****

**Application Developing Guide**

**for Indoor Location Positioning**

Sarnava Konar

Mau Cisneros

Farhan Zaki

Contents

[How to develop a cross platform app 3](#_Toc15466351)

[Setting up the coding environment 5](#_Toc15466352)

[Code for the index.html file 10](#_Toc15466353)

[Mapping of the interest area 15](#_Toc15466354)

[Code for the main script file app.js 19](#_Toc15466355)

[General information regarding the project 59](#_Toc15466356)

[Important links for the project: 64](#_Toc15466357)

# How to develop a cross platform app

**Create a folder with following folders in it:**

**Tools:**

Include all sources for tools

**Documents:**

Manuals, user guides, technical documents, this help

**Source Code:**

Source codes with date and small readme file on different versions

Final version must be fine and easy to run on at least one android device with clear explanations

**Hardware:**

1. **Any Bluetooth enables device,** we are using BLE beacons (iBeacon from Apple <https://www.beaconzone.co.uk/blog/tag/pc061/> and https://www.beaconzone.co.uk/PC062)

**Associated Software:**

Settings on the Android cell phones to switch to developer mode.

**To be able to upload to the cell phone the following steps need to be done at cell phone settings:**

1. If using stock Android, go to *Settings > About phone > Build number. On a*[*Samsung Galaxy*](http://www.samsung.com/us/mobile/galaxy/)*device, go to Settings > About device > Build number. On an* HTC device, go to *Settings > About > Software information > More > Build number*. On an LG device, go to *Settings > About phone > Software info > Build number*.
2. Tap *Build number* seven times. After the first few taps, you should see the steps counting down until you unlock the developer options. Once activated, you will see a message that reads, “You are now a developer!”
3. Go back to *Settings*, whereyou’ll find a *Developer options* entry in the menu.
4. Settings > developer options > *debugging section* – turn on *USB debugging*
5. **A USB cable**: to be able to connect cell-phone to our laptop for programming

**Tools / Softwares to install:**

Before starting, it is necessary to have the following tools: **(Steps to install)**

1. Visual Studio Tools 2017 for Apache Cordova – Cordova from

Visual Studio Tools for Apache Cordova is installed using the standard Visual Studio installer which you can download from the [Microsoft Download Center](https://aka.ms/vs/15/release/vs_enterprise.exe). Refer to [Install Visual Studio Tools for Apache Cordova](https://docs.microsoft.com/en-us/visualstudio/cross-platform/tools-for-cordova/first-steps/installation?view=toolsforcordova-2017) for complete installation instructions.

*Add “Data Storage and Processing” to have access to SQL server tools.*

*Add “Git for windows” – this can be installed from “Plugins > custom > Git” in visual studio*

1. Evothings Examples (to start with a sample project from the examples)

<https://github.com/evothings/evothings-examples>

*Downloaded as “evothings-examples-master.rar” in the Tools folder*

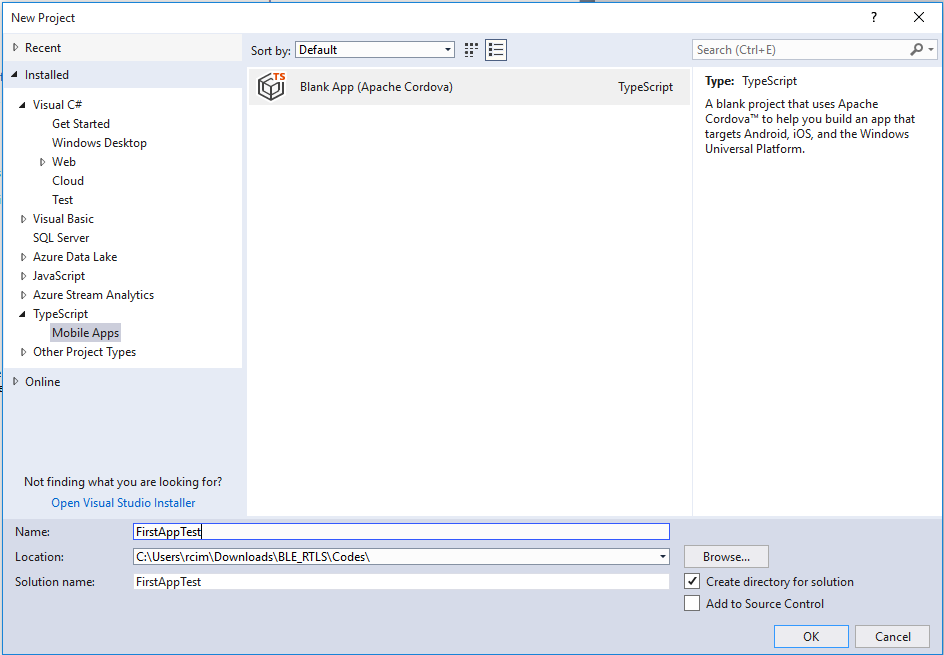
References:

* **Tutorial slides from Dr. Rashidzadeh**
* **Sample code to upload on the device (Android)**

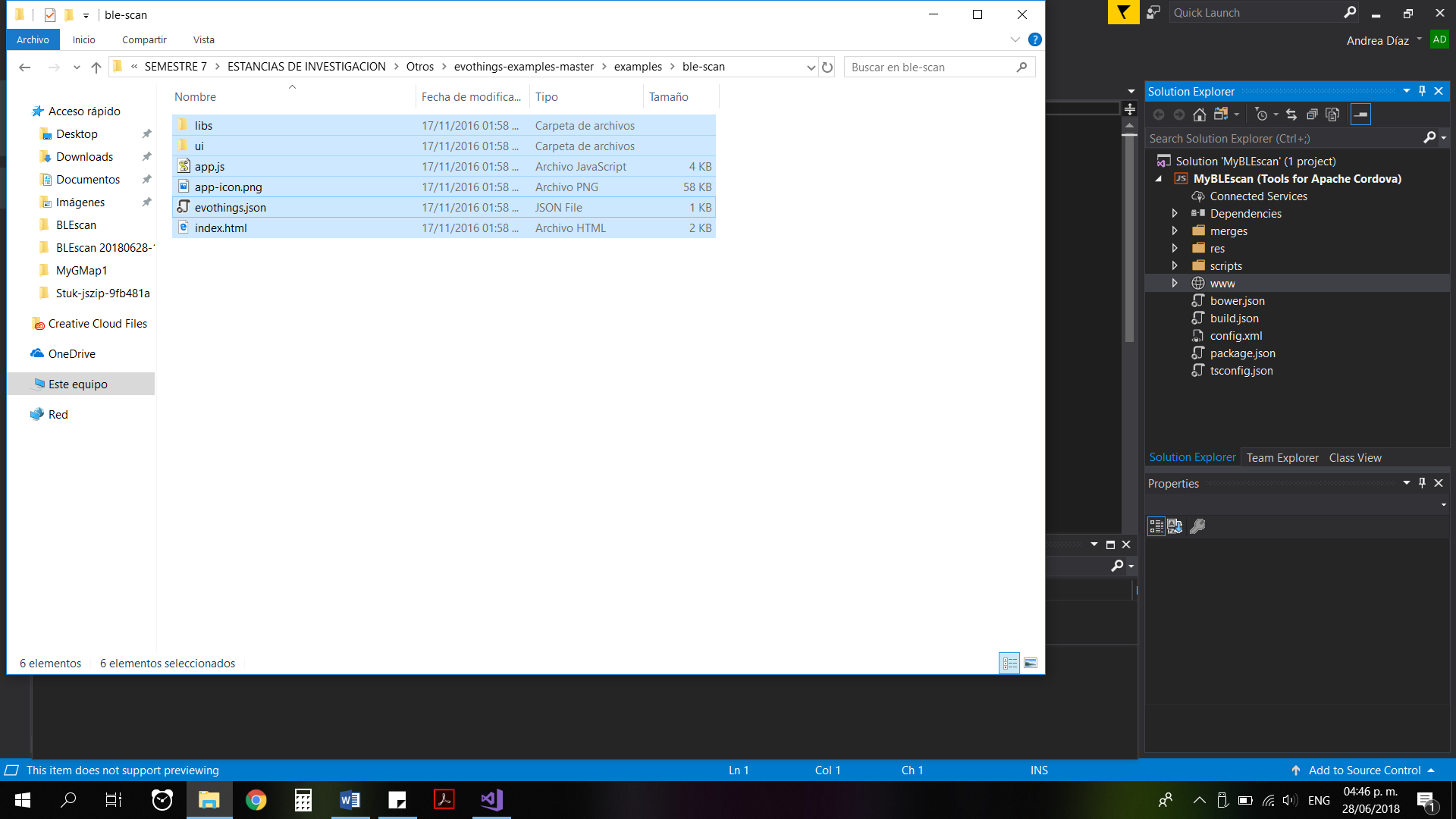
# Setting up the coding environment

Once we have everything installed, we open Visual Studio and

* create a new project TypeScript\Mobile Apps



* Copy and paste all the content that is in evothings-examples-master\examples\ble-scan to www folder in the project.



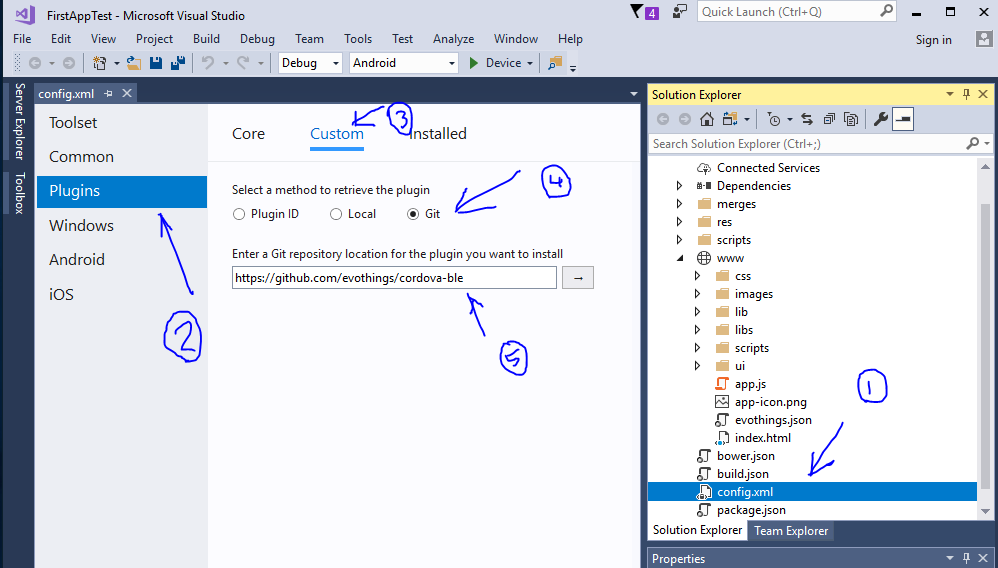
From there we start to modify and customize our application.

