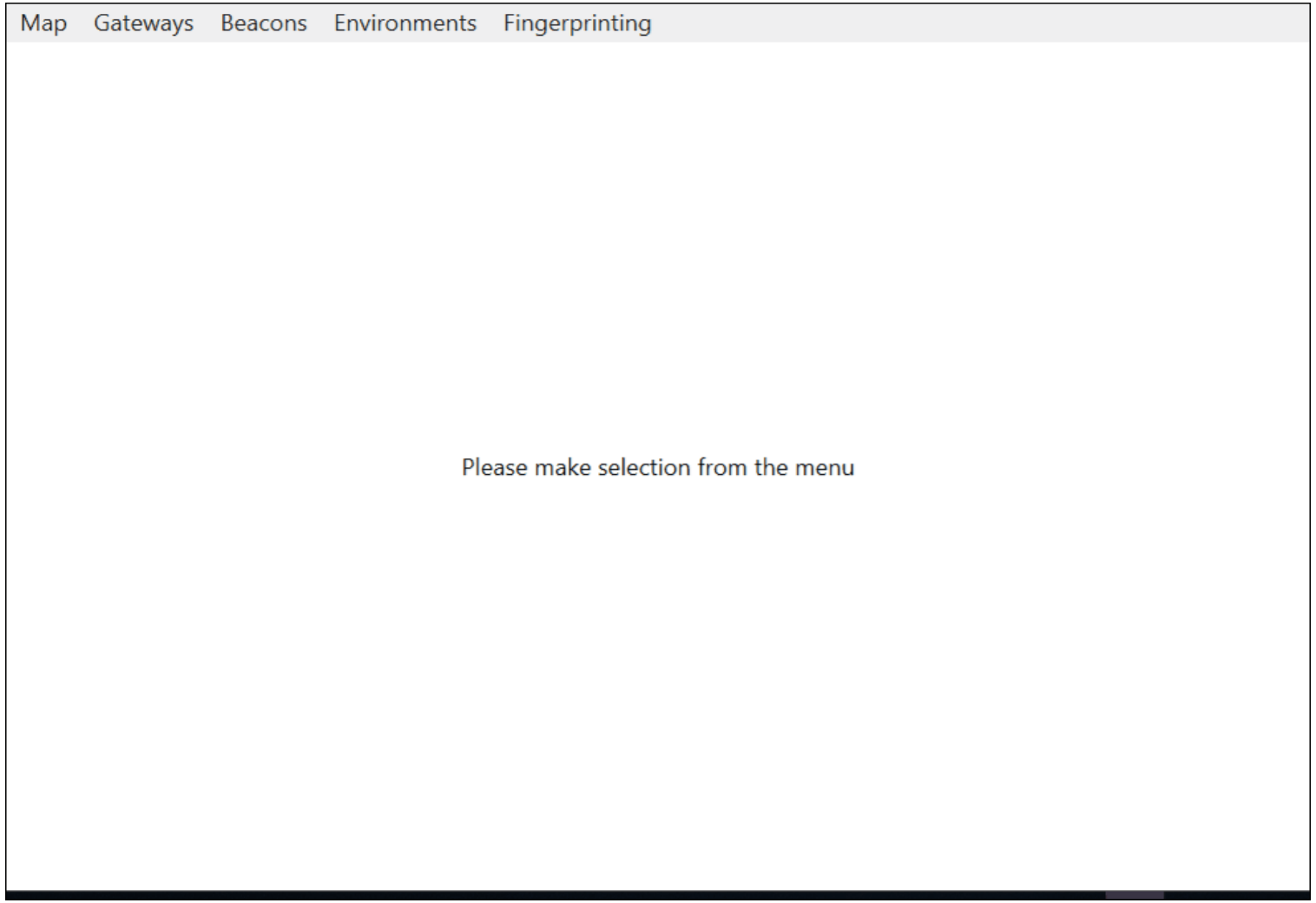
User Interface – Screens - Fingerprinting

- It allows the users to work on the fingerprinting even more easliy



User Interface – Screens - Map

- It is desiged to visualize how positioning goes at runtime.



