University Institute of Lisbon

Department of Information Science and Technology

A Tourism Overcrowding Sensor Using Multiple Radio Techniques Detection

Rúben Dias da Silva

Dissertation in partial fulfillment of the requirements for the degree of
Master in Telecommunications and Computer Engineering

Co-supervisor
Fernando Brito e Abreu, Associate Professor
ISCTE-IUL
Co-supervisor
Rui Marinheiro, Assistant Professor
ISCTE-IUL

October, 2019

Abstract

The motivation for this dissertation came from the touristic pressure felt in the historic neighborhoods of Lisbon. This pressure is the result of the rise in the number of touristic arrivals and the proliferation of local accommodation. To mitigate this problem the research project in which this dissertation is inserted aims to disperse the pressure felt by routing the tourists to more sustainable locations and locations that are not crowded.

The goal of this dissertation is then to develop a crowding sensor to detect, in real-time, the number of persons in its vicinity by detecting how many smartphones it observes in its readings. The proposed solution aims to detect the wireless trace elements generated by the normal usage of smartphones. The technologies in which the sensor will detect devices are Wi-Fi, Bluetooth and the mobile network.

For testing the results gathered by the sensor we developed a prototype that was deployed on our campus and in a museum, during an event with strong attendance. The data gathered was stored in a time-series database and a data visualization tool was used to interpret the results.

The overall conclusions of this dissertation are that it is possible to build a sensor that detects nearby devices thereby allowing to detect overcrowding situations. The prototype built allows to detect crowd mobility patterns. The composition of technologies and identity unification are topics deserving future research.

Keywords: crowding , real-time detection, multiple technologies detection

Resumo

A motivação para a presente dissertação surgiu da pressão turística sentida nos bairros históricos de Lisboa. Esta pressão é a consequência de um crescimento do número de turistas e de uma cada vez maior utilização e proliferação do alojamento local. Para mitigar este problema o projeto de investigação em que esta dissertação está inserida pretende dispersar os turistas por locais sustentáveis e que não estejam sobrelotados.

O objetivo desta dissertação é o de desenvolver um sensor que consiga detetar, em tempo real, detetar quantas pessoas estão na sua proximidade com base nos smartphones que consegue detetar. A solução proposta tem como objetivo detetar os traços gerados pela normal utilização de um smartphone. As tecnologias nas quais o sensor deteta traços de utilização são Wi-Fi, Bluetooth e a rede móvel.

Para realizar os testes ao sensor, foi desenvolvido um protótipo que foi instalado no campus e num museu durante um evento de grande afluência. Os dados provenientes destes testes foram guardados numa base de dados de séries temporais e analisados usando uma ferramenta de visualização de dados.

As conclusões obtidas nesta dissertação são que é possível criar um sensor capaz de detetar dispositivos na sua proximidade e detetar situações de sobrelotação/apinhamento. O protótipo contruído permite detectar padrões de mobilidade de multidões. A composição de tecnologias e a unificação de identidade são problemas que requerem investigação futura.

Palavras-chave: apinhamento, deteção em tempo real, deteção de múltiplas tecnologias

Acknowledgements

I would like to express my gratitude to Professor Fernando Brito e Abreu, one of my co-supervisors, for inviting me to this research project and for the guidance throughout my dissertation project.

I would like to express my very great appreciation to Professor Rui Neto Marinheiro, one of my co-supervisors, for the patient guidance, support and useful critiques of this research work.

I would also like to thank all the following professors for their collaboration and support for the realization of this dissertation: Professor Amério Rio; Professor Alexandra Paio; Professor João Oliveira.

I would like to thank to Vitrivius FabLab as ISCTE-IUL for the support in the construction of our prototype and a special thanks to João Sousa that co-designed and manufactured all the cases for the prototypes.

I am thankful for IT- Instituto de Telecomunicações and ISTAR- Information Sciences and Technologies and Architecture Research, the two research departments that gave me support and all the conditions to fulfill the objectives of my dissertation.

I am particularly gratefull for the support and fellowship from all the co-workers in this research project, Ana Rita Peixoto, João Nuno Virtudes and Duarte Almeida.

My special thanks to my girlfriend that supported me all the time, always showing a precious smile and always showing me an helping hand when needed and for always believing in my capabilities and in my work.

I thank my family for the encouragement and constant guidance. Thanks to my older sister for helping me in every way she could, and my parents for the constant love and support. A special thanks to my grandmother for always having warm and wise words to encourage me. Finally I would like to thank to my friends for their constant cheering up and support.

Contents

List of Figures

List of Tables

Abbreviations

API- Application Programming Interface.

ASNI- Abstract Syntax Notation One.

BLE- Bluetooth low energy.

DBMS- DataBase Management System.

EnodeB- Name given to one cell in a mobile network.

GSM- Global System for Mobile communications.

GUTI - Globally Unique Temporary Identifier.

HTTP- Hypertext Transfer Protocol.

IMSI - International Mobile Subscriber Identity.

IoT- Internet of Things.

JSON- JavaScript Object Notation.

LTE- Long term evolution.

LoRaWAN- Long Range Wide Area Network.

MAC address- Media Access Control address.

PHP- Hypertext Preprocessor.

REST- REpresentational State Transfer.

SDR- Software Defined Radio.

STC- Sustainable Tourism Crowding project.

UE- User Equipment.

UMTS- Universal Mobile Telecommunications System.

Wi-Fi- Technology for transmitting data based on the IEEE 802.11 standards.

Chapter 1

Introduction

1.1 Context

The Sustainable Tourism Crowding project (STC), in which scope the current work was developed, aroused from the impact of touristic activities in the population that lives in the historic neighborhoods of Lisbon. The increasing impact of the touristic activities could be traced to two main reasons, the rise of the number of tourists visiting the city and the proliferation of local accommodation.

Portugal, and most specifically its capital, Lisbon, had a sudden increase in the number of tourists and this increase is now predicted to be a constant in the near future. This is sustained by the predictions done by the world travel and tourism council in its annual reports [WTTC2018]. Also, in the case of study of Lisbon, the increasing notability as a touristic attraction that is the reflection of the awards and distinctions won, led to the increasing amount of tourists visiting the city. This rise of visits, together with the inauguration of a cruiser port close to the historic neighborhoods, led to the overcome of load capacity of different areas in a phenomenon called tourism overcrowding.

The proliferation of local accommodation highlighted the impact of touristic activities in the historic neighborhoods, by exposing the conservative culture of these communities to the more modern and upfront culture of the tourists since tourists and locals live in the same confined area. Tourism overcrowding also impacts the buildings and cultural heritage of these neighborhoods. The deterioration of these areas is caused by an increase in urban noise, less effective urban cleaning, reduced privacy and loss of authenticity, among other aspects.

1.2 Motivation and relevance

The increase in touristic pressure in Lisbon, mainly led by the phenomena of tourism overcrowding, is creating an anti-tourism feeling in the city, that if not solved could become an even bigger problem. The problem is not as badly felt as in other European cities (as in Barcelona or Venice)[Coldwell2017] where the anti-tourism feeling has resulted in public protests, uprising and even cases of violence towards tourist [Colomb and Novy2009]. In the research project in which this dissertation is inserted we aim to reduce the impact of touristic activities and its pressure in the city by dispersing the overall load of touristic activities through a larger area. By dispersing the load of these activities we intend to reduce the overall feeling of this pressure. Other approaches used to deal with this situation are: 1) Smooth visitors over time by avoiding only seasonal tourism, by scheduling activities in a less congested time; 2) Regulate accommodation supply to control the maximum amount of tourists in the city; 3) Adjust pricing to balance supply and demand, for guaranteeing that the load capacity is not surpassed; 4) Limit the access and activities to again limit the maximum amount of tourists in the city. From all the other options presented we have chosen this approach because it is the one that maintains the advantages of touristic activity, mainly its importance in the economics of the city, while mitigating the disadvantages by reducing the touristic pressure felt.

The STC project aims to reduce the touristic pressure by routing tourists through areas that are less visited and have higher sustainability. It will achieve this routing of tourists by providing a free to use mobile application that will suggest routes that will avoid crowding areas and promote sustainable points of interest. While the points of interest could be gathered previously with the help of the administration council of the city, crowding values are not, and for the effectiveness of this approach the crowding values need to be gathered in real-time as they change quickly and unpredictably.

In this dissertation, we aim to mitigate this problem by performing a real-time detection of how crowded a location is by detecting the number of mobile devices present in the area. The later is a good surrogate of the number of people, since there is a strong penetration of smartphone usage in most western countries (in the range of 75% to 80% [Newzoo]). We aim to detect the devices in real-time by detecting the trace elements generated by the usage of the different wireless technologies, being them Wi-Fi, Bluetooth, or the Mobile Network.

1.3 Research questions

This dissertation goal it to answer to the following research questions:

1. Is it possible to create a sensor that detects crowding in real-time by listening to the trace elements of the usage of Wi-Fi, Bluetooth, and the Mobile Network?
2. Is it possible to combine the information from the different technologies detected to allow an accurate estimation of the number of people present in the vicinity of the sensor?

1.4 Research steps

The steps to be taken for answering the first research question are:

- Detect nearby devices using Wi-Fi, Bluetooth and the Mobile Network;
- Develop a structure for uploading the results of the detected device;
- Develop a database to store the results gathered from the sensor;
- Implement a data visualization tool for reading the results in real-time;
- Develop a prototype for the implementation of the crowding sensor;
- Deploy the crowding sensor in real-usage scenarios.

The steps to be taken for answering the final research question are:

- Compare and analyze the data gathered from the different technologies;
- Guarantee the singularity of detection of one device in the different technologies;
- Create an overall metric of crowding.

1.5 Method

This dissertation followed the Design Science Research methodology as presented in [Vaishnavi and Kuechler2004].
The different phases of this methodology and its contribution in this dissertation are:

1. Inception phase - In Chapter 1 we identify the context and motivation of this dissertation as well as the problem we aim to solve which is diminish the tourist pressure by creating a sensor that will detect crowding in real-time and redirect tourists to less crowded locations.

2. Research phase - The research phase is present in Chapter 2, where we present the state of art of the different approaches in the literature to detect persons and how it is possible to detect devices in the different technologies that our sensor will use.

3. Implementation phase - The implementation phase consisted of the development of a prototype of the crowding sensor. We present in Chapter 3 the architecture used for our sensor and in Chapter 4 the implementation steps of the propose solution as well as the construction of the prototype.

4. Validation phase - The validation phase was done by performing three types of tests: a) Preliminary tests for perceiving the behavior of the sensor in different locations; b) 24/7 tests in different scenarios in our campus; c) Tests in a real application of the solution.

5. Conclusion phase - The conclusion phase is present in Chapter 6 where we present the conclusions and future work.

1.6 Contributions

The main contributions of this dissertation are:

1. An innovative approach for detecting the crowding.

2. An overview of the information that devices leak and the main vulnerabilities of the technologies used in smartphones.

3. The construction of a functional prototype for detecting devices in different technologies simultaneously.

4. The application of our sensor in a real scenario.

5. A published paper in a conference where the first results of our solution were shown [Rúben Dias da Silva2019].

1.7 Dissertation organization

This dissertation is organized as follows. In Chapter 2 we present a state of the art review of overcrowding detection techniques and possible techniques and attacks do detect smartphone usage. In Chapter 3 we present the overall architecture of the proposed solution. Chapter 4 presents our implementation for detecting smartphones in real-time and Chapter 5 the test and validation for our solution. Chapter 6 addresses conclusions and ongoing and future work.

Chapter 2

Literature Review

In this chapter we present the overall state of art of crowding detection and how it is possible to detect devices exploring the different flaws of each technology used by our sensor. This Chapter is organized as follows: in section 2.1 we summarise the works previously done in detecting crowds and people movements and their different approaches. Finally, in section 2.2, we present the technologies that our solution is based on and its usage and vulnerabilities.

2.1 Technology applications

Technology in tourism is relatively recent and directly related to a more accessible mean of getting information quickly in any place in the world. So, the appearance of smartphones and their accessibility has risen the interest of more technological support for tourists. This technological support was essential to support the rise in local accommodation worldwide, as it becomes possible to reserve local accommodation easily using online platforms. Also to support all the feedback and reviews generated by tourists it was essential to create platforms to consolidate and share all this information for easy access by any user.

After the growth of the supporting network for tourism, efforts were focused on processing this data for improving tourism quality and quantity. The field more interesting to be analyzed for this thesis aim is the field that is specialized in detecting overcrowded locations by detecting how many persons are in a confined area. The works previously

done in this field could be arranged in five different approaches being them sound and image or video-based, using social media, using the mobile operator's data or using wireless spectrum analysis.

2.1.1 Sound sensing approach

In a sound sensing approach, several microphones are distributed in a given area and gather the sound emitted by persons, cars, trains, buses and transmitting that information to be processed afterward. This approach was already tested in some cities in Europe, like Barcelona [Farrés] where a network of sensors was distributed and gathered ambient sound of their surroundings. That information is then transmitted to be processed in the cloud, to generate heat maps.

The need for future processing the data is one big disadvantage of this type of approach as sound needs to be analyzed, processed and classified for distinguishing the different sources of every sound and determine the number of persons present in the area. Also, the processing cost of doing this activity is very high as sound analysis and processing is a very demanding and difficult activity. Another disadvantage is the use of network bandwidth for sending a stream of sound constantly for the cloud. Which makes it impossible for implementation upon a low cost network such as a Long Range Wide Area Network (LoRaWAN) communication protocol.

Previous work in sound analysis and classification into different sources and events could be seen in [Plumbley2010], [Mesaros et al.2010], [Agarwal et al.2015] when real-life recording and single event sounds were analyzed. Classification is achieved by sampling the recordings and analyzing them to gain data from isolated events (e.g. a car passing by) or a continuous event (e.g. a conversation). After this analysis, the recording is associated with the events identified in the recording and then determine people or cars density in the range of the sensor.

Besides using a network of microphones spread around a given area, some approaches are using a people-centric approach by using their smartphones for gathering data and determine the surrounding environment. Using the built-in microphone in smartphones it is possible to listen to sound present in the area and determine what is happening around the smartphone. In [Lu et al.2009] it is created an application for iPhones called SoundSense that works either in online or offline mode, classifying sound events detected

by the microphone. Another application for sound classifying in smartphones is an application called VibN [Miluzzo et al.2011] that lets its user know what events occur around them, and what other users have detected using the application nearby.
While in people-centric approaches the detection is dependent on the usage of a given application, the first approach presented depends on the size of the network of sensors implemented. Sound analysis is demanding in computational power and network bandwidth, although less than other alternatives like image processing. However, achieved precision and detection range is much lower and it also raises similar privacy problems, since conversations may be recorded and transmitted to the cloud.

2.1.2 Image and video capturing approach

This approach may use a dedicated or an existent network of cameras, usually required for security purposes in major cities, to detect the number of individuals and their behavior in the view range of cameras. In this approach, images are processed for identifying crowds characterized by a high number of persons in a small confined area. Normally images are processed simultaneously, with sound for better results in identifying crowds [Andersson et al.2010].
The most difficult part of this approach is to identify without false positives every person in a crowd using image processing of every frame. This can be achieved using a time-of-flight camera, a camera that is a low-resolution depth focused camera as used in [Stahlschmidt et al.2016]. This work studied an implementation where a time-of-flight camera was used in a top-view position for people detection and their movements in a crowd. Also, it presented a metric for the density of people present in a given area that could be useful for detecting overcrowded areas. For better results, instead of only one time-of-flight camera, it can be used multiple depth cameras for generating 3D models of the environment and improve the counting of people and reducing the number of false positives. One example of the approach, using several depth sensors, could be seen in [Wetzel et al.2018].
Besides using a depth sensor and time-of-flight cameras, normal surveillance cameras can be used for detecting and counting the number of people in the environment. This approach is based more on image processing to identify a person in an image usually using artificial intelligence and machine learning. In [Peng et al.2015] a multiple camera scenario is used and the analysis is done using Bayesian networks for identifying each

person in a crowd and counting them.

Any of the previous alternatives present some serious disadvantages for detecting overcrowding in real-time. Image and video processing is a computational demanding procedure, not amenable for edge computing. Communication and network bandwidth costs between the sensor nodes and image processing servers (e.g. in the cloud) may be significant. Furthermore, these approaches raise serious privacy concerns, requiring complex authorization procedures, since images of individuals will be captured and processed and could cause some serious privacy issues.

2.1.3 Social media approach

In this approach, the studies gather the information that is published in the social media networks and by the geo-location data present in this information it is deducted the number of persons present in a given area. The most chosen social media network in these studies is Instagram[1] because of the direct related georeferenced data with every post made in the network.

One of the studies gathered data from posts in New York City and tried to join them in clusters to analyze the flow of persons in the city [Domínguez et al.2017]. The detection of the information in this study was not in real-time, as it was gathered with the objective of further studies in the flow of people. The information was gathered in 22 weeks and used in 30 minutes clusters to deduce the usual behaviour of New York citizens in the area. The proposed approach of clustering data and uniting the data of several locations that are in a short range is very interesting but the conclusions are somehow short as the data show always the same patterns of movement.

Another interesting study using social media gathers from Twitter[2] and Instagram simultaneously and almost in real-time (by only processing information detected in the last hour) [Ranneries et al.2016]. In this work, the information is analyzed to detect events in real-time, and the working prototype built does it with high location precision, but with several problems with false positives. This study gathered data and then analyze it using several filters. The first one was filter information by language followed by the filtering duplicates and finally, they extract the location of the post. After filtering the information, they process it by clustering it in hashtags and locations. After all this

[1] https://www.instagram.com
[2] https://twitter.com/

processing finally, they merged the duplicates for obtaining the number of persons in a certain area.

Both previous approaches are dependent on the information shared by users in social media and all that information needs to be public for permission to analyze that information. So, the information they could gather and work on is limited and probably just a portion of the reality in a social media. Also, besides being a part of the social media spectrum (public posts) they are dependent of having a location associated to the post directly (by directly saying the post was in a certain location) or indirectly (by a photo of a monument or alike that could identify a location). The data available is also dependent on the usage of the social media in question, so if a person does not use such social media it will not be detected by the program, so the analysis of this approach is limited in the number of persons detected by several factors which could not reflect the reality of the population. In other words, social media based approaches will present a lot of false negatives (low recall).

Another approach could be done using the instant message applications used in smartphones for locating and estimating the number of persons in a given place. Whatsapp[3] messages where analyzed in [Anglano2014] and by their geo-location fields it was possible to identify where and when a person sent a message. This information is private of every device so it could be only achieved on a full scale by the instant message service provider. This could be a very privacy-invading situation, as our instant message provider would know at all times our location and our usage patterns.

2.1.4 Using mobile operators' data

Mobile operators have at their disposal huge amounts of data from the usage of their networks, but the challenge presented in this type of approach is to generate relevant metrics using those big data sets. The real problem is to generate real-time data that could, for example, detect crowds in confined areas.

Spanish's Telefonica [Park et al.2017] has invested in processing historical data, namely for prediction of future movement patterns. Vodafone Analytics[4] is a well-known example of this exploration of data and provides a set of metrics and their evolution over time. The metrics are usually presented in geo-referenced maps, where spatial-temporal evolution

[3]https://www.whatsapp.com/

[4]https://geographica.com/en/showcase/vodafone-analytics/

analysis is possible. Portuguese's operator NOS used millions of cell tower pings from roaming phone users to build a Tourist Information Portal[5] that offers several indicators to allow business owners identify potential target areas and municipalities across the whole country to analyze how to allocate resources more efficiently.

All the presented tools aim to predict and detect a pattern in crowd movements using past gathered data. The main disadvantage is that the data is not obtained in real-time and is only a prediction of the movement of the crowd. Overcrowding being a very quick and unpredictable phenomenon would need a more quick response, for example, small events and conferences are not detected in these predictions, as well as sudden changes in tourist's behavior.

2.1.5 Wireless spectrum analysis approach

Early works, such as [Kashevnik and Shchekotov2012], have analyzed the architecture of different indoor positioning systems and discussed the properties and drawbacks of solutions supported by smartphones. Most solutions used an active positioning approach and require willing users to install an app. Alternatively, passive monitoring is possible by the owner of the communication platform (see the previous section). However, when that is not the case, third parties have resort to wireless spectrum and protocol analysis. The evolution and increasing affordability of SDR and open-source hardware and compatible open source software has allowed for a more flexible spectrum and protocol analysis and has sped up research on wireless eavesdrop and active detection approaches. These approaches usually explore protocol characteristics and/or small security leaks of information. Previous research works were usually focused on a specific technology, but there is a recent comprehensive work [Yang and Huang2018] that provides in-depth analysis and tutorials for several technologies. When using wireless technology for presence detection, one should consider that there is a tradeoff between range and precision as different technologies and protocols have different range limitations. In section 2.2 is presented a study of the current state of the art for detecting trace elements in smartphones using different technologies (Mobile Network, Wi-Fi and Bluetooth).

[5] http://customers.microsoft.com/en-us/story/nos-spgs-media-telco-azure-sql-r-server-portugal

2.2 Smartphones trace elements analysis

Smartphones (and wearables, despite being still much less used) have a wireless footprint since in their normal usage as the result of the different technologies that they use. For example, when browsing the web in a smartphone it is generated noise from the contents present in a website being transmitted to a smartphone over the air(either by wi-fi or the mobile network). This information flow through the air creates trace elements that can be heard by anyone listening in the specific frequencies where the data is transmitted. For this research, it is very important to highlight the work done in [Yang and Huang2018] a book where it is described several eavesdropping techniques and tools. This book is useful for initializing the work in sniffing information introducing the reader to very useful tools.

The study was focused on the technologies present in smartphones that generate a bigger amount of wireless trace elements.

2.2.1 Mobile network

The mobile network is a composition of base stations that are responsible for one cell each, having each cell different properties. Our smartphones interact with those base stations for connecting to the network and being able to communicate. This part of the network is purely wireless and it could be listened by any other device nearby. What the proposed solution aims to do is to listen to the trace elements resultant of the interaction between the user equipment and a base station. The trace elements that could be found are different in the different protocols present in the mobile network and for that, it is important to understand the evolution of the network until the present day.

The current mobile network is the result of a series of evolutions and consecutive upgrades and is usually known as generations, each representing a big change in the focus of the network and the technologies used. Generation retrocompatibility (except to the first generation that is no longer supported) is possible through the parallel usage of the different technologies. The mobile network was born in 1979 and the early to mid-1980s as an expansion of the public switched telephone network, but with wireless capabilities. In this first generation the only feature available were voice calls and all communications

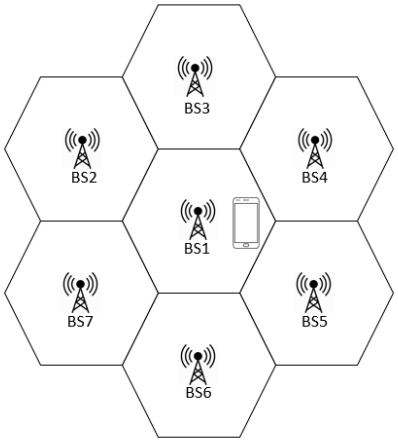


FIGURE 2.1: Base station distribution on the mobile network.

were done using analog signals.

