CptS 322- Software Engineering I

Project Title: "Smart Parking"

I. Introduction

The project is designed to provide a platform on which you can excise software design and development methodologies in Software Engineering. Smart parking is one new challenge related to space utilization efficiency. Several studies demonstrate that parking available to informed drivers could reduce congestion and emissions [4].

Your project will be to design, document, build, and release a standalone, desktop Smart Parking Simulator System (SPSS) that visualizes parking lot availability, using as data source the Bluetooth proximity-enabled cars (see Figure), in this project we will have a set of 4 sensors located strategically on the parking lot. You will design and implement the SPSS client to display the parking lot availability in real-time. The outcomes of the project will be demonstrated not just by the quality of your design and the final product, but also by the quality of the software process and the design documents developed during the course of the project.

Figure 1, parking lot representation.

II. Due Dates and Deliverables:

Deliverables will be done with four assignments.

- Assignment 1 – System Description
  - Format: Word Doc or PDF
  - Due: 2/8 – 4:15 PM
  - Submission: Canvas

- Assignment 2 – System Requirements
  - Format: Word Doc or PDF
  - Due: 3/8 – 4:15 PM
  - Submission: Canvas

- Assignment 3 – System Design Document
  - Format: Word Doc or PDF
  - Due: 3/29 – 4:15 PM
  - Submission: Canvas

- Assignment 4 – Working Code
  - Format: C++ Source Files
    - Must use Object Oriented Programming (i.e. classes)
  - Due: 4/21 11:59 PM
  - Submission: GitHub Classroom using git (we will discuss this in class)

- Each assignment has a template that you are required to follow. It is up to you to ask questions and seek clarification about the sections of each template.
- no extra credit (at this time) are allowed. You can submit as many times as needed up until the deadline.

Working Together?

Assignments 1-3:

You can work on a whiteboard together and discuss requirements of the game, user stories, etc. In fact I encourage you to do so. However, you must write your own documentation. Do not copy each other's work

Assignment 4:

You may help each other at the whiteboard, brief debugging (e.g. getting started with ncurses), and YOU MAY share a few lines of code for ncurses (e.g. colors) – but you must write your own code. Do not copy classes or large blocks of code. If you are not sure about what dictates copying etc. please talk to me.

III. Required Features

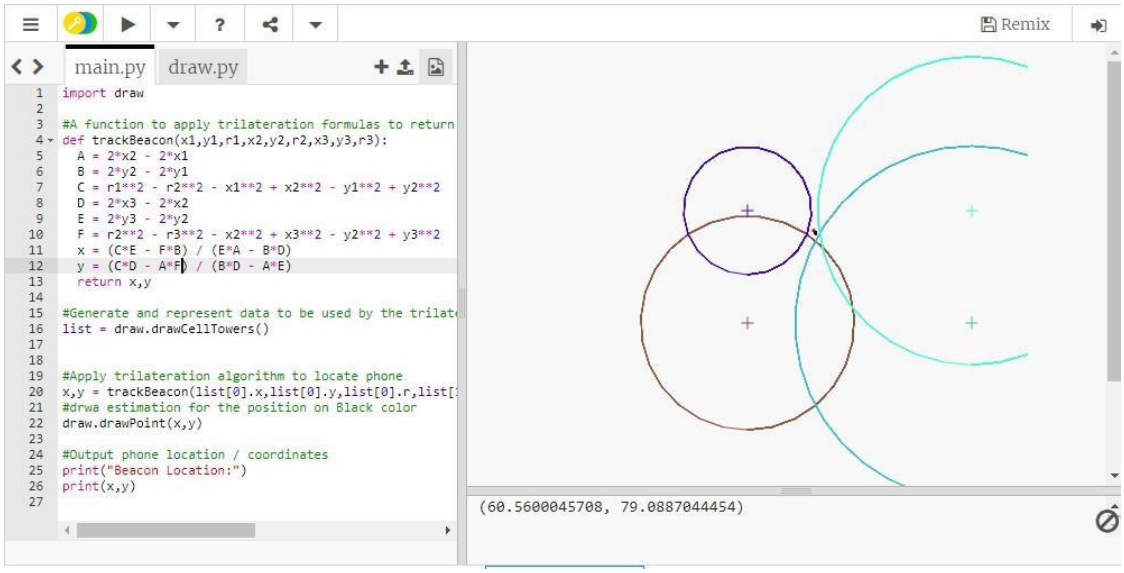
This handout describes how the SPSS will process the data collected from the sensor, how to process and display the available parking spot. Your work includes a parking lot map to visualize the parking spots and the availability on real time.

