Mobile and Social Sensing System Project: NoiseMapper

Antonino Nigro
University of Pisa
a.nigro8@studenti.unipi.it

Domingos Marques de Castro
University of Porto
d.marquesdecastro@studenti.unipi.it

Fabio Piras
University of Pisa
f.piras13@studenti.unipi.it

Giacomo Volpi
University of Pisa
g.volpi10@studenti.unipi.it

Guillaume Quint
University of Pisa
g.quint@studenti.unipi.it

***Abstract*—Noisemapper is a mobile application designed to facilitate indoor positioning and environmental noise mapping. Leveraging Bluetooth Low Energy (BLE) technology for indoor localization, the app captures ambient audio data from the user's surroundings. This data is transmitted to a central server where it is stored with data from other users. The app then is able to generate comprehensive noise maps, providing insights into the acoustic landscape of various indoor environments. Users can access these maps to make informed decisions about noise levels in specific rooms. Noisemapper represents a possible advancement in understanding and managing indoor noise pollution, offering a valuable tool for individuals and organizations alike.**

***Index Terms*—Indoor positioning, BLE, Mobile, Noise pollution**

I. Introduction

The objective of this project was to build a mobile application for mapping noise in an indoor environment, as highlighted in many research papers, the topic of indoor location is a complex one and it is difficult to achieve proper precision [1]. However given the circumstances of the use case at hand we are satisfied with room level precision, this led us to choose BLE technology for indoor localization assuming the existence of a 1:1 relationship between rooms and BLE beacons. In regards of the noise detection some papers [2] sustains the idea that smartphone microphone are "good enough" to perform noise level evaluation, however the underlining problem of the different hardware per phone remains. This is the reason why Noisemapper calculate the average value of the noise per room before showing it to the user. Alternatives modern approaches manage to ensure better precision and even 3D sampling, however this solutions require dedicated hardware and are often costly. NoiseMapper instead only requires BLE beacons and a remote storage server making easier to access to the general public and smaller industries.

II. Application highlights and feature

A. Class organization

The project structure is divided in the following way:

- BLEConfig: retrieval and management of configuration data necessary for BLE functionality within the Noisemapper application. It primarily handles the downloading and parsing of a JSON configuration file from a server
- BLEScanner: responsible for managing the scanning of BLE devices, specifically iBeacon devices, using the Kontakt.io SDK. It defines a BLEScannerActivity class that handles the lifecycle of the scanning process.
- Graph: visualization of the noise levels plot in different rooms based on the provided data. It utilizes the Lets-Plot library for data visualization.
- MainActivity: the entry point for the Noisemapper application. It prompts the user to insert the IP to the remote storage server
- NoiseActivity: the core functionality of the Noisemapper application. It is responsible for initializing the user interface, Bluetooth functionality, sensor data collection, data visualization and it handles user inputs
- NoiseMapIO: communication between the Noisemapper application and the server for retrieving configuration files and noise measurements
- NoiseMicrophone: microphone functionality within the Noisemapper application. It initializes and configures the MediaRecorder object to capture audio from the device's microphone.
- ProximitySensor: responsible for handling the proximity sensor, it registers to the events that regards the sensor and expose a property for the NoiseActivity
- PowerGovernor: component that handle all the events related to energy, it registers to battery status events and apply some policies in case of low-battery/low-power mode in order to save battery life
- Server: not part of the core on device application, it acts as a remote storage collecting noise data sample with their relative timestamp.

B. App use case

The app life-cycle can be divided into three main paths: the first is always done at the beginning of each execution, the second one is available when a switch is off and the third one when the former is activated. Fig 1 represents the first use case where the app tries to contact the server to pull the latest configuration file with all the information regarding BLE beacon-room mapping.

Fig. 2 represents instead the second use case in which the user can select a time period of his choice and the app will show the noise map associated with that time period.

The last use case is shown in Fig. 3 where the app uses different threads to collect data, send them to the server and periodically updates the map. Additionally the user can force the update of the map with the press of a button. The user can stop this actions by switching off a button, for simplicity this interaction is not shown in the diagram.

C. *Interesting features*

Here are listed some of the main application feature.

- Each time a new data sample is obtained, it is put into a queue. When enough data samples are gathered, they are all sent to the server, this allows more energy efficiency.
- If the application fails to push the gathered data to the remote server, it tries again at the next update when enough samples are collected without considering the ones not yet sent.
- The configuration of the floor-plan and BLE position is defined into a configuration file that is downloaded each time the app is started. This allows on the fly modifications without touching the app in any way.
- It is possible to use the application without the most recent configuration file, although newly inserted beacons may be missing.
- The app only sends data relative to BLE beacons it knows, this allow the system to be immune to rogue signals and to others BLE beacons in the environment.
- The application checks if the phone is inserted into the user pocket by the use of the proximity sensor, this makes possible to account for noise attenuation due to the microphone being obstructed.
- Noise samples collected for each room are aggregated using an exponential smoothing average (equation 1 with $\alpha = 0.9$), which gives more relevance to samples that have been more recently gathered.
- Energy aware decisions, if the smartphone has low battery or is in power saving mode the application slows down the frequency of map update and noise sensing.

$$\begin{cases} s_0 = x_0 \\ s_t = \alpha x_t + (1-\alpha)s_{t-1} \quad t > 0 \end{cases} \tag{1}$$

III. Live test and results

Live tests were conducted inside the hallway of Polo F of the school of engineering in Pisa. The BLE beacons were set up in different points of the hallway, meaning that, during the tests, there were not clear boundaries between zones like doors and walls, but the overall test results were not impacted too badly due to the beacons distance. Shown in Fig 4 is the initial view of the application where the user can input the IP