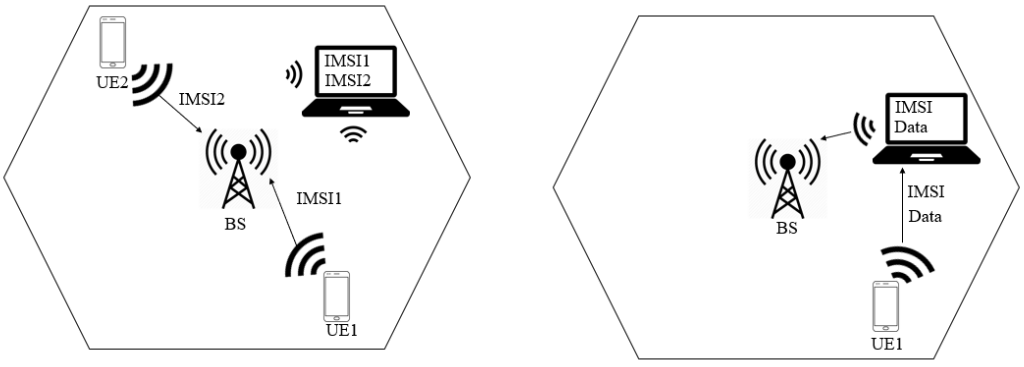
The second generation of the mobile network, also known as 2G and Global System for Mobile communications (GSM), represented the transition to digital signal as all the information transmitted stopped being analog to be only digital. This change provided the means not only to voice calls but also SMS (Short Messages Services). The third evolution of the mobile network 3G, enabled fast communications and having a wider bandwidth enabled the usage of new services like live streaming, voice calling or IPTV. The fourth generation is also known as 4G or Long Term Evolution (LTE) fulfills the need for higher speed and bigger data usage of its users. It also allowed for the other devices like laptops to connect to the network as it is an all IP based network.

All these generations have different communications protocols and requirements and so for listening and gathering trace elements of smartphones usage every generation as different flaws and approaches for exploring those flaws. Important to note that for the final sensor only passive attacks are considered as at any point interfering with the real networks is a possibility and goes against the goal in this project.

2.2.1.1 GSM

GSM was developed in the mid-1980s and first deployed in 1991. Being by today standards an old protocol it was developed with some design flaws being one of the biggest flaw is authentication procedure. On GSM networks the base stations have all the power

and control of all communications and by default do not authenticate themselves to the user equipment's. User equipment's need to authenticate themselves to the network but the network does not have the obligation to authenticate to the user. The lack of double authentication allows for a fake base station to seem a true base station and control all communication of the user equipment's in is range. Also as user equipment's when realizing the authentication process sends their unique identifier ISMI (International Mobile Subscriber Identity) unencrypted, any other device listening could intercept the message and know that the specific user equipment is in the range of detection. The second approach presented is fully passive as the detection device only needs to be listening in the right frequency to intercept those messages and compute the number of user equipment's using the mobile network in a given range. This kind of attacks are usually called of IMSI catchers as the aim to collect the IMSI details of the users in a determined area.



(a) Passive attack

(b) Active attack

FIGURE 2.2: Possible attacks on GSM.

This attacks could rapidly escalate to a man in the middle attacks as the IMSI of the user is known by the fake base station and it could start a connection with the real network with that data and serve as a bridge between the user equipment and the real mobile network. [Strobel2007]. Active attacks in GSM where used by security agencies to tap the communications of suspects and track their movement. One example of these devices was StingRay a device used by the military and intelligence of Canada, the United States, and the United Kingdom. This usage aroused privacy concerns leading to the ban of active IMSI catchers use by law enforcement's [Bates2017]. Our solution will them explore only the passive option of this attack and also for privacy concerns anonymise all data that could be used to track one single user. Our solution will then

only generate a number corresponding to how many user equipment's are there in range without knowing who they are and what are they doing.

2.2.1.2 3G/UMTS

The third generation of mobile networks implemented new authentication and key agreement mechanisms that allowed for the network to be authenticated by the user equipment. This measure eliminated the feasibility of a man in the middle attack in pure UMTS networks. This security measure only works in pure UMTS networks and that may not occur in reality. Due to the necessity of full compatibility between GSM and UMTS networks, it is possible to continue to do a man in the middle attack if impersonating a GSM network or a user equipment or even by soliciting a GSM security measure over UMTS base stations [Meyer and Wetzel2004]. This type of attacks take advantage of on key feature of UMTS and the mobile network in general, the need for full compatibility of all generations, this means that a UMTS base station (ENodeB) needs to also fully support GSM requests and connections.
In figure 2.3 are present the messages that are exchanged between a user equipment and a base station in UMTS. The first step is to establish a valid connection between both devices and after that, the user equipment needs to send an identifier to authenticate to the network. This identifier is the TMSI a temporary identifier for the user equipment previously defined by the network. If the TMSI is not valid due to being expired or the network is not being able to identify the TMSI it requests the true identity of the device, is IMSI, this could be seen in steps 3 and 4. The next messages exchanged aim to authenticate the network to the user realizing a challenge and request authentication. The last step is the base station defining the security protocols and modes chosen for communicating with the user equipment now on.

Analyzing the messages exchanged it is clear that a totally passive approach to know how many devices are there in a given are is to listen to the messages exchange and gather all the TMSI of the user equipment's nearby using the logic from IMSI catcher on GSM. The only problem with this approach is that it could not be a direct correlation between the number of TMSI listened and the number of user equipment's as the network could ask as they wish a new TSMI to a user equipment. For solving this problem usually it is used a semi passive approach. In this approach, a device impersonates as a base station a takes advantage of the timing where it needs to authenticate to the user equipment.

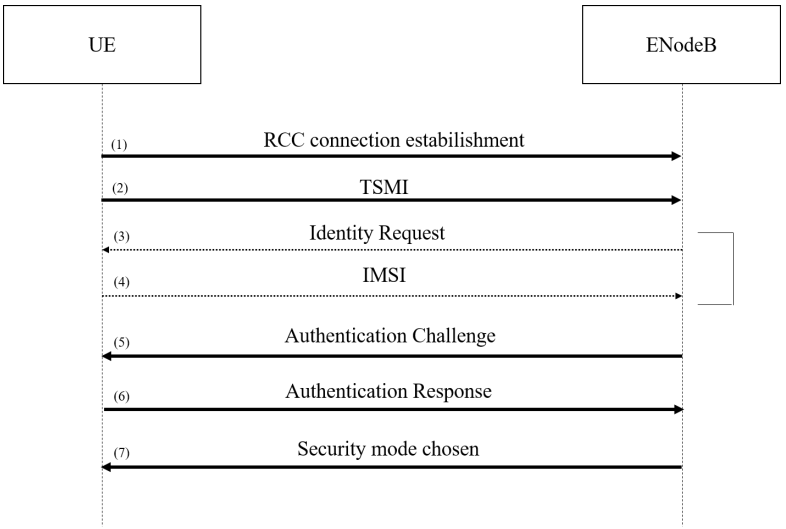


FIGURE 2.3: Message exchanged between UE and ENodeB on connection establishment [Meyer and Wetzel2004].

The user equipment needs to authenticate first to the network and only after that the network, in this case, a fake base station, needs to do is authentication challenge. So what a fake base station could do is to ask the TMSI of the user and later send an identity request message to which to user response with is true identifier the IMSI. After this message exchange, the base station rejects the connection with the device for it to connect to is true network and not affecting is normal usage. It is called a semi-passive approach because there is messages exchanged between a fake base station and a device but there no denial of any service and it does not affect in any mean the normal usage of the user equipment.

Besides the passive and semi-passive attacks presented there is also some active attacks possible, being the most used and important the man in the middle attack and the downgrade attack. The man in the middle type of attack is not possible in all UMTS networks but as the mobile network is not fully UMTS and needs to have compatibility with GSM an attacker could take advantage of this to implement a man in the middle attack. In this case, the attacker first impersonates himself as an user equipment to gather the challenge request that the network asks and later impersonates as a fake base station using the challenge obtained to impersonate as a true network to other user equipment's. Also when impersonating as a fake base station it mandates the use of less secure protocols of communication to the user equipment for even more control of the connection and the information exchanged. Despite the feasibility of this method, there is an even easier solution to an attacker, downgrading the user equipment from a UMTS network

to a GSM one and, using all the attacks available at this less secure generation. The downgrade attack is possible by sending only one message with a location update reject message to the user equipment's which will make it switch all is UMTS communications to GSM as illustrated in figure 2.4. This flaw in the UMTS protocol relies on the needed of the base station to being able to downgrade any user equipment in range.

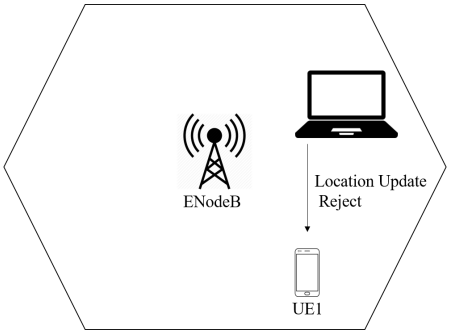


FIGURE 2.4: Downgrade attack in UMTS.

2.2.1.3 4G/LTE

The LTE protocol was developed for fulfill the demand for higher connection speeds for a wider number of services. Besides guaranteeing those speeds it also represented a change in the mobile network architecture as in LTE there is a IP based architecture. Despite this change of architecture it maintains full compatibility with previous generations like UMTS and GSM.

The compatibility to other generation exposes LTE to the flaws in those protocols by using a downgrade attack similar to the one presented in figure 2.4 but with a TAU reject message rejecting the user equipment to access to the LTE network. Using this denial of service approach the user equipment becomes vulnerable to UMTS and even to GSM attacks. Besides the downgrade attack in LTE networks there is also passive , semi-passive and other active attacks possible. In [Shaik et al.2017] all the possible attacks on LTE network are summarized and aggregated.

In passive attacks the attacking device takes advantage of the paging process that occurs in LTE[ørseth2017], [Zhou2018]. When the network receives an attempt of call, a message, or any notification (instant messaging, a game notification ...) it tries to locate the user equipment in its network for being able to deliver the notification. That attempt of location is done by sending a paging message, usually in the last base station where the user equipment was connected, and waiting for is response. This message is

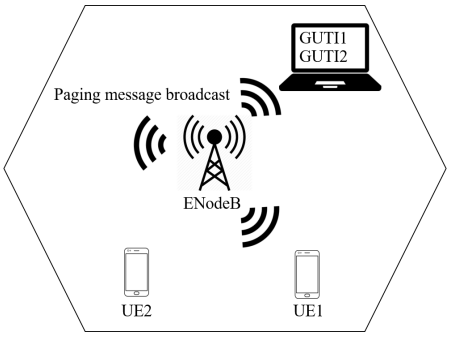


FIGURE 2.5: Location leak attack in LTE.

unencrypted and present the temporary identifier of the user, is GUTI (Globally Unique Temporary Identifier). This process is usually done using the smart paging technique where several paging requests are grouped in one single message for better efficiency. For a passive device listennig in the right frequency that means that there is the possibility to listen to a paging messages and know how many devices are nearby. The only problem could be the match between one single user and is temporary identifier GUTI as it could change in the process. Besides that in [Shaik et al.2017] it is proven that the GUTI refreshment is very low having user equipment's the same GUTI for over 3 days in row allowing for a more precise count of the number of devices nearby using this attack. The presented attack could change to a semi-passive one if there is the need to locate one single user equipment in is range. By doing so we could trigger a paging request using any social media, instant message or by sending an email to the victim. Using these paging request with the timing of the trigger paging done we could more precisely locate one single device in the network.

Other active attacks besides the downgrade attack previously mentioned could be a denial of all services attack making the user equipment unable to connect to the mobile network until reboot. This is achieved again by send TAU reject messages but instead of a downgrade using a total denial of access to the network. Also by changing the content of the TAU reject message it is possible to make the denial of specific services in the network. The last active attack possible in LTE network is a continuation of the location leak attack but this time by impersonating a base station ask for the details of the exact location of the user equipment by message exchange.

Another approach for passive attacks in LTE networks is to listen to the downlink signals transmitted from the base stations to the user equipment's. For that analysis, it is important to note the work done by [Introduction2018] and [Foddis et al.2014] where

it the first work is done an analysis of all the unencrypted signal transmitted from the main cell towers to all UE that could be used to track and identify a user in the network. In the second work presented it is analysed call behaviour and pattern detection for UE in the range of a given cell tower

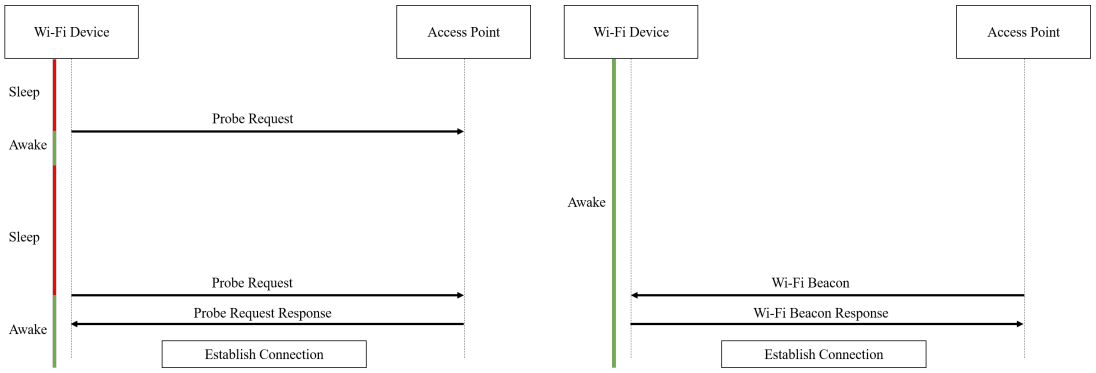
2.2.2 Wi-Fi

Wi-Fi is a commonly used protocol of communication for internet access wirelessly. It is maintained by the IEEE 802.11 standards and uses the 2.4 or 5 GHz frequency bands. The main aim of Wi-Fi is to make it accessible and easy to connect any device to the internet and maintain that connection.

In Wi-Fi, a device could be in two different states, associated and non-associated states. In an associated state, the device is connected to a Wi-Fi network and an access point and as access to an internet connection. When connected to the access point it exchanges messages wirelessly identifying the device that send them along with the content of the message. This is needed for the access point to filter and know each device in the network as send which message and forward them to the internet. While the content of the messages is fully encrypted the part of the header that identifies the user is not and along with another information could be used to profile and distinguish its users. In [Qin et al.2013] the users are profiled and classified with their usage activity analyzing all the messages exchanged and processing them. In this approach, the messages where analyzed using a big data approach, in the sniffing device data was filtered, compressed and stored in a local database. Later the data was upload to the cloud for using data mining and is analysis on the information gathered. Also in [Kalogianni et al.2015] a study was done on a university campus that allowed to group several users by their similar time and arrive and departure times. Other works like [Cheng et al.2012] , [Cunche et al.2012] and later [Cunche et al.2014] tried to get personal relationship and social interactions using different properties from the probes that their devices transmit.

However, in an associated state, the leaked information is easier to process and gather devices roaming over the city usually are not in the associated state as they are not connected to a known network so our solution is more focused on devices that are not in an associated state.

In a non-associated state, devices are not connected to a network but they send probe requests in time intervals advertising themselves to the world in the hope of a known network is in range and tries to initiate a connection with the device. This feature is used for a quicker recovery if accidentally disconnected with the network happens and also for a quicker connection when arriving at a known network range. Another way that a device could connect to a network is if it listens to the beacons regularly send by the access points announcing that a network is available, In this way devices needed to be fully active for listening for these beacons wasting a lot of energy in the process. So, for better efficiency, they usually use probe request as they only need to be awake long enough for a response to be sent.



(a) Using probe Request (b) Using Wi-Fi Beacons

FIGURE 2.6: Wi-Fi start connection methods.

An analysis of how talkative are devices sending probe requests could be found in [Freudiger2015]. When these devices do advertising they need to send an identification for the response of the network being directed to them. In this advertising if devices send their true MAC address them a device could be tracked all around the city and, for solving this privacy security flaw, MAC randomization is used in some devices (50% of the devices in the market as the study [Martin et al.2017] in 2017 but expected to rise) to protect the devices from being tracked. MAC address randomization consists of sending a random address in a probe request for protecting the user identity. As the aim of the probe request is to acknowledge the nearby networks there is no need to fully identify the user's device. The implementation of this technique is fully dependent on the manufacturer and the operating system used and if not implemented correctly could lead to identifying the user's real MAC address or by analyzing the other properties of

the probe request identifying a user uniquely once again. A study on this randomization was the study of [Martin et al.2017], [Vanhoef et al.2016], [Di Luzio et al.2016] or [Matte et al.2016] where on those studies it is gathered several random MAC address trying to decrypt them for identifying a device. In these studies, patterns are detected in randomization by manufacturer or operating system making it possible to identify a device by its pattern of emission of probe request and the changes on its MAC addresses. Other interesting work where it is shown the conversion between encrypted WPS MAC address to the real user MAC [Robyns et al.2017]. This study can convert WPS encrypted devices to the real MAC address of a device. Besides being able to do this conversion it was not tested in a mac randomization envirmment

It is also important to refer that a full network of detecting devices that combine the knowledge from MAC address anonymization and its flaws and outputs the number of devices in a given location have been implemented in literature before. Different sensors were used in [Redondi and Cesana2018] to detect Wi-Fi presence of devices in a lab and study their characteristics and located them in an indoor environment. Our solution will consist of something similar to a network of sensors but only outputting the final count of how many devices it as detected. Also, our solution will not analyze the content of the packets exchange, which is usually encrypted, as the focus of the device is a count of the devices in range and not the content of their messages.

2.2.3 Bluetooth

Bluetooth is a short-range data transmitting protocol used for wireless data transfers. In smartphones, Bluetooth is used for data transfers between wearables, other users or output/input devices. Due to its low range (just a few meters) when a device is detected in Bluetooth it is known that the device is very close to the sensor giving a better location accuracy.

Bluetooth protocol is defined by the IEE 802.15.1 standards and uses the 2.4 GHz frequency spectrum. Bluetooth as three main states: paired, discoverable mode and undiscovered mode. While in discoverable mode every device is easily identified in undiscoverable mode devices regulate the amount of information they send in their probe requests. In paired mode, a device transmits data to another device using Bluetooth using encrypted communications. Devices using Bluetooth are normally in an undiscoverable mode state for better protection of privacy and to maintain the ability to connect to

paired devices rapidly.
Besides Bluetooth classic protocol, there is Bluetooth low energy (BLE) and is generally used to communicate with wearable devices as this protocol reduces the amount of energy spent in communications saving battery life with is one of the main problems of wearable devices.

The most important analysis is them of devices in Bluetooth undiscoverable state as they are the more common state of devices and also is harder to track. In undiscoverable state devices have similar behavior of the devices in Wi-Fi nonassociated state, they send probe requests in the hope to connect to a previously paired device. This probe requests transmit the Bluetooth MAC address of a device but usually, these MAC addresses are also randomized so all the work done in randomization of MAC addresses in Wi-Fi is also relevant for Bluetooth. Also for all the discovering of Bluetooth devices signal analysis and understanding is important and for that is import to refer the work of [Chernyshev2015] where it is analyzed all the noise that could be generated by a Bluetooth device and all the inquiry scans that could be used. This study is very extensive covering diverse manufacturers and devices. In a similar way as on Wi-Fi, a network of sniffing Bluetooth devices was built in [Mueller et al.2015] where several sensors where spread alongside an area in a city for detecting movement patterns and behaviors. Its results presented a very good correlation between the daily load curves of the devices and reality.

Besides a purely Bluetooth network or Wi-Fi network literature as presented hybrid solutions where Wi-Fi and Bluetooth probes where gathered simultaneously for detecting patterns and devices counts. For being able to correlate the messages from Wi-Fi and Bluetooth a study used the RSSI from both messages and the losses of transmitting a message in the different protocols for locating and cluster messages from the same device [Longo et al.2018]. Also besides the RSSI, the similarity between the two MAC addresses could be analyzed as usually MAC addresses are assigned in the manufacturing process and are assigned at the same time. This generally makes the MAC addresses of Wi-Fi and Bluetooth to be consecutive. In [Schauer et al.2014] a hybrid network was built for detecting devices and flow patterns at a security check and compared the results to the boarding pass data achieving a good relation besides having some false positive problems at times.

In figure 2.7 we summarize all the approaches present in literature and present a qualitative comparison of those approaches in terms of range, precision and timeliness of detection. As for the latter, we consider a rough ordinal scale: “near real-time processing" (wireless spectrum analysis and sound capturing), “delayed results" (image capturing and mobile operators cell tower trace data, both requiring intensive computation, not amenable to produce immediate results) and “post-facto analysis" (e.g. in social networks based data analysis where, due to data capture delays and post-processing of big data, results may take long hours or even several days to be produced).

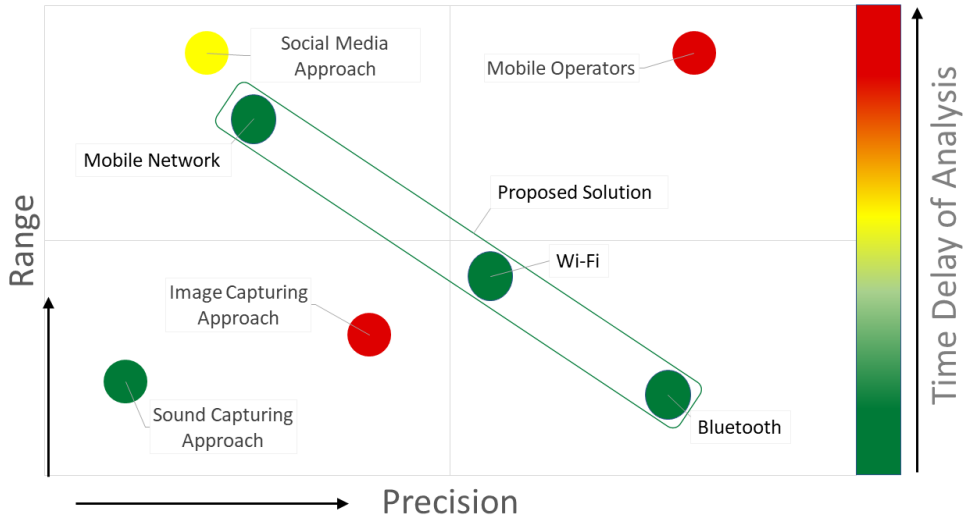


FIGURE 2.7: Quadrant analysis of the different approaches.

Chapter 3

Architecture of Solution

The development of our solution and the STC project was driven by the tourism overcrowding problem faced in the historical neighborhoods of Lisbon. The nature of these areas dictates the options that are most appropriate for the architecture.
This chapter is organized as follows, section 3.1 it is shown the overall project structure and microservices architecture, on section 3.2 we present the architecture proposed for our solution in section 3.3 we present a more detail view of every detection node components and their usage. In section 3.4 we present the software architecture of solution.