The SPSS gets the beacons data set from the common database, selects the convenient data, calculate the occupied parking spots, displays this in the map and send the list to the master server. These beacons are a set of Bluetooth enable devices with a unique identifier attach to a unique vehicle. Typically, each beacon has a set of values for the distances to each sensor called D1, D2, D3, D4. Each sensor is geolocated in the corned of the parking lot, with x and y coordinates. The data set is formatted in JSON format, using the common standard for JSON.

Overview & Requirements:

This project is very open-ended; you are free to develop your SPSS client with your own choice of features. Nevertheless, a base set of features is required:

a) Parking slot activation. To declare a parking spot occupied a carefully data processing must be performed. Is necessary to select the nearest parking spot in relation with the estimated x, y position [1]. Each beacon contain a 4 values that change in real time, these values must be used to calculate the relative x and y possible position, see the image.



b) adding to this calculation, you must develop a strategi to estimate an average for the values D1, D2, D3, D4, these values change every 500ms.

2. **Displays. SPS**S offers at least two interfaces to interact with the system process:

a) Main interface: This is the main interface to present (in real time) the status of the parking lot (available , occupied, reserve, delinquency). This interface offers a simple data visualization in order to review the data by status.

b) Login interface: a simple login with username and password

c) Simulation

When the sensor star sending data and the values D1, D2, D3, D4 star changing, the simulation start. The parking spots are empty at the beginning. The initial sensor position is a Json file provide by server. The interaction between free and occupied follow as simple binary model, free (0) and occupied (1).

"as parking spot is occupied is there a beacon with estimated coordinates x and y are inside the 70% center of the parking spot"

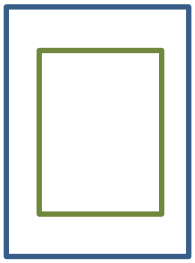


Figure 2, The green represent the 70% of the parking slot and is occupied if the x, y coordinates are inside the green portion.

3. Data and Optimization.

You need to build a local database or used local firebase to store your data and parking lot information. Your optimization (heuristic) algorithm will search the best patter to estimate the 4 diatnces D1, D2, D3, D4 on each beacon. More details on average the data can be found on [2]

Libraries:
For the interface, the SPSS client will integrate the graphic library**,** A sample code for using this will be attach on Milestone 1.

For the database, the SPSS client will integrate a firebase library **Firebase C# library**. This library is described in the creator web page (https://github.com/step-up-labs/firebase-database-dotnet) and use the following command to install Install-Package

```
Install-Package FirebaseDatabase.net
```

IV. References

[1] Cantón Paterna, Vicente, Anna Calveras Augé, Josep Paradells Aspas, and María A. Pérez Bullones. 2017. "A Bluetooth Low Energy Indoor Positioning System with Channel Diversity, Weighted Trilateration and Kalman Filtering" *Sensors* 17, no. 12: 2927. https://doi.org/10.3390/s17122927

[2] http://mdh.diva-portal.org/smash/get/diva2:1075574/FULLTEXT01.pdf

[3] Greatmaps GMap.Net examples https://github.com/radioman/greatmaps

[4] Sayarshad, Hamid R., Shahram Sattar, and H. Oliver Gao. "A scalable non-myopic atomic game for a smart parking mechanism." *Transportation Research Part E: Logistics and Transportation Review* 140 (2020): 101974.

[5] Firebase C# library https://medium.com/step-up-labs/firebase-c-library-5c342989ad18

[6] ECMA-404 The JSON https://www.json.org/