COMP3516 2024 Spring Group Project: RSSI Localization via ESP32

Indoor localization is essential to a range of applications, which, however, has been a long-standing challenge for decades. Through this project, you will build an end-to-end localization system based on Wi-Fi RSSI. The project can be divided into several parts:

1. RSSI Data Collection: Get RSSI data from an IoT device
2. Data Transportation: Send the RSSI data to an edge/cloud server for processing
3. Localization Algorithm: Estimate the client's location based on the collected RSSI data
4. System Evaluation: Evaluate the system performance both on benchmark datasets and in real-world scenarios.

The students are asked to team up as four in a group to finish the project. Each team will be provided with an ESP32 module and a RSSI dataset post-processed based on a public dataset. Upon completion of this project, you will be able to learn:

- How to use an IoT device like ESP32 and fetch data from it;
- How to configure WiFi interfaces and perform RSSI scanning;
- How to use MQTT for IoT data transportation;
- How to design and develop RSSI-based localization algorithms;
- How to evaluate a system and analyze its performance;
- How to build an end-to-end system in collaboration with your teammates.

Task Description

Task 1. Obtain RSSI on ESP32

Rather than using your own laptops or smartphones as the client, you will be provided an ESP32 module as an IoT WiFi client to measure RSSI of surrounding APs. You need to familiarize yourself with the ESP32 device and its specifications, and then develop your code to scan APs to obtain RSSI data from the device. Your code may need to support flexible configurations of parameters such as scanning intervals.

Upon completion of this step, you should receive a list of RSSI values, with corresponding MAC addresses of the source APs, every once a while.

Task 2. WiFi Connection and Data Transportation

The ESP32 should be wirelessly connected to a host machine, e.g., your laptop. This can be done in two ways:

1. Connect both your ESP32 and your host machine to the same network (e.g., HKU WiFi network, or your home router).
2. Create a hotspot on your laptop and have the ESP32 connect to your laptop.

Once the Wi-Fi connection is established, we can transmit data from the ESP32 client to the host machine via various protocols. In this project, you are required to use MQTT for the purpose. Note that, although in the project you will only have up to 2 or 3 devices in total (ESP32 and laptops), you are still required to follow a potentially scalable architecture as follows:

- ESP32: `publisher` to send RSS information, and `subscriber` to receive commands from the UI.
- Host machine (your PC): `broker` which forwards data between publishers and subscribers.
- Engine (could run on the same one as host machine): `subscriber` to receive and comsume RSS data and `publisher` to send commands (e.g., start/stop data collection).

Once you've finished the above configurations, you should be ready to perform RSSI scanning on the ESP32 and publish the RSSI data to the broker, which will forward the data to the engine for further processing in the further tasks below.

Task 3. Localization Algorithm

You will need to design and implement your own localization algorithm based on the RSSI measurements of surrounding APs, whose locations are assumed to be known. You are free to explore any algorithms in the existing literature or come up with your novel design. The algorithm is supposed to take the RSSI measurements and AP locations as input and estimate the client location as output.

In general, there are several directions for your exploration:

1. Maximum RSSI based Localization: This is by no means accurate, but you can simply locate a client as the location of the AP with the highest RSSI.
2. Weighted Centroid: You can also simply treat the observed RSSI from each AP as the weight of the distance to the corresponding AP and then take the weighted centroid as the location estimate of the client. The weight can be as simple as linear weights, i.e., $w_i = 1/RSSI_i$, while you can certainly explore more sophisticated and accurate weighting methods.
3. Ranging-based Localization: In this approach, we can leverage certain signal propagation model and derive a rough distance between the client and the AP from the measured RSSI. Specifically, you can look into certain `path loss model` and develop your own RSSI-based ranging model. Note that there are usually quite some empirical values in these path loss models, which you could directly take from existing works. Even better, perform some numerical measurements to estimate and tune those parameters. Once you have the range estimates to each AP, you can employ the trilateration method for localization by, e.g., using the Least Squares method.