3.1 Sustainable Tourism Crowding project

The STC project was design using a microservices architecture where different components work independently and communicate through defined interfaces. The overall microservices architecture of the project can be seen in figure 3.1. In this architecture the solution developed in this dissertation is represented by the microservice SNIFF that its main goal is to detect how crowded is a given area in the vicinity of the edge computing sensor. The microservice LEARN aims to store all the data from all the sensors and using this data forecast future patterns of occupancy of the different areas. Having the data from past and forecast occupancy the microservice HEAT will compose a heat map of the geo-located data for visualization of the patterns of crowded areas in the city. ROUTE is a core microservice of the project where the routes suggested for avoiding crowded areas and simultaneous have sustainable points of interest are created.

For creating these routes it consumes the data from the HEAT microservice that gives the actual and forecast crowded data and also, the points of interest in the area from the POINT microservice. POINT is a web-platform interface that allows for user input of geo-located points of interest and that stores this data. Finally, after computing the ideal route the route is feed to the APP interface that will show the route chosen through a mobile application for tourists.

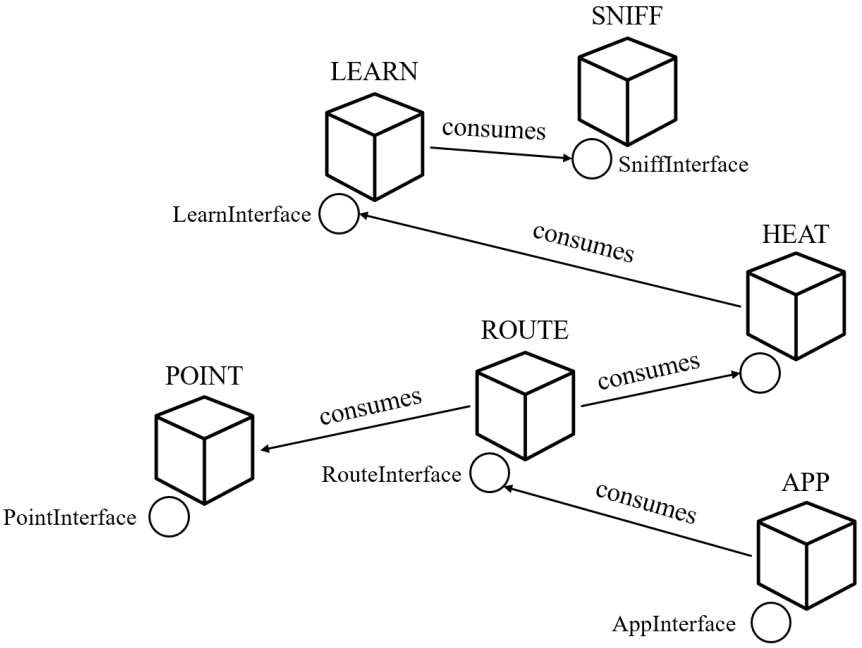


FIGURE 3.1: STC project microservices architecture

3.2 Proposed solution architecture

Our solution architecture needs to comply with several different requisites that are clustered in three different main areas being them scalability, robustness and device discreetness.

Related with scalability we have the following requisites:

- Low node cost.
- Low cost of communications (upload and download data).

These requisites are necessary to guarantee the full scalability of the solution by keeping the overall cost per node low for quicker and cheaper deployment.

Regarding robustness our solution needs to be able to endure the different deployment environments and its challenges and so the requisites are:

- Ability for 24/7 operation.
- Ability for recovering from power failures.
- Ability to endure harsh environments (e.g. rain and high temperatures).

Finally our sensor needs to be able to be discret and do not highligh from the environment either indoor or outdoor. For this the requisites found were:

- Size of sensor.
- Simple and discreet design.
- Low noise of operation.

Considering the previously mentioned requirements, we propose the architecture present in figure 3.2 for our solution, where light edge computing nodes are responsible for the detection of smartphones in the area and sending that information via LoRaWAN to several gateways that process the data received and forward it to our cloud server.

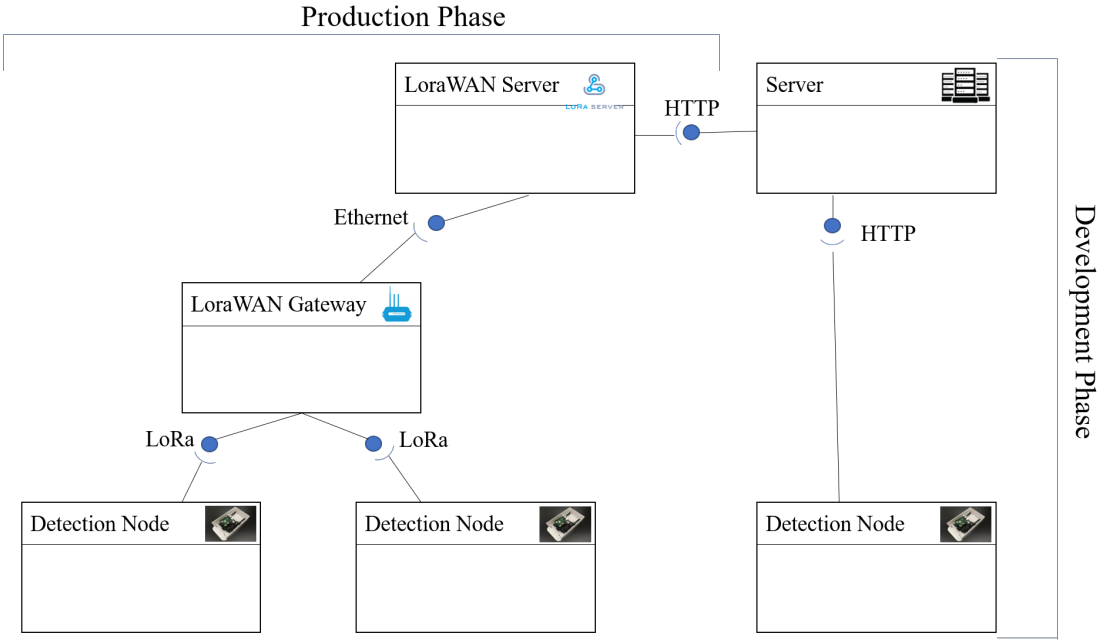


FIGURE 3.2: Component diagram of overall architecture.

We have chosen this type of architecture because it is the one that fits the best with the requirements given. In regards to requirements for low cost of nodes and communications we have chosen to deploy a network of several nodes equipped with a LoRa board and corresponding antenna, that will do all the edge computing and communicate their results via LoRa to a LoRaWAN gateway that, in its turn, will route the detection information to a cloud server, via a LoRaWAN server, for further processing and visualization of the crowding metrics generated in each node.

The implementation using LoRa is the one we aim to follow in the production phase but for the deployment of our nodes in the prototype and testing phase performed in the scope of this dissertation we wanted to keep the costs of communications even lower, so a Wi-Fi connection to the internet was used to upload data to our servers. Using this approach, our detection nodes communicates directly to our servers via HTTP requests. The description that follows describes the rest of the architecture for the production phase.

Since we are applying an edge computing approach all the hard work of processing the information gathered, filter it and generate the metric is going to be done in our detection nodes. As so the information to be passed on to the cloud server is small and the required sampling rate is low (consecutive messages will typically be a few minutes apart from each other), we will use a LoRaWAN network, which has a free spectrum use that reduces dramatically the cost of communications.

A LoRaWan network aim is to support the IoT devices that usually need only small amounts of data in widely spread uploads. Also, the limitation of data exchanges provides extra security in user privacy and the security of our nodes. Regarding user privacy, the data gathered is processed locally and only one metric of the number of devices detected is ever sent to our server. Our edge computing node, by anonymizing the data as it is gathered will guaranty user privacy. Also for our detection node, the small amount of exchanged data protects the nodes from outside attacks, as the communication line hampers almost all type of commonly used attacks.

Regarding the requisites of robustness and the need for a constant power connection we suggest the deployment of our detection nodes in electricity poles for urban lightning as illustrated in figure 3.3. By using an existing structure and making our solution small in size we aim to not affect the overall look of the historic neighborhoods where the device will be installed.

FIGURE 3.3: Illustration of the possible implementation of our solution in the field.

3.3 Components

Each crowd detection node in the proposed architecture, shown in figure 3.4, contains a processing unit and a set of sensors connected to it such as a Wi-Fi dongle, a Bluetooth dongle, a GSM dongle and/or an LTE dongle.

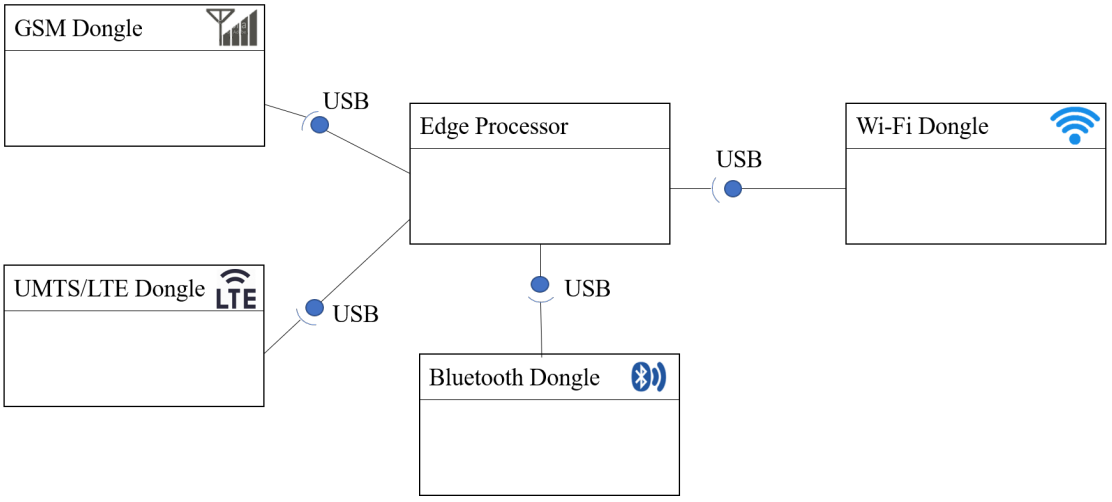


FIGURE 3.4: Component diagram of detection node.

The edge processor is responsible for providing computing resources required for gathering and processing data in real-time from multiple technologies simultaneously. Also by connecting the edge processor to several USB dongles, we maintain the flexibility to use what technologies seem necessary in any location that our detection node could be placed.

In the detection node, each USB dongles is responsible for capturing data in their respective technology, while the processing unit analyses and integrates locally all the captures. In the scope of this dissertation for each technology we generate one metric that represents the density of devices using a given technology. This metric is due to the different ranges of each technology so by using the maximum area of coverage of each technology we can infer the the number of devices per square meter.

3.4 Software architecture

In this section, we present the overall software architecture of our solution. Figure 3.5 shows an illustration of the different programs used and its interactions with our local database and each other. This illustration aims to show in a abstract way the overall operation of our edge computing detection node

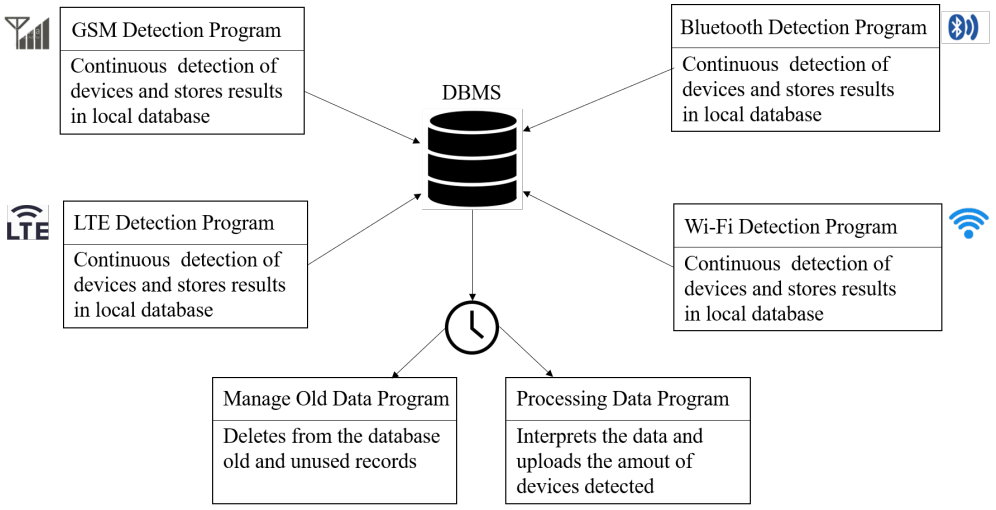


FIGURE 3.5: General software architecture.

On our implementation data is gathered by the different programs each one for its correspondent technology. All of the programs used are adaptations off open source software available for user customization for their applications.

All of those detecting programs write in the same DataBase Management System (DBMS) locally for further analysis. This database guarantees the standardization of data and easy and quick access to information. All data in the Database is filtered and compressed by several programs. These programs were fully written for the only purpose of this solution and are responsible for cleaning old and unused data and for processing all the information, generating each metric by technology and uploading them.

All programs choice, usage, features and description will be further analyzed in the implementation chapter.

Chapter 4

Implementation

This chapter discusses all the choices for the implementation of our solution since the choice of the operating system in section 4.1 to the choice of the local database to store the results of the detections in section 4.2. In sections 4.3, 4.4, and 4.5 is presented all the hardware and software used for detecting devices in the different technologies. Section 4.6 presents the programs created to manage and upload of all the data, Sections 4.7 and 4.8 present the two different upload methods used. Finally section 4.9 presents the prototype that was built for the testing and validation of our solution. All the code developed can be accessed in the our Git repository[1]

4.1 Operating system

The first step for the implementation of our solution was to decide the development environment for all the programs that our solution will use to detect the number of devices in a given area. The operating system is an important part of our solution as it will support all our applications.

To keep the costs of the solution low the choice for an open-source operating system was the option and so, a Linux based operating system was chosen. Linux allows for better flexibility in the technologies and framework used and allows also for better control of the behavior of the detecting device. Another advantage of Linux based systems is the availability of vast amounts of libraries and open source programs developed by

[1] https://github.com/rdsas-iscteiul/crowding$_sensor$

its community that speed up the development process. The next decision regards what distribution of Linux to use in the development environment and later in a production environment.

Linux distributions are developed by different teams and focus in different areas and possible applications like a server or a home theater pc. This variation in Linux distribution is possible due to the nature of Linux. Being a fully open-source operating system allows other users to compile its kernel and add their software as they wish. For our development environment, we have chosen Kali Linux, a distribution focused on cyber-security and penetration testing attacks that has a lot of pre-installed tools and offers the required frameworks for listening to the protocol trace elements in the different technologies we are concerned about (mobile network, Wi-Fi and Bluetooth).

In a production environment, when our solution is fully tested and ready to be used in the field, a more stable and secure distribution will be needed. So, in a production environment, we aim to deploy our nodes with a long term support Ubuntu[2] distribution. This distribution guarantees support and updates for five years since is release. The stability and security of this distribution will guarantee a better production environment for our solution

4.2 Local database

For the local database present in our detection device that will merge all the collected data, we have chosen an SQLite[3] relation database. This database requires a lightweight and low memory usage, while fulfilling all the requirements needed for our device.

The local database schema is shown in figure 4.1 and consists of four different tables. The main and most important table is called *Cellphone_Records*. This table as the information of all devices detected in every program at a given time. Every program detects a device and inserts or updates a line in the table for the device detected.

Cellphone_Records as 5 attributes: (1) ID- The unique identifier of a device in a given technology, it could be a MAC address , an IMSI, TMSI or GUTI ; Technology- Representing on each technology was the device detected; First_time_seen- A timestamp of when as the device first detected; Last_time_seen- A timestamp of when as the device

[2]https://www.ubuntu.com/Releases
[3]https://www.sqlite.org

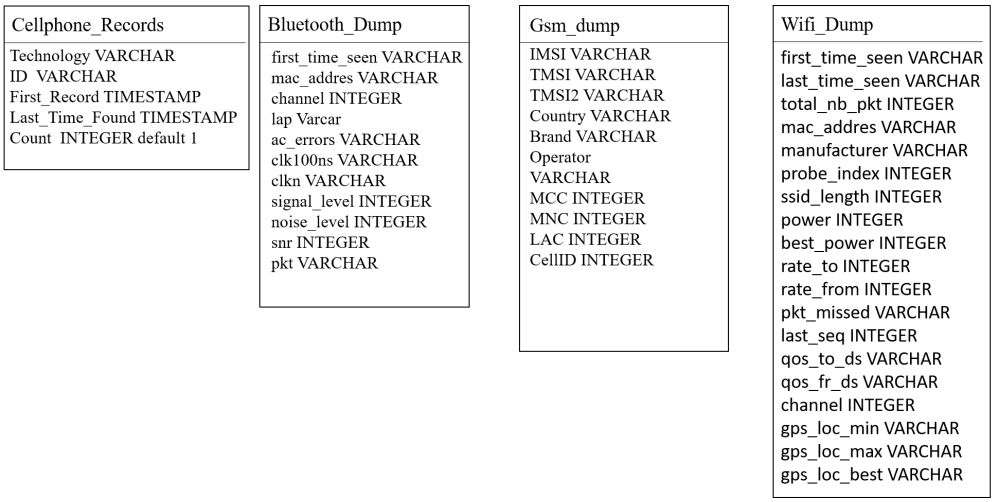


FIGURE 4.1: Architecture of local database present in detection device.

last detected; Count- Represents the number of times that a device was detected between the first and last appearance.

The attribute ID in the final phase of development will be the result of applying a one-way hash function like SHA-512 to anonymize the data. Using this function we aim to protect user privacy as all devices detected could not be traced to its original device id. It is also important to refer to some restrictions when inserting and updating data. When detecting a device for the first time, we assume that, for counting measurements, the device is present in the area for the next five minutes. So, when inserting a new row the first and last timestamps are separated by 5 minutes. Also, when updating a row, if the time elapsed between the last detection is more than fifteen minutes then a new line is created. If the time difference is less than 15 minutesm then we update the attribute count to add more detections of that device. We created this restriction for devices detected twice but very time spaced not be counted as only one record but two different records as the device have left the area and them come back. In this manner, we have a better and precise notion of when a device as arrived and left the area.

The 15 minutes interval could be changed, depending on the environment of the area around the device. If the area is a plaza where people tend to stay a long time them the interval could be higher, while if the device is located in a narrow passage the time interval could be lower as persons then to stay in the area less time but no less than 5 minutes as it is the average time between the emission of probe requests [Redondi and Cesana2018].

The remaining tables were used in the development stage and will not be used in the production stage. They were used to study the behavior of devices in the area, how frequently

they are detected and what information do they leak and could be used. *Wifi_ dump* table was used to store all possible fields on the header of every probe request for studying how anonymization is done and to explore its weakness. *Bluetooth_ Dump* had a similar function but with different header fields. Finally, *Gsm_ dump* stored all information that could be gathered from listening to the messages exchange in a GSM connection situation between the UE and the base station. LTE/UMTS detection of devices was done differently by reading captures and storing the result so results where stored directly on the main table of the local database.

4.3 Mobile network

The evolution and increasing affordability of SDR, open-source hardware and compatible open-source software has allowed for a more flexible spectrum and protocol analysis and have sped up research. This allowed for the emergence of several open-source programs that aim to clone, attack or sniff mobile network communications using SDR.

4.3.1 Hardware selection

In table 4.1 there is a summary of open hardware SDR boards available and that could be used in our device.

TABLE 4.1: Boards available for SDR usage in our solution.

Board	Cost
USRP X [4]	Very high
USRP N [5]	Very high
USRP B [6]	High
USRP E [7]	Very high
RTL-SDR [8]	Very low
HackRF [9]	Medium
BladeRF [10]	Medium
Lime-SDR [11]	Medium

[4] https://www.ettus.com/product-categories/usrp-x-series/
[5] https://www.ettus.com/product-categories/usrp-networked-series/
[6] https://www.ettus.com/product-categories/usrp-bus-series/
[7] https://www.ettus.com/product-categories/usrp-embedded-series/
[8] https://www.nooelec.com/store/sdr/sdr-receivers/nesdr-smartee.html
[9] https://greatscottgadgets.com/hackrf/one/
[10] https://www.nuand.com/product-category/bladerf-2-0-micro/
[11] https://limemicro.com/products/boards/limesdr/

USRP boards are produced by Ettus Research that produces open hardware boards for SRD applications. The hardware produced is very reliable and well constructed but it is also very expensive with prices ranging from 1 to 7 thousand euros. The aim of our solution was to develop a cheap solution for detecting devices so USRP boards could not be used.

On the other end of the spectrum, we have the RTL-SDR board costing around 20 euros. This SDR board was adapted from a dongle usually used from watching TV. The development of a new driver for the board made it possible to read raw I/Q data and, when connected to a computer, it can be used as a spectrum analyzer. This low-cost option, with capabilities to make spectrum analyses was the chosen board for our solution.

RTL-SDR is a good board for a low demanding tasks, but for the new generations of the mobile network another more powerful board was needed. The next three boards on the table are all at the same price range but have different capabilities. HackRF was the first board developed from the three and besides having a lot of support by the community its hardware is getting out of date. So, for our solution, the decision was between the BladeRf board and Lime-SDR. Comparing the two board it is clear that the Lime-SDR is the better option since it as twivce the amount of antennas of the BladeRF (4 instead of 2) and wider bandwidth. Also, it is a newly developed board with better hardware and besides having smaller community, support for the board is increasing.

Having all the hardware chosen in the next subsection it is presented all the software used for detecting devices in every technology.

4.3.2 Software selection

In figure 4.2 we present the component diagram and the interactions and interfaces used for gathering data in mobile network. The following subsections will focus in the specific tools used in each technology.

4.3.2.1 GSM

For using an SDR application software it is important to install first GNU-Radio[12] an open-source software that deals with intricacies of SDR hardware and performs almost

[12]https://www.gnuradio.org/ and installation details are provided in appendix A

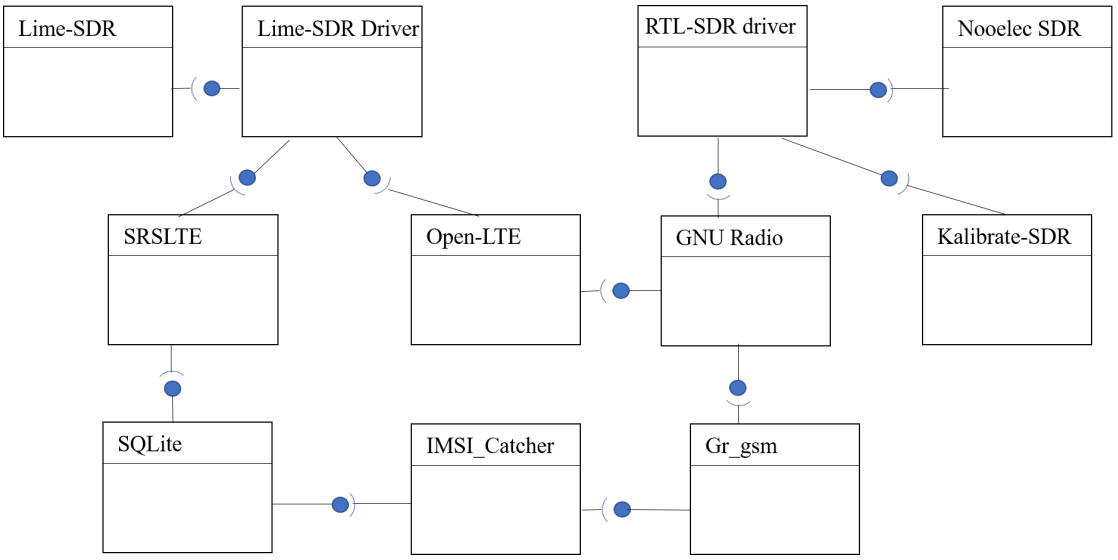


FIGURE 4.2: Mobile network component diagram.

all wireless signal processing functions. Also, it supports various types of RF hardware being fully compatible with the RTL-SDR board as well as Lime-SDR. The next step is to install the RTL-SDR driver's (see appendix B) and after that, we are prepared to explore the functionalities of our board.