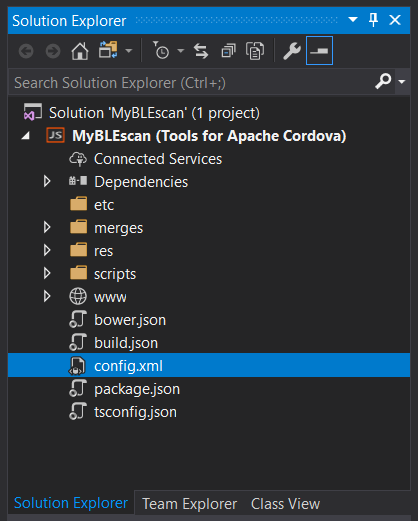
**Install the following plugins**

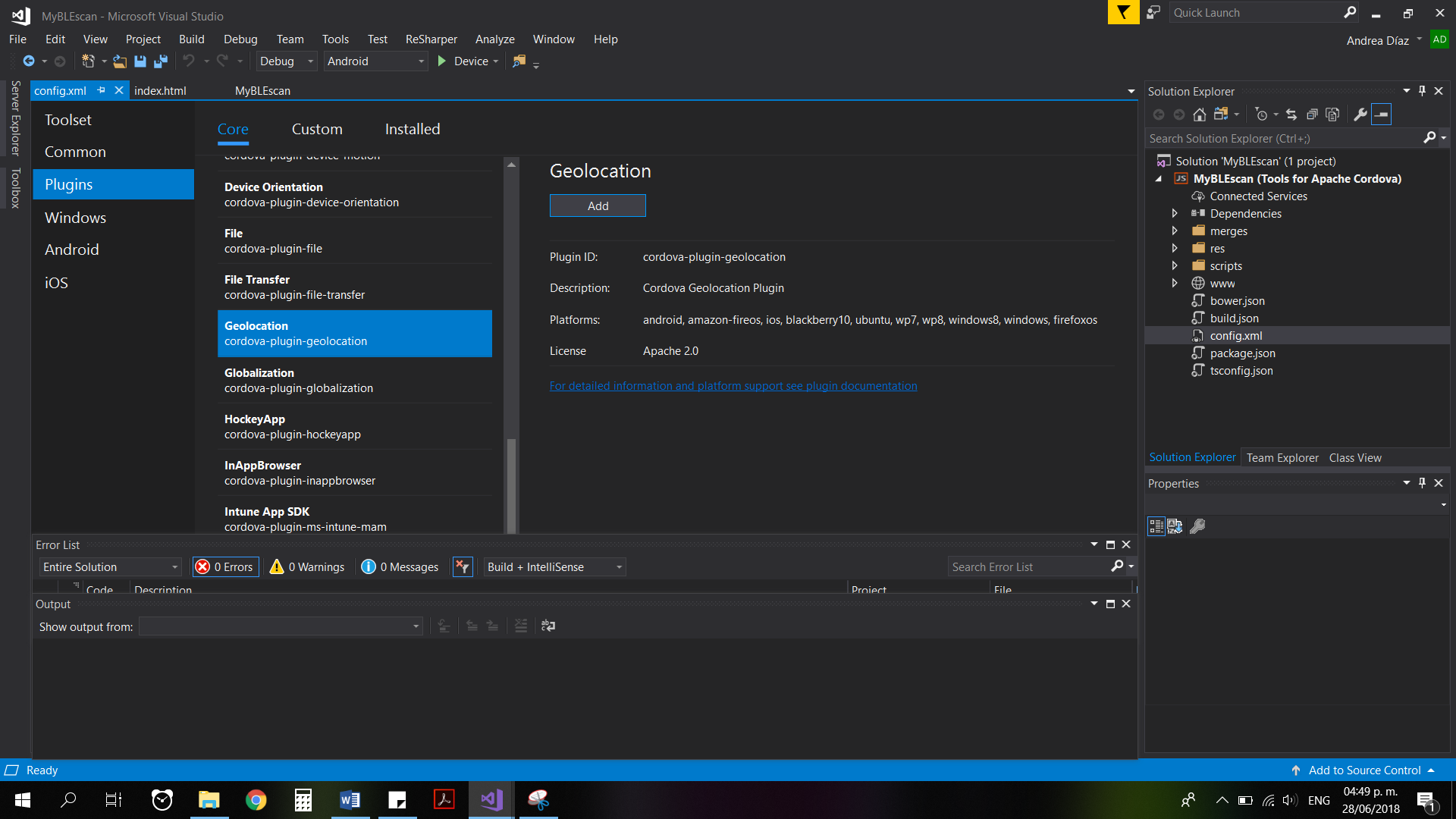
* <https://github.com/evothings/cordova-ble> - Offline file “cordova-ble-master.rar” is uploaded in *Tools* folder.

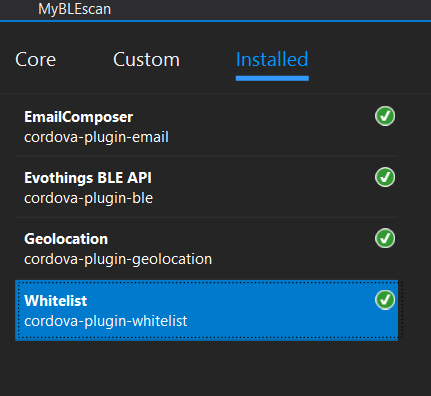
With this device has access to manage the BLE sensor

This can be installed from Visual Studio > config.xml > Plugins > Custom > Git > <https://github.com/evothings/cordova-ble> > Press Add

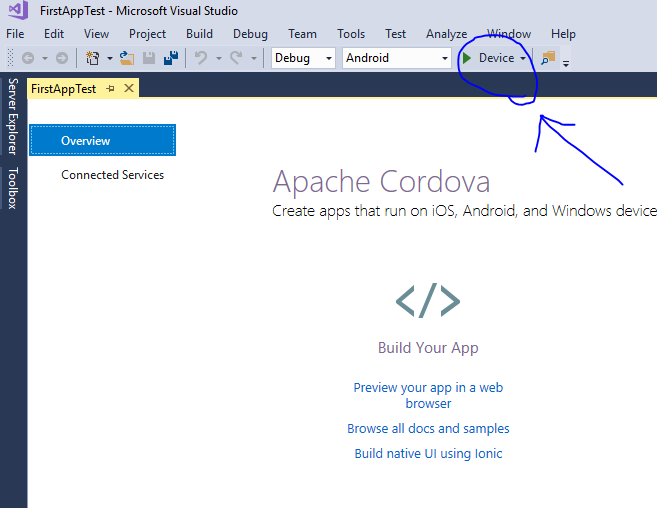






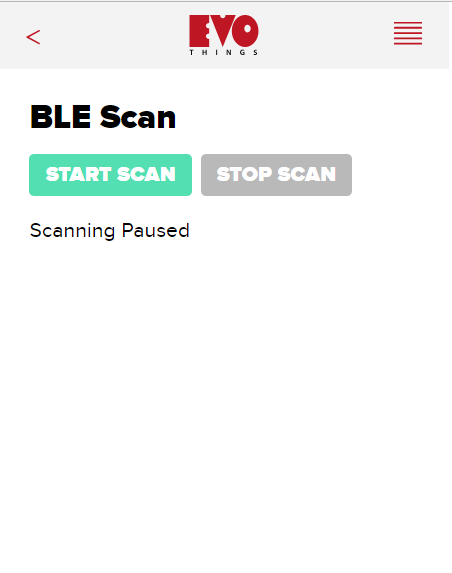


By running this application, we can upload first example on the device by running the app on the device as shown in the figure below



The permission to run the app on the device is required at first time

Start to modify the appearance of the application that up to now is like this:



# Code for the index.html file

**The code for all the files has been neatly indented and they have been commented appropriately at places to tell us what the individual functions do. Also, during the initialization of the variables, I have mentioned what is their use properly. I have deliberately left in the app the commented sections of the code as it will help us later to understand how we gradually built the app and later help us in debugging code at later stages of development.**

We change all the contents of index.html for the following:

|  |
| --- |
| <!-- This is where the ui of the app is written in normal html and javascript -->  <!DOCTYPE html>  <html>  <head>  <style>  @import 'ui/css/evothings-app.css';  </style>  <style>  .button\_yellow {  background-color: #ffd800; /\* Yellow \*/  border: none;  color: white;  padding: 20px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 24px;  margin: 10px 6px;  cursor: pointer;  border-radius: 8px;  }  .button\_red {  background-color: #f44336; /\* Red \*/  border: none;  color: white;  padding: 20px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 24px;  margin: 10px 6px;  cursor: pointer;  border-radius: 8px;  }  .button\_black {  background-color: #000000; /\* Black \*/  border: none;  color: white;  padding: 20px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 24px;  margin: 10px 6px;  cursor: pointer;  border-radius: 8px;  }  .button\_blue {  background-color: #00ffff; /\* Blue \*/  border: none;  color: white;  padding: 20px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 24px;  margin: 10px 6px;  cursor: pointer;  border-radius: 8px;  }  .button\_orange {  background-color: #ff6a00; /\* Orange \*/  border: none;  color: white;  padding: 20px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 24px;  margin: 10px 6px;  cursor: pointer;  border-radius: 8px;  }  </style>  <script>  if (window.hyper && window.hyper.log) { console.log = hyper.log }  </script>  <script src="cordova.js"></script>  <script src="libs/jquery/jquery.js"></script>  <script src="libs/evothings/evothings.js"></script>  <script src="libs/evothings/ui/ui.js"></script>  <script src="app.js"></script>  </head>  <body ontouchstart="">  <header>  <button class="back">  <h2>UWINDSOR IPS</h2>  </button>  <!--  <img class="logotype" src="ui/images/logo.png" alt="" />  <button id="menu-button" tabindex="0" onclick="this.focus()">  <img src="ui/images/menu.svg" />  </button>  -->  <!--  <menu>  <menuitem>  <a onclick="window.open('index-map.html')">Future Options</a>  </menuitem>  </menu>  -->  </header>  <div style="text-align: center">  <h1><b>Indoor Positioning System</b></h1>  </div>  <div style="text-align: center">  <!--This button will activate the scanning and read the RSS values from all the beacons-->  <button onclick="app.ui.onStartScanButton()" class="button\_yellow">  CLICK TO ACTIVATE  </button>  </div>  <!--  <button onclick="app.ui.onStopScanButton()" class="bg\_black">  STOP CALC  </button>  <button onclick="app.ui.onResetScanButton()" class="green">  REFRESH  </button>  <button onclick="showhidedlfn()" class="bg\_black">  S/H DL  </button>  <button onclick="showRSStable()" class="green">  S/H MAT  </button>  <button onclick="calcRSSmw()" class="bg\_black">  CALC RMW  </button>  <button onclick="showRSSmw()" class="green">  S/H RMW  </button>  <button onclick="calcsumcol()" class="bg\_black">  CALC SCOL  </button>  <button onclick="showsumcol()" class="green">  S/H SCOL  </button>  <button onclick="calcwttable()" class="bg\_black">  CALC WTAB  </button>  <button onclick="showweighttable()" class="green">  S/H WTAB  </button>  <button onclick="calcxy()" class="bg\_black">  CALC XY  </button>  <button onclick="showxy()" class="green">  S/H XY  </button>  <button onclick="calcroom()" class="bg\_black">  CALC ROOM  </button>  <button onclick="showroom()" class="green">  S/H ROOM  </button>  <button onclick="calcmap()" class="bg\_black">  CALC MAP  </button>  <button onclick="showmap()" class="green">  S/H MAP  </button>  -->  <div style="text-align: center">  <!--This button will do all the calculations and return me my position-->  <button onclick="doallcalc()" class="button\_blue">  WHERE AM I???  </button>  </div>  <div style="text-align: center">  <!--This button will calculate the moving average-->  <button onclick="calcmovavg()" class="button\_black">  CALC MA  </button>  </div>  <!--  <div style="text-align: center">  <button onclick="showfinalresult()" class="bg\_red">  S/H RESULTS  </button>  </div>  -->  <div style="text-align: center">  <!--This button will reset everything for the fresh reading!!-->  <button onclick="app.ui.onResetScanButton()" class="button\_red">  REFRESH  </button>  </div>  <div style="text-align: center">  <!--This button will reset everything after we get the moving average!!-->  <button onclick="refreshmovavg()" class="button\_orange">  REFRESH MA  </button>  </div>  <p id="scan-status">Click <b>CLICK TO ACTIVATE</b> button to help you!!</p>  <!-- This div shows the dynamic list of the beacons which are in our area. I have hided it for the final version as we dont need to show it-->  <div id="showhidedl" style="display: none">  <ul id="found-devices" class="dynamic"></ul>  </div>  <hr />  <!--All these divs show the different portions of the app which they are supposed to show at different points of time. They have been named according to the purpose they serve.-->  <div id="matrixdiv">  <p id="matrixshow"></p>  </div>  <div id="mwrssdiv">  <p id="mwrssshow"></p>  </div>  <div id="sumofcoldiv">  <p id="sumofcolshow"></p>  </div>  <div id="weighttable">  <p id="showweighttable"></p>  </div>  <div id="showingxy">  <p id="showxy"></p>  </div>  <div id="showingroom">  <p id="showroom"></p>  </div>  <div id="showingmap">  <p id="showmap"></p>  </div>  <div id="showingfinalresult">  <p id="showfinalresult"></p>  </div>  <div id="showingmovingaverage">  <p id="showmovavg"></p>  </div>  </body>  </html> |

