Universitat Politècnica de València

Departamento de Informática de Sistemas y Computadores

Crowdsensing solutions for urban pollution monitoring using smartphones

By

Willian Jesús Zamora Mero

Advisors:

Dr. Carlos Tavares Calafate

Valencia, Spain

December, 2018

To my daughter Isabella Zamora.

The only way forward, if we are going to improve the quality of the environment, is to get everybody involved.

Richard Rogers

Acknowledgements

Setting goals and striving to meet them is something we should adopt as a good practice. However, these sacrifices represent a clear spending of time! Thus, I begin by addressing my acknowledgements to my family, my sisters and brothers, brothers-in-law, and sisters-in-law, who always supported me towards achieving this goal. Special thanks go to my brother-in-law, Harold Romero, and his wife, my sister Lina, who were always interested in the development of my thesis. To my dear parents, how not to thank them for their effort and love, as well as their advice to keep going and never give up.

To Rene and Anna, who knew how to deliver the best advice, and who welcomed me with open arms into their home, in this country that has now become my second home. I consider them like my brothers! Thank you for everything that you did, and for making me feel like part of your family.

To Carlos Tavares Calafate, who has been my doctoral thesis supervisor, and who knew how to get the best of me at all times. His advice and experience were a bulwark for the successes I have achieved. Thanks, Carlos, for the shared moments, and for the great friendship we have!

Also, to the leaders of the GRC, especially to Juan Carlos, whom I will always remember by the joy that characterizes him and his support, always encouraging me at late hours to keep going and not faint. To Pietro for his management efforts, helping all of us who belong to his research group, and for his excellent sense of humor. In the same way to Enrique, for his joy and the knowledge that he transmitted in the different stages I have been through, including the Master.

To all my colleagues at the GRC: Elsa, Oscar, Francisco, Leonardo, the Jorges, Subhadeep, Camilo, Seilendria, Pablo, Sergio, Miguel, Johann, Ali, Rashad and Chaker, with whom we formed a large group. They always extended their hand and their teachings. Thanks to you all!

Moreover, I would like to thank my friends and colleagues, who have been part of the first year of PhD training (Master), such as: José (Puche, Duro, and

Roche), Javier Prades, Victor, Danilo, and Vanessa, with whom I shared beautiful experiences

To Elsa Vera who was present at an important stage of this thesis, and to whom I keep my most in-depth consideration and thanks for the effort made and help.

To my friends, especially Jose A., Silvia D., Luzmila, Patricia, Alex S., Doris, Daniel Z., Dario, Mario, Yandry, ... for the persistent support that they transmitted to me at a general level.

A thank you to the institution to which I belong, the "Laica University Eloy Alfaro de Manabí", and also to the "SENESCYT Scholarship Program of the Republic of Ecuador", for the economic support received during this program.

Finally, I want to thank the Cantos Cantos family, especially Johana, who at all times took care and supported my daughter Isabella to understand my absence. Thanks for everything Joy.

Never stop dreaming!

Abstract

Environmental pollution is one of the main problems that affect our planet. Industrial growth and urban agglomerations, among others, are contributing to the diversification and chronification of this problem. The presence of environmental pollutants at high levels affect human health, with air quality and noise levels being examples of factors that can cause negative effects on people both psychologically and physiologically.

Traditionally, environmental pollution is measured through monitoring centers, which are usually fixed and have a high cost. However, the ubiquity of microcomputers and the increase in the number of sensors embedded in our smartphones, have paved the way for the appearance of new strategies to measure such pollution. Thus, Mobile Crowdsensing has become a new paradigm through which smartphones emerge as an enabling technology, and whose widespread adoption provides enormous potential for growth, allowing large-scale operations, and with costs acceptable to our society. Through crowdsensing, smartphones can become flexible and multipurpose detection units that, through the sensors integrated into these devices, or combined with new sensors, allow monitoring regions of interest with good spatial and temporal granularity.

In this thesis, we focus on the design of crowdsensing solutions using smartphones. We deal with environmental pollution problems, specifically noise and air pollution. With this objective, the crowdsensing proposals that have emerged in recent years are studied in the first place. The results of our study show that there is still a lot of heterogeneity in terms of technologies used and implementation methods, although modular designs at both client and server seem to be dominant.

Concerning air pollution, we propose an architecture that allows measuring air pollution, specifically ozone, in urban environments. Our proposal uses smartphones as the center of the architecture, being these devices responsible for reading the data obtained by an external mobile sensor, and then sending such data to a central server for processing and analysis. In this proposal, several problems have been analyzed with regard to the orientation of the external sensor and the sampling time, and the proposed solution has been validated in real scenarios. The results obtained show that the orientation of the sensor and the sampling period, within certain limits, have very little influence on the captured data. Also, by comparing the heat maps generated by our solution with the data from the existing monitoring stations in the city of Valencia, we demonstrate that our approach is capable of providing greater data granularity.

Concerning noise pollution, we propose an architecture to measure noise levels in urban environments based on crowdsensing, and whose main characteristic is that it does not require user intervention. In this thesis, we detail aspects such as the calibration of smartphones, the quality of the measurements obtained, the sampling instant, the server design, and the client-server interaction. Besides, we have validated our solution in real scenarios to demonstrate the potential of the proposed solution. Experimental results show that, with our proposal, it is possible to measure noise levels in different urban or rural areas with a degree of precision comparable to that of professional devices, all without requiring the intervention of the user, and with reduced consumption of system resources.

In general, the different contributions of this doctoral thesis provide a starting point for new developments, offering efficient calibration strategies and algorithms to make representative measurements. Besides, a significant advantage of our proposal is that it can be implemented straightforwardly by both public and non-governmental institutions in a short time, as it relies on accessible technology and open source software.

Resumen

La contaminación ambiental es uno de los principales problemas que afecta a nuestro planeta. El crecimiento industrial y los aglomerados urbanos, entre otros, están contribuyendo a que dicho problema se diversifique y se cronifique. La presencia de contaminantes ambientales en niveles elevados afecta la salud humana, siendo la calidad del aire y los niveles de ruido ejemplos de factores que pueden causar efectos negativos en las personas tanto psicológicamente como fisiológicamente.

Tradicionalmente, la contaminación ambiental se mide a través de centrales de monitorización, que por lo general son fijas y tienen un coste elevado. Sin embargo, la ubiquidad de los microcomputadores, y el aumento de los sensores incorporados en nuestros smartphones, han hecho posible la aparición de nuevas estrategias para medir dicha contaminación. Así, el Mobile Crowdsensing se ha convertido en un nuevo paradigma mediante el cual los teléfonos inteligentes emergen como tecnología habilitadora, y cuya adopción generalizada proporciona un enorme potencial para su crecimiento, permitiendo operar a gran escala, y con unos costes asumibles para la sociedad. A través del crowdsensing, los teléfonos inteligentes pueden convertirse en unidades de detección flexibles y multiuso que, a través de los sensores integrados en dichos dispositivos, o combinados con nuevos sensores, permiten monitorizar regiones de interés con una buena granularidad tanto espacial como temporal.

En esta tesis nos centramos en el diseño de soluciones de crowdsensing usando smartphones donde abordamos problemas de contaminación ambiental, específicamente del ruido y de la contaminación del aire. Con este objetivo, se estudian, en primer lugar, las propuestas de crowdsensing que han surgido en los últimos años. Los resultados de nuestro estudio demuestran que todavía hay mucha heterogeneidad en términos de tecnologías utilizadas y métodos de implementación, aunque los diseños modulares en el cliente y en el servidor parecen ser dominantes.

