

iscte

UNIVERSITY
INSTITUTE
OF LISBON

Smart Tourism Toolkit for Crowd-monitoring Solutions

Tomás Miguel Mestre dos Santos

Master in Telecommunications and Computer Engineering

Supervisor:

Doctor Rui Neto Marinheiro, Associate Professor,
Iscte-IUL

Co-supervisor:

Doctor Fernando Brito e Abreu, Associate Professor,
Iscte-IUL

October, 2023

[This page has been intentionally left blank]



TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Smart Tourism Toolkit for Crowd-monitoring Solutions

Tomás Miguel Mestre dos Santos

Master in Telecommunications and Computer Engineering

Supervisor:

Doctor Rui Neto Marinheiro, Associate Professor,
Iscte-IUL

Co-supervisor:

Doctor Fernando Brito e Abreu, Associate Professor,
Iscte-IUL

October, 2023

[This page has been intentionally left blank]

Smart Tourism Toolkit for Crowd-monitoring Solutions

Copyright © 2023, Tomás Miguel Mestre dos Santos, School of Technology and Architecture, University Institute of Lisbon.

The School of Technology and Architecture and the University Institute of Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

[This page has been intentionally left blank]

[This page has been intentionally left blank]

ACKNOWLEDGEMENTS

First of all, I would like to thank Professors Rui Marinheiro and Fernando Brito e Abreu, my supervisor and co-supervisor, respectively, for their constant guidance, prompt availability, dedication, commitment, contagious motivation, and for all the knowledge they passed on throughout this dissertation.

A big thank you to engineer Filipe Martins and the rest of the Iscte's Buildings and Resources Unit team, especially Mauro and Vitor, for their support and help with the installation at the Iscte's campus. I would also like to thank Professor Bruno Mataloto for his collaboration and helping on this dissertation.

Thanks to the Parque da Pena's team, especially engineer Pedro Trocado, for all their support, collaboration and commitment to the installation at the National Palace of Pena.

My research was sponsored by a scholarship offered in the scope of the RESETTING project, funded by the European COSME Programme (EISMEA), under grant agreement COS-TOURINN 101038190.

The cloud computing infrastructure (computing and storage) used in this dissertation was provided by the INCD - National Distributed Computing Infrastructure, funded by FCT and FEDER under project 01/SAICT/2016 n°022153.

I would also like to thank IT - Institute of Telecommunications and ISTAR - Information Sciences and Technologies and Architecture Research, the two research departments that provided all the support and conditions for this dissertation.

I would also like to thank my colleagues Lucas Oliveira, João Polónio, Afonso Alves and Gonçalo Morgado for their help, support and constant cheering during this dissertation.

A special thanks to my girlfriend, for her great help, support, encouragement, and dedication during this dissertation, and for always helping me when I needed it and for always believing in my capabilities during this dissertation.

I also would like to thank my friends for their constant cheering and support throughout this dissertation. A special thanks to my dear and old friend Ricardo Moura for all the support.

Last but not least, I would like to eternally thank my family for their encouragement, patience, dedication and help whenever it was needed. In particular, I would like to thank my parents for always being by my side with unconditional support, always ready to help no matter what was needed, and for always believing in me. I would also like to thank my grandparents, for their constant affection, appreciation, and support throughout this dissertation.

Lisbon, October, 2023

Tomás Miguel Mestre dos Santos

[This page has been intentionally left blank]

ABSTRACT

There has been an increasing impact of tourism activities at popular destinations in recent years, leading to a phenomenon mostly called *overtourism*. This dissertation contributes to assess this problem by proposing a Smart Tourism Toolkit (STToolkit) for crowd-monitoring solutions.

The STToolkit uses a sophisticated, flexible, low-cost and scalable crowding monitoring architecture composed by crowding sensors. Sensors may use multiple uplink options that mitigate the network limitations at the installation location of sensors. The number of people in sensor's vicinity is detected in real-time, by counting the number of mobile devices in the same area, capturing trace elements generated by the normal usage of mobile devices in Wi-Fi. Challenges, like MAC address randomization, are addressed using fingerprinting techniques to uniquely identify devices.

For validation, field experiments were conducted at Iscte's campus to test the toolkit architecture. The fingerprinting technique was validated using a public available dataset and a dataset collected at Iscte. Finally, overcrowding sensors were also deployed at the National Palace of Pena to validate the final detection algorithm. The collected data is stored in a time-series database, and a data visualization platform is used to render crowding information.

Overall, this dissertation concludes that the STToolkit can monitor the occupation of multiple locations in real-time, allowing space managers to perceive crowding tendencies and highly-populated events with effectiveness. With the provided supported material, the STToolkit emerges as a easily deployable and powerful tool to scaffold overcrowding management.

Keywords: Wi-Fi monitoring; MAC address randomization; Crowd sensor; Overtourism; Crowd detection.

[This page has been intentionally left blank]

RESUMO

Nos últimos anos tem-se verificado um impacto crescente das actividades turísticas em destinos populares, num fenómeno designado por *overtourism*. Esta dissertação contribui para endereçar este problema ao propor um Smart Tourism Toolkit (STToolkit) para soluções de monitorização de apinhamentos.

O STToolkit utiliza uma arquitetura de monitorização de crowding sofisticada, flexível, de baixo-custo e escalável, composta por sensores de apinhamentos. Estes podem utilizar várias opções de uplink que mitigam as limitações da rede no seu local de instalação. O número de pessoas na proximidade do sensor é detectado em tempo-real, pela contagem do número de dispositivos móveis, capturando elementos rastreáveis gerados pela utilização normal de dispositivos móveis em Wi-Fi. Desafios como a randomização dos endereço MAC são endereçados utilizando técnicas de *fingerprinting* para identificar unicamente os dispositivos.

Para validação, foram realizadas experiências no campus do Iscte para testar a arquitetura do STToolkit. A técnica de *fingerprinting* foi validada utilizando um dataset público e outro recolhido no Iscte. Finalmente, foram instalados sensores de apinhamentos no Palácio Nacional da Pena para validar o algoritmo de deteção. Os dados recolhidos são armazenados numa base de dados de séries temporais, e uma plataforma de visualização é utilizada para renderizar a informação.

No geral, esta dissertação conclui que o STToolkit pode monitorizar a ocupação de múltiplos locais em tempo-real, permitindo aos gestores de espaços perceber tendências de apinhamentos e eventos altamente-populados. Com o material de apoio fornecido, o STToolkit surge como uma ferramenta facilmente instalável e poderosa para apoiar a gestão de apinhamentos.

Palavras-chave: Monitorização Wi-Fi; Randomização de endereços MAC; Sensor de apinhamento; Overtourism; Deteção de apinhamentos.

[This page has been intentionally left blank]

CONTENTS

List of Figures	xvii
List of Tables	xix
Listings	xxi
Acronyms	xxiii
1 Introduction	1
1.1 Context	3
1.2 Motivation	4
1.3 Goals	6
1.4 Research Questions	6
1.5 Research Method	6
1.6 Contributions	7
1.7 Dissertation Organization	8
2 Literature Review	9
2.1 Crowd detection approaches overview	11
2.2 Rapid Review	14
2.3 Background	15
2.3.1 MAC Address Structure, Probe Request frames and Information Elements	15
2.3.2 MAC Address Randomization	17
2.4 Crowd counting using wireless technologies	18
2.4.1 Probe Requests approach	18
3 Toolkit Architecture	23
3.1 Proposed Toolkit Architecture	25
3.2 Crowding Data Collection	27
3.2.1 Algorithm for Wi-Fi detection	28
3.3 Communication & Services	31
3.4 Visualization & Notification	33
4 Toolkit Implementation	35
4.1 Operating System	38
4.2 Local database	38

4.3	Wi-Fi detection	40
4.3.1	Hardware selection	40
4.3.2	Software selection	43
4.3.3	Information for device fingerprinting	45
4.4	Data Upload, Data Management and Tasks Automation	47
4.4.1	Data Upload	47
4.4.2	Data Management	49
4.4.3	Tasks Automation	50
4.5	Data Ingestion and Visualization	50
4.5.1	Data Ingestion	51
4.5.2	Data Visualization and Notification	55
4.6	Toolkit Prototype	57
5	Field Experiments	61
5.1	Crowding patterns perception	63
5.1.1	Deployment description	64
5.1.2	Data visualization	66
5.1.3	First Stage	69
5.1.4	Final Stage	72
5.2	Fingerprinting validation	75
5.2.1	Validation from a public dataset	76
5.2.2	Validation from a dataset collection at Iscte	79
5.3	Crowd counting approach validation	82
5.3.1	Deployment description	83
5.3.2	Data visualization	86
5.3.3	Coverage tests	87
5.3.4	Detections validation	91
6	Conclusions and Future Work	95
6.1	Conclusions	97
6.2	Future Work	99
	Bibliography	101
A	STToolkit Cloud Server Installation Manual	105
B	STToolkit Crowd Sensor Installation Manual	127
C	STToolkit Operation Manual	149
D	STToolkit User Manual	161

LIST OF FIGURES

1.1	Worldwide evolution of tourism arrivals, based on the World Bank’s data.	3
1.2	DSRM Process Model [27]	7
2.1	Qualitative comparison for crowd counting approaches.	14
2.2	MAC address structure. The seventh bit of the first byte defines the difference between a <i>real</i> and a <i>virtual</i> MAC address (based on [15]).	15
2.3	Probe Request frame (based on [15]).	16
2.4	Directed (top) and Undirected (down) Probe Requests.	17
3.1	Crowding STToolkit architecture.	27
3.2	Algorithm for Wi-Fi detection.	29
3.3	Helium hotspots (left) and The Things Network gateways (right) for grating LoRa network coverage in areas with low coverage.	33
4.1	Deployment diagram of the crowding STToolkit.	37
4.2	Sensor’s local database schema.	39
4.3	Example of device records stored in the sensor’s local database.	40
4.4	Helium network coverage in Europe. The green areas represent the Helium network coverage.	48
4.5	Helium network coverage in the cities of: a) Lisbon, b) Barcelona, c) Tirana, d) San Benedetto del Tronto. Each green hexagon represents an area with Helium network coverage.	49
4.6	Example of data stored in a SQL database (top) and in an InfluxDB database (bottom).	52
4.7	MQTT Integration to forward the crowding measurements from the Helium network to the cloud server.	54
4.8	Examples of <i>numdetections</i> measurements (top) and <i>sensorLocation</i> measurements (down) sent by sensors.	55
4.9	Example of a notification policy that can be created in Grafana for alerting users of events.	56
4.10	Third-party integration implemented with Unity, by using walking avatars on top of a BIM of an entire floor of Iscte.	57
4.11	Sensor cases. a) Larger version with no exposed antennas. b) Smaller version with exposed antennas.	58

5.1	Deployment of sensors at: a) a large study room; b) the university library; c) an indoor passageway connecting the two main buildings of the campus; d) an outdoor passageway that connects two courtyards of the campus.	65
5.2	Dashboard with a collapsed view of each sensor deployed at Iscte's Campus in a two-week period. The working days are filled in blue, and the weekends are filled in orange.	66
5.3	Dashboard for comparing crowding between locations at Iscte's Campus.	67
5.4	Dashboard with crowding geo-distribution at Iscte's Campus.	67
5.5	Final dashboard created for crowding data visualization of Iscte's Campus.	68
5.6	Number of devices detected in a four-day period at the first stage of this field experiment at: a) study room; b) university library; c) indoor passageway; d) outdoor passageway. The working days are filled in blue and a Saturday is filled in orange. The same four-day period is presented for the first three scenarios, while a different period is shown for the fourth scenario.	70
5.7	Iscte's graduation event.	71
5.8	Number of devices detected in a four-day period at the final stage of this field experiment at: a) study room; b) university library; c) indoor passageway; d) outdoor passageway. The working days are filled in blue and a Saturday is filled in orange.	73
5.9	Probe requests dataset collection at the Iscte's garage.	79
5.10	Pena Palace.	83
5.11	Pena Park that surrounds Pena Palace (red), and Picadeiro (orange), a public square of Pena Park.	84
5.12	Picadeiro area and tourists' main path to the Pena Palace. Each red line represents an entrance and exit point of Picadeiro.	84
5.13	Deployment of sensors in Picadeiro's Kiosk (left) and inside a Help-point (right).	85
5.14	Customized dashboard created for crowding data visualization of the Pena Park.	86
5.15	Dashboard created for the coverage tests at the Pena Park.	88
5.16	Coverage results for the Kiosk (top) and Help-point (down) sensors. Each heatmap circle represents a location with detection coverage, accompanied by the mode power received. The locations with a black dot and a zero are locations with no sensor coverage.	89
5.17	Demonstration of the Portuguese School of the Equestrian Art event at Picadeiro.	91
5.18	Results from validation at the Picadeiro. The event time is filled in orange.	92

LIST OF TABLES

2.1	Primary studies classification regarding the MAC address randomization.	22
3.1	Probe Request’s Information Elements used to create device fingerprint.	31
4.1	Wi-Fi cards available for detecting devices.	41
4.2	Directional antennas available for detecting devices in only specific directions. . .	42
4.3	Software programs available for detecting devices through Wi-Fi detection.	43
4.4	Information of each Information Element considered for device fingerprinting. . .	47
4.5	Tasks simultaneously scheduled in the STToolkit sensors.	50
4.6	Hardware available for the STToolkit sensors (unit costs from 4 September 2023).	58
4.7	Sensor cases dimensions and approximated unit cost (costs from 4 September 2023).	59
4.8	Helium communication costs per sensor (costs from 4 September 2023).	59
4.9	Cloud Server Hosting Pricing for different providers (costs from 4 September 2023).	60
4.10	STToolkit total installation costs simulation.	60
5.1	Selected hardware and upload option for the sensors deployed at Iscte’s Campus.	64
5.2	Main results obtained in the different deployment scenarios at Iscte’s Campus. . .	75
5.3	Devices used on the public dataset (based on [28]).	76
5.4	Device fingerprints for the devices of the public dataset. The fingerprints high- lighted in orange are fingerprints that have been used to identify more than one device.	78
5.5	Devices used for the dataset collection at the Iscte’s garage.	80
5.6	Dataset collection results at the Iscte’s garage.	81
5.7	Device fingerprints for the dataset collection at the Iscte’s garage. The fingerprints highlighted in orange are fingerprints that have been used to identify more than one device.	81
5.8	Selected hardware and upload option for the sensors deployed at Pena Park. . . .	86
5.9	Packet power filtration thresholds applied to each sensor.	90
5.10	Correlation coefficients between sensor detections and the real number of people.	92
5.11	Temporal similarity between sensor detections and the real number of people. . .	93

[This page has been intentionally left blank]

LISTINGS

[This page has been intentionally left blank]

ACRONYMS

- AP** Access Point.
- APIs** Application Programming Interfaces.
- BIM** Building Information Model.
- CCTV** Close-Circuit TeleVision.
- DBMS** DataBase Management System.
- DC** Data Credit.
- DS** Device Status.
- DSRM** Design Science Research Methodology.
- DTW** Dynamic Time Warping.
- EVOA** Tagus Estuary Birdwatching and Conservation Area.
- GDPR** General Data Protection Regulation.
- HNT** Helium Network Token.
- HTTP** HyperText Transfer Protocol.
- IE** Information Element.
- IEAA** *Information Elements Automatic Analyser.*
- IEEE** Institute of Electrical and Electronics Engineers.
- IFTA** Inter-Frame Time Arrival.
- IoT** Internet of Things.
- IP** Internet Protocol.
- LoRaWAN** Long Range Wide Area Network.
- LTE** Long-Term Evolution.
- MAC** Media Access Control.
- MQTT** Message Queuing Telemetry Transport.

NIC Network Interface Controller.

OS Operating System.

OUI Organizational Unique Identifier.

PNL Preferred Network List.

PoIs Points of Interest.

PTZ Pan-Tilt-Zoom.

RESETTING Relaunching European smart and Sustainable Tourism models Through digitalization and INnovative technoloGies.

REST API Representational State Transfer Application Programming Interface.

RP-SMA Reverse Polarity - SubMiniature version A.

RSSI Received Signal Strength Indicator.

SCBs Single Computer Boards.

SEQ SEQUENCE numbers.

SHA Secure Hash Algorithm.

SIM Subscriber Identity Module.

SMEs Small and Medium-sized Enterprises.

SQL Structured Query Language.

SSID Service Set Identifier.

SSL Semi-Supervised Learning.

STToolkit Smart Tourism Toolkit.

STTs Smart Tourism Tools.

UE User Equipment.

UI User Interface.

VPN Virtual Private Network.

VPS Virtual Private Server.

CHAPTER 1. ■

INTRODUCTION

Contents

1.1	Context	3
1.2	Motivation	4
1.3	Goals	6
1.4	Research Questions	6
1.5	Research Method	6
1.6	Contributions	7
1.7	Dissertation Organization	8

This chapter presents the context, motivation, and problems addressed in this dissertation. In addition, the research questions and method are also presented, as well as the contributions and organization of this dissertation.

This chapter is organized as follows: section 1.1 presents the context of this dissertation, section 1.2 describes the motivation, relevance, and problems of this dissertation, and section 1.3 presents the goals of this dissertation. Then, sections 1.4 and 1.5 present the research questions and research method, respectively. Section 1.6 describes the contributions to this dissertation and section 1.7 presents the dissertation organization.

[This page has been intentionally left blank]

Chapter 1

Introduction

1.1 Context

The tourism sector has been growing steadily over the last decades. Just in Europe, with many of the world's most popular destinations, almost 750 million tourist arrivals were registered in 2019, according to [Statista](#). Also, if the pre-pandemic trend is achieved from 2023 onward, tourism will reach 3 billion arrivals by 2027, based on the [World Bank development indicators](#), as shown in Figure 1.1.

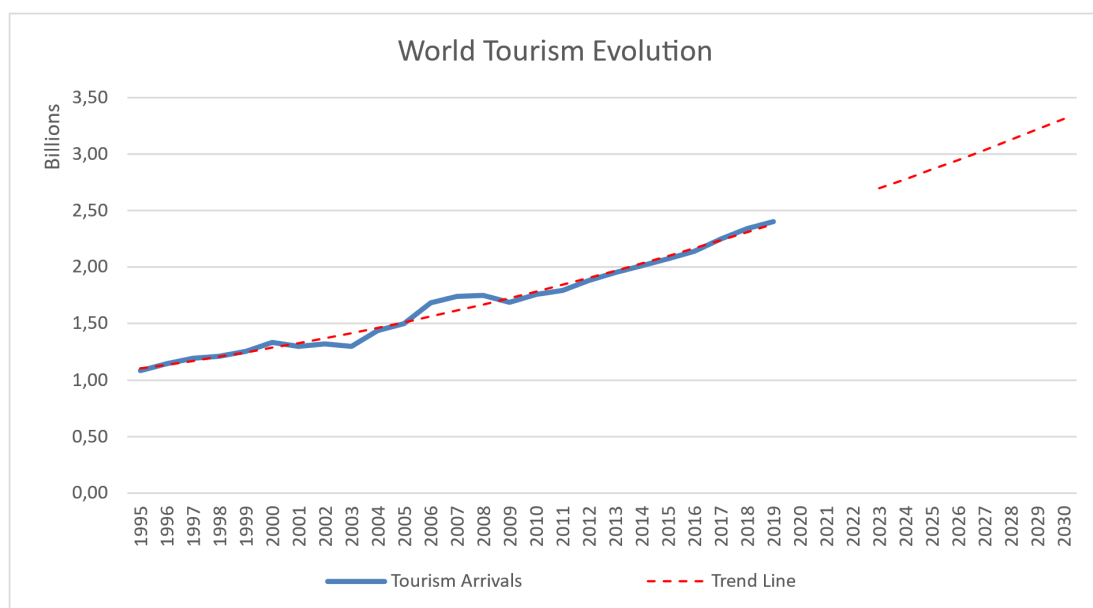


Figure 1.1: Worldwide evolution of tourism arrivals, based on the World Bank's data.

As a consequence, the impact of tourist activities in popular destinations has risen significantly over the years, often fostered by the increasing number of tourists visiting these destinations and the proliferation of cheaper local accommodations [9]. That increase led to an exceed of carrying capacity in those destinations, in a phenomenon mostly called *tourism overcrowding*, or simply *overtourism* [26].

Overtourism deteriorates not only the quality of life of stakeholders as well as the environment. It degrades the visitor's quality of experience, reducing their feeling of safety, making it more difficult to move around, enjoy the attractions, and use basic services, such as transportation and restoration, due to long wait times, while reducing the authenticity from the perspective of tourists [31]. Also, *overtourism* deteriorates the quality of life of local residents. The proliferation of cheaper local accommodations brings less privacy and personal space concerns since tourists and local residents coexist in the same space, increase in urban noise and less effective urban cleaning, higher prices for basic goods and services (as businesses seek to capitalize on the increased demand), and cultural clashes when visitors fail to respect local customs, traditions, and privacy, sometimes leading to the former expressing negatively against the

latter [3]. Last, but not least, the structures and cultural heritage of these destinations are also impacted by these over-occupation situations, which local authorities and heritage managers fight to prevent, resulting in loss of authenticity [29].

To give an answer to the above-mentioned problems, the [Relaunching European smart and Sustainable Tourism models Through digitalization and INnovative technoloGies \(RESETTING\)](#) project, funded by the [European COSME Programme](#), in which the scope of this dissertation was developed, is focused on customizing, productizing, evaluating, and deploying innovative digitally-driven [Smart Tourism Tools \(STTs\)](#) to improve tourism. That improvement may express itself in diverse ways such as by enhancing tourism experiences, improving the efficiency of resource management, ensuring that benefits are well distributed and/or maximizing destination competitiveness with an emphasis on sustainable aspects.

In particular, the RESETTING project aims at customizing, productizing and deploying STTs to facilitate the conception of tourism crowding monitoring applications.

1.2 Motivation

The *overtourism* problem has been increasing over the last decades and needs to be addressed in a serious and conscious manner.

Overtourism creates not only an anti-tourism feeling for tourists as well as a harmful feeling to local residents, as they do not feel safe, comfortable, and pleasantly at home. Besides Portugal, that had a massive increase in the number of arrivals in tourist accommodation in last years, according to [Statista](#), and more emphasized in its capital, Lisbon, in recognition of the honors and accolades received for the growing notoriety as a tourist destination, *overtourism* was already felt in other parts of Europe, such as [Venice and Barcelona](#), where there have been instances of marches, public rallies and even cases of violence as anti-tourism manifests due to this problem. Furthermore, *overtourism* has already been extended to other European cities, more precisely to [Amsterdam, Florence, and Prague](#), where protests from local residents forced city hall officials to take action.

Mitigating *overtourism* benefits all stakeholders:

- **Local residents** - with reduced stress from over-occupation of personal space and privacy, as well as improved attitude towards tourists and tourism professionals;
- **Tourism specialists** - which can speed up service delivery and quality of service;
- **Tourists** - with increased visit satisfaction, with fewer delays, and can provide more guarantees of safety and cleanliness;
- **Local authorities** - with improved services by making just-in-time decisions and planning more effectively urban cleaning and public safety routines, as well as reducing operating costs;
- **Heritage managers** - which can elude heritage degradation more effectively, thus retaining the authenticity of destinations;
- **Local businesses** - with increased share of tourism income.

In this dissertation, and also according to the goals of the RESETTING project, there is the motivation to develop a [Smart Tourism Toolkit \(STToolkit\)](#) to monitor the occupation of spaces and to trigger or access the effectiveness of overcrowding mitigation actions. This kind of toolkit is especially relevant for tourism SMEs that deal with large (and potentially excessive) numbers of people. Either in cultural or religious indoor tourism scenarios such as palaces, museums, monasteries or cathedrals, or even in outdoor scenarios like public parks, fireworks, music festivals or areas with beautiful landscapes and monuments, it is crucial to monitor the occupation of spaces, either to ensure a better visiting experience to visitors (e.g. to forward crowds to less-occupied, but equally attractive areas), for security reasons (e.g. to ensure that the carrying capacity set for an event is not surpassed, or to avoid tampering with exposed objects in a museum space), health reasons (e.g. to avoid exceeding the maximum density of people allowed by health authorities in an event space to prevent contagion in pandemic times), or simply for purposes of workforce and resource management considerations (e.g. to calculate the expected time for evacuating a concert arena to allow cleaning teams to proceed).

There are several approaches to monitoring crowds, one of which is by performing wireless spectrum analysis, characterized by exploring protocol characteristics and small information breaches on wireless technologies, such as Wi-Fi or Bluetooth. Our crowding sensors perform real-time detection of trace elements of mobile devices based on their wireless activity, namely in Wi-Fi technology. By performing this detection, a crowding sensor is capable of detecting the number of devices in its proximity, thus measuring how crowded the surrounding location is.

This dissertation also comes as an iteration from the previous work developed by [9], which was the initial reference to this dissertation. The latter proposed a crowding sensor and exploited several wireless technologies to monitor the occupation of spaces. The current work improved that pioneer work by addressing recent issues due to [Media Access Control \(MAC\)](#) address randomization, as well as the design of a flexible, low-cost, and scalable system architecture to monitor the occupation of spaces, including the provision of multiple communication protocols for uploading information, namely Wi-Fi and LoRaWAN, in order to mitigate network limitations at the installation location, something that is often disregarded by other alternative approaches.

Furthermore, this work assumes a correlation between the number of mobile devices and the real number of people present in an area, a realistic assumption in touristic scenarios aimed by the STToolkit, since tourists usually carry their mobile phones to take pictures and record videos during visits.

The STToolkit has been developed in the scope of the [European RESETTING project](#) and is aimed at tourism [Small and Medium-sized Enterprises \(SMEs\)](#) in order to improve their efficiency, sustainability, quality of service, and also enhancing tourism experience. For this context, support material is also provided to build a STToolkit, either in indoor or outdoor appliances, in an easy and non-effortlessly manner. The latter includes installation, operation, and user manuals, online video tutorials, setup images, and cost calculators to provide guidance on how to install, configure, operate, and use the STToolkit.

1.3 Goals

The main goal of this dissertation is a Smart Tourism Toolkit (STToolkit) for crowding monitoring solutions to monitor the occupation of spaces in real time. The latter is especially relevant to trigger or assess the effectiveness of overcrowding mitigation actions regarding the *overtourism* problem.

Within this main goal, there are subgoals for its achievement. To monitor the occupation of spaces, a crowding sensor and system architecture are aimed to be developed to measure how crowded several locations are in real-time. The crowding sensors detect the number of mobile devices present in their proximity, thus measuring how crowded the surrounding area is. The latter is achieved by performing a real-time detection of the wireless activity of mobile devices in Wi-Fi technology. By performing such an approach, this dissertation aims to tackle the MAC address randomization process of mobile devices, as it is one of the main drawbacks for accurately performing crowd counting based on the wireless activity of mobile devices.

1.4 Research Questions

The aim of this dissertation is to provide answers to the research questions shown below. The first research question (RQ1) is the main research question to be answered in this dissertation. The remaining research questions (RQ2, RQ3, and RQ4) are research questions implied by the crowd-counting approach adopted in this dissertation.

- RQ1 - What system architecture should be used to perform crowd detection in real-time as a low-cost, scalable, and easily installable solution?
- RQ2 - What approaches may be used to uniquely identify mobile devices when using Wi-Fi technology, in order to mitigate their IDs randomization currently used in communication?
- RQ3 - Is it possible to perceive crowding tendencies and highly-populated events by capturing the wireless activity of mobile devices?
- RQ4 - To what extent is it possible to generate unique fingerprints of each mobile device, despite the randomization of their IDs?

1.5 Research Method

This dissertation followed the [Design Science Research Methodology \(DSRM\)](#) [27] Process Model, presented in Figure 1.2.

The first stage of this methodology starts with a “problem-centred approach” since the problem that this dissertation tries to solve was already defined. In this case, the context, motivation, as well as goals, and problems intended to be mitigated are presented in this chapter. As so, the second step related to the “Objectives of the Solution” of this methodology is also presented in this chapter. Also, the problems regarding this dissertation are reasoned in Chapter 2 of this dissertation.

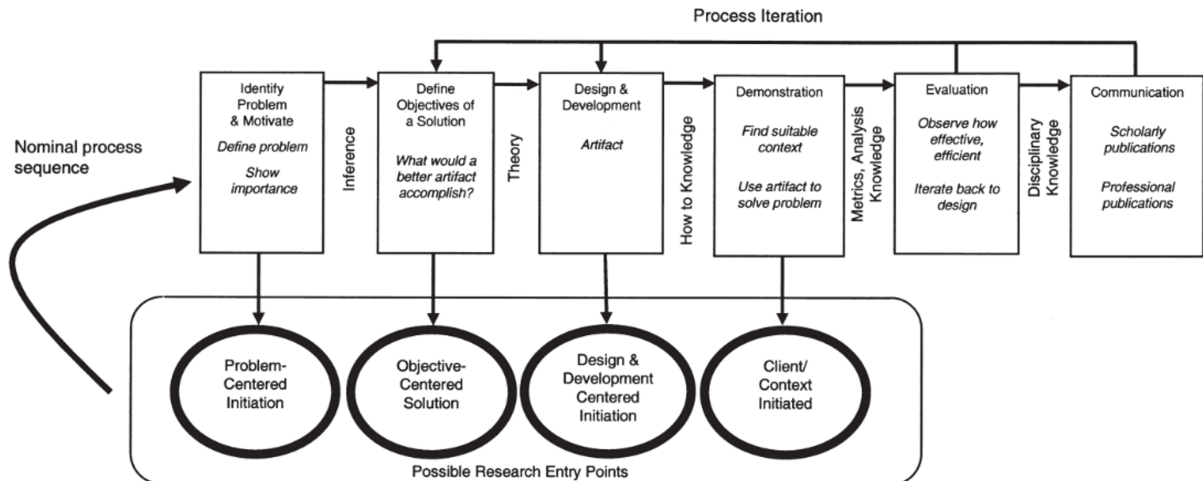


Figure 1.2: DSRM Process Model [27]

The third step concerns the “Design and Development” of the artefact. In this case, it involves the development of a better overcrowding sensor, which is the basis of the crowding STToolkit. In Chapter 3 the architecture of the STToolkit is presented and in Chapter 4 the implementation of the STToolkit is presented.

The fourth step regards the “Evaluation”, where a testing and validation phase of the artefact is performed and is presented in Chapter 5. For testing and validation, three field experiments were carried out, two of them in real crowded scenarios: 1) 24/7 real-time detection in several spots of Iscte’s Campus, shown in section 5.1, to validate the STToolkit architecture and to test the effectiveness on perceiving the crowding phenomena; 2) validation of the fingerprinting technique to uniquely identify devices based on a public dataset and based on a dataset collection of mobile device messages at the Iscte’s garage, presented in section 5.2; 3) 24/7 real-time detection in several spots of Pena Palace’s Park, flattened by overtourism all year round, presented in section 5.3, to validate the final crowd counting approach of the STToolkit. In this stage, the results obtained in the previous methodology step are compared. Depending on the results obtained regarding the precision of our STToolkit for counting the correct number of devices in a given area, a new iteration of “Design and Development” can be instantiated to improve our solution.

Finally, regarding the “Communication” final step of this methodology, conclusions and future work from this dissertation are drawn. This step is presented in Chapter 6 of this dissertation.

1.6 Contributions

The main contributions of this dissertation are:

1. An innovative approach to detect crowding levels in a given area, with anonymity to the collected information by the application of a hashing function, addresses the MAC

address randomization process of mobile devices by the application of a fingerprinting technique.

2. A low-cost, flexible, and scalable monitoring system for detecting crowding at several locations in real-time or quasi-real-time.
3. An overview of the information that devices leak and how that information can be used to monitor crowding levels.
4. The application of the STToolkit in real crowded scenarios, with relevant impact for their mitigation.
5. The development of an academic poster and demo of the preliminary results in International Workshop of RESETTING and Poster exhibition [23].
6. The submission and acceptance of a book chapter where the preliminary results were presented [22].
7. The presentation of preliminary results on a workshop collocated with CAiSE '23 international conference [21].
8. The production of support material to build, operate, and use the STToolkit. The latter includes:
 - The production of the installation, configuration, operation and user manuals for the STToolkit, namely:
 - STToolkit Cloud Server Installation Manual, presented in Appendix A;
 - STToolkit Crowd Sensor Installation Manual, presented in Appendix B;
 - STToolkit Operation Manual, presented in Appendix C;
 - STToolkit User Manual, presented in Appendix D;
 - The production of online video tutorials of how to install and configure the STToolkit, publicly available on the [RESETTING Youtube channel](#), namely:
 - [RESETTING - STToolkit Cloud Server Installation - Part 1](#);
 - [RESETTING - STToolkit Cloud Server Installation - Part 2](#);
 - [RESETTING - STToolkit Crowd Sensor Installation](#);
 - [RESETTING - STToolkit Crowd Sensor Mounting in 5 minutes](#);
 - The source code availability of the STToolkit in the [RESETTING GitHub repository](#).

1.7 Dissertation Organization

This section presents the organization of this dissertation with a brief explanation of each chapter. Chapter 2 presents a state-of-the-art review of crowd-detection techniques. Chapter 3 presents the system architecture of the STToolkit and Chapter 4 presents the implementation of the STToolkit. Chapter 5 presents the field experiments carried out to test and validate the STToolkit. Finally, Chapter 6 draws conclusions and future work.

CHAPTER
2.

LITERATURE REVIEW

Contents

2.1	Crowd detection approaches overview	11
2.2	Rapid Review	14
2.3	Background	15
2.4	Crowd counting using wireless technologies	18

This chapter presents the state of the art of crowding detection and how it is possible to detect devices in the wireless technologies that we are concerned about, namely Wi-Fi and Bluetooth.

This chapter is organized as follows: section 2.1 presents an overview of the different approaches previously done for detecting crowds and section 2.2 describes the literature review method applied in this dissertation. Then, section 2.3 gives a brief background of some information that devices leak that are important to understand. Finally, section 2.4 presents works previously done regarding crowd detection using wireless technologies with several approaches to tackle the MAC address randomization, one of the challenges addressed in this dissertation.

[This page has been intentionally left blank]

Chapter 2

Literature Review

2.1 Crowd detection approaches overview

This section presents an overview of the different approaches previously done for crowd detection. Earlier works [10, 30] proposed several approaches in order to detect the number of persons in a given area, which can be arranged into five different approaches, being them: sound-based, using social media content, using mobile operator's data, vision-based, and wireless spectrum analysis.

Sound-based approach:

This approach consists of a set of several acoustic sensors dispersed in an area capturing sounds, that can be further processed and analyzed by a cloud server, which can then retrieve crowd density and movement over time. Previous works concerning this method can be found in [1] and [20]. In addition, and apart from a network of sensors distributed in small-scale spaces, microphones and speakers embedded in people's smartphones can also be used for crowd monitoring, as [16] shows in his work, when using audio tones emitted from smartphones at non-audible frequencies, however, it requires the installation of a mobile application for this purpose in each smartphone.

In general, the precision and detection ranges achieved by this approach are low, as well as the sound analysis requires a high processing and network bandwidth and, in some cases, can raise privacy issues since conversations could be listened to and recorded [9].

Social networks approach:

This approach relies on content that is published on social media networks, such as Instagram or Twitter, and on the geo-location information associated with this information. In this approach, huge datasets of social media content are gathered, and, after some data cleaning, processing, and analysis, information can be used to infer several patterns, such as people's flow in a city over time, [Points of Interest \(PoIs\)](#), urban hotspots, delimit commercial areas and detect traffic events. In research from [36], geo-tagged social information from Twitter was used to detect traffic events. Another interesting study from [11] used the same approach from both Instagram and Twitter social media geo-referenced information to provide crowd characterization in large-scale events. The method was applied in two city-scale events, namely Sail 2015 and King's Day 2016, two popular events in Amsterdam, Netherlands, where several aspects such as crowd temporal distribution, PoIs preferences, and social media post locations were analyzed. Another study from [14] used [Weibo](#), a popular social network in China, as a data source for detecting urban hotspots and delimiting the boundaries of dynamic commercial

areas, which can change over time. The geo-social information from Weibo social network was also used in conjunction with video-surveillance content to detect city events [37].

The large amount of data that is daily published on social media networks can be used to extract several metrics and patterns. However, this approach fully depends on the usage of these platforms by its users and the fact that the geo-location information needs to be tagged in people's posts. Moreover, only a portion of this information can be accessed and utilized for this purpose since it needs to be public. As a result, the outcomes of this approach could be less representative of the actual crowding level in an area [9].

Mobile data approach:

Mobile operators have access to massive volumes of data about network usage. In the past, this kind of information was difficult to obtain, but it has now been increasingly made publicly available by telecom operators for research purposes. A study from [8] used call records provided by the Semantics and Knowledge Innovation Lab of [Telecom Italia Mobile](#), one of the largest mobile operators in Italy, to extract human activity in six Italian cities, together with socio-demographic information from Italian census and [OpenStreetMap](#) to acknowledge urban activity and diversity in these cities. Another research from [25] has also invested in past data processing in order to predict future movement patterns. [Vodafone Analytics](#) is another well-known example of this data exploration, providing a collection of measurements as well as their change through time, which are often provided on geo-referenced maps, allowing for spatial-temporal perception [9].

All these methods focus on retrieving patterns and predictions from past gathered data. As so, the main disadvantage of this approach is in generating relevant metrics from those massive data sets in real-time, since overcrowding situations are quick and unpredictable and, for that reason, require a just-in-time response. Furthermore, mobile operators are usually willing to monetize their network data, which may render this approach unfeasible for SMEs.

Vision-based approach:

This approach consists of detecting the number of persons and their behaviour by processing images or video sequences. In this approach, a network of cameras is used to provide visual information about a location within its range. The images are processed, usually with the aid of learning techniques, to detect and monitor crowds. Several types of cameras can be used for this purpose, such as [Close-Circuit TeleVision \(CCTV\)](#) cameras to provide visual information about a location, multi-view, and [Pan-Tilt-Zoom \(PTZ\)](#) cameras to provide multi-angular visual information for better tracking and space recognition, or even thermal cameras. One of the main challenges of this approach is related to the complexity of the captured scenes with many people occluding one another, leading to false positives in large and dense crowds. The research from [19] addressed this by obtaining people's feet position using multi-view cameras to combine information from different viewpoints of the same scene. For greater results, this problem can also be addressed using depth cameras for generating 3D models of a scene [17]. In this work, a stereo thermal camera setup was utilized for pedestrian counting. The two thermal

cameras gathered information from a public event with a moderate density of pedestrians and heavy occlusion, and an algorithm based on clustering and tracking of the generated 3D points gives the number of pedestrians in the captured scene. Another recent and interesting work from [18] also uses a multiple-camera setup for pedestrian detection, leveraging automatically extracted scene context, which does not require *ad-hoc* training with labelled data.

Any of the preceding alternatives have significant drawbacks for detecting overcrowding situations in real time. Processing images and videos is such a computationally demanding task that is not affordable for an edge computing approach [9]. Implementation costs may be significant since the installation of a sensor node of this type (expensive cameras) does not come at feasible prices, which makes this approach not very scalable. Also, communications costs and bandwidth can be considerable, due to the stream of frames that needs to be constantly transmitted to the cloud server for processing. Finally, an additional complication arises from the fact that images of persons are being captured and can be used for face recognition and identification, raising all kinds of privacy, financial, and logistic issues.

Wireless spectrum analysis:

Technological advancements in the last decade, with greater supply and improvements of open-source software and corresponding hardware, have simplified spectrum and communication protocol analysis and have accelerated research on passive sniffing and active sniffing approaches [9]. These methods are characterized by exploring protocol characteristics and small information breaches. As so, the research has been gradually redirected from a vision-based approach (image and video capturing), and tending to more privacy-perserving techniques with minimum public participation, in particular, towards sensor-based crowd monitoring systems [30] through the use of these methods.

Also, as [30] mentions in his work, we must develop less-intrusive and privacy-preserving crowd-monitoring techniques, fostered by recent restrictive guidelines such as the [General Data Protection Regulation \(GDPR\)](#). Such a thing can be achieved in sensor-based crowd monitoring systems, for instance, using [Secure Hash Algorithm \(SHA\)](#) encryption on the gathered information by the sensors, grating data anonymization.

The work made by [38] gives plenty of tutorials and in-depth comprehension of several technologies, some of them already proposed for crowd detection.

Also, it is important to mention that we have discarded Mobile Network detection from the previous work from [9], namely 3G and 4G [Long-Term Evolution \(LTE\)](#) detection, because it requires an active intrusive approach that infringes privacy and legal concerns. Nevertheless, Wi-Fi and Bluetooth technologies are still both valid for the goals of this dissertation.

Figure 2.1 summarizes the approaches found in the literature and compares them in terms of range, precision, time delay of analysis, and the implementation cost of each approach. The comparison is based on relative approximations concerning the characteristics and limitations of the several approaches.

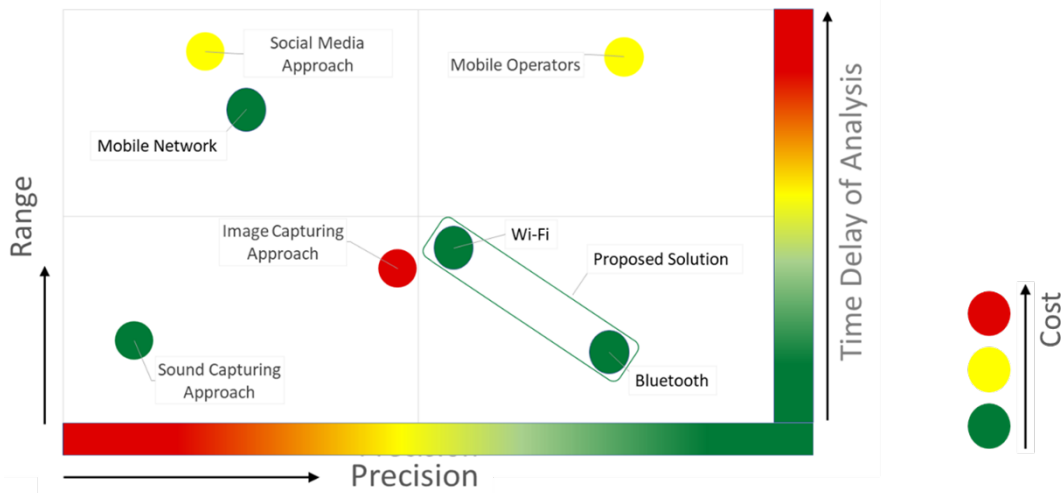


Figure 2.1: Qualitative comparison for crowd counting approaches.

2.2 Rapid Review

In order to find related works according to the aim of this dissertation, we have adopted the Rapid Review method. We have chosen the Scopus database as the only source of research and no other was considered.

In the Scopus database, the search was conducted through a query (Search string) to obtain our *Golden Set*. The latter comprehended the main and primary studies for this dissertation, which required an in-depth comprehension, and were related to the several approaches for detecting crowds using the wireless technologies that we are concerned about (Wi-Fi and Bluetooth), also tackling the MAC address randomization process from mobile devices, as it leads to overcounting estimations concerning the use of these technologies. The query was divided into three different groups: one regarding the detection technologies that we are concerned about, one regarding the MAC address randomization process, and one regarding the field of detection and some strategies for performing the detection through wireless technologies. As so, the query for our *Golden Set* that contains our primary studies is the following:

```
TITLE-ABS-KEY ( ( ("WiFi"OR "Wi-Fi"OR "Bluetooth") AND ("MAC"AND ( "Random-
ization"OR "Anonymization"))) OR ( "Crowd"AND "Counting"AND "GDPR" ) ) AND (
"Fingerprint"OR "Footprint"OR "Derandomization"OR "De-randomization"OR "Passive
sniffing"OR "Passive tracking"OR "Analysis"OR "Monitoring" ) )
```

In addition, it is also important to mention that only the study from [7], published in the 37th Twente Student Conference on IT (TScIT 37), was selected from applying backward snowballing, being the only reference from our *Golden Set* that was not obtained from the query applied to the Scopus database.

2.3 Background

The following subsections present background concepts that are key for understanding the approaches to detecting crowds using wireless technologies. Firstly, we present a brief analysis of the formats of the MAC addresses and probe requests frames, with particular emphasis on the Information Elements conveyed in these frames. Then, we illustrate the MAC address randomization process that is implemented in mobile device's operating systems.

2.3.1 MAC Address Structure, Probe Request frames and Information Elements

Each Wi-Fi device has a MAC address that allows its identification on the local network. Each MAC address consists of 48 binary bits, generally represented in hexadecimal as a sequence of six bytes. The first three bytes are assigned by the IEEE to manufacturers and are referred to as the **Organizational Unique Identifier (OUI)**. The last three bytes are assigned by the manufacturer itself to each produced network card and are referred to as the **Network Interface Controller (NIC)**. In summary, the first three bytes of a MAC address represent the device's manufacturer, and the last three bytes are assigned by the manufacturer to each device. The MAC address structure is presented in Figure 2.2 based on [15]. This format enables the identification of the MAC address manufacturer and the assignment of a unique address space to each manufacturer from which the necessary addresses for each device may be obtained [33].

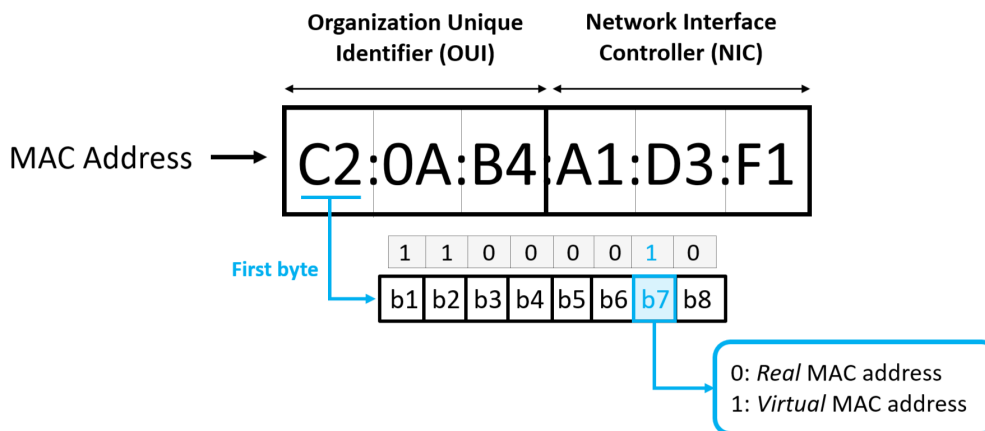


Figure 2.2: MAC address structure. The seventh bit of the first byte defines the difference between a *real* and a *virtual* MAC address (based on [15]).

