EESM5060 Embedded Systems

LAB #6

BLE Beaons based indoor positioning

ECE-EESM5060-LAB6
By Prof. Chi Ying Tsui

Version 1.0
SEP 4, 2023

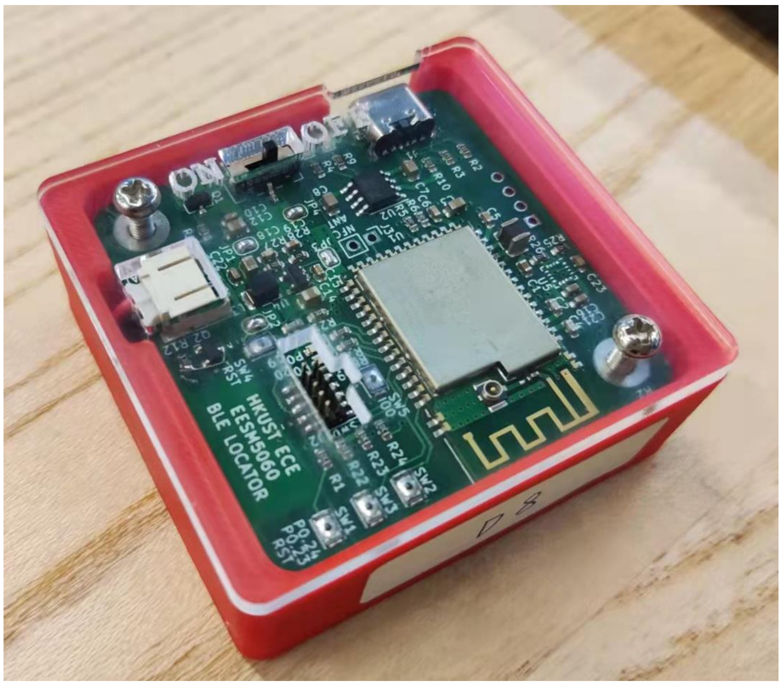
Electronic & Computer Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Introduction

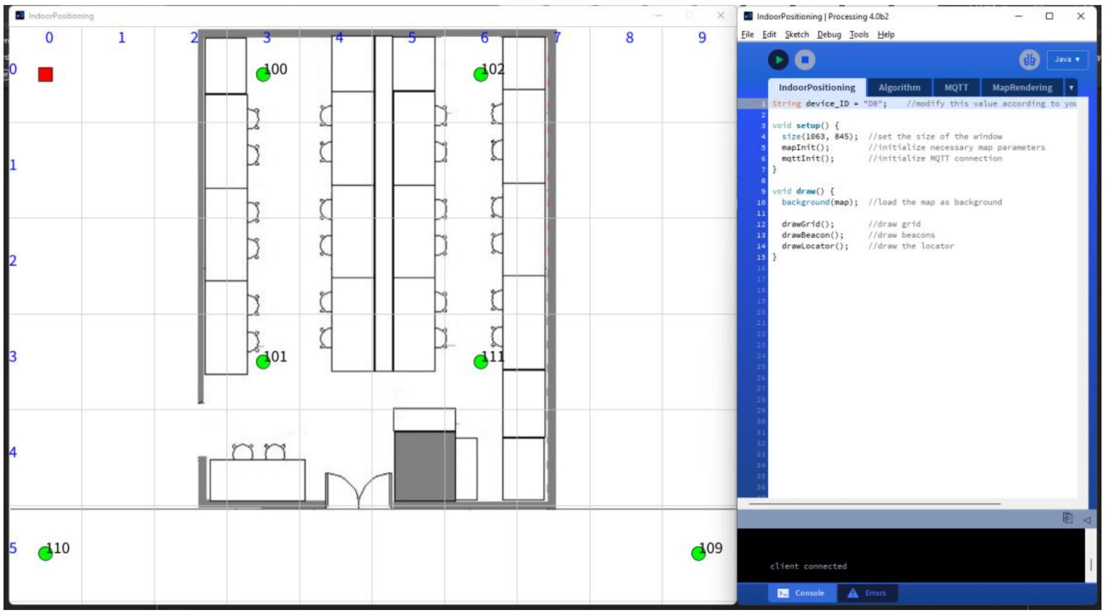
In the age of automation, the ability to navigate persons or devices in indoor environments has become increasingly important for a rising number of applications. While Global Positioning System (GPS) works very well but only in outdoor environments, indoor positioning has become a focus of research and development during the past decade. One of the simplest ways is to make use of Bluetooth signals. In this lab, we will practice a simple indoor positioning software algorithm based on Bluetooth signals with a hardware device.