Task 4. System Evaluation

Usually, one needs to conduct experiments and build his/her own dataset for evaluation. In this project, to save you some efforts in data collection, we have parsed a dataset for your evaluation based on a public dataset. The dataset contains the following information:

- **RSSI**: *[n_datapoints x n_ap]* 2D received signal strength matrix.
- **labels**: *[n_datapoints x 2]* 2D XY labels.
- **ap**: [*n_ap x 2]* 2D XY labels the n_ap APs.

You are required to evaluate your algorithm using the `RSSI` and `ap` locations as input and benchmark the location errors with the `labels` as ground truth. You can use Euclidean distance as the error metric and are suggested to report the results in a CDF (Cumulative Distribution Function) figure.

Apparently, you only need to run your algorithm and not worry about ESP32 data collection or communication for the evaluation in this part. The download link to the dataset will be released separately.

Online Resources to Get Started with ESP32

WiFi Configuration and RSSI Scanning

To set up the ESP32 as a WiFi client and perform AP scanning on it, you may take reference to the following resources to get started:

1. Install Arduino IDE on PC for image burning into ESP32 modules: PlatformIO via VSCode
2. Setup ESP32 in WiFi STA mode: Tutorial for ESP32 WiFi / **ESP32 WiFi Scan For Networks Example (Arduino IDE)**
3. Scan WiFi APs and get their RSSI values. You can find example code here: Get ESP32 WiFi Signal Strength (Arduino) & RSSI Value

MQTT

The following materials can be useful to setup MQTT service for the ESP32 device:

1. On Host PC (broker): Create an MQTT broker service using Mosquitto.
    a. Download & install Mosquitto
    b. Installation tutorial (Windows, Mac)

2. On ESP32 (publisher)
    a. Install PubSubClient in Arduino IDE
    b. Tutorial Video on Publish and Subscribe using MQTT on Platformio (Arduino)
3. On Host PC (subscriber): Run MQTT client python Paho-MQTT.
    a. YouTube: MQTT Beginner Guide with Python

Test Your System in the Wild

In this part, we will try to make more fun out of this project by testing your system in the wild for several different purposes.