It is important to mention that the seventh bit of the first byte, highlighted in Figure 2.2, plays an important role in the MAC address. If this bit is set to 0, then the MAC address is globally unique and is kept constant over time by the device; in fact, this is the *real* MAC address of the device. On the other hand, if it is set to 1, this means that the MAC address should be locally administered from the device. In this second case, the MAC address is randomly generated and may change over time and can be considered as a *virtual* MAC address [33]. This behaviour is in charge of the IEEE organization.

Devices can connect to a network by listening to the beacons regularly sent by the **Access Point (AP)**s, advertising that a network is available, however, they need to be fully active waiting to receive this message to trigger the association process (passive scanning). An alternative for

this discovery process is devices having some autonomy by sending messages called Probe Requests when their Wi-Fi interface is turned on. A device actively sends Probe Requests in order to discover the nearby networks available in its proximity (active scanning). Each message has a time restriction limit during which it must receive a response in order to connect to the chosen network. This mechanism allows better efficiency since devices only need to be awake during a specific period of time for a response to be sent, and not fully actively listening to beacons from APs, saving energy. The Probe Requests are usually sent in groups called bursts, whose length and time between each message can vary from device to device. This method is performed not only to discover available networks in proximity, allowing a quick recovery if the network is unintentionally severed and for a faster connection upon arriving at a known network range, but also for managing the connection to an AP, allowing it to remain connected to a known network if the user, for instance, moves away from the nearest AP or if there is a connectivity problem with the connected AP. Every AP that receives these messages, within the established time, will reply to the device by sending, in turn, a Probe Response frame, which contains the information needed to complete the association from the device to the AP.

Both Probe Request and Probe Response frames are two sub-types of a specific 802.11 frame type called management frames, which are divided into MAC header and Frame body, as shown in Figure 2.3. In the frame body, Probe Requests may convey some fields called [Information Element \(IE\)](#)s, whose structure is defined by the [IEEE standard 802.11-2020](#).

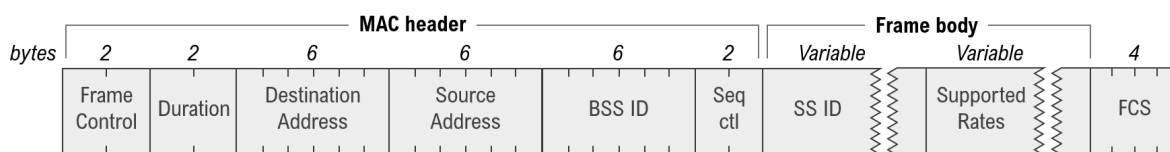


Figure 2.3: Probe Request frame (based on [15]).

Each IE has the same structure: the first byte is reserved for the IE ID, which is unique for each IE; the second byte assigns the IE's length; and the remaining bytes contain the IE Value, with the information of the IE, which size is equivalent to the length of the IE Length byte. A variable number of IEs can be transmitted in a frame. This information may contain the supported transmission rates from the device, the list of known networks, the Wi-Fi channel from which the frame was transmitted, and vendor-specific information, among other specific attributes from the device.

It is also important to mention that there are two types of Probe Requests: Directed Probe Requests and Undirected Probe Requests. The difference between these two types relies on the presence of the [Service Set Identifier \(SSID\)](#) field in the frames. In the first case, a device sends a Probe Request with a specific SSID when it tries to connect to a specific known network. Therefore, only APs matching that SSID will respond with a Probe Response. In the second case, no SSID is specified (the SSID field is present but empty), and the device is trying to find all the available networks in its proximity. As a result, all APs in range that receive this type of Probe Request will respond to the device with a Probe Response. These two cases are illustrated in Figure 2.4.

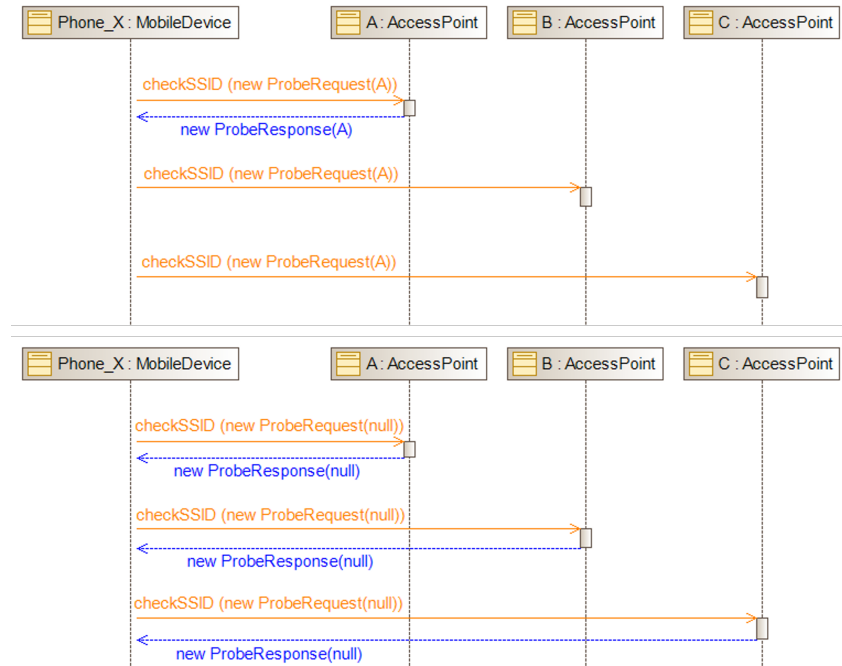


Figure 2.4: Directed (top) and Undirected (down) Probe Requests.

2.3.2 MAC Address Randomization

When a device sends Probe Request frames in order to discover nearby networks that are available to connect, these messages must provide an identification of the device so a response can be directed to it. In these messages, if devices send their real MAC address (which is kept constant, i.e. never changes) they could be easily tracked wherever it is located. As so, for user privacy and security reasons, the manufacturers of mobile devices started to hide the real MAC address on these messages using a randomly generated one (also called virtual MAC address in the following) [33]. This is done to ensure that the real MAC address from the device remains unknown so that it cannot be tracked over time, thus protecting the user's identity.

The randomization process started to be implemented on devices by 2014 and is regulated by the [Institute of Electrical and Electronics Engineers \(IEEE\)](#) standards, nevertheless, the algorithm that generates them is different for each operating system. The work by [33] compares the behaviour of several mobile device's operating systems, and differences among randomization processes on these were shown. As so, the process can be different from device to device and can vary according to different aspects, which makes it a process very complex to demystify and analyze. "Accordingly, the device uses a virtual address during the connection to a network and then switches to the real one once the connection is established, as all communication is encrypted from that point forward" [33]. However, new versions of various operating systems already do not expose their real MAC address even when they are connected to a local network (the so-called "network private MAC addresses"). This feature has been already implemented, for instance, in iOS devices.

It is extremely important to understand and acknowledge this randomization process because the majority of mobile device manufacturers already started to implement it on their

devices, and older **Operating System (OS)** versions with no MAC address randomization will soon disappear. As so, the vast majority of mobile devices will have this randomization process implemented on their OSs. As shown in the previous section, the difference between a real MAC address and a virtual MAC address consists of the 7th bit of the first byte of the MAC address. Therefore, we can simply distinguish these two types of MAC addresses by only checking this bit.

In summary, for privacy concerns, manufacturers of mobile devices started to implement MAC address randomization in their operating systems in 2014. The randomization process consists of generating a randomly assigned MAC address (virtual MAC address) conveyed in Probe Request frames that may change over time and is dependent on the manufacturer and on the operating system, which hampers the tracking of mobile devices.

It is also important to mention that this randomization process is applied for both Wi-Fi and Bluetooth technologies since each network card of each technology needs identification in the local network, which is provided by a MAC address (Wi-Fi MAC address and Bluetooth MAC address, respectively). Regarding Wi-Fi, a device may change its virtual MAC address over time in Probe Requests frames, or only change it when the Wi-Fi interface is turned on/off. Regarding Bluetooth, devices in undiscoverable mode also send Probe Requests, however, the MAC address from these messages is also randomized. As so, “all the work done in randomization of MAC addresses in Wi-Fi is also relevant for Bluetooth” [9].

2.4 Crowd counting using wireless technologies

As shown in section 2.1, the research concerning crowd monitoring and crowd counting is gradually tending from vision-based to sensor-based approaches through the use of wireless technologies. Many works have already been proposed to address the issue of MAC address randomization for crowd counting, which is specifically the same as we intend to address in this dissertation. The majority of these employ a Probe Request approach (passive sniffing approach), while others make the use of an active sniffing approach. The latter was not considered in this dissertation since it is intrusive by forcing some action on the devices.

In this section, we present previous works done that tackle the MAC address randomization issue for crowd counting. Finally, we also present a taxonomy cataloguing the different approaches mentioned, shown in Table 2.1.

2.4.1 Probe Requests approach

A robust way to perform crowd counting can be through the use of Probe Requests frames. These messages are unencrypted and, therefore, they can be simply captured using passive sniffing techniques, which are non-intrusive.

Before MAC randomization, counting the number of mobile devices in a given area was straightforward, as the real MAC address from devices was sent in these messages. The number of devices could then be determined by simply counting the number of different MAC addresses collected from these messages, as each device sent its real and unique MAC address on probe requests. However, with the emergence of this new feature on the latest operating systems of

mobile devices, crowding estimation has become a more difficult task, leading to overcounting estimations, since the ratio between a device and the number of MAC addresses generated in its Probe Requests is a 1:N problem, due to the generated virtual MAC address that change over time. As so, this new feature makes MAC addresses “not feasible to uniquely identify devices under probing; rather, it adds an extra layer of security to protect users’ privacy” [33].

Several strategies have been adopted regarding Probe Request capturing, most of them addressing the MAC address randomization process to correctly estimate the number of devices, which we can list in the following:

- **Temporal arrival of Probe Requests:** based on the temporal arrival of Probe Requests from the same source (same MAC address) to perceive if the device is in the range of the sensor. [24]
- **SSIDs Comparison:** based on comparing the known networks (SSIDs) from a device, information that is contained in the Probe Requests. [2]
- **Fingerprinting:** This strategy is based on generating a unique identification (fingerprint) from other contents of Probe Requests frames. The contents to generate this fingerprint are usually from the Information Elements (IEs) of Probe Request frames. [4, 34]
- **Fingerprinting + Clustering:** this strategy relies not only on fingerprinting from IEs, but also on other properties from these messages, such as the **SEquence numbers (SEQ)**, burst size, or the **Inter-Frame Time Arrival (IFTA)** from Probe Requests frames. A clustering algorithm considering several properties at the same time is applied, where from it, each cluster should uniquely represent a device. [5, 7, 13, 32, 33]

The previous work from [24] shows an approach to detect the presence of mobile devices in a given location. In this study, probe requests were captured by a sensor composed of multiple sets of Wi-Fi antennas for detecting these messages in several Wi-Fi channels simultaneously. An algorithm based on the temporal arrivals of probes from the same MAC address was proposed, which authors refer to as a state machine methodology. The idea is plenty simple: a device is nearby if we receive periodic probes from him, and, therefore, a state machine was developed in order to perceive the device’s presence in the sensor range. A device can have several states, such as initial stage, potential arrival, arrival confirmation, potential departure, and departure confirmation. The authors claim good precision although the solution is fully dependent on probes with the same MAC address from devices, something that can easily be displeased if manufacturers start to change the MAC address more quickly. Another interesting work regarding crowd counting is the work from [2]. In this study, the authors aimed to count the number of devices in a location and distinguish local residents from visitors based on the **Preferred Network List (PNL)** sent in Probe Requests, which contains the SSIDs from the known networks of a device. Experiments were done in the city of Alcoi, Spain, where a previous collection of all the networks (SSIDs) from the city was collected and saved in a large database. The latter was further used to compare with the list of known networks from device Probe Requests. From this, an accuracy of 83% was obtained, with some overestimation and incongruities in the number of individuals.

The implementation of the MAC address randomization added a level of complexity to uniquely identify devices. Therefore, the research has advanced towards the exploration of other properties and fields of the Probe Request frames, as well as temporal patterns in their transmission that allow uniquely identifying a device, since the MAC address is no longer a reliable option for itself. Most of these approaches are based on fingerprinting and clustering techniques, where there can be slight differences in the Probe Request's fields to generate the fingerprints. An interesting work from [34] proposed a network of sensors to estimate the attendance of public events. The sensors capture Probe Requests and generate a 6-byte fingerprint from the content of the most relevant and non-highly variable IEs from Probe Requests with a random (virtual) MAC address. As a result, devices could be uniquely identified by this 6-byte fingerprint for random MAC addresses, or by its real MAC address. The system was tested in public events with a considerable density of people, where authors claimed an accuracy close to 95%. Another work from [4] used the same approach considering not only the IEs, but also the SSIDs and the recurrence of the same MAC address to generate a fingerprint to uniquely identify mobile devices. The authors validated the proposed method during a conference that took place at the Evenia Olympic Congress Centre in Lloret de Mar, Spain.

Some other studies not only considered the contents from the IEs for fingerprinting but also combined this information with other properties or patterns from Probe Request frames. Given this proposal, many studies used clustering algorithms for considering a combination of different features from Probe Requests for achieving better results, such as the works from [5, 7, 13, 32, 33].

In [5], a method for crowd counting was proposed. The method consisted of two separate parts: an algorithm that classifies MAC addresses and analyzes frame bodies of Probe Requests to estimate the number of devices, which the authors called *Vision*, and a clustering algorithm for analyzing the SEQ of Probe Requests for counting the exact number of devices, referred as *TrueSight*. The *Vision* algorithm is capable of classifying the MAC address into three groups: (i) those of a device connected to an AP, (ii) those of a device not connected to an AP, and (iii) the MAC addresses from APs. The MAC addresses which belong to an AP or to a device connected to an AP were directly extracted from the Probe Requests. For the remaining MAC addresses, which belong to devices not connected to an AP, a fingerprint was generated considering the most relevant IEs, just like the previous approaches mentioned above. At this stage, only different models of devices could be distinguished, and, for this reason, the *TrueSight* clustering algorithm was applied in order to distinguish different devices. By observing the SEQ of the Probe Requests, it was able to differentiate one device from another even if they had the same frame body. This could only be achieved by assuming that the SEQ sent by the same device has a high chance of being consecutive (or close) and that SEQ from different devices has little chance of being close. This method was only tested with a dataset that was purposefully generated for the scope of this work, reaching an accuracy of 75%, although, it was not tested in a real crowded scenario.

Another similar work from [13] considered the IEs, the SEQ, the **Received Signal Strength Indicator (RSSI)**, and a neuronal network for estimating crowding levels. The proposed method was tested on an entire floor of a shopping mall in Hong Kong, reaching an accuracy little higher than 80%. Other approaches explored temporal patterns in conjunction with the frame

bodies of Probe Request transmissions to leverage crowding levels, such as those reported in [32, 33]. These two studies proposed a MAC address de-randomization algorithm based on fingerprinting from IEs and also from Probe Request burst properties. The fingerprinting was based on the IE IDs and lengths from probes with virtual MAC addresses, however, this feature by itself was not enough to discriminate the devices, and a more complete analysis considering the incremental speed of the SEQ was applied. After this, a temporal analysis of the burst frequency and the period of time between two consecutive probes (IFTA) was considered for clustering. The first study of this algorithm was tested at the University of Cagliari's Campus, achieving an accuracy of about 91%, however, it was not tested in a real crowded scenario. Therefore, a follow-up to this work [33] tested the algorithm first in a controlled environment, reaching an accuracy of 97%, and further inside buses in Italy for an Automatic Passenger Counting system, with a precision of 75%. This study showed a very good approach to the randomization process from operating systems, although, this solution takes some time (>10 minutes) to achieve good results, as the clustering is based on patterns from several bursts, which are periodically emitted from devices.

Other work from [7] used the same approach considering the IEs for fingerprinting and also used a clustering algorithm for combining the generated fingerprints with burst sizes and the (IFTA), and tested the method in the canteen of the University of Twente's, with an accuracy of 90%.

Table 2.1 presents the different approaches taken regarding the MAC address randomization issue from all previous works mentioned above.

Table 2.1: Primary studies classification regarding the MAC address randomization.

Authors	Probe Request capturing	MAC Address de-randomization	Strategies for crowd counting	Real Scenario Appliance	Precision
[24]	Yes	No	Temporal arrival of Probe Requests (same MAC address)	Not specified	91%
[2]	Yes	Yes	SSIDs Comparison	Alcoi (Spain) City	83%
[34]	Yes	Yes	Fingerprinting (IEs total contents)	Public events	96%
[4]	Yes	Yes	Fingerprinting (IEs, SSID's, MAC recurrency)	Conference (Lloret de Mar, Spain)	Not available
[5]	Yes	Yes	Fingerprinting (IEs) Clustering (IEs + SEQ)	Not available	75% (with simulated data)
[13]	Yes	Yes	Fingerprinting (IEs) Clustering (IEs, SEQ, RSSI)	Shopping mall at Hong Kong	80%
[32]	Yes	Yes	Fingerprinting (IE IDs + LEN) Clustering (IEs + LEN, SEQ, Bursts rate, Inter-frame time)	University of Cagliari's Campus	91%
[33]	Yes	Yes	Fingerprinting (IE IDs + LEN) Clustering (IEs + LEN, SEQ, Bursts rate, Inter-frame time)	Buses in Italy (Automatic Passenger Counting system)	97% (controlled environment) 75% (real scenario)
[7]	Yes	Yes	Fingerprinting (IEs) Clustering (IEs, Bursts size, Inter-frame rate)	Canteen of University of Twente's Campus	90%

CHAPTER
3

TOOLKIT ARCHITECTURE

Contents

3.1	Proposed Toolkit Architecture	25
3.2	Crowding Data Collection	27
3.3	Communication & Services	31
3.4	Visualization & Notification	33

This chapter describes the architecture of the STToolkit, with the appropriate options and considerations taken into account, concerning the nature of the touristic areas where the sensors are intended to be deployed in order to improve the quality of the tourism experience and promoting destination sustainability.

This chapter is organized as follows: section 3.1 presents the proposed architecture for the STToolkit, section 3.2 presents the crowding data collection from sensors, with a detailed view of the proposed algorithm for detecting devices through Wi-Fi technology, then, section 3.3 presents the communication and services of the STToolkit, and finally section 3.4 presents the visualization and notification of the collected crowding data of the STToolkit.

[This page has been intentionally left blank]

Chapter 3

Toolkit Architecture

3.1 Proposed Toolkit Architecture

The STToolkit should be capable of monitoring the occupation of spaces in several locations and also provide a quick visualization of data in order to make just-in-time decisions towards overcrowding situations to mitigate *overtourism*. As so, the architecture of the STToolkit needs to be designed in a way that accomplishes this main goal. The latter can be divided into three large groups:

- **Sensors** - As it is intended to provide crowd monitoring at several locations, the STToolkit architecture needs inherently to have several end-nodes deployed at those locations for collecting crowding data, which in our approach is performed by sensing trace elements of mobile device's wireless activity in the sensor's vicinity in real-time. This group is further described in detail in section [3.2](#).
- **Cloud Server** - The sensors require an end-point for ingesting and rendering the crowding information sent by the multiple sensors. As so, the STToolkit architecture requires a cloud server for these purposes. This group is further described in detail in section [3.3](#).
- **Users** - The STToolkit is aimed at non-technical users, allowing them to easily and non-effortlessly monitor crowding at target locations in real-time, and quickly visualize that data in order to make just-in-time decisions towards overcrowding situations. As so, the architecture of the STToolkit also has to provide easy visualization and access to data, and the creation of notification policies when those situations occur, without requiring any high expertise. This group is further described in detail in section [3.4](#).

Furthermore, the STToolkit has also been built in the scope of the [RESETTING](#) project. The latter is aimed at facilitating the transition towards a more sustainable operation of tourism-related SMEs that deal with potentially excessive tourism crowding levels, where overcrowding situations may arise, to allow optimizing their quality of service and delivery as a crowding monitoring solution. As so, the architecture of the STToolkit also needs to consider these target SMEs.

The overcrowding sensors are the basis of the STToolkit, and should meet several requisites, concerning the nature of the touristic areas where they are intended to be deployed, and the forecasting services for users. As so, and also according to the [RESETTING](#) goals, the requisites for the crowding sensors can be grouped into three main categories, being them: scalability, robustness, and discreteness.

In terms of scalability, the STToolkit's sensors should comply with the following requisites:

- Low-cost solution.
- Easy installation and configuration.

- Flexible deployment.
- Connection independency.

These requisites must be accomplished to ensure complete scalability by keeping the overall cost per node low for faster and less expensive deployment. Also, an easily installable and configurable device must be provided, as well as a flexible deployment, so that any typical user can implement its own crowding monitoring system in a very quick and simple manner. Furthermore, connection independence needs to be provided, since overcrowding situations may occur in both enclosed or open space environments, thus ensuring that our sensors will always be capable of communicating data whatever its location.

Regarding robustness, the STToolkit's sensors should meet the following requisites:

- Ability of 24/7 runtime operation.
- Resilient to failures (e.g., connection failures).
- Ability to endure severe conditions (e.g., rain and high temperatures).

By complying with these requisites, we ensure that our sensors can monitor crowding ever since they are deployed, with the assurance that they will always be capable of uploading data regardless of the installation environment.

Finally, related to discreteness, we have the following requisites:

- Simple and discreet design.
- Reduced size.
- Lightweight device.
- Low operation noise.

By accomplishing these requisites, the sensors will not stand out from the environment where they are deployed, whether in both indoor or outdoor appliances and also not disturbing the surrounding environment.

Considering all requisites previously mentioned, the proposed architecture of the STToolkit is presented in Figure 3.1. The components filled in yellow are re-used open-source components, while the components filled in blue are customized components purposely created for the STToolkit. A flexible architecture has been designed, where sensors collect the crowding information in its vicinity and periodically report that information to a cloud server. The latter also has other components for making downlink communication transparent and providing uplink services for rendering the information and creation of notification policies.

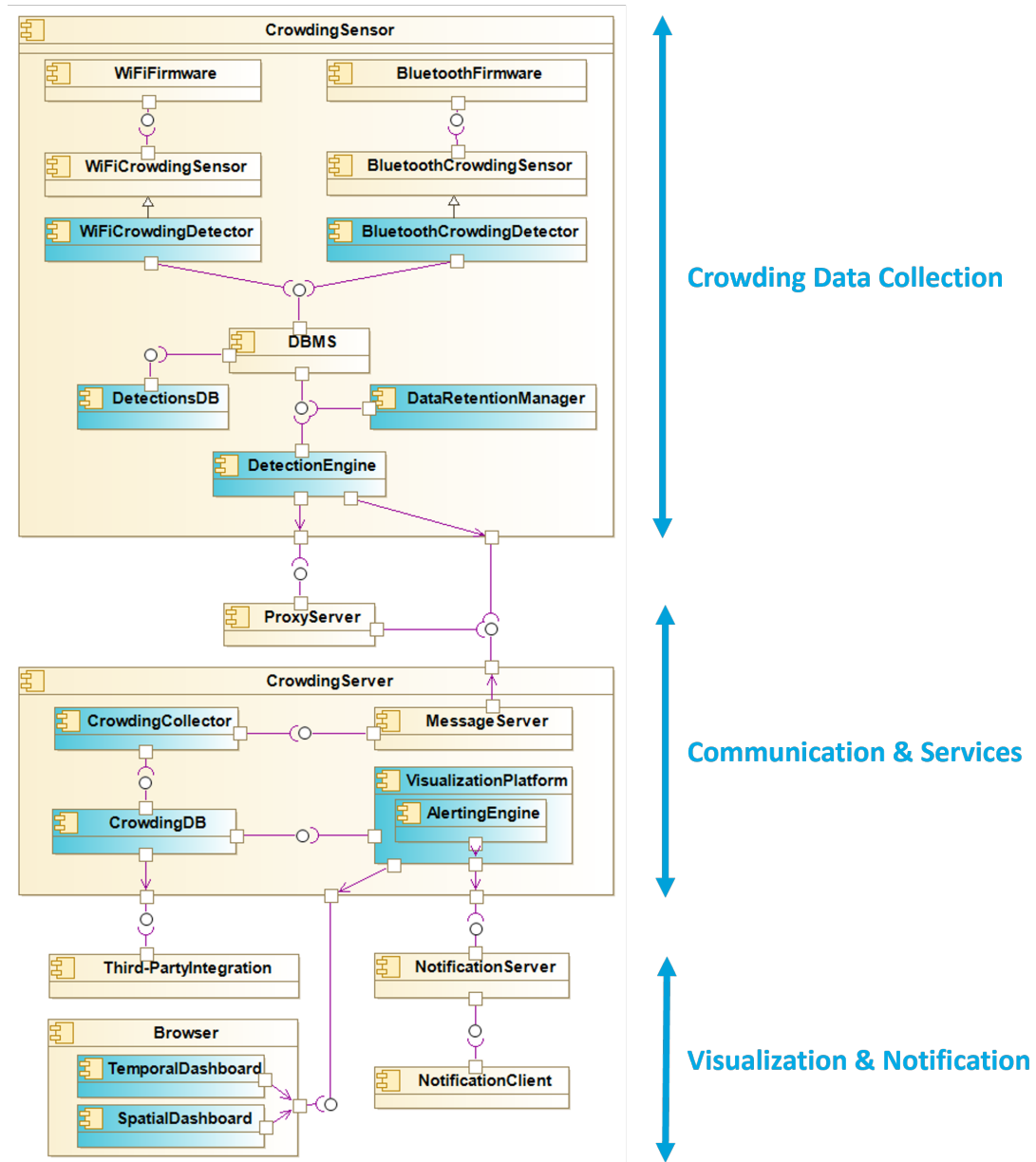


Figure 3.1: Crowding STToolkit architecture.

3.2 Crowding Data Collection

Each *CrowdingSensor* has firmware that controls the hardware used to detect mobile devices in each wireless technology (Wi-Fi and Bluetooth), namely the *WiFiFirmware* and the *BluetoothFirmware*. The latter task is performed by the *WiFiCrowdingSensor* and the *BluetoothCrowdingSensor*, which, from these, the *WiFiCrowdingDetector* and the *BluetoothCrowdingDetector*, inherit its properties. These are custom-made components purposely customized for passively capturing mobile devices' trace elements in each wireless technology, and inserting the collected data into the *DetectionsDB*, an anonymized local database where all gathered information is stored. The latter task is only possible due to the existence of the *DBMS*, the database management system of the *CrowdingSensor*, responsible for managing all captured data by the sensor. As a result, all collected data is inserted into the same local database, which ensures data standardization and

quick local access to the stored crowding information. The *DBMS* also provides an interface for two other custom-made components, namely the *DataRetentionManager*, responsible for only retaining the useful information for device counting by cleaning outdated and unnecessary data from the *DetectionsDB*, and the *DetectionEngine*, responsible for counting the number of devices by analyzing the information contained in the *DetectionsDB* and periodically reporting the crowding information to the *CrowdingServer*, the system's cloud server. The sensors can perform only Wi-Fi or Bluetooth detection, or both at the same time, where switching between the technologies used for detection should be quick and simple, thus providing flexibility for detecting devices.

Each sensor generates a metric for each detection technology (Wi-Fi and Bluetooth), to perceive the density of people more clearly according to the corresponding range. Regarding user privacy, all gathered data is anonymized before being stored in the *DetectionsDB*, which is ensured by applying a hash function to the gathered information. This is clearly an edge computing approach since most of the processing for collecting data and generating crowding level measurements is performed locally in each sensor so that only the number of detected devices is sent to the cloud server. As so, the information to be passed to the cloud server is minimal, not requiring a high sampling rate for data transmission, and also protecting nodes from outside threats, since the communication line prevents the majority of attack types. Furthermore, limiting data exchange not only reduces communication costs, but also eases protection complexity for the node, and makes it easier to guarantee user privacy.

The following subsection explains and describes in detail the developed algorithm for detecting mobile devices through Wi-Fi technology, also tackling the MAC address randomization process of mobile devices.

3.2.1 Algorithm for Wi-Fi detection

The introduction of the MAC address randomization process in mobile devices, for security and privacy concerns, has made the unique identification of a mobile device a much more difficult task and, consequently, more difficult to accurately perform device counting. Therefore, one of the primary focuses of this research work was on addressing this randomization process of mobile devices, for more accurate crowd counting.

For this purpose, an algorithm was designed for the detection of mobile devices through Wi-Fi detection, tackling the MAC address randomization issue using a fingerprinting technique, presented in Figure 3.2. The fingerprinting technique was chosen due to being the most cost-effective approach of those presented in 2.4.1 to tackle the MAC address randomization issue, considering the underlying schedule of this dissertation. The fingerprinting technique relies on the IEs conveyed in probe request frames and contains meaningful information about the device, such as the supported transmission rates, Wi-Fi and radio capabilities, and vendor-specific information, among others.

The explanation of each step of the algorithm is further presented below. A similar algorithm is also envisaged for Bluetooth detection since the randomization problem is also pertinent to this technology.

The *WiFiCrowdingDetector*, seen in Figure 3.1, is responsible for processing each Wi-Fi

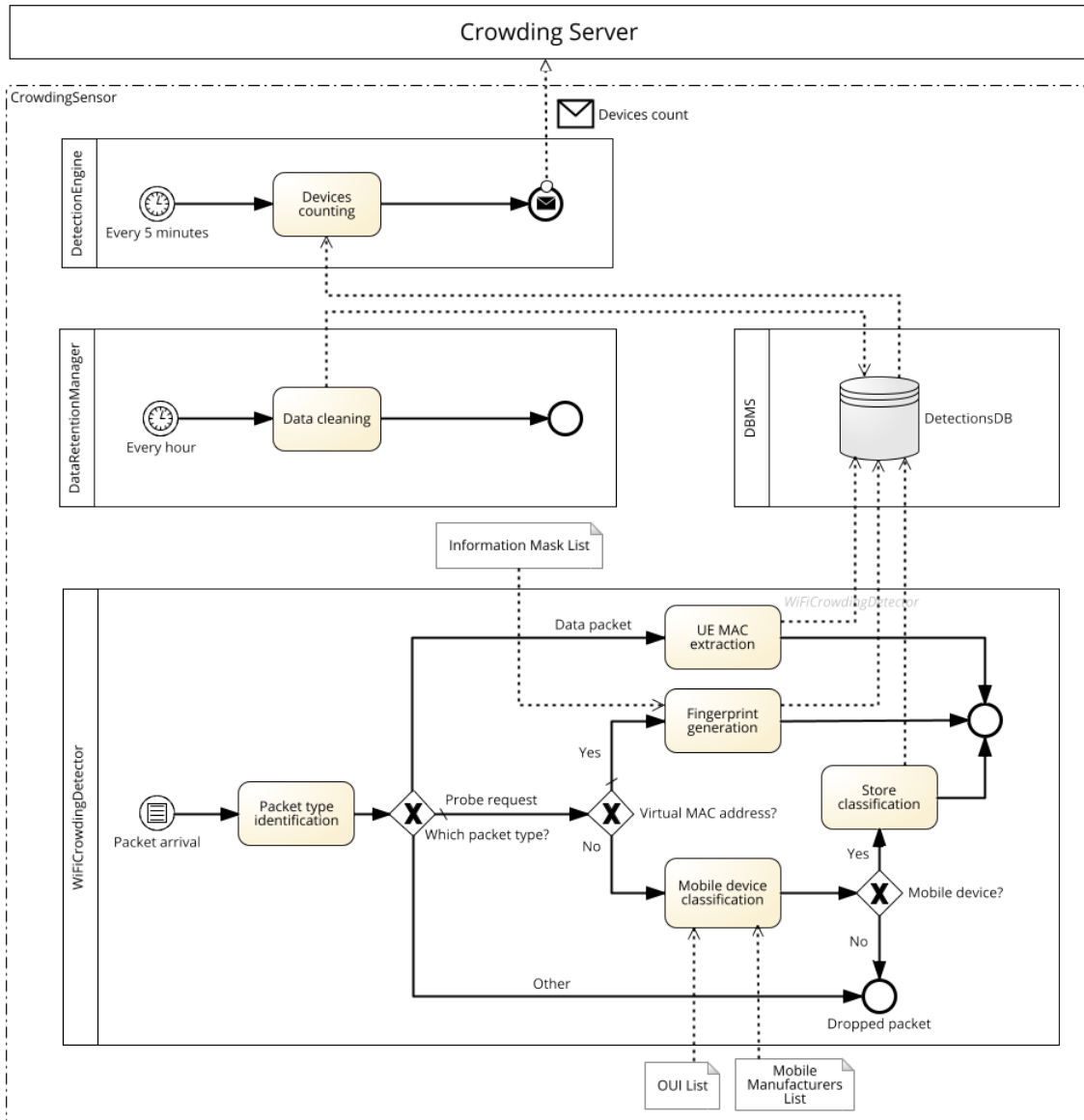


Figure 3.2: Algorithm for Wi-Fi detection.

packet captured by the sensor. The first operation performed is a packet type identification to identify data packets and probe requests. Data packets will be used for counting the number of devices connected to an AP, i.e., connected to a Wi-Fi network, and probe requests for counting the number of devices not connected to any AP. The packet type can be obtained by checking the Frame Control field, presented in Figure 2.3, more specifically its type and subtype subfields. Packets that are not either data packets or probe requests are immediately dropped since they are not relevant for device counting.

Regarding data packets, only the **User Equipment (UE)** MAC address needs to be accounted for. For this purpose, it is necessary to first locate it in the frame, which is performed by checking the **Device Status (DS)** information in the packet frame header, since the UE MAC address position may vary according to the direction of the frame, which is conveyed in the DS information. These MAC addresses can be directly counted as single devices because when

a device is connected to an AP, the MAC address is kept constant throughout the connection and, therefore, will not change randomly. Therefore, after the UE MAC address extraction, it is directly stored in the *DetectionsDB*, the sensor's anonymized local database.

Regarding probe requests, the first operation performed is aimed at distinguishing the MAC address conveyed in the Source Address, i.e., the address that conveys the device identification, between a real and a virtual MAC address. This is performed by checking the 7th less significant bit of the 1st octet of the MAC address, as already shown in Figure 2.2.

To follow the trace of a real MAC address, a device classification is applied, aimed at only accounting for MAC addresses that belong to mobile devices. This is done by checking the address's manufacturer: if the manufacturer of the MAC address, obtained from the *OUI List*, a list with OUIs¹ and respective manufacturer obtained from the [Wireshark manufacturer database](#), matches one of the manufacturers of the *Mobile Manufacturers List*, a list with all mobile device manufacturers obtained from [Phone Arena](#), the MAC address should be considered as a mobile device and it must be counted, and, therefore, stored in the sensor's local database; otherwise, the MAC address is not considered as belonging to a mobile device and the packet is discarded.

To follow the trace of a virtual MAC address, a fingerprinting technique must be performed, since this type of address is generated by devices that use MAC address randomization and, therefore, should be uniquely identified regarding this technique. For this, the IEs contained in the frame body of the probe request are analyzed. Table 3.1 shows the IEs used to create a device fingerprinting from probe requests with a virtual MAC address. The procedure used to determine which information should be considered and which should be ignored for each IE is further explained in detail in section 4.3.3. The latter contributed to the generation of the *Information Mask List*, a list that contains the specific bytes or bits of each IE that should be ignored for the fingerprint, which are summarized in Table 4.4. After analyzing all IEs, also having into consideration the *Information Mask List*, a hash function is applied to the selected information. As a result, a 64-bit fingerprint is generated and stored in the *DetectionsDB*. Then, all the devices that are trying to connect to a Wi-Fi network, by sending probe requests with a virtual MAC address that may vary over time to discover available networks in proximity, are expected to be uniquely identified by the fingerprint. As so, each fingerprint should be counted as one mobile device that is trying to connect to a Wi-Fi network that uses MAC Address randomization.

Also, to avoid counting the same device twice, if the same device is captured in both data packets and probe requests, it is only accounted for once. However, this can only be accomplished for data packets and probe requests with real MAC addresses, as its fingerprints are purely based on the MAC address for both cases and so they can be matched, while fingerprints from probe requests with virtual MAC addresses are based on the IEs, making it impossible to match those fingerprints with the ones from data packets.

While the *WiFiCrowdingDetector* is continuously collecting data and storing the crowding

¹OUI is a part of the MAC address that identifies the vendor of the network adapter.

²Direct-Sequence

³High Throughput

⁴Very-High Throughput

⁵Resource Management

Table 3.1: Probe Request's Information Elements used to create device fingerprint.

Information Element	IE ID	IE Length	Description
Supported Rates	1	<8	Data transfer rates supported by the device.
Extended Supported Rates	50	<256	Other bit rates supported by the device.
DS ² Parameter Set	3	1	Device's channel setting when sending a probe request.
HT ³ Capabilities	45	26	Compatibility with the 802.11n standard.
VHT ⁴ Capabilities	191	12	Compatibility with the 802.11ac standard.
Extended Capabilities	127	<256	Other device capabilities.
RM ⁵ Enabled Capabilities	70	5	Information for measuring radio resources.
Interworking	107	<9	Interworking service capabilities of the client.
Vendor Specific	221	<256	Vendor Specific information (e.g., device manufacturer)

data into the *DetectionsDB*, the sensor's local database, the *DetectionEngine* will periodically analyze the stored information, and count the number of devices detected within a sliding window of X minutes, i.e., the number of devices detected in the last X minutes, and upload that information every Y minutes to the *CrowdingServer*. Both the sliding window period and data sampling rate can be independently and easily configured by the user, with a minimum of 1 minute each. Also, to ensure real-time or near real-time data availability, it is important to acknowledge that although users can have control over the periodicity of crowding measurements, the data sampling rate of the sensors needs to be as little as possible to accomplish this requisite. The number of devices detected sent in each measurement is the sum of (i) the number of devices connected to an AP, obtained from the UE MAC addresses from data packets, summed with the number of devices not connected to any AP, obtained from (ii) the MAC addresses from probe requests with real MAC addresses and (iii) the number of different fingerprints from probe requests with virtual MAC addresses.

3.3 Communication & Services

To better address installation location requirements and connectivity limitations, a flexible deployment regarding uplink technologies has been considered for the STToolkit. Crowding data can be uploaded to the *CrowdingServer* by using a variety of communication protocols, such as Wi-Fi or [Long Range Wide Area Network \(LoRaWAN\)](#). If Wi-Fi is available on site, data can be uploaded directly to the *Message Server* via the [Message Queuing Telemetry Transport \(MQTT\)](#) protocol, a lightweight method of carrying out messaging, using a publish/subscribe model,

widely used for [Internet of Things \(IoT\)](#) applications. This option can be applied straightforwardly in indoor tourism scenarios, for instance, in a museum, which generally provides a Wi-Fi network to visitors.

In outdoor scenarios, such as public parks or city squares, where overtourism situations can also arise, Wi-Fi coverage may not be available. Since sensors must upload crowding information, other approaches rely on mobile operators' communication, which may be an expensive option, usually with monthly fees depending on the number of sensors used, each using a [Subscriber Identity Module \(SIM\)](#) card. To mitigate this problem the STToolkit also offers the option for uploading data via the LoRaWAN protocol, a low-cost alternative that provides scalability and is also feasible for our application since sensors only communicate a small amount of data, i.e., the number of detected devices. Regarding this option, sensors must be equipped with a dedicated LoRa board and corresponding antenna, to communicate the crowding information to a LoRaWAN gateway, presented as the *ProxyServer* in [Figure 3.1](#), that, in its turn, will route the information to the *Message Server* also via MQTT protocol. Regarding coverage, there are a few LoRa networks, designed for IoT appliances, that can be used for uploading data, like [The Things Network](#) open collaborative network or the, also crowdsourced, [Helium](#) network, a decentralized wireless infrastructure supported by blockchain.

As so, to solve the uplink communication problem with connectivity, users can choose between three options for uploading data to the *CrowdingServer*:

- Wi-Fi on site (when available);
- LoRa with third-party LoRaWAN service provider (e.g., Helium-IoT);
- LoRa with private LoRaWAN server;

As so, from all the options available, users must choose the option that best fits their needs and requirements for implementing its own system based on the STToolkit. To help with this purpose, the STToolkit will include a sensor deployment calculator for users to estimate the most cost-effective uplink alternative according to the installation location of each sensor.

As already shown in [Figure 3.1](#), the *Message Server* is the only entrance point for all messages in the *CrowdingServer*, independently of the communication protocol used for uploading the crowding information. This provides transparency since all messages are received in the *CrowdingServer* via the MQTT protocol independently of the communication technology adopted for uploading the crowding information. Furthermore, there can also be areas with low or no Wi-Fi or LoRa coverage, such as a remote camping site or a rural tourism facility. In those cases, it is also possible to acquire equipment for providing such coverage, for instance, a Wi-Fi mesh system, that will allow expansion of the Wi-Fi network coverage, or a Helium hotspot or a The Things Network gateway, with a few examples shown in [Figure 3.3](#), for grating LoRa network coverage for uploading data via the LoRaWAN protocol.

As so, not only by ensuring that network coverage can always be provided for server connectivity but also by offering multiple uplink communication options to upload the crowding information to the *CrowdingServer*, something that has been disregarded by other approaches, the network limitations at the installation location of sensors are mitigated, thus providing scalability for the STToolkit.



Figure 3.3: Helium hotspots (left) and The Things Network gateways (right) for grating LoRa network coverage in areas with low coverage.

Following the trace, after being received in the *Message Server*, the crowding measurements must then be stored in a database for further visualization. The *CrowdingCollector* is in charge of that task, by pushing all measurements received in the *Message Server* into the *CrowdingDB*, a time-series database responsible for ingesting and storing the data sent by all sensors. The *CrowdingDB* can be viewed as the main database of the STToolkit since all uplink communication will be stored in this component, which is capable of querying data rapidly as well as provides support for data visualization platforms for users to visualize the crowding data in real-time.

In summary, the STToolkit offers multiple uplink options for communicating the crowding information to the cloud server, namely either by Wi-Fi or LoRaWAN protocols, thus mitigating the network limitations at the installation location of the sensors, something that has been disregarded by alternative approaches. Such flexible deployment ensures that sensors can always connect to the cloud server and upload data whatever its location: for indoor scenarios, where Wi-Fi coverage is much more likely provided, the information can be uploaded directly; for outdoor scenarios, if Wi-Fi is not available on-site, the STToolkit also offers the possibility to upload the information via LoRaWAN protocol through a LoRa network. Here, users can choose between two alternatives for uploading crowding information: using LoRa with a private LoRaWAN server, or using LoRa with a third-party LoRaWAN service provider. Even if neither Wi-Fi nor LoRa network coverage is provided, it is also possible to acquire equipment for that purpose. As a result, from all the communication options available, users must choose the option that best fits their needs and requirements for implementing their own system at target locations. The same applies to the tourism-related SMEs affiliated with the RESETTING project.

3.4 Visualization & Notification

The *CrowdingServer* also has several components to make downlink communication transparent and provide several uplink services that can be used by users to understand the crowding levels in areas where each sensor is deployed in a clear and simple perspective. The crowding services can provide:

- Rendering of temporal information.

- Rendering of geographic information.
- Notification policies (e.g., when a crowding threshold is reached).
- Raw data for custom-made integrations (e.g., spatial visualization using avatars in a [Building Information Model \(BIM\)](#)).

The *VisualizationPlatform* is responsible for the data visualization of the STToolkit. The latter can connect directly to the *CrowdingDB* for rapidly querying the ingested data from the database, which can further be visualized in dashboards. Data can be used to render temporal information, with graphs that allow users to perceive people's concentration and flows during specified periods throughout the day, highly populated events and their duration, and for comparing crowding between locations at any particular time or time range. Data can also be used for rendering geographic information, for instance, with heatmaps that allow users to understand how people are distributed through the several locations where the sensors are deployed. For facilitating its installation and configuration, the STToolkit also provides some pre-configured dashboards to users for a quick and straightforward data visualization setup with all the previously mentioned features.

The *VisualizationPlatform* also has the *AlertingEngine* component, which provides the creation of notification policies for notifying users of events, such as when a crowding threshold is reached, or to simply advise on waiting times at a particular time of the day. As so, and also empowered by the provided real-time or near real-time data availability, the notification policies enable the possibility of making just-in-time decisions upon sudden overcrowding situations, as they can be detected almost as fast as they arise, through several contact points. Also, besides being pre-configured, dashboards can be edited and customized by the final user according to its preferences, where it is possible to change the dashboard layout, background, and colour palette, and also create new graphs and notification policies with ease since it can be performed graphically and directly on the dashboards, without requiring any high expertise. For this purpose, and also regarding the *RESETTING* project goals, support material is also provided, such as operation and user manuals, and online video tutorials, to provide guidance to SMEs on how to implement these modifications and create new panels and notification policies in their dashboards.

Then, users can visualize the crowding data by simply using a web browser to access the *CrowdingServer*, where the *VisualizationPlatform* is installed. From there, after an authentication procedure is completed by introducing its credentials, the user gains access to all configured and customized dashboards for visualizing the crowding data gathered by the sensors at the target locations.

Furthermore, the raw crowding data can also be used for third-party integrations. For this, other applications can directly connect to the *CrowdingDB* for querying the raw crowding data, which can further be used for custom-made integrations, for instance, using the crowding data with walking avatars on top of a BIM to achieve a more realistic perception of space occupancy. The *Third-PartyIntegration* component represents those integrations that can be created by directly gathering the crowding data from the *CrowdingDB*.