Lab Materials

1. A hardware set will be provided, which is a box that integrates a micro controller with Bluetooth and Wi-Fi. It can scan the nearby Bluetooth signals and report these data to the server through Wi-Fi. We will call it a locator in this lab manual. After turning on it, it initializes the Wi-Fi connection, and a green LED will light up. The other LED blinks every time it transmits data out. Note that if two green LED flashing at the same time, it means the battery level is low. Please ask TA for help.



2. A simple incomplete program will be provided to cooperate with the locator. It is written by Processing (https://processing.org/) in Java, which receives the data from the locator, performs some positioning algorithms and draws a map to show the locator's real-time position. The positioning algorithm in the provided code is incomplete and thus cannot locate the locator correctly. Your task is to complete the algorithm.



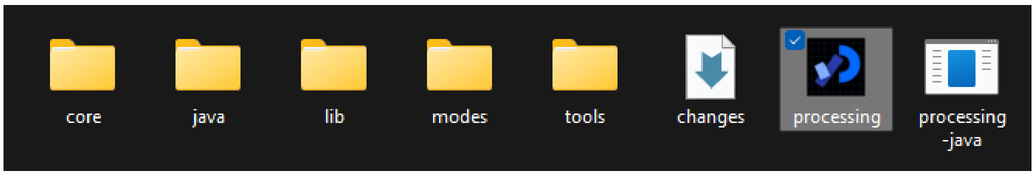
```
String device_ID = "D8";     //modify this value according to you

void setup() {
  size(1063, 845);  //set the size of the window
  mapInit();        //initialize necessary map parameters
  mqttInit();       //initialize MQTT connection
}

void draw() {
  background(map);  //load the map as background

  drawGrid();       //draw grid
  drawBeacon();     //draw beacons
  drawLocator();    //draw the locator
}
```

```
client connected
```

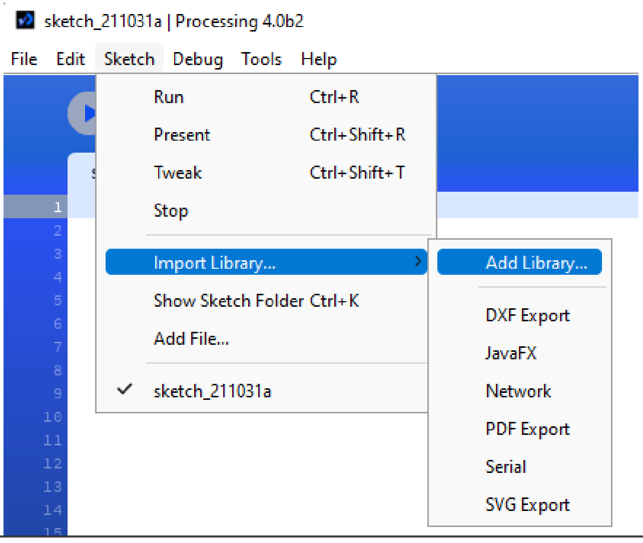
3. In the lab, some Bluetooth beacons are deployed on the wall, which broadcast data via Bluetooth periodically. Each frame of data consists of the beacon's unique ID. When our locator receives this signal, it not only knows the ID of the beacon, but also knows the received signal's strength (knowns as the RSSI value, which means Received Signal Strength Indicator, with a power unit in dBm). It passes this two information to the Processing program. The program obtains the beacon's position by its unique ID, and calculates the distance between the locator and the beacon based on the RSSI value. By doing so, the program can obtain a rough position of the locator. If we make use of more beacons, a more accurate position can be found. In fact, our locator reports four nearby beacon signals with best signal strength every one second.