# Mapping of the interest area

**Before I move on to document the other files of the code, I will like to tell you about how the mapping of the building was done. This was one of the most important tasks of our project as wrong mapping would later lead to wrong results. For our project purpose we chose a portion of the second floor of the CEI building. The way it was mapped is discussed in the following few paragraphs.**

**Mapping**

The mapping process started by measuring the lengths of a part of the second floor of the Centre for Engineering Innovation (CEI) building at the University of Windsor. The measurements included the lengths and widths of the corridors, along with the rooms in each one of them. The area where the system operated was from 0.1 to 27.6 m in the X-axis and from 0.1 to 64 m in the Y-axis.

In order to detect the position of the user through the application, the displacement of beacons along the corridors was needed. The models of the beacons used are PC037 and PC038, which use the microcontroller CC2541 from Texas Instruments. These beacons use Bluetooth Low Energy (BLE) technology to transmit the information they store. The position of each beacon was defined by a circular system configuration, which also achieved the formation of triangles along the corridors. The purpose of the configuration was the proper transmission of the signals of each beacon, so that in any place along the corridor a signal would be received by the user. Figure 1 shows the configuration of the beacons in the second floor of CEI. In Table 1, the coordinates of each beacon are found.

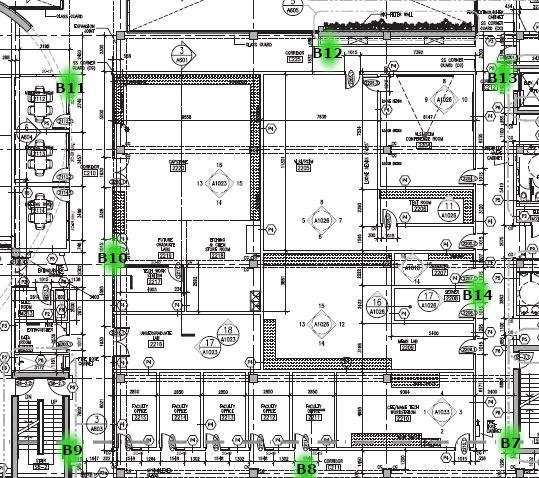


Figure 1. Position of the beacons.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Model* | *Name in map* | *Name programmed* | *X* | *Y* |
| PC038 | B7 | 20\_350 | 2 | 35 |
| PC037 | B8 | 1\_479 | 1 | 47.9 |
| PC038 | B9 | 20\_640 | 2 | 64 |
| PC037 | B10 | 143\_610 | 14.3 | 61 |
| PC038 | B11 | 247\_640 | 24.7 | 64 |
| PC037 | B12 | 276\_475 | 27.6 | 47.5 |
| PC038 | B13 | 261\_350 | 26.1 | 35 |
| PC037 | B14 | 146\_373 | 14.6 | 37.3 |

Table 1. Coordinates of the beacons.

The beacons were programmed with their names as their coordinates in a *x-axis-coordinate\_y-axis-coordinate* fashion. To avoid the use of decimal digits in the name of the beacons, a multiplication by 10 was made in each coordinate. In the app, a division by 10 takes place in order to obtain the original values of the coordinates. Also, the transmission period for every beacon was defined as 100 ms, which is the minimum the models used allowed.

By identifying the position of the rooms, areas of recognition were assigned by determining four coordinates, creating a rectangle along the corridor. These areas of recognition were defined by how big each room was and where the user should be informed of its proximity. If the user is inside the rectangle, the app would show through a list the rooms that belong to that rectangle. The list would have as a first item the closest room to the position detected, the next item would be the second closest room, and so on. The rooms considered in the app are shown in Figure 2.

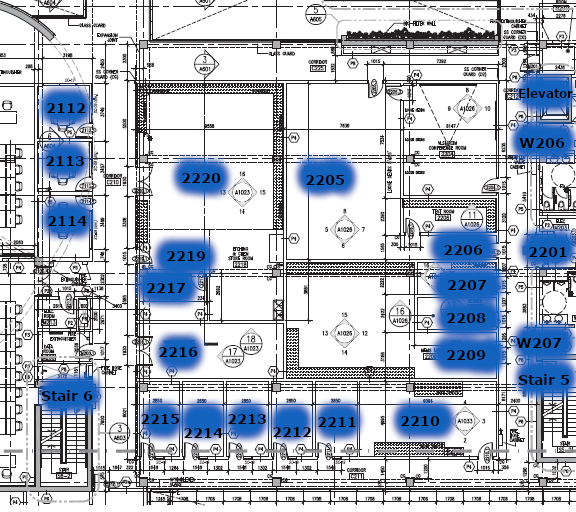


Figure 2. Rooms of the second floor of CEI considered in the app.

To test the system, the app was used in 27 points of the second floor of CEI. In Figure 3, these points are shown and in Table 2, the coordinates for each are displayed.

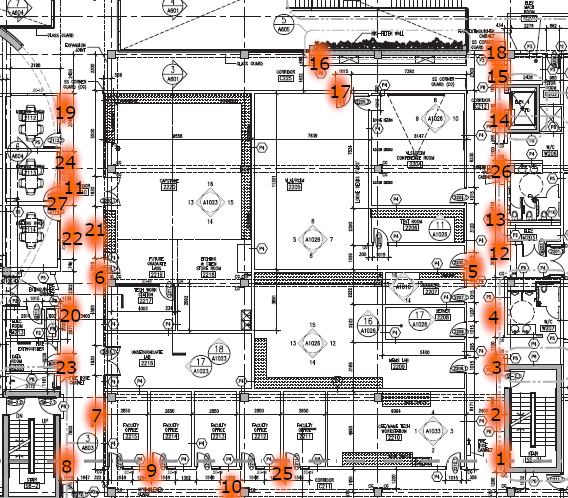


Figure 3. Position of the measurements taken.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Number* | *X* | *Y* | *Number* | *X* | *Y* | *Number* | *X* | *Y* | *Number* | *X* | *Y* |
| 1 | 2 | 35 | 8 | 2 | 63.5 | 15 | 26.1 | 35.5 | 22 | 17.8 | 63 |
| 2 | 6 | 35.5 | 9 | 1.6 | 55.6 | 16 | 27.6 | 47.5 | 23 | 9.5 | 63 |
| 3 | 9.37 | 35.4 | 10 | 0.5 | 47.9 | 17 | 26.5 | 45.7 | 24 | 22.4 | 63 |
| 4 | 12 | 36 | 11 | 20.1 | 61.5 | 18 | 27.6 | 35 | 25 | 1.4 | 45.5 |
| 5 | 15 | 37.3 | 12 | 16.4 | 36 | 19 | 24.7 | 64 | 26 | 21 | 35.4 |
| 6 | 14.3 | 61.5 | 13 | 17 | 36.3 | 20 | 11.83 | 63 | 27 | 20.1 | 62 |
| 7 | 6.5 | 62 | 14 | 24.6 | 35.5 | 21 | 17.8 | 62 |  |  |  |

Table 2. Coordinates of the measurements taken.