On GSM detection, our solution first performs a scan of all nearby mobile network towers in the area. This scan can be done either using Kalibrate-rtl[13] a C++ application or grgsm_scanner[14] an application developed purely in C. After having the list of all the nearby towers our program selects the towers with the highest signal power and round-robin each one of the towers for sent messages. For listening to the messages exchange we use grgsm_livemon in the specific frequency to listen. This application only listens to traffic using GNU-Radio and do not read and decipher the messages. For reading and decipher the messages exchange it is used an IMSI_Catcher[15]. The IMSI_Catcher that our solution uses is simple_IMSI-catcher.py, an open-source free application developed in Python that deciphers all the bits gathered by grgsm_livemon and prints the results on the console. Our solution adapts this program for saving the results in our local database for further use.

The summarized process and programs used could be seen in figure 4.3.

[13]https://github.com/steve-m/kalibrate-rtl and installation details are provided in appendix C
[14]https://github.com/ptrkrysik/gr-gsm and installation details are provided in appendix D
[15]https://github.com/Oros42/IMSI-catcher and installation details are provided in appendix E

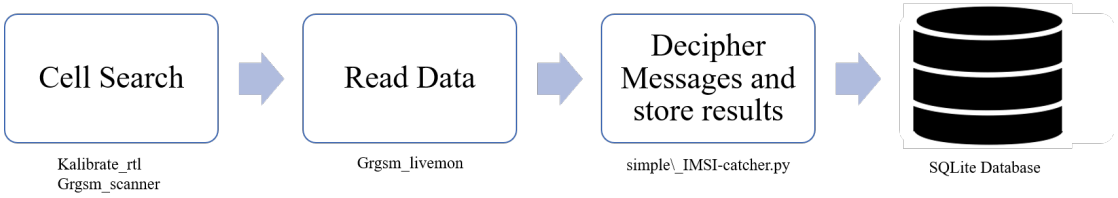


FIGURE 4.3: GSM detection process and programs used.

4.3.2.2 LTE

On LTE data is gathered by exploring the location leak vulnerability. In paging requests, cell towers transmit the users GUTI in an unencrypted manner and vulnerable to be intercepted. To simulate the behavior of a UE and being able to read those messages there could be used two different programs, openLTE[16] or srsLTE[17]. Our solution uses srsLTE instead of openLTE because of the simpler and more straightforward usage. The program srsLTE comes with some pre-built configurations that could be used and adapted to our solution. The cell_search program is used to know the cell towers that are the closest to the sensor and their power output and choosing the preferred cell tower for our trace listening. After this step, our solution uses the program ./pdsch for emulating the behavior of a UE and listening to paging requests. All paging request is written directly to a file as the output of the program execution.
All the messages stored by ./pdsch are in ASNI (Abstract Syntax Notation One) and need further processing to be interpreted. The next step is to use the Decoce_ASNI.py program that will decipher all the ASNI messages and store them in our local database.

The summarized process and programs used can be seen in figure 4.4.

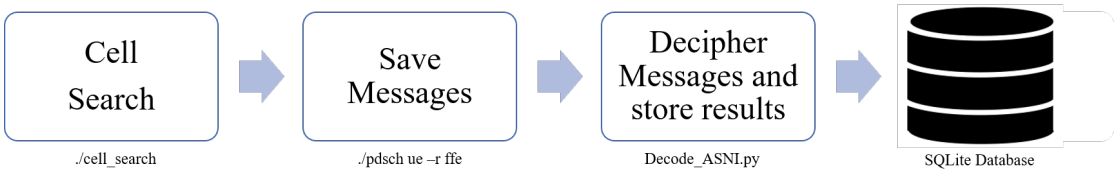


FIGURE 4.4: LTE detection process and programs used.

[16] https://sourceforge.net/p/openlte/wiki/Home/ and installation details are provided in appendix F
[17] https://github.com/srsLTE/srsLTE and installation details are provided in appendix G

4.4 Wi-Fi

4.4.1 Hardware Selection

On Wi-Fi, in terms of hardware, for detecting devices it is only needed a Wi-Fi card that supports monitor mode. Monitor mode allows for the board to listen to all the information that is exchanged wirelessly nearby. The only requisites that we had for choosing our wireless card board were a good range of detection and the ability to listen to data in both 2.4GHz and 5GHz bands. The chosen board for our solution was the Alfa Network awu036ac [18] a board that combines a low cost with high performance, having two antennas for dual-band detection and also protection against interference from Bluetooth devices something that is very important for our detecting device.

4.4.2 Software Detection

Having decided on which hardware to use its now time for software. Figure 4.5 shows the interfaces and the different components used in our solution for gathering data on Wi-Fi.

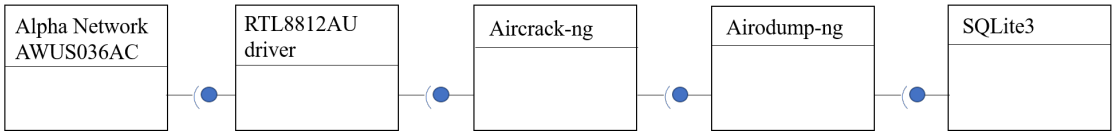


FIGURE 4.5: Wi-Fi component diagram.

In terms of open-source software, there are a lot of alternatives to use having between them different approaches and features available. Some off them use an approach off writing capture data to a file for further analyzed and other programs use real-time detection and data output. On table 4.2 all the relevant programs found where summarized and the program that was chosen for our solution highlighted.

[18] https://www.alfa.com.tw and driver installation details are provided in appendix H

TABLE 4.2: Programs available for detecting Wi-Fi devices.

Program	Open-source	Community
Wireshark[19]	No	Large
Kismet[20]	No	Medium
BoopSuite[21]	Yes	Small
Netsniff-ng[22]	Yes	Small
Aircrack-ng[23]	Yes	Large

Wireshark is a widely used program for capturing data and analyze packets and messages exchanged. It is used for analyzing protocol behavior and debugging situations. It can be used to detect probe requests from devices but for further analyzes, it needs to save the results into a file and another program would need to constantly grab the file and process it. This behavior is very slow as writing into a file much slower than processing information directly in memory. That is why, despite of the great community around Wireshark we needed another program to perform our detections.

Kismet is a more specialized program for detecting devices in Wi-Fi networks than Wireshark as is has several features for detecting Wi-Fi beacons and probe requests. The problem with Kismet is similar to the one of Wireshark, where data needs to be saved in a file for further processing. Again, that is why we have not chosen this program.

The remaining programs are all open source and could be adapted to write directly into our local database. Aircrack-ng is a well-established tool developed fully in C used for several kinds of attacks to Wi-Fi networks and for detecting devices. It has a very large community and support and could be customized by the user. Netsniff-ng is an aircrack-ng program but with optimizations in speed reading of devices. It has a smaller community and support and documentation is scarcer. Finally, BoopSuite is an Aircrack-ng alternative but fully developed in Python. It is still in development for having all the features offered for Aircrack-ng and is a solution for applications where Aircrack-ng is not compatible or usable. Aircrack-ng was chosen because of its large community and documentation and we had no problem with its installation, since it was the most straightforward solution for adapting to detect devices and store data in our local database.

[19] https://www.wireshark.org/
[20] https://www.kismetwireless.net/
[21] https://github.com/MisterBianco/BoopSuite
[22] http://netsniff-ng.org/
[23] https://www.aircrack-ng.org/ and instructions for installing Aircrack-ng can be found in appendix I

Aircrack-ng as several different applications and our solution uses two applications, in particular, airmon-ng and airodump.ng. It uses airmon-ng to enable monitor mode in our Wi-Fi board and later airodump-ng to perform the devices detection. For detecting only probe request messages airodump-ng was changed to only intercept these messages and store their data in our local database.

The summarized process and programs used could be seen in figure 4.6.

FIGURE 4.6: Wi-Fi detection process and programs used.

4.5 Bluetooth

For detecting Bluetooth devices it is necessary a dedicated board. The effeteness of a board detection is related to how many devices can it detect, its range and in what modes can it detect devices.

4.5.1 Hardware Selection

In table 4.3 is a summary of open hardware boards available that could be used in our solution for detecting trace elements of devices in Bluetooth.

TABLE 4.3: Boards available for detecting Bluetooth devices.

Board	Cost
SMK-Link nano [24]	Low
Sena UD-100[25]	Medium
Ubertooth-One[26]	High
Nordic Semiconductor nRF51-DK[27]	Medium

[24]https://www.smklink.com/pages/nano-dongle-bluetooth-v4-0-le-edr-users-guide
[25]http://www.senanetworks.com/ud100-g03.html?sc=14category=3968
[26]https://greatscottgadgets.com/ubertoothone/ and instructions for installing ubertooth-one can be found in appendix K
[27]https://pt.mouser.com/ProductDetail/949-NRF51-DK?R=nRF51-DKvirtualkey57440000virtualkey949-NRF51-DK

SMK-Link nano is an example of a simple small Bluetooth board that is capable of 4.0 Bluetooth communications. These kinds of boards are generally used to pairing Bluetooth devices with a personal computer. Because of that, it can only detect devices that are in discoverable mode and waiting to be paired. Since most devices we aim to detect will not be in this mode this board was not enough for our solution.
Sena UD-100 is a larger dongle that has a much larger range of detection. Its range could be expanded even more with the usage of compatible antennas. Nevertheless, it suffers from the same problems as the previous board, only detecting devices in discoverable mode making it not enough for our device.
The Nordic Semiconductor board and the Ubertooth-one both can discover devices in undiscoverable mode but the Nordic Semiconductor as a very limited range and low reliability. Furthermore the Ubertooth-one board as open-source software specially developed for it that tracks and detected every device in Bluetooth in the nearby area. For these reasons the board chosen for our solution was the Ubertooth-one.

4.5.2 Software Selection

In figure 4.7 we present the overall interfaces and components used for detecting devices in Bluetooth.

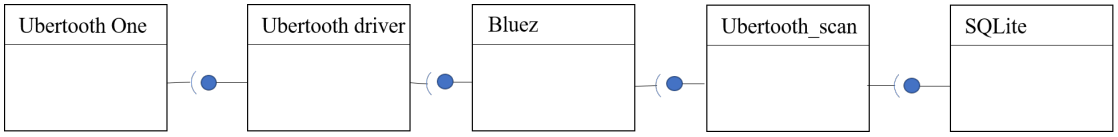


FIGURE 4.7: Bluetooth component diagram.

For using the Ubertooth-one board it is first necessary to install the bluez package, a package that contains tools and frameworks for Bluetooth usage in Linux. With the package installed it is possible to start a Bluetooth device using the hciconfig command. For detecting Bluetooth devices we adapted the open-source program from Ubertooth. Our solution uses the ubertooth_scan program to detect devices in undiscoverable mode and write directly to our local database every time it founds a device.

The summarized process and programs used could be seen in figure 4.8.

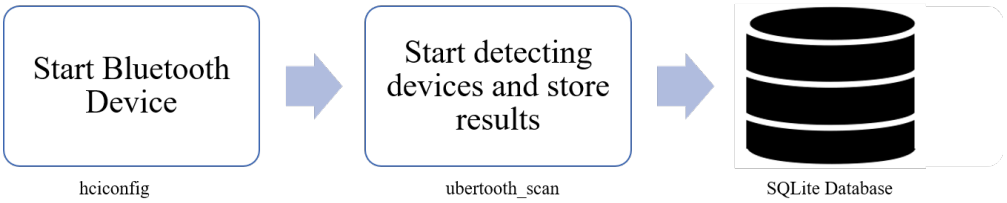


FIGURE 4.8: Bluetooth detection process and programs used.

4.6 Upload and data management programs

To manage all the different detecting programs to be run in parallel, our solution uses crontab[28] to schedule the start time and the duration of detection of every used program. Crontab allows for commands to be executed in predefined time schedules allowing for the start of our capturing programs when the edge processor is booted and upload the data stored in our local database to the server at a regularly timespaced interval.
Crontab schedules several different tasks simultaneously in our solution being them: (1) execute a python program at startup that will activate the monitor mode for our Wi-Fi card(airmon-ng) and start our Bluetooth device(hciconfig); (2) execute a python program to start the gsm detection; (3) execute a python program to start the Wi-Fi and Bluetooth detection program ; (4) execute a python program that uploads data to one of our servers depending on the phase of development; (5) execute a python program that will delete old and unused data from the local database (records where the last time saw, was over X hours ago).

4.7 Web-service and website

In the development phase, there was a need to demonstrate the feasibility of the detector and the importance of the data collected. This need of demonstrating the detector in its first steps of development was due to the need of signing a collaboration agreement with the council administration of one of the biggest parishes of Lisbon for the future deployment of our overcrowding detector. Also, several reunions and demonstrations were done to get support for the overall project. For all the demonstrations we created a temporary solution to show our data.

[28] https://linux.die.net/man/5/crontab

The behavior of our web-service implementation is represented in figure 4.9. We used a REST API to deal with all the communications to our server. A REST API uses HTTP GET, PUT DELETE and POST requests to send and receive data. Data is sent from our detection device using a POST HTTP request that is processed by our web-service, written in PHP, that stores the data received on a database built in PHP MyAdmin. We developed two different web-services that support the two different types of demonstrations done. One demonstration used data gathered by GSM devices while another shown data from Bluetooth, Wi-Fi and GSM devices.
Each website created from each demonstration used an HTTP GET request to gather the data needed and received a list of JSON objects containing the information to be shown.

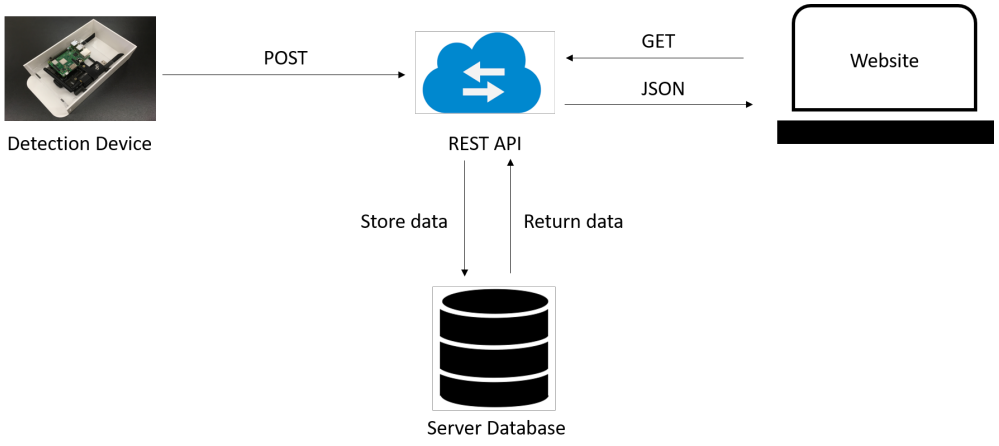


FIGURE 4.9: Architecture of web-service implementation.

To support the two different demonstrations we created a database in MySQL to stored the data to be shown. The Architecture of the database created in our server for this support could be seen in figure 4.10.

Our GSM demonstration aims to show the information leaked from GSM devices including their country of origin and the cellphone tower details to each it is connected. To support this demonstration we used the table Cellphone and Cell_Towers. Cell_Towers contains information of all Cellphone Towers located in Portugal and was populated using data from OpenCellID[29]. OpenCellID is an open database containing the details of cellphone towers around the world. The information contained in this table was used to get the longitude and latitude coordinates of the tower to which a device is connected. By Intercepting the data leaked from the device about the tower to which it is connected

[29]https://www.opencellid.org/

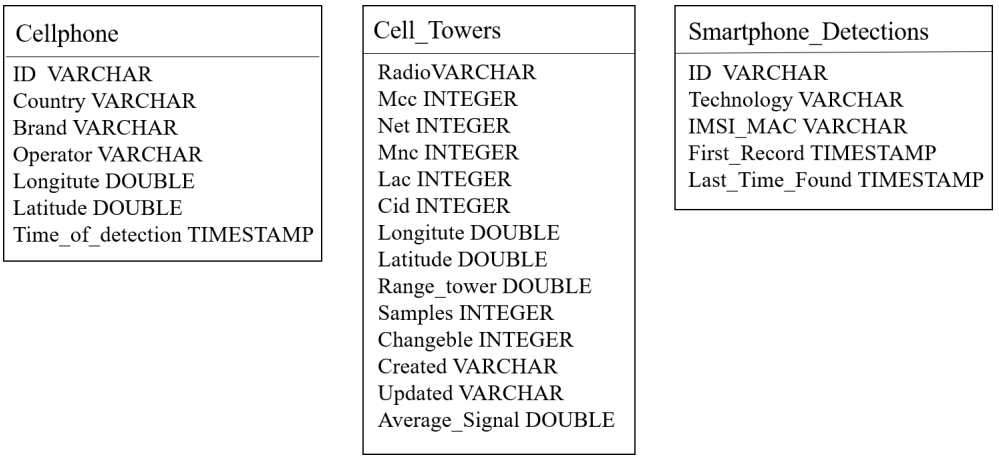


FIGURE 4.10: Architecture of database for website support.

we could get the coordinates and possible radius of where the device is. The table Cellphone contained the information of every device detected including their unique ID, their country of origin, their brand a manufactures as well as the time when the detection has occurred and the coordinates of the tower to which the device is connected.
The table Smartphone_Detections supported the other demonstration created. This table contains a unique ID for every record, the technology in which the device was detected as long with the MAC address or ISMI detected. It also stored the first and last time that the device was detected.

The first demonstration was realized in the historic downtown of Lisbon and aimed to show the different home countries of every device detected. The demonstration occurred in Junta de Freguesia de Sta. Maria Maior the parish county that encompasses the historic neighborhoods of Lisbon. For this demonstration, our detection device used the data gathered in real-time from GSM devices and for each device detected created a JSON object and sent it using an HTTP POST request to our web-service. The code that deals with this request could be seen in appendix L. For showing the GSM devices detected nearby by our solution we created a website that plot a bar chart representing the number of devices detected by country of origin. A snapshot of the website created could be seen in figure 4.11.

This website receives data in a JSON object containing by country of origin the number of devices detected. It obtains this JSON from an HTTP GET request which code could be seen in appendix M. This web-service could also be used to get the total records of

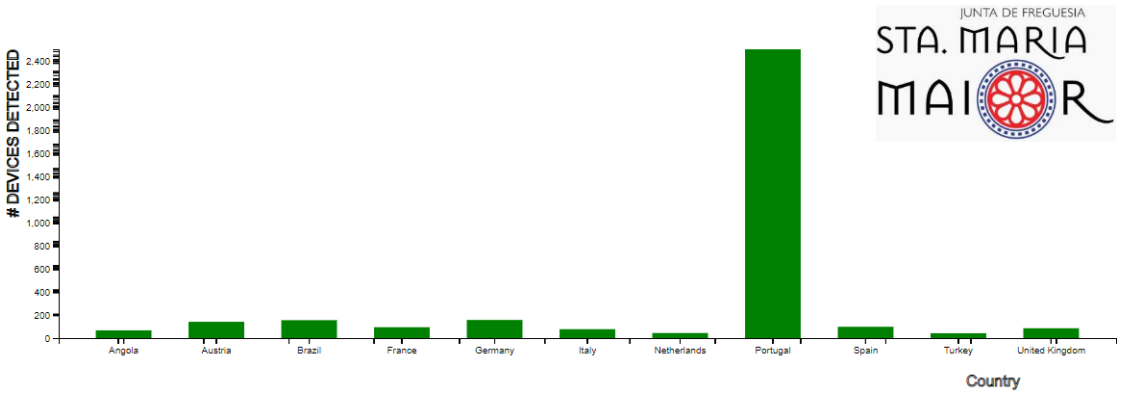


FIGURE 4.11: Website created for the demonstration of GSM detection.

cellphones detected as also one specific tower. It achieves this depending on the fields provided in the HTTP GET request.

The second demonstration made of the solution aimed to show simultaneous technologies detection in real-time. It was composed of two parts, the first was to show a pie graph(see figure 4.12) that regularly updates with the number of devices detected in the three technologies used in detection (Bluetooth, Wi-Fi and GSM). The second part of the demonstration was to show in a table-wise format all the data collected and ask the participants to find their smartphone ID in the data shown as seen in figure 4.13.

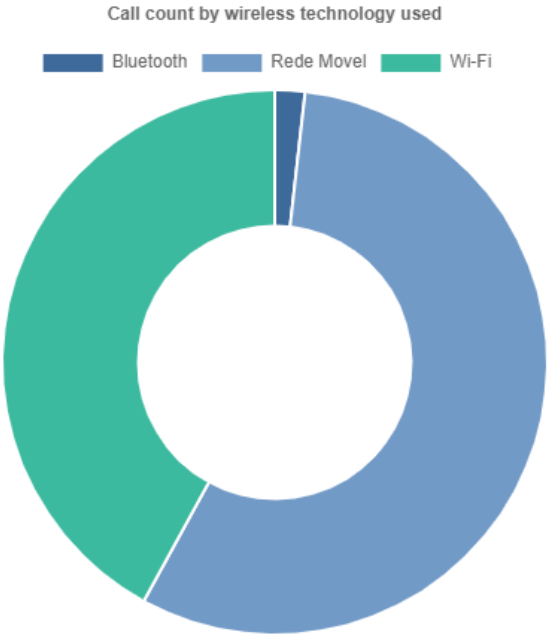


FIGURE 4.12: Snapshot of website created for the demonstration of real-time detection.