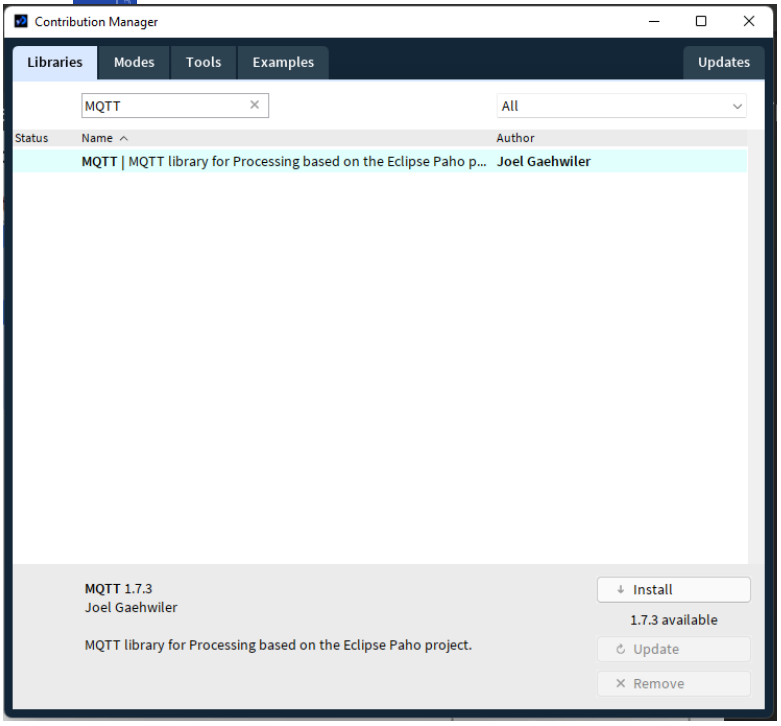
Setting up processing

1. Download processing from https://processing.org/download
2. Unzip it to any folder
3. Run processing



4. Click "Sketch->Import Library...->Manage Library...", search for "MQTT" and click "Install".





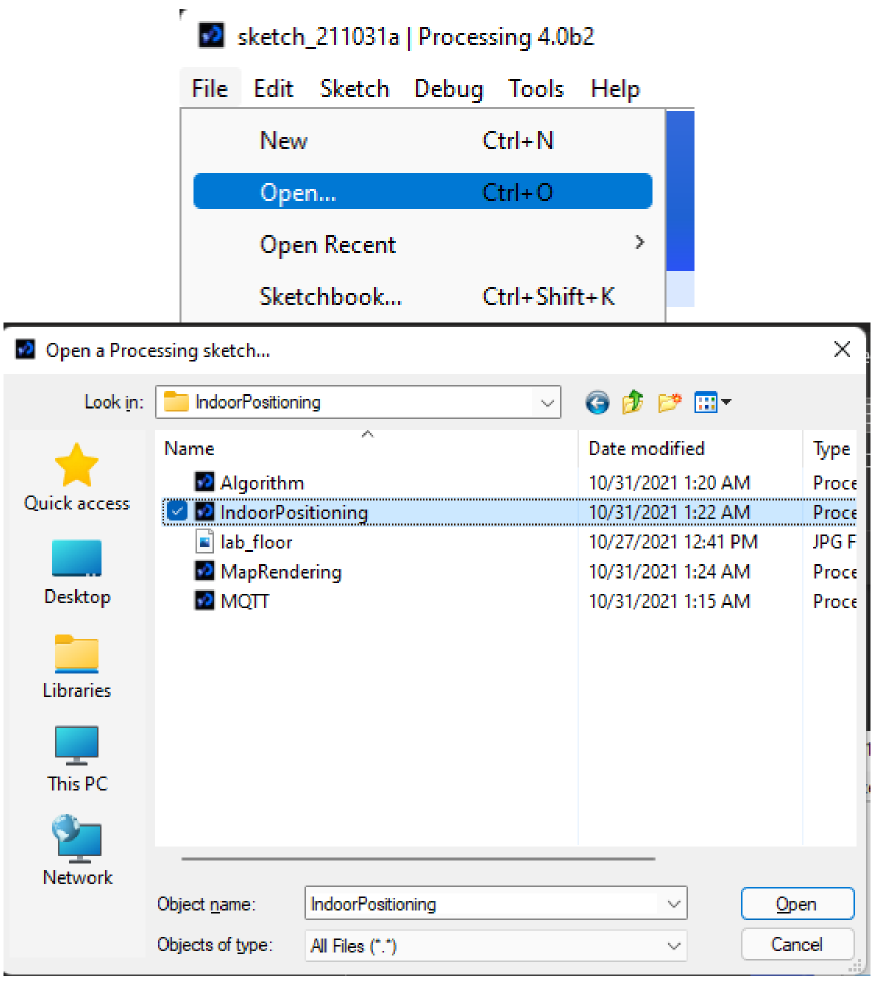
Status	Name	Author
	MQTT | MQTT library for Processing based on the Eclipse Paho p...	Joel Gaehwiler