CHAPTER
4

TOOLKIT IMPLEMENTATION

Contents

4.1	Operating System	38
4.2	Local database	38
4.3	Wi-Fi detection	40
4.4	Data Upload, Data Management and Tasks Automation	47
4.5	Data Ingestion and Visualization	50
4.6	Toolkit Prototype	57

This chapter discusses all the choices and considerations for the implementation of the crowding STToolkit. Section 4.1 starts presenting the chosen operating system for the sensors, followed by the choice of its local database to store the gathered information from mobile devices in section 4.2. Section 4.3 presents the hardware and software used for detecting mobile devices regarding Wi-Fi technology and section 4.4 presents the programs used for data upload, data management, and the automation of all tasks by the sensors. Then, section 4.5 describes all details about the data ingestion and data visualization, and finally section 4.6 presents the available hardware for the STToolkit sensors used for testing and validation and cost breakdowns of the STToolkit. All code developed for the STToolkit is open-source and can be accessed in the [RESETTING GitHub repository](#).

[This page has been intentionally left blank]

Chapter 4

Toolkit Implementation

Figure 4.1 presents the deployment diagram representing an implementation of a network of sensors with the STToolkit. The components filled in yellow are re-used open-source components, and the components filled in blue are customized components purposely created for the implementation of the STToolkit. The following sections will walk through the considerations and decisions for each of these components, from the sensors, the cloud server for data ingestion and storage, to the data visualization platforms used to visualize the crowding data.

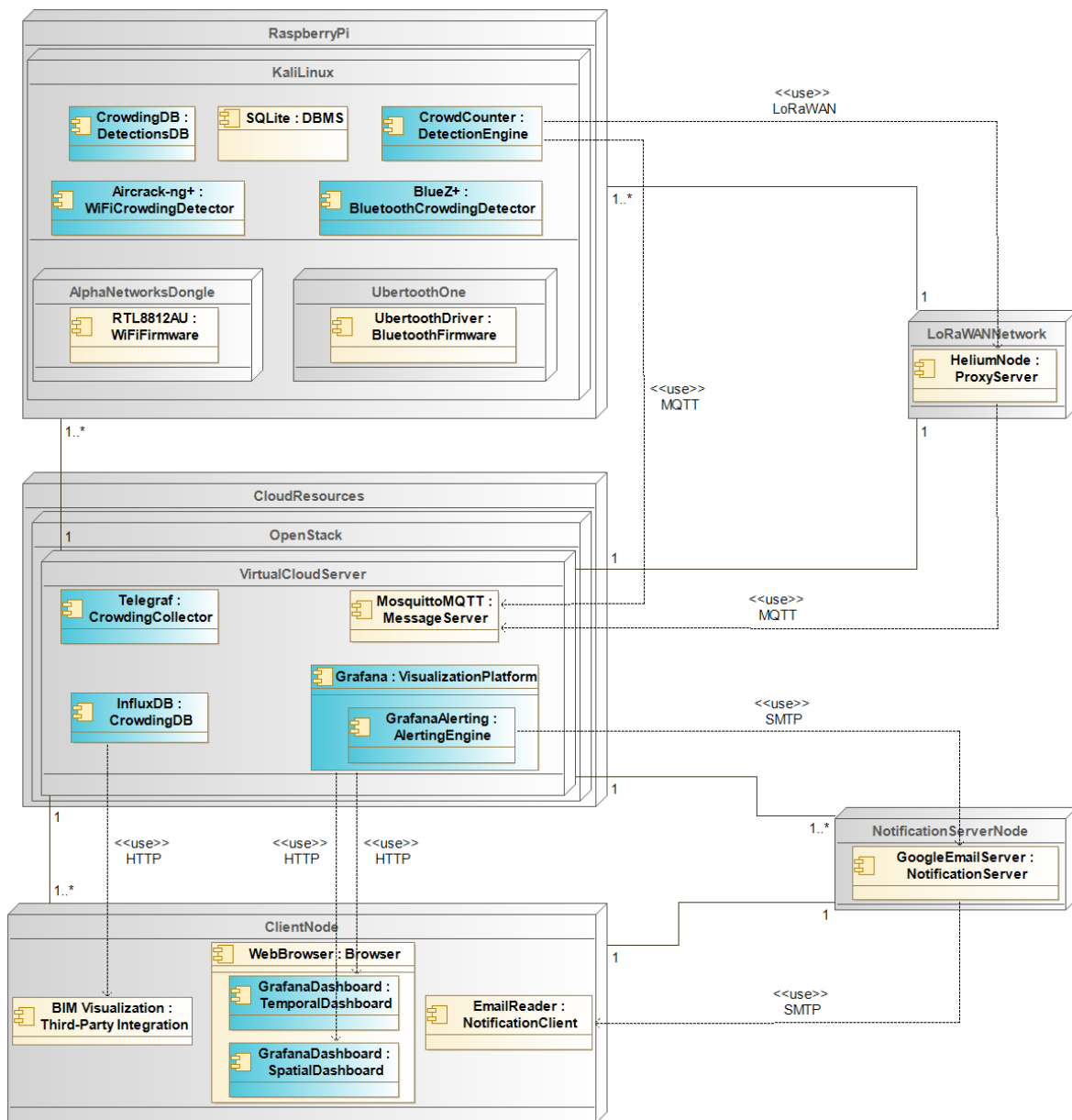


Figure 4.1: Deployment diagram of the crowding STToolkit.

Also, it is important to acknowledge that the Bluetooth detection of devices was not implemented, as well as the upload via LoRaWAN and the notification policies for alerting users.

4.1 Operating System

The initial step for the implementation of the STToolkit was to decide the development environment for all sensor programs. The sensor's operating system has an important role in this part since it will support all applications used to detect the number of mobile devices in its proximity.

For keeping the costs low, an open-source operating system was the best alternative and so, a Linux-based operating system was chosen. Linux concedes better flexibility in the technologies and framework used and better control of the behaviour of the detecting device, as well as its distributions are mainly lightweight, not consuming many system resources. Besides, Linux-based systems offer plenty of open-source programs and libraries due to their large community support, which contributes to a faster development process. The second decision involves deciding which Linux distribution should be used by the STToolkit sensors.

There are numerous Linux distributions with various focus and applications developed by different teams. This variation in Linux distributions is only possible due to the nature of Linux itself, as it is an open-source operating system, users can compile its kernel and add software as they wish with ease. For the STToolkit sensors, a [Kali Linux](#) distribution was chosen. It focuses on cybersecurity and penetration testing attacks and comes with a large number of pre-installed tools and frameworks, which can be used for capturing and inspecting the protocol trace elements of mobile devices (sniffing tools) in different wireless technologies, such as Wi-Fi and Bluetooth technologies.

4.2 Local database

The STToolkit sensors require a local database from which all collected data will be stored when performing device counting. This database will be the data storage of the sensors and, for this reason, needs to be lightweight, reliable, and capable of inserting and querying data rapidly, and is related to the *DetectionsDB* component, seen in Figures 3.1 and 4.1. Considering the previous requisites, [SQLite](#) was chosen for the local database of sensors. SQLite is a well-known self-contained, serverless, zero-configuration relational database system. In fact, it is the most widely deployed database engine worldwide, and it can be used for numerous applications, including high-profile projects. Furthermore, SQLite also provides plenty of [Application Programming Interfaces \(APIs\)](#) that can be used for operating a [DataBase Management System \(DBMS\)](#) on applications. In particular, SQLite provides a C-language API that can be used by other applications developed in C-language to implement a DBMS. This feature is particularly important since the detection software implemented on sensors is also written in C, where this API was used to store the collected crowding information in the sensor's local database, as further described in section 4.3. Also, it is important to mention that this database was created taking only into account the requirements needed for storing information on devices detected through Wi-Fi technology, as this was the only detection technology implemented in this dissertation.

The STToolkit sensor's local database schema is presented in Figure 4.2. As shown, the database has two different tables, namely the *Data_Packets* and the *Probe_Requests*. The first, as

its name suggests, has the information of devices detected from data packets, and will be used for counting devices connected to APs, i.e., connected to a Wi-Fi network, as already explained in subsection 3.2.1. In the same way, the second has the information of devices detected from probe requests and will be used for counting devices not connected to any AP, i.e., not connected to a Wi-Fi network.

Data_Packets		Probe_Requests	
Frame_Type	TEXT	Frame_Type	TEXT
ID 🔗	TEXT	ID 🔗	TEXT
First_Record	DATETIME	First_Record	DATETIME
Last_Time_Found	DATETIME	Last_Time_Found	DATETIME
Manufacturer	TEXT	Manufacturer	TEXT

Figure 4.2: Sensor's local database schema.

The Wi-Fi detection program, seen in section 4.3, detects a device, and inserts or updates a line into one of the two tables, depending on the packet type: if the device has never been seen, i.e., does not have any record in the database, then a new line is inserted; otherwise, only the `Last_Time_Found` of the device is updated to the current timestamp. An example of how data is stored in the sensor's local database is shown in Figure 4.3. The explanation of each attribute of the tables is presented below.

Both *Data_Packets* and *Probe_Requests* tables have the same 5 attributes:

1. **Frame_Type** - identifies the packet type (Data Packet or Probe Request);
2. **ID** - the unique identifier of the device, and primary key of both tables;
3. **First_Record** - timestamp that indicates when the device was first detected;
4. **Last_Time_Found** - timestamp that indicates when the device was last detected;
5. **Manufacturer** - the device's manufacturer. It is obtained from the OUI of the device's MAC address, as shown in Figure 2.2. For real MAC addresses, the manufacturer is always obtained, while for virtual MAC addresses the manufacturer is 'Unknown'.

The ID attribute needs to be emphasized, since it can originate in different ways, according to the packet type on which the device is detected. In the case of data packets and probe requests with real MAC addresses, the ID is related to the device's MAC address, since it is sufficient enough for its unique identification, followed by applying a one-way hash function to anonymize the data. On the other hand, in the case of probe request frames with virtual MAC addresses, the ID is related to the device fingerprint generated by applying the fingerprinting technique considering the IEs conveyed in these frames, as also described in subsection 3.2.1.

Data Packet	E92FDC9EC5A00A29	2023-07-20 22:54:00	2023-07-20 22:54:00	Unknown
Data Packet	EC3BDBE18E6B74B	2023-07-20 22:54:00	2023-07-20 23:11:00	Unknown
Data Packet	8B99712459FEF8C9	2023-07-20 22:56:00	2023-07-20 23:12:00	Unknown
Data Packet	2271790E51EF76B2	2023-07-20 22:56:00	2023-07-20 23:12:00	Apple, Inc.
Data Packet	79DA88001AE31420	2023-07-20 22:56:00	2023-07-20 23:11:00	Hon Hai Precision Ind. Co.,Ltd.
Data Packet	E768E46392BEE78D	2023-07-20 22:58:00	2023-07-20 23:08:00	Unknown
Data Packet	AFA11F35D78A6253	2023-07-20 22:58:00	2023-07-20 22:58:00	Unknown
Data Packet	5C3CFD47F0A6400E	2023-07-20 23:08:00	2023-07-20 23:08:00	Unknown
Data Packet	4B9CF13C6286356E	2023-07-20 23:08:00	2023-07-20 23:08:00	LG Innotek
Data Packet	741441AF4B6C4CC1	2023-07-20 23:08:00	2023-07-20 23:08:00	AzureWave Technology Inc.
Data Packet	BD0E71080B23D537	2023-07-20 23:08:00	2023-07-20 23:12:00	Unknown
Data Packet	48E54209CC2DA133	2023-07-20 23:08:00	2023-07-20 23:12:00	Xiaomi Communications Co Ltd
Data Packet	9966427BFF7548A7	2023-07-20 23:08:00	2023-07-20 23:30:00	Unknown
Data Packet	3E723B609B1AF3C	2023-07-20 23:30:00	2023-07-20 23:30:00	Unknown
Probe Request	B1AB1479823DC14D	2023-07-20 22:55:52	2023-07-20 22:55:52	Google, Inc.
Probe Request	F39E118ED1E7E5E1	2023-07-20 22:56:12	2023-07-20 22:56:12	Samsung Electronics Co.,Ltd
Probe Request	DADF5E344A5CD3AD	2023-07-20 22:56:38	2023-07-20 22:56:38	Unknown
Probe Request	74D2F2028E353981	2023-07-20 22:56:42	2023-07-20 22:56:43	Unknown
Probe Request	DD52F096E6C86503	2023-07-20 22:57:06	2023-07-20 22:57:06	Xiaomi Communications Co Ltd
Probe Request	A25BF0A246650224	2023-07-20 22:57:11	2023-07-20 22:57:11	Apple, Inc.
Probe Request	2E32F5695A1F98CF	2023-07-20 22:57:13	2023-07-20 22:57:14	Unknown
Probe Request	60A2CE21307F065D	2023-07-20 22:57:15	2023-07-20 22:57:15	Xiaomi Communications Co Ltd
Probe Request	F67CE218C0293586	2023-07-20 22:57:40	2023-07-20 23:08:11	Unknown
Probe Request	CC874E0D9D08EA72	2023-07-20 22:57:54	2023-07-20 23:30:22	Unknown
Probe Request	D8101D4F93606B02	2023-07-20 22:57:57	2023-07-20 22:58:07	Samsung Electronics Co.,Ltd
Probe Request	1C782B64510E8A52	2023-07-20 23:06:40	2023-07-20 23:06:40	Unknown
Probe Request	CD5EEB6AE84530E	2023-07-20 23:06:51	2023-07-20 23:30:02	Google, Inc.

Figure 4.3: Example of device records stored in the sensor’s local database.

As a result, the ID attribute will always have anonymized data, not containing sensitive data like the device’s MAC addresses that could be used for tracking, thus protecting user privacy, as all IDs can not be traced to the original devices due to the one-way hash function applied.

Also as already explained in subsection 3.2.1, devices that are captured in both data packets and probe requests, are only accounted for once. However, this can only be applied for data packets and probe requests with real MAC addresses, as its fingerprints are purely based on the MAC address. For accomplishing the same for probe requests with virtual MAC addresses, the *Probe_Requests* needed to have an additional field to store the randomized MAC address of these messages, for matching those with the ones from data packets.

In addition, the *DataRetentionManager*, seen in Figure 3.1, is responsible for clearing unnecessary data from this local database that will be no longer used by the sensor. As so, this component will only retain the useful information that is necessary at the time the sensor calculates the number of devices, and delete the remaining information that is no longer necessary for this purpose, thus also contributing to user privacy.

4.3 Wi-Fi detection

4.3.1 Hardware selection

For performing Wi-Fi detection the only equipment needed in terms of hardware is a Wi-Fi card that can support monitor mode. The monitor mode allows the board to passively listen to all the Wi-Fi traffic in its proximity. Therefore, the only requirements for choosing the best Wi-Fi card for the STToolkit sensors were the ability to support monitor mode, a good detection

range, and the ability to listen to traffic in both 2.4 GHz and 5 GHz bands, as they are the most common Wi-Fi bands supported by mobile devices.

Table 4.1 summarizes the several Wi-Fi cards available for detecting devices that support monitor mode and highlights the ones selected for the STToolkit.

Table 4.1: Wi-Fi cards available for detecting devices.

Wi-Fi card	Chipset	Antennas	Standards	Frequency
TP-Link TL-WN722N V4 ¹	Atheros AR9002U	1 x 4 dBi	802.11 b/g/n	2.4GHz
Panda Wireless PAU09 N600 ²	Ralink RT5572	2 x 5 dBi	802.11 ac/a/b/g/n	2.4GHz / 5GHz
Alfa Network AWUS1900 ³	Realtek RTL8814AU	4 x 5 dBi	802.11 ac/a/b/g/n	2.4GHz / 5GHz
Alfa Network AWUS036H ⁴	Realtek RTL8187L	1 x 5 dBi	802.11 b/g	2.4GHz
Alfa Network AWUS036NHA ⁵	Atheros AR9271	1 x 5 dBi	802.11/b/g/n	2.4GHz
Alfa Network AWUS036AC ⁶	Realtek RTL8812AU	2 x 5 dBi	802.11 ac/a/b/g/n	2.4GHz / 5GHz
Alfa Network AWUS036ACM ⁷	Mediatek MT7612U	2 x 5 dBi	802.11 ac/a/b/g/n	2.4GHz / 5GHz
Alfa Network AWUS036ACH ⁸	Realtek RTL8812AU	2 x 5 dBi	802.11 ac/a/b/g/n	2.4GHz / 5GHz

As can be seen, three Wi-Fi boards were selected and tested for the STToolkit's sensors, namely, the Alfa Network AWU036AC, the Alfa Network AWU036ACM, and the Alfa Network AWU036ACH. All three Wi-Fi card models have a USB port to connect the Wi-Fi dongle and have [Reverse Polarity - SubMiniature version A \(RP-SMA\)](#) connectors to attach and detach its antennas.

The Alfa Network AWU036AC provides high performance at a low cost, having two antennas for dual-band detection - one for the 2.4 GHz band, and another one for the 5 GHz band, both with a 5 dBi high gain, which provides a great detection range without interfering with Bluetooth devices that are also relevant to detect. This board was used in the field experiments conducted at Iscte's Campus and Pena Palace, both further explained in detail in Chapter 5.

In addition, we also tested the functionality of the recent Alfa Network AWU036ACM

¹<https://www.tp-link.com/support/download/tl-wn722n/>

²<https://www.pandawireless.com/pandaN600ant.htm>

³<https://www.alfa.com.tw/products/awus1900>

⁴<https://www.alfa.com.tw/products/awus036h>

⁵<https://www.alfa.com.tw/products/awus036nha>

⁶<https://www.alfa.com.tw/products/awus036ac>

⁷<https://www.alfa.com.tw/products/awus036acm>

⁸<https://www.alfa.com.tw/products/awus036ach>

and Alfa Network AWU036ACH Wi-Fi cards for the sensors, both successors of the Alfa Network AWUS036AC. Also, by doing this, we aim to provide several alternatives for users to choose the best cost-effective Wi-Fi card according to their needs. The AWUS036AC and the AWUS036ACM have the exact same characteristics, except for the chipset used, and the AWUS036AC and the AWUS036ACH share exactly the same specifications. Nevertheless, it is known that the AWUS036ACH takes advantage by having a power amplifier, thus leading to a higher detection range among the others. The latter was also tested in the field experiment conducted at Pena Palace, presented in Chapter 5.

Also, another important aspect to take into consideration is the detection field of the sensor. This field can be controlled by the radiation pattern of the antennas. As it is intended that each sensor detects the devices in its proximity, the most straightforward option is to choose an omnidirectional antenna, which radiates equal radio power in all directions. However, in some cases, it may be intended to only detect the number of people present in a specific direction, instead of all directions. A good example of this case is a scenario where a museum wishes to determine the number of tourists within a room near the museum entrance, without knowing the number of people who are in the waiting queue at the museum entrance. For this, the sensor should only detect devices inside the room, and not detect devices in all directions, which could lead to overcounting estimations. These possible use case scenarios were also taken into account in the STToolkit. For these cases, a directional antenna is the best option, where is possible to direct the radiation pattern to a specific direction, and thus only detect devices in that direction. Table 4.2 summarizes the directional antennas available for detecting devices in only a specific direction and highlights the selected one for the STToolkit. The criteria used for choosing the best directional antenna was the detection range, which is usually determined by the antenna gain, and the ability of dual-band detection in both 2.4 GHz and 5 GHz Wi-Fi bands.

Table 4.2: Directional antennas available for detecting devices in only specific directions.

Antenna	Frequency	Antenna Gain	Beamwidth	Connector
Alfa Network APA-M04 ⁹	2.4GHz	7 dBi	(Not specified)	RP-SMA Male
Alfa Network APA-L2458- 08A ¹⁰	2.4GHz 5GHz	8 dBi (2.4 GHz) 8 dBi (5 GHz)	40° (Vertical) 90° (Horizontal)	N-Female
Alfa Network AC600U ¹¹	2.4GHz 5GHz	8 dBi (2.4 GHz) 8 dBi (5 GHz)	(Not specified)	USB
Alfa Network APA-M25 ¹²	2.4GHz 5GHz	8 dBi (2.4 GHz) 10 dBi (5 GHz)	16° (Vertical) 66° (Horizontal)	RP-SMA Male

⁹<https://alfa-network.eu/alfa-apa-m04>

¹⁰<https://alfa-network.eu/apa-l2458-08a>

¹¹<https://alfa-network.eu/alfa-panel-outdoor-antenna-ac600u>

¹²<https://alfa-network.eu/apa-m25>

The Alfa Network APA-M04 is an indoor directional antenna. It has a 7 dBi directional antenna, however, it only works on the 2.4 GHz Wi-Fi frequency band. The Alfa Network APA-L2458-08A is a low-profile dual-band outdoor directional antenna. It works on both 2.4 GHz and 5 GHz bands, with an 8 dBi gain in both, and has a horizontal beamwidth of 40 degrees and a vertical beamwidth of 40 degrees. The Alfa Network AC600U is an outdoor panel antenna. It also works on both Wi-Fi bands, with an 8 dBi gain. Nevertheless, this antenna has a USB connector, which makes it incompatible with the RP-SMA connectors of the Wi-Fi cards.

The Alfa Network APA-M25 was the chosen directional antenna for the STToolkit. It works on both 2.4 GHz and 5 GHz Wi-Fi bands, with a gain of 8 dBi for 2.4 GHz, and 10 dBi for 5 GHz, with a horizontal beamwidth of 16 degrees and a vertical beamwidth of 66 degrees, which is perfectly feasible for restricting the detection of mobile devices in only one specific direction. Also, this antenna has an RP-SMA connector, which is compatible with the Wi-Fi card connectors.

As a result, the STToolkit also offers the possibility of choosing between an omnidirectional antenna, for detecting devices in all sensor vicinity, or directional antennas, for detecting devices in specific directions, according to the requirements of each target location where the sensors can be deployed.

4.3.2 Software selection

After deciding on which hardware to employ for the STToolkit sensors, the next step aims at choosing the software for detecting mobile devices through Wi-Fi detection. For this purpose, there are several alternatives to choose from, with different focus and approaches. Some of them use an approach of writing captured data into an output file for further analysis, while others use real-time detection and data output. Table 4.3 summarizes known software programs capable of detecting devices through Wi-Fi and highlights the chosen program for the STToolkit sensor.

Table 4.3: Software programs available for detecting devices through Wi-Fi detection.

Program	Open-source	Community
Trackerjacker ¹³	Yes	Small
Kismet ¹⁴	Yes	Medium
Netsniff-ng ¹⁵	Yes	Medium
Wireshark ¹⁶	Yes	Large
Aircrack-ng¹⁷	Yes	Large

Trackerjacker is a security tool that can be used for mapping Wi-Fi networks. It allows mapping and tracking devices using the 802.11 protocol, and can also be used for penetration

¹³<https://github.com/calebmadrigal/trackerjacker>

¹⁴<https://www.kismetwireless.net/>

¹⁵<http://netsniff-ng.org/>

¹⁶<https://www.wireshark.org/>

¹⁷<https://www.aircrack-ng.org/>

testing such as de-authentication attacks. The tool comes with plugin support so that it can interact with other tools, however, it has a small community support.

Kismet is a specialized program for detecting devices in Wi-Fi networks and has several features for detecting Wi-Fi beacons and probe requests. It is available open-source, meaning that it could be properly customized to write the captured data directly into the sensor's local database. Kismet also has fairly good community support.

Another software program is Netsniff-ng. This software is developed in C-language and is a high-performance Linux network sniffer for packet inspection. It can be used for protocol analysis, reverse engineering, and network debugging. It also has fairly decent community support.

Another software considered was Wireshark, a well-known and broadly used program for capturing data and analyzing packets and messages exchanged, being generally used for debugging situations and analysing protocol behaviour. It can also be used to detect Wi-Fi packets, including data packets and probe requests, and is also open-source, with a large community of users.

Finally, there is also Aircrack-ng, a well-established open-source tool, fully developed in C-language, and used for several types of attacks on Wi-Fi networks and capturing data from Wi-Fi devices. Just like Wireshark, it has a very large community and support, can detect Wi-Fi packets, including data packets and probe requests, and can be customized by the user. In fact, this has been the software program used in the previous work developed by [9] for detecting Wi-Fi devices.

From all these available software programs, the Aircrack-ng was the chosen tool for detecting Wi-Fi devices. It is the most straightforward option to detect Wi-Fi devices and store the captured data in the sensor's local database, because of its large community and documentation, the ease of its installation, and also due to being the same software program used in our previous work.

As already mentioned, Aircrack-ng has several different applications for performing several types of Wi-Fi attacks and detecting devices. In particular, two of these applications are used, namely the *airmon-ng* and the *airodump-ng*. The *airmon-ng* is used for enabling the monitor mode in the Wi-Fi board, and the *airodump-ng* is used to perform device detection, and are both used by the *WifiCrowdingDetector*, seen in Figure 3.1, the component responsible for detecting Wi-Fi devices.

Furthermore, it is important to mention that the *airodump-ng* was properly customized for the implementation of the algorithm for Wi-Fi detection, already described in subsection 3.2.1. This application was purposefully configured to detect and parse probe requests and data packets, the only Wi-Fi packets relevant for device counting. The fingerprinting technique was implemented by parsing each IE of probe requests. The specific bytes and bits that should be ignored for each IE, are referred to as the *Information Mask List* in subsection 3.2.1 and also presented in Table 4.4, were also hard-coded in the *airodump-ng*, where the mask '0' was applied for the specific bytes or bits of each IE of probe requests, for ignoring the specific information that should not be considered for the device fingerprint. Also, the SQLite C-language API was used in this application, for inserting the captured data directly into the *DetectionsDB*, the sensor's local database.

4.3.3 Information for device fingerprinting

As already mentioned in subsection 3.2.1, a fingerprinting technique was developed to tackle the MAC Address randomization process performed by mobile devices, in order to uniquely identify them by a fingerprint, based on the IEs conveyed in probe requests frames. This subsection aims to present all considerations and choices to determine which information from the IEs should be considered for this fingerprinting technique. For this purpose, a dedicated software program was developed to obtain the *Information Mask List*, with the specific bytes or bits of each IE that should be ignored for the fingerprint. The explanation of each step to determine the information used for device fingerprinting is described below.

Firstly, it was needed to determine which IEs should be considered for probe requests. For this, all possible IEs in probe requests conveyed were considered as the best option, as the more information we can extract from these messages, the better the chances of creating a unique fingerprint for each device. If only the more common IEs conveyed in probe requests were considered, it would be more likely that the same fingerprint could be generated for distinctive devices, as less information would be considered for its generation.

As so, all IE IDs from probe requests needed to be discovered for the device fingerprint. The latter was achieved in two separate ways: (1) regarding the work by [33], and (2) from a large dataset of probe requests collected during a typical working day in a large study room at the Iscte's campus. The research done by [33] presents the IE IDs that have been observed in a two-hour-long capture and that were used at least 5% of the total number of frames by devices. From this, we could already obtain the majority of IE IDs present in the probe requests from this work. Additionally, a similar procedure was also performed in a large study room at the Iscte's campus, where probe requests were collected for three hours in two working days. After analysing the probe requests from (1) and (2), all IE IDs conveyed in probe requests were known, which are the ones shown in Table 3.1.

After discovering all IE IDs that can be conveyed in probe requests, the next step regarded determining which information from these IEs was valuable enough to consider for device fingerprinting, for instance, should one only consider the IE IDs and the IE Lengths, just like [33] did, or consider all information from IEs. Once again, the same principle has been followed: the more information to consider, the better.

As so, and contrary to other similar approaches, all information from IEs was considered, namely the IE ID, IE length, and Value bytes. For those IEs with substantially varying values across probe requests emitted from the same device (e.g., DS Parameter Set), only the IE ID and the IE Length were considered. However, from the data collected in the Iscte's study room, it was noticed that the values of some other IEs also varied, but only in just some particular bytes, or even bits, and not the entirety of the IE Value. This behaviour was crucial to address since we aim to have only one device fingerprint for each device, which can be compromised by these particular variations. This problem could be rapidly solved by removing all these particular variations as long as they are detected. However, such a method could lead to considering only a small portion of the IE values for the fingerprint, which could then lead to generating the same fingerprint for different devices, thus leading to undercounting and inaccurate crowd counting. As so, a compromise between not considering the entirety of values from the IEs,

which could lead to generating multiple fingerprints for the same device, and also not removing every particular information of values that varies, which could lead to generating the same fingerprint for different devices, had to be established.

For this purpose, a dedicated software program was developed, as we refer to the *Information Elements Automatic Analyser (IEAA)* tool. As its name indicates, the IEAA tool will automatically analyse the IEs conveyed in probe requests from a dataset, and retrieve the bytes and bits that vary between probe requests with the same MAC address. The IEAA tool was developed fully in Python and was purposely created for facilitating and helping with the analysis of the IEs of probe requests, to determine which information should be considered for each IE, and what particular information of the IEs values should be ignored, having in mind the established compromise for the fingerprint generation. As so, the IEAA tool contributed to the determination of the *Information Mask List*, the list with the information of each IE that should be ignored, and hard-coded in *airodump-ng*, the software detection program of the STToolkit's sensors.

In addition, it is important to acknowledge how the IEAA tool analyses the IEs of probe requests from the same MAC address. The IEAA tool individually analyses all IEs that vary for all the fingerprints generated from each MAC address and retrieves the set of bytes and bits that vary for each IE. However, the same IE can have different lengths in different fingerprints. A good and common example of this can occur with the Supported Rates IE, where the device can specify in different fingerprints a smaller or larger set of bit rates supported by the device. Nevertheless, the value information of each IE is normalized, i.e., each byte/bit has an organization and a meaning on each position of the IE. Considering this, the IEAA tool only analyses up to the minimum common length for all respective IEs from all fingerprints from the same MAC address.

To determine which variations from the values of each IE should be ignored, two parameters were determined: (1) the percentage of variation of the bit, obtained from the number of probe requests that conveyed that specific bit and the total number of probe requests with that MAC address; and (2) the minimum number of total probe requests with the same MAC address. These parameters were necessary to determine what bytes/bits revealed substantial variations from probe requests emitted from the same MAC address, which should be removed from the fingerprint generation, and only retaining the ones that showed sporadic variations. For instance, a bit from an IE that had the value '1' in one of one hundred probe requests for the same MAC address in the dataset, and the value '0' in the remaining ninety-nine probes (corresponding to a percentage variation of 1% for the value '1', and 99% for the value '0'), should not be ignored for the fingerprint, as this variation only occurred in 1% of a total of one hundred probe requests. On the other hand, a bit with a percentage variation of 50% for the same one hundred probe requests should be ignored, as this bit has the same probability of being '0' or '1'. By accomplishing this, we aim to ensure the compromise of not having a fingerprint not so generalized, but also that would not consider specific value bytes or bits of IEs that are very susceptible to variation.

To determine what information of each IE should be considered for the fingerprinting technique, the public dataset of [28] was used for this purpose. This dataset contains labelled device probe requests in different modes. It has a total of 69701 probe request frames from

twenty-two mobile devices in six different modes considering three aspects, namely (i) if the screen of the mobile devices was active or not, (ii) if the Wi-Fi interface was switched on, and (iii) if the power saving mode was active. From the total number of probe requests captured from the twenty-two devices, 49893 of them were from devices with MAC Address randomization, comprising a total of 72 % of packets with virtual MAC addresses from seventeen devices.

The IEAA tool analysed the probe requests from this dataset, with the following parameters:

- Percentage of variation: > 10%
- Minimum total number of probe requests for the same MAC address: 20

From this, with the use of the IEAA tool, besides the IEs that already showed substantial variations across probe requests emitted from the same device, such as the DS Parameter Set, the information considered and removed for each IE for creating a device fingerprint is presented in Table 4.4.

Table 4.4: Information of each Information Element considered for device fingerprinting.

Information Element	Information Considered			Bits/Bytes Removed
	IE ID	IE Length	ID Value	
Supported Rates	Yes	Yes	Yes	None
Extended Supported Rates	Yes	Yes	Yes	None
DS Parameter Set	Yes	Yes	No	None
HT Capabilities	Yes	Yes	Yes	5th byte
VHT Capabilities	Yes	Yes	Yes	None
RM Enabled Capabilities	Yes	Yes	Yes	None
Interworking	Yes	Yes	Yes	None
Vendor Specific	Yes	Yes	Yes	6th and 8th bytes

As shown in Table 4.4, with the use of the IEAA tool applied to the public dataset of [28], the 6th and 8th byte of the Vendor Specific, and the 5th byte of the HT Capabilities were determined as highly-variable, and thus not considered for fingerprinting. The validation of the fingerprinting technique, after determining the information of each IE that should be considered and ignored for the device fingerprint, is further described in section 5.2.

4.4 Data Upload, Data Management and Tasks Automation

4.4.1 Data Upload

For the crowding data upload, either via Wi-Fi or LoRaWAN, the two alternatives to be offered for the STToolkit, a software program was created for this purpose. In fact, this program is related to the *DetectionEngine* component, presented in Figures 3.1 and 3.2.

The program was developed in Python and analyses the information contained in the sensor's local database, and, as already explained in subsection 3.2.1, will periodically retrieve

the number of detected devices within a sliding window of X minutes every Y minutes. A 5-minute period was chosen for the data sampling rate, as it is a sufficient time period for providing near-real-time data availability, and also for detecting the mobile devices near the sensor. The same 5-minute period was chosen for the sliding window, i.e., the sensors will send in each crowding measurement the number of devices detected within the last 5 minutes, so that each crowding measurement comprises all detected devices until the next measurement is sent. The sliding window period is specified as an argument in this program, and the data sampling rate is specified on a cronjob, which allows the execution of the program on predetermined time schedules. This cronjob is further described in subsection 4.4.3.

This program is fully responsible for the data upload of the sensor and has three input arguments: (1) the device name, defined by the user; (2) the protocol used for the data upload (Wi-Fi or LoRaWAN); and (3) the sliding window period. The first is required for matching the crowding measurements sent by the device to the respective sensor, the second to indicate the sensor which protocol it should use for uploading data, and the third indicates the sliding window period.

Usually, the upload via Wi-Fi is sufficient enough to ensure constant connectivity to the cloud server. Nevertheless, if unstable Wi-Fi connectivity is expected at the deployment location of the sensor, a Wi-Fi connectivity test can be performed, where the sensor tries to send the information via Wi-Fi and, in case of failure, the user can eventually manually switch to the upload via LoRaWAN protocol.

Also regarding the upload via LoRaWAN protocol, the LoRa network chosen was the Helium network. Helium is the fastest-growing IoT network with LoRaWAN compatibility. It provides a large coverage in many countries in Europe, such as those involved in the RESETTING project. Figures 4.4 and 4.5 show, respectively, the [Helium network coverage](#) in Europe and in cities where several RESETTING project partners are based.



Figure 4.4: Helium network coverage in Europe. The green areas represent the Helium network coverage.

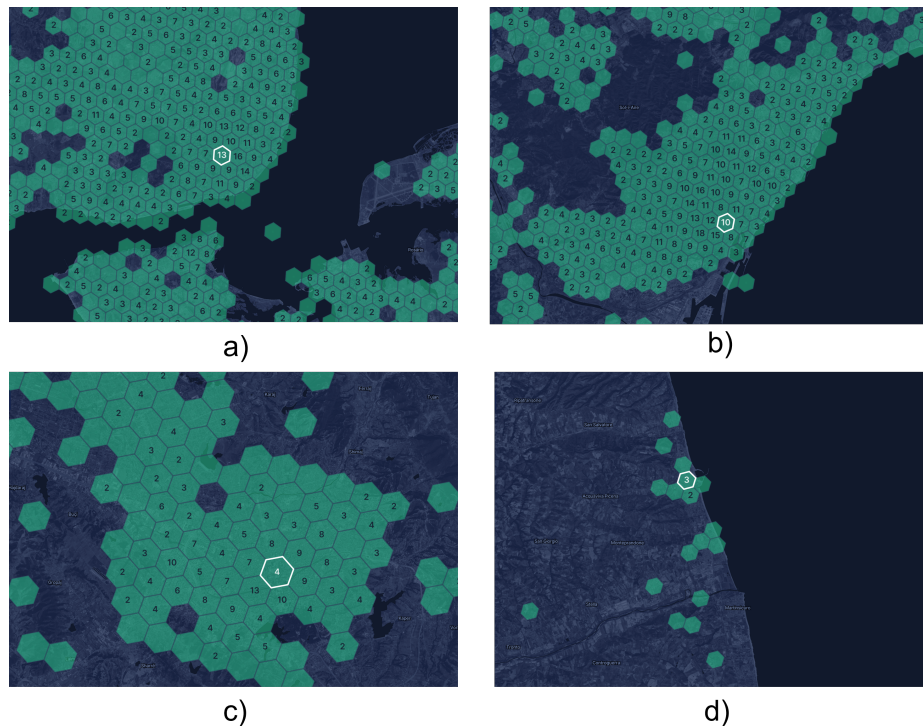


Figure 4.5: Helium network coverage in the cities of: a) Lisbon, b) Barcelona, c) Tirana, d) San Benedetto del Tronto. Each green hexagon represents an area with Helium network coverage.

Regarding the Helium network, users can choose between two alternatives for uploading crowding information: using Helium with a 3rd-party LoRaWAN service provider, such as the [Helium-IoT](#), or using Helium with a private LoRaWAN server.

4.4.2 Data Management

For data management, two Python programs were developed for the STToolkit's sensors.

The first is related to the *DataRetentionManager*, seen in Figure 3.1, and, as already mentioned, is responsible for only retaining the useful and valuable information to calculate the number of detected devices, and removing outdated and unnecessary data. For this purpose, this component periodically deletes from the sensor's local database devices whose last records (*Last_Time_Found* attribute - see Figure 4.2) are older than 30 minutes ago. This 30-minute time period was decided as a compromise between a time period that did not lead to taking much memory of sensors, and also that allowed users to change the data sampling rate of the crowding measurements to some minutes, by having device records until that time period. Nevertheless, this data retention time period can be easily modified by the user, as it is an input argument for this program.

The second program is responsible for updating the *OUI List*, presented in Figure 3.2, obtained from the [Wireshark manufacturer database](#) and used in the mobile device classification task of the algorithm for Wi-Fi detection.

4.4.3 Tasks Automation

For managing all programs of the STToolkit's sensors, the [Crontab](#) tool is used. Crontab allows commands and scripts to be executed on predetermined regular time schedules called cronjobs. The cronjobs are saved in a file called crontab, where users can schedule the periodicity of execution of each command or script. Cronjobs can be configured to schedule scripts or commands to run automatically by using a specific syntax, as follows:

```
a b c d e /directory/command output
```

The first five fields "**a b c d e**" specify the periodicity of the cronjob, to the minimal time granularity of minutes. The next section "**/directory/command**" specifies the the directory and the script to be executed. The third and final section "**output**" is optional and defines how the system notifies the job completion.

As so, Crontab allows the automation of all tasks of sensors, thus not requiring user intervention for its management, such as starting the Wi-Fi detection program, periodically uploading the crowding data to the cloud server, periodically deleting outdated and unnecessary data from the sensors local database, among others.

The tasks simultaneously scheduled for the STToolkit sensors are shown in Table 4.5:

Table 4.5: Tasks simultaneously scheduled in the STToolkit sensors.

Task	Time-Schedule
Execution of the Python program for activating the monitor mode of the Wi-Fi card (<i>airmon-ng</i>) and starting the Wi-Fi detection (<i>airodump-ng</i>).	Sensor Boot
Execution of the Python script for activating and configuring the LoRaWAN upload (if the sensor is equipped with the respective board and antenna for upload via LoRaWAN protocol).	Sensor Boot
Execution of the Python program for analysing the information contained in the sensor's local database, and uploading the number of devices detected to the cloud server, either via Wi-Fi or LoRaWAN protocol.	Every 5 minutes
Execution of the Python program for retaining only the useful and valuable information to calculate the number of detected devices, and remove outdated and unnecessary data (deleting device records whose last time seen is older than 30 minutes ago).	Every hour
Execution of the Python program for updating the <i>OUI List</i> , obtained from the Wireshark manufacturer database.	Every week

4.5 Data Ingestion and Visualization

This section presents the considerations and decisions related to the data ingestion and visualization of the STToolkit on the *CrowdingServer* component, the system's cloud server. The latter is a [Virtual Private Server \(VPS\)](#), hosting all the software to ingest and store the crowding measurements sent by all sensors, and also to provide the uplink services for data visualization.

There are many VPS providers, such as [OpenStack](#), [DreamHost](#), [Contabo](#), and [Kamatera](#), all well-known VPS providers.

In particular, the *CrowdingServer* was hosted on a Virtual Machine on OpenStack for testing and validating the STToolkit, also due to the courtesy of the [National Distributed Computing Infrastructure](#). Nevertheless, any VPS provider is valid for hosting the cloud server of the STToolkit.

4.5.1 Data Ingestion

This subsection describes in detail the data ingestion of the crowding information sent by the multiple sensors of the STToolkit. Firstly, it describes the options and considerations for choosing the main database of the system that will store the crowding data sent by all sensors. Then, it describes the intermediate components, also installed on the cloud server, that route the information to the main database. Finally, it describes the data structure of the crowding measurements sent by the sensors, and how they are stored in the main database of the STToolkit.

The STToolkit requires a database where crowding measurements sent by all sensors can be stored and retrieved. Its main requirements are to be: (i) able to store large amounts of data in a compressed and memory-efficient manner, (ii) optimized for querying time series data, (iii) compatible with data visualization platforms in order to provide a quick and easy integration, allowing users to observe the crowding information in real-time and (iv) open source.

Concerning the previous requirements, we chose the open-source time-series database [InfluxDB](#), referred to as *CrowdingDB* in [Figure 3.1](#). InfluxDB is widely used in fields such as operations monitoring, application metrics, IoT sensor data, and real-time analytics. It is optimized for fast, high-availability storage and retrieval of time-series data, and also has compatibility with many frameworks for real-time data visualization, which makes it the most straightforward option to choose for storing the results from all sensors. InfluxDB is available in two separate forms: as a cloud-based pay-as-you-go database, or by downloading the database and setting up and configuring everything required for its usage. For cost effectiveness sake, we adopted the second option.

The concept of a table does not exist in an InfluxDB database the same way that it does in a [Structured Query Language \(SQL\)](#) relational database. Relational databases are designed to structure data in rows and columns, while time-series databases, such as InfluxDB, store data over time. As so, all records in time-series databases require a timestamp, in a similar way to a primary key in a relational database. InfluxDB stores data as measurements and not records. Each measurement associates a timestamp to particular data conveyed in fields, and tags are used to add more details about the measurement. There is no limitation to the number of fields and tags that can be conveyed in measurements. Furthermore, InfluxDB databases do not have a normal database schema, where it is possible to alter it over time. Such provided flexibility is feasible for solubility and future upgrades because the information conveyed in measurements may change over time without requiring changing the database schema; simply transmit a new measurement with new fields and tags, and those are available for retrieval. [Figure 4.6](#) shows an example of how the same data is stored in a SQL database and in an InfluxDB database.

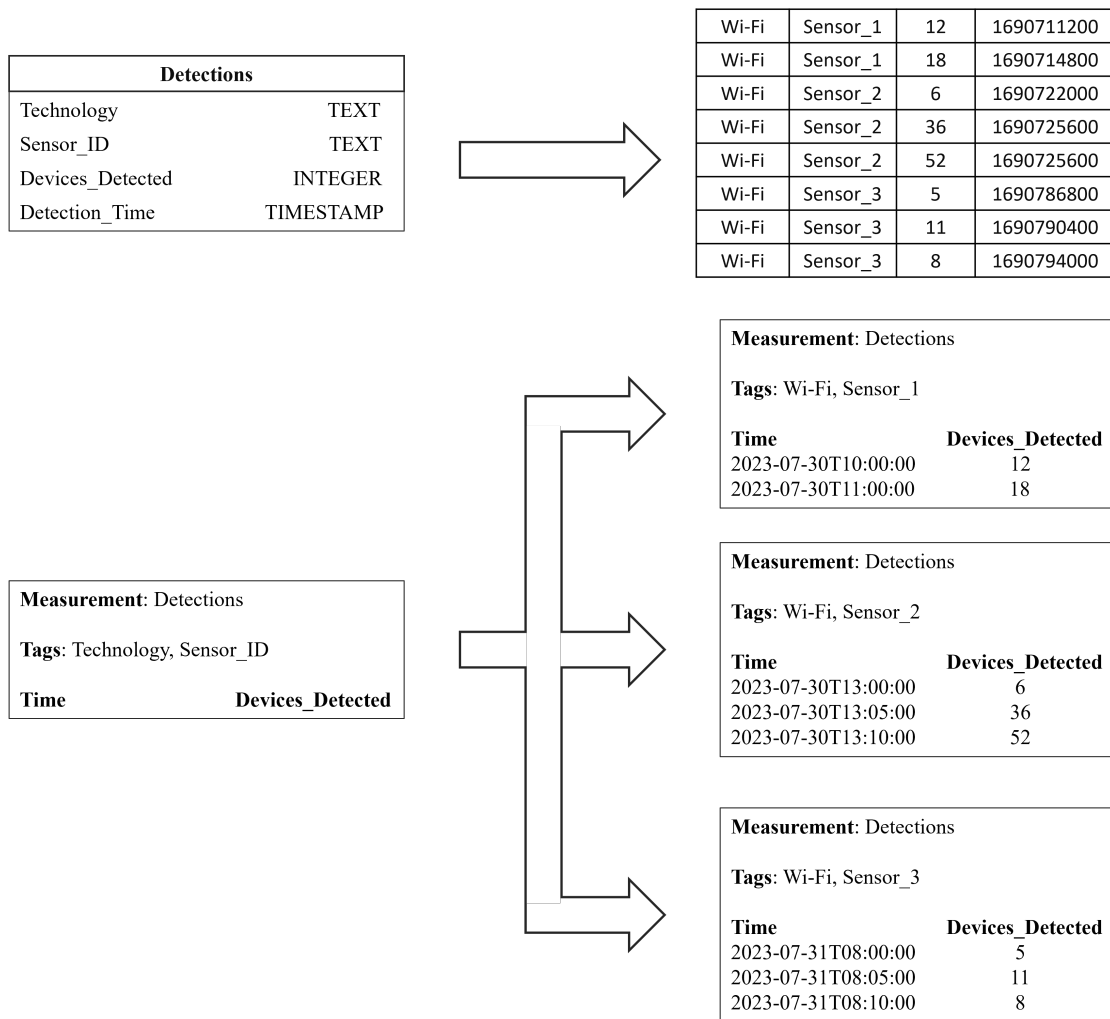


Figure 4.6: Example of data stored in a SQL database (top) and in an InfluxDB database (bottom).

