Indoor Localization Activity

Overview

Indoor localization, i.e., the accurate, automatic determination of a person's (device's) position inside an indoor environment is a hard, still unsolved problem when it comes to reliability and universal applicability – and is very different from outdoor localization!

In this exercise you will explore indoor localization through Bluetooth. Submit your answers as a PDF via this deliverable format doc.

ONLINE: You will need 2+ smartphones/tablets/other devices with BLE, and a computer with Chrome installed. The more devices you have, the better, so if you can, we recommend 4/5 devices + a computer.
ON-CAMPUS: You will need 1 smartphone/tablet/any other device with BLE, and your laptop with Chrome installed.

Have fun!

Preparation

1. **BLE introduction:**
   Read this article to learn about BLE in general, it should take you 15 mins or less.

2. **BLE scanning and advertising software on your phone:**
   a. Now that you learned BLE in general, it is time to actually play with BLE. Install **nrf connect app** on your smartphone. (link to Android version, link to iOS version)
   b. The main screen of the app is the scanner view. You can see all the nearby BLE devices and their RSSI (received signal strength indicator). Also play with the **Peripheral** tab which turns your phone into a BLE beacon. Use your teammate's phone (or another device if you are an OMS student) to scan and see it.

3. **BLE scanning on your laptop**:
   a. We recommend using a Mac (macOS Sierra or newer) because the BLE module works better. If you are using Windows you will need to upgrade to Windows 10 or newer. Older versions don't support BLE. If using Linux we recommend an Ubuntu flavor with version 14.04 or newer.
   Note: Linux users please run the line below to install BLE related packages:
   ```
   sudo apt-get install bluetooth bluez libbluetooth-dev libudev-dev
   sudo setcap 'cap_net_raw,cap_net_admin+eip' $(readlink -f $(which python))
   ```
   b. If you haven't, install/upgrade Google Chrome to the latest version on your laptop.
   c. Your Chrome browser has a built-in BLE scanning page. In the address bar, input: `chrome://bluetooth-internals/#devices` to access it.

d. Enable or turn on bluetooth on your operating system.
e. Click ‘start scan’ on the top right corner to see BLE beacons around you.

Task 1 (50 pts)

In this task, you will learn how the value of BLE RSSI (received signal strength indication) behaves through a series of experiments. You will collect data and analyze them. The intuition you get from this task should help you with later tasks. Try to find a place with a minimal number of moving people to do the experiment. You will need two BLE devices for your experiment. One acts as a scanning device and the other one as a target device.

Do this as a team to save time. For example, if you decide to get the values manually using the phone app, one person can shout the values and another person inputs the data into an excel sheet or the tool of your choice. **Read through the entire task before you begin** so you can plan your experiment wisely and save some time.

OMS folks: this activity is substantially easier if you have a friend/coworker/roommate you can complete this activity with. We recommend asking for some help from someone around you if you can, otherwise you can complete it by yourself.

Target device - Phone/laptop/bluetooth headphone/other BLE devices, turn on bluetooth discovery. Or use the nrf Connect app to advertise.

- NRF Connect Settings
    - iOS
        - Settings > Scanner > Scanner Timeout > Never
        - Add Advertiser on the Peripheral Tab
        - Toggle the switch for Editable to Locked
    - Android
        - Tx Power Level - 7dBm
        - Options: Connectable
        - Advertising Data
            - Tx Power Level
            - Complete Local Name

Scanning device - Use your partner’s phone (or the Chrome bluetooth page if you do not have any other devices around)

(20 points) Part A: RSSI as a function of distance

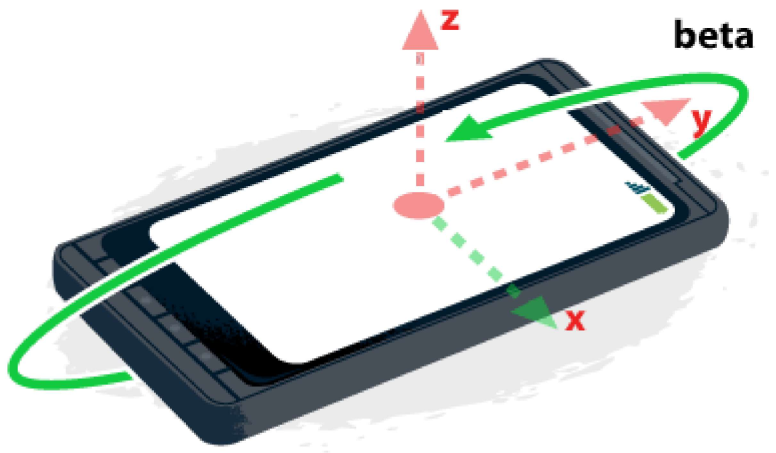
1. Keep both devices (scanning device and target device) still and 2+ meters apart. Make sure nothing obstructs the two devices, ensuring a clear line of sight. We recommend placing each device on a table to avoid erroneous movement. Look at the RSSI values on the scanning device. Think about what RSSI means and discuss it among your team.
2. This experiment helps you observe the change in RSSI as you move closer/farther from your BLE target device.
    a. Start with 1m distance from your device and report the RSSI value. Then, increase the distance between your scanning device and target BLE device by 1m for two further measurements (so, measure for 3 distances in total). For each distance, record the