MQTT 1.7.3
Joel Gaehwiler

MQTT library for Processing based on the Eclipse Paho project.

1.7.3 available

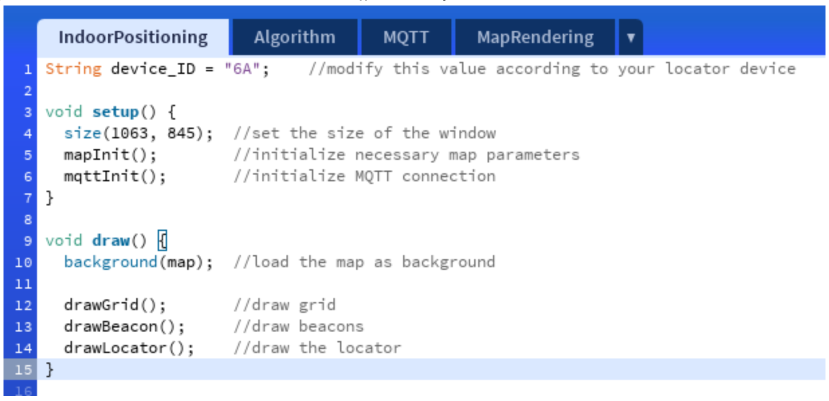
5. Click "File->Open...", navigate to the download "IndoorPositioning" folder and open the "IndoorPositioning" project.


sketch_211031a | Processing 4.0b2
File
Edit
Sketch
Debug
Tools
Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook... Ctrl+Shift+K
Open a Processing sketch...
Look in: IndoorPositioning
Quick access
Desktop
Libraries
This PC
Network
Object name: IndoorPositioning
Objects of type: All Files (*.*)
Open
Cancel

Name	Date modified	Type
Algorithm	10/31/2021 1:20 AM	Proce
IndoorPositioning	10/31/2021 1:22 AM	Proce
lab_floor	10/27/2021 12:41 PM	JPG F
MapRendering	10/31/2021 1:24 AM	Proce
MQTT	10/31/2021 1:15 AM	Proce