Con respecto a la contaminación del aire, proponemos una arquitectura que permita medir la contaminación del aire, concretamente del ozono, dentro de entornos urbanos. Nuestra propuesta utiliza smartphones como centro de la arquitectura, siendo estos dispositivos los encargados de leer los datos de un sensor móvil externo, y de luego enviar dichos datos a un servidor central para su procesamiento y tratamiento. En esta propuesta se han analizado varios problemas con respecto a la orientación del sensor externo y al tiempo de muestro, y se ha validado la solución propuesta en escenarios reales. Los resultados obtenidos demuestran que la orientación del sensor y el período de muestreo, dentro de ciertos límites, tienen muy poca influencia en los datos capturados. Además, mediante una comparativa de los mapas de calor generados por nuestra solución con los datos de las estaciones de monitorización existentes en la ciudad de Valencia, demostramos que nuestro enfoque es capaz de proporcionar una mayor granularidad de los datos.

Con respecto a la contaminación acústica, proponemos una arquitectura para medir los niveles de ruido en entornos urbanos basada en crowdsensing, y cuya característica principal es que no requiere intervención del usuario. En esta tesis detallamos aspectos tales como la calibración de los smartphones, la calidad de las medidas obtenidas, el instante de muestreo, el diseño del servidor, y la interacción cliente-servidor. Además, hemos validado nuestra solución en escenarios reales para demostrar el potencial de la solución alcanzada. Los resultados experimentales muestran que, con nuestra propuesta, es posible medir niveles de ruido en diferentes zonas urbanas o rurales con un grado de precisión comparable al de los dispositivos profesionales, todo ello sin requerir intervención del usuario, y con un consumo reducido en cuanto a recursos del sistema.

En general, las diferentes contribuciones de esta tesis doctoral ofrecen un punto de partida para nuevos desarrollos, ofreciendo estrategias de calibración y algoritmos eficientes de cara a realizar medidas representativas. Además, una importante ventaja de nuestra propuesta es que puede ser implementada de forma directa tanto en instituciones públicas como no gubernamentales en poco tiempo, ya que utiliza tecnología accesible y soluciones basadas en código abierto.

Resum

La contaminació ambiental és un dels principals problemes que afecten el nostre planeta. El creixement industrial i els aglomerats urbans, entre altres, estan contribuint al fet que aquest problema es diversifique i es cronifique. La presència de contaminants ambientals en nivells elevats afecta la salut humana, sent la qualitat de l'aire i els nivells de soroll exemples de factors que poden causar efectes negatius en les persones, tant psicològicament com fisiològicament.

Tradicionalment, la contaminació ambiental es mesura a través de centrals de monitoratge, que en general són fixes i tenen un cost elevat. No obstant això, la ubiqüitat de les microcomputadores i l'augment dels sensors incorporats als nostres telèfons intel·ligents han fet possible l'aparició de noves estratègies per a mesurar aquesta contaminació. Així, el mobile crowdsensing s'ha convertit en un nou paradigma mitjançant el qual els telèfons intel·ligents emergeixen com a tecnologia habilitadora, i l'adopció generalitzada d'aquest proporciona un enorme potencial per al seu creixement, ja que permet operar a gran escala i amb uns costos assumibles per a la societat. A través del crowdsensing, els telèfons intel·ligents poden convertir-se en unitats de detecció flexibles i multiús que, a través dels sensors integrats en els esmentats dispositius, o combinats amb nous sensors, permeten monitoritzar regions d'interès amb una bona granularitat, tant espacial com temporal.

En aquesta tesi ens centrem en el disseny de solucions de crowdsensing usant telèfons intel·ligents, on abordem problemes de contaminació ambiental, específicament del soroll i de la contaminació de l'aire. Amb aquest objectiu, s'estudien, en primer lloc, les propostes de crowdsensing que han sorgit en els últims anys. Els resultats del nostre estudi demostren que encara hi ha molta heterogeneïtat en termes de tecnologies utilitzades i mètodes d'implementació, encara que els dissenys modulars en el client i en el servidor semblen ser dominants.

Pel que fa a la contaminació de l'aire, proposem una arquitectura que permeta mesurar la contaminació d'aquest, concretament de l'ozó, dins d'entorns urbans.

La nostra proposta utilitza telèfons intel·ligents com a centre de l'arquitectura, sent aquests dispositius els encarregats de llegir les dades d'un sensor mòbil extern, i d'enviar després aquestes dades a un servidor central per al seu processament i tractament. En aquesta proposta s'han analitzat diversos problemes pel que fa a l'orientació del sensor extern i al temps de mostratge, i s'ha validat la solució proposada en escenaris reals. Els resultats obtinguts demostren que l'orientació del sensor i el període de mostratge, dins de certs límits, tenen molt poca influència en les dades capturades. A més, mitjançant una comparativa dels mapes de calor generats per la nostra solució amb les dades de les estacions de monitoratge existents a la ciutat de València, vam demostrar que el nostre enfocament és capaç de proporcionar una major granularitat de les dades.

Pel que fa a la contaminació acústica, proposem una arquitectura per a mesurar els nivells de soroll en entorns urbans basada en crowdsensing, i la característica principal de la qual és que no requereix intervenció de la persona usuària. En aquesta tesi detallem aspectes com ara el calibratge dels telèfons intel·ligents, la qualitat de les mesures obtingudes, l'instant de mostratge, el disseny del servidor i la interacció client-servidor. A més, hem validat la nostra solució en escenaris reals per a demostrar el potencial de la solució assolida. Els resultats experimentals mostren que, amb la nostra proposta, és possible mesurar nivells de soroll en diferents zones urbanes o rurals amb un grau de precisió comparable al dels dispositius professionals, tot això sense requerir intervenció de l'usuari o usuària, i amb un consum reduït quant a recursos del sistema.

En general, les diferents contribucions d'aquesta tesi doctoral ofereixen un punt de partida per a nous desenvolupaments, i ofereixen estratègies de calibratge i algorismes eficients amb vista a realitzar mesures representatives. A més, un important avantatge de la nostra proposta és que pot ser implementada de forma directa tant en institucions públiques com no governamentals en poc de temps, ja que utilitza tecnologia accessible i solucions basades en el codi obert.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Currently, human beings are contributing to the degradation of the environment, which has become one of the most significant problems for humanity. The economic growth, the industrial development, the increasing number of vehicles in circulation, and large metropolis, create a series of conditions that affect, to a lesser or greater extent, the quality of the environment. In these circumstances, industrialized societies have altered the ecosystem by introducing different types of pollutants, including air pollution, water contamination, soil contamination, noise pollution, and thermal and radioactive contamination, among others.

In general, these pollutants are harmful to human health in different ways. In particular, people living in urban environments are prone to experience high air pollution levels, as well as increasing environmental noise. The first of these problems clearly affect the life quality of people, causing health problems, and also having a negative impact on the whole planet by provoking climatic changes and reduced agricultural production [3, 93, 94]. Regarding the second problem, it has been shown that people subjected to a certain level of noise can present alterations such as loss of hearing ability, disturbed sleep and rest, communication difficulties, irritability, aggressiveness, as well as problems to develop attention and mental concentration.

Taking the issues mentioned before into account, current concerns for environmental protection require undertaking a constant supervision. Traditional solutions based on static environmental monitoring stations, are in general characterized by having high costs and large sizes, thereby allowing to obtain pollution values with a high precision, but with a low level of granularity. On the other hand, technological developments have enabled the used of alternative approaches based on crowdsensing to take advantage of the resources and sensors integrated into smartphones, promoting collaborative monitoring in densely populated areas.

In this regard, crowdsensing solutions aimed at monitoring environmental noise and air quality are characterized by having designs where the quality of the data obtained, as well as energy consumption issues, have not been fully addressed. Therefore, getting an integral and reliable solution that allows measuring the levels of environmental contamination through crowdsensing could be an effective approach to facilitate the establishment of effective policies to improve our environment.