# Code for the main script file app.js

Finally, the main document **“app.js”**

|  |
| --- |
| // This is the real script which does all our work!!!!!!  // Defining the variables  var app = {};  app.devices = {};  app.ui = {};  app.ui.updateTimer = null;  // These variables store the average the beacons coordinate which are deployed(active) for the normalization  var avgbx = 11.971;  var avgby = 49.171;  // The beacons with MAC till 8C were registered in the last year and now I have registered the rst of the MACs for the implementation of the IPS  var beaconsMac = ['D4:F5:13:FF:11:4C', '20:C3:8F:E0:83:5B', '20:C3:8F:E0:90:9A' /\*'7C:EC:79:E0:20:24'/\*'D4:F5:13:FE:81:6D'\*/, '20:C3:8F:E0:90:8C', '7C:EC:79:3C:8E:86', '7C:EC:79:3C:93:F6',  '7C:EC:79:3C:A4:A9', '7C:EC:79:3D:21:1A', '7C:EC:79:3D:A5:95', '7C:EC:79:3C:F6:9E', '7C:EC:79:3C:A4:D9', '7C:EC:79:3D:BD:1D', '7C:EC:79:3D:A0:11',  'EC:11:27:29:B1:8F', '20:C3:8F:E0:90:9D'];  // The use of this variable is documented in the documentation  // The sampleMatrix has the samples of the RSS readings of the beacons in dBm  var sampleMatrix = [[], [], [], [], [], [], [], [], [], [], [], [], [], [], []];  // The use of this variable is documented in the documentation  var init = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3];  // Initializing the sampleMatrix matrix with all zeroes  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  sampleMatrix[i][j] = 0;  }  }  // The mwrss has the samples of the RSS readings of the beacons in milli watts and that is why the name is like that  var mwrss = [[], [], [], [], [], [], [], [], [], [], [], [], [], [], []];  // Initializing the sampleMatrix matrix with all zeroes  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  mwrss[i][j] = 0;  }  }  // This will have the sum of all the beacon readings at a particular time ie of a column  var sumofbeaconsreadingsincol = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  // This matrix will contain the weights of the various beacons at their respective times. This will be a 15\*23 matrix  var weighttable = [[], [], [], [], [], [], [], [], [], [], [], [], [], [], []];  // Initializing the weighttable matrix with all zeroes  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  weighttable[i][j] = 0;  }  }  // This array will have the 20 probable Xs  var arrx = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  // This array will have the 20 probable Xs  var arry = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  // This variable will store the avg X for a position  var xx = 0;  // This variable will store the avg Y for a position  var yy = 0;  // This map will store the scores for the rooms for the first algo  const m = new Map();  // This map will store the scores for the rooms for the second algo  const m2 = new Map();  // This map will store the scores for the rooms for the first algo for the moving avg coord  const m3 = new Map();  // This map will store the scores for the rooms for the second algo for the moving avg coord  const m4 = new Map();  // This variable will store the min of the score which will be attained by the rooms  var min = 1000000;  // This array will store the scores of all the rooms for the first algo  var scorearr = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  // This array will store the scores of all the rooms for the second algo  var scorearr2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  // This variable will store the optimum room which has been found  var room = -1;  // This variable is used to hold the scores of the individual rooms  var score = 0;  // This variable will store the count of the no of rooms that we want to display in our result  var count = 0;  // This array will have the 4 readings for the moving avg of the x coordinates for the first algo  var xmovavg = [0, 0, 0, 0];  // This array will have the 4 readings for the moving avg of the y coordinates for the first algo  var ymovavg = [0, 0, 0, 0];  //This variable will be used to calculate the moving average for the first algo  var indexmovavg = 0;  // This array will have the 4 readings for the moving avg of the x coordinates for the second algo  var xmovavgsecond = [0, 0, 0, 0];  // This array will have the 4 readings for the moving avg of the y coordinates for the second algo  var ymovavgsecond = [0, 0, 0, 0];  //This variable will be used to calculate the moving average for the second algo  var indexmovavgsecond = 0;  // These variables will store the result of our standard deviation of the coordinates  var sdx = 0;  var sdy = 0;  // These variables will contain the extremes for our standard deviation algorithm  var maxsdx = 0;  var minsdx = 0;  var maxsdy = 0;  var minsdy = 0;  //These variables will hold the new average we have found out using the second algorith of using standard deviation  var newavgx = 0;  var newavgy = 0;  // This is the data of the POI points in the  /\*  Room no Xmin Xmax Ymin Ymax  2106 7.97 11.86 85.46 85.46  2107 0.1 4.03 85.46 85.46  2108 4.04 7.97 85.46 85.46  2109 14.37 18.97 85.46 85.46  2110 18.97 22.53 85.46 85.46  2111 22.53 26.02 85.46 85.46  2112 23.38 26.85 62.59 65.84  2113 19.9 23.38 62.59 65.84  2114 14.61 19.9 62.59 65.84  2201 14.35 21.71 33.65 33.65  2203 0.1 10 0.1 30  2205 13.34 20.00 35.92 35.92  2207 11.09 13.34 35.92 35.92  2208 8.77 11.09 35.92 35.92  2209 1.94 8.77 35.92 35.92  2210 0.1 4 34 43  2211 1.64 1.64 46.1 48.94  2212 1.64 1.64 48.94 51.8  2213 1.64 1.64 51.8 54.66  2214 1.64 1.64 54.66 57.31  2215 1.64 1.64 57.31 59.34  2216 7.71 14.28 59.34 59.34  2219 14.28 21.15 59.34 59.34  2220 21.15 26.06 59.34 59.34  2105B 26.85 26.85 73.14 81.94  2105C 26.85 26.85 65.84 73.14  2105D 12.31 14.61 62.59 65.84  2105A 11.86 14.37 62.59 85.46  Elevator(2nd floor) 24.9 25.9 33.65 33.65  Stair 5 7.06 8.28 33.65 33.65  Stair 6 5.36 6.55 62.59 62.59  Stair 7 0.1 0.1 88.01 89.1  Washroom W206 21.71 24.9 33.65 33.65  Washroom W207 9.23 14.35 33.65 33.65  THESE DATAS ARE FOR FINETUNING PURPOSE. THEY ARE ALL OF ROOM 2203  SECTION 1 0.1 5 0.1 2.60  SECTION 2 5 10 0.1 2.60  SECTION 3 0.1 5 2.60 8  SECTION 4 5 10 2.60 8  SECTION 5 0.1 5 8 12.2  SECTION 6 5 10 8 12.2  SECTION 7 0.1 5 12.2 16.5  SECTION 8 5 10 12.2 16.5  SECTION 9 0.1 5 16.5 20.7  SECTION 10 5 10 16.5 20.7  SECTION 11 0.1 5 20.7 25  SECTION 12 5 10 20.7 25  SECTION 13 0.1 5 25 31  2203 0.1 3 31 31  \*/  // This matrix will store the Xmin, Xmax, Ymin, Ymax for all the POIs of our project  var roommatrix = [[], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [],  [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], []];  // Storing the data in the roommatrix matrix  //2106  roommatrix[0][0] = 2106;  roommatrix[0][1] = 7.97;  roommatrix[0][2] = 11.86;  roommatrix[0][3] = 85.46;  roommatrix[0][4] = 85.46;  //2107  roommatrix[1][0] = 2107;  roommatrix[1][1] = 0.1;  roommatrix[1][2] = 4.03;  roommatrix[1][3] = 85.46;  roommatrix[1][4] = 85.46;  //2108  roommatrix[2][0] = 2108;  roommatrix[2][1] = 4.04;  roommatrix[2][2] = 7.97;  roommatrix[2][3] = 85.46;  roommatrix[2][4] = 85.46;  //2109  roommatrix[3][0] = 2109;  roommatrix[3][1] = 0.1;  roommatrix[3][2] = 4.03;  roommatrix[3][3] = 85.46;  roommatrix[3][4] = 85.46;  //2110  roommatrix[4][0] = 2110;  roommatrix[4][1] = 18.97;  roommatrix[4][2] = 22.53;  roommatrix[4][3] = 85.46;  roommatrix[4][4] = 85.46;  //2111  roommatrix[5][0] = 2111;  roommatrix[5][1] = 22.53;  roommatrix[5][2] = 26.02;  roommatrix[5][3] = 85.46;  roommatrix[5][4] = 85.46;  //2112  roommatrix[6][0] = 2112;  roommatrix[6][1] = 23.38;  roommatrix[6][2] = 26.85;  roommatrix[6][3] = 64;  roommatrix[6][4] = 64;  //2113  roommatrix[7][0] = 2113;  roommatrix[7][1] = 19.9;  roommatrix[7][2] = 23.38;  roommatrix[7][3] = 64;  roommatrix[7][4] = 64;  //2114  roommatrix[8][0] = 2114;  roommatrix[8][1] = 14.61;  roommatrix[8][2] = 19.9;  roommatrix[8][3] = 64;  roommatrix[8][4] = 64;  //2201  roommatrix[9][0] = 2201;  roommatrix[9][1] = 14.35;  roommatrix[9][2] = 21.71;  roommatrix[9][3] = 35;  roommatrix[9][4] = 35;  //2203  roommatrix[10][0] = 2203;  roommatrix[10][1] = 0.1;  roommatrix[10][2] = 10;  roommatrix[10][3] = 0.1;  roommatrix[10][4] = 30;  //2205  roommatrix[11][0] = 2205;  roommatrix[11][1] = 13.34;  roommatrix[11][2] = 21;  roommatrix[11][3] = 37.3;  roommatrix[11][4] = 37.3;  //2207  roommatrix[12][0] = 2207;  roommatrix[12][1] = 11.09;  roommatrix[12][2] = 13.34;  roommatrix[12][3] = 37.3;  roommatrix[12][4] = 37.3;  //2208  roommatrix[13][0] = 2208;  roommatrix[13][1] = 8.77;  roommatrix[13][2] = 11.09;  roommatrix[13][3] = 37.3;  roommatrix[13][4] = 37.3;  //2209  roommatrix[14][0] = 2209;  roommatrix[14][1] = 1.94;  roommatrix[14][2] = 8.77;  roommatrix[14][3] = 37.3;  roommatrix[14][4] = 37.3;  //2211  roommatrix[15][0] = 2211;  roommatrix[15][1] = 1.64;  roommatrix[15][2] = 1.64;  roommatrix[15][3] = 46.1;  roommatrix[15][4] = 48.94;  //2212  roommatrix[16][0] = 2212;  roommatrix[16][1] = 1.64;  roommatrix[16][2] = 1.64;  roommatrix[16][3] = 48.94;  roommatrix[16][4] = 51.8;  //2213  roommatrix[17][0] = 2213;  roommatrix[17][1] = 1.64;  roommatrix[17][2] = 1.64;  roommatrix[17][3] = 51.8;  roommatrix[17][4] = 54.66;  //2214  roommatrix[18][0] = 2214;  roommatrix[18][1] = 1.64;  roommatrix[18][2] = 1.64;  roommatrix[18][3] = 54.66;  roommatrix[18][4] = 57.31;  //2215  roommatrix[19][0] = 2215;  roommatrix[19][1] = 1.64;  roommatrix[19][2] = 1.64;  roommatrix[19][3] = 57.31;  roommatrix[19][4] = 59.34;  //2216 7.71 14.28 59.34 59.34  roommatrix[20][0] = 2216;  roommatrix[20][1] = 7.71;  roommatrix[20][2] = 14.28;  roommatrix[20][3] = 61;  roommatrix[20][4] = 61;  //2219 14.28 21.15 59.34 59.34  roommatrix[21][0] = 2219;  roommatrix[21][1] = 14.28;  roommatrix[21][2] = 21.15;  roommatrix[21][3] = 61;  roommatrix[21][4] = 61;  //2220 21.15 26.06 59.34 59.34  roommatrix[22][0] = 2220;  roommatrix[22][1] = 21.15;  roommatrix[22][2] = 26.06;  roommatrix[22][3] = 61;  roommatrix[22][4] = 61;  //2105B 26.85 26.85 73.14 81.94  roommatrix[23][0] = "2105B";  roommatrix[23][1] = 26.85;  roommatrix[23][2] = 26.85;  roommatrix[23][3] = 73.14;  roommatrix[23][4] = 81.94;  //2105C 26.85 26.85 65.84 73.14  roommatrix[24][0] = "2105C";  roommatrix[24][1] = 26.85;  roommatrix[24][2] = 26.85;  roommatrix[24][3] = 65.84;  roommatrix[24][4] = 73.14;  //2105D 12.31 14.61 62.59 65.84  roommatrix[25][0] = "2105D";  roommatrix[25][1] = 12.31;  roommatrix[25][2] = 14.61;  roommatrix[25][3] = 64;  roommatrix[25][4] = 64;  //2105A 11.86 14.37 62.59 85.46  roommatrix[26][0] = "2105A";  roommatrix[26][1] = 11.86;  roommatrix[26][2] = 14.37;  roommatrix[26][3] = 62.59;  roommatrix[26][4] = 85.46;  //Elevator(2nd floor) 24.9 25.9 33.65 33.65  roommatrix[27][0] = "Elevator(2nd floor)";  roommatrix[27][1] = 24.9;  roommatrix[27][2] = 25.9;  roommatrix[27][3] = 35;  roommatrix[27][4] = 35;  //Stair 5 7.06 8.28 33.65 33.65  roommatrix[28][0] = "Stair 5";  roommatrix[28][1] = 7.06;  roommatrix[28][2] = 8.28;  roommatrix[28][3] = 35;  roommatrix[28][4] = 35;  //Stair 6 5.36 6.55 62.59 62.59  roommatrix[29][0] = "Stair 6";  roommatrix[29][1] = 5.36;  roommatrix[29][2] = 6.55;  roommatrix[29][3] = 64;  roommatrix[29][4] = 64;  //Stair 7 0.1 0.1 88.01 89.1  roommatrix[30][0] = "Stair 7";  roommatrix[30][1] = 0.1;  roommatrix[30][2] = 0.1;  roommatrix[30][3] = 88.01;  roommatrix[30][4] = 89.1;  //Washroom W206 21.71 24.9 33.65 33.65  roommatrix[31][0] = "Washroom W206";  roommatrix[31][1] = 21.71;  roommatrix[31][2] = 24.9;  roommatrix[31][3] = 35;  roommatrix[31][4] = 35;  //Washroom W207 9.23 14.35 33.65 33.65  roommatrix[32][0] = "Washroom W207";  roommatrix[32][1] = 9.23;  roommatrix[32][2] = 14.35;  roommatrix[32][3] = 35;  roommatrix[32][4] = 35;  //SECTION 1 0.1 5 0.1 2.60  roommatrix[33][0] = "2203 SECTION 1";  roommatrix[33][1] = 0.1;  roommatrix[33][2] = 5;  roommatrix[33][3] = 0.1;  roommatrix[33][4] = 2.60;  //SECTION 2 5 10 0.1 2.60  roommatrix[34][0] = "2203 SECTION 2";  roommatrix[34][1] = 5;  roommatrix[34][2] = 10;  roommatrix[34][3] = 0.1;  roommatrix[34][4] = 2.60;  //SECTION 3 0.1 5 2.60 8  roommatrix[35][0] = "2203 SECTION 3";  roommatrix[35][1] = 0.1;  roommatrix[35][2] = 5;  roommatrix[35][3] = 2.6;  roommatrix[35][4] = 8;  //SECTION 4 5 10 2.60 8  roommatrix[36][0] = "2203 SECTION 4";  roommatrix[36][1] = 5;  roommatrix[36][2] = 10;  roommatrix[36][3] = 2.6;  roommatrix[36][4] = 8;  //SECTION 5 0.1 5 8 12.2  roommatrix[37][0] = "2203 SECTION 5";  roommatrix[37][1] = .1;  roommatrix[37][2] = 5;  roommatrix[37][3] = 8;  roommatrix[37][4] = 12.2;  //SECTION 6 5 10 8 12.2  roommatrix[38][0] = "2203 SECTION 6";  roommatrix[38][1] = 5;  roommatrix[38][2] = 10;  roommatrix[38][3] = 8;  roommatrix[38][4] = 12.2;  //SECTION 7 0.1 5 12.2 16.5  roommatrix[39][0] = "2203 SECTION 7";  roommatrix[39][1] = 0.1;  roommatrix[39][2] = 5;  roommatrix[39][3] = 12.2;  roommatrix[39][4] = 16.5;  //SECTION 8 5 10 12.2 16.5  roommatrix[40][0] = "2203 SECTION 8";  roommatrix[40][1] = 5;  roommatrix[40][2] = 10;  roommatrix[40][3] = 