Understanding the code

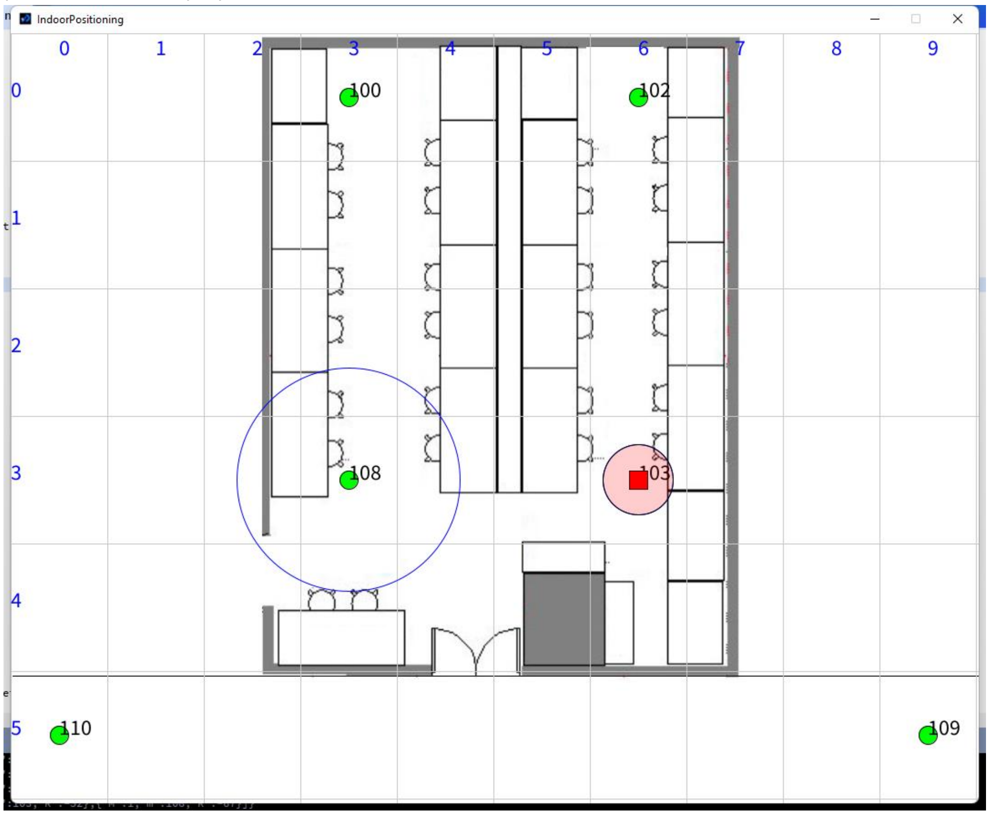
1. The entry point of the program is in "IndoorPositioning" tab, where function "setup()" is executed once and "draw()" is repeated executed.



```
String device_ID = "6A";    //modify this value according to your locator device

void setup() {
  size(1063, 845);  //set the size of the window
  mapInit();        //initialize necessary map parameters
  mqttInit();       //initialize MQTT connection
}

void draw() {
  background(map);  //load the map as background

  drawGrid();       //draw grid
  drawBeacon();     //draw beacons
  drawLocator();    //draw the locator
}
```

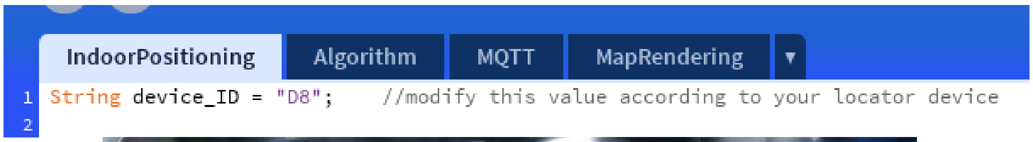
2. When you run the program, it loads the lab map and connects to the locator hardware (through an IoT message protocol called MQTT, just for your information in case you are interested in). It also draws grid, beacons and our locator in different colors.

a. Grid is the grey horizontal and vertical lines, which helps us to discretize our map. Our positioning work will be based on this grid coordinate system, which means the minimum positioning resolution will be one grid.
b. Beacons are the green circles. The nearby numbers are their corresponding id. They map to the physical position of the Bluetooth beacons in the lab, which broadcast Bluetooth signals periodically. The blue circles centered at each beacon is the possible range that the locator is in. The closer your locator is to the beacon, the smaller the range will be.
c. The locator's position is indicated by the red block. Initially its position is not correct. Your task is to modify the code so that it can show the rough position of the physical locator.