1.2 Objectives and Methodology

The main objective of this thesis is the creation and development of crowdsensing solutions for monitoring environmental pollution using smartphones, and requiring a minimum user intervention. Specifically, we will propose a solution for measuring air quality, and another solution to measure noise levels in urban environments.

To achieve the general goals of the thesis, the following specific objectives are defined:

- Study the state of the art to determine the most relevant contributions in the crowdsensing area and propose a generic crowdsensing architecture based on a client-server approach.

Regarding air pollution:

- Propose a solution for environmental monitoring for measuring air pollution (specifically, ozone levels), including sensors, Android applications, and a web-based cloud application, and validate the proposed solution through real experiments.

- Analyze the data sampling process on how to properly calibrate the sensor, and how to reduce time variability.

- Study the best strategy to use mobile sensors by first determining the influence of sensor orientation on the captured values, and then analyzing the influence of time and space sampling in the interpolation process.

- Validate the proposed architecture through comparison against infrastructure-based data.

Regarding noise pollution:

- Study the feasibility of using commercial smartphones as noise level measurement units by determining their accuracy when compared to a professional noise measurement unit.
- Determine the best sampling strategy and algorithm to retrieve noise level samples using different types of smartphones.
- Determine the most adequate strategy for capturing environmental noise using smartphones by accounting for their context while minimizing energy consumption.
- Develop an Android application for noise sensing offering reliable and meaningful measurements.
- Develop a web interface based on central management of crowdsensing tasks accounting for spatial and temporal restrictions, and offering transparent task dissemination to the system users.
- Validate the proposed noise sensing architecture in real environments.

Regarding the methodology used in this thesis, first the different sensors adopted will be calibrated by relying on professional devices, which will be used as reference. Then, a development phase will follow where the different restrictions of the crowdsensing architecture are taken into account for creating both the client and the server software sides. Finally, the effectiveness of the proposed crowdsensing architecture will be assessed through real field tests.

1.3 Organization of the thesis

The rest of this thesis is organized as follows. In the next chapter, we present the state of the art regarding smartphone-based crowdsensing solutions. To properly analyze these previous works, we first define a reference framework based on which we proceed to classify the different proposals. Chapter 3 we present an overview of the environmental pollution problem. Chapter 4 presents Ecosensor, an architecture that enables performing air quality measurements based on crowdsensing. In Chapter 5 we present our full architecture for noise pollution monitoring. Chapter 6 details the procedure to achieve accurate ambient noise levels using commercial smartphones. In Chapter 7 we show how to make sure that meaningful noise values are obtained by proposing a decision tree that avoids invalid samples while minimizing resource consumption. In Chapter 8 we present the validation of the proposed architecture in real environments. Finally, In Chapter 9 we present the conclusion of this work, as well as ideas for future work. It also includes a summary of the different publications related to this thesis.

Chapter 2

Smartphone-based CrowdSensing: Existing Solutions

In this chapter, we analyze a set of previous works regarding crowdsensing solutions based on smartphones. To this aim, we first describe the mobile crowdsensing concept and analyze different surveys carried out to date. Then, we propose a taxonomy to classify the existing proposals adequately, and we do proceed to analyze, with appropriate discussion the different works available in the literature from the perspective of the proposed taxonomy.

2.1 The CrowdSensing concept

THE use of mobile phones has experienced a significant increase in the past decade. In fact, according to the 2015 International Telecommunication Union (ITU) report [56] for 2015, the ratio of cellular phone subscriptions was 97%, which represents 7084 million subscribers in the world. In addition, this subscriber increase is reflected in the technological advantages offered by mobile devices. Furthermore, mobile devices available nowadays have a high computational power, and include different communication technologies (e.g., WiFi, 4 Generation Telecommunication (4G), Bluetooth), and have multiple embedded sensors (e.g., Global Positioning System (GPS), gyroscope, accelerometer, microphone and camera, among others). This technological growth, together with the increasing number of subscribers, has caused the community of researchers and developers to create different applications using smartphones as sensors.

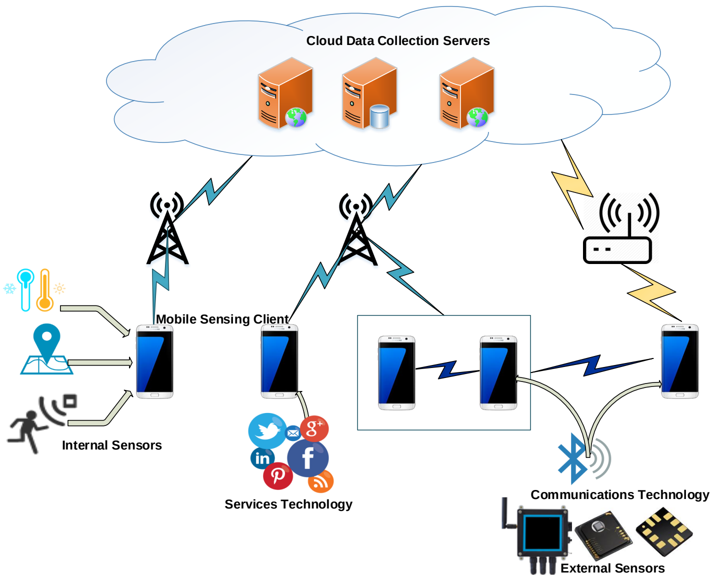


Figure 2.1: Generic structure of crowdsensing solutions.

Pioneering research anticipated the arising of such new applications, describing them as "participatory sensing" [22] or "people-centered sensing" [25]. In both cases, the idea is that the user should be able to gather data anywhere, anytime, by making use of mobile sensor devices for information retrieval, processing and sharing. Later on, researchers considered this new paradigm as a subtype of crowdsensing denoted as "mobile phone sensing" [72].

Mobile phone sensing benefits from the processing and communication capabilities of available smartphones which, combined with one or more sensors, becomes an enabling technology to support different types of applications. Moreover, mobile crowdsensing relies on a large number of participants to collect data from the environment through its integrated sensors and, after capturing the data, these data are sent to a server to perform data mining tasks including data-fusion, analysis, and information dissemination. Typically, sensors that register participant information (e.g., location, movements) and environmental data (e.g., images, sounds) are very common. On top of that, some solutions use external sensors, which are integrated into the mobile solution through its communication interfaces, including sensors for environmental pollution and health monitoring. In this sense, mobile crowdsensing provides new perspectives for improving living conditions in our digital society. A general example of a mobile crowdsensing solution is shown in Figure 2.1.

To date, we can find different surveys that study and summarize the many existing proposals. Below we proceed to describe briefly the different surveys found in the literature according to their chronological order of publication.

- Lane et al. [72] made a pioneer survey addressing the use of mobile phones as sensors, analyzing the significant progress mobile phones have experienced in order to incorporate multiple sensors. Also, they describe various existing proposals according to certain algorithms, applications and systems developed to date. Similarly, they describe some proposals grouped by areas such as transportation, environmental monitoring, and health. They also propose an architecture composed by three different elements dedicated to sensing (mobile phone), learning (analysis) and informing (shared data).

- Ganti et al. [44] proposed the term Mobile Crowd-Sensing (MCS) and described a reference architecture. This survey only showed a few proposals categorized as participatory (users are involved) and opportunistic (users are not involved). Additionally, it identifies some characteristics that influence these solutions such as limited resources, privacy and security, data integrity, data aggregation and data analytics.

- Khan et al. [68] present a taxonomy where they differentiate between personnel sensing, social sensing and public sensing. This classification is performed from the point of view of participatory and opportunistic sensing.