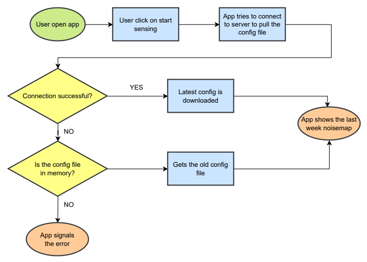
User open app
User click on start sensing
App tries to connect to server to pull the config file
Connection successful?
YES
Latest config is downloaded
App shows the last week noisemap
NO
Is the config file in memory?
Gets the old config file
NO
App signals the error

Fig. 1. Flow of execution on app opening

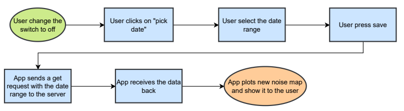
User change the switch to off
User clicks on "pick date"
User select the date range
User press save
App sends a get request with the date range to the server
App receives the data back
App plots new noise map and show it to the user

Fig. 2. Flow of execution when the switch is off

of the server to connect to, then, if there are no samples yet available, the app presents the floor-plan with no noise level like in Fig. 5. Finally after the user starts sensing and moves around the noisemap is updated and colored according to the maximum and minimum noise values registered like shown in Fig. 6.

Alternatively the user, if he is not in sensing mode, can press the ”Pick date” button and the app will show the range picker presented in Fig. 7. After selecting the time interval, the user presses save and the app queries the server and after that it updates the map.

IV. Conclusion

The goal of the project of building a reliable noise mapping application for mobile devices was achieved. The use of battery efficient techniques like the queue push update is a big plus given the criticality of battery duration in modern mobile devices. Regarding the indoor localization technique, the BLE beacon strategy proved to be a good one giving the application a satisfactory precision without requiring complex infrastructure to be installed. The server, although being a very simple remote storage, allowed for operational flexibility and combining data gathered from different users. However given

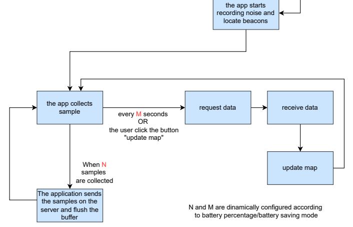
NO
switch off
the user switch on
the app request permission on bluetooth, microphone and GPS
user allows the permission
YES
the app starts recording noise and locate beacons
the app collects sample
every M seconds
OR
the user click the button "update map"
request data
receive data
update map
When N samples are collected
The application sends the samples on the server and flush the buffer
N and M are dinamically configured according to battery percentage/battery saving mode

Fig. 3. Flow of execution when the swhitch is on

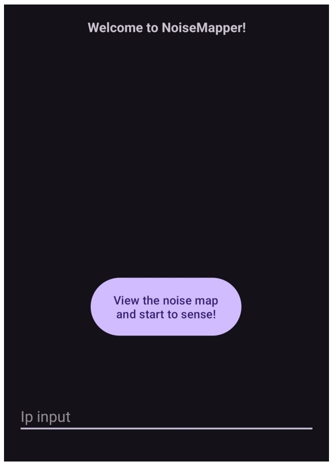

Fig. 4. Initial view of the application

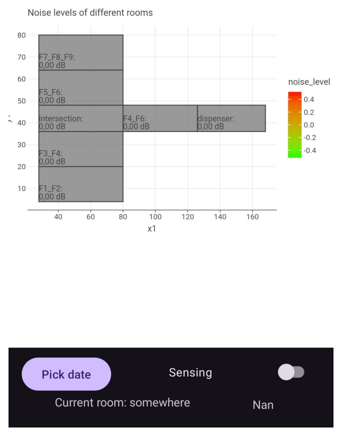

Fig. 5. Application view before sensing

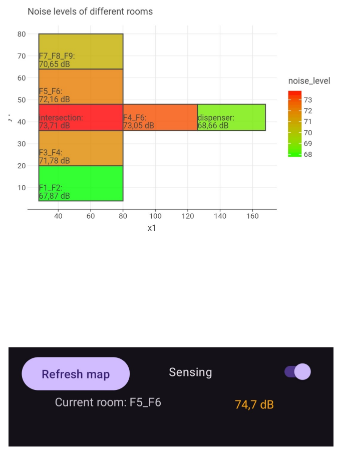

Fig. 6. Application view after sensing

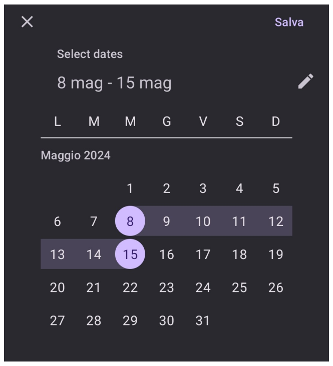

Fig. 7. Application view after of the range picker

the use of the Kontakt.io BLE beacons and their API, the GPS is a requirement for the application to work properly on devices with Andoird API level greater than 31, so for future updates, it would be wise to try finding alternatives to avoid using GPS and be even more energy efficient.

REFERENCES

[1] J. Choi, G. Lee, S. Choi and S. Bahk, ”Smartphone Based Indoor Path Estimation and Localization Without Human Intervention,” in IEEE Transactions on Mobile Computing, vol. 21, no. 2, pp. 681-695, 1 Feb. 2022, doi:10.1109/TMC.2020.3013113.

[2] Z. Qin and Y. Zhu, ”NoiseSense: A Crowd Sensing System for Urban Noise Mapping Service,” 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), Wuhan, China, 2016, pp. 80-87, doi: 10.1109/ICPADS.2016.0020.