The measurements are stored in a named location called *buckets*, and can store any type of measurement, i.e., it is not required to specify which measurements the bucket is allowed to store. All *buckets* also have a retention policy, where measurements that are older than this are dropped. The retention policy can be set to any time granularity, from the last seconds, hours, days, months, years, or even forever.

In addition, InfluxDB provides a user-friendly graphical [User Interface \(UI\)](#), although a command line interface can also be used. This [UI](#) allows managing organizations, users, buckets, dashboards and API authentication tokens used for secure interaction between InfluxDB and external tools. We used it for the pre-configuration of the *CrowdingDB*.

InfluxDB also provides a built-in [Representational State Transfer Application Programming Interface \(REST API\)](#) for external tools and applications to interact with the database in a simple manner. This built-in functionality facilitates an easier and quicker deployment of this database since the API for inserting and querying data was already specified, where the sole need for using it is a query to the database and the user’s API authentication token. The previous work by [10] used this API for directly entering data to the InfluxDB database, using a curl command with an [HyperText Transfer Protocol \(HTTP\)](#) POST for directly uploading the data to the cloud

server, with the respective fields and tags in its measurements. This uploading method was also used in both field experiments conducted at Iscte's Campus and Pena Palace, further explained in Chapter 5, to upload the crowding information from the deployed sensors to the *CrowdingDB*. This method could also be an option for the production environment of the STToolkit, however, for deployment flexibility, and as already mentioned in section 3.3, all messages sent by sensors will be received at the same point - the *MessageServer* - via MQTT protocol, a publish/subscribe lightweight protocol widely used for IoT applications. For establishing communication via MQTT protocol, the *CrowdingDB* needed to have a MQTT service running on top of it for receiving the crowding measurements from sensors via this protocol. As so, we installed a [Mosquitto MQTT Broker](#) for this purpose, a lightweight open-source message broker that implements MQTT, and thus corresponding to the *Message Server* component. Thanks to this component, the cloud server can now serve as an endpoint for MQTT messages.

As so, when entering data into the InfluxDB database, if Wi-Fi coverage is available on site, the sensors can upload directly the crowding information by sending a MQTT message with only the number of devices detected in the payload of these messages. However, in case Wi-Fi coverage is not available, it will be also possible to upload the information via LoRaWAN protocol. In this case, the sensor sends the number of detected devices to a Helium gateway, that, in its turn, sends it to a Helium server. For sending the information from the Helium server to the cloud server, it is necessary to configure an MQTT integration, which endpoint will be the cloud server. This integration can be easily configured using the [Helium Console](#), a web-based device management tool hosted by the Helium Foundation that allows developers to register, authenticate, and manage their devices on the Helium network, and also provides pre-built connections to route device data via numerous protocols, such as HTTP or MQTT, or directly to cloud services like [AWS IoT](#) - the so-called integrations.

Once the MQTT integration is created, the messages sent by sensors, upon reaching the Helium server, can now be forwarded to the *MessageServer*. For receiving the MQTT messages, the *MessageServer* just has to subscribe to the topic of these messages. Figure 4.7 shows an example of an MQTT integration created in the Helium Console for forwarding the crowding measurements from the Helium network to the *MessageServer*. It requires an endpoint, which in this case is the *CrowdingServer* (the username and password credentials, and the public server [Internet Protocol \(IP\)](#) address and port have been omitted for security reasons), and two topics: one for uplink communication, and another one for downlink communication. The first one has more importance since the main information exchange will be the crowding measurements sent periodically from sensors to the cloud server, and so the communication will mainly be in this direction.

At this stage, the crowding measurements sent by sensors can be forwarded to the *Message Server* via MQTT protocol, but still not to the *CrowdingDB*, the InfluxDB database, for its storage and further visualization. Concerning this, data also needs to be pushed from the *Message Server* to the *CrowdingDB*. For this purpose, we used [Telegraf](#). Telegraf is a server-based agent for collecting and sending metrics and events from databases, systems, and IoT sensors. It is written in Go and compiles into a single binary with no external dependencies and requires minimal memory. Also, Telegraf is developed by InfluxData, the same developing team as InfluxDB, and has a lot of compatibilities and shared features. For instance, Telegraf can be used for collecting

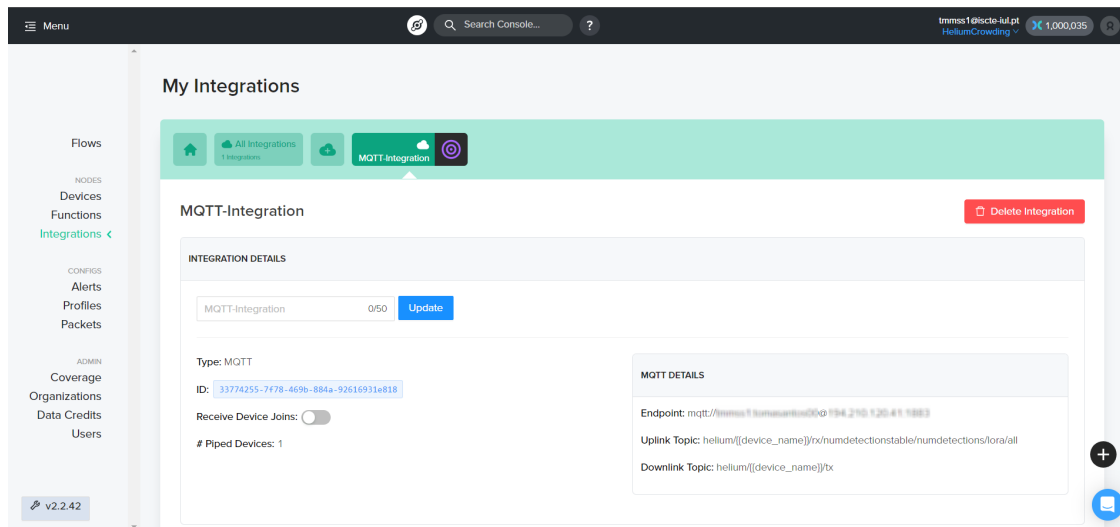


Figure 4.7: MQTT Integration to forward the crowding measurements from the Helium network to the cloud server.

data from other data sources and inserting it into an InfluxDB database. Telegraf agents can be implemented using the InfluxDB UI. As so, a Telegraf agent was implemented in order to collect all MQTT messages received in the *Message Server* and insert them in the correct format into the *CrowdingDB*. The Telegraf agent corresponds to the *CrowdingCollector*, seen in Figure 3.1. As a result, the crowding measurements sent by the sensors are inserted in the *CrowdingDB* in the same format, independently of the communication protocol used for uploading the data (Wi-Fi or LoRaWAN), thus providing channel transparency.

In the *CrowdingDB*, all measurements are sent to a single bucket called *DeviceDetections*, which stores all measurements sent by sensors. The sensors can send two types of measurements to this bucket, being them:

- ***numdetections*** - a measurement with the number of detected devices from the sensor;
- ***sensorLocation*** - a measurement with geographic information of the location where the sensor is currently deployed.

The *numdetections* measurements are used to report the number of devices detected by each sensor. It is through these measurements that each sensor transmits the number of devices present in its proximity to the cloud server.

The *sensorLocation* measurements are used to report where each sensor is deployed. It contains geographic information, the latitude and longitude coordinates, which allows one to spot where the sensor is currently deployed. These two measurements were implemented independently to avoid the constant sending of geographic information with every crowding measurement, which is not necessary since the sensor will always be at the same location once it is deployed at one spot. Instead of this, once the sensor is deployed, it sends a *sensorLocation* measurement, with latitude and longitude coordinates, of the current location where it is deployed. These measurements are also used by the data visualization platform, presented in subsection 4.5.2, to render the location of each sensor on maps. Examples of both measurements sent by the crowding sensors are shown in Figure 4.8.

Measurement: numdetections			
Time	Technology	Device	Devices_Detected
2023-07-30T08:05:00	Wi-Fi	Sensor_1	12
2023-07-30T08:10:00	Wi-Fi	Sensor_1	26
2023-07-30T10:45:00	Wi-Fi	Sensor_2	48
2023-07-30T10:50:00	Wi-Fi	Sensor_2	36
2023-07-30T12:05:00	Wi-Fi	Sensor_3	6

Timestamp
Tags
Fields

Measurement: sensorLocation			
Time	Device	Latitude	Longitude
2023-07-30T08:00:00	Sensor_1	38.7478	-9.1489
2023-07-30T10:40:00	Sensor_2	38.7285	-9.1529
2023-07-30T12:00:00	Sensor_3	38.7121	-9.3094

Timestamp
Tags
Fields

Figure 4.8: Examples of *numdetections* measurements (top) and *sensorLocation* measurements (down) sent by sensors.

As it can be seen in Figure 4.8, the *numdetections* measurements have two tags: Technology and Device. The Technology tag informs the detection technology used by the sensor to detect devices, and the Device tag identifies the sensor that has sent the measurement. It also has the *Devices_Detected* field with the number of detected devices within the sliding window period.

The *sensorLocation* measurements have a Device tag, to identify the sensor, and two fields: Latitude and Longitude, that store the geographic location where the sensor has been deployed. It is also relevant to notice that both measurements share the same Device tag. This tag is used to match the crowding measurements to the locations where each sensor is deployed. Such information is further rendered by the data visualization platform.

4.5.2 Data Visualization and Notification

The platform for data visualization of the STToolkit needed to comply with several requisites, being them: (1) be compatible with the database chosen for the data ingestion, namely, the InfluxDB; (2) provide the creation of dashboards with elements that allowed a good interpretation of the crowding information, such as graphs, maps, and tables with ease; (3) have the ability to show real-time data for analysis; (4) provide creation of notification policies, so that users can be notified of events, for instance, when a crowding threshold is reached; and (5) provide creation of several user profiles, with different visualization permissions and restrictions.

There are several data visualization platforms, such as [Dynatrace](#), [Datadog](#), [IBM Instana](#), and [Grafana](#). All of these provide the ability to show real-time data, the creation of customized dashboards and alerting notifications, and also the possibility of managing several user profiles. However, Grafana is the only platform that is also compatible with InfluxDB.

As so, the chosen data visualization platform was Grafana, an open-source analytics and monitoring tool that is compatible with several data sources, and implements the *VisualizationPlatform* in Figure 3.1. Grafana enables users to query, visualize, alert, and explore metrics that are stored in data sources, including InfluxDB. It allows users to create dashboards with advanced visualizations from data in InfluxDB. For this, users just need to configure the connection with the InfluxDB database and query the data that is then used for input on different graphs, maps, and tables in dashboards, allowing a quick visualization and interpretation of crowding in real-time from sensors with different spatiotemporal levels of granularity. The creation and configuration of dashboards can be done either by the system administrator or even by the final users themselves, as the installation and configuration of this visualization setup is straightforward, without requiring any high expertise. In addition, Grafana also provides extensive and comprehensive documentation for its users.

Dashboards allow users to select time ranges for crowding data temporal visualization, perceive people’s flows and concentration during specified periods of time, compare crowding between locations and also to identify highly populated events. Apart from the temporal rendering of information, the crowding data can also be used for spatial visualization, with maps that highlight the crowding geo-distribution in each location where the sensors are deployed in the form of heatmaps. As already mentioned in section 3.4, the STToolkit provides some pre-configured dashboards for a quick and straightforward data visualization setup.

Furthermore, users can also create notification policies with Grafana, allowing them to be notified in case of events through several contact points, such as e-mail, [PaperDuty](#), or [Slack](#). These notifications are related to the *AlertingEngine* component, presented in Figure 3.1. Figure 4.9 presents a notification policy that can be created to alert users of events.

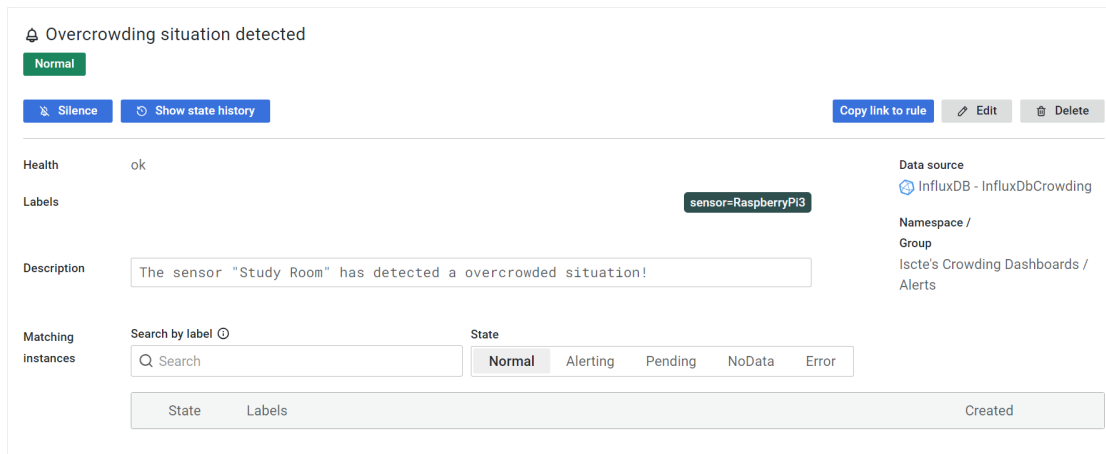


Figure 4.9: Example of a notification policy that can be created in Grafana for alerting users of events.

In addition, other visualization platforms can also be used for visualizing the raw crowding data from the STToolkit. One of those examples is [Unity](#). Through this, the raw crowding data can also be used for achieving a more realistic perception of space occupancy, for instance, by using walking avatars on top of a BIM. An example of this third-party integration is presented in Figure 4.10, where the crowding data collected at Iscte’s Campus was used with a BIM of an entire building floor of the university.

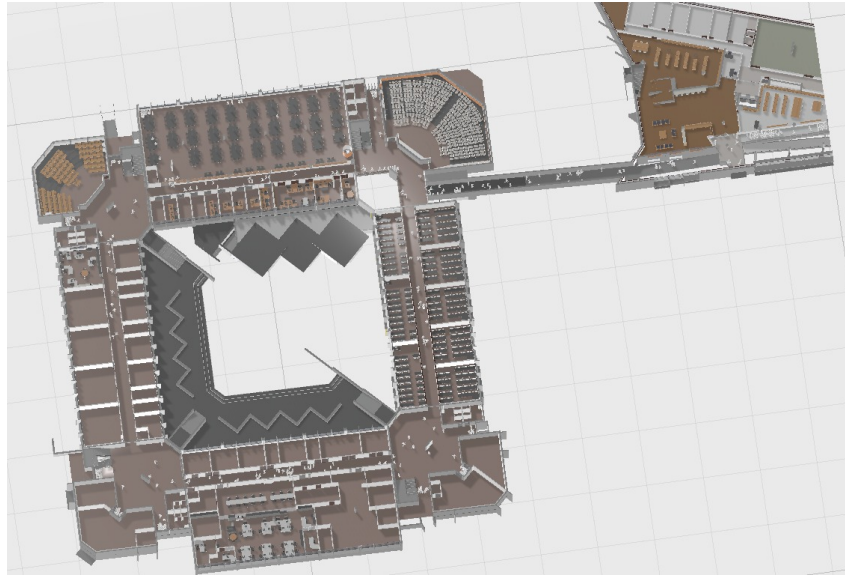


Figure 4.10: Third-party integration implemented with Unity, by using walking avatars on top of a BIM of an entire floor of Iscte.

4.6 Toolkit Prototype

To conduct testing and validation, a sensor prototype was required for the crowding STToolkit. This prototype and its components had to be designed taking into consideration all the requisites for the STToolkit. For grating scalability, the prototype needs to be a low-cost device and also compatible with the tourism-attractive locations where they are intended to be deployed, either in indoor or outdoor scenarios, where overcrowding situations may occur.

Table 4.6 presents the hardware available for the STToolkit sensors, with the respective unit costs and several alternatives for some of its functions, such as [Single Computer Boards \(SCBs\)](#) for the sensor processing unit, and Wi-Fi boards for detecting devices through Wi-Fi technology. Also, even though the Bluetooth detection and the LoRaWAN upload were not implemented in this dissertation, the STToolkit already has hardware for implementing both features. Also, all costs shown in this section were consulted on 4 September 2023.

The hardware selected for each deployment in real crowded scenarios, presented in Chapter 5, are described in the respective sections, namely, Table 5.1 presents the hardware selected for the sensors deployed at Iscte's Campus, and Table 5.8 presents the hardware selected for the sensors deployed at Pena Palace.

The STToolkit prototype sensors also have custom-designed cases adapted to deployment locations, either with exposed antennas or not, as shown in Figure 4.11. The latter can be manufactured in [Vitrurius Fablab - ISCTE](#), an architecture lab on Iscte for knowledge-sharing, manufacturing and innovation, with numerous tools that can be used for creation and construction. The cases can be constructed using a 3D printer or by using a laser-cutting machine, and conceals all hardware components inside of it. In particular, two sensor case versions were designed: a larger version with no exposed antennas, and a smaller version with exposed antennas. The two versions have small dimensions and were designed to fit all the hardware of the STToolkit's sensors. Table 4.7 presents the dimensions and the approximated fabrication cost

Table 4.6: Hardware available for the STToolkit sensors (unit costs from 4 September 2023).

	Function	Equipment	Unit Cost
Computing Resources	Sensor Power	Raspberry Pi 3 Micro-USB Power Supply	8.75 €
		Raspberry Pi 4 USB-C Power Supply	8.65 €
		Banana Pi USB-A Power Supply	7.44 €
	Sensor memory	SanDisk 32GB Micro SDXC High Endurance	12.25 €
	Sensor Processing	Raspberry Pi 3 1GB Model B	39.93 €
		Raspberry Pi 4 1GB RAM	44.44 €
		Banana Pi M2 Pro	81.68 €
Banana Pi M2 Ultra		61.12 €	
Detection	Wi-Fi detection	Alfa Network AWUS036AC	36.95 €
		Alfa Network AWUS036ACM	38.95 €
		Alfa Network AWUS036ACH	57.95 €
	Bluetooth detection	Ubetooth-One	152.78 €
	Detect devices in specific diretions	Alfa Network APA-M25	17.75 €
Upload	LoRa upload	Raspberry Pi IoT LoRa pHAT	29.23 €

of each sensor case version.

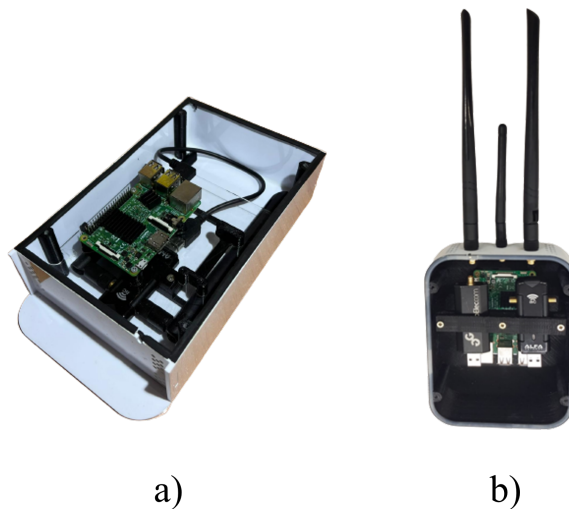


Figure 4.11: Sensor cases. a) Larger version with no exposed antennas. b) Smaller version with exposed antennas.

Furthermore, although not implemented, the communications costs regarding LoRaWAN protocol were also estimated. For the open collaborative The Things Network network, there are no costs associated since this network is free to use and so the sensors can communicate the

Table 4.7: Sensor cases dimensions and approximated unit cost (costs from 4 September 2023).

Sensor Case	Dimensions			Unit Cost (approx.)
	Width	Length	Depth	
Larger version with no exposed antennas	24 cm	13,5 cm	6 cm	90,00 €
Smaller version with exposed antennas	16,5 cm	12,4 cm	5,1 cm	50,00 €

data freely without any cost. The same does not apply to the Helium network. In Helium, data can only be transferred by using [Data Credit \(DC\)](#)s. Data Credits allow users to transfer bytes via Helium and are fixed in USD (1 Data Credit = \$0.00001). To acquire DCs, network users need to convert [Helium Network Token \(HNT\)](#), the Helium cryptocurrency, or obtain them from an HNT owner. Every 24 bytes sent in a packet costs 1 DC. As so, the communications costs associated with this network are feasible for using Helium for uploading data to the cloud server, as only the number of devices detected is sent on the payload of each measurement, which is far less than 24 bytes. Therefore, and considering that every measurement is sent by sensors every 5 minutes, the communication costs of LoRaWAN protocol were leveraged, and presented in Table 4.8. Also, Helium provides a [Helium Data Transfer Calculator](#) to estimate deployment costs based on the number of devices and how often they send data.

Table 4.8: Helium communication costs per sensor (costs from 4 September 2023).

#Sensors	Communication Costs	
	Month	Year
1	0,09 €	0,98 €
10	0,80 €	9,60 €
20	1,61 €	19,32 €
30	2,42 €	29,04 €

Following the trace, the cloud server of the STToolkit also has some costs associated. As already mentioned, the cloud server is hosted by a VPS, providing all resources for this component. Usually, suppliers charge monthly fees for hosting VPS. Table 4.9 shows the hosting costs of two VPS configurations for different providers.

Finally, estimations of the total installation cost for a STToolkit can be estimated based on the previously described partial costs. Table 4.10 presents the installation costs of the STToolkit for different number of sensors, considering sensors with Wi-Fi detection, the LoRa board for upload via LoRaWAN protocol, and the larger sensor case with no exposed antennas, as well as the cheapest VPS pricing, which, from the Table 4.9, is from Contabo.

As can be seen in the table, the first year represents the highest investment, since all the hardware for the STToolkit needs to be acquired, namely the sensors and the cloud server,

¹⁸<https://www.dreamhost.com/cloud/>

¹⁹<https://contabo.com/en/>

²⁰<https://www.kamatera.com/>

Table 4.9: Cloud Server Hosting Pricing for different providers (costs from 4 September 2023).

VPS Provider	Cloud Resources	Hosting Cost	
		Month	Year
DreamHost ¹⁸	1CPU + 2GRam	11,00 €	132,00 €
	4CPU + 8GRam	45,00 €	540,00 €
Contabo ¹⁹	1CPU + 2GRam	-	-
	4CPU + 8GRam	6,00 €	72,00 €
Kamatera ²⁰	1CPU + 2GRam	6,00 €	72,00 €
	4CPU + 8GRam	37,00 €	444,00 €

which only have an initial cost. In the following years, the annual cost is reduced substantially, as only the communication and server costs are charged.

Table 4.10: STToolkit total installation costs simulation.

# Sensors	1	10	20	30
Hardware	€131,52	€1 315,20	€2 630,40	€3 945,60
Cases	€90,00	€900,00	€1 800,00	€2 700,00
Communications	€0,98	€9,60	€19,32	€29,04
Server	€72,00	€72,00	€72,00	€72,00
Total = 1st Year	€294,50	€2 296,80	€4 521,72	€6 746,64
Total > 1st Year	€72,98	€81,60	€91,32	€101,04

A similar deployment calculator will be available for the STToolkit, which will allow users to estimate the installation cost of the STToolkit considering all the possibilities provided for the deployment and configuration of this crowding monitoring system.

Additionally, the STToolkit also includes support material for its installation, configuration, operation, and usage such as installation, configuration, operation, and user manuals, online video tutorials and setup images that will allow users to build, operate, and use a STToolkit.

CHAPTER 5.

FIELD EXPERIMENTS

Contents

5.1	Crowding patterns perception	63
5.2	Fingerprinting validation	75
5.3	Crowd counting approach validation	82

This chapter presents the field experiments carried out to test, calibrate and validate the ST-Toolkit.

This chapter is organized as follows: section 5.1 presents the evaluation of the perception of sensors in perceiving crowding trends in a field experiment carried out at Iscte’s Campus, our university campus, section 5.2 presents the results from the developed fingerprinting technique for tackling the MAC address randomization issue from devices, and section 5.3 presents the field experiment carried out at Pena Palace, one of the most iconic tourism sites in Portugal, flattened by overtourism all year round, where the crowd counting approach, already with the fingerprinting technique implemented, was validated.

[This page has been intentionally left blank]

Chapter 5

Field Experiments

In this chapter, we describe several tests that were conducted to calibrate and validate the STToolkit.

First, in section 5.1, the perception of sensors in perceiving crowding tendencies and correlating the detections with the general use of spaces needed to be evaluated, as well as how rapidly they could detect highly-populated events, with a sudden rise in the number of people in a given area. For this, a network of sensors was deployed across Iscte's Campus, in several locations with different crowding patterns, to evaluate how the sensors could correlate the detections with the general use of spaces around the university.

The fingerprinting technique also needed to be validated. For this, labelled datasets of probe requests were used to determine how effectively and accurately the developed fingerprinting technique tackles the MAC address randomization process of mobile devices. A public dataset of labelled devices and a dataset collection at Iscte's garage were conducted for this validation. The results from the fingerprinting validation are presented in section 5.2.

Finally, section 5.3 presents the validation of the crowd-counting approach, with the final Wi-Fi detection algorithm implemented, including the fingerprinting technique. For this, a field experiment was carried out at Pena Palace to evaluate the correlation of the detections with the real number of people during a public event that occurred in Pena Park, a large park that surrounds the Pena Palace, as an incentive action for visitors, to allow tourism managers measure its impact.

5.1 Crowding patterns perception

A field experiment was conducted at Iscte's Campus, to test and validate the STToolkit architecture. The aim of this field experiment was to evaluate if the sensors allowed the perception of crowding tendencies according to the general use of each space where they were deployed. As so, this field experiment focused on seeking several crowding patterns that can occur on the university campus, such as time breaks between classes, lunch periods, as well as highly-populated events, and how quickly the sensors are able to detect them. The latter is extremely important to achieve, given the tourism sites where the sensors are intended to be deployed, where overcrowding situations can quickly occur, It is necessary to detect these situations as quickly as they arise.

In addition, the dashboards created for the crowding data visualization in the several locations where the sensors were deployed were also tested. It was evaluated whether they provided a quick and intuitive visualization for end users, as well as if it enabled them to freely and quickly inspect the collected data and perceive crowding patterns.

As so, the main objective of this field experiment was to assess the perception of sensors in perceiving crowding patterns and trends throughout the day, rather than the accuracy regarding the real number of people and the number of detections from sensors in each space. The accuracy is further addressed in section 5.3.

5.1.1 Deployment description

A network of sensors was placed at several spots on Iscte's Campus, and crowding information was collected for one-year long, from September 2022 until September 2023. The sensors performed simultaneously 24/7 real-time detection and periodically uploaded the number of devices detected in its proximity to the cloud server every 5 minutes. The sensors uploaded data via Wi-Fi, which was possible by connecting them to the university network, provided throughout the campus.

The sensors aimed to be deployed in typical usage scenarios, in both indoor and outdoor environments, with different crowding patterns, so that the STToolkit could be tested in different areas with different crowding behaviours, such as areas with a heavy traffic flow of people, or areas with an extended permanency time. As so, the sensors were deployed in the following scenarios at the Iscte's Campus:

1. A large study room, where the university students stay for long periods of time;
2. The university library, another place where students tend to stay for long periods of time;
3. An indoor passageway connecting the two main campus buildings, extensively used by students, professors, and university staff.
4. An outdoor passageway connecting the two main campus courtyards, extensively used by people who attend the university, especially during time breaks between classes and during lunch times;

Figure 5.1 shows and highlights the deployment of the sensors in each scenario, followed by the selected hardware and upload technology for each sensor presented in Table 5.1.

Table 5.1: Selected hardware and upload option for the sensors deployed at Iscte's Campus.

Location	Processing Unit	Wi-Fi card	Antenna Type	Upload
Study Room	Raspberry Pi 4	Alfa Network AWUS036AC	2 x 5 dBi Omni-directional	Wi-Fi
University Library	Raspberry Pi 3	Alfa Network AWUS036AC	2 x 5 dBi Omni-directional	Wi-Fi
Indoor Passageway	Raspberry Pi 3	Alfa Network AWUS036AC	2 x 5 dBi Omni-directional	Wi-Fi
Outdoor Passageway	Raspberry Pi 4	Alfa Network AWUS036AC	2 x 5 dBi Omni-directional	Wi-Fi

Several metrics were extracted from sensors and sent to the cloud server for further analysis. For the production phase, only one metric will be sent with the number of detected devices to the cloud server.

This field experiment can be divided into two stages, with different approaches for device counting, referred to as First Stage and Final Stage.



Figure 5.1: Deployment of sensors at: a) a large study room; b) the university library; c) an indoor passageway connecting the two main buildings of the campus; d) an outdoor passageway that connects two courtyards of the campus.

In the First Stage, the sensors performed device counting based on the MAC address of any Wi-Fi packet. By doing so, a superior number of detections than the actual number of people present at each location where the sensors were deployed was expected, nevertheless, crowding patterns with correlations to the usage of each deployment area were expected to be perceived. The sensors uploaded three metrics: (1) the number of devices that used random MAC addresses, (2) the number of devices that did not use random MAC addresses, and (3) the number of devices detected in Wi-Fi, as the sum of (1) and (2). The results from the First Stage are described in subsection 5.1.3.

In the Final Stage, the sensors only detected data packets and probe request frames, the packets that showed to be relevant for our device counting approach. It is important to mention that the fingerprinting technique was not implemented at this stage, and so this approach did not take into account the randomization issue from mobile devices. Also, all probe requests were considered in this approach, and so some of these messages could not be only from mobile devices, but also from other devices that also emitted these frames. The metrics periodically uploaded to the cloud server by each sensor at this stage were: (1) the number of devices detected from data packets, (2) the number of devices detected from probe requests, and (3) the number of detected devices in Wi-Fi technology, as the sum of (1) and (2). The results from the Final Stage are described in subsection 5.1.4.

All the data shown in this field experiment was rendered and displayed in real time using

Grafana, the data visualization platform chosen for the STToolkit. Furthermore, and as already referred to in subsection 4.4.1, it is important to keep in mind that the data gathered by the sensors uses a sliding window of 5 minutes, and so each data point represents the number of devices detected within the last 5 minutes.

5.1.2 Data visualization

The gathered information from the network of sensors deployed across the Iscte's Campus should be quickly and easily visualized, for a fast and intuitive interpretation of the crowding phenomena in the target areas where the sensors were deployed. As so, this field experiment also contributed to testing the crowding visualization dashboards that will be further used for the production phase of the STToolkit.

An advanced visualization system was developed to visualize the crowding data collected by the sensors on the campus in a clear and simple manner. The latter was developed using Grafana, the data visualization framework for the STToolkit, and is composed of several dashboards that allow quick visualization of crowding in real-time throughout the campus.

For rendering temporal information, several dashboards were created, with graphs that allow users to perceive people's concentration and flows in specific time periods. At first, a dashboard that gives a collapsed view of each location where the sensors were deployed on the campus was developed. Figure 5.2 presents this dashboard with crowding data collected in a two-week period. The working days are filled in blue, and the weekends are filled in orange.



Figure 5.2: Dashboard with a collapsed view of each sensor deployed at Iscte's Campus in a two-week period. The working days are filled in blue, and the weekends are filled in orange.

As it can be seen in this dashboard, it is possible to easily distinguish between working days of classes and weekends, where working days have noticeably higher amounts of detected devices along each day compared to the weekends, with much lower detections since on these days there are only a few classes on Saturdays.

Furthermore, regarding the rendering of temporal information, it was also intended to give users the possibility of comparing crowding between locations. This would give users a chance to understand where people tend to concentrate compared to other locations, and at which

time those circumstances occur. As so, a dashboard was also created for this purpose. Figure 5.3 presents this dashboard, where it is possible to compare the crowding levels between all sensors deployed at the Iscte's Campus during a normal day of classes in the university.

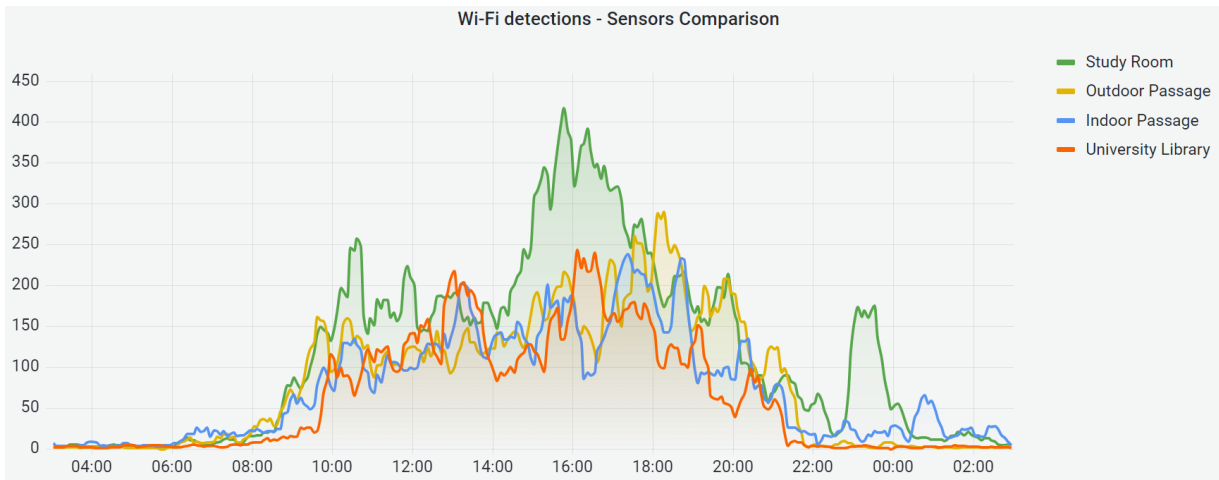


Figure 5.3: Dashboard for comparing crowding between locations at Iscte's Campus.

In addition to temporal visualization, the collected crowding data on the university campus was also used for spatial visualization. For this, a dashboard with a map that highlights the crowding geo-distribution in each location where the sensors were deployed in the form of heatmaps was created, presented in Figure 5.4. Each circle represents the locations where the sensors were deployed on the campus and is accompanied by the name of the deployment scenario and the last crowding measurement sent by each sensor. With this map, users can easily and quickly visualize and understand how crowded each location is on the campus in real time.

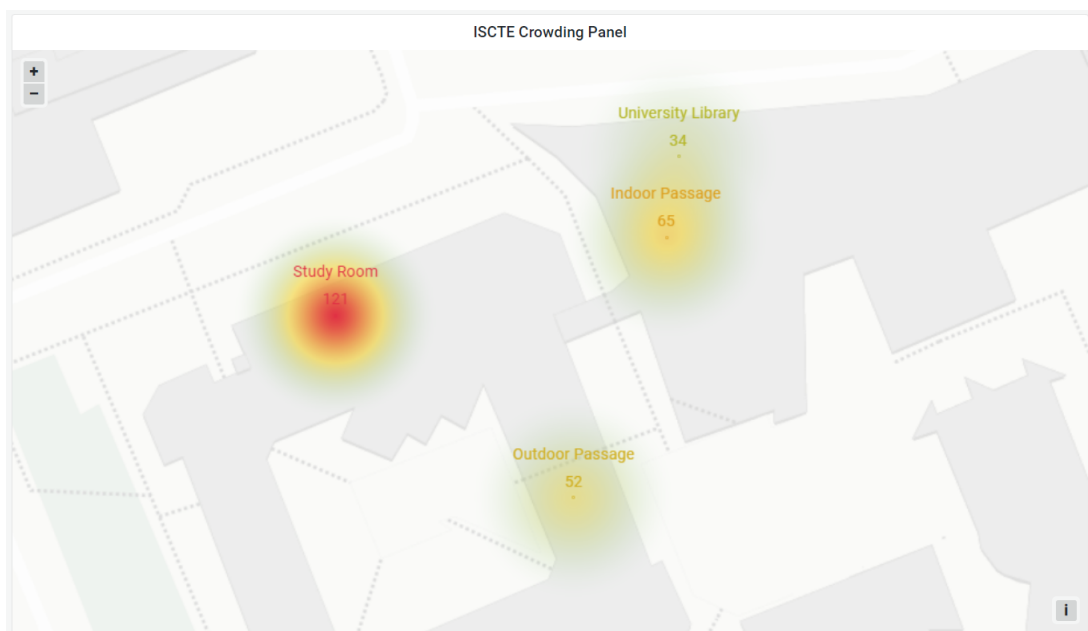


Figure 5.4: Dashboard with crowding geo-distribution at Iscte's Campus.

In order to provide the visualization of the collected crowding data information in a single instance, a final dashboard was developed that comprises all renderings of information, both temporally and spatially. This final dashboard allowed the visualization of all crowding information through one single dashboard, instead of forcing users to move between several dashboards to view the different renderings of information. As so, this final dashboard provided a much easier access and interpretation of the crowding information. Figure 5.5 presents this final dashboard that was used for visualizing the crowding information at the university campus.

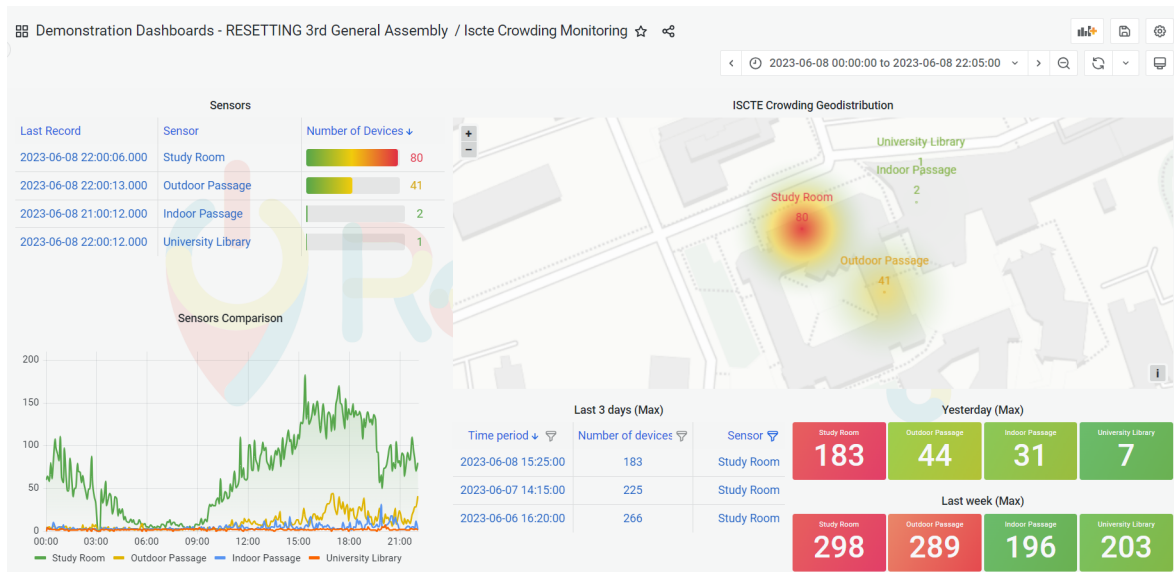


Figure 5.5: Final dashboard created for crowding data visualization of Iscte's Campus.

On the top left corner, the dashboard has a table that indicates the general information about each deployed sensor. It shows the last record sent by each sensor, that can be used to keep track of the sensor operation, the name of the sensor, and also the number of detected devices in its last crowding measurement sent. The top right corner has a map with the crowding geo-distribution in each location for spatial visualization. Following the trace, the bottom left corner has a graph that can be used for comparing crowding between locations for temporal visualization. The time range for this graph is selected on the top right corner of the dashboard and can be specified for any time period by the user with ease. Finally, in the bottom right corner, it has other elements that show users the highest amount of detected devices in three different periods of time: one day ago, three days ago in the form of a table, and seven days ago. These elements provide users a rapid perception of how heavily crowded each location has currently been in different time periods, and avoid the need for users to use the sensors comparison graph for perceiving those metrics.

The dashboards created for visualizing the crowding data collected at several spots of the Iscte's Campus showed that they are capable of providing a quick and easy interpretation of the crowding phenomena. Grafts allows users to visualize the sensor's last records for perceiving crowding tendencies and events during specific time periods, which can be set by the user as they wish. Maps allow users to perceive the crowding geo-distribution in real-time throughout the several locations where the sensors are deployed, and tables allow a quick interpretation

of the general information of the system. As so, this field experiment has demonstrated that the dashboards created enable users to easily, quickly, and non-effortlessly monitor several locations in real-time, and use the collected data to perceive crowding patterns on specific time periods with ease.

5.1.3 First Stage

In the First Stage, the sensors performed device counting based on the MAC addresses of every Wi-Fi packet that was captured.

A time period that comprised normal working days of classes and weekends was intended to evaluate how the sensors were capable of perceiving crowding in both situations. The university is usually much more occupied on working days as students attend the university for their classes, while on weekends the number of people is generally less, as there are only a few classes on Saturdays. In addition, we also intended to test the sensors during a highly-populated event on the university campus to evaluate how quickly the sensors were able to detect its start and duration.

Figure 5.6 presents a snapshot of the crowding data collected in a four-day period at this stage for the four scenarios where the sensors were deployed. The first three days (in blue) are normal working days of classes and the fourth day (in orange) is a Saturday. The same four-day period is presented for the first three scenarios, while a different period is shown for the fourth scenario. The latter corresponds to a period when a highly-populated event took place on the Iscte's Campus on a Saturday.

At first, it is possible to observe the pattern of devices detected along the passage of each day. It can also be noticed immediately that the working days reveal far more detections than on Saturday, except for the sensor deployed at the outdoor passageway. This can be justified by the fact that on working days students are at the university to attend their classes, while at the weekend the university is generally way less occupied, where there are only a few classes on Saturdays.

The collected data reveals a close correlation with the general usage of each space. Students' presence in the study room is generally continuous, however, they tend to use it more often in the morning and in the afternoon, and less at night. As this study room is open 24 hours a day, students can use it during the night, such things that can be visualized, although in lower numbers than other parts of the day.

In the university library, during the first three days of classes, detections start to rise from 9 AM, the library's opening hour, and suddenly drop to almost zero around 9 PM, the library's closing hour. This is an interesting fact that could contribute to validating that the sensors are able to detect rapidly when numerous devices suddenly leave and arrive at the sensor proximity. In addition, during the opening time of the library, the number of devices detected increases approximately every hour and a half. This phenomenon can be related to the time breaks between classes, as all classes have the same standard time duration in the university, where some students arrive at the library.

Concerning the indoor passageway, a narrow passage between the two main buildings of the university campus, and generally characterized by having a high traffic flow of people, the

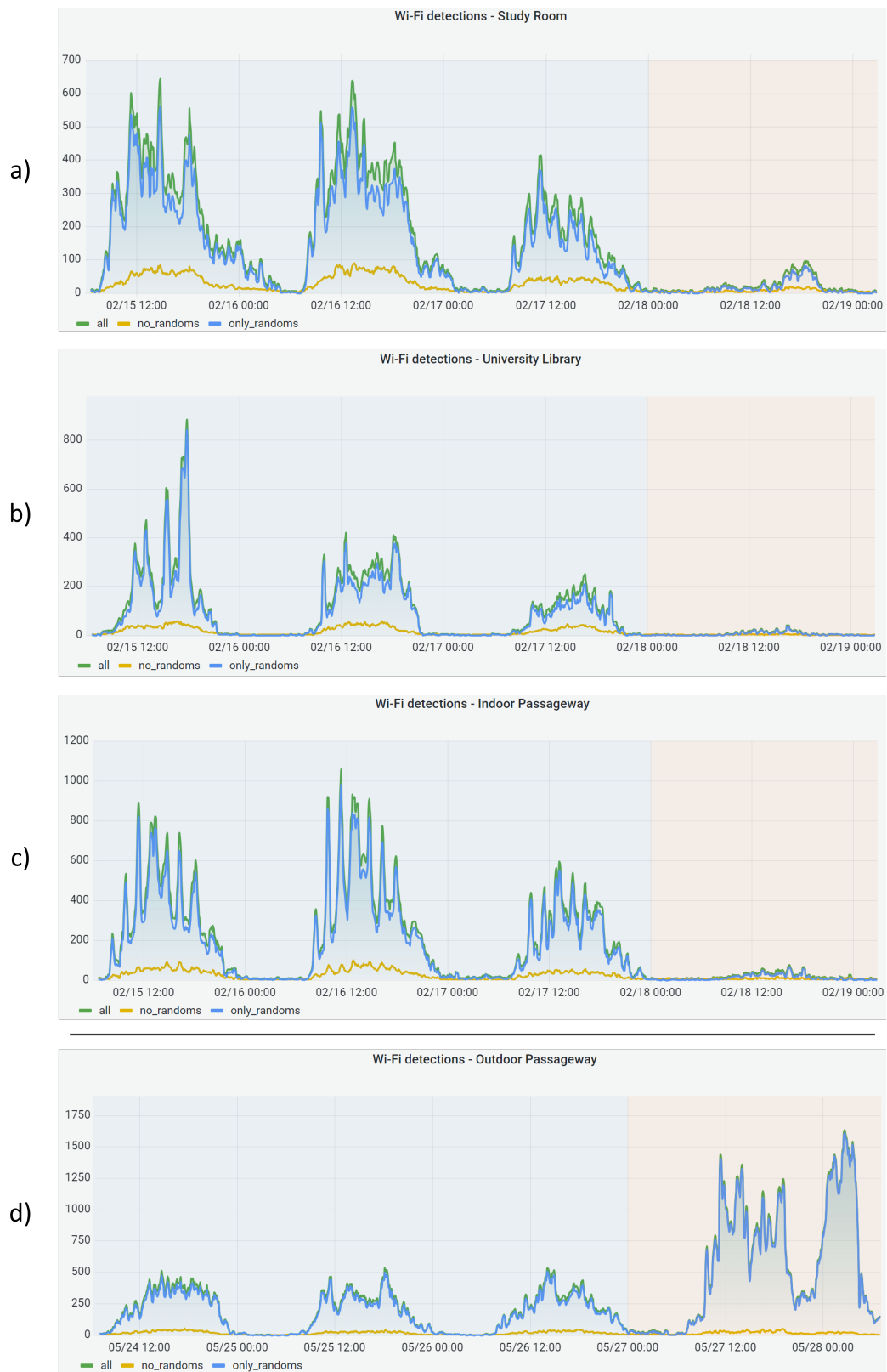


Figure 5.6: Number of devices detected in a four-day period at the first stage of this field experiment at: a) study room; b) university library; c) indoor passageway; d) outdoor passageway. The working days are filled in blue and a Saturday is filled in orange. The same four-day period is presented for the first three scenarios, while a different period is shown for the fourth scenario.

number of detections starts to rise around 8 AM, when the classes start at the university, and decrease to very low detections around 10 PM, the time when the last classes finish at the campus. Also, during the passage of each day, there can be seen several spikes. These spikes are approximately distanced by an hour and a half, and their highest value occurs precisely 5 minutes after the starting time of classes at the university. As so, these spikes are related to the time breaks between classes, where students use their mobile phones before attending classes. Finally, concerning the outdoor passageway, the same crowding pattern can be visualized for the first three days of classes.