For sending the data that support these websites our solution uses an HTTP POST request every time it detects a device. Our web-service receives that request and then

imsi_mac	first_time_detected	last_time_detected
DA:A1:19:5C:78:A4	2018-12-11 16:22:11	2018-12-11 16:22:13
58:FB:84:6B:8B:B8	2018-12-11 16:42:06	2018-12-11 16:41:57
D6:FA:E1:B9:12:79	2018-12-11 16:22:16	2018-12-11 16:22:17
16:DA:04:A8:A3:5A	2018-12-11 16:22:18	2018-12-11 16:22:18
7C:01:91:5E:54:FC	2018-12-11 16:42:16	2018-12-11 16:42:15
DA:A1:19:97:A2:56	2018-12-11 16:22:21	2018-12-11 16:22:21
DA:A1:19:43:06:BA	2018-12-11 16:22:21	2018-12-11 16:22:21
DA:A1:19:C6:1B:CF	2018-12-11 16:22:21	2018-12-11 16:22:21
DA:A1:19:5B:9D:51	2018-12-11 16:22:22	2018-12-11 16:22:22
DA:A1:19:82:1F:D3	2018-12-11 16:22:22	2018-12-11 16:22:22
88:78:73:BB:15:02	2018-12-11 16:37:41	2018-12-11 16:37:36
E4:E0:A6:19:BA:A8	2018-12-11 16:41:45	2018-12-11 16:41:43
00:28:F8:FC:CE:A7	2018-12-11 16:37:41	2018-12-11 16:37:40
DA:A1:19:42:98:37	2018-12-11 16:22:26	2018-12-11 16:22:27
9C:B6:D0:60:B3:79	2018-12-11 16:40:30	2018-12-11 16:40:23
AC:ED:5C:BE:74:13	2018-12-11 16:37:41	2018-12-11 16:37:38
64:5D:86:84:5E:4F	2018-12-11 16:32:48	2018-12-11 16:32:43
58:00:E3:5F:A3:59	2018-12-11 16:42:16	2018-12-11 16:42:07
74:8D:08:71:91:01	2018-12-11 16:29:58	2018-12-11 16:29:58
DA:A1:19:AA:01:B5	2018-12-11 16:22:33	2018-12-11 16:22:34
DA:A1:19:75:DD:91	2018-12-11 16:22:34	2018-12-11 16:22:34

imsi_mac	first_time_detected	last_time_detected
??:??:??:F0:70:D9	2018-12-11 16:42:06	2018-12-11 16:41:57
??:??:??:84:5E:53	2018-12-11 16:41:03	2018-12-11 16:41:01
??:??:??:6F:A7:1D	2018-12-11 16:25:18	2018-12-11 16:25:08
??:??:??:9C:F3:74	2018-12-11 16:42:26	2018-12-11 16:42:25
??:??:??:06:AE:82	2018-12-11 16:24:38	2018-12-11 16:24:38
??:??:??:56:30:52	2018-12-11 16:25:07	2018-12-11 16:25:00
??:??:??:29:8E:21	2018-12-11 16:25:28	2018-12-11 16:25:23
??:??:??:3D:6C:6E	2018-12-11 16:40:09	2018-12-11 16:40:08
??:??:??:0C:A2:2A	2018-12-11 16:28:05	2018-12-11 16:28:05
??:??:??:CA:96:09	2018-12-11 16:30:23	2018-12-11 16:30:23
??:??:??:10:2D:8A	2018-12-11 16:31:54	2018-12-11 16:31:50
??:??:??:5E:54:FD	2018-12-11 16:42:06	2018-12-11 16:42:02
??:??:??:18:9F:17	2018-12-11 16:36:39	2018-12-11 16:36:39
??:??:??:DD:0D:F8	2018-12-11 16:37:15	2018-12-11 16:37:15
??:??:??:7D:0F:8F	2018-12-11 16:38:23	2018-12-11 16:38:23
??:??:??:DD:0E:06	2018-12-11 16:41:48	2018-12-11 16:41:48
??:??:??:29:AA:C3	2018-12-11 16:41:59	2018-12-11 16:42:03

imsi_mac	first_time_detected	last_time_detected
268 03 2102985179	2018-12-11 16:23:02	2018-12-11 16:23:02
268 03 2204373421	2018-12-11 16:39:16	2018-12-11 16:39:07
268 03 2104096941	2018-12-11 16:23:05	2018-12-11 16:23:05
268 03 2105310385	2018-12-11 16:23:05	2018-12-11 16:23:05
268 03 1600991334	2018-12-11 16:23:05	2018-12-11 16:23:05
268 03 2104331787	2018-12-11 16:23:07	2018-12-11 16:23:07
268 03 2107740716	2018-12-11 16:23:07	2018-12-11 16:23:07
268 03 2105550531	2018-12-11 16:23:33	2018-12-11 16:23:25
268 03 1600084672	2018-12-11 16:23:08	2018-12-11 16:23:08
268 03 2104536679	2018-12-11 16:24:25	2018-12-11 16:24:23
268 03 1602548941	2018-12-11 16:39:05	2018-12-11 16:38:55
268 03 1600696707	2018-12-11 16:23:09	2018-12-11 16:23:09
268 03 2104532766	2018-12-11 16:23:11	2018-12-11 16:23:11
268 03 2106246335	2018-12-11 16:32:48	2018-12-11 16:32:47
268 03 2204353976	2018-12-11 16:23:13	2018-12-11 16:23:13
268 03 2105266638	2018-12-11 16:23:14	2018-12-11 16:23:14
234 30 4161553360	2018-12-11 16:23:16	2018-12-11 16:23:16
268 03 2105999926	2018-12-11 16:38:55	2018-12-11 16:38:47

FIGURE 4.13: From left to right Wi-Fi,Bluetooth and GSM devices detected.

query's the database to see if the device was previously detected and, if so, update only the last time seen field. If it was the first detection then it created a new record. The code for leading with this request could be seen in appendix N.

The website presented then do an HTTP GET request to gather the information to be shown. In this request, it is sent in the form of a parameter what information it wants to receive. It could be the total number of detected devices by technologies or the device's information and details for one specific technology. The code for leading with this request could be seen in appendix O.

4.8 InfluxDB and Grafana

After all the demonstrations it was needed to build a new structure for our solution. This new structure had to facilitate the analysis of results and store the data in a more compressed and correct manner. Also, this structure needed to be very close to the structure for the production environment so, for this structure it was chosen a micro-services architecture. This architecture enables quick deployment and migration of different modules and isolates all functions of a module in a container allowing communications between modules using only specific APIs defined. For this, we used Kubernetes containers for each module created, the database and the visualization framework.

A database for storing all the results of our device in real-time was needed and it needed to fulfill some requisites being them: (1) be a light database that could store a lot of data in a compressed and low memory usage manner; (2) query data rapidly usually by timestamps as the output would be mainly how many device where they in a given

time; (3) be optimized to store data from reading of sensors; (4) have support for data visualization tools to being able to see the results in real-time.

Having into consideration these requisites InfluxDB[30] was chosen. InfluxDB is an open-source time-series database that is focused on the Internet of Things applications. Being a time series database it is optimized for querying data based in timestamps and time intervals and present quick read and write speeds enabling real-time analysis of big data sets. Also InfluxDB as a lot of compatible frameworks for data visualization in real-time. This database is offered in two different options, as a cloud usage pay as you go database or by downloading the database and configuring all things necessary for usage. Our option was to download the open-source database, configuring it to our needs and deploy it in a Kubernetes container.

In InfluxDB there is not the concept of a table like in a SQL database, as it is a time-series database this concept of a table does not exist. In a time-series database, a timestamp is a point that identifies a single measurement. It works as a primary key for a timestamp in regular relational databases. Because of the absence from the normal base schema, it is possible to have fields to measurements as they are needed. There is no limitation of the number of fields like a table in an SQL, in a time-series database, it is possible to alter the schema over time. This flexibility is very important for solubility and future upgrades as the data measured and its formats could change over time there is no need to formally change the schema of our database, simply send a new measurement with the new fields and those fields are them available for consult.

InfluxDB stores data as measurements and not records. Because of that, it uses tags to insert details from measurements and associate a timestamp to a given value. A comparison of how data is stored in SQL relational databases and InfluxDB could be seen bellow in figures 4.14 and 4.15.

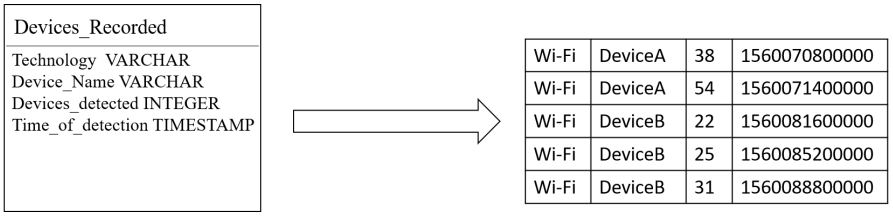


FIGURE 4.14: Example of records stored in a relational database.

Besides the differences of how data is stored InfluxDB query's data in SQL-like statements where data could be queried by the different tags of measurement or by time. It

[30]https://www.influxdata.com/

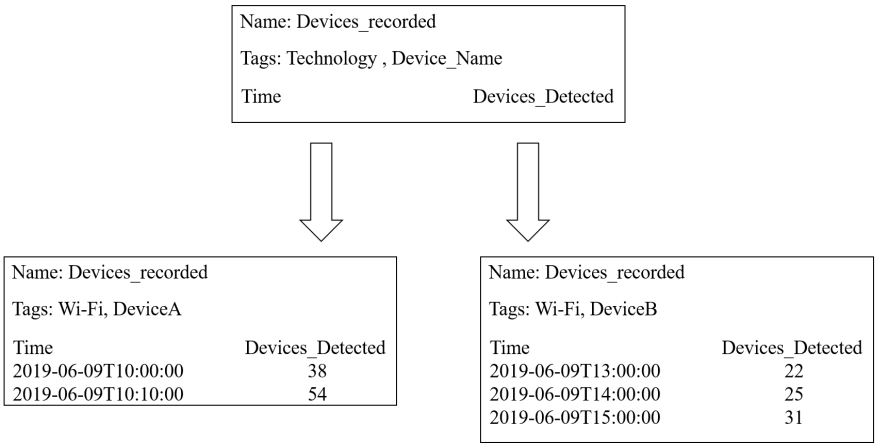


FIGURE 4.15: Example of records stored in a time series database.

as some limitations regarding updating and deleting data, for updating it needs to add the measurement again but with different values while for deleting it is needed to be given the exact time of the measurement besides is tags.

InfluxDB offers also a REST API built-in for inserting and querying data. Having this built-in functionality allowed for the easier and quicker deployment of this database in a Kubernetes container as the API for communications was already defined. For using this API it is only needed the query to be done to the database and authentication token of the user.

Our detection device when in need of inserting data, uses a curl command using an HTTP POST with the different tags (device name and location) and the number of devices detected. When having an internet connection available our solution could simply execute the curl command for inserting data, but when the internet is not available we propose that the information is sent using LoRaWan with only the amount of devices detected. The LoRaWan server then gathers this information and constructs the curl command for inserting data and executes it.

The data visualization framework will also use the HTTP API and curl commands to gather data using SQL-like query statements. The data visualization framework had to meet some requirements being them; (1) compatibility with time series databases most specific InfluxDB; (2) have the possibility to create new graphic and generate them with ease; (3) have the ability to show real-time data in graphics for analysis; (4) have the possibility to generate alerts for input data, for example if data is not received in a time interval sent a notification.

Having into consideration these requirements the framework chosen was Grafana[31]. Grafana is an open-source analytics and monitoring tool that is compatible with several databases including InfluxDB. It allows for configuring the connection with InfluxDB and querying data for input of different graph types. For creating a graph it only needs the SQL-wise query to gather the data and it automatically plots the information in real-time. It also allows for automatic notifications via email if the data received from a query matches a given pattern. The graphs generated in our testing phase after the deployment of our sensor where generated using this framework.

In figure 4.16 we present a deployment diagram representing the components deployed in the implementation of our network of sensors. It consists on a series of edge computing devices that upload data to our server via HTTP and store it in InfluxDB. Grafana will later receive this data and show it in real-time in a series of graphs.

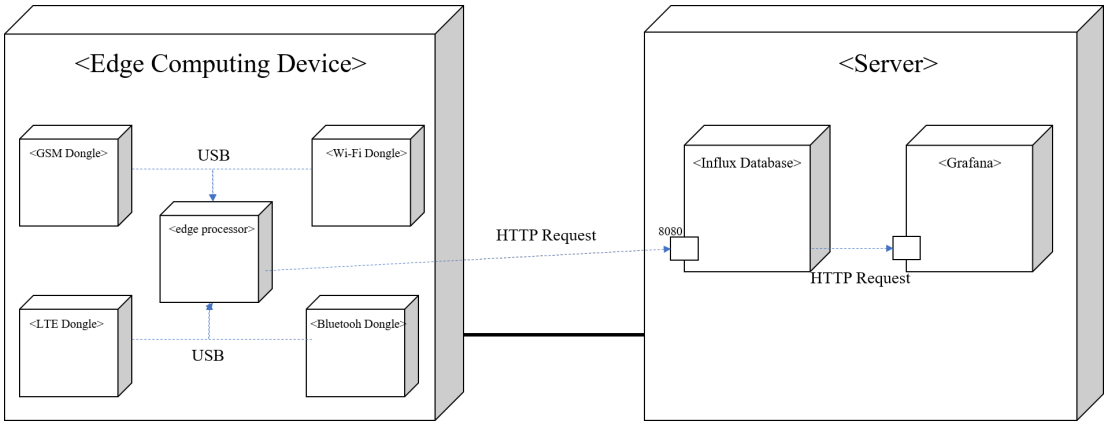


FIGURE 4.16: Deployment diagram of development phase.

4.9 Prototype built for our solution

In order to perform our testing and validation it was necessary to built a prototype that could be used for the deployment of our solution. This prototype and its components had to be designed having in consideration all the phases of testing to be done by our solution. A list of all the components available to be used in the solution are presented in figure 4.4.

For our prototype we aimed at a low cost device that fulfill most of the needs in tech-

[31] https://grafana.com

TABLE 4.4: Components used in the detection nodes.

COMPONENT	**FUNCTION**
Raspberry-Pi	Coordinate and Compute
Ubertooth One	Detect Bluetooth Devices
Alfa Network awus036ac	Detect Wi-Fi Devices
Lime-SDR	Detect 4G Devices
Nooelec NESDR SMArTee	Detect GSM/3G Devices

nology coverage. Also, the components present in the prototype needed to be compatible with the locations in which the device will be deployed. Our final validation phase will be in the museum of science and natural history participating in the European night of researchers event. The sensors will be deployed in an indoor environment where the coverage by the mobile network is weak to nonexistent. This is due to the properties of the building, an old building with large and thick stone walls.

Having this in consideration we selected for our prototype mainly components that did not detected devices in mobile network. Being that said the components present in the final prototype were: Ubertooth One for detecting devices using Bluetooth; Alfa Network awus036ac for detecting devices using Wi-Fi; Nooelec NESDR SMArTee for detecting devices using GSM.

With every component selected we contacted the department of architecture of our university for collaborating with the design and construction of the device. The Vitruvius FabLab[32] is an architecture lab that helps with the needs of architecture students and the architectural issues of our campus. This lab collaborated with us for the construction of the prototype. With the shared knowledge from this collaboration it where constructed two types of cases for our prototype. Both cases needed to withstand harsh environmental conditions including high temperatures and rain as the solution could be deployed in an outdoor environment for testing purposes. Also, both cases needed to be discrete to blend with the environment in which they were going to be deployed. Another requisite for the cases was that they could be easily and rapidly manufactured if in need for guaranteeing flexibility for the deployment of new cases. Considering these requirements the cases were modeled in a 3D framework and them printed in 3D printers available in the lab or constructed from acrylic using a laser for cutting a sheet of acrylic and then glue the different parts of the case. Also, both cases had rubber rings or silicone in the removable parts for guaranteeing the tightness of the case.

[32] http://vitruviusfablab.iscte-iul.pt/

The case for the prototype shown in figure 4.17 was constructed from a single white sheet of acrylic that was cut using a laser. All the components inside the case that secure the raspberry pi and the different USB dongles were 3D printed. This case had the main objective of concealing all the antennas inside the case. Concealing the antennas inside the case had some drawbacks like a reduced range of detection due to obstruction of the line of sight of the antennas and also made the case larger. While being a bigger case it presented better results blending with the environment where they were deployed. This is mainly due to its shape and appearing like a normal Wi-Fi access point and not arousing curiosity from people passing by.

The prototype case present in figure 4.18 was constructed using a 3D printer except for the cover that was made from an acrylic sheet. This case aimed at being a more discrete implementation by having a smaller size case and using the antennas in the exterior of the case. While having a smaller size it arouse more curiosity from people passing by mainly because of the antennas and this caused this type of cases to highlight from the environment in which they were deployed.

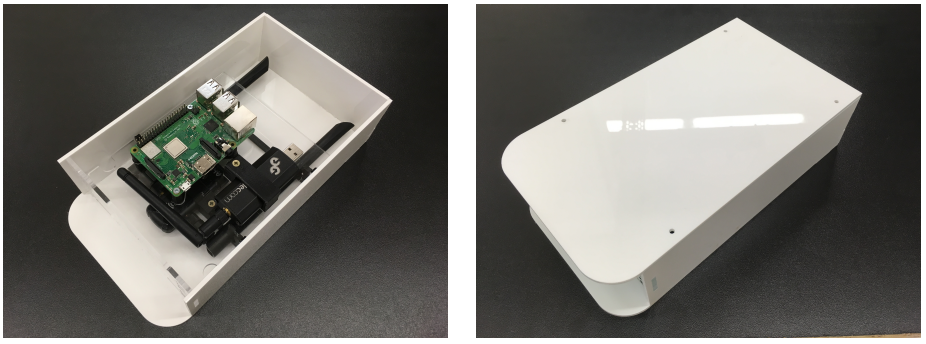


FIGURE 4.17: Prototype larger in size but with no exposed antennas.

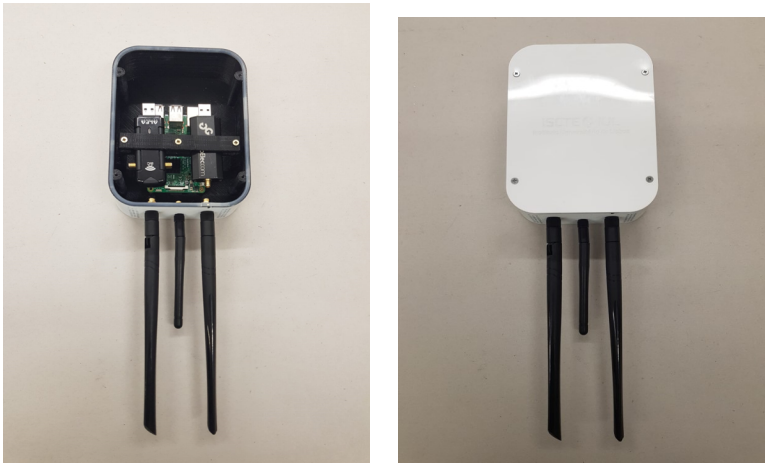


FIGURE 4.18: Prototype smaller in size but with exposed antennas.

Chapter 5

Testing and Validation

In this chapter we present the results gathered from all the tests done to validate our solution. For performing these tests we have used the prototype mentioned in section 4.9 that was built to withstand all the different environments were our solution could be deployed. All the tests performed after the construction of the prototype aimed to validate and calibrate our solution. Having the solution calibrated and tested the main object of the project was to represent the research group of our university in the European night of researchers and demonstrate real-time detection to the audience of the event.
This chapter is organized as follows, in section 5.1 we present the results of our preliminary tests and in section 5.2 we present the results of the 24/7 testing done around the university campus. Finally in section 5.3 we present the final validation test for our solution that were gathered in the European night of researchers.

5.1 Preliminary tests

In this section we present the preliminary results of the first tests done with our solution. These results were also displayed in a previously written paper[Rúben Dias da Silva2019]. Because we foresee that our solution will be deployed in areas with different characteristics we have opted for tests in typical usage scenarios. These scenarios were in both indoor and outdoor environments as well as in areas with high and low passage of people and high and low permanency time. We measured the amount of devices detected from our solution in the different technologies. These scenarios represent the wide possible

deployment options in which we aim to apply our solution. By testing our solution in this manner we aim to foresee the behaviour of the solution in the production phase and adapt the prototype in order to have the best behaviour possible.

During these tests we have compared the number of devices detected with direct a observation of the number of people in the vicinity of the sensor. Using this method of manual count we aim to calculate the effectiveness of our detection approach and to calibrate our sensor for future use. The manual counting method introduces some validation threats to our solution that could lead to a miss calibration of the device. This validation threats are: (1) human error, since in the reading process the manual count could lead to people not accounted or even more people accounted than the one present in the area.; (2) Periodicity of counting, because the manual count process was done in different moments each one spaced in regular intervals from each other, and this periodicity could not be enough to reflect the change in the amount of people present in the area.; (3) different radius of detection, different technologies have different radius of detection and one single count of the amount of people present in the area could not be enough to calibrate all technologies present in the solution; (4) free flow of people, since people could move in and out from the area of detection freely and the manual count method did not take this into account, only giving a single counting in a single moment in time.; (5) People that are not in line of sight, because our solution could be detecting devices that are not in line of sight but are present in the area.

While some validity threats impact in validation can be mitigated, others are inherent of the method and will always be present in the validation process. For example threats (1) and (3) could be mitigated with manual counts from two or more different persons at the same time and by having different counts based on how far from the device every person is. Threat (2) could not be mitigate because we are always going to compare some static reading to a real-time continuous reading. Having in consideration the methods of detection and validation used we present bellow a series of tests in different scenarios. All the graphs shown in this section were obtained using python scripts produced mainly for this purpose using the matplotlib[1] library.

The results in figure 5.1 were obtained when the sensor was deployed in a scenario with high flow of people through a narrow passage in an indoor environment.

The data obtained for the graph uses a sliding window approach, where each data point

[1] https://matplotlib.org

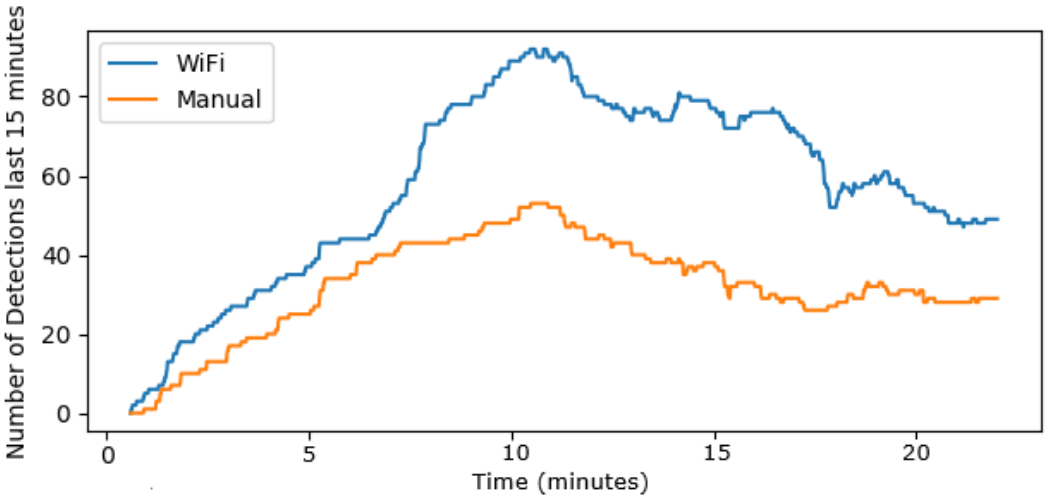


FIGURE 5.1: Number of devices detected on Wi-Fi vs manual detection in high flow.

represents the number of people that have passed close to our sensor in the last 15 minutes. This sliding window approach was also used for plotting the graphs in figures 5.2, 5.3 and 5.4.