12.2;  roommatrix[40][4] = 16.5;  //SECTION 9 0.1 5 16.5 20.7  roommatrix[41][0] = "2203 SECTION 9";  roommatrix[41][1] = 0.1;  roommatrix[41][2] = 5;  roommatrix[41][3] = 16.5;  roommatrix[41][4] = 20.7;  //SECTION 10 5 10 16.5 20.7  roommatrix[42][0] = "2203 SECTION 10";  roommatrix[42][1] = 5;  roommatrix[42][2] = 10;  roommatrix[42][3] = 16.5;  roommatrix[42][4] = 20.7;  //SECTION 11 0.1 5 20.7 25  roommatrix[43][0] = "2203 SECTION 11";  roommatrix[43][1] = 0.1;  roommatrix[43][2] = 5;  roommatrix[43][3] = 20.7;  roommatrix[43][4] = 25;  //SECTION 12 5 10 20.7 25  roommatrix[44][0] = "2203 SECTION 12";  roommatrix[44][1] = 5;  roommatrix[44][2] = 10;  roommatrix[44][3] = 20.7;  roommatrix[44][4] = 25;  //SECTION 13 0.1 5 25 31  roommatrix[45][0] = "2203 SECTION 13";  roommatrix[45][1] = 0.1;  roommatrix[45][2] = 5;  roommatrix[45][3] = 31;  roommatrix[45][4] = 31;  //2203 .1 3 31 31  roommatrix[46][0] = "2203";  roommatrix[46][1] = .1;  roommatrix[46][2] = 3;  roommatrix[46][3] = 31;  roommatrix[46][4] = 31;  //2210 0.1 4 34 43  roommatrix[47][0] = "2210";  roommatrix[47][1] = 0.1;  roommatrix[47][2] = 4;  roommatrix[47][3] = 34;  roommatrix[47][4] = 43;  // The code for starting the scan of the phone to scan the ble devices  app.startScan = function (callbackFun) {  evothings.ble.stopScan();  evothings.ble.startScan(  function (device) {  if (device.rssi <= 0) {  callbackFun(device, null);  }  },  function (errorCode) {  callbackFun(null, errorCode);  }  );  };  // This controls the ui on pressing the start scan button  app.ui.onStartScanButton = function () {  // Here we again initialize our variables with their initial values  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  sampleMatrix[i][j] = 0;  }  }  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  mwrss[i][j] = 0;  }  }  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  weighttable[i][j] = 0;  }  }  init = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3];  sumofbeaconsreadingsincol = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  arry = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  arrx = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  xx = 0;  yy = 0;  // Clearing the 1st map on the starting of the calculations  m.clear();  // Clearing the 2nd map on the starting of the calculations  m2.clear();  m3.clear();  m4.clear();  min = 1000000;  scorearr = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  scorearr2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  room = -1;  score = 0;  count = 0;  sdx = 0;  sdy = 0;  maxsdx = 0;  minsdx = 0;  maxsdy = 0;  minsdy = 0;  newavgx = 0;  newavgy = 0;  //xmovavg = [0, 0, 0, 0];  //ymovavg = [0, 0, 0, 0];  //indexmovavg = 0;  if (indexmovavg > 3)  indexmovavg = 0;  if (indexmovavgsecond > 3)  indexmovavgsecond = 0;  // Initially there should be no writing on the screen  var displayvanish = document.getElementById("showfinalresult");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showmap");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showroom");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showxy");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showweighttable");  displayvanish.style.display = "none";  displayvanish = document.getElementById("sumofcolshow");  displayvanish.style.display = "none";  displayvanish = document.getElementById("matrixdiv");  displayvanish.style.display = "none";  displayvanish = document.getElementById("mwrssshow");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showmovavg");  displayvanish.style.display = "none";  app.startScan(app.ui.deviceFound);  $('#scan-status').html('<b>Helping you...Calibrating results</b>');  app.ui.updateTimer = setInterval(app.ui.displayDeviceList, 100);  setTimeout(function () { $('#scan-status').html('<b>Results ready..Click WHERE AM I??? to see!!</b>'); }, 3000);  };  // For clearing the matrix after completion of training  app.clearMatrix = function (matrix) {  // This empties the init array  init.length = 0;  matrix.forEach(function (element, index) {  // This empties the contents of the RSS and time for each of the beacons from the sampleMatrix matrix  sampleMatrix[index].length = 0;  // Pushes 3 at the end of the init array  init.push(3);  });  };  // This controls the ui on pressing the reset button.  app.ui.onResetScanButton = function () {  // Here we again initialize all the variables to their initial values as we are resetting for our fresh calculations  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  sampleMatrix[i][j] = 0;  }  }  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  mwrss[i][j] = 0;  }  }  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  weighttable[i][j] = 0;  }  }  init = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3];  sumofbeaconsreadingsincol = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  arry = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  arrx = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  xx = 0;  yy = 0;  // Clearing the 1st map on the resetting of the calculations  m.clear();  // Clearing the 2nd map on the resetting of the calculations  m2.clear();    m3.clear();  m4.clear();  min = 1000000;  scorearr = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  scorearr2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  room = -1;  score = 0;  count = 0;  sdx = 0;  sdy = 0;  maxsdx = 0;  minsdx = 0;  maxsdy = 0;  minsdy = 0;  newavgx = 0;  newavgy = 0;    //xmovavg = [0, 0, 0, 0];  //ymovavg = [0, 0, 0, 0];  //indexmovavg = 0;  if (indexmovavg > 3)  indexmovavg = 0;  if (indexmovavgsecond > 3)  indexmovavgsecond = 0;  // All the writings on the screen should vanish on pressing the RESET button  var displayvanish = document.getElementById("showfinalresult");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showmap");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showroom");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showxy");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showweighttable");  displayvanish.style.display = "none";  displayvanish = document.getElementById("sumofcolshow");  displayvanish.style.display = "none";  displayvanish = document.getElementById("matrixdiv");  displayvanish.style.display = "none";  displayvanish = document.getElementById("mwrssshow");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showmovavg");  displayvanish.style.display = "none";  evothings.ble.stopScan();  app.devices = {};  app.clearMatrix(beaconsMac);  $('#scan-status').html('Click <b>CLICK TO ACTIVATE</b> button to help you!!');  app.ui.displayDeviceList();  clearInterval(app.ui.updateTimer);  };  // This function will reset everything to initial state after calculating moving average  function refreshmovavg() {  // Here we again initialize all the variables to their initial values as we are resetting for our fresh calculations  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  sampleMatrix[i][j] = 0;  }  }  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  mwrss[i][j] = 0;  }  }  for (i = 0; i < 15; i++) {  for (j = 0; j < 23; j++) {  weighttable[i][j] = 0;  }  }  init = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3];  sumofbeaconsreadingsincol = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  arry = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  arrx = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  xx = 0;  yy = 0;  // Clearing the first map on the resetting of the calculations  m.clear();  // Clearing the 2nd map on the resetting of the calculations  m2.clear();  m3.clear();  m4.clear();  min = 1000000;  scorearr = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  scorearr2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];  room = -1;  score = 0;  count = 0;  xmovavg = [0, 0, 0, 0];  ymovavg = [0, 0, 0, 0];  indexmovavg = 0;  xmovavgsecond = [0, 0, 0, 0];  ymovavgsecond = [0, 0, 0, 0];  indexmovavgsecond = 0;  sdx = 0;  sdy = 0;  maxsdx = 0;  minsdx = 0;  maxsdy = 0;  minsdy = 0;  newavgx = 0;  newavgy = 0;  // All the writings on the screen should vanish on pressing the RESET button  var displayvanish = document.getElementById("showfinalresult");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showmap");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showroom");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showxy");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showweighttable");  displayvanish.style.display = "none";  displayvanish = document.getElementById("sumofcolshow");  displayvanish.style.display = "none";  displayvanish = document.getElementById("matrixdiv");  displayvanish.style.display = "none";  displayvanish = document.getElementById("mwrssshow");  displayvanish.style.display = "none";  displayvanish = document.getElementById("showmovavg");  displayvanish.style.display = "none";  evothings.ble.stopScan();  app.devices = {};  app.clearMatrix(beaconsMac);  $('#scan-status').html('Click <b>CLICK TO ACTIVATE</b> button to help you!!');  app.ui.displayDeviceList();  clearInterval(app.ui.updateTimer);  }  // This controls the ui on pressing the stop scan button  app.ui.onStopScanButton = function () {  evothings.ble.stopScan();  app.devices = {};  $('#scan-status').html('Calculation Stopped');  app.ui.displayDeviceList();  clearInterval(app.ui.updateTimer);  };  // The following part actually registers our beacons to whom we are listening  app.ui.deviceFound = function (device, errorCode) {  var wantedBeacon = false;  beaconsMac.forEach(function (element, index) {  if (element == device.address) {  wantedBeacon = true;  }  });  if (wantedBeacon == true) {  if (device) {  device.timeStamp = Date.now();  app.devices[device.address] = device;  }  else if (errorCode) {  $('#scan-status').html('Scan Error: ' + errorCode);  }  };  };  // This part displays our devices in the form of the table  app.ui.displayDeviceList = function () {  $('#found-devices').empty();  var timeNow = Date.now();  $.each(app.devices, function (key, device) {  if (device.timeStamp + 5000 > timeNow) {  var rssiWidth = 100; // Used when RSSI is zero or greater.  if (device.rssi < -100) { rssiWidth = 0; }  else if (device.rssi < 0) { rssiWidth = 100 + device.rssi; }  var coordinates = device.name.split("\_");  var x = parseFloat(coordinates[0]) / 10;  var y = parseFloat(coordinates[1]) / 10;  var h = new Date();  var element = $(  '<li>'  + 'Name: <strong>' + device.name + '</strong><br />'  + 'X: ' + x + '<br/>'  + 'Y: ' + y + '<br/>'  + 'MAC: ' + (evothings.os.isIOS() ? '' : device.address + '<br />')  + 'Hour of register: ' + h.getHours() + ':' + h.getMinutes() + ':' + h.getSeconds() + ':' + h.getMilliseconds() + '<br />'  + 'RSS: ' + device.rssi + '<br />'  + '<div style="background:rgb(213, 149, 99);height:20px;width:'  + rssiWidth + '%;"></div>'  + '</li>'  );  //this actually appends to the list which shows the list of available ble devices  $('#found-devices').append(element);  beaconsMac.forEach(function (element, index) {  if (element == device.address) {  if (init[index] <= 22) {  sampleMatrix[index][0] = device.address;  sampleMatrix[index][1] = x;  sampleMatrix[index][2] = y;  sampleMatrix[index][init[index]] = device.rssi;  init[index] = init[index] + 1;  }  };  });  //};  }  });  };  // This function shows the matrix sampleMatirx as a table of lists  function showRSStable() {  text = "Showing RSS values in dBm" + "<br>";  text = text + "<ol> ";  for (var i = 0; i < 15; i++) {  text = text + "<li> "  for (var j = 0; j < 23; j++) {  text = text + sampleMatrix[i][j] + " ; ";  }  text = text + "</li> ";  }  text = text + "</ol> ";  document.getElementById("matrixshow").innerHTML = text;  var x = document.getElementById("matrixdiv");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  // This function shows/hides the dynamic list of the devices on clicking the SHOW/HIDE DL button  function showhidedlfn() {  var x = document.getElementById("showhidedl");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  // This function will shift the coordinates of the beacons by avgbx and avgby for normalization  function shiftcoordinates() {  for (var i = 0; i < 15; i++) {  if (sampleMatrix[i][0] != 0) {  sampleMatrix[i][1] = sampleMatrix[i][1] - avgbx;  sampleMatrix[i][2] = sampleMatrix[i][2] - avgby;  }  }  }  // This function calculates the RSS values in miliwatts from the dBm values which we got from the beacons  function calcRSSmw() {  for (var i = 0; i < 15; i++) {  for (var j = 0; j < 23; j++) {  if (j == 0) {  mwrss[i][j] = sampleMatrix[i][j];  } else if (j == 1) {  mwrss[i][j] = sampleMatrix[i][j];  } else if (j == 2) {  mwrss[i][j] = sampleMatrix[i][j];  } else {  if (sampleMatrix[i][j] != 0)  mwrss[i][j] = Math.pow(10, (sampleMatrix[i][j] / 10));  else  mwrss[i][j] = 0;  }  }  }  }  // This function shows the RSS values in miliwatts to the user  function showRSSmw() {  text = "Showing RSS values in milli watts" + "<br>";  text = text + "<ol> ";  for (var i = 0; i < 15; i++) {  