RSSI value of your target device every 2s, for at least 2 minutes (≥ 60 data points for each distance). You can do this manually by watching the RSSI in nRF connect each second. Once your data is collected, you should have a dataset of 3 columns (1m, 2m, 3m distances) with 60 rows (the measurements you collected every 2sec). Ensure you have a header row explaining the contents of each column (i.e. label your columns). Please upload this sheet to canvas. **(4 points)**

- **b.** Take the average of your RSSI at the locations measured above. Plot average RSSI vs distance in a graph. Label your axes. Upload a PDF of your plot. (**HINT:** make a scatter plot with the distances and the averages). **(4 points)**
- **c.** Describe what you see in the plot. What function would you fit it to if you were to fit a function to it? (min 10 words, max 50 words) **(4 points)**
- **d.** Inspect your RSSI values for each distance, and think about the variations in your RSSI values. Calculate the 25 percentile, median, 75 percentile of the data and create a box plot showing your data for each distance. Upload a PDF with your graph. **(4 points)**
- **e.** Explain the plot. Is there a pattern? If so, what pattern do you see and what do you think is the cause for this? (min 20 words max 50 words) **(4 points)**

(15 points) Part B: RSSI as a function of rotation

Make sure nothing is obstructing the path between the two devices and that the devices are at fixed positions.

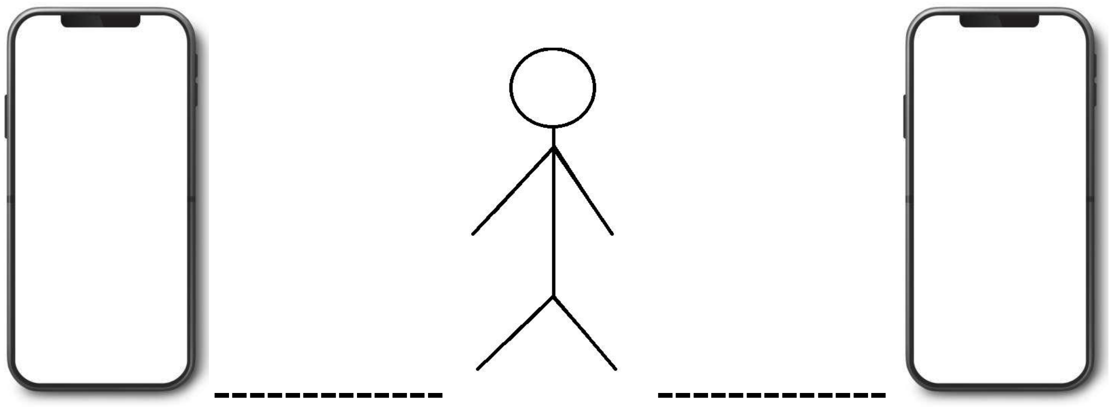


- **a.** Rotate 4 times by 90 degrees (either clockwise or anti-clockwise) across the green axis labeled above (i.e., your phone moves from being parallel to the ground to perpendicular to the ground) to complete an entire circle. Make sure the angle you turn each time is roughly the same (Use the compass app on your phone). Record the RSSI value of your target device every 2s, for at least 2 minutes (≥ 60 data points for each angle position). Please add your measurements to the excel sheet from part A. . Your spreadsheet should look similar to part A, except instead of varying distance you are varying angle. **(5 points)**
- **b.** Plot the average of RSSI value vs every angle position. Label your graph. Upload a PDF of your graph (again, similar to part A). **(5 points)**

c. Explain the plot. Is there a pattern? If so, what pattern do you see and what do you think is the cause for this? (min 20 words max 50 words) **(5 points)**

(15 points) Part C: RSSI Interference

Same setup as experiment A, but make sure you or your teammate (or a big dense object) stand between your scanning device and target BLE device, blocking it.



a. Using the experiment setup above from experiment A, record the RSSI value of your device every 2s, for at least 2 minutes (you should get ≥ 60 data points per each of the 1/2/3m distances). Again, edit your excel spreadsheet to contain your measurements with a distance of 1m, 2m, and 3m between the devices. **(5 points)**
b. Take the average of your RSSI at the locations measured above. Plot average RSSI vs distance in a graph. Label your axes. Upload a PDF of your plot. (**HINT:** make a scatter plot with the distances and the averages). **(5 points)**
c. Compare it with what you see from experiment A. Explain the difference. (min 20 words, max 50 words) **(5 points)**