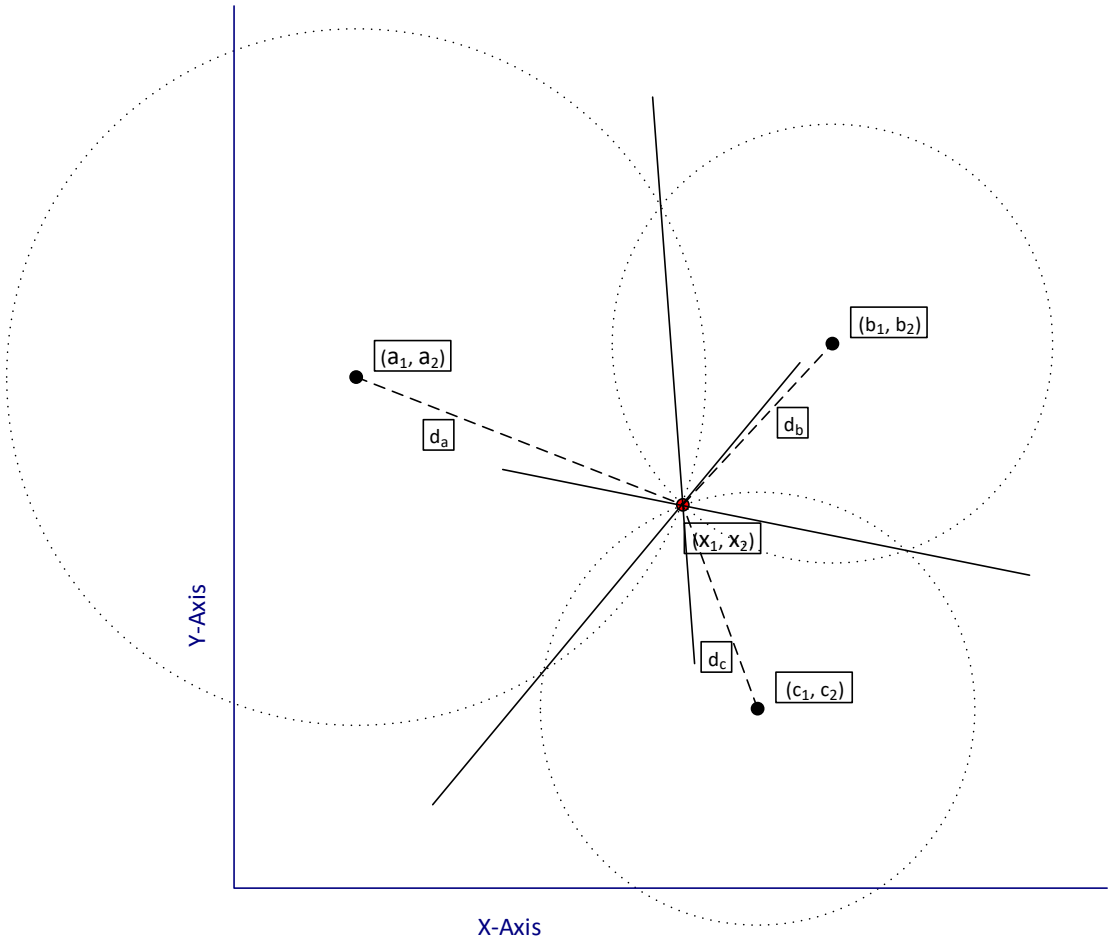
Positioning Algorithms – KNN Classifier

- K-Nearest neighbors classifier algorithm trained by fingerprinting data tries to classify the collected RSSI values from the gateways to one of the reference points not to coordinate.
- Pseudocode:

```
k-Nearest Neighbor
Classify (X, Y, x) // X: training data, Y: class labels of X, x: unknown sample
for i = 1 to m do
  Compute distance d(X_i, x)
end for
Compute set I containing indices for the k smallest distances d(X_i, x).
return majority label for {Y_i where i ∈ I}
```

Positioning Algorithms – KNN Proximity

- We make use of KNN to find the nearest neighbors and measure the distances. It is basically an implementation of trilateration.
- $$2\begin{bmatrix} a_1 - b_1 & a_2 - b_2 \\ a_1 - c_1 & a_2 - c_2 \\ b_1 - c_1 & b_2 - c_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (\|a\|^2 - d_a^2) - (\|b\|^2 - d_b^2) \\ (\|a\|^2 - d_a^2) - (\|c\|^2 - d_c^2) \\ (\|b\|^2 - d_b^2) - (\|c\|^2 - d_c^2) \end{bmatrix}$$
- The equation is solved for each of the pairs of two points and the average of the solution is taken as a result.



Positioning Algorithms – RSSI Proximity

- The same equation in KNN Proximity is utilized without appealing to KNN and fingerprinting dataset. The RSSI values read from each of the gateways were considered as the distances of the gateways to the beacon (we already know positions of the gateways) and according to that, the equation solved.

Conclusion

- Hard to localize the Bluetooth capable devices indoor because of the fluctuation, signal reflection, and weak data.
- Fingerprinting is one of the ways that have the potential to increase the accuracy of the positioning and supporting it with machine learning or deep learning algorithms can be even more beneficial.

THANK YOU