text = text + "<li> ";  for (var j = 0; j < 23; j++) {  text = text + mwrss[i][j] + " ; ";  }  text = text + "</li> ";  }  text = text + "</ol> ";  document.getElementById("mwrssshow").innerHTML = text;  var x = document.getElementById("mwrssshow");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  function calcsumcol() {  for (var j = 3; j <= 22; j++) {  for (var i = 0; i <= 14; i++) {  sumofbeaconsreadingsincol[j - 3] = sumofbeaconsreadingsincol[j - 3] + mwrss[i][j];  }  }  }  function showsumcol() {  var textcol = "Showing sum of RSS values in dBm per column" + "<br>";  textcol = textcol + "<ol> ";  for (var i = 0; i < 20; i++) {  textcol = textcol + "<li> " + sumofbeaconsreadingsincol[i] + "</li> ";  }  textcol = textcol + "</ol> ";  document.getElementById("sumofcolshow").innerHTML = textcol;  var x = document.getElementById("sumofcolshow");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  function calcwttable() {  for (j = 0; j <= 22; j++) {  for (i = 0; i <= 14; i++) {  if (j == 0) {  weighttable[i][j] = mwrss[i][j];  } else if (j == 1) {  weighttable[i][j] = mwrss[i][j];  } else if (j == 2) {  weighttable[i][j] = mwrss[i][j];  } else {  if (sumofbeaconsreadingsincol[j - 3] != 0) {  weighttable[i][j] = mwrss[i][j] / sumofbeaconsreadingsincol[j - 3];  }  }  }  }  //document.getElementById("showweighttable").innerHTML = "jjj";  }  function showweighttable() {  text = "Showing weights of the various beacons" + "<br>";  text = text + "<ol> ";  for (var i = 0; i < 15; i++) {  text = text + "<li> ";  for (var j = 0; j < 23; j++) {  text = text + weighttable[i][j] + " ; ";  }  text = text + "</li> ";  }  text = text + "</ol> ";  document.getElementById("showweighttable").innerHTML = text;  var x = document.getElementById("showweighttable");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  function calcxy() {  for (var j = 3; j <= 22; j++) {  for (var i = 0; i <= 14; i++) {  arrx[j - 3] = arrx[j - 3] + weighttable[i][j] \* weighttable[i][1];  arry[j - 3] = arry[j - 3] + weighttable[i][j] \* weighttable[i][2];;  }  }  }  function showxy() {  xx = 0;  yy = 0;    for (var i = 0; i < 20; i++) {  xx = xx + arrx[i];  yy = yy + arry[i];  }  xx = xx / 20;  yy = yy / 20;  text = "Showing the possible X and Y (20 values)" + "<br>";  text = text + "<ol> ";  for (var i = 0; i < 20; i++) {  text = text + "<li> X : " + arrx[i] + " Y : " + arry[i] + " </li> ";  }  text = text + "</ol> ";  text = text + " Average X and Y is : " + "<br>";  text = text + " X : " + xx + " && Y : " + yy + "<br>";  document.getElementById("showxy").innerHTML = text;  var x = document.getElementById("showxy");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  // This finds the average of the x and y values  function findxandyavg() {  xx = 0;  yy = 0;  sdx = 0;  sdy = 0;  for (var i = 0; i < 20; i++) {  xx = xx + arrx[i];  yy = yy + arry[i];  }  xx = xx / 20;  yy = yy / 20;  /\*  for (var i = 0; i < 20; i++) {  sdx = sdx + (arrx[i] - xx) \* (arrx[i] - xx);  sdy = sdy + (arry[i] - yy) \* (arry[i] - yy);  }  sdx = sdx / 20;  sdy = sdy / 20;  sdx = Math.sqrt(sdx);  sdy = Math.sqrt(sdy);  maxsdx = xx + sdx;  minsdx = xx - sdx;  maxsdy = yy + sdy;  minsdy = yy - sdy;  newavgx = 0;  newavgy = 0;  // This variable will count the no of good coordinates we are getting from the pool of 20 coordinates  var countavg = 0;  for (var i = 0; i < 20; i++) {  if (arrx[i] >= minsdx && arrx[i] <= maxsdx && arry[i] >= minsdy && arry[i] <= maxsdy) {  newavgx = newavgx + arrx[i];  newavgy = newavgy + arry[i];  countavg++;  }  }  newavgx = newavgx / countavg;  newavgy = newavgy / countavg;  \*/  }  // This function calculates the scores of all the rooms and assigns them to the map and the array  function calcroom() {  score = 0;  // This map will store the scores for the rooms  //var m = new Map();  m.clear();  m2.clear();  // Calculations for the first algo  for (var i = 0; i <= 47; i++) {  score = 0;  for (var j = 1; j <= 4; j++) {  if (j == 1)  score = score + Math.abs((xx + avgbx) - roommatrix[i][j]);  else if (j == 2)  score = score + Math.abs((xx + avgbx) - roommatrix[i][j]);  else if (j == 3)  score = score + Math.abs((yy + avgby) - roommatrix[i][j]);  else if (j == 4)  score = score + Math.abs((yy + avgby) - roommatrix[i][j]);  }  m.set(roommatrix[i][0], score);  scorearr[i] = score;  }  // These variables are required for calculating the scores of the rooms for the 2nd algorithm  var xavg = 0, yavg = 0, calc = 0, result = 0;      // Calculations for the second algo  for (var i = 0; i <= 47; i++) {  score = 0; xavg = 0; yavg = 0; calc = 0; result = 0;  for (var j = 1; j <= 4; j++) {  if (j == 1)  xavg = xavg + roommatrix[i][j];  else if (j == 2)  xavg = xavg + roommatrix[i][j];  else if (j == 3)  yavg = yavg + roommatrix[i][j];  else if (j == 4)  yavg = yavg + roommatrix[i][j];  }  xavg = xavg / 2;  yavg = yavg / 2;  calc = ((xx + avgbx) - xavg) \* ((xx + avgbx) - xavg) + ((yy + avgby) - yavg) \* ((yy + avgby) - yavg);  result = Math.sqrt(calc);  m2.set(roommatrix[i][0], result);  scorearr2[i] = result;  }  }  // This function displays the scorearr and the most probable room  function showroom() {  min = scorearr[0];  room = [0][0];  for (i = 1; i <= 47; i++) {  if (scorearr[i] < min) {  min = scorearr[i];  room = roommatrix[i][0];  }  }  text = "The scores of the rooms are " + "<br>";  text = text + "<ol> ";  for (i = 0; i <= 47; i++) {  text = text + "<li> Room : " + roommatrix[i][0] + " && score : " + scorearr[i] + "</li>";  }  text = text + "</ol>";  text = text + " The most appropriate room is " + room + "<br>";  document.getElementById("showroom").innerHTML = text;  var x = document.getElementById("showroom");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  // This function sorts the map in the ascending order of the values for both the algorithm  function calcmap() {  m[Symbol.iterator] = function\* () {  yield\* [...this.entries()].sort((a, b) => a[1] - b[1]);  }  m2[Symbol.iterator] = function\* () {  yield\* [...this.entries()].sort((a, b) => a[1] - b[1]);  }  }  // This shows the entire map after it is sorted if it is shown after the sorting is done or it shows the normal way if the sorting is not done  function showmap() {  text = "The map entries are " + "<br>";  text = text + "<ol>";  for (let [k, v] of m) {  text = text + "<li> Room : " + k + " && Score : " + v + "</li>";  }  text = text + "</ol>";  document.getElementById("showmap").innerHTML = text;  var x = document.getElementById("showmap");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  // This shows the final result to our user  function showfinalresult() {  count = 0;  /\*  text = "<b>You are near rooms from first algorithm : </b>" + "<br>";  text = text + "<ol>";  for (let [k, v] of m) {  if (count <= 3) {  text = text + "<li><b><u> Room</b></u> : " + k + " && score : " + v + " </li>";  count++;  }  }  text = text + "</ol>";  count = 0;  text = text + "<br>";  text = text + "<b>You are near rooms from second algorithm : </b>" + "<br>";  text = text + "<ol>";  for (let [k, v] of m2) {  if (count <= 3) {  text = text + "<li><b><u> Room</b></u> : " + k + " && score : " + v + " </li>";  count++;  }  }  text = text + "</ol>";  count = 0;  text = text + "<br>";  \*/  // This is for the first algorithm  text = "Your coordinates from first algo are <b> X : </b> " + (xx + avgbx) + " && <b> Y : </b> " + (yy + avgby) + "<br>";  // This is for the second algorithm  //text = text + "Your coordinates from second algo are <b> X : </b> " + (newavgx + avgbx) + " && <b> Y : </b> " + (newavgy + avgby) + "<br>";  if (indexmovavg <= 3) {  xmovavg[indexmovavg] = xx + avgbx;  ymovavg[indexmovavg] = yy + avgby;  indexmovavg++;  }  if (indexmovavgsecond <= 3) {  xmovavgsecond[indexmovavgsecond] = newavgx + avgbx;  ymovavgsecond[indexmovavgsecond] = newavgy + avgby;  indexmovavgsecond++;  }  document.getElementById("showfinalresult").innerHTML = text;  var x = document.getElementById("showfinalresult");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  }  // The final button which will do all our calculations  function doallcalc() {  shiftcoordinates();  calcRSSmw();  calcsumcol();  calcwttable();  calcxy();  findxandyavg();  calcroom();  calcmap();  showfinalresult();  }  // This function will calculate our moving average  function calcmovavg() {  var curr\_avgx, prev\_avgx, curr\_avgy, prev\_avgy, threshold = 0.1;  /\*  curr\_avgx = (xmovavg[0] + xmovavg[1]) / 2;  curr\_avgy = (ymovavg[0] + ymovavg[1]) / 2;  prev\_avgx = curr\_avgx;  prev\_avgy = curr\_avgy;  curr\_avgx = (prev\_avgx + xmovavg[2]) / 2;  curr\_avgy = (prev\_avgy + ymovavg[2]) / 2;  prev\_avgx = curr\_avgx;  prev\_avgy = curr\_avgy;  curr\_avgx = (prev\_avgx + xmovavg[3]) / 2;  curr\_avgy = (prev\_avgy + ymovavg[3]) / 2;  prev\_avgx = curr\_avgx;  prev\_avgy = curr\_avgy;  \*/  // These arrays will store the moving average results for the x and y coordinates  var arrxmovavg = [];  var arrymovavg = [];  for (var i = 0; i <= 200; i++) {  arrxmovavg[i] = 0;  arrymovavg[i] = 0;  }  // This calculates the starting of the moving average for forst algorithm  arrxmovavg[0] = (xmovavg[0] + xmovavg[1]) / 2;  arrymovavg[0] = (ymovavg[0] + ymovavg[1]) / 2;  arrxmovavg[1] = (xmovavg[2] + arrxmovavg[0]) / 2;  arrymovavg[1] = (ymovavg[2] + arrymovavg[0]) / 2;  arrxmovavg[2] = (xmovavg[3] + arrxmovavg[1]) / 2;  arrymovavg[2] = (ymovavg[3] + arrymovavg[1]) / 2;  // This loop will calculate the moving average for the first algo  for (var i = 3; i <= 200; i++) {  arrxmovavg[i] = (arrxmovavg[i - 1] + arrxmovavg[i - 3]) / 2;  arrymovavg[i] = (arrymovavg[i - 1] + arrymovavg[i - 3]) / 2;  }  // This is for the first algorithm  text = "Your moving avg coordinates from 1st algo are <b> X : </b> " + arrxmovavg[200] + " && <b> Y : </b> " + arrymovavg[200] + "<br>";  /\*  m3.clear();  m4.clear();  var score = 0;  // Calculations for the first algo  for (var i = 0; i <= 47; i++) {  score = 0;  for (var j = 1; j <= 4; j++) {  if (j == 1)  score = score + Math.abs(arrxmovavg[200] - roommatrix[i][j]);  else if (j == 2)  score = score + Math.abs(arrxmovavg[200] - roommatrix[i][j]);  else if (j == 3)  score = score + Math.abs(arrymovavg[200] - roommatrix[i][j]);  else if (j == 4)  score = score + Math.abs(arrymovavg[200] - roommatrix[i][j]);  }  m3.set(roommatrix[i][0], score);  //scorearr[i] = score;  }  // These variables are required for calculating the scores of the rooms for the 2nd algorithm  var xavg = 0, yavg = 0, calc = 0, result = 0;  // Calculations for the second algo  for (var i = 0; i <= 47; i++) {  score = 0; xavg = 0; yavg = 0; calc = 0; result = 0;  for (var j = 1; j <= 4; j++) {  if (j == 1)  xavg = xavg + roommatrix[i][j];  else if (j == 2)  xavg = xavg + roommatrix[i][j];  else if (j == 3)  yavg = yavg + roommatrix[i][j];  else if (j == 4)  yavg = yavg + roommatrix[i][j];  }  xavg = xavg / 2;  yavg = yavg / 2;  calc = (arrxmovavg[200] - xavg) \* (arrxmovavg[200] - xavg) + (arrymovavg[200] - yavg) \* (arrymovavg[200] - yavg);  result = Math.sqrt(calc);  m4.set(roommatrix[i][0], result);  //scorearr2[i] = result;  }  \*/  /\*  m3[Symbol.iterator] = function\* () {  yield\* [...this.entries()].sort((a, b) => a[1] - b[1]);  }  m4[Symbol.iterator] = function\* () {  yield\* [...this.entries()].sort((a, b) => a[1] - b[1]);  }  count = 0;  text = text + "<b>You are near rooms from first algorithm by moving avg : </b>" + "<br>";  text = text + "<ol>";  for (let [k, v] of m3) {  if (count <= 3) {  text = text + "<li><b><u> Room</b></u> : " + k + " && score : " + v + " </li>";  count++;  }  }  text = text + "</ol>";  count = 0;  text = text + "<br>";  text = text + "<b>You are near rooms from second algorithm by mov avg : </b>" + "<br>";  text = text + "<ol>";  for (let [k, v] of m4) {  if (count <= 3) {  text = text + "<li><b><u> Room</b></u> : " + k + " && score : " + v + " </li>";  count++;  }  }  text = text + "</ol>";  count = 0;  text = text + "<br>";  \*/  /\*  for (var i = 0; i <= 200; i++) {  arrxmovavg[i] = 0;  arrymovavg[i] = 0;  }  arrxmovavg[0] = (xmovavgsecond[0] + xmovavgsecond[1]) / 2;  arrymovavg[0] = (ymovavgsecond[0] + ymovavgsecond[1]) / 2;  arrxmovavg[1] = (xmovavgsecond[2] + arrxmovavg[0]) / 2;  arrymovavg[1] = (ymovavgsecond[2] + arrymovavg[0]) / 2;  arrxmovavg[2] = (xmovavgsecond[3] + arrxmovavg[1]) / 2;  arrymovavg[2] = (ymovavgsecond[3] + arrymovavg[1]) / 2;  // This loop will calculate the moving average for the second algo  for (var i = 3; i <= 200; i++) {  arrxmovavg[i] = (arrxmovavg[i - 1] + arrxmovavg[i - 3]) / 2;  arrymovavg[i] = (arrymovavg[i - 1] + arrymovavg[i - 3]) / 2;  }  // This is for the second algorithm  text = text + "Your moving avg coordinates from 2nd algo are <b> X : </b> " + arrxmovavg[200] + " && <b> Y : </b> " + arrymovavg[200] + "<br>";  \*/    document.getElementById("showmovavg").innerHTML = text;  var x = document.getElementById("showmovavg");  if (x.style.display === "none") {  x.style.display = "block";  } else {  x.style.display = "none";  }  } |