In this graph it is possible to compare the amount of devices detected in Wi-Fi with the manual counting. The manual counting done in this scenario was slightly different from the other scenarios. In this scenario, because there was a narrow passage, we could have a more precise reading of the people passing by. So, for every person that passed right next to our device we registered it. It is possible to see a close correlation between the devices detected and the manual counting done. The amount of devices are higher than manual counting and this could be caused by the noise generated by devices that use MAC randomization. Regardless, our solution was very effective in detecting the patterns of movement of people adjusting to an intense flow or a less intensive flow of people, in sync with the data from the manual count. When observing closely the two patterns of detection we have seen that they behave in close similarity but usually with some delay from each other. Wi-Fi detections tended to show the arrival of people sooner than the manual count and this disparity made sense taking in consideration that Wi-Fi detected the device when it appeared in its range of detection but the device was only counted manually when it passed right next to the sensor. The next two tests where both performed in an open space area, with a low flow of people, for indoor (figure 5.2) and outdoor (figure 5.3) environments. It is important to learn how our solution behaves in this type of scenarios as it will also be common in overcrowded situations, where people will tend to move at lower speed or have higher permanency periods.

Figure 5.2 represents a situation where the sensor was exposed to a sudden increase of the number of people present in the area. The sensor when subjected to this situation had some delay until the number of devices detected matched the number of people present in the vicinity of the sensor. After this situation, the sensor tends to closely follow the manual count values with a very good correlation showing the ability to a stable reading when the amount of persons in the area doesn't change. This test also shows how our solution reacts to a sudden crowding situation. Our sensor had some delay until it reaches a stable value, but nevertheless, provided the feedback of this sudden increase of devices in the area in quasi-real-time.

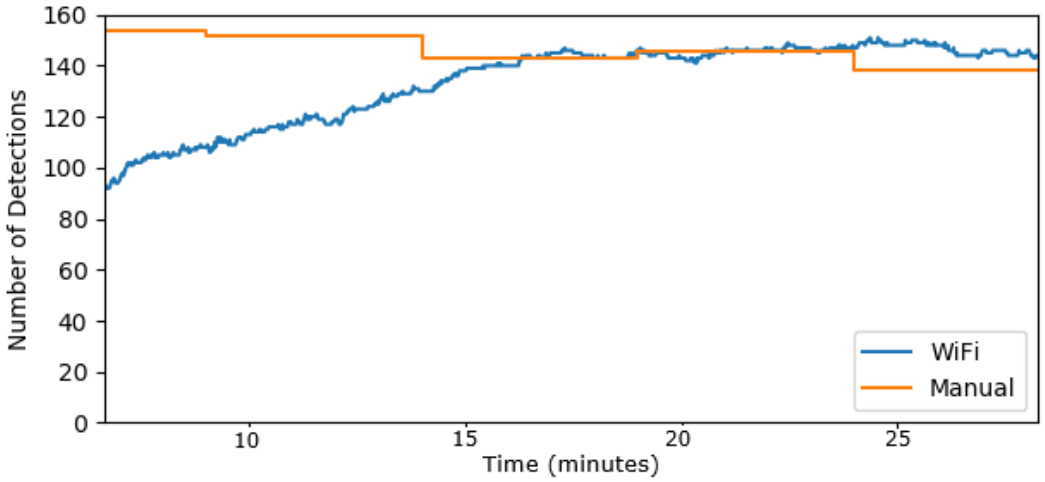


FIGURE 5.2: Indoor- Number of devices detected on Wi-Fi vs manual.

Figure 5.3 shows the behaviour of our sensor in an outdoor environment in a open space. In this situation the number of devices detected by the sensor start at a slightly higher value than the manual counting but both values approximate each other in the end of the experience. This difference confirms our suspicions regarding the characteristics of the outdoor environment used for the test, a courtyard surrounded by the main building of the campus. As such, the sensor is counting not only devices in line-of-site, but also devices inside the building. By the end of this test there was a break time between classes and students tended to left their classrooms and walk to the courtyard. This situation could be the motive for the manual count only matches the number of devices detected in the end of the test, as only then the detected devices, more precisely the person wearing the devices, could be seen in line of sight. Nevertheless this test confirms that detections are a fair estimate for manual observations.

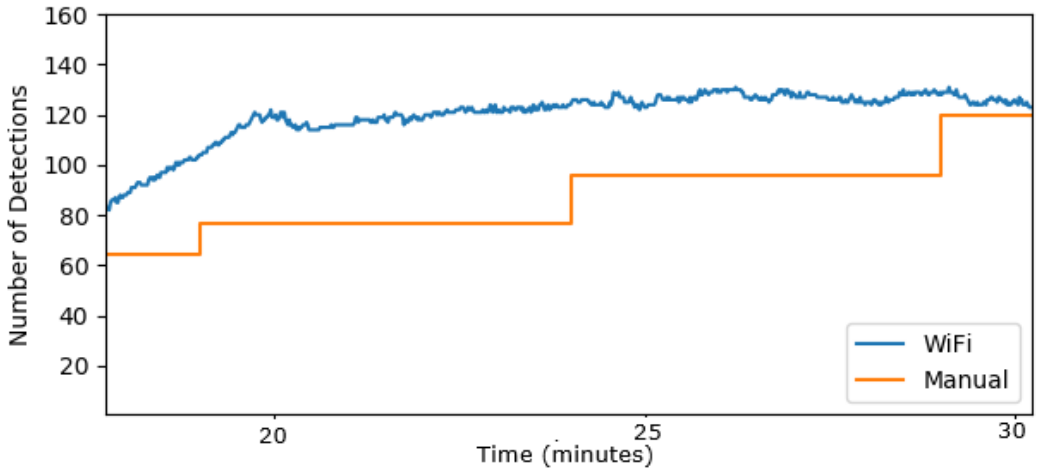


FIGURE 5.3: Outdoor- Number of devices detected on Wi-Fi vs manual.

Figure 5.4 shows detections performed by our solution using several technologies simultaneously, in an open space area. In this test it is possible to see the different behaviour of each technology when exposed to this situation. Wi-Fi detection presents a convincing result when compared to reality confirming previous results in tests, closely matching the growth of the number of people in the area closely but presenting some deviations from manual data in some particular times. Regardless of these particular points in time, it is the technology that more closely correlates the amount of devices detected with the number of persons present in the vicinity of the sensor.

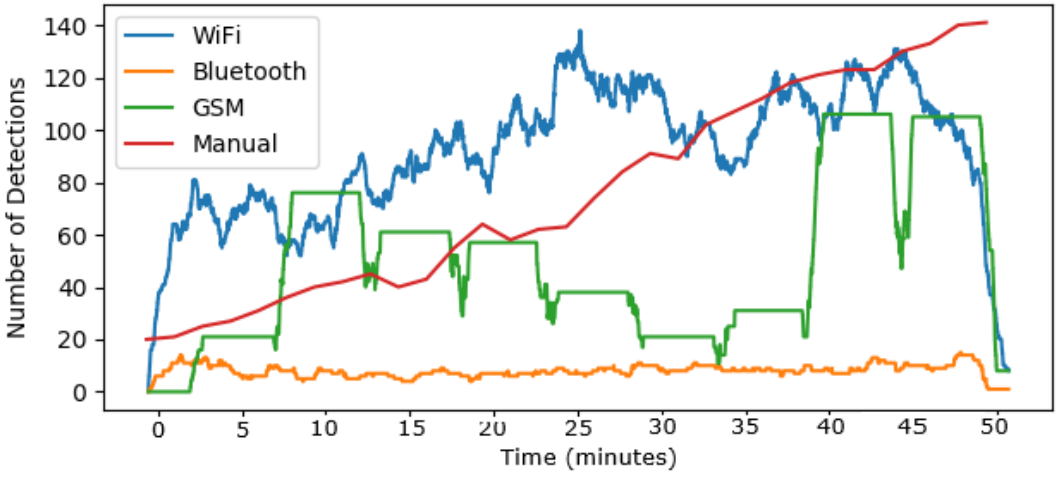


FIGURE 5.4: Number of devices detected using several technologies simultaneously.

When idealizing this test we predicted that Bluetooth would have some difficulties in detecting all the devices present in the vicinity of the sensor due to its limited range of

detection. By analysing the results of this test we confirm our predictions that Bluetooth shows a much lower detection rate when compare to Wi-Fi. The data shown by Bluetooth detections indicates that the number of devices detected did not changed over time, being mostly constant throughout the test which is not true according to manual count. Fusing the Wi-Fi and Bluetooth data we may make the assumption of how the room was filled in time. Using different ranges and number of devices detected from both technologies we could deduct that closer to the sensor the number of devices didn't change during the realization of the test, being the room filled with people in the areas further way from our sensor. This data integration could be very interesting in order to know patterns of movement of crowds and is spacial growth rate by adjusting the data with the amount of devices detected and the range on the technology that had done this detection.
In GSM data collection was done by round robin for a few minutes period each of the nearest cell towers. This could be confirmed in in the figure where different steps of data in GSM detection are shown, representing each one the amount of devices detected in a given cell tower. Between these steps a sudden drop of values could be seen and this is due to the hand-off time that our solution has when passing from one cell tower to another. We have used this approach for detecting devices in GSM due to limitations of hardware, that will be solved in a future iteration of our solution with the simultaneously analysis of different cell towers using a board with a wider bandwidth. For this reason, the obtained results are not yet conclusive.

5.2 24/7 real-time detection on university campus

In this section we present a series of tests that where obtained from having a network of sensors spread across our campus that were performing real-time detection simultaneously and uploading of the gathered data to our server. In this phase in order to validate the prototype so we have used several metrics to upload to our server for further analyses. While in the production phase the data to be sent will be only one metric of the number of devices detected in the vicinity of the sensor, in this phase we have sent the amount of devices detected in Bluetooth or Wi-Fi and also the amount of devices that were using mac randomization. We also sent the average staying times of the detected devices, to analyse if the sensor was recognizing the movement patterns in the vicinity.
We deployed our sensor in areas with different properties in order to test the behaviour of

our solution in different scenarios. The scenarios used were: (1) A very large study room where students stay for large periods of time; (2) Our university library, another place where students tend to stay for large periods of time; (3) One corridor that connects the two main building of our campus and is used intensively by teachers and students; (4) A tunnel that connects the two courtyards of the campus and is used for going from one courtyard to another; (5) One of the main entrances of our campus.
All the data shown in these tests was displayed in real-time using the data visualization framework Grafana. Also, the data gathered by our sensor in all the tests uses a sliding window approach, where each data point represents the number of people that have passed in the detection range of our sensor in the last 15 minutes. Due to hardware limitations only in scenarios (1) and (3) Bluetooth and Wi-Fi where detected simultaneously. In the other scenarios only Wi-Fi devices were detected.

5.2.1 Scenario 1- Large study room

In this scenario our solution was deployed in the main study room of our campus to perform real-time detections in Wi-Fi and Bluetooth technologies.
In figure 5.5 we present a snapshot of data gathered continuously in Wi-Fi in a three day interval. In the graph it is possible to see the pattern of devices detected along the passage of each day. The data collected shows a close correlation with the general usage of this space by students. Students tend to use this study room more in the morning hours than in the evening and in even lower numbers in the night. The study room is open 24 hours a day so students could use the study room in the night and with the data present here we can see that students have used this area at night. It is also interesting to note that the data gathered show a decrease in number of detections from around 12 AM to 2 PM each day. This fact is due to the departure of students from the area during lunch time. Also important to note that in the last day of the data presented the number of devices detected is much lower than in previous days and this could be justified by the fact that this day was a Saturday and so the study room is generally way less occupied in the weekends.

In figure 5.6 we present the data collected in the same conditions as the previous graph but for Bluetooth devices. In this graph there seems to be a correlation with the pattern of detections made from Wi-Fi and Bluetooth along the time of the experience.

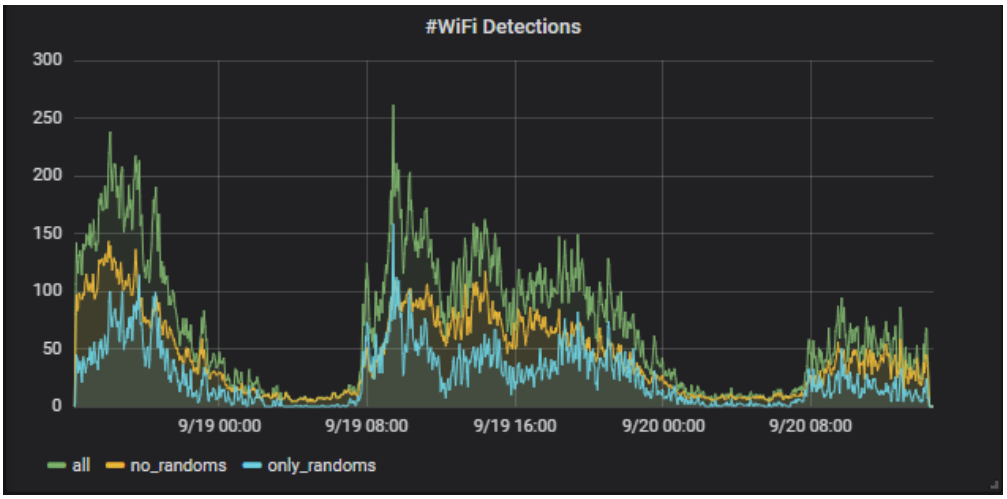


FIGURE 5.5: Number of detected devices in Wi-Fi in scenario 1 in a three day period.

Correlating this information with the fact that both technologies have different detection ranges we could assume that the study room was filled in an uniform was as the overall reading patterns from Wi-Fi are reflected in Bluetooth as well.

Also important to note that in it is possible to see some sudden spikes of increasing

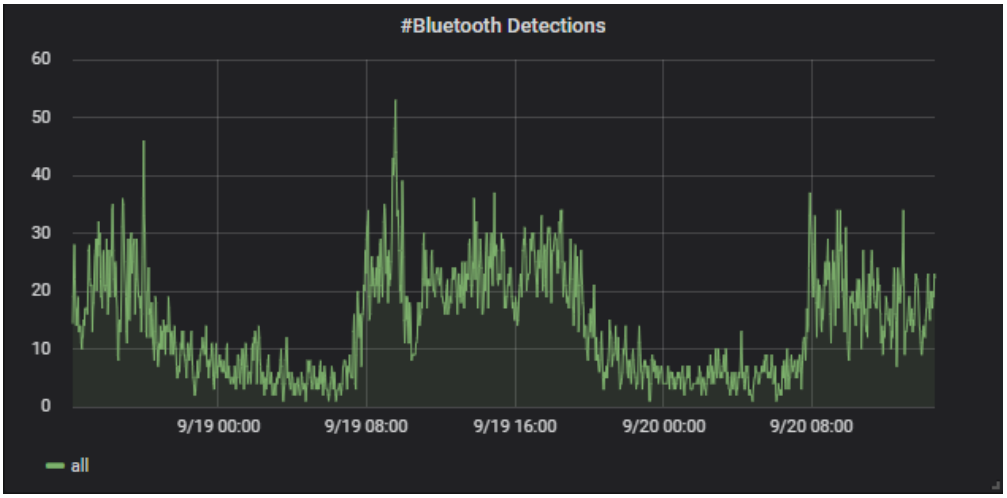


FIGURE 5.6: Number of detected devices in Bluetooth in scenario 1 in a three day period.

amount of detected devices in both technologies. This spikes are usually a reflection of the randomizatoin done by the operating system in MAC addresses. One interesting issue to see that in those spikes is that some times they appear simultaneously in Bluetooth and Wi-Fi leading to the conclusion that the same device is simultaneously using MAC randomization in both technologies. The median staying times in both situations were identical with staying time in day period of around 30 minutes and in night time up to

3 hours. For detections with randomly generated MAC addresses median staying times were always close to 5 minutes, the minimum amount of time configured for our sensor.

5.2.2 Scenario 2- Library

In this scenario our sensor was deployed in the largest open space present in the library of our campus. In figure 5.7 presents the results of a three day snapshot of data collected by our sensor.

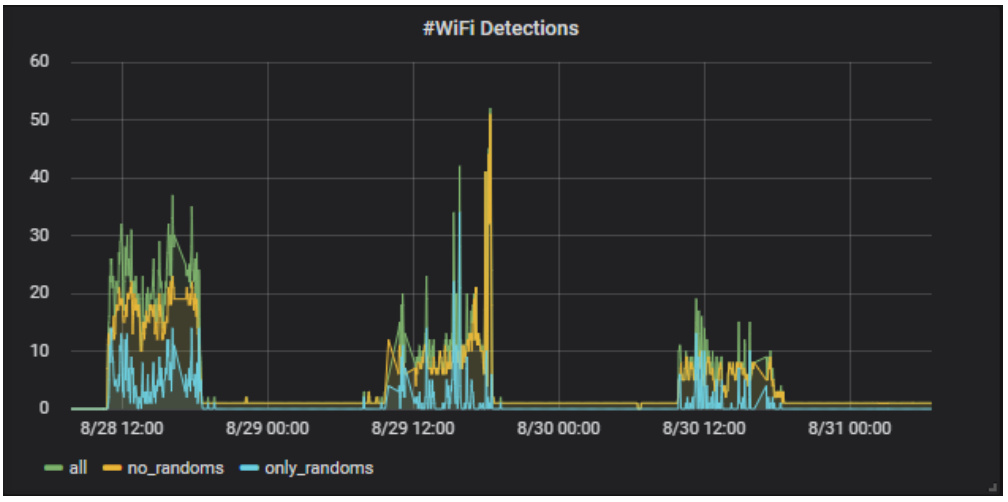


FIGURE 5.7: Number of detected devices in Wi-Fi in scenario 2 in a three day period.

For analysing this results it is important to refer the opening and closing hours of the library in the time of detection. The library opens at 10 AM and closes at 7 PM. This is an important fact to understand the behaviour of the detection patterns that our solution was able to gather. In this scenario, the amount of devices detected is slightly constant when the library is open, having only particular sudden spikes in the amount of devices detected. That behaviour matches the occupation usage of the library by the students. An interesting fact about the graph is the lack of detection when the library is closed and the sudden drop of devices detected right after the closing of the area. This fact could help validate that our sensor is able to detect rapidly and with some effectiveness when various devices leave an area suddenly. Regardless of staying times in this scenario they tended to be greater that in the previous scenario, being about three to four hours on average. Devices using MAC randomization had the same median staying times as the previous scenario.

5.2.3 Scenario 3- Indoor passage between two buildings

In this scenario our sensor was deployed at the corridor that connects the two main buildings of our campus. The sensor gathered data of the amount of devices detected in both Wi-Fi and Bluetooth. This area is characterized by having a narrow passage and a high flow of people that go from one building to another.

In figure 5.8 presents the results of the data gathered by our sensor for Wi-Fi devices. In the graph it is noticeable like in previous scenarios the low amount of devices detected during the night. Contrary to what happen in the library where the area becomes inaccessible this corridor could be used freely by the students at any time and that can be confirmed by the amount of devices detected during the night.

Also, this corridor is generally used in break times and during lunch and dinner hours

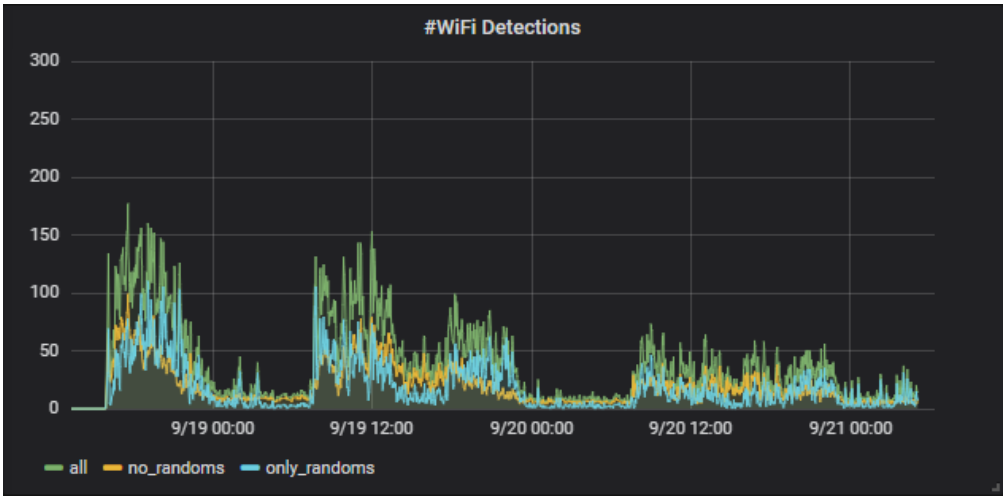


FIGURE 5.8: Number of detected devices in Wi-Fi in scenario 3 in a three day period.

for moving from one building to another. This situation could be verified in the graph with the appearing of more pronounced and constant detection of devices at the end and beginning of classes and during lunch and dinner breaks.

In this scenario, the average staying time for the detected devices was around 10 minutes and this matches the characteristics of the area. Figure 5.9 presents the data gathered by our sensor in the same condition of the above graph but in this case for Bluetooth devices. It seems that the data from both graphs have a good similarity having almost always the same pattern of detection.

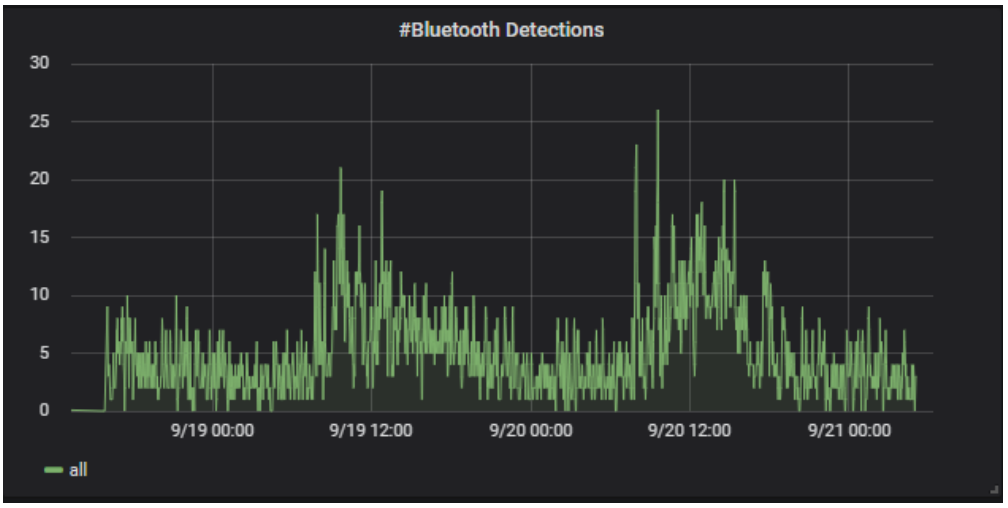


FIGURE 5.9: Number of detected devices in Bluetooth in scenario 3 in a three day period.