- Zhang et al. [148] propose an approach which characterizes the various crowdsensing proposals in four stages: task creation, task assignment, individual task execution, and crowd data integration. These features are described as What, When, Where, Who, and How (4W1H).

- More recently, both Zhang et. al. [152] and Jaimes et al. [58] proposed a classification based on incentive mechanisms for mobile crowdsensing. The former classified incentives into three categories: entertainment, services, and economic. The latter used incentive mechanisms as metrics to evaluate crowdsensing, and introduced a tree-level taxonomy for crowdsensing incentive mechanisms. In the same context, different authors [60, 123, 61] propose incentive mechanisms that rely on auction techniques for evaluating quality awareness in the mobile crowdsensing context. In particular, the first uses combinatorial auction models, while the second extends that work by introducing more fine-grained techniques; concerning the third work, it proposes a framework that integrates incentives, data aggregation and data perturbation mechanisms. However, they did not propose any reference architecture, addressing solely the taxonomy of their proposals, and the algorithms supporting these proposals.

- Guo et al. [47] propose a new sensing paradigm called Mobile Crowd Sensing and Computing (MCSC) that empowers ordinary citizens to contribute data sensed or generated from their mobile devices, aggregating and fusing the data in the cloud for crowd intelligence extraction and human-centric service delivery. This paper proposes a taxonomy and a reference architecture for MCSC. In the taxonomy the proposals are classified as: mobile sensing (user involvement, data contribution, user awareness, sampling), crowd data collection (networking, incentives, scale), crowdsourced data processing and intelligence extraction (processing architecture, intelligence, purpose, data mining, data quality), hybrid human-machine system, and security & privacy. With regard to the architecture, the proposals presented in this paper are divided into several levels: crowdsensing, data collection, data processing, and applications.

As it quickly becomes evident through this brief state-of-the-art analysis, to date only few surveys [44, 148, 152, 58, 47] specifically addressed existing crowdsensing solutions, is that some authors focused on specific issues such as incentives, and yet others focused on sensing styles. Instead, in this chapter, we will propose a reference client-server architecture. Then, based on that architecture, we proceed to classify up to 64 different proposals, thus providing a more comprehensive view than the surveys presented before on this topic (see Table 2.1 for details). Notice that the number of peer-reviewed publications only takes into account those references actually classified according to the proposed taxonomies.

Figure 2.2 shows the number of crowdsensing-related proposal search on different impact journal and conference, and that mobile crowdsensing applications have experienced a significant increase in the last five years. Specifically, researchers have focused their efforts on various areas including environment monitoring [105, 108, 113, 145], transportation and urban sensing [6, 26, 67, 75, 117], healthcare [54, 78, 100, 128, 155], social issues [21, 49, 83, 92, 130], and others [38, 97, 103, 135, 146].

Table 2.1: Crowdsensing surveys.

Solutions	Year	# Publications reviewed
Lane et al. [72]	2010	25
Ganti et al. [44]	2011	13
Khan et al. [68]	2013	43
Calabrese et al. [23]	2014	26
Zhang et al. [148]	2014	15
Zhang et. al. [152]	2015	32
Jaimes et al. [58]	2015	22
Guo et al. [47]	2015	41

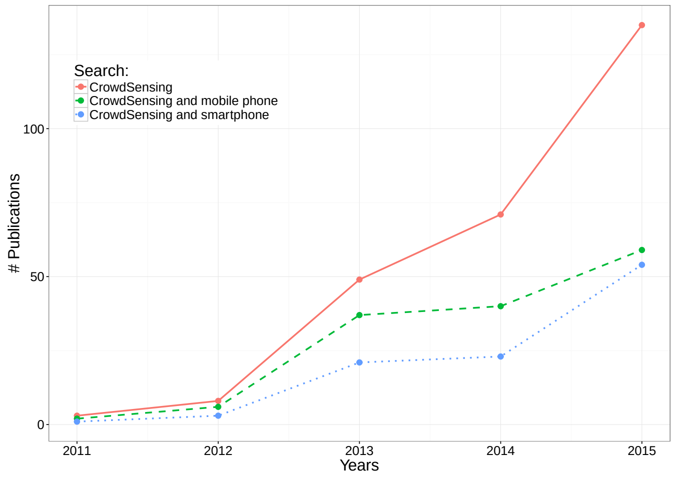


Figure 2.2: Number of Crowdsensing-related proposals in the past 5 years.

The different crowdsensing proposals available are characterized by having different designs, and involve different architectural levels. For instance, some authors propose solutions they call framework, middleware, or system, among other terms. No matter which term is used, these solutions can have a global approach (full architecture), or only specify a subset of the architecture by describing one or more components.

2.2 Reference Architecture for Mobile CrowdSensing

In this section, we propose a client-server design which can be adapted to the different mobile crowdsensing architectures available in the literature. By making the different proposals fit into our architecture, in sections that follow it will then become straightforward to compare these proposals in terms of scope, complexity, and completeness.

Our proposed architecture integrates two main modules: the Mobile Sensing Client (MSC) module, and the Cloud Data Colletion Server (CDCS) module. These two modules are connected to each other through a Data Transmission network, as shown in Figure 2.3. The MSC is the mobile phone, or the set of mobile phones, that provide sensing functionality by capturing data, and then relaying that data to the CDCS. The latter is a single server or a server farm that allows receiving, processing, analyzing, and sharing sensed data. Typically, data sharing also includes the delivery of reports to participants MSC.

For both the MSC and the CDCS we have considered four subcomponents, some of them sharing common characteristics on both MSC and CDCS. For in-

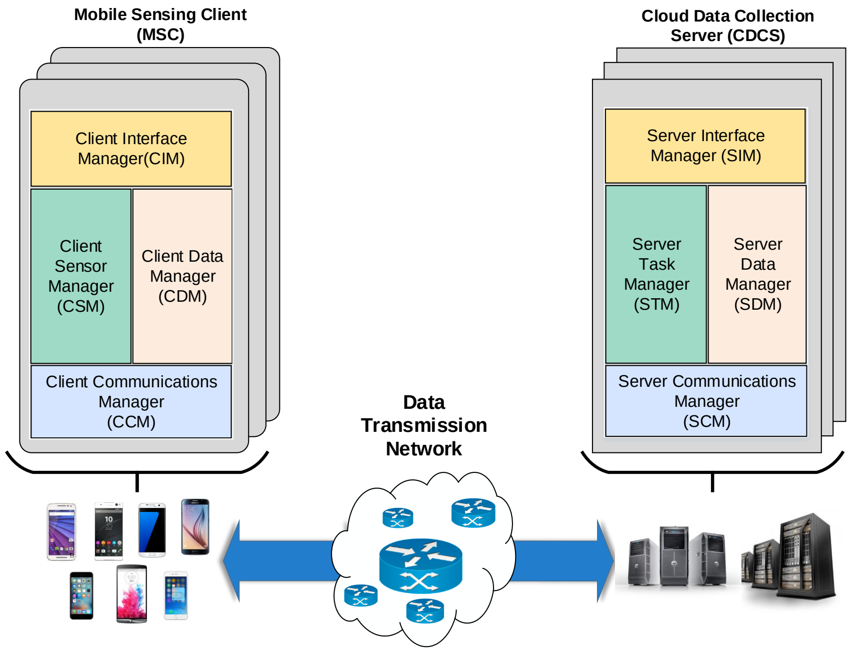

Mobile Sensing Client (MSC)
Client Interface Manager(CIM)
Client Sensor Manager (CSM)
Client Data Manager (CDM)
Client Communications Manager (CCM)
Data Transmission Network
Cloud Data Collection Server (CDCS)
Server Interface Manager (SIM)
Server Task Manager (STM)
Server Data Manager (SDM)
Server Communications Manager (SCM)

Figure 2.3: Proposed Mobile CrowdSensing architecture.

stance, both Client Interface Manager (CIM) and Server Interface Manager (SIM) provide a graphical interface to a regular user or to the system administrator through the respective Interface Managers. On the bottom of the architecture, the Server and Client Communications Manager (CCM) have also a similar purpose, typically being the component on the client that establishes connections with the server component since it should always be available. Nevertheless, configuration and task instructions, along with data reports, can also be transmitted from the server to client through a push procedure.