Task 2 (45 points)

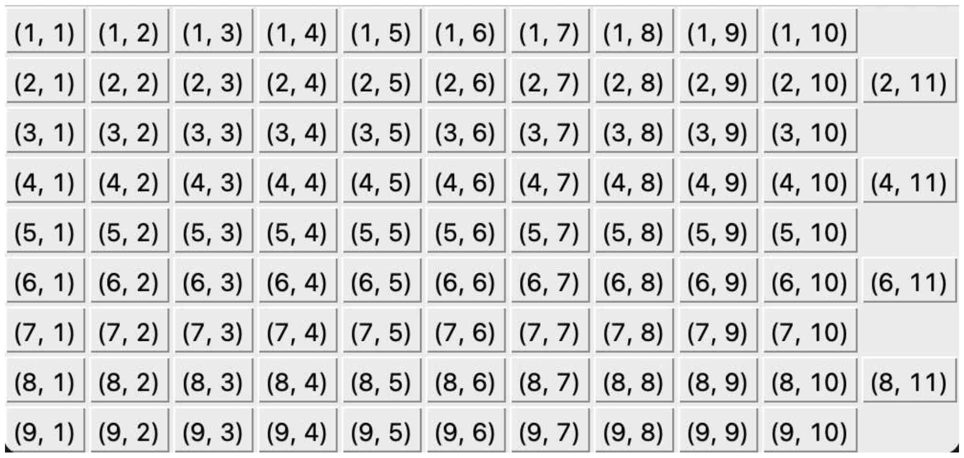
Knowing that there exists a relationship between distance and RSSI values, a handful of RSSI advertisers can be used to localize a device using **fingerprinting**. First, the RSSI values for each of the advertisers must be sampled at different points in space (the fingerprints). Then, these samples, along with their ground truth position values, i.e. the fingerprints, can be used to train a model to predict location.

In this task, you will be training, testing, and comparing RSSI-based fingerprinting for indoor localization.

Instead of predicting location in meters, the model you create will predict which classroom seat you are in ([row, col] as shown below). From the HCI or user-experience perspective, providing the location in meters from a defined origin may not be particularly useful. Instead, we consider the scenario of a classroom with assigned seats, where your model helps check attendance automatically by predicting which seat you and your device are in.

ON-CAMPUS: use the following room schematic to guide your exercise completion:

WhiteBoard



(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)	(1, 7)	(1, 8)	(1, 9)	(1, 10)	
(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)	(2, 7)	(2, 8)	(2, 9)	(2, 10)	(2, 11)
(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)	(3, 7)	(3, 8)	(3, 9)	(3, 10)	
(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)	(4, 7)	(4, 8)	(4, 9)	(4, 10)	(4, 11)
(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)	(5, 7)	(5, 8)	(5, 9)	(5, 10)	
(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)	(6, 7)	(6, 8)	(6, 9)	(6, 10)	(6, 11)
(7, 1)	(7, 2)	(7, 3)	(7, 4)	(7, 5)	(7, 6)	(7, 7)	(7, 8)	(7, 9)	(7, 10)	
(8, 1)	(8, 2)	(8, 3)	(8, 4)	(8, 5)	(8, 6)	(8, 7)	(8, 8)	(8, 9)	(8, 10)	(8, 11)
(9, 1)	(9, 2)	(9, 3)	(9, 4)	(9, 5)	(9, 6)	(9, 7)	(9, 8)	(9, 9)	(9, 10)	

OMS: divide your room into 4 quadrants and place your devices in random spaces around the room (as far from each other as possible).

The GitHub repo for Task 1: https://github.gatech.edu/jwomack30/indoor-localization

(25 points) Part A: Room Mapping & Model Testing

The TAs have already collected a preliminary dataset for experimenting with fingerprinting based localization, which you will do in this task. The TA dataset in file **ta-data.json** (found in the GitHub repo), will be your **training data for Part A**. The **testing data for Part A** will be **25 samples of RSSI readings** from BLE advertisers around the classroom that you will collect now.