5.2.4 Scenario 4- Tunnel

In this scenario our sensor was deployed in an outdoor environment in the tunnel that connects the two courtyards of our campus. This area is characterized by having a mix of students passing from one courtyard to another with students that tend to stay in the area. In Figure 5.10 we present the data gathered from our sensor for a full day period. The behaviour of the data collected by our solution is similar to the behaviour in previous scenarios regardless the day and night cycle. Also in this scenario we verified that the number of devices detected is higher during lunch time. This relation is related

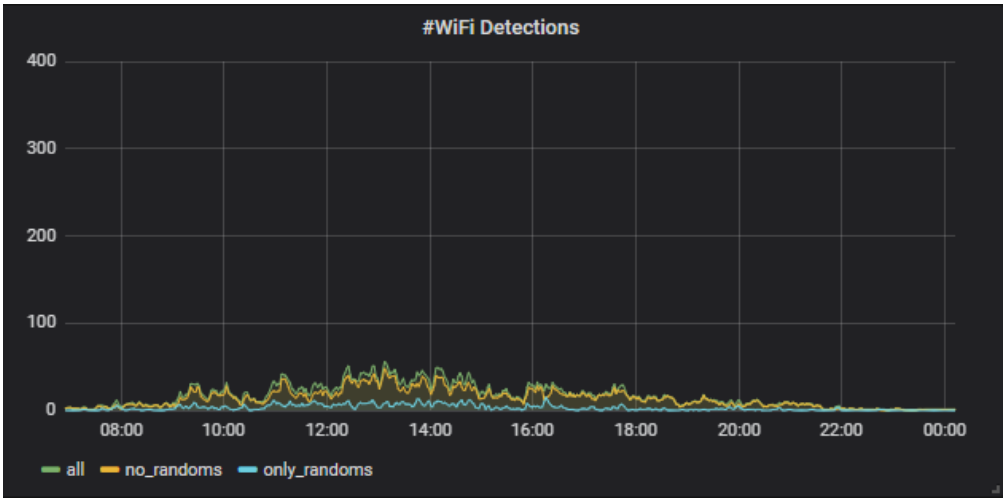


FIGURE 5.10: Number of detected devices in Wi-Fi in scenario 3 in a 24 hours period.

to the fact that the tunnel is used to access a courtyard that gives access to one of the

main cafeterias of our campus. Also during lunch time students tend to stay in the area of the tunnel for a big part of their lunch break. This hypothesis is sustained by the increase in value of the average staying time on lunch hours of around one our to the 30 minutes verified in the rest of the time.

In Figure 5.11 we present the data collected from our device during another time period for the same scenario. In this graph the data has some strange pattern behaviours compared to previous experiences. In this data set during the evening and the night our sensor detected a huge increase in the number of devices in the area. This high number of devices detection remained until around 3 to 4 AM. After further investigation we have discovered that a student party had taken place in the tunnel and one of the courtyard. This student party aimed to celebrate the start of the school year.

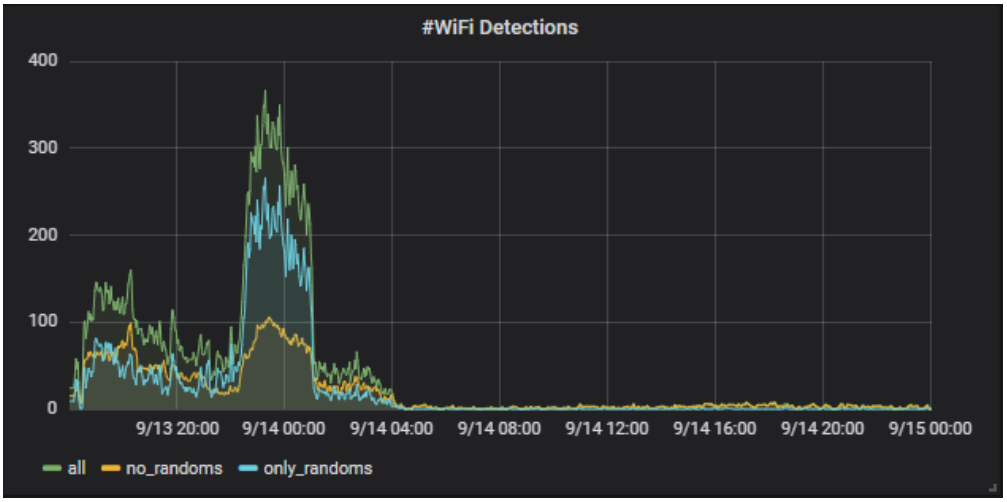


FIGURE 5.11: Number of detected devices in Wi-Fi in scenario 4 in a 24 hours period.

This scenario perfectly illustrated what our sensor needs to detect, sudden increases in the number of devices in the area that are not usual in the given time and location. Our sensor will be exposed to this type of scenarios when trying to detect overcrowded areas and is satisfactory that it presented good results identifying this situation.

5.2.5 Scenario 5- Campus entrance

In this scenario our sensor was deployed in an outdoor environment in one of the main entrances of our campus. In figure 5.12 we present the data collected from our sensor

in a 3 day period in this scenario. From the graph we could behold again the night and day cycles and their different pattern of detection. Also we could see that the highest amounts of devices detected where obtained during the start of the day, at around 8 to 9 AM, and at the late evening, around 5 to 6 PM. This data matches the patterns of arrivals of students in our campus as Classes start every day at 8 AM and the daytime students finish their classes at 5:30 PM. Also at 6 PM there is the start of classes of the post work students. In this scenario the average staying time was around 20 minutes.

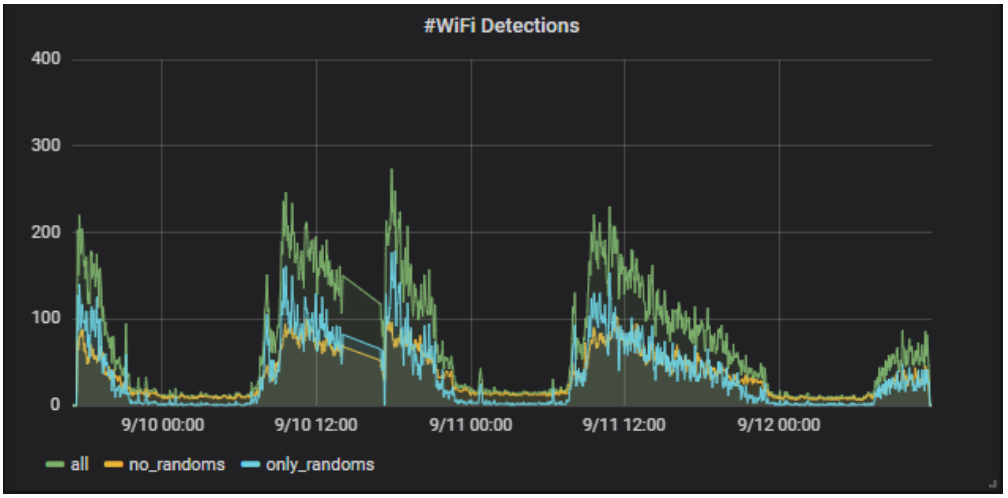


FIGURE 5.12: Number of detected devices in Wi-Fi in scenario 5 in a three days period.

In table 5.1 we summarise the main results of the deployment of our solution in the different scenarios.

TABLE 5.1: Main results from the tests in the different scenarios.

Scenario	Average staying times	Results
Scenario 1- Large study room	30 minutes	Lunch break and day and night pattern detection
Scenario 2- Library	3-4 hours	Correlation between detections and library open hours quick arrival and departure detection.
Scenario 3- Indoor passage between two buildings	10 minutes	Breaks between classes and lunch break pattern detection
Scenario 4. Tunnel	30 minutes	Detecting normal usage and crowding patterns
Scenario 5- Campus entrance	20 minutes	Detecting arrival and departure patterns

5.2.6 Controlled environment tests

For validating the results of our sensor besides the previously mentioned tests we performed some tests in a controlled environment. These controlled environments were usually isolated classrooms or auditoriums.

In figure 5.13 we present the data collected by our device in a full week of classes. Analysing the results we could perceive the start and end time of a contiguous block of classes and even the break times between each class. The aim of these type of tests is to compare the values from the card readings of the entrance in the auditorium and the data gathered and calibrate our sensor. While not having access to that information from the IT department we did some manual counting to some blocks of classes.

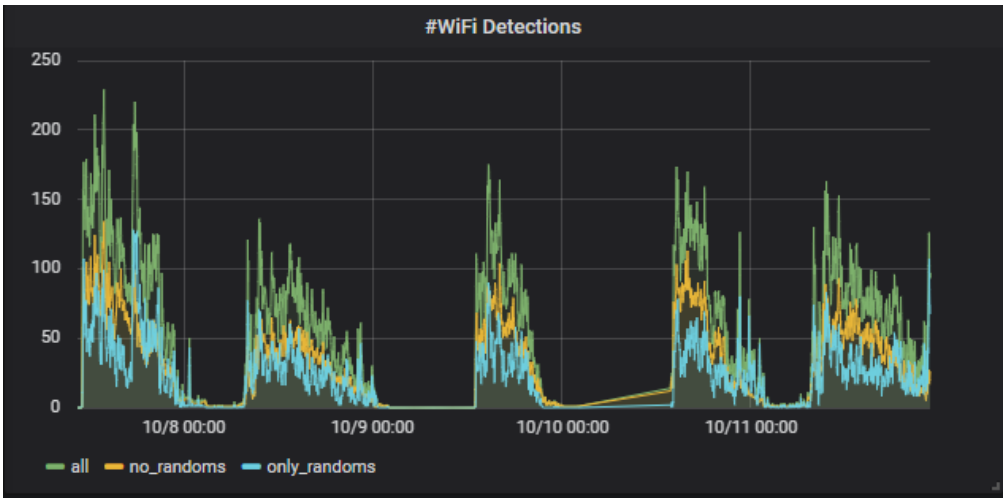


FIGURE 5.13: Number of detected devices in Wi-Fi on an auditorium in week.

In figure 5.14 we present only first day of data from the previous graph. Having now a more simple graph we could compare the values from the readings from our solution to the manual counting done. Also in this graph is more perceptible the breaks that happen between classes represented usually by a sudden decrease in the number of devices detected. The first class shown in the graph started at 11 AM and finished at at 12:30. In this class there were present 73 persons. After this time it is observed a decrease of the amount of detections until the next class start, at 1 PM. The next classes happened from 1 PM to 2:30 PM, 2:30 PM to 4 PM and from 4 PM to 5:30 PM. In these classes there were 67, 44 and 41 persons respectively. Comparing the given values with the values from the sensor we could see some disparity between them. We aim to reduce this disparity by calibrating the values of the sensor using further testing and analysis.

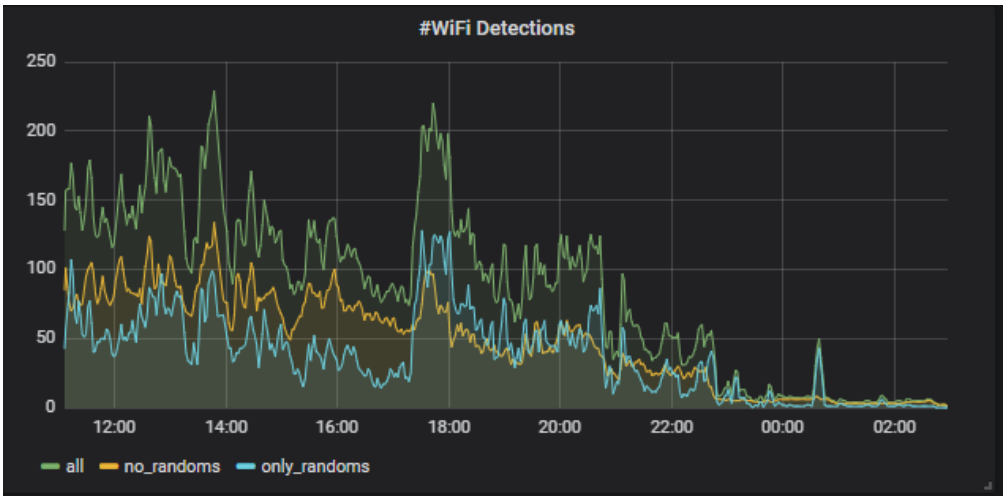


FIGURE 5.14: Number of detected devices in Wi-Fi on an auditorium in one day.

The next two tests where done in classrooms close to each other. In figure 5.15 we present the data collected from our sensor in a classroom. During the time in the graph there were two different classes, one from 7:30 PM to 9 PM and another from 9 PM to 10:30 PM. In each class there was 29 and 27 devices using Wi-Fi respectively. There is a difference between these values and value presented by our solution and that difference could be due the background noise generated by devices in nearby floors and classrooms. For determining that background noise all devices were turned of at 8:32 PM and turned back on at 8:56. They were also turned of at 10:08 and turned back on at 10:28 PM. In

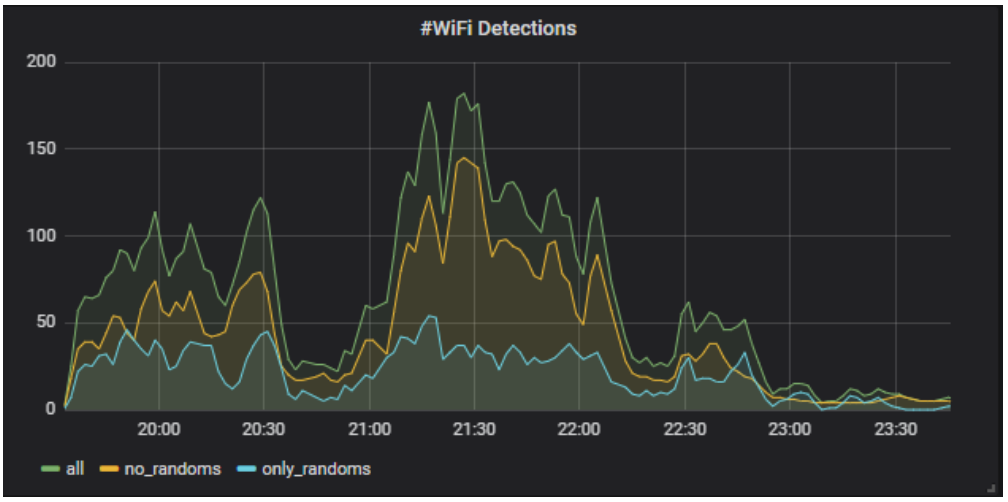


FIGURE 5.15: Number of detected devices in Wi-Fi on classroom 1.

figure 5.16 we present the results of detection in the nearby classroom. In this classroom there was only one class from 9 PM to 10:30 PM. In this class there were present 16 devices using Wi-Fi. In this experience all devices were turned off ant 10:08 PM and

turned back on at 10:28 PM. The results present in this graph are even disparate than previous experiences. We perceived that this disparity was due the proximity off both classrooms. Due to this proximity devices from both classrooms could have been detected from the device of the other classroom interfering in the final results.

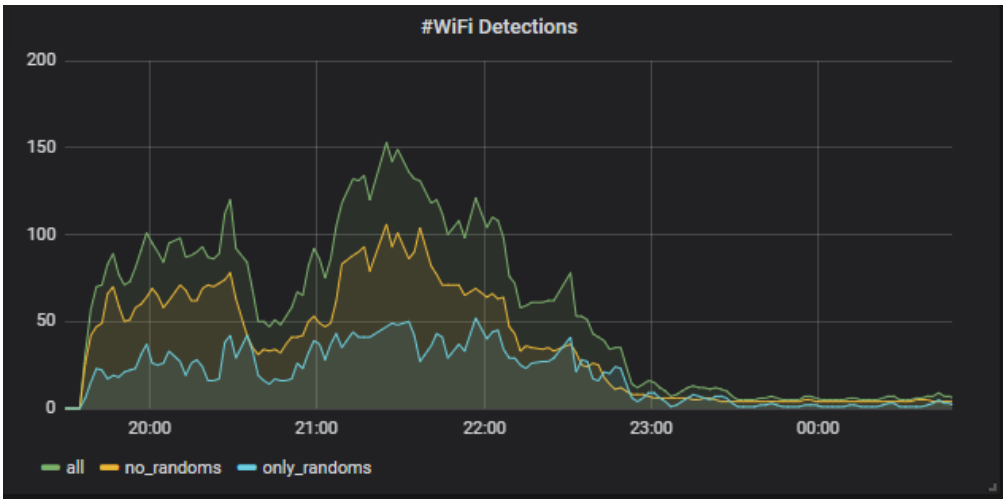


FIGURE 5.16: Number of detected devices in Wi-Fi on classroom 2.

5.3 European night of researchers 2019

In this section we present the last validation tests applied to our sensor. These tests consisted in the deployment of a network of sensors and counted in real-time the amount of people present in the event. The event location was the museum of science and natural history where several universities and their research groups gathered to show their top projects to the general public. Our project was chosen to represent one of the research groups of our university.

For presenting the work done in our project we wanted to show real-time detection of devices of our sensor to the public but the challenge was how to show this information in the best manner. The solution that we found was to develop a 3D map of the museum and represent the amount of devices detected in the map in real-time. For developing this 3D map we had the collaboration of the board of administration of the museum that gave us all the blueprints necessary for building this map. Having the blueprints of the device our research group plotted the blueprint in a program for representing architectural projects Revit[2] and built the 3D model. After the model had been done in 3D it

[2]https://www.autodesk.pt/products/revit/overview

was transferred for Unity for creating all the animations and connections to InfluxDB for gathering the data.

Another challenge that arouse after the making of the 3D map of the building was how would we represent the amount of people present in the event in the map. We proposed to solve this problem by plotting in the map stick-man figures representing each device detected. When plotting every stick-man we had to locate it in the vicinity of our sensor and the problem was first where is the area covered by this sensor and inside this area where to plot the figure. So two different studies where done. The first study was to perceive how distance could affect the possibility of detection of the device. The results of the mentioned study are present in figure 5.17 and where obtained in an isolated and controlled environment, on one floor of the garage in our campus.

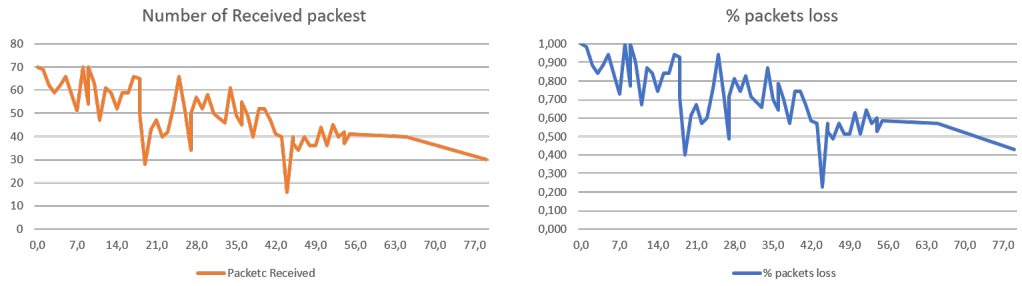


FIGURE 5.17: Tests done to perceive the behaviour of packet loss due to the increase in distance.

In this test a smartphone was used to send the same number of packets in the same time interval, 80 packets in a one minute interval. This action was done in increasingly larger distances for counting the amount of packets that where loss. The graphs presented shows that there is a direct correlation with the amount of packets received and the distance at the packet loss ratio in relation with the distance s7ms to be constant.
The other study done was to perceive where was the range of detection of every sensor. By combining the results of both these studies we could now decide where to deploy our figures representing the devices. After deploying the figures we used a script for simulating the movement of a person in the area of detection.

In figure 5.18 we present every location of the deployment our sensors in the building. these locations where chosen for having the biggest amount of area coverage of the building.

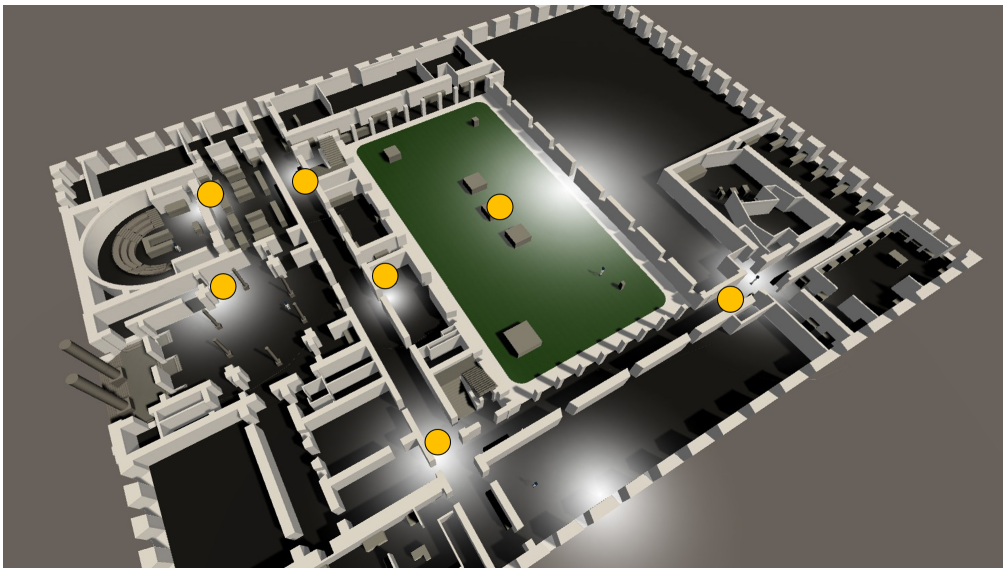


FIGURE 5.18: Location of the deployment of every sensor in the museum.

In figure 5.19 we present a snapshot of the map shown in event with real-time data of the amount of devices detected gathered from our sensors.

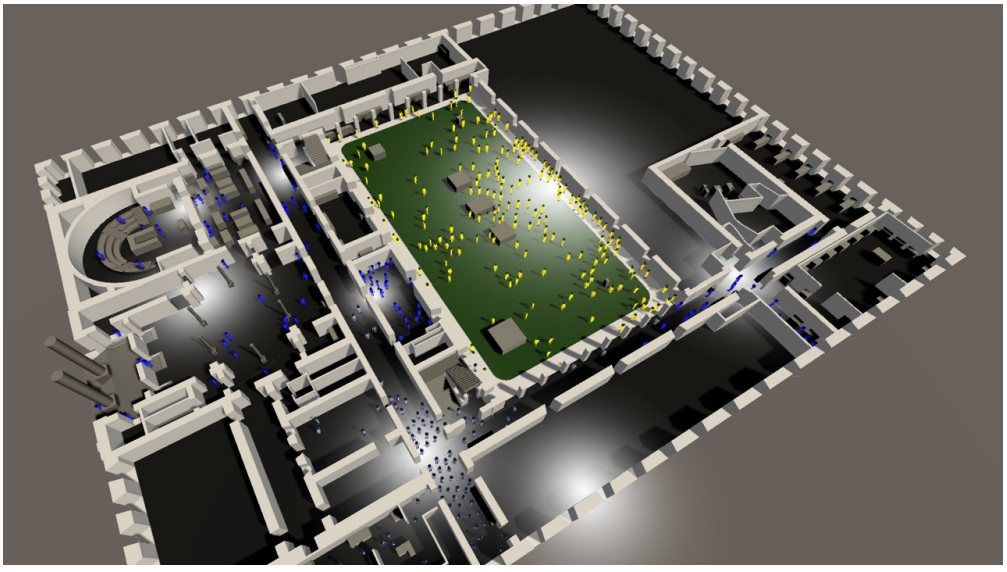


FIGURE 5.19: Snapshot of the real-time visualization of the amount of devices detected in the museum.

Our solution during the event shown in real time the main overcrowded areas present in the museum. The sensor shown that the main entrance and the cloister in the center of the building were the ones that were more crowded. By visual confirmation we can affirm that this real time prediction matched very closely what had happened in the museum during the event.