The Data Management components at client and server sides, also have some similarities, both being responsible for data processing, storage, and query. The main difference between these subcomponents is that, in the CDCS, the computation, storage, and analysis is made at a level and dimension that is clearly superior to the one made at the client, which has fewer resources.

Two distinctive components in our architecture are the Client Sensor Manager (CSM), responsible for the administration of the sensors, and the Server Task Manager (STM), that handles different tasks mostly related to data processing. Below we proceed to describe the different architectural elements in more detail.

2.2.1 Mobile Sensing Clients (MSCs)

In the scope of mobile crowdsensing, the main goal of the mobile client devices is performing data sensing, and forwarding sensed data to the main server, although global data reports can also be returned to clients.

Concerning the target areas to be sensed, these can differ greatly depending on the type of application (inside buildings, outdoor, underground, in public places, etc.). In addition, each specific application will also have different requirements in terms of required sensors. For instance, sensors able to monitor the environment greatly differ from those able to monitor social interactions or the effectiveness of public transportation. In addition, sensing tasks can be triggered automatically (either periodically or based on events), or manually through an explicit user intervention. Typically, automatic mechanisms follow server instructions, while manual interactions are made possible through a user interface specifically developed for that purpose. Independently of the actual mode of operation, the application can offer certain incentives in the form of a game [119] or other approaches, to motivate users into adopting it. Such incentives become specially important when the user interest in the global generated data is based on the aggregation and processing of all measurements at the server. Also, it often occurs that the user interest remains low (e.g., data being sensed is not a concern to the user); in those cases, complementary sources of motivation are required to make users run the crowdsensing application.

Focusing on the client architecture, Figure 2.4 shows that, to support all user activities, we have a set of managers responsible for all tasks: Client Interface Manager (CIM), Client Data Manager (CDM), Client Sensor Manager (CSM) and Client Communications Manager (CCM). Each of these four components has a controller subcomponent, being the different controller elements the ones actually responsible for supporting bidirectional interactions between the different system elements. We now proceed to detail each of the Client components.

Client Interface Manager (CIM)

This component allows applications to interact with the user (Graphical User Interface (GUI)). The user interface allows displaying the values obtained from sensors in real time, to visualize previous traces through a query to its internal data storage, or to query the server in order to retrieve global data reports about a specific target area. The values can be visualized through the use of graphics, maps, or other forms of representation. To achieve this goal two subcomponents are proposed: the Client User Interface, and the Interface Controller:

- *Client User Interface.* It allows configuring the different parameters associated to sensing tasks, such as regulating the data acquisition frequency, defining when data should be sent to the server, and also when captures should start and stop, among others. It can also show the user feedback

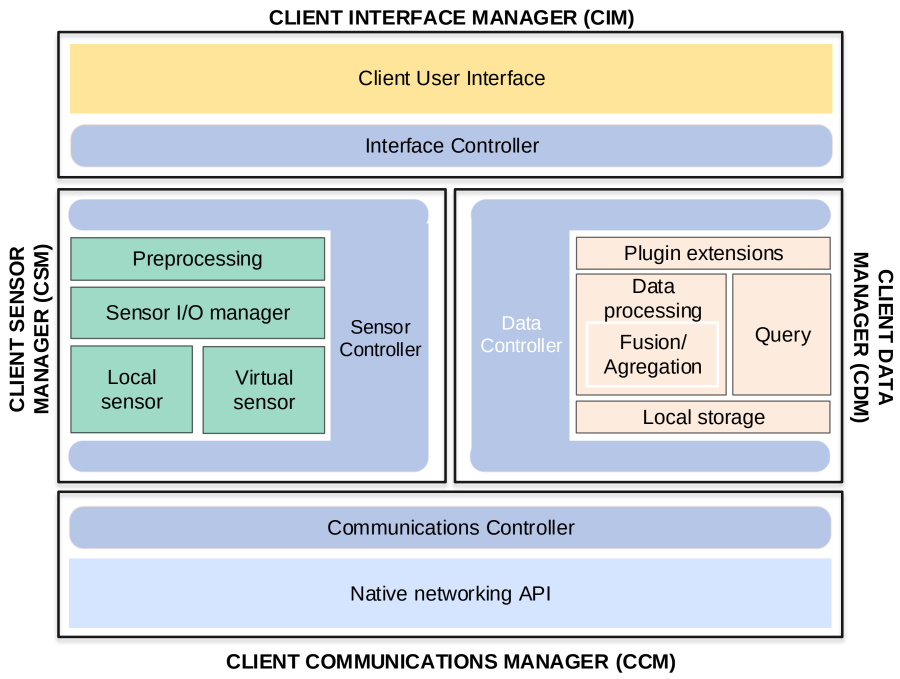


Figure 2.4: Mobile Sensing Client (MSC) Components.

about ongoing or past captures, as well as global reports. It is worth highlighting that some crowdsensing solutions have no interface at the client side, meaning they only process captured data and relay them to the server.

- *Interface Controller.* It provides the needed services to format data for presentation through the User Interface. For this endeavor, it must interact with the local storage or with the server, and it may rely on different external libraries as well (e.g. graphical representation of captured values in a map using Google Maps).

Client Data Manager (CDM)

This element, responsible for data handling and storage, is one of the main architectural elements of the client. It is composed by five different subcomponents: Data Controller, Plugin extensions, Data Processing, Local Storage, and Query. We now proceed to detail each of them:

- *Data Controller.* It is the most critical subcomponent, providing the services and functions required to interact with the different subcomponents of the CDM. This interaction is made with the client via the Interface Controller, with the server via the Communications Controller, and with the sensors via the Sensor Controller. Also, it can handle data collection tasks as defined by the user, or defined by the server through task pushing. It includes classes and methods to start, stop, and configure these tasks.

- *Plugin extensions.* This element allows integrating specialized plugins for a specific task such as data analytics, or to add listeners to social networks including Facebook and Twitter, among others. The advantage of these plugins is that they can be easily incorporated to mobile devices via repositories such as Google Play or similar ones. Additionally, it allows plugging-in a set of algorithms that perform functions including audio processing, online programming algorithms, and spatial coverage analysis.

- *Data Processing.* This element processes raw data based on application requirements before displaying them to the end user or submitting them to the server. Although data processing can also be executed at the server side CDCS, doing it at the client allows reducing the amount of unnecessary data produced by sensors, while also maximizing energy savings and communications bandwidth, and so it is often preferred. Typically, data processing elements include either filtering, aggregation, or both functions. An example of filtering is the removal of unnecessary data fields. Examples of aggregation/fusion of data include the unification of data from different sensors, or of different samples from the same sensor.

- *Local Storage.* This element allows storing the captured data in a local data structure, which is usually a simple database like SQLite. Some solutions available in the literature skip this component, and they only process data and forward them to the CDCS. The local storage allows users to perform queries, inserts, updates, and deletes to the data according to application requirements. Typically, when storing data coming from sensors, it is often preprocessed before storage. In the context of crowdsensing applications, the main types of data stored include location information, energy levels, and sensor-specific values.

- *Query.* This element allows, through structured language queries, to access data from sensors. In particular, it will interact with all the components that make up the CDM. Among its typical features, one that stands out is the use of mobile analytics for optimizing data streaming from sensors. In some cases, this component facilitates the interaction with external databases at the CDCS in order to retrieve global data reports.