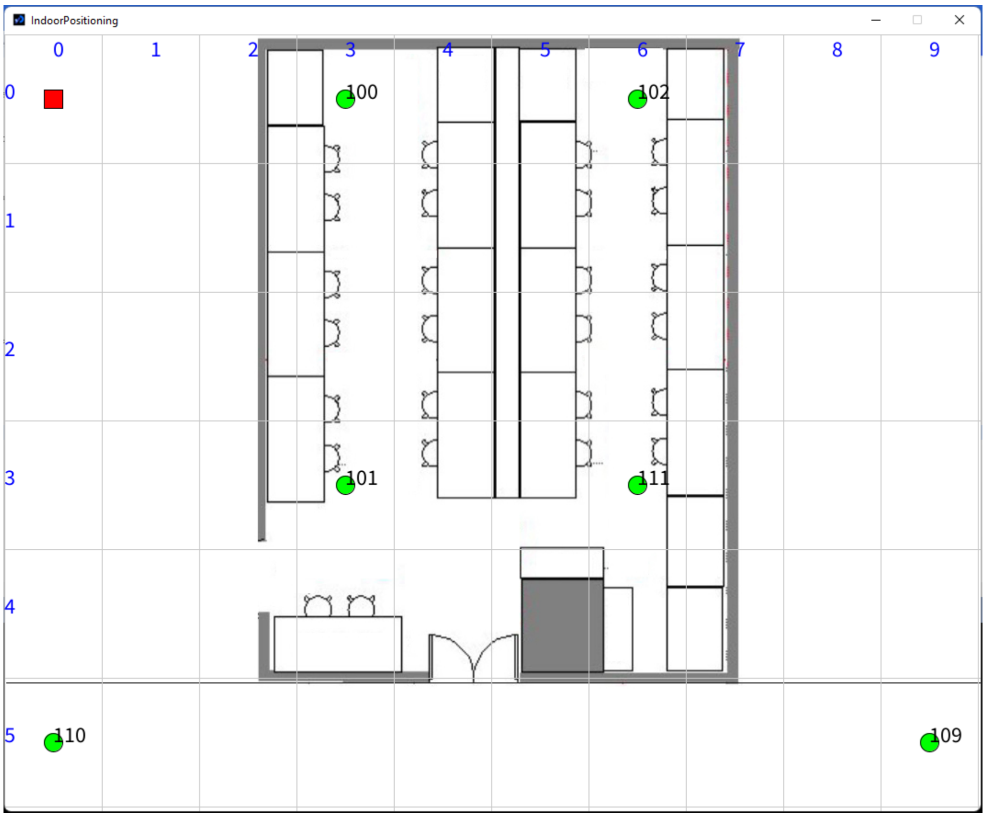
Modifying the code

1. Each locator (the hardware) has different ID. You can find your locator's ID on the device's label. Modify the ID in the first line so that you can receive correct data from your device.



```
String device_ID = "D8";     //modify this value according to your locator device
```

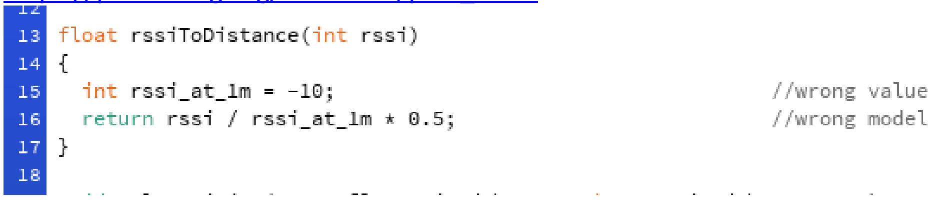
2. When you run the code, the initial position of your locator is incorrect. The beacons' range circles are incorrect as well. This is because the mathematics model for converting RSSI value into distance is incorrect.



3. The RSSI to distance model is implemented in the following function in "Algorithm" tab.

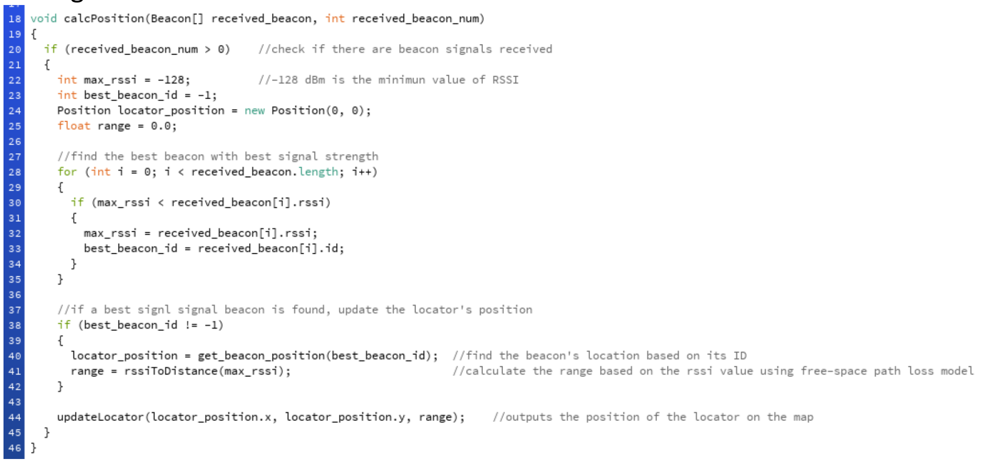
```
float rssiToDistance(int rssi)
```

Your task is to find a proper model for it. In fact, for each environment, due to the complex attenuation, reflection and other factors, there is no perfect model for it. You may try to start with the simplest one, free-space path loss model. You may find the following link useful.
https://en.wikipedia.org/wiki/Free-space_path_loss
https://en.wikipedia.org/wiki/DBm
Please note the RSSI value is in dBm, which means you may need to convert it into linear scale power unit. The official documentation of Processing would also be useful.
https://processing.org/reference/sqrt_.html
https://processing.org/reference/pow_.html



```
float rssiToDistance(int rssi)
{
  int rssi_at_1m = -10;                                     //wrong value
  return rssi / rssi_at_1m * 0.5;                           //wrong model
}
```

4. The locator's position is calculated in the following function in "Algorithm" tab. In function
`void calcPosition(Beacon[] received_beacon, int received_beacon_num)`
the input parameters `Beacon[] received_beacon` is `Beacon` type object array, where each element contains two member variables: id and rssi. It stores the beacons received from the locator. `int received_beacon_num` is the number of elements in the array. Every time a data package is received from the locator, this function is called. It uses a simple method to find the nearest beacon and draw a rough range. If you have time, you may try to modify the code and perform a better algorithm to achieve a better result. For example, using a better calibrated model or adding some filters to filter out unstable data.



```
void calcPosition(Beacon[] received_beacon, int received_beacon_num)
{
  if (received_beacon_num > 0)    //check if there are beacon signals received
  {
    int max_rssi = -128;          //-128 dBm is the minimun value of RSSI
    int best_beacon_id = -1;
    Position locator_position = new Position(0, 0);
    float range = 0.0;

    //find the best beacon with best signal strength
    for (int i = 0; i < received_beacon.length; i++)
    {
      if (max_rssi < received_beacon[i].rssi)
      {
        max_rssi = received_beacon[i].rssi;
        best_beacon_id = received_beacon[i].id;
      }
    }

    //if a best signl signal beacon is found, update the locator's position
    if (best_beacon_id != -1)
    {
      locator_position = get_beacon_position(best_beacon_id);  //find the beacon's location based on its ID
      range = rssiToDistance(max_rssi);                        //calculate the range based on the rssi value using free-space path loss model
    }

    updateLocator(locator_position.x, locator_position.y, range);    //outputs the position of the locator on the map
  }
}
```

5. Here is a hint for getting a good "rssi_at_1m" value. You may use "println" to print out data to the console. You can print out the RSSI value from a specific beacon while keeping your locator at about 1m away from the beacon and observe the RSSI. This can be done in `calcPosition` function as it's called when data received. You may refer to the beacons' IDs on the map and find the corresponding Bluetooth beacons on the wall in the lab. This would help you narrow down the range of "rssi_at_1m" value.
https://processing.org/reference/println_.html