Another interesting point that can be noticed is that the number of detections decreases between 12 AM and 1 PM of each working day of classes, except for the sensor deployed at the indoor passageway. This is related to lunchtime, when students generally leave the spaces, namely, the university room, library, and courtyards, to go to the university canteen to have lunch. On the other hand, the indoor passage reveals the contrary to the other locations, where the number of detections increases in this time period. This is a curious fact and can be assertively justified since this narrow passage is used by people to move from one building to another on the university campus during lunchtime.

The results from the outdoor passageway sensor on the fourth day, besides being a Saturday, clearly stands out, with a significantly higher number of detections compared to the other days, and event compared to the other scenarios. This was due to the Iscte's graduation event. This event celebrates the graduation of the university students and diplomas are conferred to them, and broods together university directors, professors, and students' families, friends, and colleagues, so it has a large attendance, as shown in Figure 5.7.



Figure 5.7: Iscte's graduation event.

The collected data also reveals a close correlation with the scheduled time of the event. The event started at 11 AM, followed by lunchtime from 2 PM until 3 PM, and finished by 7 PM. Detections start to rise in the early morning, which may correspond to the event staff and security arriving at the university to make the final preparations for the event. At nearly 11 AM, the detections suddenly rise, which reflects the start of the event. Until lunchtime, two ceremonies took place for different courses, where the transition between them can be noticed by a sudden decrease followed by a rise in the number of detected devices. At lunchtime, the detections decreased to almost the same value as when the event started. After the lunch break, two more ceremonies took place, and then finally, at around 7:10 PM, the detections rapidly

dropped again, thus indicating the end of the event. Despite the end of the event, during the night the sensor detected again a significant increase in the number of detected devices. This high number of detections remained from 11 PM to 4 AM. This was related to a student party that had taken place in the tunnel and in one courtyard in the university to celebrate the graduated students, which had the same time schedule according to the sudden rise and drop detections from the sensor. As so, this sensor was capable of perceiving and detecting two highly-populated events, and also its duration, where the number of detections clearly matched the start and finish of each event.

This particular scenario showed that the sensors are capable of perceiving precisely what it is intended for the STToolkit - detect overcrowding situations as rapidly as they can occur, characterized by sudden increases in the number of detected devices in the sensor proximity, and also perceive people's flow and highly-populated events and its duration. As so, the sensors showed promising results regarding the detection of overcrowding situations, such thing that is very important to provide, especially for SMEs that potentially deal with excessive numbers of people.

Finally, regarding detections from each MAC address type, it can easily be seen that the number of devices detected from virtual MAC addresses (randoms) was always significantly higher than the number of devices detected from real MAC addresses (no_randoms) for the four days of each scenario, with a clear discrepancy in the detections of each MAC address type. This clearly emphasizes the presence and the need for tackling the MAC address randomization process on devices, where the same device can generate multiple MAC addresses over time, thus likely leading to overcounting situations.

In general, the results from this stage showed what was expected for this field experiment, where crowding tendencies and the correlation to the general use of each space could be perceived, as well as high-populated events and their duration, which stand out with a significantly higher number of detections, although the number of detections should be higher than the real number of people present in each location.

5.1.4 Final Stage

In the Final Stage, only data packets and probe requests were detected by the sensors, the frames that showed relevance for device counting.

Just like in the previous stage, a time period that comprised both working days and weekends was intended to evaluate how the sensors were capable of perceiving crowding in both situations, as on working days the university is more occupied than at the weekends.

Figure 5.8 presents a snapshot of the crowding data collected in a four-day period at this stage for the four scenarios where the sensors were deployed. The first three days are working days, and the fourth day is a Saturday, however, the second and fourth days are also public holidays. As previously, working days are filled in blue, and Saturdays in orange. Also, it is important to acknowledge that this four-day period is related to the exams period at the university, and so does not correspond to normal days of classes like in the previous stage. Nevertheless, students also attend the university campus to study and take exams.

The first thing that emerges is a significant reduction in the total number of devices detected

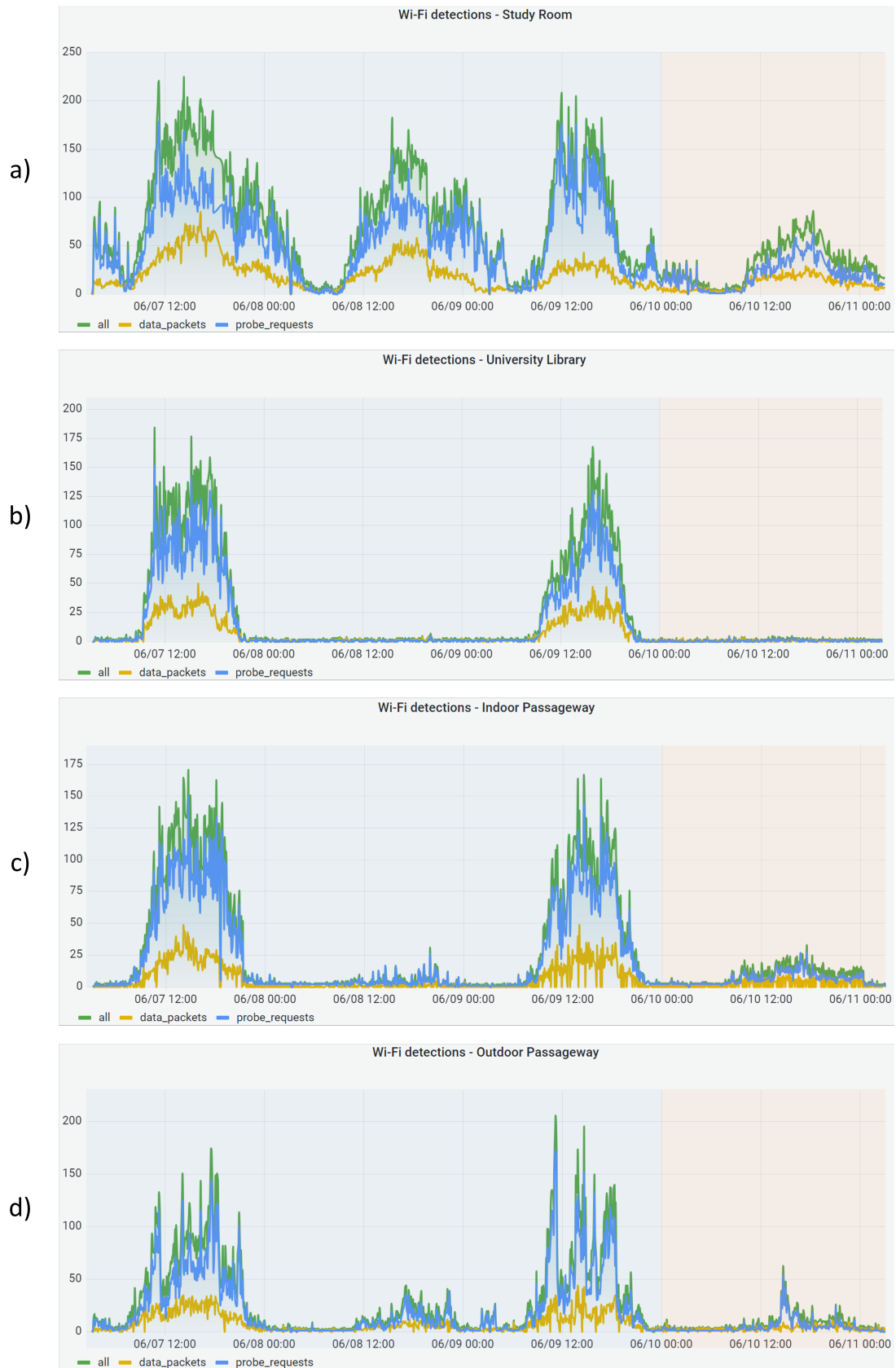


Figure 5.8: Number of devices detected in a four-day period at the final stage of this field experiment at: a) study room; b) university library; c) indoor passageway; d) outdoor passageway. The working days are filled in blue and a Saturday is filled in orange.

from sensors in the four scenarios compared with the First Stage. This is due to the different device counting approach, detecting only data packets and probe requests, and so the total number of detected devices is much lower and likely closer to the real number of people present in the locations.

Nevertheless, the collected data also showed a close correlation with the general use of each space just like in the First Stage. The study room revealed detections on all four days. Although the second and fourth days were public holidays, there were also detections on these days. This can be justified by the fact that the study room remains open on holidays, which students attend, especially during the exam period when the data was collected. The first three days show more detections than the fourth day, which may be justified by the fact that the fourth day is also a Saturday when there are usually fewer people at the university at the weekend. Furthermore, there were also more detections during the day and the evening, but also during the night, which indicates the late night studies from students in this room for the exams.

For the remaining scenarios, it can be easily seen that there was only detections on the first and third day and much lower (and almost an absence) detections on the second and fourth days. This absence of detections was expected in the remaining scenarios, as these days were holidays when the library was closed and exams were not allowed in the university. This justifies the absence of detections in the university library on holidays. However, both indoor and outdoor passageways still revealed a few detections on these days. As both passages are public spaces, people could also use them to move around the university.

Regarding the first and third days, the collected data also shows a close correlation with the general use of each space. At the library, detections start to rise at 9 AM, when the library opens, remain slightly constant during the day, and around 7 PM start to gradually decrease until reaching almost zero around 9 PM, when the library closes. For the indoor passageway, detections start to rise in the morning, when the first exams occur in the university, tend to stay slightly constant during the afternoon, and then around 9 PM stabilize again at almost no detections, the time when the last exams finish on campus. Also on both days, we can observe some spikes during the afternoon, spaced around three hours each, more emphasized on the third day than on the first. In our university, exams usually have a 3-hour duration, and so those spikes can be related to imminent times before and after exams, where entire classes of students move around the campus to attend them. The same crowding pattern applies to the outdoor passage.

Finally, regarding detections from each packet type, it can easily be seen that the number of devices detected from probe requests is always higher than the number of devices detected with data packets in all four days. Device detections from probe requests revealed fluctuations throughout each day, which can be related to the MAC Address randomization process that was not taken into account at this stage. Following the trace, device detections from data packets were of a lower magnitude, and slightly more constant throughout the days, more in the study room and in the library than on both passageways. This behaviour was expected since the study room and the library are high-permanency time spaces, which students tend to use for long periods of time to study, usually making use of the Internet for research and web browsing, which is only possible if the devices are connected to a local network, usually to the university local network via Wi-Fi. On the other hand, detections from data packets revealed much more

fluctuations on both indoor and outdoor passageways. This can be justified by the fact that these areas have a high flow of people, with a short permanency time, and so there are not many devices connected to a Wi-Fi network continuously present in the area.

In general, the results from this stage showed that the same correlation with the general use of spaces can be perceived by the sensors, also likely with a total number of detections closer to the real number of people present in the area by only detecting the frames from mobile devices that shown relevance for device counting.

As so, the results from this field experiment showed that the sensors are able to perceive people's concentration and flow according to the general usage of spaces. Although we were not able to validate the number of devices detected in each scenario, this field experiment allowed us to understand that the STToolkit is capable of perceiving people's tendencies, and sudden arrivals and departures of people in quasi-real-time throughout the day, as well as highly-populated events and its duration, such things that are extremely important to perceive, as the overcrowding situations have the same nature.

Table 5.2 summarises the main results obtained from this field experiment in the different scenarios where the sensors were deployed on the Iscte's Campus.

Table 5.2: Main results obtained in the different deployment scenarios at Iscte's Campus.

Scenario	Results
Study Room	Correlation between student's study time throughout the day and night, and lunch break.
University Library	Correlation between library's opening and closing hours, lunch break, and quick arrival and departure detections.
Indoor Passageway	Correlation between time breaks between classes, detections when classes start and finish lunch times, and starting/ending exam times.
Outdoor Passageway	Correlation between time breaks between classes, lunch breaks, starting/ending exam times, and highly-populated events detection and duration.

5.2 Fingerprinting validation

This section describes the validation of the fingerprinting technique for tackling the MAC Address randomization process of mobile devices, based on the Information Elements (IEs) of probe requests. This validation proceeded after determining what information should be considered for each IE of the probe requests, as already described in section 4.3.3.

The validation of the fingerprinting technique was performed in two stages:

1. **Validation from a public dataset** - Based on the public dataset of [28], a dataset of labelled device probe requests from a total of twenty-two devices, seventeen of which with MAC address randomization. In fact, this was the dataset used to determine what information should be considered for each IE of the probe requests. The results from this stage are presented in subsection 5.2.1.

2. **Validation from a dataset collection at Iscte** - Based on a dataset collected on the Iscte’s garage. A dataset of probe requests was also collected at the Iscte’s garage to validate the fingerprinting technique. The results from this stage are presented in subsection 5.2.2

5.2.1 Validation from a public dataset

After determining what information should be ignored for the device fingerprint for each IE, the fingerprinting technique was applied to the public dataset of [28]. As already described in section 4.3.3, this dataset contains probe requests from twenty-two devices, seventeen of which with MAC Address randomization, being the ground-truth number of devices for this validation. The information of each device of this public dataset is presented in Table 5.3.

Table 5.3: Devices used on the public dataset (based on [28]).

Device ID	Device OS	Device OS version	Device vendor	Device model	Random MAC
A	Android	11	Samsung	Galaxy M31	Yes
B	Android	6.00.01	Xiaomi	Redmi 4	Yes
C	Android	4.02.02	Samsung	Galaxy S4	No
D	Android	6.0	Huawei	ALE-L21	Yes
E	Android	10	Xiaomi	Mi A2 Lite	Yes
G	Android	10	Huawei	CLT-L09 (P20)	No
H	Android	7.0	Samsung	Galaxy S6 edge+	No
I	Android	8.00.00	Samsung	Galaxy S7	Yes
J	Android	8.01.00	Xiaomi	Redmi 5 Plus	Yes
K	Android	10	Samsung	Galaxy J6	Yes
L	Android	11	Google	Pixel 3A	Yes
M	iOS	14.05.01	Apple	XS max	Yes
N	iOS	12.05.02	Apple	Iphone 6	Yes
O	Android	Oxygen 11	One Plus	Nord	Yes
Q	Android	9	Huawei	VTR-L09 (P10)	No
R	Android	9	Huawei	STF-L09 (Honor 9)	Yes
S	Android	10	Xiaomi	Redmi Note 7	Yes
T	Android	11	Xiaomi	Redmi Note 9S	Yes
U	iOS	14.6	Apple	Iphone XR	Yes
V	Android	11	Google	Pixel 3A	Yes
W	iOS	14.05.01	Apple	Iphone 12	Yes
X	iOS	14.6	Apple	Iphone 7	Yes

In addition, this dataset contains a total of 3201 random MAC addresses, which emphasizes the need for tackling this issue due to the large quantity of different MAC addresses that were

generated from only twenty-two devices.

After applying the fingerprinting technique to the public dataset, the results are presented in Table 5.4. The fingerprints highlighted in orange are fingerprints that have been used to identify more than one device. The devices that do not have random MAC addresses are not included, since they do not need a fingerprint for their identification.

Considering the number of fingerprints generated for each device, a total count of 41 fingerprints was generated. However, 4 of these fingerprints are used to identify more than one device, leading to a total of 37 different fingerprints generated from 17 devices with MAC address randomization. In fact, the authors of the public dataset claim that device D has MAC address randomization, however, the MAC addresses from its probe requests are not virtual MAC addresses, i.e., they are not randomized, and so a fingerprint was not generated to this device.

The fingerprints generated for more than one device reveal some interesting correlations between devices:

- The fingerprint '12B6A1F245BD148C' was generated for the devices B, E and J. These devices belong to the same vendor.
- The fingerprint '1E9F3FF37375CF9' was generated for the devices E and J, also belonging to the same vendor.
- The fingerprints 'F25223F8379E2761' and '550A0B16FB57D876' were generated for the devices L and V. These devices are identical, having the same OS, OS version and model.

Considering this, there is a strong indication that the fingerprinting technique may be capable of distinguishing device models, but not distinguishing devices with the same model and OS version.

Furthermore, it can also be seen that more than one fingerprint was generated for the majority of the devices, namely, for 12 of the 17 devices, which is not desired as a device should be identified by a unique fingerprint, which occurred for the remaining 5 of the 17 devices. Nevertheless, some of these fingerprints were generated sporadically from a minimal number of probe requests. In fact, 12 out of the 37 different fingerprints were only generated considering less than two probe requests from the same device, which could be easily dropped after some post-processing applied to these results, resulting in a total of 25 different fingerprints.

As so, the precision of this fingerprinting technique regarding this dataset was 46%, although it can be potentially and easily leveraged to 68% if the sporadic fingerprints are removed after some simple processing.

The results from this validation show that the fingerprinting technique can fairly minimize the MAC address randomization process from mobile devices, as from a total of 3201 different MAC addresses, the results that previous device counting approaches without tackling the MAC randomization process would obtain, could be reduced to a total of 37 unique fingerprints, which could also be easily dropped to 25 fingerprints. In addition, these results also show that the fingerprinting technique may be capable of distinguishing devices from different device models, but not distinguishing devices from the same model and OS version.

Table 5.4: Device fingerprints for the devices of the public dataset. The fingerprints highlighted in orange are fingerprints that have been used to identify more than one device.

Device ID	#Fingerprints	Fingerprints
A	1	72DF2B3EB9CC86E5 485 in 485 (100%)
B	6	12B6A1F245BD148C 619 in 22100 (3%)
		35648FB92F5C6BD9 21477 in 22100 (97%)
		A6C10FEBE7431E49 1 in 22100 (0%)
		80533174893342C2 1 in 22100 (0%)
		E481DF8DD397543D 1 in 22100 (0%)
		D649D593560F56F6 1 in 22100 (0%)
E	4	1E9F3FF37375CF9 1723 in 2309 (75%)
		B60395B224395FAF 1 in 2309 (0%)
		FDFD7021FE90779 1 in 2309 (0%)
		12B6A1F245BD148C 584 in 2309 (25%)
I	2	ACC3DFAC77E4BE9E 696 in 697 (100%)
		133DC71887F405A0 1 in 697 (0%)
J	6	1E9F3FF37375CF9 7521 in 7759 (97%)
		AEDA9E47F05835D5 1 in 7759 (0%)
		E35C46FA5208AB5C 1 in 7759 (0%)
		4CD2969BF74113EE 1 in 7759 (0%)
		4D2A643B2C21C0B8 1 in 7759 (0%)
		12B6A1F245BD148C 234 in 7759 (3%)
K	1	B40D327F7B8BA054 699 in 699 (100%)
L	3	F25223F8379E2761 380 in 533 (71%)
		550A0B16FB57D876 129 in 533 (24%)
		90307FC366B5C7F7 24 in 533 (5%)
M	2	A4FF5114B0DF0887 196 in 799 (26%)
		69F97A08B8420A65 603 in 799 (74%)
N	1	C329DBB3B7966211 787 in 787 (100%)
O	3	41DD93B6DEFA74A3 422 in 729 (58%)
		DAE490AF24C93ADE 307 in 729 (42%)
		4F007772C0962F2E 1 in 729 (0%)
R	1	4B7FD878B1500464 2254 in 2254 (100%)
S	2	16ECC1B78E41BBD8 380 in 5571 (7%)
		705FBF6D3B6F8466 5191 in 5571 (93%)
T	2	550227F7D83006C6 1392 in 1423 (98%)
		BEAFF3634780AF22 31 in 1423 (2%)
U	2	D16AD66A222A83FD 85 in 146 (58%)
		CC874E0D9D08EA72 61 in 146 (42%)
V	3	550A0B16FB57D876 70 in 267 (26%)
		F25223F8379E2761 127 in 267 (48%)
		E6AD41355DAA8CC7 70 in 267 (26%)
W	1	F67CE218C0293586 2237 in 2237 (100%)
X	2	245ACB6C8CEA0C6 75 in 1097 (7%)
		7F4FD1B2D22D9414 1022 in 1097 (93%)
Total	41	37

5.2.2 Validation from a dataset collection at Iscte

A second validation of the fingerprinting technique, with the already known information to be considered and ignored for each IE of probe requests, was also performed using a dataset created at Iscte.

This dataset was obtained from collecting probe request frames from known devices at the Iscte's garage, with a similar procedure to the dataset collection from [28]. This location was selected since it was the most isolated location of the entire university, not having either mobile or network coverage, as well as only a few people use this location during the day, mainly the university's professors and staff to park their cars. Having the most isolated location was intended to minimize the external interference of other people's devices that could noise the creation of this dataset. In particular, this dataset collection was performed on the lowest floor of Iscte's garage, to minimize even more the external interference of devices. Figure 5.9 presents the above-described dataset collection.

Also, it is important to mention that this was not the ideal test environment for this dataset generation. As it is not a completely isolated environment, the creation of this dataset was subject to external interference from other devices in the garage during its execution. Nevertheless, these external occurrences were perfectly identified during the dataset collection.



Figure 5.9: Probe requests dataset collection at the Iscte's garage.

In this dataset collection, probe requests were collected and labelled from a total of nine devices. Care was taken to select mobile devices from different manufacturers with different OSs and OS versions in order to have a diversity of devices. In addition, at least two devices with the same vendor and OS version were intended to evaluate if the fingerprinting technique would be able to distinguish them, something that did not occur in the previous dataset. Table 5.5 presents the nine mobile devices used for this dataset collection.

As it can be seen, from the total nine devices, six of them use MAC Address randomization, while the remaining three use their real MAC address on probe request frames. Despite the latter not being taken into account for the fingerprinting validation, having devices with their

Table 5.5: Devices used for the dataset collection at the Iscte’s garage.

Device ID	Device OS	Device OS version	Device vendor	Device model	Random MAC
A	Android	8.0	Huawei	P10 Lite	Yes
B	Android	8.0	Huawei	P10 Lite	Yes
C	iOS	16.5	Apple	Iphone 13 Pro	Yes
D	iOS	15.6	Apple	Iphone 7 Plus	Yes
E	iOS	13.7 (17H35)	Apple	Iphone 6S Plus	Yes
F	iOS	15.5	Apple	Iphone SE	Yes
G	Android	7.0	Huawei	GT3	No
H	Android	7.0	Vodafone	Smart N8	No
I	Android	6.0	Sony	Xperia	No

real MAC addresses would help in perceiving external interferences on the dataset collection, as these devices can be easily identified by their real MAC addresses. In addition, devices A and B have the same vendor and OS version, which were also intended to evaluate the performance of the fingerprinting technique to distinguish between both.

For detecting the probe requests emitted from the devices, three sensors were specifically configured for this field experiment. The only difference between the three sensors was on the Wi-Fi board used for detecting the device’s frames. This was performed also to evaluate and compare the performance of the different Wi-Fi boards that were available for the STToolkit. As so, each sensor used one of the following Wi-Fi boards: (1) Alfa Network AWUS036AC, (2) Alfa Network AWUS036ACH, and (3) Alfa Network AWUS036ACM.

The procedure for detecting devices was identical to the procedure from [28], except for the time duration each device was individually detected. The three sensors detected probe request frames on the 2.4 GHz Wi-Fi band and on channels 1,6, and 11. For the dataset collection, the sensors were programmed to start sensing data in their proximity once they boot up. The gathered information was stored in a .pcap file for each mobile device, containing all Wi-Fi frames detected by each sensor. Each mobile device was detected individually by the three sensors for 10 minutes, instead of 20 minutes as the procedure from [28]. Nevertheless, this time period was also sufficient for capturing the probe requests from the devices, as most of them send probes within a time range of only a few minutes, not exceeding 10 minutes.

For this, at the beginning of every 10 minutes, the sensors were booted and the mobile device was placed near the sensors with its Wi-Fi interface switched on, and remained stationary during this time period. At the end of the 10 minutes, the sensors were switched off, and another device was used. This procedure was followed for the nine devices. As so, a total of nine .pcap files were collected for each sensor, one for each mobile device, whose results are presented in Table 5.6.

As it can be seen, the Sensor 3, equipped with the Alfa Network AWUS036ACM board did not collect information from all mobile devices, as the driver installed on this sensor revealed

Table 5.6: Dataset collection results at the Iscte’s garage.

Device ID	Sensor 1		Sensor 2		Sensor 3	
	File size [kB]	#Probe Requests	File size [kB]	#Probe Requests	File size [kB]	#Probe Requests
A	3	30	5	52	-	-
B	3	23	3	23	1	10
C	4	26	3	23	2	14
D	9	80	3	20	3	24
E	14	110	13	106	6	50
F	30	216	24	169	8	52
G	12	76	16	97	4	21
H	18	84	49	231	9	40
I	54	268	83	411	20	95
Total [A - F]	63	485	51	393	20	150
Total	147	913	199	1132	53	306

performance issues. Therefore, this sensor could never be considered for validation. From the results gathered from the remaining Sensor 1, equipped with the Alfa Network AWUS036AC, and Sensor 2, equipped with the Alfa Network AWUS036ACH, the information from the first was considered, as this was the sensor that collected more information for the devices that have MAC address randomization, the ones that are relevant for this validation.

After the creation of this dataset, the fingerprinting technique was applied, and its results are shown in Table 5.7. The fingerprints highlighted in orange are fingerprints that have been used to identify more than one device. The devices that do not have random MAC addresses, namely, the devices G, H, and I, are not included, since they do not need a fingerprint for their identification.

Table 5.7: Device fingerprints for the dataset collection at the Iscte’s garage. The fingerprints highlighted in orange are fingerprints that have been used to identify more than one device.

Device ID	#Fingerprints	Fingerprints
A	1	D1B9420FB3166EDA 30 in 30 (100%)
B	1	D1B9420FB3166EDA 23 in 23 (100%)
C	1	FE0ADDA77B86B474 52 in 52 (100%)
D	1	B16CD6FEFE1B2C52 80 in 80 (100%)
E	1	A4A2846DFC6683A1 110 in 110 (100%)
F	1	C329DBB3B7966211 216 in 216 (100%)
Total	6	5

As can be seen, a total of 5 different fingerprints were generated for the six devices with MAC Address randomization. The same fingerprint was generated for devices A and B, the ones with the same device model and OS version. This concludes that the fingerprinting technique was not capable of distinguishing devices from the same model and OS version, just like occurred in the previous validation from the public dataset. The remaining four devices C, D, E and F were only identified by one fingerprint, which was the desired outcome. Therefore, although this was a small dataset collection with only a few devices with MAC address randomization, the precision from this validation was 83%.

The results obtained enforce those observed from the public dataset, where the fingerprinting technique has a high entropy for detecting different device models, but more steps are required to distinguish devices with the same model and OS version. In fact, the fingerprints retrieved from this technique can be a correlation between the number of device models and the real number of devices present in the area, if the talkativeness of each device model is also known.

Although this is a preliminary approach, the results show the fingerprinting technique can fairly minimize the impact of the MAC Address randomization for more precise device counting. This is clearly shown with the validation performed with the public dataset of [28], where a total of 3201 different MAC addresses from twenty-two devices could be reduced for a total of 37 fingerprints, which, in fact, could even be easily dropped to 25 fingerprints after some simple post-processing.

Furthermore, higher accuracies should be leveraged if this fingerprinting technique is combined with other properties of the device's probe requests. For instance, the sequence numbers (SEQ), the time between probe requests within a burst, also known as the Inter-Frame Time Arrival (IFTA), and other temporal analyses of probe requests may be combined in a clustering algorithm for more accurate crowd counting, just like other approaches have proceeded. This fingerprinting technique was implemented on the Wi-Fi detection algorithm, presented in subsection 3.2.1, for the validation of the final crowd-counting approach. This validation is presented in the next section.

5.3 Crowd counting approach validation

The National Palace of Pena, better known simply as the Pena Palace, is a Romanticist castle in São Pedro de Penaferrim, in the municipality of Sintra. It has been recognized as a national monument since 1910 and constitutes one of the major expressions of 19th-century Romanticism in the world. The palace is a [UNESCO World Heritage Site](#) and one of the [Seven Wonders of Portugal](#), being one of the most popular and iconic tourism sites in the country, flattened by overtourism all year round. Figure 5.10 illustrates the Pena Palace.

A field experiment was also carried out at Pena Palace to demonstrate how effectively the sensors were able to detect events, with sudden increases in the number of people arriving at a location, and also the precision regarding the number of devices detected and the real number of people present in a given area.

This field experiment is divided into four stages. The first stage, presented in subsection 5.3.1, regards the deployment description at the Pena Palace, including the decision on the



Figure 5.10: Pena Palace.

location of the sensors. Then, the subsection 5.3.2 includes the customized dashboards for visualizing the crowding data collected from the sensors. The third stage, presented in subsection 5.3.3, regards a calibration action for restricting sensors' detection range to a target area. Finally, the fourth and last stage, presented in subsection 5.3.4, regards the validation of the final version of the Wi-Fi detection algorithm, including the fingerprinting technique to tackle the MAC address randomization, during a public event.

5.3.1 Deployment description

Pena Palace is surrounded by Pena Park, an 85-hectare walkable park with several gardens and landscaped areas with indigenous and exotic species from all over the world, as well as waterfalls, tanks, lakes, and fountains, along with small decorative buildings that visitors can explore. Figure 5.11 shows the wide area of the Pena Park that surrounds the Pena Palace, highlighted in red, and also the Picadeiro, a public square of Pena Park, highlighted in orange.

One of the main concerns that tourism managers face derives from the fact that most people come to Pena Park just to visit the Pena Palace, as it is the main attraction for tourists. Consequently, due to its noticeable attractiveness, the latter is faced with permanent overcrowding, leading to long queues and unpleasant waiting times which deteriorates visitor's quality of experience and satisfaction. On the other hand, Pena Park is much less visited and ends up being a relatively unused space, with such a wide and large area to explore which most visitors usually do not take advantage of.

Faced with this problem, tourism managers have been trying actions to increase Pena Park's attractiveness and encourage tourists to visit other areas than just the Pena Palace to mitigate the overtourism problem, and also to enhance visitor's quality of experience, as they are introduced to the numerous attractions that Pena Park has to offer.

One of those incentive actions conducted involved holding small public events in Pena Park, more precisely in one of its most interesting locations called Picadeiro, presented in Figure 5.12. Picadeiro is a rectangular square originally conceived for horse riding demos and is located right next to the main path that tourists use to go to the Pena Palace. It has a small Kiosk where people can buy snacks and where they can sit and rest during the visit. In addition,

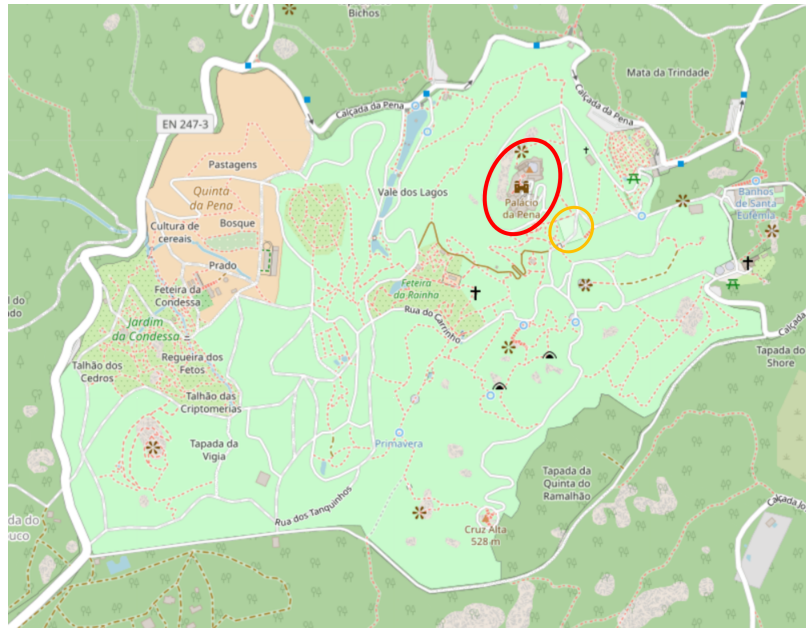


Figure 5.11: Pena Park that surrounds Pena Palace (red), and Picadeiro (orange), a public square of Pena Park.

this area also serves as a connection point between several visiting paths of the park, having several entrance and exit points, as shown in Figure 5.12. One type of event that takes place at Picadeiro is a demonstration of the [Portuguese School of the Equestrian Art](#), with precise choreography of thoroughbred [Lusitano horses](#) in front of a public audience. These actions are an incentive from tourism managers to encourage tourists to visit the Picadeiro, instead of going directly to the Pena Palace. Some of these events occurred in the last week of July and August 2023, from Tuesday to Friday, starting at 3:30 PM, and had a 20-minute duration.

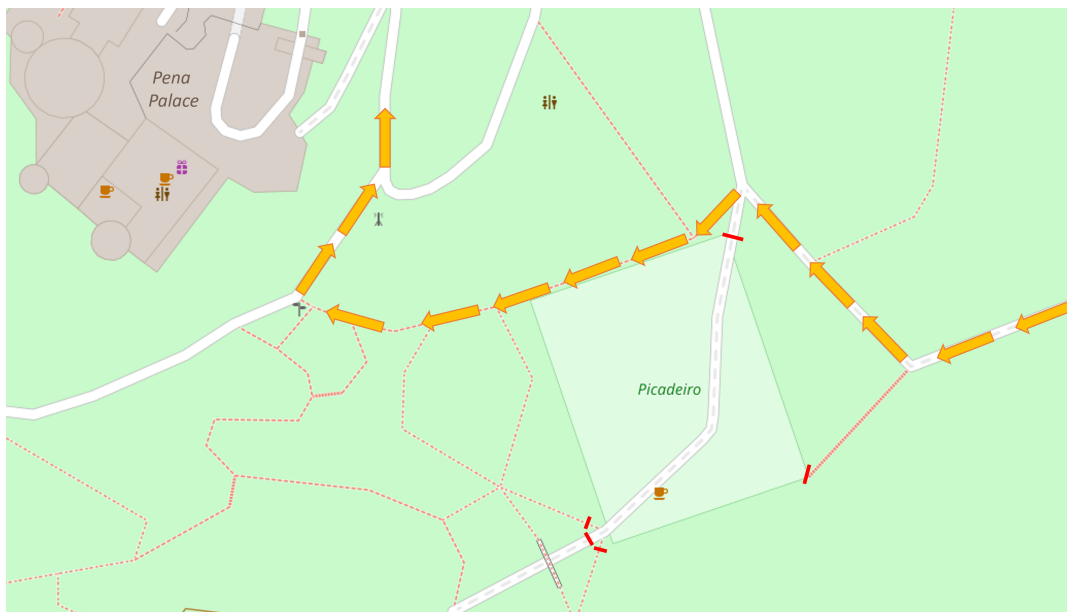


Figure 5.12: Picadeiro area and tourists' main path to the Pena Palace. Each red line represents an entrance and exit point of Picadeiro.

As so, to help measure the impact of these events on Picadeiro, a deployment of the sensors

was intended to monitor this area. The challenge imposed regards what is the best configuration of sensors to detect people in the Picadeiro area. As it is a rectangle public square, the ideal deployment location to monitor the Picadeiro would be placing a sensor on the centre of the rectangle with an omnidirectional antenna, however, due to infrastructure limitations, the deployment at this location was not possible. As so, two sensors were deployed in two different locations to monitor the Picadeiro area: one inside the Kiosk, and another inside a Help-point next to its main entrance. Figure 5.13 presents and highlights the deployment of the sensors in both locations. Network coverage for the uplink and electrical power were provided in both locations so the deployment of sensors was perfectly feasible for both. The Kiosk provided stable Wi-Fi coverage and the sensor deployed at this location uploaded the crowding information via Wi-Fi, while the Help-point provided a cabled network, and so its respective sensor uploaded data via Ethernet.

The sensors performed 24/7 real-time detection and periodically uploaded the number of devices detected in its proximity to the cloud server. In addition to the previous deployment at the Iscte's Campus, the final version of the Wi-Fi detection algorithm, presented in subsection 3.2.1, was tested and validated in this field experiment, already with the fingerprinting technique to tackle the MAC address randomization process of mobile devices.

The sensors sent periodically crowding measurements every 5 minutes, with the number of devices detected within a sliding window of also 5 minutes. In addition, a new dedicated information system was created for this deployment, with a new InfluxDB bucket and user profile. A [Virtual Private Network \(VPN\)](#) connection was also provided that allowed remote access to both sensors. This enabled the ability to make over-the-air updates for quick updates on sensors, such as adding packet power filtration thresholds, as further described in this section.



Figure 5.13: Deployment of sensors in Picadeiro's Kiosk (left) and inside a Help-point (right).

The sensor deployed in the Kiosk had two omnidirectional antennas, while the sensor

deployed in the Help-point had a directional antenna pointed to the Picadeiro area. The latter was necessary to only detect devices that were inside Picadeiro, and not the devices from the remaining visitors going into the Pena Palace in the opposite direction. Furthermore, for testing different hardware, the sensors used different network cards. Table 5.8 summarizes the hardware used and the upload communication option of both sensors deployed at Picadeiro.

Table 5.8: Selected hardware and upload option for the sensors deployed at Pena Park.

Sensor	Processing Unit	Wi-Fi card	Antenna Type	Upload
Kiosk	Raspberry Pi 4	Alfa Network AWUS036AC	2 x 5 dBi Omni-directional	Wi-Fi
Help-point	Raspberry Pi 3	Alfa Network AWUS036ACH	1 x 8 dBi Directional	Ethernet

Also, it is important to mention that both sensors performed detection of devices in only the 1, 6, and 11 channels in the 2.4 GHz Wi-Fi band, as most devices communicate in those channels, as well as the 2.4 GHz is the most common Wi-Fi band used by devices.

5.3.2 Data visualization

For the data visualization of the crowding information collected by the two sensors deployed in Pena Park, a customized Grafana dashboard was created purposefully for this deployment and is presented in Figure 5.14. This dashboard is composed of graphs for the temporal rendering of information, allowing users to observe crowding during specific time periods, and also maps to show the location of each sensor and its crowding levels in real time. This dashboard also had user profile restrictions so that only the system administrator and the *penapalace* user, created for the Pena Palace’s tourism managers to visualize the crowding data, were the only allowed users with visualization permissions. Also, for coherency, the *penapalace* user could only visualize this dashboard.

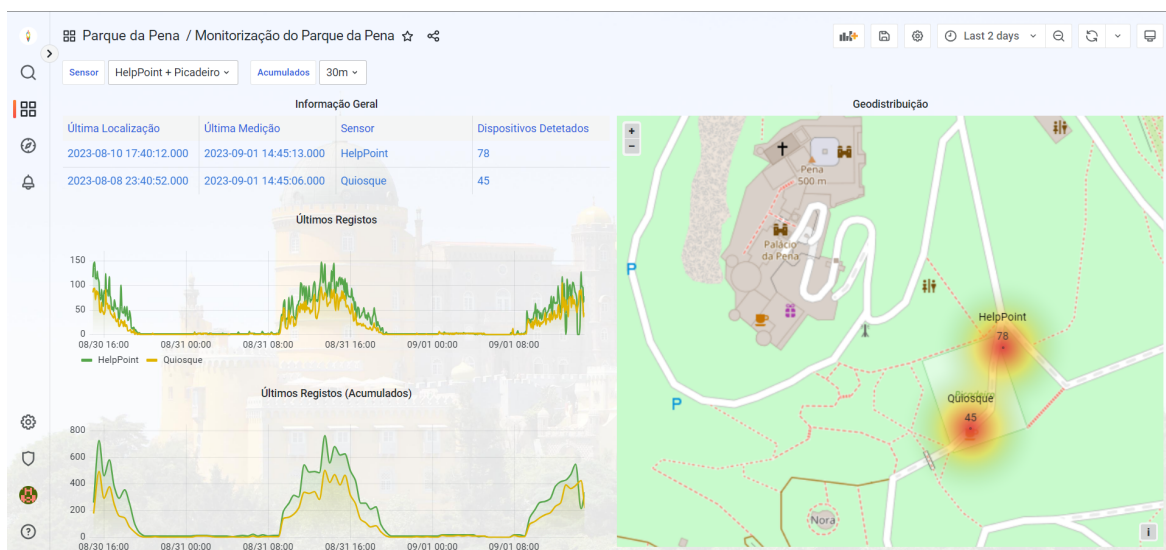


Figure 5.14: Customized dashboard created for crowding data visualization of the Pena Park.

The dashboard is composed of a table, in the top left corner, with general information about the system, for instance, the last location of each sensor, its last crowding measurement sent, the sensor's location, and the number of devices detected in its last measurement sent. This table could be used for a quick analysis of the operability of the system.

Furthermore, there are two graphs in the bottom left corner. These graphs show the crowding measurements sent by the sensors within a selected time range, which users can select as they wish. Users can also choose from which sensors they want to see the measurements through the '*Sensor*' field on the top left corner above the general information table. The data points in both graphs are the total number of devices detected from probe requests and data packets. The top graph shows each data point sent in the selected time range, while the bottom graph shows a moving/rolling average number of detected devices. The latter was specifically requested by the tourism managers, where users can select the time period using the '*Acumulados*' field next to the '*Sensor*' field, and detections from each sensor are summed according to the time period. Several time periods can be chosen, where the ones that make sense are 10 minutes, 30 minutes, 1 hour, 6 hours, 12 hours, 1 day, 2 days, 7 days, 14 days, or 30 days.

Finally, there is also a map on the right side of the dashboard. This map is used for spatial visualization of the crowding information, showing each location where both sensors were deployed at Pena Park in the form of heat maps. It also shows the number of detected devices at each location in real time.

As a result, this customized dashboard allowed a quick and easy interpretation of crowding at Pena Park, more precisely at Picadeiro, the area that concerned the Pena's tourism managers to be able to perceive the number of people present throughout the day.

5.3.3 Coverage tests

As already mentioned, two sensors were strategically deployed at Picadeiro, one of the most interesting locations of Pena Park, to monitor crowding only in the Picadeiro area. Nevertheless, the detection range of the sensors was not known, so they could in reality detect devices outside of Picadeiro, thus leading to overcounting estimations.

Therefore, sensor coverage tests were carried out to delimit the detection range of sensors to the Picadeiro area. As a result, the sensors will only detect devices in this area.

For this, a dedicated sensor was configured. The sensor was programmed to only detect probe request frames sent from one specific and known mobile device, with its unique MAC address. The talkativeness of this device was also known. This was important to acknowledge for knowing what was the sufficient time to detect the device during these tests. The device sends probe request bursts every 15 seconds, which is the minimum time period to detect the device at a given location.

The dedicated sensor performed continuous detection of probe request frames from this device and, every 4 minutes, uploaded information to the cloud server with several metrics, such as: (i) the number of total packets, (ii) the maximum power, (iii) minimum power, and (iv) the mode of power (the most common packet power) received within the last 2 minutes. To ensure rigour in the execution of these tests, the same hardware used for the sensors deployed at both locations was used by the dedicated sensor.

Also, a dedicated dashboard was created for the sole purpose of this coverage test, presented in Figure 5.15. This dashboard allowed a quick analysis of the measurements sent periodically by the dedicated sensor during the execution of the coverage tests.

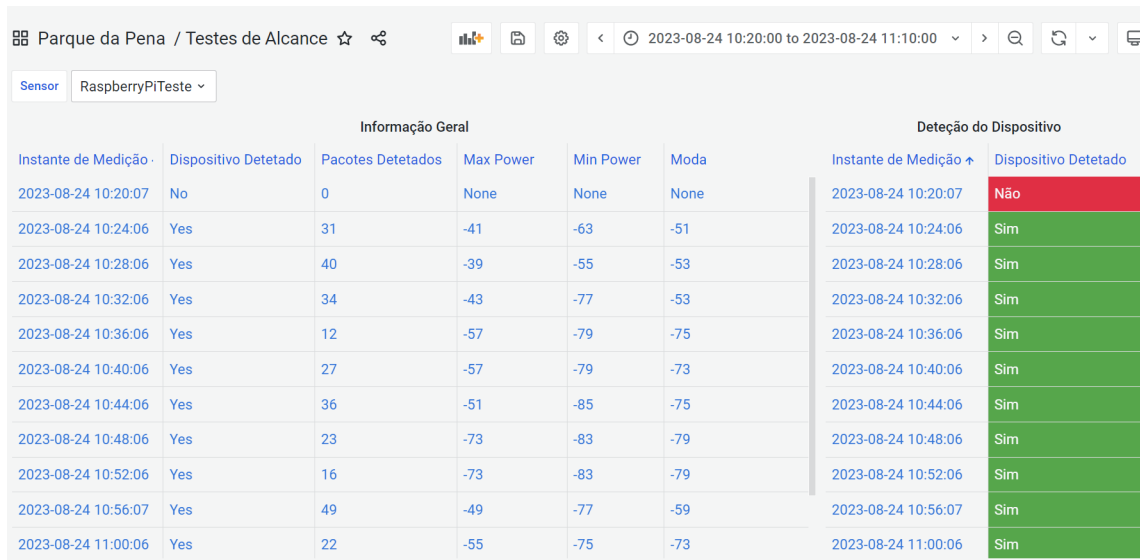


Figure 5.15: Dashboard created for the coverage tests at the Pena Park.