* Description of all functions and variables in the **app.js** file:

In the function **app.ui.onStartScanButton()**

1. The function **app.startScan()** is called with the function **app.ui.deviceFound()** as parameter

app.startScan(app.ui.deviceFound());

1. With this line we set how often the list is updated (100 miliseconds):

app.ui.updateTimer = setInterval(app.ui.displayDeviceList, 100);

In the function **app.ui.onStopScanButton()**:

1. Stop the Bluetooth scan:

evothings.ble.stopScan();

1. Clear the list of devices:

app.devices = {};

1. Call the function app.ui.displayDeviceList() and clearInterval():

app.ui.displayDeviceList();

clearInterval(app.ui.updateTimer);

In the function **app.ui.displayDeviceList()**:

1. Clear device list and set timeNow to Date.now():

$('#found-devices').empty();

var timeNow = Date.now();

1. Update the list only with the devices available in the last 5 seconds:

$.each(app.devices, function (key, device) {

if (device.timeStamp + 5000 > timeNow) {

1. Map the RSSI value to a width in percent for the indicator:

var rssiWidth = 100; // Used when RSSI is zero or greater.

if (device.rssi < -100) { rssiWidth = 0; }

else if (device.rssi < 0) { rssiWidth = 100 + device.rssi; }

1. The name of the beacons are their coordinates in the format x\_y with one decimal, separate them to show them properly:

var coordinates = device.name.split("\_");

var x = parseFloat(coordinates[0]) / 10;

var y = parseFloat(coordinates[1]) / 10;

1. Create a new date object to record the time when the samples are taken:

var h = new Date();

1. Create tag for device data:

var element = $(

'<li>'

+ 'Name: <strong>' + device.name + '</strong><br />'

+ 'X: ' + x + '<br/>'

+ 'Y: ' + y + '<br/>'

+ 'MAC: ' + (evothings.os.isIOS() ? '' : device.address + '<br />')

+ 'Hour of register: ' + h.getHours() + ':' + h.getMinutes() + ':' + h.getSeconds() + ':' + h.getMilliseconds() + '<br />'

+ 'RSS: ' + device.rssi + '<br />'

+ '<div style="background:rgb(213, 149, 99);height:20px;width:'

+ rssiWidth + '%;"></div>'

+ '</li>'

);

1. Show the tag:

$('#found-devices').append(element);

In the function **evothings.ble.stopScan()**, format: callbackFun(deviceInfo, errorCode):

1. call the callback function when a device is found:

evothings.ble.stopScan();

evothings.ble.startScan(

function (device) {

1. Sometimes an RSSI of +127 is reported. We filter out these values here:

if (device.rssi <= 0) {

callbackFun(device, null);

}

},

function (errorCode) {

// Report error.

callbackFun(null, errorCode);

}

);

};

In the function **app.ui.deviceFound()**, called when a device is found:

app.ui.deviceFound = function (device, errorCode) {

1. Create a variable to tell when a wanted beacon is found:

var wantedBeacon = false;

1. Filters only the beacons that we need:

beaconsMac.forEach(function (element, index) {

if (element == device.address) {

wantedBeacon = true;

}

});

1. In the case a wanted device is found:

if (wantedBeacon == true) {

if (device) {

1. Set timestamp for device (this is used to remove inactive devices):

device.timeStamp = Date.now();

1. Insert the device into table of found devices:

app.devices[device.address] = device;

}

1. Handle the error if the device is not found:

else if (errorCode) {

$('#scan-status').html('Scan Error: ' + errorCode);

}

};

};

In the function **app.clearMatrix(matrix)** a matrix is received as parameter:

1. Clear everything based on the matrix received:

init.length = 0;

matrix.forEach(function (element, index) {

sampleMatrix[index].length = 0;

init.push(3);

});

};

List of variables created in **app.js**:

* Application object:

var app = {};

* Device list:

app.devices = {};

* UI methods:

app.ui = {};

* Timer that updates the device list and removes inactive devices in case no devices are found by scan:

app.ui.updateTimer = null;

* List of beacons that we use, here we have to add them manually. (In the case we want to add more beacons, we must add a column in init var and a row in sampleMatrix var):

var beaconsMac = ['D4:F5:13:FF:11:4C', '20:C3:8F:E0:83:5B', '7C:EC:79:E0:20:24'/\*'D4:F5:13:FE:81:6D'\*/, '20:C3:8F:E0:90:8C'];

* Matrix starts in 0,0 in the upper left corner first specify row and then column (MAC, X, Y, RSS1, RSS2, RSS3, RSS4 etc.):

var sampleMatrix = [[], [], [], []];

* Defines the start column for RSSI and time. The number of columns in init and beaconsMac must be the same:

var init = [3, 3, 3, 3];

**Apart from the definitions mentioned here all the functions and variables have also been clearly defined and initialized in the code and have been properly commented. So, I will suggest the reader to read through the code to get better knowhow of how things are being done.**

# General information regarding the project

**I will now put some general information about the project and how we went through it apart from the above documentation of the code so that it becomes easy for the future students to pursue this project and take it to greater heights to make it more marketable.**

There have been many algorithms to find indoor positioning but we use a lightweight and simple algorithm to find indoor positioning up to a reasonable error. This project focuses on implementing that algorithm on a cell phone for room level resolution. The purpose of this paper is to give a detailed report on how those algorithms can be implemented practically for detecting indoor locations. Although there have been many proposed methods to find out indoor location there is no authentic report on how they perform on a mobile device and how different algorithms compare when operated from cell phones.