Client Sensor Manager (CSM)

The Client Sensor Manager is the element responsible for the actual sensing tasks. Typically, it relies on high-level sensor abstractions to manage the underlying physical sensors (internal or external) as well as virtual sensors. Its functions usually include sensor discovery and sensing capabilities. Furthermore, it manages the sensor sampling frequency, as well as the preprocessing of captured data. The preprocessing executed at the CSM is only performed if necessary and considering the actual characteristics of the sensor. Finally, the integration of external sensors and virtual sensors is performed by the sensor controller via the communications manager.

The CSM has five subcomponents: Sensor Controller, Preprocessing, Sensor Input/Output (I/O) Manager, Physical Sensor and Virtual Sensor. Below we describe each of its components:

- *Sensor Controller.* It enables access to the services offered by the Sensor Manager, thus providing access to virtual sensors, gyroscope, and GPS, among others.

- *Preprocessing.* It allows the data delivered by the Sensor I/O Manager to be processed before being passed to other components. An application example is an audio capture which must be classified into voice and non-voice regions so that the individual speaker is segmented. Another example is the raw accelerometer data that is provided for the three axes, which can be combined to obtain the total value. In some cases, these raw data can be processed at both CSM and CDM.

- *Sensor I/O Manager.* It allows a level of abstraction for accessing both physical and virtual sensors, getting the raw data for subsequent treatment. This way, upper layers do not have to be aware of the type of sensor (physical/virtual) and its actual location.

- *Local Sensor.* These are sensors available either on mobile devices themselves, or nearby external sensors directly accessible by the mobile device. Concerning the type of sensor, most internal sensors used belong to the generic or media type. Generic sensors are those sensors embedded in mobile devices for general-purpose applications. Examples of these sensors include: GPS, accelerometer, gyroscope, magnetometer, and barometer, among others. With regard to media sensors, it refers to embedded sensors that provide support to multimedia applications via microphone or camera. Finally, external sensors typically extend the sensing functionality by providing sensing capabilities not supported by the smartphone itself.

- *Virtual Sensor* is a logical type of sensor based on an abstract class that acts as a wrapper, encapsulating information that can be produced by a real sensor, a mobile phone, or a combination of other virtual sensors. Virtual Sensors can have multiple input data streams that can be other virtual sensors or sensors accessible through a network, but there can be only one output data stream towards the sensing application. The Global Sensor Networks (GSN) standard data model [71] is a good example of such a class of sensors.

Client Communications Manager (CCM)

The Client Communications Manager is responsible for the transmission and reception of the data through the network. Since nowadays mobile phones include several communication interfaces including Wireless Fidelity (WiFi), Bluetooth, or Cellular, this empowers them to communicate in all sorts of environments, being able to adapt to different network topologies (centralized, distributed, or hybrid). The MSC may transfer the data to a primary server (centralized), toward several servers (distributed), or among themselves Peer-To-Peer (P2P). The latter occurs when there are nodes that serve as intermediaries for the transmission of data between nodes, and that have a limited ability to process and filter data from the sensor.

The CCM is composed by two subcomponents: the Communications Controller and the Native Networking - Application Programming Interface (API). In detail, these subcomponents are responsible for the following tasks:

- *Communications Controller.* It provides access to the services of the underlying communications network, allowing to create a data channel towards the CDCS. In particular, it is an abstract component that allows encapsulating Simple Object Access Protocol (SOAP) and Representational State Transfer (RESTful) web services, where the first is an Extensible Markup Language (XML)-based protocol that uses service interfaces to expose the business logic, and the second is an architectural paradigm that supports different data formats including JavaScript Object Notation (JSON), XML, Hypertext Markup Language (HTML), and Plain Text (TXT). Since communication between clients may also be required, this component will be endowed with P2P networking capabilities, possibly acting as a relay between other clients and the server(s).

- *Native networking API.* This component is inherent to each mobile operating system platform, and it is the one providing the actual establishment of end-to-end connections between client and server.

2.2.2 Cloud Data Collection Server

In the context of Crowdsensing applications, the main goal of the server component, which may physically consist of a single server or a server farm, is to collect all data gathered by the different clients, storing the data, and then performing all sorts of data analytics. Then, to provide the administrator or clients themselves with a summary of the most relevant information. Also, the server allows defining and automating some of the data collection tasks. For example, the administrator can create new tasks, and these can be deployed to clients either automatically or manually; an example of this can be the collection of the noise levels for a given target area during a given period of time. Figure 2.5 shows our proposed architecture for the CDCS, which includes four components: Server Interface Manager (SIM) , Server Task Manager (STM), Server Data Manager (SDM), and Server Communications Manager (SCM). Notice that each of these components includes a controller. Such controllers have a critical function in the scope of our architecture, as it is the communication between adjacent controllers that allows the different components to work together, similarly to the situation at the client side. Compared to clients, CDCS elements have much greater processing and storage capabilities. Thus, data are typically processed for a better understanding through different statistical techniques (Data Mining). Also, the management interface is usually web-based, allowing the administrator to manage, visualize and share large amounts of data easily.

Depending on network scalability requirements, servers may work in either centralized, distributed, or cloud-based environments. The latter allows to benefit from deployment facilities, reduced cost, and optimized resource usage, thereby minimizing infrastructure requirements. Concerning available technologies, server solutions may rely on a wide range of platforms, from distributed architectures in the cloud such as Amazon Web Services (AWS) infrastructure services (Amazon Elastic Compute Cloud (EC2) and Amazon Simple Cloud Storage Service (S3)) [35], and Google Cloud Messaging (GCM) [36], to open source approaches such as Apache Tomcat [103, 84, 112, 65], BPEL4People [31, 55], WS-HumanTask, and JBoss JBPM [54].

Below we describe in more detail the different components at the server side.

Server Interface Manager (SIM)

The Server Interface Manager is responsible for the interaction between user and system for task and data handling. It includes two components: the Server User Interface and the Interface Controllers.

- *Server User Interface.* It allows the user to manage and schedule sensing tasks interactively. It also supports the visualization of charts relative to sensed data. Both these actions are performed using a graphical interface that is in general web-based, meaning that the system manager can operate remotely.

- *Interface Controller.* This is the component actually in charge of communicating with other components to meet the service requirements. An example

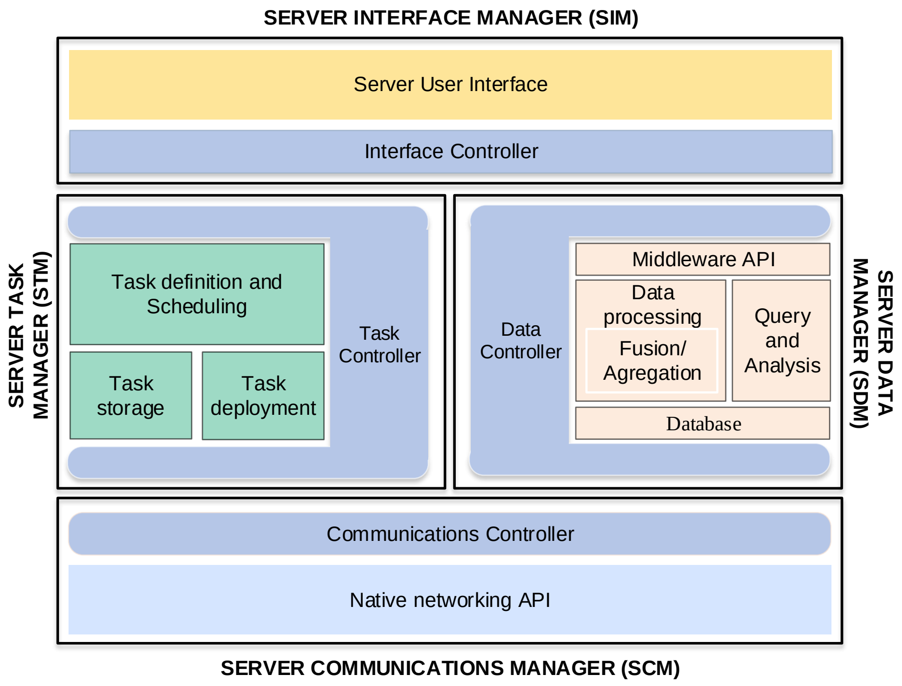


Figure 2.5: Cloud Data Collection Server Components.

is the programming of a sensing task, where the interface controller coordinates with task controllers for task planning and dissemination, and with the data controller for handling data storage. In addition, it also provides API to allow developers to participate in the development of different crowdsensing applications and services.