As so, the detection range of each sensor could be perceived by walking through several locations around Pena Park and observing the coverage test measurements sent for each location. At the beginning of every 4 minutes, the device's Wi-Fi interface was switched on and the device remained stationary at the test location for 1 minute. After the first minute, the interface was switched off and the next location was proceeded. This procedure was repeated for several locations for both the Kiosk and Help-point sensors.

For both sensors, the detection coverage was tested at the boundaries of the Picadeiro area and on the visitor paths of Pena Park around Picadeiro. It was assumed that if the detection coverage was provided at the boundaries of Picadeiro, it was also provided throughout the entire Picadeiro area. In addition, some of the test locations were also stipulated considering the antenna type of sensors, particularly the Help-point sensor, which uses a directional antenna pointed towards the area of the Picadeiro. The most important factor to obtain from these coverage tests was a difference in the packet received power between the boundaries of Picadeiro and the remaining test locations. As a result, power filtration could be applied to limit the detection range to the Picadeiro area.

After carrying out the coverage tests for both sensors, the results obtained are presented in Figure 5.16 in the form of heatmaps. Each location with a heatmap circle represents a location with sensor coverage, i.e., where at least one package of the device was detected at that location. The locations with a black dot and a zero are locations with no sensor coverage, i.e., no device packets were detected at those locations. Each heatmap circle is accompanied by the mode of power received at that location. The mode (most recurrent value) was considered the most reliable and relevant metric, as a surrogate of the received power from the several bursts of probe requests sent at each location.

Regarding the Kiosk sensor, all tested locations had detection coverage, i.e., the device was

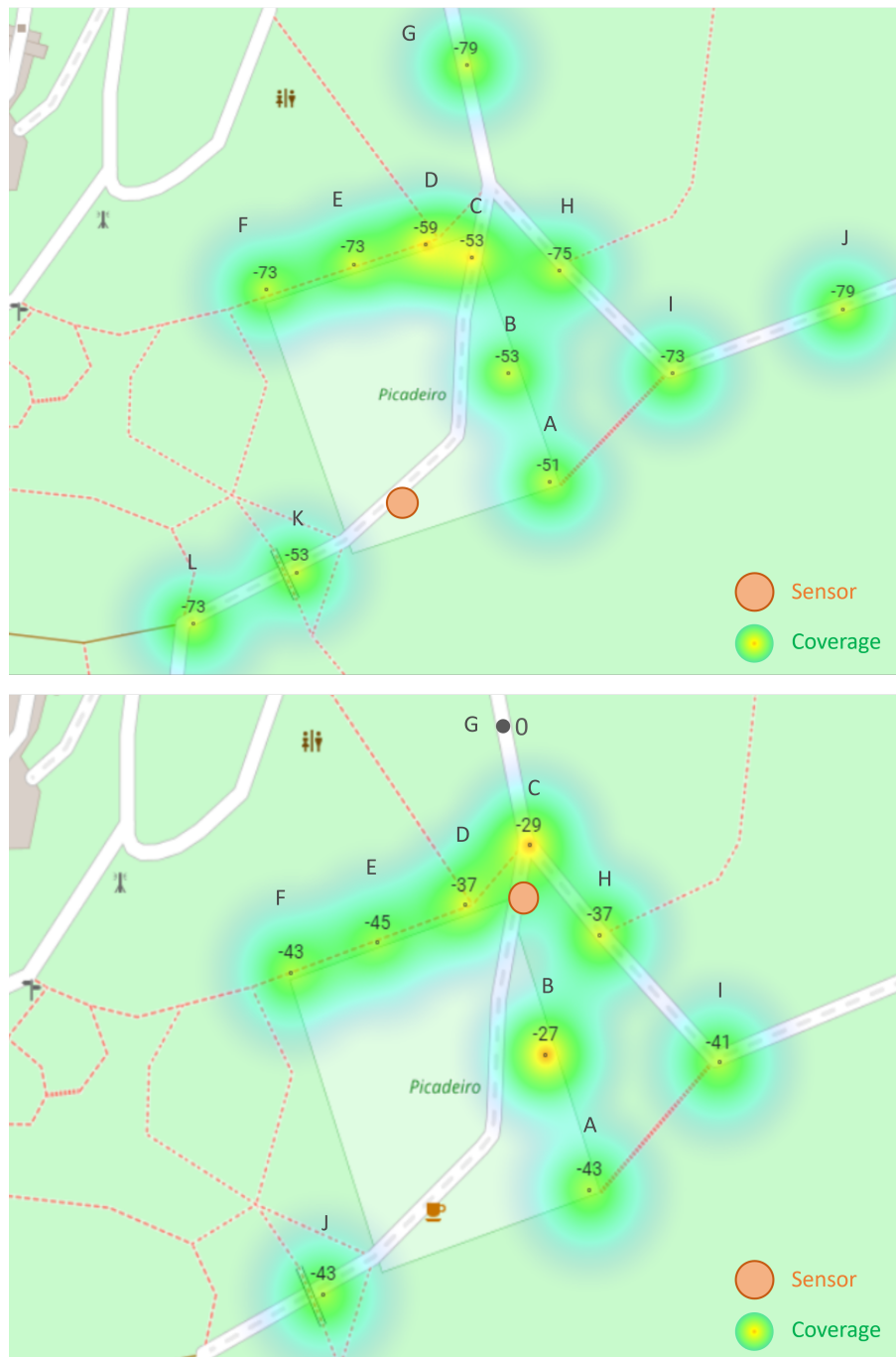


Figure 5.16: Coverage results for the Kiosk (top) and Help-point (down) sensors. Each heatmap circle represents a location with detection coverage, accompanied by the mode power received. The locations with a black dot and a zero are locations with no sensor coverage.

detected at every test location. A, B, and C are the only locations that belong to the Picadeiro area, so the remaining locations should not be considered. As so, a packet power threshold that only considered the locations A, B, and C was intended. It is possible to observe a difference in the mode of powers received between the boundaries of Picadeiro, around -51 dB and -53 dB, and the remaining locations outside of Picadeiro. It is also possible to observe that the packet received power decreases with the distance from the sensor, which is expected due to the signal propagation loss through space. Considering these results, a packet power filtration of -55 dB

was applied to the Kiosk sensor, considering the locations A, B, C, and K. The latter was not possible to strict as the mode of power was as high as from the locations inside Picadeiro. This is due to the omnidirectional antenna equipped in the Kiosk sensor, that is near to this location. As a result, the detection range of this sensor could be delimited to the Picadeiro area as it was intended, and barely outside one of the entrances of Picadeiro. This also clearly shows the importance of the sensor positioning for detecting devices in target areas.

Regarding the Help-point sensor, all test locations had detection coverage except location G. This behaviour was expected since this sensor has a directional antenna pointed towards the Picadeiro area, and so devices behind the antenna were not detected due to its detection radiation. The same behaviour did not occur for the test location C for this sensor, where the device was detected despite this location being immediately behind the directional antenna. This can be justified by the fact that the directional antenna does not completely isolate its radiation in the opposite direction in which it is pointed. This also justifies the fact that the device was detected immediately behind the sensor at test location C, but not when it was relatively far from the sensor at test location G. The remaining locations had detection coverage, with received mode powers from -27 dB to -45 dB. The mode power values were significantly higher than the previous sensor, which is related to the different network cards used, with a different sensibility and performance from the card of the Kiosk sensor. In addition, it is also possible to notice that the received power decreases with the distance from the sensor due to the signal propagation loss through space. The test locations A and B were the only locations inside the Picadeiro, with received mode powers of -43 dB and -27 dB, respectively. Therefore, the packet power threshold needed to be at least -43 dB to comprise the entire Picadeiro area. However, the received powers obtained for the remaining locations outside Picadeiro had the same magnitude. Therefore, it was more difficult to determine a power threshold to delimit the detection range to the Picadeiro area, since the sensor detects devices that are within the Picadeiro area as well as those outside of Picadeiro, with similar received packet powers. Considering these results, a packet power filtration of -45 dB was applied to this sensor.

Table 5.9 summarizes the packet power filtration thresholds applied to each sensor.

Table 5.9: Packet power filtration thresholds applied to each sensor.

Sensor	Packet Power threshold
Kiosk	-55 dB
Help-point	-45 dB

The coverage tests conducted show that it is extremely necessary to take into account the physical limitations of the locations where the sensors are. The sensor positioning is also a crucial factor for detecting devices in the target area. As shown in this field experiment, the ideal location for detecting devices in Picadeiro was in the centre of the rectangle public square and then a packet power threshold could be delimited accurately to only detect devices inside the rectangle. However, as this was not possible due to physical and infrastructure limitations, a sensor was placed near one of the entrances with an omnidirectional antenna, which led to

detect devices barely outside of the target area.

5.3.4 Detections validation

After carrying out the coverage tests, the accuracy of the crowding sensors was validated regarding the number of devices detected and the real number of people present in the Picadeiro.

It is important to notice that the detections from the Kiosk sensor are much more reliable than the Help-point sensor by the fact that on the former a packet power filtration to only detect devices inside Picadeiro was more closely applied, while the same was not possible in the latter since it detected devices inside of Picadeiro as well as outside of it. Therefore, some of the detections from the Help-point sensor could also be relative to the flow of visitors at Pena Park's paths, thus not reliably attaining the intended reality of detection. As so, the most reliable detections are the ones from the Kiosk sensor.

This validation was carried out during one of the days in which the event of the [Portuguese School of the Equestrian Art](#) took place. As previously mentioned, this event took place in the Picadeiro and occurred during the last week of each summer month, from Tuesday to Friday. The event started at 3:30 PM and lasted 20 minutes. Validation was carried out over a period of approximately 1 hour, from 3:05 PM to 4:10 PM, on one of the days of this event. The event attracted quite a few visitors from the start, dragged by curiosity, and quickly an audience was formed around the venue, as presented in Figure 5.17.



Figure 5.17: Demonstration of the Portuguese School of the Equestrian Art event at Picadeiro.

For validation, the real number of people was counted during this period by direct observation of people. In order to obtain rigour in this validation, the number of visitors to Picadeiro was counted every 5 minutes (in accordance with the crowding measurements periodicity and the sliding window period of sensors). For this, at the beginning of every 5 minutes, the number of people inside Picadeiro was counted, and then only the number of entries on Picadeiro was counted. In this way, at the end of each 5 minutes, the total number of visitors to Picadeiro within the last 5 minutes was obtained.

The results obtained during the validation are presented in Figure 5.18, with the event time filled in orange.

As can be seen, the real number of people was always higher than the sensor detections during the entire validation, except for the first and last 5 minutes for the Help-point sensor, with little detections higher than the actual number of people. This can be justified by the fact

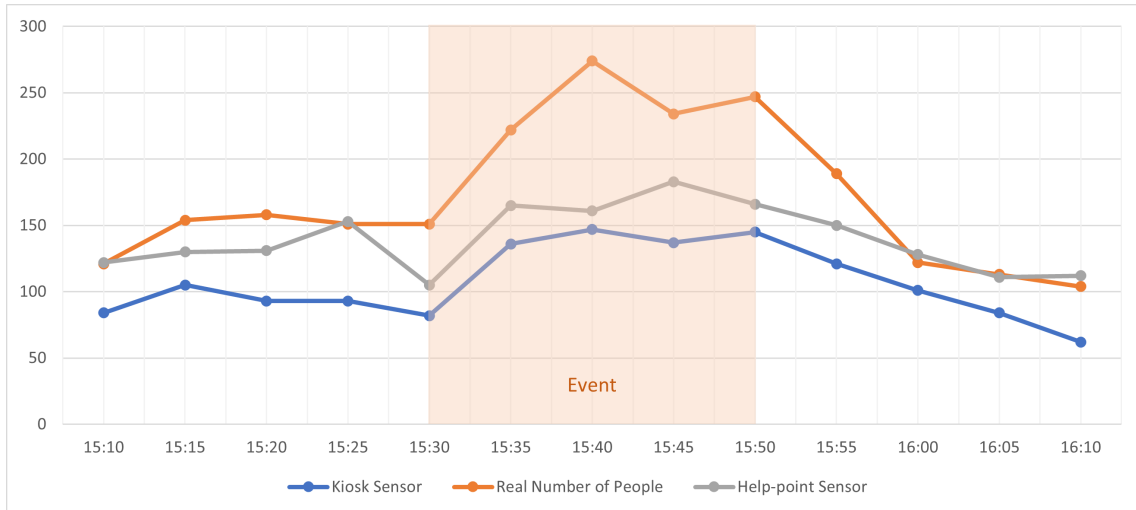


Figure 5.18: Results from validation at the Picadeiro. The event time is filled in orange.

that not every person could have a mobile phone, or even if so, its Wi-Fi interface could not be turned on, thus not being detected. The Help-point sensor had higher detections of mobile devices throughout the entire validation. This is justified by the consideration mentioned above, where the Help-point sensor easily detected devices outside of Picadeiro, namely immediately behind it, where there is usually a high flow of gathering people in the direction of Pena Palace, and also devices in the main path for the Pena Palace. As so, the constant deviation between the measurements of the two sensors and the real number of people can be explained by these reasons.

Before the event, the detections remain slightly constant as the real number of people. Then, when the event starts, the number of people rises, reaching a peak at 3:40 PM, then decreases a little, and starts dropping at 3:50 PM, when the event finishes. The same pattern can also be visualized on sensors, although with not-so-significant variations. The reason behind this could be that the fingerprinting technique can generate the same fingerprint for different devices with the same model, thus not coping with the slope of the increases and decreases.

A correlation function was applied to measure the collected data. This function calculates a Euclidean correlation between two arrays, in this case, two time sequences, resulting in a correlation coefficient between them. In this case, there are three time sequences: (1) the crowding measurements from the Kiosk sensor, (2) the crowding measurements from the Help-point sensor, and (3) the real number of people, as the ground-truth values. A coefficient closer to 1 indicates a closer correlation between the time sequences. The results obtained are shown in Table 5.10.

Table 5.10: Correlation coefficients between sensor detections and the real number of people.

Correlation between	Correlation coefficient
Kiosk and Real Number of people	0.9448
Help-point and Real Number of people	0.8502

As can be seen, the correlation between the Kiosk sensor and the real number of people has the highest correlation coefficient, with a value very close to 1, which means that the Kiosk

sensor can effectively correlate with the real number of people inside Picadeiro. On the other hand, the correlation between the Help-point sensor and the real number of people was also relatively close to 1, but lower compared to the Kiosk sensor. Although this sensor had absolute values closer to the real number of people compared to the Kiosk sensor, the results show that the detections from the Help-point sensor are not so correlated with the detections inside Picadeiro, which was expected due to the reasons explained above. Therefore, these results corroborate the coverage detection tests from both sensors, where the Kiosk sensor can perceive the crowding tendencies more precisely inside Picadeiro than the Help-point sensor.

Furthermore, for measuring the similarity of the results from both sensors to the ground-truth values, a [Dynamic Time Warping \(DTW\)](#) algorithm was also applied. This algorithm is suitable for this type of data and is widely used in the context of crowd counting, as in the works from [6, 12, 35]. The application of this algorithm is also important because it allows us to assess the similarity between the time sequences, even if there are minimal temporal divergences, something that is not considered in the correlation function.

The DTW was implemented in Python and considers the temporal distortion needed to effectively align two time sequences, just as performed for the correlation function described above. The number of detections from the three time sequences was normalized to only measure the correlation of the crowding tendencies, not considering the absolute values. A lower DTW value indicates that the two sequences have more temporal similarity. According to the Python implementation of the DTW algorithm, the results obtained are shown in Table 5.11.

Table 5.11: Temporal similarity between sensor detections and the real number of people.

	Score
Distance between	DTW
Kiosk and Real Number of people	1.084
Help-point and Real Number of people	1.924

The results are consistent with the correlation function applied above. The distance between the Kiosk sensor and the real number of people has a lower score value for both variants of the algorithm. This also indicates that the Kiosk sensor can perceive crowding tendencies with more correlation than the Help-point sensor, as was expected.

Furthermore, the detections allowed us to measure the impact of the public events held at Pena Park, as well as it allowed us to make more informed decisions to route visitors to other areas of Pena Park. Therefore, this field experiment showed that the STToolkit can have a positive impact on overcrowding management, something that was extremely important to evaluate.

In general, the detections from both sensors Kiosk and Help-point sensors reveal a good correlation with the crowding tendencies of the real number of people present in the Picadeiro area. The Kiosk sensor can definitely perceive crowding tendencies with more correlation, which was expected since this sensor was the most reliable due to the coverage tests applied and described in the previous subsection. As so, both sensors are capable of reproducing crowding tendencies in the Picadeiro area, with a clearer emphasis on the Kiosk sensor, which was expected to be validated in this subsection.

[This page has been intentionally left blank]

CHAPTER 6.

CONCLUSIONS AND FUTURE WORK

Contents

6.1	Conclusions	97
6.2	Future Work	99

This chapter draws conclusions from this dissertation and presents the future work to be leveraged.

[This page has been intentionally left blank]

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The main goal of this dissertation was a Smart Tourism Toolkit (STToolkit) for crowd-monitoring solutions. The latter should be capable of monitoring the occupation of spaces in order to trigger or assess the effectiveness of overcrowding mitigation actions. Therefore, a flexible, low-cost, and scalable crowding monitoring system has been developed. The toolkit has also been built in the scope of the [European RESETTING project](#) and is aimed at tourism-related SMEs that potentially deal with excessive crowding levels for optimizing their quality of service and service delivery, and to reduce the overwhelming feeling of pressure in critical hotspots.

To achieve the above-mentioned goals, and also to provide an answer to research question RQ1 (see section 1.4), a system architecture has been designed for the STToolkit. The latter is composed of multiple crowding sensors, deployed at several locations, each collecting the crowding information in their vicinity based on the wireless activity of mobile devices, and periodically uploading crowding levels to a cloud server. The crowding sensors may perform Wi-Fi and/or Bluetooth detection, thus providing flexibility in the technologies used for detecting devices. For communication, multiple uplink options are provided for uploading data to the cloud server, such as Wi-Fi or LoRaWAN protocols, something often disregarded by alternative approaches. Such flexible deployment mitigates the network limitations at the installation location of sensors. Moreover, user privacy rights are also ensured, as all gathered data from sensors is anonymized before being stored in the sensor's local database, not being possible to trace the original devices, as well as no sensitive data is uploaded on the cloud server.

The cloud server has components for making downlink communication transparent and provides uplink services for rendering spatiotemporal information through dashboards, with graphs for perceiving crowding tendencies along specific time ranges, maps to perceive the crowding geo-distribution, and also provides the creation of notification policies for alerting users of overcrowding situations.

Furthermore, the low-cost off-the-shelf hardware of sensors, also with open-source software, and the low hosting and communication cloud server costs, make the STToolkit a scalable solution.

For monitoring the occupation of spaces, a crowd detection approach based on sensing the number of mobile devices was developed. The crowding sensors are capable of sensing the mobile device's trace elements generated by their wireless activity, namely in Wi-Fi technology. One of the challenges imposed by this approach is due to the recent MAC Address randomization implemented on mobile devices which leads to overcounting estimations.

To tackle this issue, and also to provide an answer to research question RQ2, several strategies have been adopted to accurately estimate the number of devices, which can be summed in three main strategies: SSIDs comparison; Fingerprinting; and Fingerprinting + Clustering. Most strategies mainly rely on one of the last two.

In this case, a fingerprinting technique based on the Information Elements of probe request frames was developed to tackle the MAC address randomization issue. The results obtained from validation show that this fingerprinting technique can fairly tackle the MAC address randomization process, reducing its impact on device counting, as the number of fingerprints obtained is far closer to the ground-truth number of devices, compared to original approaches that performed device counting only based on the device's MAC addresses (not tackling the randomization issue). The results also strongly indicate that the developed fingerprinting technique has a high entropy to detect device models, but more steps are required to distinguish devices with the same model and OS version, as the same fingerprint was generated for those cases. Therefore, and also providing an answer to research question RQ4, we can conclude that it is possible to uniquely identify different device models, but the number of fingerprints can be a correlation between the number of device models and the real number of devices. Nevertheless, although preliminary, this approach revealed promising results to tackle the MAC address randomization issue for more accurate estimations, and can further be combined with other properties of device messages, such as the sequence numbers, burst sizes, or the time between messages in a burst for higher accuracies.

To test and validate the STToolkit, two field experiments were conducted in real crowded scenarios: a first one at Iscte's campus, and a second one at the National Palace of Pena, flattened by overtourism all year round.

A deployment at several locations across the Iscte's Campus was conducted to test and validate the STToolkit architecture and to evaluate the perception of the STToolkit in perceiving the crowding phenomena. From this field experiment, and also according to the research question RQ3, the results show that the sensors are capable of correlating the detections with the general use of spaces, either in high-permanency locations or in areas with a high traffic flow of people, and rapidly detect sudden increases and decreases of the number people. The results also showed that the sensors are capable of detecting highly-populated events and their duration, which is extremely important to perceive, as overcrowding situations have the same nature, characterized by a sudden and unpredictable increase in the number of people in an area. Moreover, the STToolkit architecture is also peer-reviewed, as publications related to this research work with this system architecture have already been submitted and accepted.

A second field experiment was conducted at Pena Park, a huge natural park that surrounds Pena Palace, to test and validate the final version of the Wi-Fi detection algorithm during a public event. From this experiment, detection coverage tests showed that is possible to delimit the detection range of sensors to specific areas with effectiveness by applying power filtration to the packets captured by the sensors, as it is also possible to detect devices in only specific directions by using dedicated hardware, such as directional antennas. This field experiment also showed that the sensor positioning is a crucial aspect to have in consideration to detect a desired area, as the miss-placement of the sensor could lead to detecting devices outside the intended area, for instance, deploying a sensor in the corner of a rectangle area, or not correctly pointing the directional antenna towards the correct direction. Furthermore, the validation carried out at this tourism site has reinforced the correlation of detections with the use of spaces that the sensors can provide, where the number of devices detected had the same crowding tendency compared to the real number of people.

Furthermore, this field experiment also showed that the STToolkit can have a positive impact on overcrowding management, as the detections from the sensors allowed to measure the impact of the public events held at Pena Park, as well as it allowed to make more informed decisions to route visitors to other areas of Pena Park.

In addition, these field experiments also contributed to demonstrating that the visualization setup created for the STToolkit, with dashboards with spatiotemporal rendering of the crowding information, allows an intuitive interpretation of crowding at the target locations where the sensors are deployed. Therefore, we can also conclude that the STToolkit allows easy visualization of the crowding phenomena at several locations in real-time, or quasi-real-time.

This research work also showed that is possible to provide multiple uplink options for communicating the crowding data to the cloud server, namely via Wi-Fi and LoRaWAN protocols. Although not implemented, preliminary tests have been carried out, not on a large scale, for the upload via LoRaWAN protocol, which proved that this communication option is valid for the STToolkit. The implementation of this communication option in large scale is envisaged for future work.

Last, but not least, for easiness in building, operating, and using the STToolkit, support material has been also provided, including installation, operation, and user manuals, online video tutorials, setup images, and source code availability, which allows general users with minimal technical skills to easily build, operate, and use a STToolkit.

Overall, this dissertation shows that the Smart Tourism Toolkit is capable of monitoring the occupation of several locations based on the wireless activity of mobile devices and perceiving the crowding phenomena with effectiveness, including overcrowding situations. Therefore, the STToolkit comes as a powerful tool to scaffold *overtourism* management, being validated with tests in real crowded scenarios and with accepted publications based on those tests. The STToolkit is then, a relevant and innovative tool for crowd-monitoring solutions.

6.2 Future Work

For future work, the research could be divided into three main threads, being them:

1. Multiple intelligent crowd detection;
2. Communication enhancements;
3. Overcrowding mitigation actions.

The research thread (1) regards the provision of multiple and intelligent crowd detection. For this, we could develop a more sophisticated and more accurate Wi-Fi detection algorithm, with improvements to the developed fingerprinting technique. To determine the relevant information for the fingerprinting technique, we could apply a [Semi-Supervised Learning \(SSL\)](#) approach to labelled datasets, from public datasets, and also from datasets collected in noisy and crowded scenarios at Iscte's Campus. Furthermore, we also intend to simultaneously combine the device fingerprints with other properties of device messages in a clustering algorithm, such as sequence numbers, burst size, or the time between messages in a burst, for more accurate precision of the number of devices, just like other similar approaches have proceeded.

Additionally, we also aim to perform Bluetooth detection of devices. As it is also a wireless technology extensively used by devices, combining the detected devices from both technologies could leverage more accurate crowd estimations. The latter will also imply device unification, as the same device can be detected simultaneously in both technologies.

In addition to the passive detection carried out by the sensors, we also plan to rely on user feedback for crowd counting in certain use case scenarios. For this, we aim to develop a dedicated mobile app, showing real-time occupancy levels where the sensors are deployed, allowing visitors to make informed decisions about which rooms to visit, helping them avoid waiting times, with the condition to consent being monitored by the app. The devices actively monitored by this application will also contribute to more precise crowd counting.

The research thread (2) regards communications enhancements. For this, we aim to implement the already tested data upload via LoRaWAN protocol in large-scale, and also test and validate other communication options for the STToolkit, such as Wi-Fi mesh, and Narrow-band IoT. For failure handling, an automatic handover mechanism of communication is also envisaged. The latter will allow an automatic switch in the communication protocol used in case of failure, for instance, switching to LoRaWAN upload if Wi-Fi connectivity is lost. Moreover, downlink communication is also envisaged, allowing users to define and control several parameters of the system with more ease, such as changing the data sampling rate of sensors and their geographic location.

Finally, in (3) we aim to perform overcrowding mitigation actions based on the crowding information collected, such as forward crowds to less-occupied areas in case of overcrowding situations. The mobile app mentioned above can contribute to this research thread, as the real-time occupancy can allow visitors to make informed decisions about which rooms to visit, or what is the best route to visit. Expected case studies include the [Pena Palace](#) in Sintra, where a field experiment was already carried out, the [Tagus Estuary Birdwatching and Conservation Area \(EVOA\)](#), campsites in the Lisbon area, and the Iscte's Campus itself.

BIBLIOGRAPHY

- [1] R. Agarwal, S. Kumar, and R. M. Hegde. “Algorithms for Crowd Surveillance Using Passive Acoustic Sensors Over a Multimodal Sensor Network.” In: *IEEE Sensors Journal* 15.3 (Mar. 2015), pp. 1920–1930. DOI: [10.1109/jsen.2014.2369474](https://doi.org/10.1109/jsen.2014.2369474).
- [2] A. Berenguer, D. F. Ros, A. Gómez-Oliva, J. A. Ivars-Baidal, A. J. Jara, J. Laborda, J.-N. Mazón, and A. Perles. “Crowd Monitoring in Smart Destinations Based on GDPR-Ready Opportunistic RF Scanning and Classification of WiFi Devices to Identify and Classify Visitors’ Origins.” In: *Electronics* 11.6 (Mar. 2022), p. 835. DOI: [10.3390/electronics11060835](https://doi.org/10.3390/electronics11060835).
- [3] M. Biendicho, E. Papaoikonomou, and D. Setó-Pamies. “Tourists Go Home! Examining Antitourism in Barcelona from an Emotions Perspective.” In: *Tourism Culture & Communication* 22.3 (Sept. 2022), pp. 275–295. DOI: [10.3727/109830421x16345418234010](https://doi.org/10.3727/109830421x16345418234010).
- [4] T. Bravenec, J. Torres-Sospedra, M. Gould, and T. Fryza. “What Your Wearable Devices Revealed About You and Possibilities of Non-Cooperative 802.11 Presence Detection During Your Last IPIN Visit.” In: *Proceedings of the 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, Sept. 2022. DOI: [10.1109/ipin54987.2022.9918134](https://doi.org/10.1109/ipin54987.2022.9918134).
- [5] Y. Cai, M. Tsukada, H. Ochiai, and H. Esaki. “MAC address randomization tolerant crowd monitoring system using Wi-Fi packets.” In: *Asian Internet Engineering Conference, AINTEC 2021* (Dec. 2021), pp. 27–33. DOI: [10.1145/3497777.3498547](https://doi.org/10.1145/3497777.3498547).
- [6] M. Chen, X. Yang, Y. Jin, and M. Zhou. “Dynamic Time Warping Based Passive Crowd Counting Using WiFi Received Signal Strength.” In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12737 LNCS (2021), pp. 667–677. DOI: [10.1007/978-3-030-78612-0_54](https://doi.org/10.1007/978-3-030-78612-0_54). URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85112045379&doi=10.1007%2f978-3-030-78612-0_54&partnerID=40&md5=7b17c7dc85602087d1d5deabb4b17354.
- [7] A.-I. Covaci. *Wi-Fi MAC address randomization vs Crowd Monitoring*. Tech. rep. University of Twente, July 2022. URL: <http://essay.utwente.nl/91744/>.
- [8] M. De Nadai, J. Staiano, R. Larcher, N. Sebe, D. Quercia, and B. Lepri. “The Death and Life of Great Italian Cities.” In: *Proceedings of the 25th International Conference on World Wide Web* (Apr. 2016). DOI: [10.1145/2872427.2883084](https://doi.org/10.1145/2872427.2883084).
- [9] R. Dias da Silva. “A tourism overcrowding sensor using multiple radio techniques detection.” Master’s thesis. Iscte, 2019. URL: <http://hdl.handle.net/10071/20212>.

- [10] R. Dias da Silva, R. Neto Marinheiro, and F. Brito e Abreu. “Crowding Detection Combining Trace Elements from Heterogeneous Wireless Technologies.” In: *Proceedings of the 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC)*. IEEE, Nov. 2019. DOI: [10.1109/wpmc48795.2019.9096131](https://doi.org/10.1109/wpmc48795.2019.9096131).
- [11] V. X. Gong, W. Daamen, A. Bozzon, and S. P. Hoogendoorn. “Crowd characterization for crowd management using social media data in city events.” In: *Travel Behaviour and Society* 20 (July 2020), pp. 192–212. DOI: [10.1016/j.tbs.2020.03.011](https://doi.org/10.1016/j.tbs.2020.03.011).
- [12] G. He, B. Li, H. Wang, and W. Jiang. “Cost-Effective Active Semi-Supervised Learning on Multivariate Time Series Data with Crowds.” In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.3 (2022), pp. 1437–1450. DOI: [10.1109/TSMC.2020.3019531](https://doi.org/10.1109/TSMC.2020.3019531). URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85125368888&doi=10.1109%2fTSMC.2020.3019531&partnerID=40&md5=e6570c0a8a0e26bfe6e28f0547b4beec>.
- [13] T. He, J. Tan, and S.-H. G. Chan. “Self-Supervised Association of Wi-Fi Probe Requests Under MAC Address Randomization.” In: *IEEE Transactions on Mobile Computing* (2022), pp. 1–14. DOI: [10.1109/tmc.2022.3205924](https://doi.org/10.1109/tmc.2022.3205924).
- [14] Q. Hu, G. Bai, S. Wang, and M. Ai. “Extraction and monitoring approach of dynamic urban commercial area using check-in data from Weibo.” In: *Sustainable Cities and Society* 45 (Feb. 2019), pp. 508–521. DOI: [10.1016/j.scs.2018.11.039](https://doi.org/10.1016/j.scs.2018.11.039).
- [15] *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. DOI: [10.1109/ieeestd.2021.9363693](https://doi.org/10.1109/ieeestd.2021.9363693).
- [16] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L.-S. Peh. “Low cost crowd counting using audio tones.” In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems* (Nov. 2012). DOI: [10.1145/2426656.2426673](https://doi.org/10.1145/2426656.2426673).
- [17] M. Kristoffersen, J. Dueholm, R. Gade, and T. Moeslund. “Pedestrian Counting with Occlusion Handling Using Stereo Thermal Cameras.” In: *Sensors* 16.1 (Jan. 2016), p. 62. DOI: [10.3390/s16010062](https://doi.org/10.3390/s16010062).
- [18] A. López-Cifuentes, M. Escudero-Viñolo, J. Bescós, and P. Carballeira. “Semantic-driven multi-camera pedestrian detection.” In: *Knowledge and Information Systems* 64.5 (Apr. 2022), pp. 1211–1237. DOI: [10.1007/s10115-022-01673-w](https://doi.org/10.1007/s10115-022-01673-w).
- [19] L. Maddalena, A. Petrosino, and F. Russo. “People counting by learning their appearance in a multi-view camera environment.” In: *Pattern Recognition Letters* 36 (Jan. 2014), pp. 125–134. DOI: [10.1016/j.patrec.2013.10.006](https://doi.org/10.1016/j.patrec.2013.10.006).
- [20] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen. “Acoustic event detection in real life recordings.” In: *European Signal Processing Conference* (Aug. 2010), pp. 1267–1271. URL: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=7096611.

- [21] T. Mestre Santos, R. Neto Marinheiro, and F. Brito e Abreu. “Making Tourists Experience Smarter by Mitigating Overtourism.” In: *35th International Conference on Advanced Information Systems Engineering. Third International Workshop on Information Systems Engineering for Smarter Life (ISESL)*. Springer, June 2023. URL: <http://isesl23.cnam.fr/program.php>.
- [22] T. Mestre Santos, R. Neto Marinheiro, and F. Brito e Abreu. “Wireless Crowd Detection for Smart Overtourism Mitigation.” In: *Smart Life and Smart Life Engineering: Current State and Future Vision – Lecture Notes in Business Information Processing*. Springer, Dec. 2023, pp. 1–20.
- [23] T. Mestre Santos, R. Neto Marinheiro, and F. Brito e Abreu. “Wireless Sensor for Tourism Overcrowding.” In: *International Workshop of RESETTING and Poster exhibition*. Jan. 2023. URL: <https://sites.google.com/iscte-iul.pt/resetting-project/home/posters-workshop>.
- [24] L. Oliveira, D. Schneider, J. De Souza, and W. Shen. “Mobile Device Detection Through WiFi Probe Request Analysis.” In: *IEEE Access* 7 (2019), pp. 98579–98588. DOI: [10.1109/access.2019.2925406](https://doi.org/10.1109/access.2019.2925406).
- [25] S. Park, M. Bourqui, and E. Frias-Martinez. “MobInsight: Understanding Urban Mobility with Crowd-Powered Neighborhood Characterizations.” In: *Proceedings of the 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, Dec. 2016. DOI: [10.1109/icdmw.2016.0192](https://doi.org/10.1109/icdmw.2016.0192).
- [26] P. Peeters, S. Gössling, J. Klijs, C. Milano, M. Novelli, C. Dijkmans, E. Eijgelaar, S. Eijgelaar, J. Hartman, R. Heslinga, O. Isaac, S. Mitas, J. Moretti, B. Nawijn, A. Papp, and Postma. *Research for TRAN Committee-Overtourism: impact and possible policy responses*. Tech. rep. Brussels: European Parliament, 2018.
- [27] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. “A Design Science Research Methodology for Information Systems Research.” In: *Journal of Management Information Systems* 24.3 (Dec. 2007), pp. 45–77. DOI: [10.2753/mis0742-1222240302](https://doi.org/10.2753/mis0742-1222240302).
- [28] L. Pintor and L. Atzori. “A dataset of labelled device Wi-Fi probe requests for MAC address de-randomization.” In: *Computer Networks* 205 (2022), p. 108783. ISSN: 1389-1286. DOI: [10.1016/j.comnet.2022.108783](https://doi.org/10.1016/j.comnet.2022.108783).
- [29] H. Seraphin, P. Sheeran, and M. Pilato. “Over-tourism and the fall of Venice as a destination.” In: *Journal of Destination Marketing & Management* 9 (Sept. 2018), pp. 374–376. DOI: [10.1016/j.jdmm.2018.01.011](https://doi.org/10.1016/j.jdmm.2018.01.011).
- [30] U. Singh, J.-F. Determe, F. Horlin, and P. De Doncker. “Crowd Monitoring: State-of-the-Art and Future Directions.” In: *IETE Technical Review* 38.6 (Aug. 2020), pp. 578–594. DOI: [10.1080/02564602.2020.1803152](https://doi.org/10.1080/02564602.2020.1803152).
- [31] O. Tokarchuk, J. C. Barr, and C. Cozzio. “How much is too much? Estimating tourism carrying capacity in urban context using sentiment analysis.” In: *Tourism Management* 91 (Aug. 2022), p. 104522. DOI: [10.1016/j.tourman.2022.104522](https://doi.org/10.1016/j.tourman.2022.104522).

- [32] M. Uras, R. Cossu, E. Ferrara, O. Bagdasar, A. Liotta, and L. Atzori. “WiFi Probes sniffing: an Artificial Intelligence based approach for MAC addresses de-randomization.” In: *Proceedings of the 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, Sept. 2020. DOI: [10.1109/camad50429.2020.9209257](https://doi.org/10.1109/camad50429.2020.9209257).
- [33] M. Uras, E. Ferrara, R. Cossu, A. Liotta, and L. Atzori. “MAC address de-randomization for WiFi device counting: Combining temporal- and content-based fingerprints.” In: *Computer Networks* 218 (Dec. 2022), p. 109393. DOI: [10.1016/j.comnet.2022.109393](https://doi.org/10.1016/j.comnet.2022.109393).
- [34] M. Vega-Barbas, M. Álvarez-Campana, D. Rivera, M. Sanz, and J. Berrocal. “AFOROS: A Low-Cost Wi-Fi-Based Monitoring System for Estimating Occupancy of Public Spaces.” In: *Sensors* 21.11 (June 2021), p. 3863. DOI: [10.3390/s21113863](https://doi.org/10.3390/s21113863).
- [35] Y. Xiao, J. Xu, M. Chraibi, J. Zhang, and C. Gou. “A generalized trajectories-based evaluation approach for pedestrian evacuation models.” In: *Safety Science* 147 (2022). DOI: [10.1016/j.ssci.2021.105574](https://doi.org/10.1016/j.ssci.2021.105574). URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85119370277&doi=10.1016%2Fj.ssci.2021.105574&partnerID=40&md5=5df8ab74f263bd3731c0572fe3e59dae>.
- [36] S. Xu, S. Li, W. Huang, and R. Wen. “Detecting spatiotemporal traffic events using geosocial media data.” In: *Computers, Environment and Urban Systems* 94 (June 2022), p. 101797. DOI: [10.1016/j.compenvurbsys.2022.101797](https://doi.org/10.1016/j.compenvurbsys.2022.101797).
- [37] Z. Xu, L. Mei, Z. Lv, C. Hu, X. Luo, H. Zhang, and Y. Liu. “Multi-Modal Description of Public Safety Events Using Surveillance and Social Media.” In: *IEEE Transactions on Big Data* 5.4 (Dec. 2019), pp. 529–539. DOI: [10.1109/tbdata.2017.2656918](https://doi.org/10.1109/tbdata.2017.2656918).
- [38] Q. Yang and L. Huang. *Inside Radio: An Attack and Defense Guide*. New York, United States: Springer Publishing, 2018.

APPENDIX **A** ■■

STToolKIT CLOUD SERVER INSTALLATION MANUAL



D3.5 - Toolkit for implementing tourism crowd detection solutions

Cloud Server Installation Manual

This **Cloud Server Installation Manual** includes guidelines and instructions for the integration/configuration of the STToolkit's Cloud Server. It is targeted for the **STToolkit Instantiator role**. This manual includes the step-by-step detailed tutorial for the installation and configuration of the Cloud Server of the STToolkit, and the software that live on them.



This project has received funding from the COSME Programme (EISMEA) under grant agreement No.101038190

Contents

CLOUD SERVER INSTALLATION AND CONFIGURATION.....	3
1. Cloud Server Acquisition.....	3
2. Cloud Server Hardening.....	5
3. InfluxDB database Installation and Configuration.....	10
3.1 <i>InfluxDB database installation</i>	10
3.2 <i>InfluxDB database configuration</i>	11
4. Grafana Installation and Configuration.....	15
4.1 <i>Grafana Installation</i>	15
4.2 <i>Grafana Configuration</i>	17
REFERENCES.....	20



Cloud Server Installation and Configuration

Follow the steps below in the correct order to install and configure the STToolkit's Cloud Server. The steps for chapters 1 and 2 are also available as a video tutorial on the [RESETTING Youtube channel](#). The steps for chapters 3 and 4 are also available as a video tutorial on the [RESETTING Youtube channel](#).

1. Cloud Server Acquisition

These steps regard the acquisition of a Virtual Private Server (VPS) for hosting the STToolkit's Cloud Server. The steps reproduce the acquisition of a VPS from [Contabo](#), however, any other VPS provider is valid for the STToolkit.

- 1) Create a user account on [Contabo](#).
- 2) Once logged in, on the 'New Order' yellow panel, select 'VPS'.

Several VPS configurations should appear, with different cloud resources and monthly charges.

- 3) Select a VPS configuration.

Note: We recommend choosing a VPS with at least **4 CPUs** and **4 GB RAM**. In the case of Contabo, the VPS configuration that fits these recommended requirements is a VPS with **4 CPUs** and **8 GB RAM**.

- 4) Configure the VPS.

Insert the following server configurations:

- i) Term length: (as desired by you)
- ii) Region: European Union (Germany)
- iii) Storage Type: 200 GB SSD or 400 GB SSD
- iv) Image: Ubuntu 22.04
- v) Login & password for your server:
 - a. Username: root
 - b. Password: (as desired by you)

Note: Please do not forget this password! It will be later necessary to remotely access the VPS.

- vi) Networking:
 - a. Private Networking: No Private Networking
 - b. IPv4: 1 IP Address
- vii) Add-ons: None

- 5) Order the VPS.



- You should receive an email from Contabo confirming your payment and requiring some data verification before setting up the server. Please respond to this e-mail with the required information, as only after this verification procedure the server will be ready for use.

Note: If you made this step on a weekend, the verification should only be proceeded on the next working day. If so, please be patient and wait for the next working day.

- After the data verification process from Contabo, you should receive an e-mail with your login data, with the server public IP address and other useful information of the server, similar to the one shown below.



Dear *********,

With this e-mail we would like to welcome you once again at Contabo!
Your order has been processed successfully. You can find all login data and details necessary to manage your order below.

If you just started using our services, we recommend you check our [First Steps with Contabo](#) tutorial.

Customer ID 123281148 (Order ID 123281148)

Please quote your customer ID in all communications with us, whether on the telephone or by e-mail. This ensures the fastest processing of your requests.

Your VPS

IP address	server type	Location	VNC IP and port	VNC password	user name	password
192.168.1.100	1GB 1 vCPU	Frankfurt, DE	192.168.1.100:5901	*****	root	as chosen by you during order process

The Cloud Server is now available for the STToolkit. Let's try to remotely access it via SSH.

- Open a command line terminal.

Windows:

Open the **Start menu** or press the **Windows** key, and type **cmd**. Press Enter.

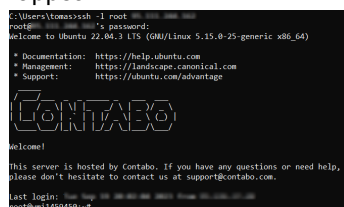
MacOS:

Go to **Applications > Utilities** and double-click on **Terminal**.

- Remotely access the Cloud Server via SSH using the root user credentials.

```
$ sudo ssh -l root <server_ip_address>
```

Insert your root password, created in step 4.v) when the VPS was ordered. A similar output on the terminal should then appear.



Success! You have now accessed the cloud server of the STToolkit via SSH. You can now proceed to the [Server Hardening](#) section.



2. Cloud Server Hardening

These steps aim to customize and give more protection and security to the Cloud Server.

Please follow the steps below in the correct order.

1) Create a new *sttoolkit* user

i) Log into your Cloud Server via SSH:

```
$ ssh root@<server-ip-address>
```

ii) Create a new *sttoolkit* user:

```
$ adduser sttoolkit
```

iii) Then, add *sttoolkit* user to sudo group (Super User):

```
$ usermod -aG sudo sttoolkit
```

iv) Log into the Cloud Server via SSH with the new user:

```
$ ssh sttoolkit@<server-ip-address>
```

v) Confirm that you can update the server with the new user:

```
$ sudo apt update && sudo apt upgrade
```

vi) Restart the server:

```
$ sudo shutdown -r now
```

2) Disable root user login

After creating the new *sttoolkit* user, the initial root user is no longer needed. Therefore, we can disable its login.

i) Login and open the SSH configuration file using the Nano editor. (Guide of how to use the Nano editor [here](#).)

```
$ sudo nano /etc/ssh/sshd_config
```

- ii) Look for the PermitRootLogin line, uncomment it (remove the #) and set the value to "no".

```
PermitRootLogin no
```

- iii) To apply the new settings, restart SSH service:

```
$ sudo service ssh restart
```

3) Set Up SSH Keys only login

The most secure way to connect to the server is by using SSH keys only, and not using account passwords. Therefore, we can set up logins only using SSH keypairs, and disable the username/password authentication method.

- i) Create a .ssh directory on the Cloud Server:

```
$ mkdir ~/.ssh
```

- ii) Create a new SSH keypair on the client machine (usually your personal computer):

Windows:

Open a new PowerShell window by clicking on the 'Windows' key and type 'Windows PowerShell'.