To collect your data:

1. On your laptop, copy and paste the underlined link into your google **chrome** browser chrome://bluetooth-internals/#devices. This site allows you to see RSSI values from bluetooth devices around you.
2. Create a new JSON file titled **$gtusername-data.json**, replacing the username with your own. **NOTE:** only make this file the first time you begin collecting data, since all data will go in the same json file.
3. Click the "Start Scan" button in the top right corner and wait >5 seconds. Make sure you see the devices the TAs have set up as beacons.
    a. **ON-CAMPUS:** the TAs will provide you this list in class.
    b. **OMS**: these are your own devices you placed around the room.
4. For each of the BLE beacons set up around the room, scroll through the Chrome page you have open and record the RSSI for each device in the copied doc you made on a sticky note or other notation.
5. In the json doc you edited, add a new object matching the following specs:
    a. If this is the first sample you have collected, add a [ ] to the json file. There should be **only one set of square brackets** in the json file.
    b. Add a new json object matching this form:

```
{x: $x, y: $y, $d1Name: $d1RSSI, $d2Name: $d2RSSI, ..... : $d5Name: $d5RSSI}
```

    c. Replace the following contents:
        i. **$x** with the row coordinate you are at
        ii. **$y** with the column coordinate you are at
        iii. **$d{i}Name** with the i'th BLE beacon
        iv. **$d{i}RSSI** with the RSSI of the i'th BLE beacon (notated earlier)
6. Repeat 24 more times, eventually yielding data from 5 different seats, 5 samples per seat.

Your final json file should look like this, except it contains 25 rows (1 for each sample, 5 samples per seat):

```
[{x: $x, y: $y, $d1Name: $d1RSSI, $d2Name: $d2RSSI, ..... : $d5Name: $d5RSSI},
 {x: $x, y: $y, $d1Name: $d1RSSI, $d2Name: $d2RSSI, ..... : $d5Name: $d5RSSI},
 {x: $x, y: $y, $d1Name: $d1RSSI, $d2Name: $d2RSSI, ..... : $d5Name: $d5RSSI},
                                      .....
 {x: $x, y: $y, $d1Name: $d1RSSI, $d2Name: $d2RSSI, ..... : $d5Name: $d5RSSI}]
```

Getting Model Accuracy

To accurately identify someone's location indoors, we will be creating a model. This implementation uses the K Nearest Neighbors algorithm to achieve this (if you have never heard of this algorithm, here is a good resource explaining the process).

Build a KNN model and test its accuracy on the 25 data points you collected by running knn.py. This script takes in command line arguments for training and testing data, using training data as **ta-data.json** and testing data as your json file. Here is the command to run the script:

```
python3 knn.py --training_data_file /path/to/ta-data.json --testing_data_file
/path/to/gtusername-data.json
```

The script will output the accuracy of the model as a percentage of correct predictions multiple times (varying the acceptable error bound ranging from +/- 0 seats to +/- 5 seats). This script also generates a graph showing error bound on the x axis and its corresponding accuracy on the y axis.

NOTE: for OMS students, you will not have ta-data since you are in your own apartment. Just collect more data points and make your own training data, just like you made your testing data above.

- **a.** Explain the KNN algorithm in brevity; how does it work? What are some limitations? **(10 points)**
- **b.** Upload your graph from knn.py as a PNG file. **(5 points)**
- **c.** What did you observe from the graph? How accurate was your model? Is there any trend you see from the graph? (min 20 words, max 50 words) **(10 points)**

(20 points) Part B: Crowdsourcing **(On-Campus Only)**

As you can imagine, the more data (fingerprints) we have collected the more accurate fingerprinting will be. To verify this, we will pool all student collected data to create a much larger fingerprinting database to use as your training set. For this portion, your **training dataset** will be the **crowdsourced data WITHOUT YOUR DATA** and the **testing dataset** will be **your 25 samples (same as part A)**.

Getting Crowdsourced Model Accuracy

Upload your data points from your json to Microsoft Sharepoint (link provided in class via a Canvas announcement), so the class has a crowdsourced database. Make sure that your filename is formatted as **$gtusername-training-data.json**.

Once all of your classmates have uploaded their data, download the Sharepoint folder as a zip and extract it to your computer. **NOTE:** you must remove your data from this folder before the next step.

Run the **merger.py** script to combine all of these json files into a single large training json file:

```
python3 merger.py --json_folder /path/to/sharepoint/folder
```

Rerun the KNN script from before, passing in the new training data file:

```
python3 knn.py --training_data_file /path/to/merged_data.json --testing_data_file
/path/to/gtusername-data.json
```

- **a.** Upload your new graph from knn.py as a PNG file. **(5 points)**
- **b.** Compare the crowdsourced graph with that from part A. Is the accuracy better or worse? Explain your results. If the accuracy is better, explain why this would be. If it is the same (or worse), explain factors which could have influenced this result. **(10 points)**
- **c.** Rerun the KNN script again on the crowdsourced data, this time adjusting k (the number of neighbors to consider):

```
python3 knn.py --training_data_file /path/to/merged_data.json --testing_data_file
                    /path/to/gtusername-data.json --k $new_k
```

What do you observe? Discuss the results. **(5 points)**