Server Task Manager (STM)

Task Management is one of the main components at the server side according to our proposed architecture, being responsible for the planning, scheduling, and pushing of crowdsensing tasks. Tasks can be deployed to mobile devices either manually or automatically, and in general, they rely on a system-specific language that typically differs from one solution to another due to lack of standardization. It is also worth highlighting that most implementations rely on open source tools.

The subcomponents that integrate the STM are the following:

- *Task Controller.* It works as a handler, providing the functionality required by the server task manager. Typically it must attend administrator requests and may push new scheduled tasks onto clients. It can also make use of learning or approximation algorithms that optimize data collection in order to minimize energy/resource consumption at the client side. Additionally, this subcomponent includes classes and methods to start, stop, and configure the different tasks. Finally, it provides the services and functions required to interact with the other controller components. To contact with clients, some implementations are based on Publish/Subscribe approaches where a server (or servers) provides a set of services to users. Additionally, in many of these publish/subscribe systems, the server can take intermediary functions where publishers send the messages to such intermediary server (broker), and the subscribers subscribe to information considered to be of interest, thus making this server responsible for handling the filtering, storage, and management toward the subscribers.
- *Task Definition and Scheduling.* Among its features we can find the allocation of time and frequency of sensing, the number of mobile devices to be enabled for data collection, and the characteristics of the sensor to monitor, among others.
- *Task Deployment.* It allows the deployment of tasks to MSCs, which can be a mere set of instructions interpreted by the existing applications. To support this option, a language defined by the application is often used, and it is typically based on Structured Query Language (SQL), XQuery, or XML. Alternatively, a new application/component is pushed to the mobile terminal whenever new functionalities must be supported.
- *Task Storage.* This component is responsible for the storage of current and past tasks. Since requirements are typically low, any database system suffices. In fact, it is not necessary to rely on a standard database, being also common to use a set of files, where each file describes a single task.

Server Data Manager (SDM)

This component is responsible for the processing, storage, and analysis of the data. It is composed of a data controller, middleware APIs, a data processing element, a query and analysis element, and a database. Below we describe in more detail each of these components.

- *Data Controller.* It offers access to the services offered by the SDM, supporting a set of algorithms or applications that allow handling of data in collaboration with the task controller, interface controller, and other system components. In addition, it acts as a handler for communications to/from middleware APIs.

- *Middleware API.* A middleware API is typically an extension providing more sophisticated data processing/analysis. It can incorporate data analysis tools such as Data Mining, analytical libraries or other, allowing to easily handle large volumes of data. Deployments at this level can be drivers or web services that enable access to databases through Java Database Connectivity (JDBC) or other methods, such as CUPUS [11] and CAROM [146], which additionally provide data fusion and data filtering techniques.

- *Data Processing.* The functionality of this component is similar to the data processing made at the client CSM. The main difference is the volume of data that has to be handled at the server side. Typically, it provides functions to filter and merge multiple streams of data, providing aggregation levels that clearly surpass those levels achievable at the client side. With this purpose, it uses techniques that require a higher level of processing, such as Fluid Structure Interaction (FSI) [146], among others.

- *Query and Analysis.* This component integrates both query and data analysis functionalities. It allows, through a structured query language, to access the resources available at the server's database. Additionally, it can rely on different analysis tools to meet the requirements of other system components.

- *Database.* This component provides a database management system that allows storing the gathered data coming from the different Mobile Sensing Clients (MSCs). In the scope of the SDM, it is mandatory since it is a basic system requirement. It should be noted, though, that the database itself is not necessarily contained in a single server, and so distributed storage environments are contemplated as well. Common database management systems include MySQL and PostgreSQL, among others.

Server Communications Manager (SCM)

The Server Communications Manager is responsible for interacting with the different clients, having characteristics similar to the Client Communications Manager. The interaction with clients is bidirectional: we have transmission toward the client when pushing new tasks, and we have transmissions from clients when receiving sensed data. The SCM has two main components, the Communications Controller, and the Native networking API, both of which we now detail:

- *Communications Controller.* It offers the services necessary to establish communication between the MSC and the CDCS, usually as listeners for data gathering, or starting connections when task pushing is required. Additionally, it can rely on high-level communication services like SOAP, and can also have adapters for any specific protocol or method of communication used by different server components. An example can be a REST-SOAP

Adapter, which receives a SOAP request and adapts it to a REST-service format.

- *Native networking API.* This component is the one responsible for actually communicating with client devices through the establishment of end-to-end connections. Typically, reliable Transmission Control Protocol (TCP) connections are established.

2.3 Analysis of existing CrowdSensing proposals

In this section, we provide an analysis of the different solutions available in the literature, using the architecture proposed in section 2.2 as a reference for our classification. For our study, we focused on research works published in the crowdsensing field during the past five years, with a special emphasis on smartphone-based crowdsensing solutions.

For the sake of clearness and completeness, our analysis was split into four well-defined parts: (1) general analysis, (2) client-side analysis, (3) server-side analysis, and (4) data delivery approaches. The first part presents a general analysis of an extensive set of the various proposals and performs a synthesis of the different contributions in the scope of our architecture. In the second part we have addressed in more detail those proposals detailing a client-side architecture, that is, describing the Client User Interface (CUI), CDM, and MSC components, while for the third one we detail server-side architectures, describing the SIM, STM, and SDM components. Also, we have assessed to what degree the different solutions can provide all the functionalities envisioned in our proposed architecture. It is worth highlighting that both client and server analysis include not only proposals specific to client/server sides but also global solutions whenever they provide details about all the elements involved in the end-to-end interaction.

Finally, we have classified those solutions by providing details about the communications system defined for clients to server interactions, and also about supported topology, selected technology, and other relevant features. Again, for this data delivery analysis, any proposal providing enough details was included, no matter how broad or how specific was the proposal itself.

2.3.1 General analysis

In our general analysis of the different crowdsensing solutions, we have classified information based on three parts. In the first one we provide generic information about the different proposals, in the second one we describe aspects related to security/privacy and energy consumption, and finally we provide a summary of contributions for each proposed architecture. This classification and characterization is presented in Table 2.2.

General features

With regard to the *general features*, we found that the vast majority of solutions propose an integral solution to the sensing tasks at both client and server sides. Other solutions propose a specific middleware to help in the tasks of data collection and processing.

Concerning the strategy adopted for data collection, most proposals opted for a participatory approach for data sensing where users are fully aware of the data collection process, and they actively participate in that process. Other approaches, however, prefer using opportunistic systems that operate in a more autonomous manner, gathering information in the background at appropriate times; finally, a few proposals combine these two approaches to achieve a more complete functionality.

Regarding the target applications addressed in the different works, we found that the majority of the proposals are flexible enough to embrace heterogeneous applications, i.e., they can adapt to generic sensing tasks, although we can also find proposals that are specific to transportation and urban sensing environments, and to a lesser extent to health, social, and other environments.