On the PowerShell, generate a new SSH keypair with the ssh-keygen command.

```
$ ssh-keygen
```

Press Enter for all parameters.

MacOS:

On a terminal window, create a new SSH keypair using the ssh-keygen command

```
$ ssh-keygen
```

Press Enter for all parameters.

- iii) Copy the SSH public key to the Cloud Server:

Windows:



On the PowerShell window, copy the *id_rsa.pub* public key to the Cloud Server using the command below. Replace the <server_ip_address> with the IP address of the Cloud Server.

```
type $env:USERPROFILE\.ssh\id_rsa.pub | ssh \  
sttoolkit@<server_ip_address> "cat >> .ssh/authorized_keys"
```

MacOS:

Copy your public key to the Cloud Server using the command below. Replace the <server_ip_address> with the IP address of the Cloud Server.

```
$ ssh-copy-id sttoolkit@<server-ip-address>
```

iv) Log into the remote host without providing the remote account's password:

```
$ ssh sttoolkit@<server-ip-address>
```

If successful, notice that your client machine logged in to the Cloud Server without using any password. We can now remove the login via SSH using account passwords.

Note: Only proceed with the following steps if you successfully accomplished the previous step, otherwise you can lose connection to the Cloud Server. If you did not accomplish the previous steps, you can stick with the username/password authentication method and proceed with step 4) of this manual.

v) Open up the SSH daemon's configuration file:

```
$ sudo nano /etc/ssh/sshd_config
```

vi) Disable your ability to log in via SSH using account passwords by looking for the *PasswordAuthentication* line, uncomment it (remove the #) and set the value to no:

```
PasswordAuthentication no
```

At this point, the only way to log in to the Cloud Server is using SSH keys. Only the client machines that setup a SSH keypair with the Cloud server (described in steps ii) and iii)) will be able to connect to the Cloud Server. **You can repeat the steps ii), iii) and iv) for each client machine you wish to connect to the Cloud Server.**

4) Set local date (optional)

Run the *dpkg-reconfigure* command to set the datetime according to your timezone.

```
$ sudo dpkg-reconfigure tzdata
```

5) Setup Automatic Updates

- i) Install the unattended-upgrades package and activate it.

```
$ sudo apt install unattended-upgrades
```

- ii) Enable automatic updates in order to generate the files you need to customize:

```
$ sudo dpkg-reconfigure --priority=low unattended-upgrades
```

- iii) Enable auto reboot and notifications:

```
$ sudo apt install update-notifier-common
```

- iv) Open the 50unattended-upgrades file with Vim to configure what you need to be updated.

```
$ sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Enable reboots after updating. Locate the line below and uncomment it, then update them accordingly.

```
Unattended-Upgrade::Automatic-Reboot "true";
```

You can set a specific time for the reboot based on the 24hour clock system. We recommend the reboot for the early morning, such as 4 AM:

```
Unattended-Upgrade::Automatic-Reboot-Time "04:00"
```

6) Change Server Hostname

- i) Set your Ubuntu server hostname as follows.

```
$ sudo hostnamectl set-hostname crowdingServer
```

- ii) Edit the hosts file – /etc/hosts

```
$ sudo nano /etc/hosts
```

Change the old name to 'crowdingServer' in the hosts file.

The file should look as follows:



```
127.0.0.1    localhost
127.0.1.1    crowdingServer  crowdingServer

# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
ff02::3     crowdingServer  crowdingServer
```

iii) Restart the server:

```
$ sudo shutdown -r now
```

7) Enable Firewall

i) Allow SSH connections so that you can log into your server next time by typing:

```
$ sudo ufw allow OpenSSH
```

ii) Run the following command in order to allow Webmin through the firewall:

```
$ sudo ufw allow 10000
```

iii) Enable the HTTP port:

```
$ sudo ufw allow 80
```

iv) Enable the InfluxDB port:

```
$ sudo ufw allow 8086
```

v) Enable the Grafana port:

```
$ sudo ufw allow 3000
```

vi) Enable the firewall:

```
$ sudo ufw enable
```

vii) You can see what network connections are allowed by typing:

```
$ sudo ufw status
```




The firewall is now active and enabled on the Cloud Server, and will only allow network traffic from the applications needed for the STToolkit.

The Cloud Server is now more secure and protected. It is now necessary to install and configure the software components for crowd monitoring, namely:

- The InfluxDB database, responsible for the data ingestion of all crowding sensors;
- The Grafana data visualization platform that will render the collected information from sensors to end users.

3. InfluxDB database Installation and Configuration

3.1 InfluxDB database installation

The steps given below describe the installation of the InfluxDB database on the Cloud Server.

1) Start with a system update

```
$ sudo apt-get update
```

2) Import the InfluxDB GPG key

```
$ wget -q https://repos.influxdata.com/influxdata-archive_compat.key
```

3) Add APT repository

```
$ echo  
'393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c \  
influxdata-archive_compat.key' | sha256sum -c && cat \  
influxdata-archive_compat.key | gpg --dearmor | sudo tee \  
/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg > /dev/null
```

```
$ echo 'deb  
[signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] \  
https://repos.influxdata.com/debian stable main' | sudo tee \  
/etc/apt/sources.list.d/influxdata.list
```

4) Update the system

```
$ sudo apt-get update
```

5) Install InfluxDB on the server

```
$ sudo apt-get install influxdb2
```



6) Start and Enable the InfluxDB service

```
$ sudo systemctl start influxdb
$ sudo systemctl enable influxdb
```

7) Check the status of the InfluxDB service

```
$ sudo systemctl status influxdb
```

An output similar to the following should appear. If the status of the InfluxDB service appears as 'active (running)', the InfluxDB service is available and running on the Cloud Server.

```
root@vmi1459450:~# sudo systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time series database
   Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-09-19 20:30:46 CEST; 5h 26min ago
     Docs: https://docs.influxdata.com/influxdb/
    Main PID: 4598 (influxd)
      Tasks: 11 (limit: 9457)
     Memory: 52.8M
        CPU: 31.994s
    CGroup: /system.slice/influxdb.service
            └─4598 /usr/bin/influxd

Sep 19 23:30:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T21:30:46.839061Z lvl=info msg="
Sep 19 23:30:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T21:30:46.841931Z lvl=info msg="
Sep 20 00:00:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T22:00:46.844342Z lvl=info msg="
Sep 20 00:00:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T22:00:46.851257Z lvl=info msg="
Sep 20 00:30:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T22:30:46.843526Z lvl=info msg="
Sep 20 00:30:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T22:30:46.848431Z lvl=info msg="
Sep 20 01:00:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T23:00:46.948204Z lvl=info msg="
Sep 20 01:00:47 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T23:00:46.962993Z lvl=info msg="
Sep 20 01:30:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T23:30:46.845342Z lvl=info msg="
Sep 20 01:30:46 vmi1459450.contaboserver.net influxd-systemd-start.sh[4598]: ts=2023-09-19T23:30:46.854114Z lvl=info msg="
```

The InfluxDB database is now installed in the Cloud Server. It is now needed to configure and setup this database.

3.2 InfluxDB database configuration

The configuration of the InfluxDB database can be done in just a few simple steps using the InfluxDB User Interface (UI).

Let's open the InfluxDB UI.

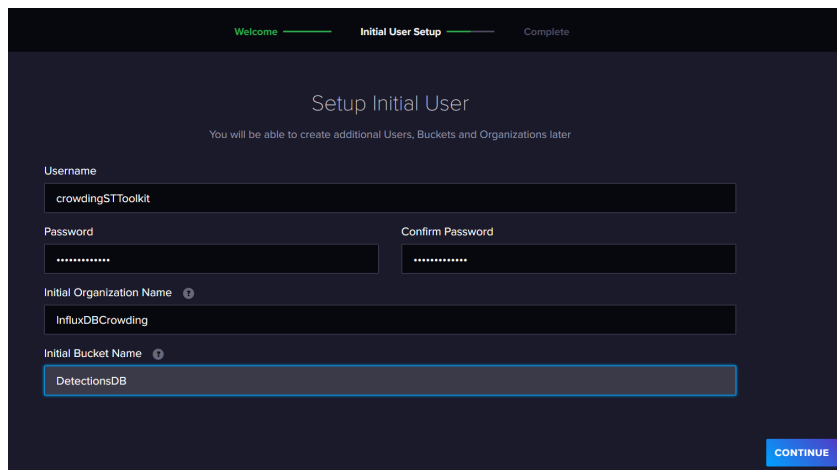
- 1) Open a browser and type: **http://<server_ip_address>:8086**
- 2) A window similar to the following should appear on your browser. This is how you can access the InfluxDB database. Click on 'Get Started'.
- 3) An initial user setup is required.

Insert the following fields:

- Username: **crowdingSTToolkit**

- Password: (as desired by you)
- Initial Organization Name: **InfluxDBCrowding**
- Initial Bucket Name: **DetectionsDB**

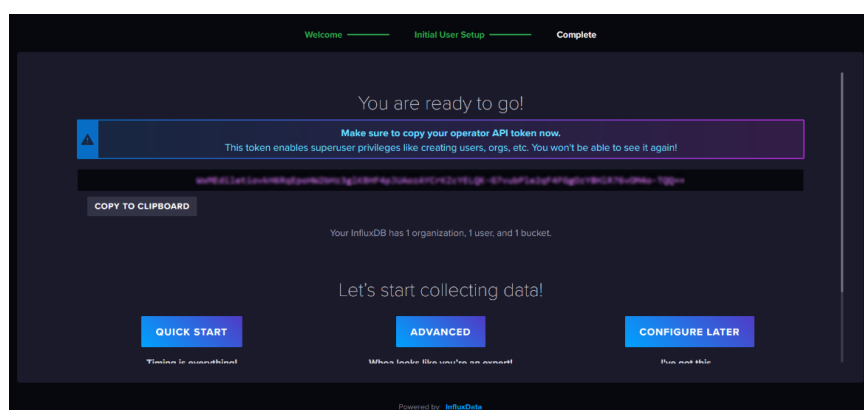
Note: Please make sure that you correctly insert these fields, without spaces. These will be the parameters that the multiple crowding sensors will need to also set for updating information to this InfluxDB database, as well the parameters for establishing a Grafana connection, as further described in [Grafana Configuration](#).



This will be the initial configuration for the user, organization, and bucket. No additional users, organizations, or buckets will be further needed to be created for the STToolkit.

Click on 'Continue'.

- 4) The initial user setup was completed, and an authorization token with superuser privileges appears. This superuser token enables the creation of users, orgs, buckets, etc.

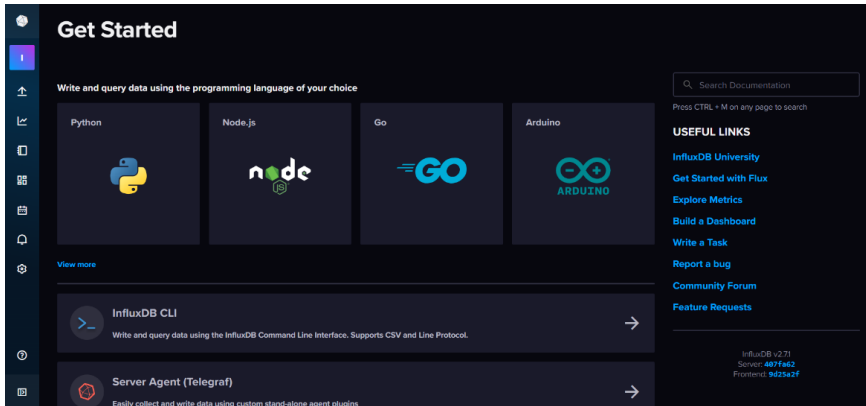




Note: Please make sure you copy and save this superuser token, as you won't be able to generate it and see it again!

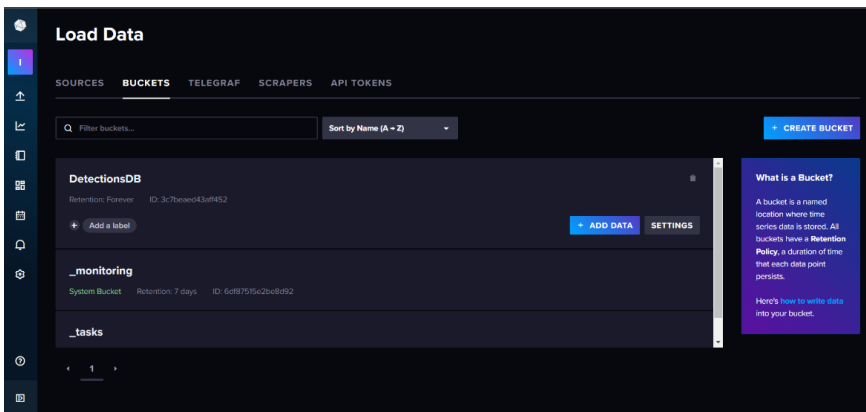
Click on 'Configure Later', as no other configurations are required.

- 5) After this, the main menu of the InfluxDB UI appears. Here you can manage the InfluxDB database, such as manage organizations, buckets, and authorization tokens.



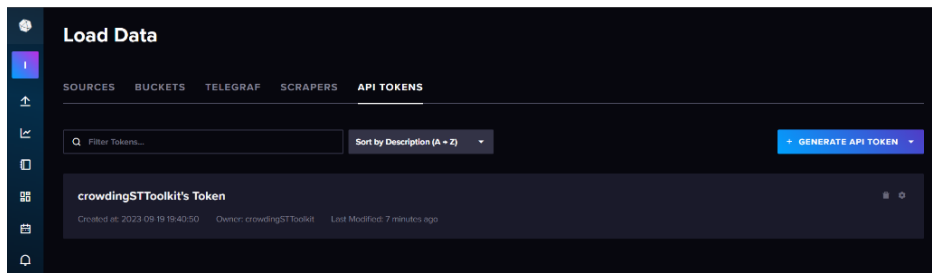
- 6) Get familiar with the InfluxDB Buckets:

On the left side bar, click on 'Buckets'. This will direct you to the buckets that you have in the InfluxDB database, namely, the initial 'DetectionsDB' bucket created in the initial setup.



- 7) Get familiar with the InfluxDB API Tokens:

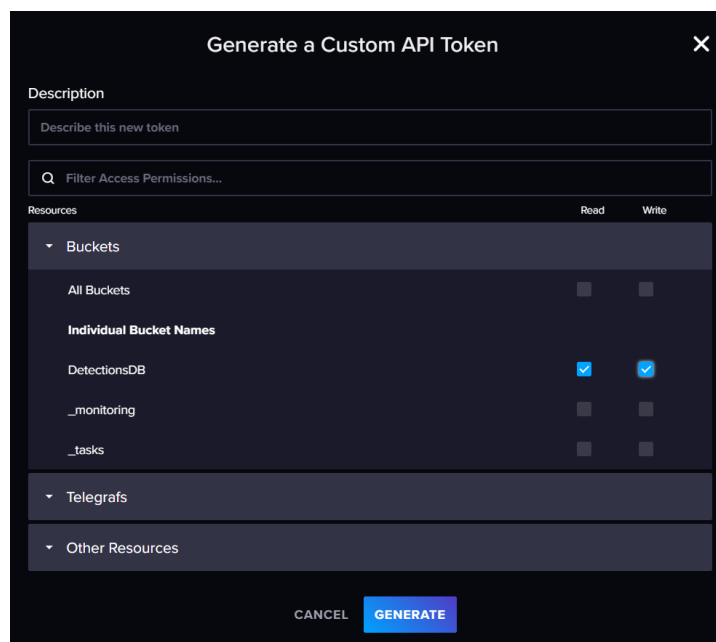
On the left side bar, click on 'API Tokens'. This will direct you to the API Tokens that you have created in the InfluxDB database. The superuser token appears, here you can see its permissions, although you cannot obtain it again.



Let's create another API token that will be used for the sensors to upload the crowding information to this InfluxDB database. (It is not recommended to use the superuser token for that purpose).

8) Create a new API Token.

On the 'API Tokens' tab, click on 'Generate API Token' > 'Custom API Token'. On Resources, click on 'Buckets', and only mark the 'read' and 'write' options for the 'DetectionsDB' bucket. Then, click on 'Generate'.



You have successfully created a new API Token. **Please make sure you copy and save this token, as you won't be able to see it again!**

You can also change the name of the new API Token as you wish.

Everything is now set for the InfluxDB database. You can now log out if you want.



Note: From now on, for accessing the InfluxDB database from your browser, you only have to repeat the step 1), and insert the user credentials created for the InfluxDB in the initial setup on step 3):

- Username: crowdingSTToolkit
- Password: (as chosen by you in the initial setup in step 3))

4. Grafana Installation and Configuration

4.1 Grafana Installation

Complete the following steps to install the Grafana data visualization platform in the Cloud Server.

i) Install the prerequisite packages:

```
$ sudo apt-get install -y apt-transport-https \
software-properties-common wget
```

ii) Import the GPG key:

```
$ sudo mkdir -p /etc/apt/keyrings/

$ wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | \
sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null
```

iii) Add a repository for stable releases:

```
$ echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] \
https://apt.grafana.com stable main" | sudo tee -a \
/etc/apt/sources.list.d/grafana.list
```

iv) Update the list of available packages:

```
$ sudo apt-get update
```

v) Install Grafana:

```
$ sudo apt-get install grafana
```

vi) Start the Grafana service:

```
$ sudo systemctl start grafana-server
```

vii) Check if the Grafana service is running:

```
$ sudo systemctl status grafana-server
```

An output similar to the following should appear. If the status of the InfluxDB service appears as 'active (running)', the InfluxDB service is available and running on the Cloud Server.

```
root@vmi1459450:~# sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-09-19 21:12:20 CEST; 23h ago
     Docs: http://docs.grafana.org
   Main PID: 6354 (grafana)
    Tasks: 14 (limit: 9457)
   Memory: 76.8M
     CPU: 6min 46.051s
   CGroup: /system.slice/grafana-server.service
           └─6354 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana

Sep 20 19:52:30 vmi1459450.contaboserver.net grafana[6354]: logger=cleanup t=2023-09-20T19:52:30.673135077+02:00 level=info
Sep 20 19:52:30 vmi1459450.contaboserver.net grafana[6354]: logger=plugins.update.checker t=2023-09-20T19:52:30.77494757
Sep 20 19:52:30 vmi1459450.contaboserver.net grafana[6354]: logger=grafana.update.checker t=2023-09-20T19:52:30.81232264
Sep 20 20:02:30 vmi1459450.contaboserver.net grafana[6354]: logger=cleanup t=2023-09-20T20:02:30.660263147+02:00 level=info
Sep 20 20:02:30 vmi1459450.contaboserver.net grafana[6354]: logger=plugins.update.checker t=2023-09-20T20:02:30.7764423+
Sep 20 20:02:30 vmi1459450.contaboserver.net grafana[6354]: logger=grafana.update.checker t=2023-09-20T20:02:30.81391427
Sep 20 20:12:30 vmi1459450.contaboserver.net grafana[6354]: logger=cleanup t=2023-09-20T20:12:30.676171459+02:00 level=info
Sep 20 20:12:30 vmi1459450.contaboserver.net grafana[6354]: logger=grafana.update.checker t=2023-09-20T20:12:30.81579702
Sep 20 20:12:30 vmi1459450.contaboserver.net grafana[6354]: logger=plugins.update.checker t=2023-09-20T20:12:30.83484990
Sep 20 20:13:09 vmi1459450.contaboserver.net grafana[6354]: logger=infra.usagestats t=2023-09-20T20:13:09.025878168+02:00
```

4.2 Grafana Configuration

After the installation of Grafana, it is now necessary to configure this data visualization platform for the STToolkit.

1) Sign in to Grafana

- i) Open a browser and type the following URL: **http://<server_ip_address>:3000**

The Grafana login panel should appear.

- ii) On the signin page, enter **admin** for username and password.

- iii) Click **Sign In**.

If successful, you will see a prompt to change the password.

- iv) Click **OK** on the prompt and change your password.

Note: We strongly recommend that you change the default administrator password.

After this, you will be prompted to the Grafana Main Menu panel.

2) Configure the InfluxDB data source

- i) Click Connections in the left-side menu.



ii) Click **Data source > Add new data source**.

iii) Enter **InfluxDB** in the search bar.

iv) Select **InfluxDB**.

The **Settings** tab of the data source is displayed.

It is now necessary to configure this data source to the InfluxDB database.

It will be through this connection that Grafana will be able to query, and render the crowding information stored in the InfluxDB database.

v) Set the data source configuration as follows:

a. **Name:** (as desired by you) (**Suggestion:** InfluxDB Data Source)

b. **Query Language:** Flux

c. **HTTP:**

- URL: http://<server_ip_address>:8086
- Allowed cookies: (Leave blank)
- Timeout: (Leave blank)

d. **Auth:** Basic auth

e. **Basic Auth Details:**

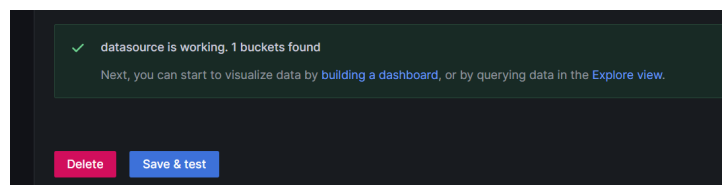
- User: crowdingSTToolkit
- Password: <InfluxDB_user_password>

f. **InfluxDB Details:**

- Organization: InfluxDBCrowding
- Token: (Insert the Authorization token created in the step 8) of the [InfluxDB database configuration](#))
- Default Bucket: DetectionsDB
- Min time interval: 10s
- Max series: 1000



vi) Click **Save & test**.

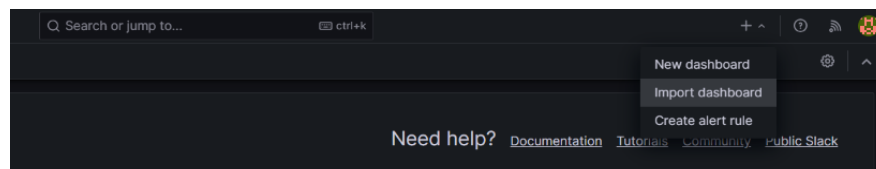
If the output “datasource is working. 1 bucket found” shows, this means that you have correctly configured the InfluxDB data source.



3) Import the STToolkit Crowding monitoring dashboard:

The last step regards the pre-configured Grafana dashboard of the STToolkit. This dashboard will allow you (and also the end users) to monitor crowding at the several locations where the sensors are deployed.

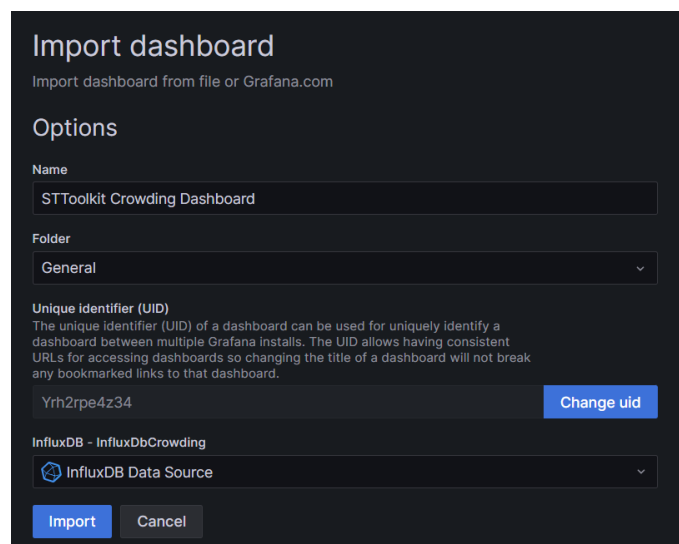
- i) First, download the STToolkit Crowding dashboard .json file to your personal computer from the [RESETTING GitHub repository](#).
- ii) Then, on Grafana, click on the Grafana  icon on the top left corner to go to **Home**.
- iii) On the top right corner, select the  > **Import dashboard**.



- iv) Click **Upload dashboard JSON file**.
- v) Select the .json file of the downloaded STToolkit Crowding dashboard.

The dashboard options are already set to this dashboard.

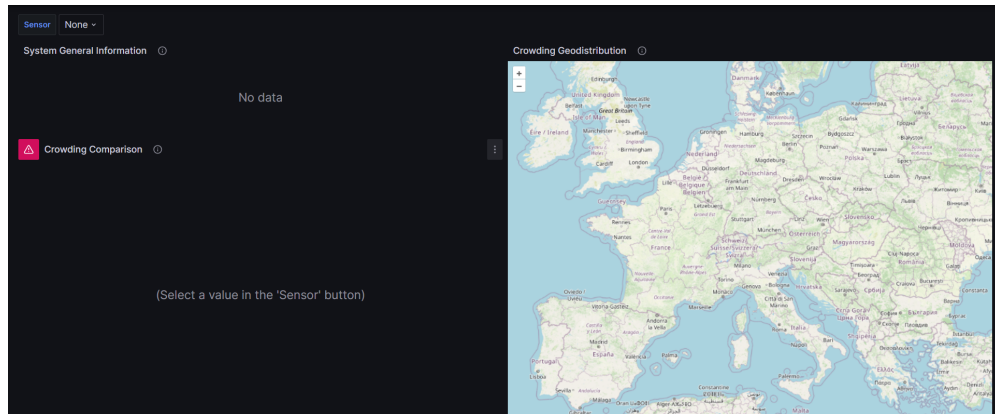
You only need to select the InfluxDB data source previously created.

A screenshot of the 'Import dashboard' form in Grafana. The form is titled 'Import dashboard' and has a subtitle 'Import dashboard from file or Grafana.com'. Under the 'Options' section, there are three input fields: 'Name' (containing 'STToolkit Crowding Dashboard'), 'Folder' (a dropdown menu set to 'General'), and 'Unique identifier (UID)' (containing 'Yrh2rpe4z34' and a 'Change uid' button). Below these fields, there is a section for 'InfluxDB - InfluxDbCrowding' with a dropdown menu set to 'InfluxDB Data Source'. At the bottom of the form, there are two buttons: 'Import' and 'Cancel'.



vi) Click **Import**.

The dashboard is now imported on Grafana and is shown in the image below.



The dashboard does not show any data because any sensor was not installed and configured yet. Once you install and configure each sensor, the data will automatically appear on this dashboard.

The explanation of how to use and edit this dashboard is further explained in the [Operation Manual](#).

You have now successfully installed and configured the STToolit's Cloud Server. You can now proceed with the installation and configuration of the STToolkit's Crowd Sensors described in the [Crowd Sensor Installation Manual](#).

References

For more information about the technologies used in the STToolkit's Cloud Server, check the documentation pages in the following links:

- Contabo documentation: <https://docs.contabo.com/>
- InfluxDB OSS documentation: <https://docs.influxdata.com/influxdb/v2/>
- Grafana documentation: <https://grafana.com/docs/grafana/latest/>

APPENDIX
B.

STToolKIT CROWD SENSOR INSTALLATION MANUAL



D3.5 - Toolkit for implementing tourism crowd detection solutions

Crowd Sensor Installation Manual

This **Crowd Sensor Installation Manual** includes guidelines and instructions for the integration/configuration of the STToolkit's crowd sensors. It is targeted for the **STToolkit Instantiator role**. This manual includes the step-by-step detailed tutorial for the installation and configuration of the STToolkit's crowd sensors, and the software that live on them.



This project has received funding from the COSME Programme (EISMEA) under grant agreement No.101038190

Contents

CROWD SENSOR INSTALLATION GUIDE	3
1. Sensor Equipment Acquisition	3
2. Flashing OS image on SD card	4
3. Sensor Configuration	6
3.1 <i>Preliminary configurations</i>	7
3.2 <i>Cloud Server connectivity configuration</i>	15
3.3 <i>Sensor location configuration</i>	16
3.4 <i>Tasks Automation configuration</i>	17
4. Sensor Mounting	20



Crowd Sensor Installation Guide

This section describes the installation and configuration of the STToolkit sensors. The sensors monitor the crowding levels at the target locations and upload the crowding information to the cloud server. This guide describes the step-by-step installation and configuration of the STToolkit sensors, after installing and configuring the cloud server.

These steps for the crowd sensor installation are also available as a video tutorial on the [RESETTING YouTube channel](#), which you can also use to install the crowd sensor.

1. Sensor Equipment Acquisition

First, it is necessary to acquire the equipment for mounting each sensor. The several hardware options for each sensor component are described on the Bill of Materials in the Deliverable. This guide presents the several options and the recommended option for each component. Make sure that you have at least one of this hardware options for mounting the sensor:

- **Single Computer Board (SBC):**
 - Raspberry Pi 4 4 GB RAM (recommended)
 - Raspberry Pi 3 1 GB RAM
- **Micro-SD card:**
 - 16 GB (minimum)
 - 32 GB (recommended)
 - > 64 GB (optimum)
- **Wi-Fi card:**
 - Alfa Network AWUS036ACH
 - Alfa Network AWUS036AC (recommended)
- **Raspberry Pi Power Supply:**
 - Raspberry Pi 4 USB-C Power Supply (recommended)
 - Raspberry Pi 3 Micro-USB Power Supply
- **USB Extension cable (Male to Female) with at least 10 cm length**
- **Sensor Case:**
 - Larger version with no exposed antennas (recommended)
 - Smaller version with exposed antennas

The sensor cases can be manufactured using 3D printing or a laser cutting machine. The files used to build them are available in the [RESETTING GitHub repository](#), along with some images of both case versions. These cases can be fabricated at a lab with a 3D printer or a laser-cutting machine. For instance, your sensor cases can be fabricated at [Vitruvius Fablab](#), an architecture lab at Iscte, which you can contact to manufacture these cases.

2. Flashing OS image on SD card

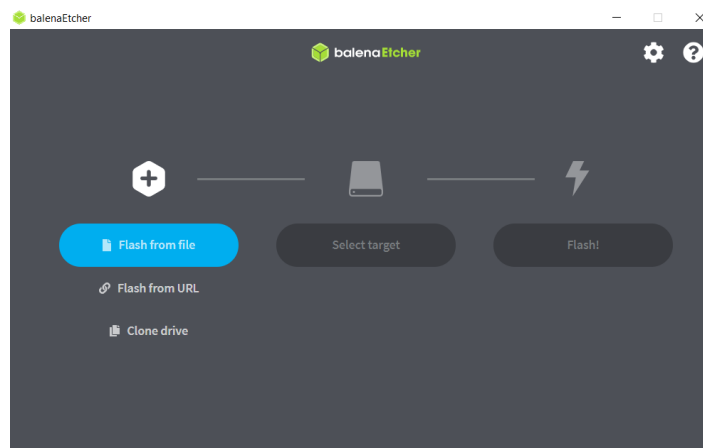
This section describes the steps for flashing the SD card with the software required for the sensor. Make sure that you have the following requisites.

Requirements:

- Windows computer with at least 15 GB free disk space and with SD card reader.
- Empty micro-SD card with at least 16 GB of space.

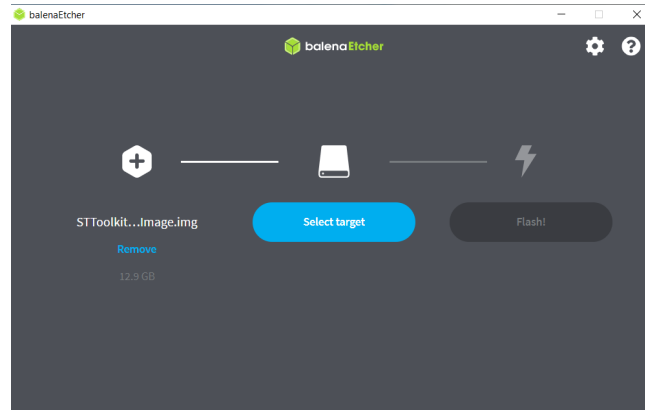
The steps for flashing the SD card are the following:

- 1) Download the “STToolkitCrowdSensorImage.img” compressed image file from the [FileSender FCCN platform](#). This file contains the setup image with all the installed software for the crowd sensor.
- 2) Extract the software image file.
- 3) Connect the empty micro-SD card to your personal computer.
- 4) Flash the SD card:
 - i) Download the [BalenaEtcher](#) software.
 - ii) Open the BalenaEtcher application on your computer.

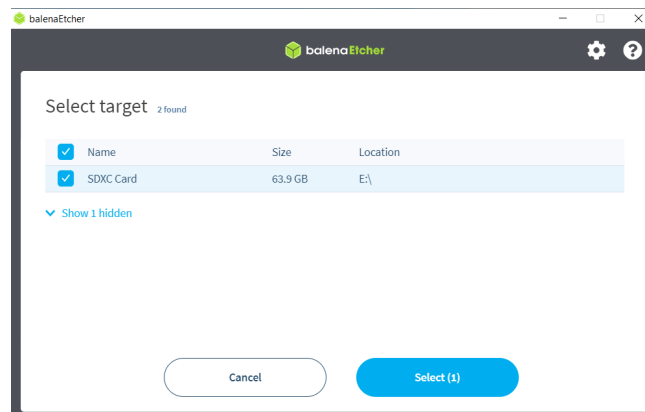




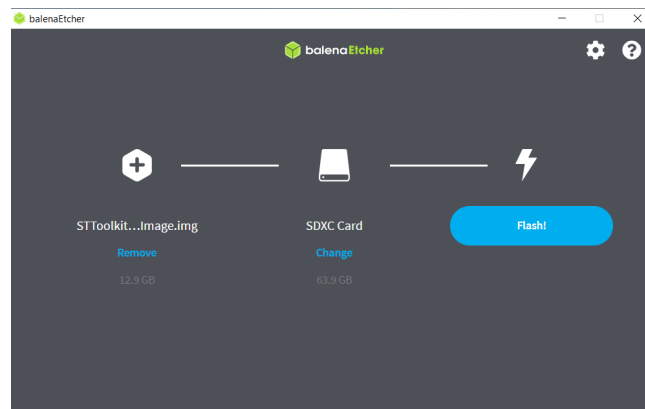
iii) Select the downloaded image file by clicking on “Flash from file”.



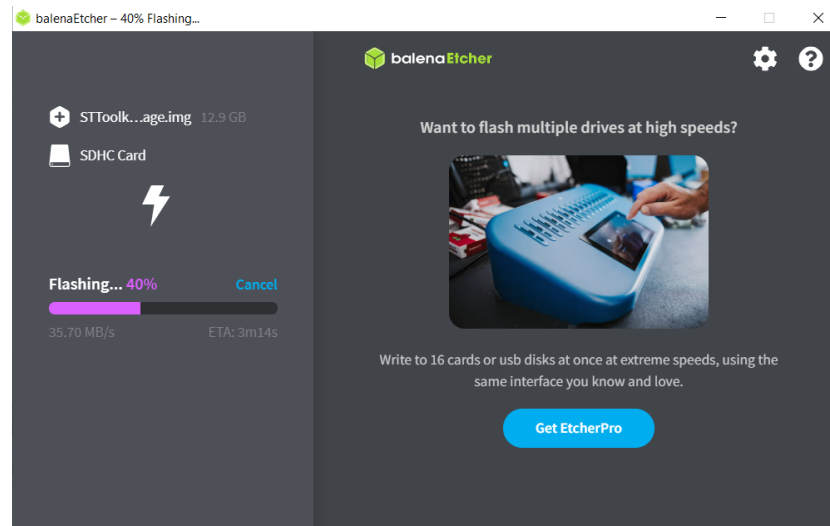
iv) Select the micro-SD card by clicking on “Select target”. (If not already selected by the application).



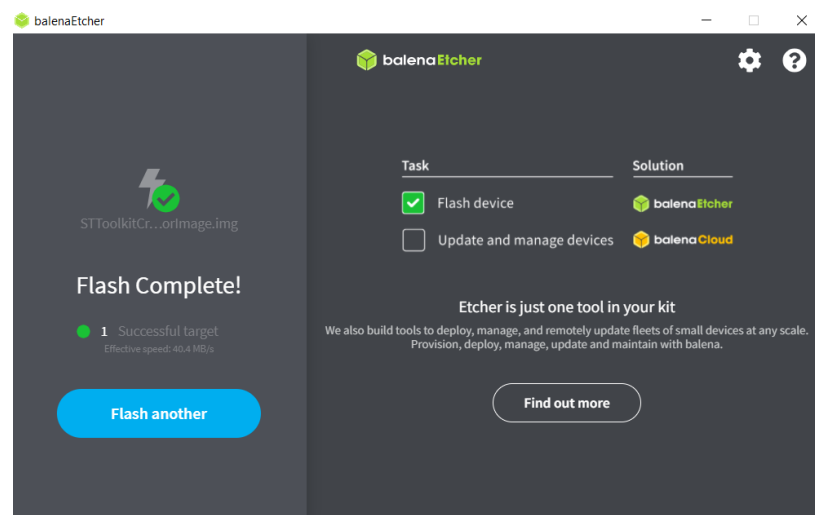
v) Flash the OS image by clicking on “Flash”.



- vi) The software will flash the OS image to the SD card. If the message “Insert a disk in SDXC” or similar appears just ignore and close the window. This process might take a few minutes so be patient and wait until it finishes.



- vii) If “Flash complete” appears on the application, the OS image was successfully flashed onto the SD card.



The OS image is now flashed on the SD card and the sensor is now ready to be configured.



3. Sensor Configuration

After flashing the OS image on the SD card, the next steps regard the configuration of the sensor.

3.1 Preliminary configurations

These steps require graphical interaction and so it is required to connect the Raspberry Pi to a monitor.

Requirements:

- Raspberry Pi;
- Raspberry Pi Power Supply;
- Micro SD-card (with the flashed OS image);
- Monitor, wired keyboard, and wired mouse;
- Micro-HDMI adapter to connect to the Raspberry Pi;
- Wi-Fi card for Wi-Fi detection of devices (see [Sensor Equipment Acquisition](#)).

The steps are the following:

- 1) Insert the SD card with the flashed OS image on the Raspberry Pi.
- 2) Connect the Wi-Fi card to the Raspberry Pi through the USB port.

Note: Connect the Wi-Fi card on the 2.0 USB ports (the ones not blue-coloured).

- 3) Connect the wired keyboard and wired mouse to two USB ports of the Raspberry Pi.
- 4) Connect the Raspberry Pi to the monitor.
- 5) Power the Raspberry Pi.

After a few seconds, a loading screen should appear asking for username and password credentials.

- 6) Insert the default user credentials:
 - **username:** kali
 - **password:** kali

A desktop should appear. You can now do the preliminary configurations of the sensor.

A) Change user password (optional)

This step may be important for security reasons, as the default password for accessing the sensor is very weak.

- 1) Open a new terminal window by pressing 'CTR + ALT + T'.
- 2) Enter the following command to change the kali user password.

```
$ sudo passwd
```

- 3) Enter the original password 'kali' and replace it with the new password.

This will be the new password required for accessing the sensor.

- 4) Close the terminal window.

B) Change local time zone (optional)

For convenience, you may want to set the sensor date and time according to the local time zone. For this, you need to find out the long name of the time zone you want to use. The time zone naming convention usually uses a "Region/City" format.

- 1) Open a new terminal window by pressing 'CTR + ALT + T'.
- 2) To view all available time zones, use the `timedatectl` command:

```
$ timedatectl list-timezones
```

- 3) Once you identify which time zone is accurate to your location, run the following command:

```
$ sudo timedatectl set-timezone <your_time_zone>
```

- 4) For example, to change the system's timezone to `America/New_York` you would type:

```
$ sudo timedatectl set-timezone America/New_York
```

- 5) To verify the change, invoke the `timedatectl` command:

```
$ timedatectl
```

You have now successfully changed the system's time zone to your location.

- 6) Close the terminal window.



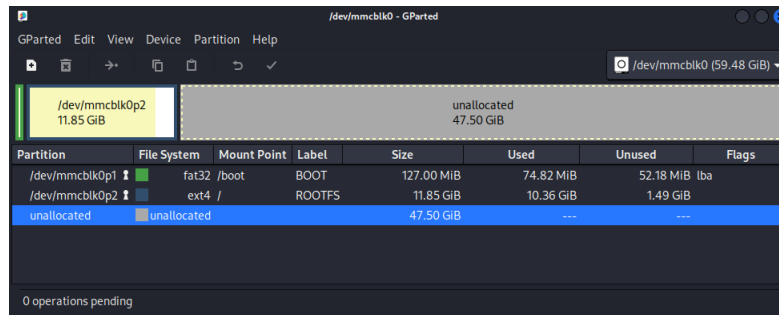
C) Extending available disk space

For extending the available disk space to the total space of the SD card do the following steps:

- 1) Open a new terminal window by pressing 'CTR + ALT + T'.
- 2) Open GParted application using sudo privileges.