After the event the administration council of the museum were very pleased with the

results and proposed the implementation on our solution in the museum at full-time. They wanted to know how people move throughout the museum and the most visited expositions and our sensor could be a good solution for getting this data in real-time and also storing it for further analyses.

Chapter 6

Conclusion

The main research question of this dissertation was if it was possible to create a sensor that detects crowding in real-time by listening the trace elements of the usage of Wi-Fi, Bluetooth, and the Mobile Network. With the results obtained we can conclude that it is possible to build such a sensor to estimate crowding in its vicinity, by detecting the operation of multiple wireless technologies currently used by personal mobile devices.

The prototype developed is capable of detecting devices mainly in Wi-Fi and Bluetooth but also devices using the mobile network. It was also composed an infrastructure including a time-series database, a data visualization tool for real-time analysis and several APIs for communication purposes. The focus in Wi-Fi and Bluetooth technologies detection was due to the shift in final scenario of the application of the solution, firstly it was meant to be deployed in a outdoor scenario but it was latter deployed in an indoor scenario at the European Night of Researchers.

Our prototype has shown the ability to detect patterns in the movement of devices in its vicinity, the sudden rising in the number of devices or detecting the same patterns along a several days period and perceiving anomalies.

The other research question was if it is possible to combine the information from the different technologies detected. In response to this question we can only conclude that by analyzing the different counts of devices in each technology and its range we could determine some movement patterns of devices. For example, when correlating the data from the detection of Bluetooth and Wi-Fi devices in an open-area we can conclude how the area is filled by combining the results of the number of devices present in close vicinity to the sensor given by Bluetooth, and the overall amount of devices in the area,

given by Wi-Fi.

After the analysis of the results of our sensor, we foresaw the possibility of several applications of the solution besides crowding detection. Our sensor can be useful for several different purposes, such as quick law enforcement actions (e.g. for dispersing sudden hostile purpose gatherings) or short-term, unplanned, urban cleaning actions, due to the arrival of cruise ships, or other unforeseen manifestations. Also, by analyzing the geographical and temporal distribution of the crowding patterns, local authorities can improve their planning and assign their resources more efficiently, in areas such as security patrolling or waste collection, thereby improving urban management with benefits for both residents and visitors.

6.1 Future Work

The future work is organized in following research threads: 1) Upgrades to our prototype; 2) Validation and calibration of the sensor; 3) Deployment of the solution in the historic neighborhoods of Lisbon.

In (1) we aim to develop the capability to communicate the data gathered using LoRaWAN technology. For this, our prototype will use a raspberry shield that will enable data transfers using this technology. Our prototype will need to be able to receive software updates remotely and allow for the remote management of the sensor.Also our prototype needs upgrades to its sturdiest and resilience allowing the sensor to deal with spikes in electrical power, being abruptly turned off and failure recovery. The final upgrade to our prototype is to develop some artificial intelligence in the sensor that will allow it to perceive in which environment it is deployed and adjust different properties for a better accuracy (e.g. sliding window duration)

Regarding the domain (2) we aim to analyze the data-set of data collected and calibrate our sensor using a machine learning approach. By submitting the data collected to a machine learning algorithm together with the manual counting we will have a direct correlation of the different weights that the readings in the different technologies have in relation to the actual number of devices in the area. Also by combining this data with the range-precision range of each technology it will be possible to infer more accurate

movement patterns. Another validation procedure to be done to our data-set is to infer the relation between the data collected from different technologies and identifying devices that are represented in more than on technologies simultaneously and in future cases counting those devices only one time.

In domain (3) we aim to study how to implement our solution in the historic neighborhoods of Lisbon and chose the correct technologies to use in different nodes location. The different usage of technologies in different nodes guarantees a better coverage of the area by utilising their different range-precision properties. Also after this study is completed we aim to implement our solution in the historic neighborhoods of Lisbon.

Bibliography

[Agarwal et al.2015] Agarwal, R., Kumar, S., and Hegde, R. M. (2015). Algorithms for crowd surveillance using passive acoustic sensors over a multimodal sensor network. *IEEE Sensors Journal*, 15(3):1920–1930.

[Andersson et al.2010] Andersson, M., Ntalampiras, S., Ganchev, T., Rydell, J., Ahlberg, J., and Fakotakis, N. (2010). Fusion of acoustic and optical sensor data for automatic fight detection in urban environments. *13th Conference on Information Fusion, Fusion 2010*.

[Anglano2014] Anglano, C. (2014). Forensic analysis of whats app messenger on Android smartphones. *Digital Investigation*, 11(3):201–213.

[Bates2017] Bates, A. (2017). Stingray A New Frontier in Police Surveillance. (809).

[Cheng et al.2012] Cheng, N., Mohapatra, P., Cunche, M., Kaafar, M. A., Boreli, R., and Krishnamurthy, S. (2012). Inferring user relationship from hidden information in WLANs. In *Proceedings - IEEE Military Communications Conference MILCOM*, pages 1–6.

[Chernyshev2015] Chernyshev, M. (2015). An overview of bluetooth device discovery and fingerprinting techniques – assessing the local context FINGERPRINTING TECHNIQUES – ASSESSING THE LOCAL. In *13th Australian Digital Forensics Conference*, volume 2015, Perth.

[Coldwell2017] Coldwell, W. (2017). First Venice and Barcelona: now anti-tourism marches spread across Europe | Travel | The Guardian. *The Guardian*, (August):1.

[Colomb and Novy2009] Colomb, C. and Novy, J. (2009). *Protest and Resistance in.* Number Routledge 2011.

[Cunche et al.2014] Cunche, M., Kaafar, M. A., and Boreli, R. (2014). Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 11:56–69.

[Cunche et al.2012] Cunche, M., Mohamed Ali Kaafar, and Boreli, R. (2012). I know who you will meet this evening! Linking wireless devices using Wi-Fi probe requests. *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2012 - Digital Proceedings*.

[Di Luzio et al.2016] Di Luzio, A., Mei, A., and Stefa, J. (2016). Mind your probes: De-anonymization of large crowds through smartphone WiFi probe requests. *Proceedings - IEEE INFOCOM*, 2016-July(Section IV).

[Domínguez et al.2017] Domínguez, D. R., Díaz Redondo, R. P., Vilas, A. F., and Khalifa, M. B. (2017). Sensing the city with Instagram: Clustering geolocated data for outlier detection. *Expert Systems with Applications*, 78:319–333.

[Farrés] Farrés, J. C. Barcelona noise monitoring network. Technical report.

[Foddis et al.2014] Foddis, G., Garroppo, R. G., Giordano, S., Procissi, G., Roma, S., and Topazzi, S. (2014). LTE traffic analysis and application behavior characterization. *EuCNC 2014 - European Conference on Networks and Communications*, pages 2–6.

[Freudiger2015] Freudiger, J. (2015). Short: How talkative is your mobile device? An experimental study of Wi-Fi probe requests. In *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2015*, pages 1–6, New York, New York, USA. ACM Press.

[Introduction2018] Introduction, M. (2018). UE Detection Using Downlink Signals. pages 1–11.

[Kalogianni et al.2015] Kalogianni, E., Sileryte, R., Lam, M., Zhou, K., Van der Ham, M., Van der Spek, S., and Verbree, E. (2015). Passive WiFi monitoring of the rhythm of the campus. *Proceedings of The 18th AGILE International Conference on Geographic Information Science; Geographics Information Science as an Enabler of Smarter Cities and Communities, Lisboa (Portugal), June 9-14, 2015; Authors version.*

[Kashevnik and Shchekotov2012] Kashevnik, A. and Shchekotov, M. (2012). Comparative analysis of indoor positioning systems based on communications supported by

smartphones. In *2012 12th Conference of Open Innovations Association (FRUCT)*, volume 2012-Novem, pages 1–6. IEEE.

[Longo et al.2018] Longo, E., Redondi, A. E., and Cesana, M. (2018). Pairing Wi-Fi and Bluetooth MAC addresses through passive packet capture. *2018 17th Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2018*, pages 1–4.

[Lu et al.2009] Lu, H., Pan, W., Lane, N. D., Choudhury, T., and Campbell, A. T. (2009). SoundSense. pages 165–178.

[Martin et al.2017] Martin, J., Mayberry, T., Donahue, C., Foppe, L., Brown, L., Riggins, C., Rye, E. C., and Brown, D. (2017). A Study of MAC Address Randomization in Mobile Devices and When it Fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383.

[Matte et al.2016] Matte, C., Cunche, M., Rousseau, F., and Vanhoef, M. (2016). Defeating MAC address randomization through timing attacks. *WiSec 2016 - Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 15–20.

[Mesaros et al.2010] Mesaros, A., Heittola, T., Eronen, A., and Virtanen, T. (2010). Acoustic event detection in real life recordings. *European Signal Processing Conference*, (June):1267–1271.

[Meyer and Wetzel2004] Meyer, U. and Wetzel, S. (2004). A man-in-the-middle attack on UMTS. *Proceedings of the 2004 ACM Workshop on Wireless Security, WiSe*, pages 90–97.

[Miluzzo et al.2011] Miluzzo, E., Papandrea, M., Lane, N. D., Sarroff, A. M., Giordano, S., and Campbell, A. T. (2011). Tapping into the Vibe of the city using VibN, a continuous sensing application for smartphones. *Proceedings of 1st international symposium on From digital footprints to social and community intelligence - SCI '11*, page 13.

[Mueller et al.2015] Mueller, M., Schulz, D., Mock, M., and Hecker, D. (2015). Detecting Mobility Patterns with Stationary Bluetooth Sensors: A real-world Case Study. *18th AGILE International Conference on Geographic Information Science. 2015.*

[Newzoo] Newzoo. Global mobile market report 2018.

[Park et al.2017] Park, S., Bourqui, M., and Frias-Martinez, E. (2017). MobInsight: Understanding Urban Mobility with Crowd-Powered Neighborhood Characterizations. *IEEE International Conference on Data Mining Workshops, ICDMW*, pages 1312–1315.

[Peng et al.2015] Peng, P., Tian, Y., Wang, Y., Li, J., and Huang, T. (2015). Robust multiple cameras pedestrian detection with multi-view Bayesian network. *Pattern Recognition*, 48(5):1760–1772.

[Plumbley2010] Plumbley, M. D. (2010). Acoustic Event Detection in Real-life Environments. *18th European Signal Processing Conference*, (642685):642685.

[Qin et al.2013] Qin, W., Zhang, J., Li, B., and Sun, L. (2013). Discovering human presence activities with smartphones using nonintrusive Wi-Fi sniffer sensors: The big data prospective. *International Journal of Distributed Sensor Networks*, 2013.

[Ranneries et al.2016] Ranneries, S. B., Kalør, M. E., Nielsen, S. A., Dalgaard, L. N., Christensen, L. D., and Kanhabua, N. (2016). Wisdom of the local crowd. *Proceedings of the 8th ACM Conference on Web Science - WebSci '16*, pages 352–354.

[Redondi and Cesana2018] Redondi, A. E. and Cesana, M. (2018). Building up knowledge through passive WiFi probes. *Computer Communications*, 117(December 2017):1–12.

[Robyns et al.2017] Robyns, P., Bonné, B., Quax, P., and Lamotte, W. (2017). Noncooperative 802.11 MAC Layer Fingerprinting and Tracking of Mobile Devices. *Security and Communication Networks*, 2017.

[Rúben Dias da Silva2019] Rúben Dias da Silva, Rui Neto Marinheiro, F. B. e. A. (2019). Crowding Detection Combining Trace Elements from Heterogeneous Wireless Technologies. *2019 the 22nd international symposium on wireless personal multimedia communications, WPMC 2019 - Conference*, pages 1–6.

[Schauer et al.2014] Schauer, L., Werner, M., and Marcus, P. (2014). Estimating crowd densities and pedestrian flows using Wi-Fi and bluetooth. *MobiQuitous 2014 - 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 171–177.

[Shaik et al.2017] Shaik, A., Borgaonkar, R., Asokan, N., Niemi, V., and Seifert, J.-P. (2017). Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. (February):21–24.

[Stahlschmidt et al.2016] Stahlschmidt, C., Gavriilidis, A., Velten, J., and Kummert, A. (2016). Applications for a people detection and tracking algorithm using a time-of-flight camera. *Multimedia Tools and Applications*, 75(17):10769–10786.

[Strobel2007] Strobel, D. (2007). IMSI Catcher. In *Chair for Communication Security*, Bochum. Ruhr-Universitat Bochum.

[Vaishnavi and Kuechler2004] Vaishnavi, V. and Kuechler, B. (2004). Design Science Research in Information Systems Overview of Design Science Research. Technical report.

[Vanhoef et al.2016] Vanhoef, M., Matte, C., Cunche, M., Cardoso, L. S., and Piessens, F. (2016). Why MAC Address Randomization is not Enough. pages 413–424.

[Wetzel et al.2018] Wetzel, J., Zeitvogel, S., Laubenheimer, A., and Heizmann, M. (2018). Towards Global People Detection and Tracking using Multiple Depth Sensors. *2018 13th International Symposium on Electronics and Telecommunications, ISETC 2018 - Conference Proceedings*, pages 1–4.

[WTTC2018] WTTC (2018). Economic Impact 2018 World.

[Yang and Huang2018] Yang, Q. and Huang, L. (2018). *Inside Radio: An Attack and Defense Guide*.

[Zhou2018] Zhou, S. X. (2018). Investigation of LTE Privacy Attacks by Exploiting the Paging Mechanism. (May).

Appendices

A GNU-Radio Installation

For installing GNU-radio first it needs to be installed all the dependent libraries. All the libraries needed and the command to install them are show below

```
sudo apt-get -y install git-core cmake g++ python-dev swig
pkg-config libfftw3-dev libboost1.55-all-dev libcppunit-dev
libgsl0-dev libusb-dev libsdl1.2-dev python-wxgtk2.8 python
-numpy python-cheetah python-lxml doxygen libxi-dev python-
sip libqt4-opengl-dev libqwt-dev libfontconfig 1-dev
libxrender-dev python-sip python-sip-dev
```

Finally for installing GNU-Radio in an Ubuntu or Debian based Operating System, just run the next command with sudo privileges

```
sudo apt install gnuradio
```

B RTL-SDR driver installation

For installing the RTL-SDR diver it is necessary to install its libusb dependence

```
sudo apt-get install libusb-1.0-0-dev
```

Install the RTL-SDR driver

```
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr/
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
sudo cp ../rtl-sdr.rules /etc/udev/rules.d/
```

Finally blacklist the default driver by adding the following line located in /etc/modprobe.d

```
    blacklist dvb_usb_rtl28xxu
```

C Kalibrate-SDR installation

To install Kalibrate-SDR import the project from git using the following commands

```
git clone https://github.com/steve-m/kalibrate-rtl.gitcd
   kalibrate-hackrf
./bootstrap
./configure
make
sudo make install
```

D gr_gsm installation

First is necessary to install all the dependencies needed.

```
  sudo apt-get install git cmake libboost-all-dev libcppunit-
    dev swig doxygen liblog4cpp5-dev python-scipy
```

To install the gr_gsm program just import the project from git using the following commands

```
git clone https://github.com/ptrkrysik/gr-gsm.git
cd gr-gsm mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

E IMSI_Catcher installation

To install IMSI_Catcher the following dependencies

```
sudo apt install python-numpy python-scipy python-scapy
```

Finally import the project from github

```
git clone https://github.com/Oros42/IMSI-catcher
cd IMSI-catcher
```

F OpenLTE installation

To install OpenLTE it is necessary to install its libpolarssl dependence

```
sudo apt-get install libpolarssl-dev
```

Finnaly install openLTE from the source code available at sourceforge

```
wget https://sourceforge.net/projects/openlte/files/openlte_v00
    -20-05.tgz/download
tar -xvzf openlte_v00-20-05.tgz
cd openlte_v00-20-05
mkdir build
cd build
sudo cmake ../
sudo make
sudo make install
```

G srsLTE installation

To install srsLTE it is necessary to install its dependencies

```
sudo apt-get install cmake libfftw3-dev libmbedtls-dev libboost
    -program-options-dev libconfig++-dev libsctp-dev
```

To Install srsLTE import the source code code

```
git clone https://github.com/srsLTE/srsLTE.git
cd srsLTE
mkdir build
cd build
cmake ../
make
make test
sudo make install
srslte_install_configs.sh user
```

H Driver installation for alpha network AWUS036AC

To install the driver for the Alpha Network AWUS036AC import the following github project and install it

```
git clone https://github.com/abperiasamy/rtl8812AU_8821AU_linux
    .git
cd rtl8812AU_8821AU_linux
make
sudo make install
```

I Aircrack-ng installation

To install Aircrack-ng it is necessary to install its dependencies

```
sudo apt-get install build-essential autoconf automake libtool
    pkg-config libnl-3-dev libnl-genl-3-dev libssl-dev ethtool
    shtool rfkill zlib1g-dev libpcap-dev libsqlite3-dev libpcre3
    -dev libhwloc-dev libcmocka-dev hostapd wpasupplicant
    tcpdump screen iw usbutils
```

Finally install Aircrack-ng from the source code available at github

```
git clone https://github.com/aircrack-ng/aircrack-ng.git
cd aircrack-ng
./configure
make
make test
```

J Bluez installation

To install Bluez first intall its dependencies

```
sudo apt-get install dbus kmod libasound2 libc6  libdbus-1-3
   libdw1 libglib2.0-0  libreadline7 libudev1 lsb-base udev
```

Now install the Bluezpackage

```
sudo apt-get install bluez
```

K Ubertooth installation

To install Ubertooth it is necessary to install its dependencies

```
sudo apt-get install cmake libusb-1.0-0-dev make gcc g++
   libbluetooth-dev \
pkg-config libpcap-dev python-numpy python-pyside python-qt4
```

Install the library libbtbb

```
wget https://github.com/greatscottgadgets/libbtbb/archive
   /2018-12-R1.tar.gz -O libbtbb-2018-12-R1.tar.gz
tar -xf libbtbb-2018-12-R1.tar.gz
cd libbtbb-2018-12-R1
mkdir build
cd build
```

```
cmake ..
make
sudo make install
sudo ldconfig
```

Install ubertooth from the source code available at github

```
wget https://github.com/greatscottgadgets/ubertooth/releases/
    download/2018-12-R1/ubertooth-2018-12-R1.tar.xz
tar xf ubertooth-2018-12-R1.tar.xz
cd ubertooth-2018-12-R1/host
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

L HTTP POST request for inserting a detection of a GSM device

The PHP code that deals with the HTTP POST request and inserts the data in the database could be seen bellow.

```
case 'POST':
 echo 'POST metodo: ';
 $sql="insert into cellphones (pais, marca, latitude, longitude,
    operador)
 values ('$pais', '$marca', '$latitude', '$longitude', '$operador
    ') ";
 $result=$db->query($sql);
 print "new register:" . $db->insert_id;
break;
```

M HTTP GET request for getting the amount of GSM detected devices

The PHP code that deals with the HTTP GET request and return a JSON with the list of countries and the amount of devices detected could be seen bellow

```
case 'GET':
 if(isset($path_params[1])){
  if($path_params[1] == "cellphones"){
   $sql="select * from cellphones";
   $result=$db->query($sql);
   while($row = $result->fetch_assoc()){
    $list[]=$row;
   }
   header("Access-Control-Allow-Origin: *");
   header('Content-Type: application/json');
   print json_encode($list, JSON_UNESCAPED_UNICODE);
  } else {
   if ($path_params[1] == "towers") {
    $sql="select latitude , longitude from towers";
         if (isset($path_params[2]) && is_numeric($path_params[2])
   ){
          $sql.= " where lac=" . $path_params[2]  ;
          $sql.=" and mnc=".$path_params[3] ;
         }
         $result=$db->query($sql);
         while($row = $result->fetch_assoc()){
          $list[]=$row;
         }
         header("Access-Control-Allow-Origin: *");
         header('Content-Type: application/json');
         print json_encode($list, JSON_UNESCAPED_UNICODE);
        }
        else{ if($path_params[1] == "call_count"){
         $sql="select pais , count(*) as calls from cellphones
   group by pais";
         $result=$db->query($sql);
         while($row = $result->fetch_assoc()){
```

```
          $list[]=$row;
         }
         header("Access-Control-Allow-Origin: *");
         header('Content-Type: application/json');
     print json_encode($list, JSON_UNESCAPED_UNICODE);
        }
    else{
         print "needs parameter 1";
        }
   }
  }
 }
break;
```

N HTTP POST request for inserting a detection a device in all technologies

The PHP code that deals with the HTTP POST request and inserts the data in the database could be seen bellow.

```
case 'POST':
 $sql="SELECT * from deteccao_geral where imsi_mac='$idRegisto'";
 $result=$db->query($sql);
 print $result->num_rows ;
 if($result->num_rows > 0){
  $sql2="update deteccao_geral set last_time_detected='
   $lastTimeDetected'
  where imsi_mac='$idRegisto' ";
   $result2=$db->query($sql2);
 }else{
  $sql2="insert into deteccao_geral (tecnologia, imsi_mac,
   first_time_detected, last_time_detected) values ('$tecnologia',
    '$idRegisto', '$firstTimeDetected', '$lastTimeDetected') ";
  $result2=$db->query($sql2);
 }
break;
```

O HTTP GET request for getting the list of all devices detected

The PHP code that deals with the HTTP GET request and return a JSON with the list of all devices detected or the overal count of devices by technology could be seen bellow.

```
case 'GET':
 if($path_params[1] == "technologies"){
  $sql="select tecnologia , count(*) as calls from deteccao_geral
   group by tecnologia";
  $result=$db->query($sql);
  while($row = $result->fetch_assoc()){
   $list[]=$row;
  }
  header("Access-Control-Allow-Origin: *");
  header('Content-Type: application/json');
  print json_encode($list, JSON_UNESCAPED_UNICODE);
 }
 else{
  if($path_params[1] == "Wi-Fi"){
   $sql="select imsi_mac , first_time_detected ,
   last_time_detected from deteccao_geral where tecnologia='Wi-Fi'
   ";
   $result=$db->query($sql);
   while($row = $result->fetch_assoc()){
    $list[]=$row;
   }
   header("Access-Control-Allow-Origin: *");
   header('Content-Type: application/json');
   print json_encode($list, JSON_UNESCAPED_UNICODE);
   }else {
    if($path_params[1] == "Rede Movel"){
     $sql="select imsi_mac , first_time_detected ,
   last_time_detected  from deteccao_geral where tecnologia='Rede
   Movel'";
     $result=$db->query($sql);
     while($row = $result->fetch_assoc()){
```

```
      $list[]=$row;
     }
     header("Access-Control-Allow-Origin: *");
     header('Content-Type: application/json');
     print json_encode($list, JSON_UNESCAPED_UNICODE);
         }
         else {
          if($path_params[1] == "Bluetooth"){
           $sql="select imsi_mac , first_time_detected ,
   last_time_detected  from deteccao_geral where tecnologia='
   Bluetooth'";
           $result=$db->query($sql);
           while($row = $result->fetch_assoc()){
            $list[]=$row;
           }
           header("Access-Control-Allow-Origin: *");
           header('Content-Type: application/json');
           print json_encode($list, JSON_UNESCAPED_UNICODE);
          }
         }
    }
   }
break;
```