For our research and experiment purpose, we have taken a portion of the 2nd floor of the Ed Lumley Centre for Engineering Innovation and we designated the various rooms as our objects which we shall try to track with our mobile phone application.

Our target is to build a complete application which can faithfully tell us the rooms we are near a point. The entire code for the project will be open-sourced for the benefit of the scientific community and for further development and research in this field. We started developing our own code on top of the code which [referencing the Evothings studio GitHub link for the ble- <https://github.com/evothings/evothings-examples/tree/master/examples/ble-scan> ] Evothings provided for the scanning of Bluetooth Low Energy devices. We are using iBeacons for the testing of our application.

First, we scan all BLE devices which are available at a certain point of our interest area. We shall only take into account those devices which were updated in the last 5 seconds. The data which we get from the BLE devices like their RSS values and their X and Y coordinates are stored in a matrix. For each beacon we sample 20 readings, each beacon having a reading at a particular time. We have also normalized the coordinates of each of the beacons. We then process that information in the sequential step as described in our algorithm.

The entire programming for the application has been done using HTML5, CSS3 and JavaScript. It is simple and performs computationally very less complex tasks that cell phones of all capabilities can run the application efficiently with very little battery usage. All our tests has been carried out using two phones. One Samsung M20(4 GB RAM) and (Blade V8 SE, 2GB RAM).

We tried to implement tracking in indoor locations by including room level resolution tracking in our application. We use two different algorithms for this and we test out the performance of the algorithms using our application for better insights. Computationally both of them are very simple and run in O(n) complexity.

We take the help of JavaScript maps for assigning the scores to the rooms. The Xmin, Xmax, Ymin, Ymax of the each of the rooms has been stored in a matrix in our script.  From our mapping, we see that we have 4 coordinates for each room: 2 X coordinates and 2 Y coordinates. We assign the score of each room by using our algorithms.

Our map has the keys as the room no and their corresponding values are the scores as calculated from above. Next, we sort the map in ascending order of the scores of the rooms. The room which will have the least score will most likely be our room we are nearest too. Since for the work of the project, we were concerned with room-level accuracy so we display to the user the first 4 rooms with the lowest scores as rooms they are nearest too. We do this because rooms are located side by side and at any time a user will be in close proximity to 3 to 4 rooms, so for better guidance of the user we display to him the best 4 possibilities where he can be.

The project showed that how we can implement basic algorithms for indoor positioning systems with accuracy good enough for room level resolution. We also implemented the moving average which vastly improved our results. The application ran efficiently on different smartphones and there was no requirement of any internet connection or any connection to the server for the running of our application. The code for the paper has been made open source for the betterment of the cast scientific community working in this field [<https://github.com/sarnava1/MITACS_Project_19>. The app could predict the rooms in our vicinity all the times the experiment ran, and we tested our application with two different algorithms. The error we got for the indoor positioning is well within the range for acute room level positioning. We are next planning to upgrade our current work for real time monitoring and tracking of people and making the precision more accurate so that our app could be practically used in museums or stores in malls.

**Improvements that can be made:**

We could make more improvements to the project by:

1. Eliminating human errors during the mapping phase of our area
2. Using better beacons with stronger range and reach
3. Using uniform beacons all throughout the interest area
4. Using a more denser sensor network for more and more accurate readings

**Our USP:**

1. First time we have implemented the indoor location detection in a mobile platform by developing the application
2. We have implemented room level resolution and correctly identified the various rooms in our vicinity. We have used 2 different algorithms for it, and we have compared the performance of both of them
3. We have used very cheap and non-uniform sensors to get accuracy based on RSS readings where the error is so less and that is a huge improvement over the previous works. We have over here have compared 2 algorithms and generated all their graphs and did the analytics. The readings and the accuracy would have been much better if we had used uniform and more costly sensor. But our product needs to be marketable and must be cost efficient. To model practical situations, we used cheap sensors and used non uniform sensors as uniform sensors may not be available all the time and thus for IPS we may need to use non uniform sensors. Even then the average error we got was very less and good enough for room level resolution. The moving average algorithm worked excellent and the results are a proof of this.
4. Our application is very light weight and our algorithms are computationally very less complex. It does not require any mapping, internet connection or any server to connect to. It is a completely offline solution and it does not require you to connect to the internet. For the product development phase, we did not even require a database. The app can be easily scaled and can be easily integrated with a database for indoor location detection of large buildings.
5. In short we have developed a product which is foolproof and is market ready. It is an efficient, cheap and deployable solution for Indoor Location Detection Purpose.
6. We have also implement moving average algorithm for the first time for IPS and it greatly improves our performance.

All the readings and experiments for the project was done for all types of positions, that is, positions which were very near to the beacons, positions which were close to the beacons and positions which were neutral as in like they were mid-way between two beacons or any random position. Thus, we got readings of all types and the errors we got thus were holistic in nature and not specific sort of. The readings were taken in each position in a random manner, that is, the phone was kept randomly in our hands while we took the readings. So, there was no bias or inclination to get good readings.

**Complexity of our algorithms:**

Our algorithm uses the simplest of calculations to give us results which are very efficient and better than many other complex calculations. Presently we were using 8 beacons for our area and later for comparing the usage of more no of beacons we increased them to 14 beacons. We were taking 20 RSS samples from each beacon at an instant of time. Our algorithm comprises of finding out the weights of the individual beacons at a instant of time and then using it to find the x and y coordinates of a place. The other thing we need to calculate are the weights of the rooms at an instant. We also need to sort them according to the scores to find out the best possible rooms near our vicinity. The test setup included 48 rooms or Points of Interests(POIs).

Let us consider that we are having n no of beacons for a large indoor space. We need to traverse through the matrices to calculate our weights of the beacons which will take approximately O(20n) which is equal to O(n) time for the calculations. 20 is being multiplied with n as there are 20 samples for each beacon so our matrices are two-dimensional matrices of dimension n\*20. If n is small, it is equivalent of taking around constant time for the computations. So for small values of n, which is the no of beacons we will always take O(1), that is, constant time for our calculations as n is very small. The other calculations include finding out the average of the 20 possible X or Y coordinate from which we get our final X and Y coordinate which will always take constant or O(1) time. The next computation we do is to assign the scores to the rooms. If there are m no of rooms then we take O(m) time to assign the scores. If m is small it our computations will again take constant or O(1) time for doing the calculations. Next we need to sort the rooms according to the scores they receive in ascending order. This will take O(mlogm) time if m is big. We can safely say that even this computation will take constant or O(1) time if m is small.

Thus we can conclude that our algorithm for small values of m or n will take nearly constant or O(1) time to finish the computations but for large values of n or m will take O(n) for calculating the weights of the beacons by traversing the matrices or it shall take O(mlogm) time to sort the rooms according to their scores. So our time complexity for our algorithm is O(n) if O(n) > O(mlog) OR O(mlogm) if O(mlogm) > O(n).

The part where we calculate the moving average to improve the accuracy of our algorithm takes constant time or O(1) time to finish the computations. The logic behind this is since for moving average we require only 4 values of X and Y coordinates for a position and then we calculate the moving average in a loop which iterates for 200 times. So effectively it shall take O(1) time to complete the computations which makes algorithm very simple and fast.

**NB.:** Theantennas in our cell phones are isotropic. So, they pick up signals from almost all directions equally in ideal conditions. But for practical reasons it will always not be so. The performance will be far from ideal of these mobile phone antennas. The cell phones may pick up stronger signals from the beacons they are closer to than the ones they are far from and, they can pick up stronger signals from the beacon they are facing than the beacon they are not facing. Also, we have no idea on how the user of our app will have the orientation of the phone in his hand. It is here that our moving average algorithm comes to the fore with huge improvements in result and to model for the fact that the user can be anywhere in an indoor environment with his cell phone in any orientation in his hand that while taking the reading we randomly moved our hands while standing in a particular area for taking the readings to model these practical situations. Thus, our readings are holistic in nature and has been taken with no bias to improve the results. The ideal the antennas of the phones tend to be the better it is for indoor positioning system. So, for that we have utilized 3 phones of different capabilities and processing power and cost to take our readings so that our results can give a general picture when our algorithm will be in use. Also, while recording our readings we kept in mind of what our coordinates were. Sometimes our hands were extended out and we considered all those extensions produced when we tried to take the measurements while calculating our error.

# Important links for the project:

There will be some problems initially with the setting up of the Visual Studio environment. I followed the solutions given in the below links to get through my problems when I faced them initially while setting up my environment.

* <https://stackoverflow.com/questions/25994163/could-not-resolve-all-dependencies-for-configuration-classpath>
* <https://stackoverflow.com/questions/34089085/deploying-to-android-device-not-found-in-visual-studio>
* <https://docs.microsoft.com/en-us/xamarin/android/troubleshooting/questions/update-jdk?tabs=windows>
* <http://www.jiajianhudong.com/question/492765.html>
* <https://github.com/sindresorhus/gulp-imagemin/issues/178>

The below links will give you some general information about the project and the necessary knowhow that you should have for the project.

* <http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/>
* <https://flaviocopes.com/javascript-data-structures-map/>
* <https://thispointer.com/how-to-sort-a-map-by-value-in-c/>
* <https://www.techiedelight.com/sort-map-values-cpp/>
* <https://stackoverflow.com/questions/37982476/how-to-sort-a-map-by-value-in-javascript>
* <https://stackoverflow.com/questions/49625761/how-to-sort-map-of-objects-by-values-ascendingly>
* <https://stackoverflow.com/questions/31158902/is-it-possible-to-sort-a-es6-map-object>
* <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Algorithmic%20Complexity/complexity.html>
* <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Algorithmic%20Complexity/complexity.html>
* <https://www.youtube.com/watch?v=V42FBiohc6c&list=PL2_aWCzGMAwI9HK8YPVBjElbLbI3ufctn>