```
$ sudo gparted
```

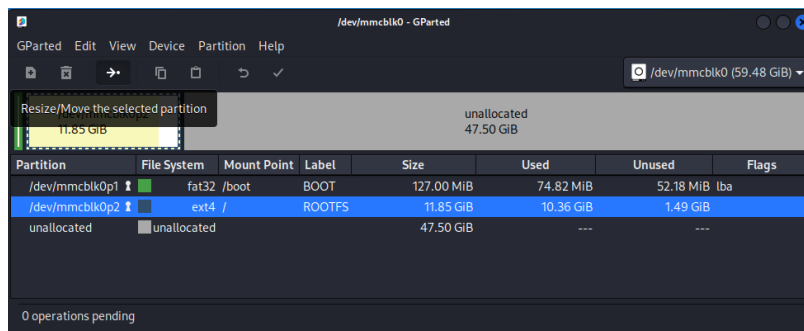
- 3) When the GParted application opens, it should appear a window similar to the following:



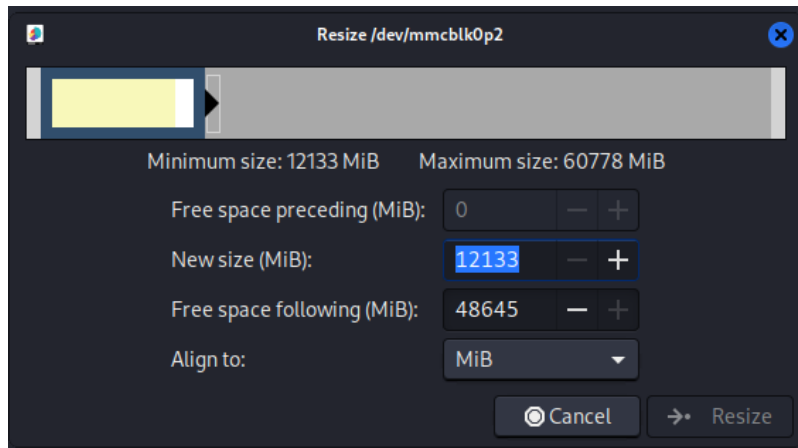
Here notice a few things:

- There are two partitions: BOOT and ROOTFS.
- There is a huge part of the disk with unallocated space. This is the remaining part of the SD card that is necessary to extend to the disk.

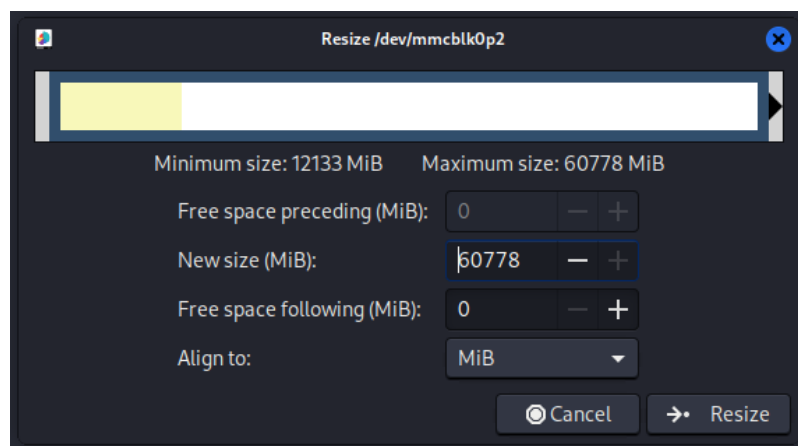
- 4) For extending the disk space to the total available space of the SD card, select the ROOTFS partition. Then click on 'Resize/Move the selected partition'.



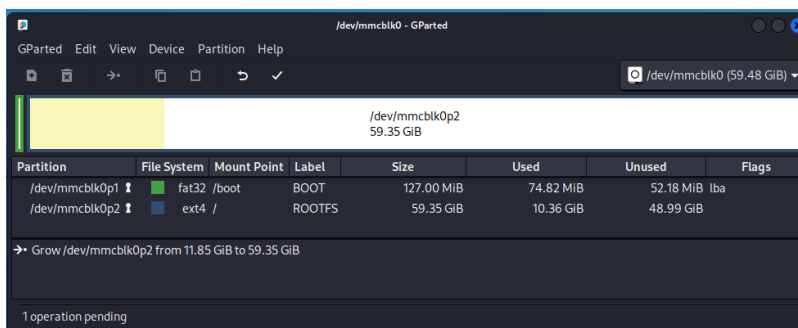
5) A window similar to the following should pop up:



6) Drag the right bar to the right as much as possible.

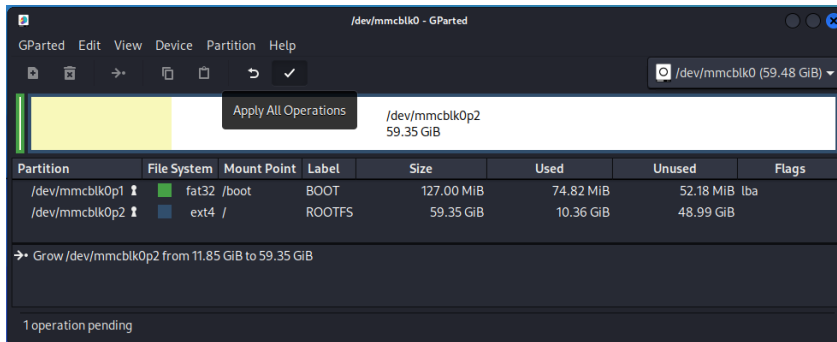


7) Press the 'Resize' button. You will return to the GParted window. This time it will look similar to the following.

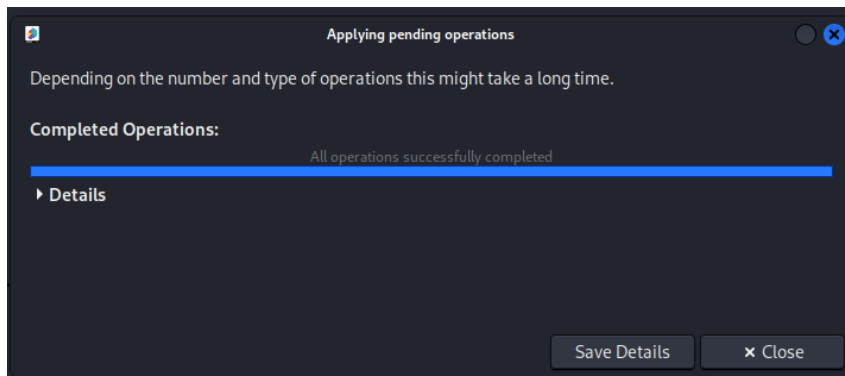




- 8) Finally, commit the changes by clicking on 'Apply All Operations'. Confirm the operation by clicking on 'Apply' in the pop up window. It will perform the extension so it can take a minute or two, but most of the time it finishes quickly.



- 9) If the operation succeeds, it should produce a similar output to the following. Afterwards you can close GParted.



Now the OS image is using the full space of the SD card.

D) Connect to a Wi-Fi network graphically

A Wi-Fi connection is necessary for the operation of the sensor. It is required for the data upload, and also for the remote access to the sensor, allowing to perform over-the-air updates as well as changing configuration parameters as desired by the STToolkit operator.


The configuration of a Wi-Fi connection can be performed graphically in just a few steps.

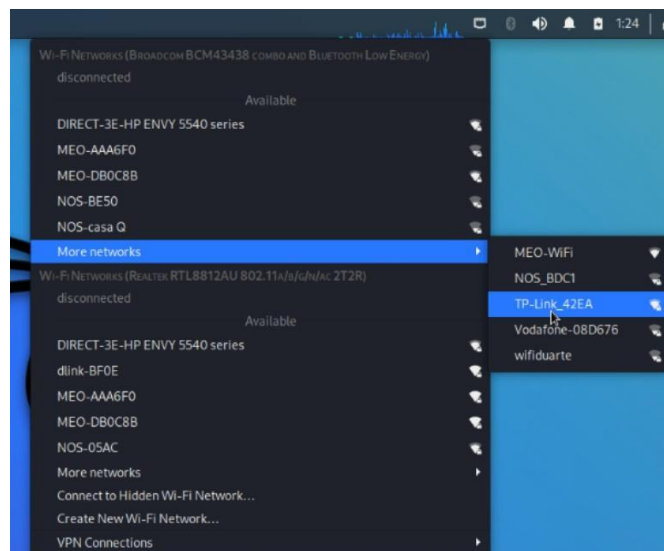
IMPORTANT NOTES:

- The configuration of the Wi-Fi connection requires having the Wi-Fi card connected to the Raspberry Pi. **If that is not the case, connect the Wi-Fi card to the Raspberry Pi on the USB 2.0 ports and reboot the sensor.** This is very important because the wireless interfaces have an order that needs to be followed, otherwise the sensor will not run correctly.
- It is recommended to configure the Wi-Fi connection where the sensor will be deployed, i.e., where the Wi-Fi network coverage that the sensor will connect to is available. If that is not possible go to the [Configure Wi-Fi connection not on deployment location](#) section.

Configure Wi-Fi connection on deployment location

Having the Wi-Fi card connected to the Raspberry Pi and being on the network coverage provided by the Wi-Fi network that the sensor will connect to, the Wi-Fi connection can be configured.

- 1) On the Desktop, click on the network icon displayed as .
- 2) Left-click on the network icon. It will show the available Wi-Fi networks on each Wi-Fi card. On the networks available for the 'Broadcom BCM 43438' board, select the Wi-Fi network to connect to.



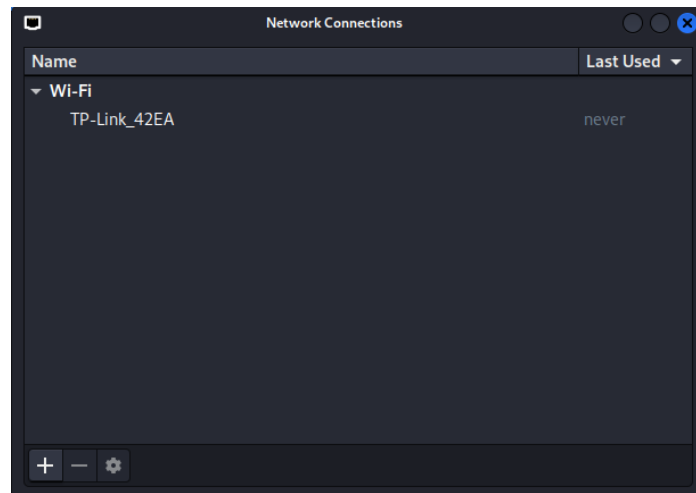


- 3) Select the correct Wi-Fi adapter and insert the password of the Wi-Fi network (if required).



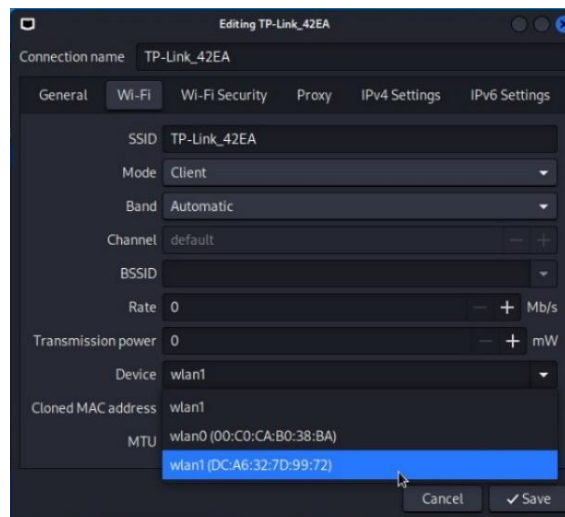
At this point you should have established the connection to the Wi-Fi network.

- 4) To strict the Wi-Fi connection to the Wi-Fi card of the Raspberry Pi, open the **'Advanced Network Configuration'** by searching it on the **Start menu**. You should see the network configuration that you have just created. (If you see other network connections, delete them by clicking on the "-" button.)



- 5) Select the created network connection and edit it by clicking on the wheel button.
- 6) Click on the 'Device', and select one of the following options according to the Raspberry Pi model of the sensor:
 - Raspberry Pi 4 wlan1 (DC:A6:32:XX:XX:XX)
 - Raspberry Pi 3 wlan0 (DC:A6:32:XX:XX:XX)

The image below shows an example using a Raspberry Pi 4, where you should select 'wlan1 (DC:A6:32:XX:XX:XX)'.



7) Afterwards, you can close the Advanced Network Configuration window.

Configure Wi-Fi connection not on deployment location

Having the Wi-Fi card connected to the Raspberry Pi and not being on the network coverage provided by the Wi-Fi network that the sensor will connect to, the Wi-Fi connection can also be configured.

- 8) Open the '**Advanced Network Configuration**' by searching it on the **Start menu**.
- 1) Create a new network connection by clicking on the "+" icon.
- 2) Select 'Hardware' > 'Wi-Fi', and then 'Create...'.
 - 3) On the 'Wi-Fi' section, insert the SSID of the Wi-Fi network, and on the 'Device' select one of the following options according to the Raspberry Pi model of the sensor:
 - Raspberry Pi 4 wlan1 (DC:A6:32:XX:XX:XX)
 - Raspberry Pi 3 wlan0 (DC:A6:32:XX:XX:XX)
 - 4) In the 'Wi-Fi Security' section, insert the Security required for the Wi-Fi network. The most usual are the 'WPA & WPA2 Personal' or 'WPA & WPA2 Enterprise' options.
 - 5) Afterwards, you can close the Advanced Network Configuration window.



E) Change the *dataRetentionManager.py* script

It is necessary to change the operation of the *dataRetentionManager.py* script. It is a minimal, but crucial change to the script correctly delete old and necessary data from the sensor local database.

- 1) Go to the Desktop.
- 2) Locate the *dataRetentionManager.py* python script. Right-click and then select "Open With 'Mousepad'".
- 3) A new window pops up with the script code, as shown below.

```

1 import sqlite3
2 import datetime as dt
3 import pytz
4 import sys
5
6 if (len(sys.argv) < 2) :
7     print("Argument not provided. Please specify an argument with the 'sliding window' time in minutes.")
8     exit()
9
10 if (len(sys.argv) > 2) :
11     print("Too many arguments required. Please specify only one argument with the 'sliding window time', in minutes.")
12     exit()
13
14 if not sys.argv[1].isdigit():
15     print("The sliding window is not an integer. Please set the 'sliding window' time, in minutes, as an integer.")
16     exit()
17 else:
18     dataActual = dt.datetime.now(pytz.utc).replace(tzinfo=None)
19     dataAnalyzer= dataActual - dt.timedelta(minutes=int(sys.argv[1]))
20
21     #Delete old and unnecessary data from the local database
22
23     connwifi = sqlite3.connect('/home/kali/Desktop/DB/DeviceRecords.db', timeout=30)
24     cwifi = connwifi.cursor()
25
26     cwifi.execute("""DELETE FROM Data_Packets WHERE ((First_Record > ? and First_Record <= ?) or (Last_Time_Found > ? and Last_Time_Found <= ?))""", (dataAnalyzer, dataActual,
27 dataAnalyzer, dataActual))
28     connwifi.commit()
29     cwifi.execute("""DELETE FROM Probe_Requests WHERE ((First_Record > ? and First_Record <= ?) or (Last_Time_Found > ? and Last_Time_Found <= ?))""", (dataAnalyzer,
30 dataActual, dataAnalyzer, dataActual))
31     connwifi.commit()
32     cwifi.close()

```

- 4) On the lines 26 and 28 of the code, add the word "NOT" after the word "WHERE", as shown in the image below.

```

21
22     #Delete old and unnecessary data from the local database
23
24     connwifi = sqlite3.connect('/home/kali/Desktop/DB/DeviceRecords.db', timeout=30)
25     cwifi = connwifi.cursor()
26     cwifi.execute("""DELETE FROM Data_Packets WHERE NOT ((First_Record > ? and First_Record <= ?) or (Last_Time_Found >
27 dataAnalyzer, dataActual))
28     connwifi.commit()
29     cwifi.execute("""DELETE FROM Probe_Requests WHERE NOT ((First_Record > ? and First_Record <= ?) or (Last_Time_Found
30 dataActual, dataAnalyzer, dataActual))
31     connwifi.commit()
32     cwifi.close()

```

- 5) Save the script on File > Save or by clicking 'CTR + S'.
- 6) Close the window.

The *dataRetentionManager.py* script is now with the correct configuration for its execution.

3.2 Cloud Server connectivity configuration

After doing all the preliminary configurations. It is now time to configure the cloud server connectivity. For this, we will make the use of the custom-made scripts developed for the sensor.

IMPORTANT NOTE:

- These steps assume that the Cloud Server has already been installed and configured, including the InfluxDB database. If not so, please go back to the [Cloud Server Installation](#) and then proceed to this section.
- We also recommend proceeding with these steps by remotely accessing the Raspberry Pi, and not using the graphical connection. You can use the [Advanced IP Scanner](#) tool to discover the IP address of the Raspberry Pi to remotely connect to.

1) Open a terminal window by clicking 'CTR + ALT + T'.

2) Go to the Desktop directory.

```
$ cd /home/kali/Desktop
```

The sensor needs an upload configuration to upload the crowding data to the Cloud Server. This configuration can be configured using the `influxDBUploadConfig.py` python script.

3) Run the `influxDBUploadConfig.py` python script.

```
$ python3 influxDBUploadConfig.py
```

The script will ask you to insert some parameters. These parameters are necessary to configure the data upload to the InfluxDB database already created when the Cloud Server was installed.

The Cloud Server IP address and the Authorization Token will differ for you. The InfluxDB Organization and Bucket names are the default names already created when installing the InfluxDB database on the Cloud Server.

- i) Insert the Cloud Server IP address.
- ii) Insert the default InfluxDB Organization Name '**InfluxDBCrowding**'. This is the default name of the default InfluxDB Organization already created in the Cloud Server.
- iii) Insert the default InfluxDB Bucket Name '**CrowdingDB**'. This is the default name of the default InfluxDB Bucket already created in the Cloud Server.
- iv) Insert the Authorization Token created during the installation of the Cloud Server.



- v) Finally, if the message 'Done! The new InfluxDB upload configuration was correctly saved.', you have successfully created the upload configuration to the Cloud Server.
- 4) Verify the current upload configuration by running the `checkInfluxDBConfig.py` python script.

```
$ python3 checkInfluxDBConfig.py
```

The output should be something similar to this:

```
(kali@ kali-raspberry-pi) ~/Desktop
└─$ python3 checkInfluxDBConfig.py
The active InfluxDB Upload Configuration is:

Cloud Server IP Address: 192.120.31.23
InfluxDB Organization Name: InfluxDBCrowding
InfluxDB Bucket Name: DetectionsDB
InfluxDB Authorization Token: qxVvbfKPAJs0X0e_9jYo_gyF9Z4XGNsdUQVICJjQqZcteDniPpqGsu4Uf3L60HPd6YTGPSzqG0FAUt9eQctQ==
```

3.3 Sensor location configuration

At this stage it is required to set the sensor location where it will be deployed. For this, use the `sendSensorLocation.py` python script.

The sensor location can always be configured and altered by running this script. For instance, if you want to change the sensor's current location to another location, simply run this script again, making sure that you specify the correct sensor name.

- 1) Open a terminal window by clicking 'CTR + ALT + T'.
- 2) Go to the Desktop directory.

```
$ cd /home/kali/Desktop
```

- 3) Run the `sendSensorLocation.py` python script.

```
$ python3 sendSensorLocation.py
```

- i) Insert the sensor name. We recommend that the name should be related to the deployment location of the sensor. E.g.: 'Main Entrance', 'Public Square'.

Note: Do not use spaces on the sensor name. Use the '_' character instead of spaces. E.g.: 'Main_Entrance', 'Public_Square'.

- ii) Enter the latitude and longitude of the current deployment location of the sensor. You can use Google [Maps](#) for this purpose. Search for the location, then right-click on the location, and copy and paste the latitude and longitude coordinates.

Note: The latitude and longitude are inserted individually. First the latitude and then the longitude.

- iii) Confirm the location to be set/altered for the sensor.
- iv) An output similar to the following should appear.

```
root@kali:~/Desktop# python3 sendSensorLocation.py
Welcome! This script allows to set/alter the sensor location where it is currently deployed.
IMPORTANT NOTE: If a sensor location was already sent with the same sensor name, the location
of that sensor will be altered to this new location.

Please enter the sensor name: Main\ Entrance
The sensor name to set/alter the location is: Main\ Entrance

Please enter the latitude of the sensor: 38.74872
Please enter the longitude of the sensor: -9.15364

The latitude and longitude (38.74872, -9.15364) will be set for the 'Main\ Entrance'. Are you sure? (Yes/No) Yes

HTTP/1.1 204 No Content
K-Influxdb-Build: OSS
K-Influxdb-Version: v2.3.0+SNAPSHOT.090f681737
Date: Sat, 16 Sep 2023 13:21:51 GMT

Location for the 'Main\ Entrance' sensor sent to the cloud server.
```

If the 'HTTP/1.1 204 No Content' message appears on the output, the sensor location was successfully sent to the InfluxDB, meaning that the sensor location was received in the Cloud Server.

If you want to change the location of the sensor to another location, simply run this script again, making sure that you specify the same sensor name.

3.4 Tasks Automation configuration

The final steps regard the automation of the tasks of the sensor. This is mandatory for the automatization of the sensor.

The automation of the sensor tasks can be configured in the crontab file. This file contains several commands that are executed on predetermined time schedules.

- 1) Edit the crontab file by entering the following command.

```
$ crontab -u kali -e
```

This command will open your crontab file in the system's default text editor.

The final part of the file contains the several tasks that the sensor needs to perform, as shown below. The function of each task is identified with a comment above it.



```
# Insert your tasks to run automatically above:
#
@reboot sudo service smbd restart
# Start monitor mode on Wi-Fi network card
#@reboot sudo airmon-ng start wlan0
# Wi-Fi detection of devices
*/10 * * * * timeout -k 1 590s sudo airodump-ng --background 1 wlan0
*/10 * * * * sleep 595 && sudo pkill airodump-ng
# Periodic upload of crowding data to the Cloud Server
*/5 * * * * /usr/bin/python3 /home/kali/Desktop/sendCrowdingData.py SensorTest Wifi 5 wlan1
# Periodic delete of outdated and unnecessary data from local database
0 * * * * /usr/bin/python3 /home/kali/Desktop/dataRetentionManager.py 30
# Weekly upload of OUI list
0 0 * * 0 /usr/bin/python3 /home/kali/Desktop/macOUIUpdater.py
# Daily reboot
0 4 * * * sudo reboot
```

2) Uncomment all tasks in the crontab file.

For uncommenting all tasks, press the letter 'I' to 'Insert'. This will enter the file in 'Insert' mode to edit it.

Then, go to the last part of the file, where the commented tasks are, and uncomment them by simply removing the '#' character at the beginning of each task.

```
# Insert your tasks to run automatically above:
#
@reboot sudo service smbd restart
# Start monitor mode on Wi-Fi network card
@reboot sudo airmon-ng start wlan0
# Wi-Fi detection of devices
*/10 * * * * timeout -k 1 590s sudo airodump-ng --background 1 wlan0
*/10 * * * * sleep 595 && sudo pkill airodump-ng
# Periodic upload of crowding data to the Cloud Server
*/5 * * * * /usr/bin/python3 /home/kali/Desktop/sendCrowdingData.py SensorTest Wifi 5 wlan1
# Periodic delete of outdated and unnecessary data from local database
0 * * * * /usr/bin/python3 /home/kali/Desktop/dataRetentionManager.py 30
# Weekly upload of OUI list
0 0 * * 0 /usr/bin/python3 /home/kali/Desktop/macOUIUpdater.py
# Daily reboot
0 4 * * * sudo reboot
-- INSERT --
```

3) Configure the tasks

Now let's walk through all the sensor tasks required for the sensor operation.

The parameters that you can change for each task are highlighted in blue.

i) Start monitor mode on Wi-Fi network card

Task: `@reboot sudo airmon-ng start wlan0`

Configuration:

- wlan0 Change the interface according to the Raspberry Pi model
 - Raspberry Pi 4 wlan0 (do not change)
 - Raspberry Pi 3 wlan1

ii) Wi-Fi detection of devices

Task: `*/10 * * * * timeout -k 1 590s sudo airodump-ng background 1 wlan0`

Configuration:

- wlan0 Change the interface according to the Raspberry Pi model
 - Raspberry Pi 4 wlan0 (do not change)
 - Raspberry Pi 3 wlan1

iii) Periodic upload of crowding data to the Cloud Server

Task: `* /5 * * * * /usr/bin/python3 /home/kali/Desktop/sendCrowdingData.py SensorTest Wifi 5 wlan1`

Configuration:

- SensorTest Specify to the name of the sensor (Default: SensorTest)
- wlan1 Change the interface according to the Raspberry Pi model
 - Raspberry Pi 4 wlan1 (do not change)
 - Raspberry Pi 3 wlan0

iv) Periodic delete of outdated and unnecessary data from local database

Task: `0 * * * * /usr/bin/python3 /home/kali/Desktop/dataRetentionManager.py 30`

Configuration: do not change anything.

v) Weekly upload of the OUI list

Task: `0 * * * * /usr/bin/python3 /home/kali/Desktop/macOUIupdater.py`

Configuration: do not change anything.

vi) Daily reboot

Task: `0 4 * * * sudo reboot`

Configuration: do not change anything.

- 4) Save the crontab file by clicking 'ESC', followed by ':wq'. Then press Enter to save the file.
- 5) Confirm the changes by entering the command.

```
$ crontab -u kali -l
```

The output should be the crontab file with the alterations applied. If not, go back to step 1) and repeat the process.

The sensor is now configured and ready to be deployed. The final step regards mounting the hardware inside the sensor case.



4. Sensor Mounting

These steps regard the sensor mounting at the deployment location, after configuring the sensor.

The sensor mounting is also available as a video tutorial on the [RESETTING YouTube channel](#), which you can also use to mount the sensor.

Make sure that you have the following equipment:

- Raspberry Pi 3 or Raspberry Pi 4;
- Raspberry Pi Power Supply (according to the Raspberry Pi model);
- Micro-SD card (with OS image flashed and configured from the previous steps);
- Alfa Network AWUS036AC or Alfa Network AWUS036ACH;
- USB extension cable (Male to Female);
- Sensor case;
- Screwdriver.

Follow the steps below to mount the sensor.

- 1) With the help of a screwdriver, open the sensor case;
- 2) Attach the Wi-Fi card on the bottom of the case;
- 3) Connect the USB extension cable to the Wi-Fi card;
- 4) Insert the SD card into the Raspberry Pi;
- 5) Attach the Raspberry Pi to the plastic sheet of the sensor case;
- 6) Connect the Raspberry Pi to the Wi-Fi card using the USB extension cable;
- 7) Connect the Power Supply to the Raspberry Pi;
- 8) With the help of a screwdriver, close the sensor case.

Done! The sensor is now ready to be deployed.

Power the sensor by plugging it on an electrical socket.

APPENDIX
C.

STToolKIT OPERATION MANUAL



D3.5 - Toolkit for implementing tourism crowd detection solutions

STToolkit Operation Manual

This **STToolkit Operation Manual** includes guidelines to operate and troubleshoot each STToolkit. It is targeted for the **STToolkit Operator role**. It includes guidance of how to use the several applications installed on the Cloud Server, namely the InfluxDB database and the Grafana data visualisation platform, and how to troubleshoot some circumstances that can occur during the STToolkit runtime.



This project has received funding from the COSME Programme (EISMEA) under grant agreement No.101038190

Contents

GRAFANA	3
Log in into Grafana	3
Admin user	3
Create a new user	3
<i>Create a new visualization user for the STToolkit's end users</i>	4
Crowding data visualization	4
<i>How to use the crowding dashboard</i>	5
<i>System General Information panel</i>	6
<i>Crowding Comparison panel</i>	6
<i>Crowding Geodistribution</i>	6
INFLUXDB	7
Log in into InfluxDB	7
Manage InfluxDB database	7
Sensor measurements visualization	7
CROWD SENSORS	8
Remotely access the crowd sensor	9
Change the sensor location	9
Change the InfluxDB upload configuration	9
Test the Cloud Server connectivity	10
TROUBLESHOOTING	11



Operation Manual

This manual includes guidelines to operate and troubleshoot each STToolkit after its installation. It assumes that the Cloud Server and also the sensors are already installed and configured for each STToolkit.

Grafana

The Grafana is the data visualization platform of the STToolkit. It is on this platform that you can visualize the crowding data collected by the several sensors deployed at several locations.

Log in into Grafana

You can access Grafana by inserting **<your_server_ip_address:3000>** on your browser.

The default user credentials are **admin** for both username and password. However, we expect that you have already changed the admin password to another stronger password, as described in the installation manual.

The Grafana also saves the log in sessions, for users do not have to introduce your user credentials every time they access Grafana.

Admin user

The STToolkit has access to the admin user. This user has permissions to do everything on Grafana, such as creating new users, changing their passwords, creating new organizations, creating new dashboards and editing them, etc.

The admin user has access to the **Administration** section on Grafana. This section includes information for Grafana administrators, team administrators, and users performing administrative tasks.

For more information about this section, check this [link](#).

Create a new user

You can create a new user on **Administration > Users > New user**.

Insert the required fields and then you have a new user created.

Each user has roles and permissions associated. By default, the new users have a default 'Viewer' role, which means that users can only visualize the information on Grafana, and do not have editing permissions.

For more information about roles and permissions, check this [link](#).

Create a new visualization user for the STToolkit's end users

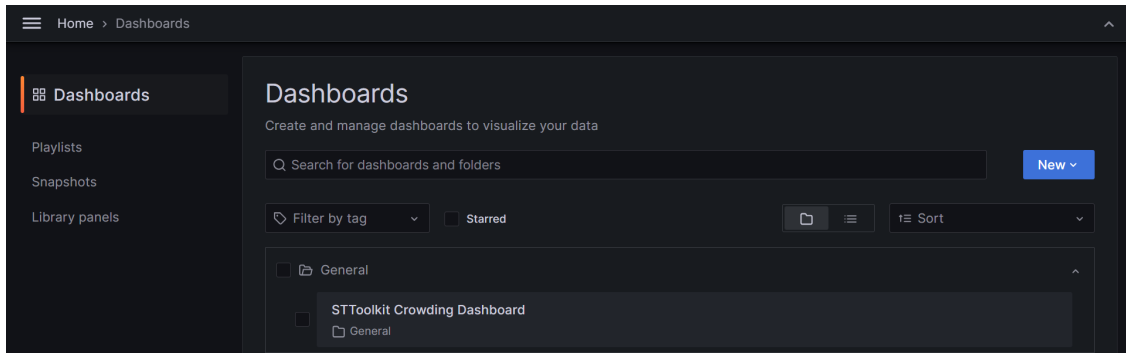
We recommend creating a new visualization user for the STToolkit, with visualization permissions only (*Viewer* role). As so, this new user will only be capable of visualizing the Grafana dashboards.

This new visualization user can then be provided to STToolkit's end users, e.g. tourists, to visualize the crowding information by providing its credentials.

Crowding data visualization

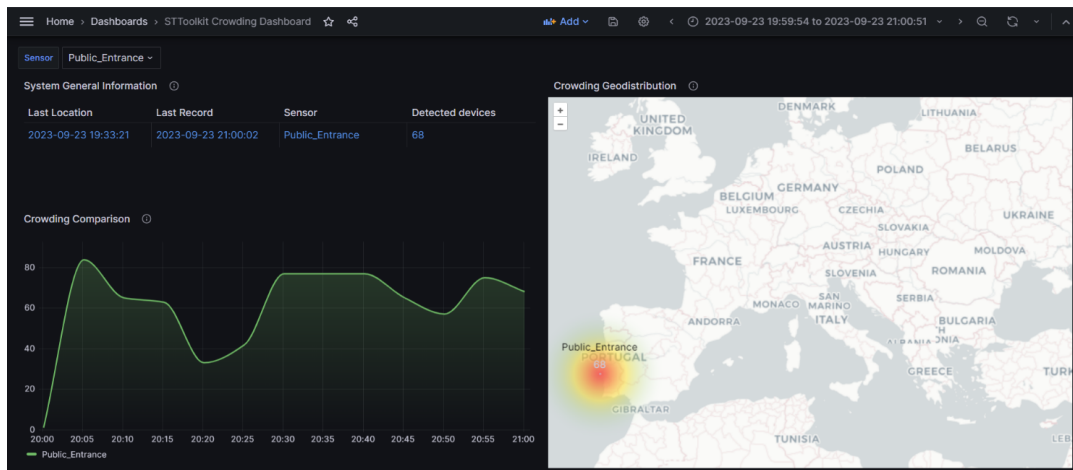
If you have correctly installed and configured Grafana on the dashboard, you have already installed the pre-configured STToolkit Crowding dashboard on Grafana to visualize the crowding data.

You can access this dashboard by clicking **Dashboards** in the toggle menu on the left side, and then on **STToolkit Crowding Dashboard**.



If you have installed and configured the crowd sensors correctly, the crowding measurements from sensors should automatically be rendered on the dashboard.

An example of how data is rendered on this dashboard is shown in the image below.





How to use the crowding dashboard

The dashboard is composed by three main panels:

- **System General Information** – This panel is a table that shows the general information of the STToolkit. It queries information within the last hour. This table has 4 columns:
 - **Last Location** - The last sensor location measurement sent by each crowd sensor;
 - **Last Record** – The last crowding measurement sent by each sensor;
 - **Sensor** - The name of the sensor;
 - **Detected devices** – The number of devices detected in the last crowding measurement sent by each sensor.
- **Crowding Comparison** – This panel is a graph that allows a temporal rendering of the crowding information. This graph shows the last crowding measurements from the sensors selected in the 'Sensors' button on the top left corner of the dashboard. It queries the crowding information within the selected time range selected on the top right corner. The time range can be selected to any start and end time as you wish. This graph allows you to visualize crowding tendencies along the passage of each day, compare crowding from sensors, and also to perceive highly-populated events.
- **Crowding Geodistribution** – This panel is a map that allows you to visualize the crowding geodistribution. It queries information within the last hour. With this map you can observe how the crowding is spatially distributed at the several locations where the sensors are currently deployed. It renders and matches both sensor locations and crowding measurements. Each heatmap circle represents the location where the sensors are deployed, and are accompanied by the number of detected devices of the last crowding measurement sent by each sensor.

This map allows a quick and easy interpretation of the crowding phenomena at the several target locations where the sensors are deployed, allowing to perceive crowding tendencies throughout the days with the temporal graph, and also to observe how the crowding is spatially distributed across the several locations.

As the Operator, you can edit this dashboard, add new panels, change their settings and configurations as you wish, although the dashboard already allows a good and quick interpretation of crowding with the default panels.

For more information about Grafana dashboards, check this [link](#).

For more information about Grafana panels and visualization, check this [link](#).

System General Information panel

There are no meaningful editions to perform on this panel.

You can order the table by a column by selecting it.

Crowding Comparison panel

This panel shows the crowding levels on the time range selected on the top right corner of the dashboard.

Choose the sensors which you want to visualize the crowding levels on the 'Sensor' button on the top left corner.

Note: If this panel shows an error when the dashboard is first loaded, just **unselect and select again a sensor on the 'Sensor' button.**

You can also change the name that is displayed for each sensor. To do this, you need to add an Override to the name of each sensor. To add an Override, click on the three dots icon and then on 'Edit' to edit the graph. In the right-hand menu, click on 'Overrides'. Click 'Add field override'. Select 'Fields with name'. Choose the sensor name you wish to change. Click 'Add override property', and select 'Standard options > Display name'. Enter the name of the sensor to be shown. Once you add the Override, the name of the sensor shown in the graph has changed to the name you chose in Override. You can change the name that is displayed for each sensor by adding a Override just like this to each sensor.

Crowding Geodistribution

Once you import the pre-configured dashboard, the 'Crowding Geodistribution' map comes with a very zoomed out initial view by default. The initial view configures how the map renders when the panel is first loaded. You can (and should) set the initial view of the map to an area closer to your sensors. For this, you need to edit this map.

To edit the 'Crowding Geodistribution' map, click on the three dots icon on the top right corner of the map, and select **Edit**.

Use your mouse to drag and the mouse wheel to zoom in and out until you find the correct initial view for the map. Then, click on 'Use current map settings' to set the initial view. Then, click on 'Apply' to apply the changes. Click on 'Save' on the top right corner to save the dashboard. Reload the dashboard. The initial view when the dashboard is loaded has now changed to the view that you specified.

For more information about this map, check the [Grafana geomap documentation](#).

You can also change the that is displayed for each sensor by adding an Override to each sensor name, as shown for the previous panel.



InfluxDB

The InfluxDB is the main database of the STToolkit and is responsible for the data ingestion and storing the collected data from all sensors.

Log in into InfluxDB

You can access InfluxDB by inserting `<your_server_ip_address:8086>` on your browser.

The user credentials are the ones created when the initial setup configuration of the InfluxDB database was created.

Manage InfluxDB database

You can manage organizations, buckets, and the authorization tokens using the InfluxDB UI.

To manage users, you need to use the InfluxDB CLI.

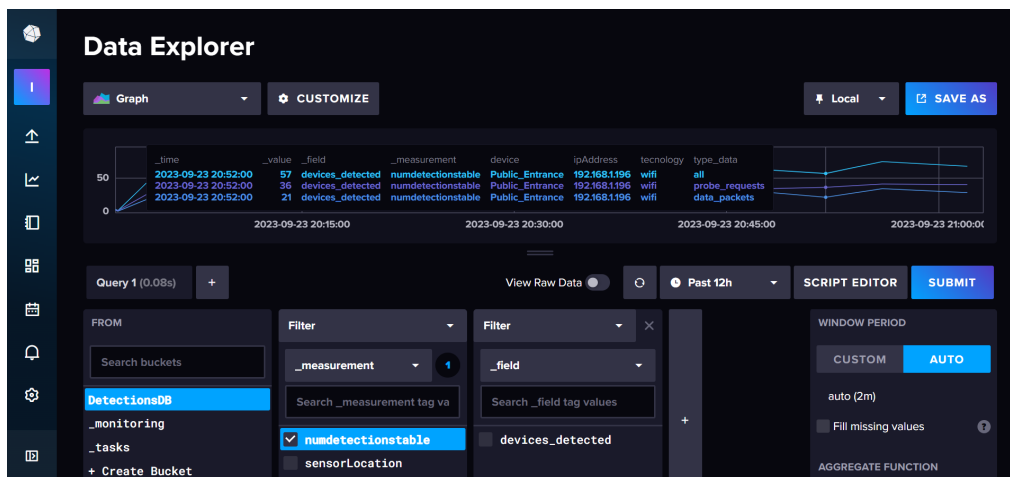
- You can add users by following this [link](#).
- You can change the user's password by following this [link](#).

For more information about InfluxDB, check the [InfluxDB documentation](#).

Sensor measurements visualization

In InfluxDB, you can also visualize the measurements sent by the sensors on the **Data Explorer** section. The **Data Explorer** allows you to build, execute, and visualize queries.

The image below shows how you can visualize the crowding level measurements sent to the *DetectionsDB* bucket.



Note that there is also an `ipAddress` field that is also sent in the crowding level measurements. This field is also sent to the Operator know which is the IP address that the sensor is using to upload the crowding data to the InfluxDB. Furthermore, this IP address is also sent so that the Operator can also know where to remotely access the sensor via SSH.

Therefore, the Operator should go into the InfluxDB Data Explorer section to know how to remotely access each sensor via SSH, as the IP address of the sensor is required.

Crowd Sensors

This section provides guidance on how to operate the STToolkit's crowd sensors.

The relevant files for the crowd sensor operation are all in the `/Desktop` directory. **Please do not change the location or the name of any file in this directory, as it can compromise its operation.**

In the Desktop of the sensor we have:

- `aircrack-ng-1.7` – This folder contains the aircrack-ng software suited with for the detection of Wi-Fi devices;
- `DB` – This directory contains the two sensor local databases:
 - `DeviceRecords.db` – This database contains the device's trace elements captured by the sensor;
 - `InfluxDBConfiguration.db` – This database contains the InfluxDB upload configuration with the required parameters for uploading the crowding data to the InfluxDB database.
- `macOUIUpdater.py` – A python script for updating the OUI list used to identify the manufacturer of the captured Wi-Fi frames.
- `dataRetentionManager.py` – A python script that deletes old and unnecessary data form the `DeviceRecords.db`, and only retaining useful information that is needed for device counting;
- `sendSensorLocation.py` – A python script to send the current location where the sensor is deployed;
- `sendCrowdingData.py` – A python script to send the number of devices detected to the InfluxDB. In fact, this script is responsible for the periodic upload of the crowding levels in the sensor's vicinity to the InfluxDB.
- `influxDBUploadConfig.py` – A python script to configure the InfluxDB upload configuration, with the required parameters to the sensor be able to send measurements to the InfluxDB;
- `checkInfluxDBConfig.py` – A python script to check the current InfluxDB upload configuration.



If the crowd sensor was correctly installed and configured, the operator does not have to perform anything, as the sensor will automatically operate on its own without any problem.

The operator should be able to change the configuration of the crowd sensor with ease if desired. The following sections show how to change the sensor configuration..

Remotely access the crowd sensor

To remotely access the crowd sensor, check the assigned IP address for the sensor on the InfluxDB Data Explorer, as shown in [this section](#).

Then, open a terminal window and type:

```
$ ssh -l kali <sensor_ip_address>
```

Enter the password for the kali user. The default password is also kali, but you could already have changed it during the sensor installation.

Change the sensor location

To change the sensor location, run the *sendSensorLocation.py* script. Make sure that you insert the same sensor name to change its location.

Change the InfluxDB upload configuration

To change the InfluxDB upload configuration, run the *influxDBUploadConfiguration.py* script. Insert the new parameters, namely, the Cloud Server IP address, InfluxDB organization, InfluxDB bucket, and the Authorization token.

Test the Cloud Server connectivity

To test the Cloud Server connectivity, run the *sendCrowdingData.py* script. If a 'HTTP 204' output appears, you have successfully sent a measurement to the InfluxDB database of the Cloud Server.

Troubleshooting

This section describes possible errors that can occur and how the operator should act upon them.

- **The 'Crowding Comparison' panel shows an 'undefined identifier' error**
Unselect and then select the sensor in the 'Sensor' button.
- **There are no available sensors to select on the 'Sensor' button'**
There are no crowding measurements sent by any sensor within the selected time range. Check if the time range is correctly selected to show the data on the panel.
- **Data shows on the 'Crowding Comparison' panel but not in the 'System General Information' and in the 'Crowding Geodistribution' panels**

This can be due to two reasons:

1. All sensors did not send any measurement in the last hour. You can also verify this using the 'Crowding Comparison' map. It is possible that the sensors lost connectivity with the Cloud Server. Try to reboot them.
 2. Check if the sensor location (*sendSensorLocation.py*) and the crowding level measurements (*sendCrowdingData.py*) have the same sensor name for the sensors. If not, you need to remotely access them, and make sure that both measurements use the same sensor name.
- **Data shows on the 'System General Information' and on the 'Crowding geodistribution' but not in the 'Crowding Comparison' panel**
There are no crowding measurements sent by any sensor within the selected time range. Check if the time range is correctly selected to show the data on the panel.
 - **I have set the wrong name for the sensor**
You have two options:
 1. Override the sensor name in the dashboard panels. (most straightforward solution)
 2. Remotely access the sensor and send a new sensor location measurement with the correct name of the sensor, and set the same new name to the crowding level measurements (more extensive solution, just as if a new sensor was installed)
 - **I have set the wrong location for the sensor**
Remotely access the sensor, and run the *sendSensorLocation.py* script again. Insert the sensor name you want to change the location and set it to the correct location. You can run this script and change the sensor location the times you want.
 - **I have set the wrong InfluxDB upload configuration**
Remotely access the sensor and run the *influxDBUploadConfig.py* script again. Make sure that you insert the correct server IP address, InfluxDB organization, InfluxDB bucket, and Authorization token.



- **The dashboard showed but now does not show any data on the three panels**

This is very unlikely to happen. This means that any sensor sent any measurement at least within the last hour. Probably all sensors have lost connectivity to the Cloud Server. You can check when the last crowding measurement sent by each device was using the 'Crowding Comparison' panel and selecting different time ranges. Try to reboot the sensors, or unplug and plug it back to the electrical socket for a 'manual reboot'.
- **The dashboard never showed any data on the three panels**

This is very unlikely to happen. Follow the steps below to troubleshoot this case:

 1. Check if the InfluxDB upload configuration of the sensors is correctly configured by running the `checkInfluxDBConfig.py` script. If not, run the `influxDBUploadConfig.py` script and reconfigure it.
 2. Test if the sensors can send information to the Cloud Server by manually running the `sendCrowdingData.py` script. If a 'HTTP 204' shows in the output, you have sent the measurement to the InfluxDB. If not, go to step 3.
 3. Remotely connect to the Cloud Server, and check if the 8086 port is allowed in the firewall by running the `sudo ufw status` command. If not, add this port by running `sudo ufw allow 8086`. The same applies to the 3000 port. Run the `sudo ufw status` command to confirm the changes.
- **I have deconfigured the dashboard and do not know how to put it back**

Delete the dashboard and import it again from the [GitHub repository](#).
- **The sensor is always sending measurements, but with '0' devices detected**

The InfluxDB upload configuration is correctly set for the sensor, but the sensor is not performing the device counting. Make sure that you have uncommented the tasks for enabling the monitor mode on the Wi-Fi board and for performing the Wi-Fi detection of devices. If so, please make sure that you have the Wi-Fi board attached to the Raspberry Pi.

APPENDIX
D.

STTOOLKIT USER MANUAL



D3.5 - Toolkit for implementing tourism crowd detection solutions

STToolkit User Manual

This **STToolkit Operation Manual** illustrates the experience of an STToolkit end user. It is targeted for the **STToolkit User role**. It includes guidelines for end users, e.g. tourists, on how to use STToolkit's crowding information.



This project has received funding from the COSME Programme (EISMEA) under grant agreement No.101038190

Contents

USER MANUAL	3
Accessing the crowding information	3
How to use the crowding dashboard	4
Sign out	4



User Manual

This manual includes the experience of an STToolkit end user.

The end user can visualize the crowding information collected at the several spots where the STToolkit's crowd sensors are currently deployed. With this information, users can visualise the crowding levels in real-time at several hotspots, thus allowing them to make more informed decisions, for instance, to know where they should go if one of the locations is overcrowded.

Accessing the crowding information

The end user can visualize the crowding information by accessing the "STToolkit Crowding Dashboard" on Grafana, the data visualization platform of the STToolkit. This dashboard contains several panels that allows users to visualize the crowding data collected by the crowd sensors, deployed at several hotspots.

Users can visualize this dashboard either in a computer or in a mobile phone.

To access this dashboard, follow the steps below:

- 1) Open a web browser and type the following URL: **<server_ip_address>:3000**

The <server_ip_address> needs to be provided by the STToolkit Operator. This is the IP address of the STToolkit's Cloud Server, where the data visualization application is installed.

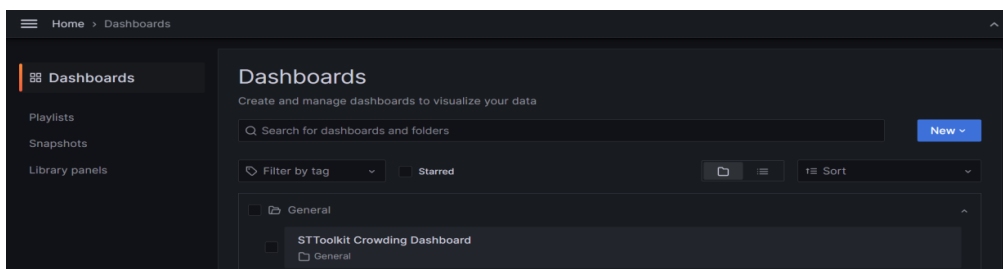
- 2) A log in screen appears. Insert the end user credentials provided by the STToolkit Operator.

Username: (provided by the STToolkit Operator)

Password: (provided by the STToolkit Operator)

You will be directed to the Grafana Main Menu.

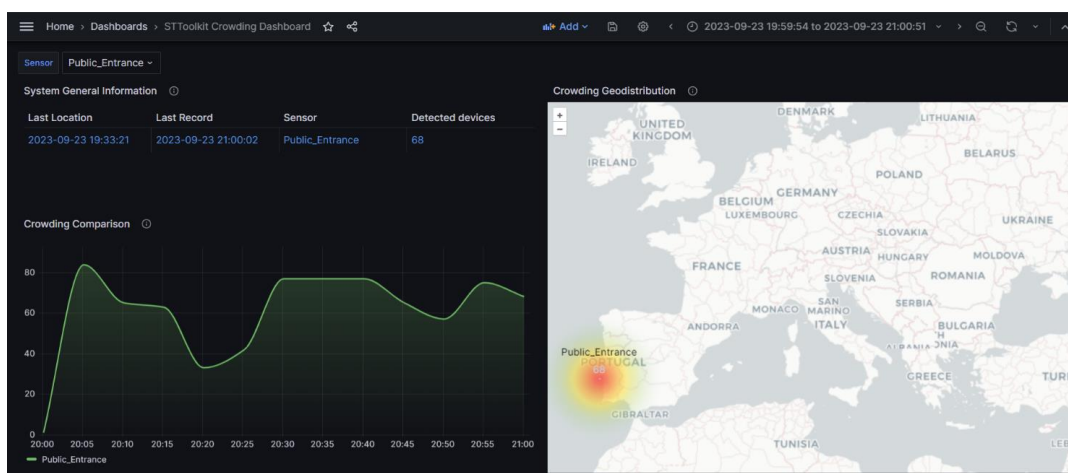
- 3) On the left-side menu, click **Dashboards** and then on **STToolkit Crowding Dashboard**.



You have entered on the "STToolkit Crowding Visualization" dashboard. You can then visualize the crowding data using this dashboard.

How to use the crowding dashboard

A snapshot of the crowding dashboard is shown below.



The 'System General Information' shows the general information of the system. It shows the last location measurement sent by each sensor, the last crowding level measurement sent by each sensor, the name of the sensor, and also the number of detected devices in the last measurement for each sensor.

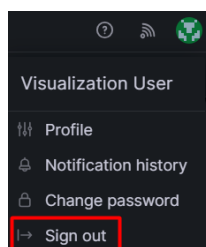
The 'Crowding Comparison' graph shows the last crowding measurements sent by the sensors selected in the 'Sensor' button on the top left corner of the dashboard. The time range for this graph is selected on the top right corner of the dashboard.

The 'Crowding Geodistribution' map shows the geographical location of each sensor. The locations where the sensor are deployed are shown with a heatmap circle, and are accompanied by the name of the sensor and also the number of detected devices in the last measurement sent by each sensor.

As the end user, you only have visualization permissions to this dashboard. Feel free to navigate through this dashboard and visualize the crowding data collected by the several sensors deployed at hotspot locations.

Sign out

You can sign out from Grafana by clicking on the user's icon, and then **Sign Out**.



[This page has been intentionally left blank]

**Smart Tourism Toolkit for
Crowd-monitoring Solutions**

Tomás Santos