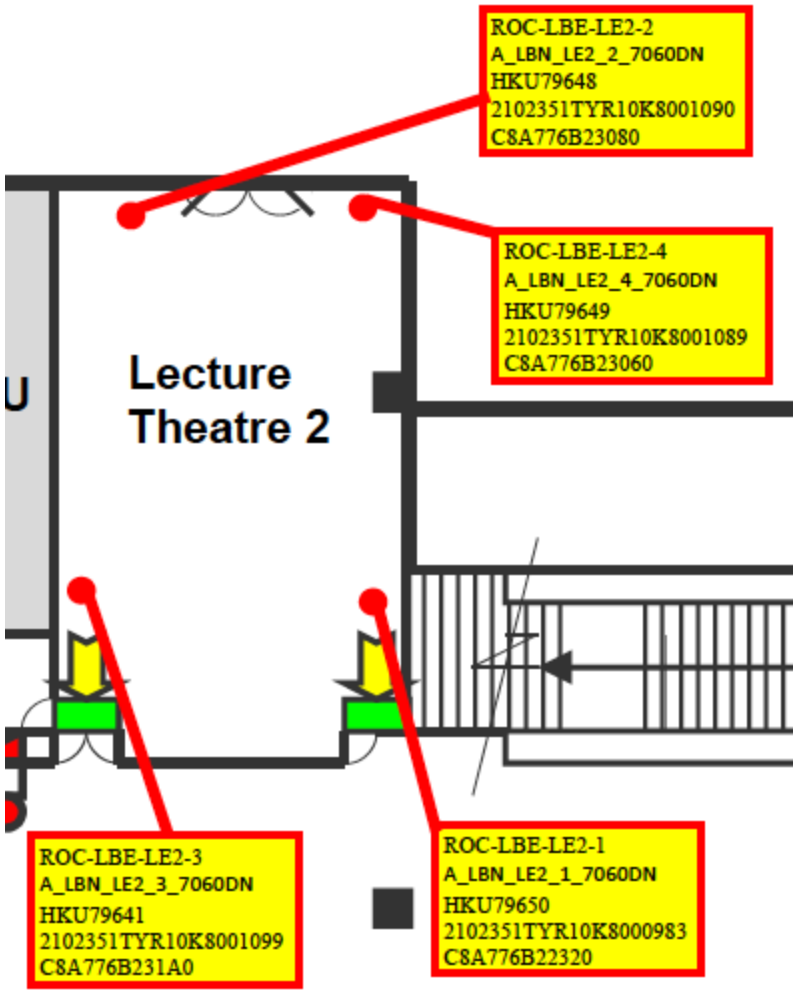
AP War-Driving

Wardriving is the act of searching for Wi-Fi wireless networks using a laptop or smartphone. In this part, we will do war-driving on HKU campus. Specifically, you are asked to walk around the Chi Wah Learning Commons along while performing RSSI observations using your system. Your task is to report at least 5 APs with their most likely locations (your best estimates based on the RSSI traces of each AP around the space) on Level-1 (CPG-1) of the Chi Wah Learning Commons. Please report the MAC address, SSID (AP name), and their location description (e.g., closest to Seat xxx/Outside Room xxx/etc) and mark on top of the below floorplan.

LE2 Localization

There are four APs inside LE2, our classroom this semester. We can utilize them for localizing ourselves (the ESP32 device). To complete this task, go to the LE2, pick a random seat (and mark the seat number as the reference ground truth location (in the format of <Row xx, Column xx>) and place your ESP32 device on the chair. Assuming the rough location of the four APs are given, perform RSSI measurements from the four APs in the room and estimate the location of the device. You can capture multiple measurements at the same location.

A reference floor plan of LE2 is given below.



In-class Competition

Last, we will have an in-class competition to challenge your design and implementation. There will be two different tasks as follows:

Task 1: Client Localization: This is the same task as `LE2 Localization` in the above, but now in a live session. Each group is seated somewhere (e.g., <Row X, Column Y>) and is required to estimate your own location based on the four APs in the classroom. The location needs to reported by your system, rather than by yourselves, apparently:-D

Task 2: AP Localization: We will install one **hidden** AP somewhere in the classroom. Each team will be allow to move around and perform RSSI measurements at a set of specified locations. Then each team need to output their best guess of the hidden AP's location. The team gives the closest estimate will be the champion!

More details about the in-class competition part will be announced later in due course. In-class competition is supposed to happen after the deadline of submission.

Submission

Your need to submit both your source codes (including those running on ESP32 and on the host machine) and a technical report. The report should be up to 4 pages (including figures, tables, and references, if any) containing the following sections:

- System Overview: Describe the overall architecture of your system
- Algorithm Design: Describe how your algorithm works
- Evaluation: Report the evaluation results based on the given datasets. Report the mean error, median error, and the CDF.
- Real-world Experiments:
    - War-Driving: Report your war-driving results. Please include one RSSI trace as the example to show how you determine the AP location.
    - LE2 Localization: Report your measurements and tests in LE2, along with your results.
- Contribution Statement: You must include a section of contribution statement to describe the contributions made by each team member. Note that the contribution statement will be referenced for grading and it is presumed that team-wide consensus has been achieved.

A template will be provided for the project report. The in-class competition part is not required in the report.

Alternative Project Topic

Besides the above topic, we accept up to **two** groups of students to work on a different topic, which is more research-oriented, while following a similar setting as the default one. Particularly, in this project, you are asked to design a fall detection system using a low-cost thermal array sensor (which can be viewed as an extremely low-resolution thermal camera). Similar to the above default project, you will be provided with an ESP32, equipped with the thermal array sensor, and you need to finish all the components for building an end-to-end system, including data acquisition, data transportation, etc. Importantly, you will need to come up with an effective algorithm to detect a fall and further implement it as a real-time demo system. You will be invited to do a live demo in the class during the in-class competition session.

For you to get a sense of the sensor output, below is an example thermal map captured by the sensor with two users standing in front of it: