Shelly Christmas Coding Challenge: Documentation

Jernej Cukjati

December 30, 2023

Abstract

For Shelly Christmas Coding Challenge, the Indoor Positioning System (*IPS*) was selected. The *IPS* focuses primarily on the usage of smart Shelly devices with Bluetooth Low Energy (*BLE*) capability. They are used as Bluetooth beacons, which are detected by Android smart phones (with Android, minimal sdk version 33). Smart phone application considers the known locations of the beacons and their Received Signal Strength Indicator (*RSSI*), to compute approximate location of the user. The *IPS* was tested with three Shelly devices, in a single floor of a family house. Nevertheless, this can be applied to a larger building, using a sufficient number of beacons. Furthermore, project can be extended to other Bluetooth devices as well as other technologies (e.g. WiFi). To comply with the project requirements, the primary focus was on Shelly.

1 User interface

Application, which provides the User Interface (*UI*), was written in Java, for Android smart phones, with minimal sdk version 33. In my case Xiaomi Redmi Note 11 pro 5G was used. To make the *UI* (figure 1) as simple and user friendly as possible, it consist only of:

- **custom made toolbar (1)**,
- **TouchImageView (2)**
- and the **button (3)**, which triggers the interpolation algorithm.

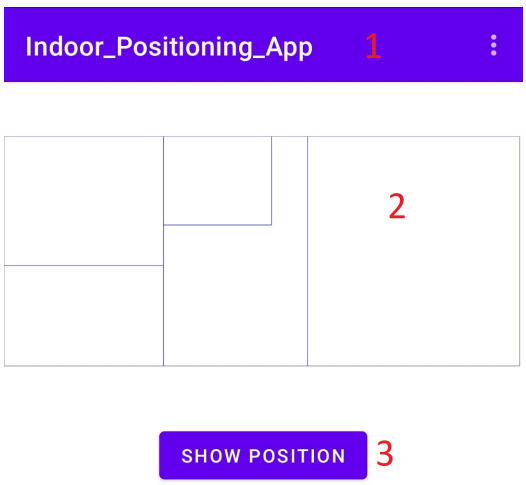

Figure 1: Initial UI

1.1 Toolbar

Toolbar provides basic drop-down menu with two options (figure 2):

- **Show devices**
- and **Show grid**.

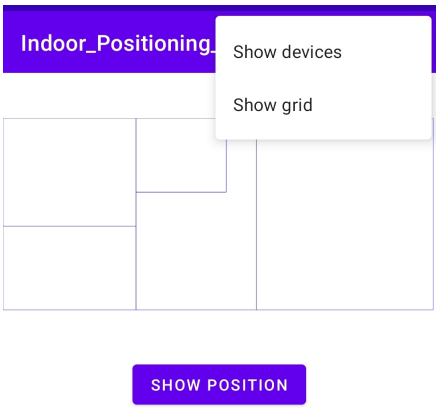

Figure 2: Initial UI

Option **Show devices** allows the user to see the location of the Shelly devices currently detected (figure 3)

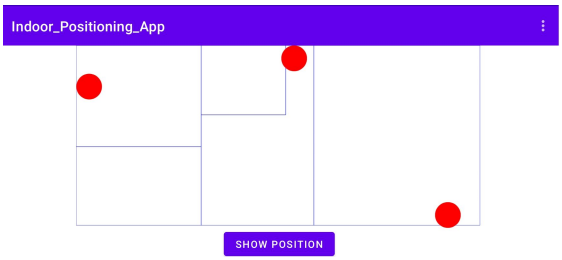

Figure 3: Location of the Shelly devices are marked with the red dots

Show grid shows the gridded area, where each rectangle contains interpolated value of *RSSI*. Those values are considered when calculating the user position (figure 4).

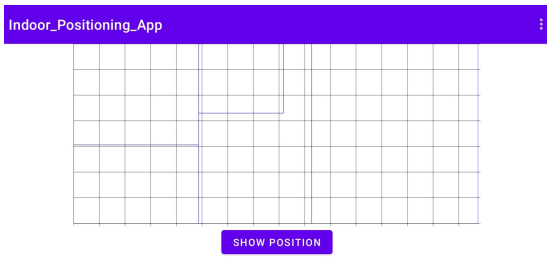

Figure 4: Gridded area

1.2 TouchImageView images preparations

TouchImageView displays considered floor of the selected building. Floor plans were prepared as Multipolygon layers in GeoPackage (*GPKG*) files. This was done by using the QGIS software. It should

be noted that even though these plans are based on the real building, some rooms were modified and details (such as doors, windows, electrical installation etc.) were omitted. This was done for privacy reasons.

To read and display *GPKG* files, GeoPackage Android Lib was used. Initially, it was though that application would considered multiple floors of the building. However, due to the lack of "Shelly beacons", only one floor was considered. Nevertheless, some leftovers for multiple floor usage are still present in our code. Additionally, if anyone would like to update *UI* to list multiple floors, **FloorAdapter** and **Floor** class can be used. They were removed in commit 7103330, on the main branch.

2 Backend

2.1 MQTT

As seen in figure 5, Message Queuing Telemetry Transport (*MQTT*) protocol was used to establish connection between our devices.

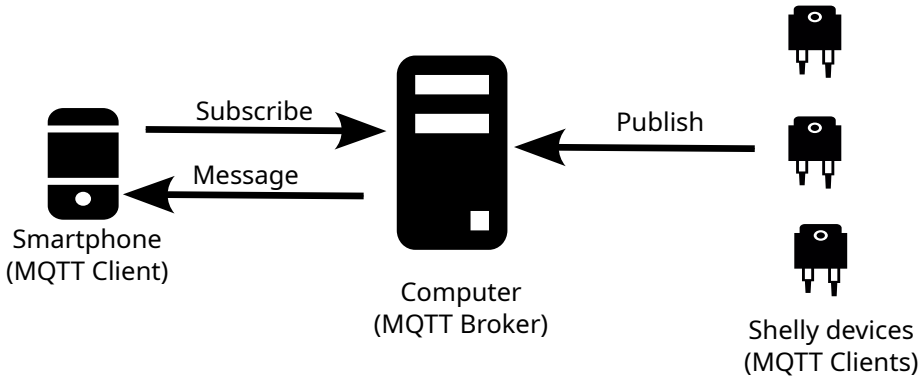

Figure 5: MQTT Device communication

2.1.1 MQTT Broker

Firstly, MQTT Broker was established. I used open source MQTT broker Eclipse Mosquitto. Setup was done according to YouTube tutorial, on a PC ASUS GL553VD.

2.1.2 MQTT Clients

Secondly, Shelly devices were connected to MQTT Broker. This was done with the help of this video. Shelly devices than run a script which would publish a message to our MQTT Broker. In detail, each Shelly device, had hard coded constants, which would represent their position in the building. Secondly, the device utilized its Bluetooth capability to scan for other Shelly devices and obtain their *RSSI* value (figure 6). At the end, the script published MQTT message, which consisted of: source device location, its name, name of the detected device and its *RSSI* (table 1).

The message was separated with the hash sign, for example:
80:64:6f:d0:33:22##1435##0##0##ShellyPlusPlugS-80646FD03320##-71##shellyplusi4-083af2013470.

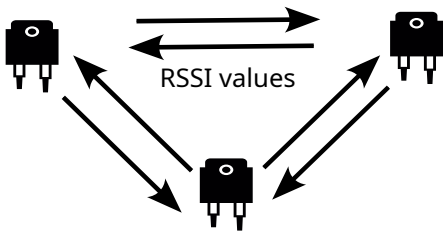

Figure 6: Shelly devices obtaining *RSSI* values

Table 1: MQTT message

Detected Bluetooth MAC	**Source location** X	Y	Floor	Detected name	Detected RSSI	Source name

I have three Shelly devices which support scripting and have Bluetooth. These are two Shelly Plus Plug S, and a Shelly Plus i4.

The proper functioning of the script, was tested firstly with MQTT Explorer (figure 7).

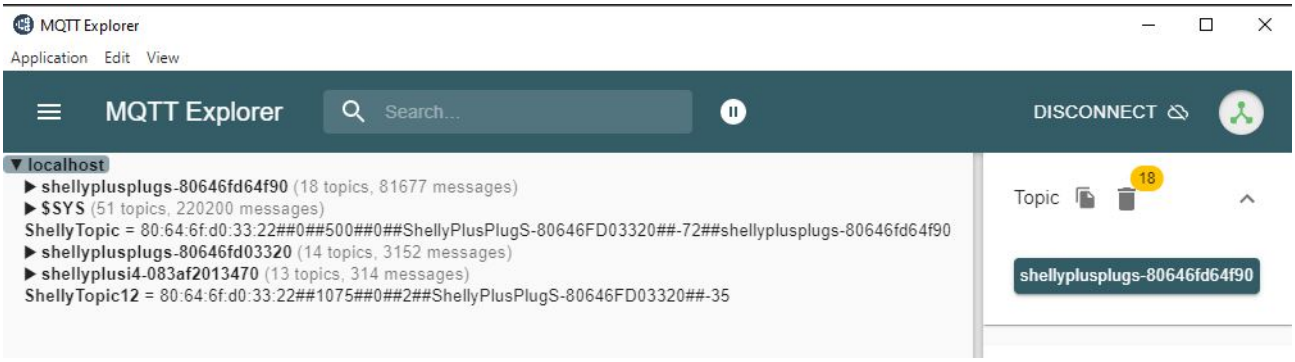

Figure 7: Messages in MQTT explorer

To subscribe to a MQTT topic, published by Shelly devices, MQTT Android Service library was used.

2.2 Android Bluetooth scan

To locate the user within the building, *RSSI* values of Shelly devices from users smartphone were also needed. To obtain this values, functions of BeaconScanner class were taken from existing Indoor Positioning System repository, Beacon-scanner project.

2.3 Indoor Positioning algorithm

To estimate the users location interpolation algorithm was utilized. As seen in the section 1, the observed area was firstly gridded. For each Shelly device obtained through MQTT, *RSSI* values were assigned and interpolated. In my case Inverse Distance Weighting (*IDW*) was used. Lastly, values in each cell were compared against *RSSI* values obtained from smartphone device. The cell, where the sum of distances were the smallest, denoted users position. The algorithm is seen in figure

3 Additional notes

To work location and bluetooth permission must be on for the specific app

4 Conclusion

- Error handling: try-catch handling in java code
  - User friendly messages - TODO - App has a user friendly interface
  - Input validation Shelly script was validated if the obtained data is string and name of the device is shelly
  -validation for MQTT data - TODO
  - API Compatibility - restful
  - well written code - bracket consistency - clear function names - usage of classes - (Proper) indentation and code formating was used
  - Project was published on the public GitHub repository