Finally, with respect to the number of differentiated elements defined for each proposed architecture, we found that there are significant differences among authors. For instance, [128, 55, 84] split their proposed functionality into four different levels, similarly to our proposal. In particular, NoizCrowd [128] defines an architecture based on four components which are: Data Gathering, Data Storage, Noise Modeling, and Data analytics/visualization. SmartCity [55] also defines a four-element architecture composed of Social Networks, Ubiquitous Sensors, a Mobile Context-Aware Platform, and the Cloud Platform. MCSaaS [84] defines four core sub-modules, namely Cloud Broker, Orchestrator, Customization Service, and Deployment Manager. In general, most proposed architectures only defined two or three levels, as is the case of [117], which defines a generic publish-subscribe communication with three roles (producers, services providers, and consumers), along with Analytics Components.

Privacy and Energy issues

In general, the success of mobile crowdsensing applications is dependent on how each solution addresses user concerns about their his/her own privacy. In the context of mobile solutions for measuring environmental noise, an example could be a solution where noise is captured raw, and forwarded to a server. Another problem is the personal information when reading any sensor, damaging the user's privacy in the same action. The users' mobile energy consumption is another critical issue, as most users will reject applications draining a significant amount of battery power. So, both energy and privacy issues are relevant in the scope of crowdsensing solutions, a reason why different researchers have addressed them.

Our analysis has shown that most studied proposals have addressed energy efficiency issues, while only some of them have introduced mechanisms to mitigate security and privacy concerns. In fact, we find that only a few solutions [38, 113, 2, 83, 134] actually account for both privacy and energy efficiency issues. We now proceed to discuss these prominent solutions in more detail.

PRISM [38] supports privacy through a registration process on a PRISM server for each enabled terminal. The registration is maintained by software and it expires within a given period of time. When the registration period expires, terminals wait for a random time and proceed to register again. Concerning energy consumption, PRISM maintains a control of energy consumption on mobile phones through its prism sandbox, which is able to perform coarse-grain power monitoring.

Anonysense [113] uses a server that is responsible for registering and authorizing mobile phones. During registration, Anonysense installs its software along with the Internet Protocols (IP) addresses and certificates for its task service and report service. Concerning energy consumption, the tasks can be divided into two sub-operations: sensing and signing. In the first, the RogueFinder application is used to detect rogue Access Points (APs) in a given area, while the ObjectFinder application attempts to find a specific Bluetooth Media Access Control (MAC) address. The second group addresses whether a data report contains sensitive data. Additionally, it estimates the energy cost associated to these operations.

Usense [2] includes a component for securing communications. Additionally, it manages user preferences in terms of resource and privacy restrictions. These features are processed through the sensing agent, which is an application deployed on the device itself. In addition, Usense's middleware is able to save energy using a mechanism that avoids taking measurements in those areas where it already has enough data, or when the phenomenon is mostly invariant.

SenSocial [83] has a module for privacy management control which allows managing policies regarding the type and level of granularity of sensed data, deciding what will be stored and made available to the different middleware components. SenSocial uses filtering rules for maintaining energy efficiency, thereby restricting transmissions only to those cases passing the set of defined rules. Also, SenSocial discriminates the energy consumption associated to the accelerometer sensors, microphone, GPS, Bluetooth, and WiFi. Similar works [120, 91, 150] authors propose to reduce energy consumption by focusing on optimizing the use of GPS.

The last proposal in this group is Anonymity [134], which proposes an anonymous data reporting protocol for participatory applications. The idea is that the protocol avoids including identification information that can be vulnerable. The anonymous data protocol is divided into two stages: the first is a slot reservation stage (scheme based on public key encryption), while the second one is a data submission stage (scheme based on an XOR operation). Through comparison against a similar study, authors show how it is able to improve data submission performance. With regard to energy consumption, the smartphone's battery values are measured using a multimeter. It also presents an analysis of the energy overhead associated to data submission.

Overview of the contributions for each proposal

The main goal of this section is to assess the actual contributions made by the different authors taking as reference the architecture proposed in section 2.2. So, the last part of table 2.2 (columns MSC, Transmit (Tx), and CDCS) provides a first insight into the actual contribution made by the different components at the client (MSC) and server (CDCS) sides, in addition to the end-to-end transmission process itself (Tx).

We provide a three-level classification of proposals, where a dark star means that the particular solution fulfills the expected functionality for that component, while a white star means that the solution only provides a partial fulfillment of the selected characteristics. The non-fulfillment of the characteristics of a component is represented by the absence of any star.

Overall, we can observe that the majority of the proposals are quite representative in the scope of our architecture, providing most of the expected functionalities. Nevertheless, we can also find solutions such as MOSDEM [97] and SenseDroid [109] that focus mostly on MSC-related functionality. Similarly, we can find solutions such as MCSaaS [84] that focus on the CDCS instead.

Table 2.2: Classification of the different crowdsensing proposals technologies according to the proposed architecture.

Publication	Year	Proposal	Type	Recollection strategy	Target	# level	Priva-cy	Ener-gy	MSC				TX	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
PRISM [38]	2010	Platform	Prototype	Both	Heterogeneous	3	X	X	★	☆	☆	☆		☆	☆	☆	☆
Ear-Phone [105]	2010	Client-Server	Real/ Simulations	Participatory	Environment monitoring	2		X	☆	☆	☆	☆			★	☆	☆
AnonySense [113]	2011	Framework	Prototype	Opportunistic	Environment monitoring	3	X	X	☆	☆	☆	★		★	★	★	★
Medusa [103]	2012	Framework	Prototype	Participatory	Heterogeneous	2	X		★	☆	☆	★		★	☆	★	☆
Pogo [20]	2012	Middleware	Real imp.	Participatory	Heterogeneous	2	X	X	☆	☆	☆	★		★	☆	☆	☆
DAM4GSN [96]	2012	Architecture	Real imp.	Participatory	Heterogeneous	3		X	★	☆	★	★		★	★	☆	☆
MECA [135]	2012	Middleware	Prototype	Participatory	Heterogeneous	3				☆	☆	☆		★	★	★	☆
StressSense [78]	2012	Framework	Real imp.	Participatory	Healthcare	1			★	☆	☆	☆		☆	★	☆	☆
CrowdITS [6]	2012	Framework	Real imp.	Participatory	Transportation and Urban Sensing	3		X	★	☆	☆	★		★	★	★	★
SmartCity [117]	2013	Framework	Real imp.	Participatory	Transportation and Urban Sensing	4			☆	☆	☆	☆		★	★	☆	☆
ILR [118]	2013	Scheme	Simulations / Real imp.	Participatory	Location Services	3	X		☆	☆	☆	☆		★	☆	☆	☆
CAROM [146]	2013	Framework	Real imp.	Participatory	Heterogeneous	3		X	★	☆	★	☆		☆	★	☆	★
SoundOfTheCity [108]	2013	Client-Server	Real imp.	Both	Environment monitoring	2		X	★	☆	☆	☆		☆	★	☆	☆
MoPS [145]	2013	Middleware	Prototype / Real imp.	Both	Environment monitoring	3		X	★	★	☆	★		☆	☆	★	☆
NoiseNYC [155]	2013	Framework	Real imp.	Participatory	Healthcare	3				☆			☆	★	★	☆	☆
Vita [54]	2013	System	Real imp.	Participatory	Healthcare	2		X	★	☆	☆	★	☆	★	☆	☆	☆
Matador [27]	2013	Framework	Simulations	Both	Location Services	-		X	☆	☆	☆	★	☆	★	☆	★	★

Continued on Next Page...

Table 2.2 : Continued from previous page

Publication	Year	Proposal	Type	Recollection strategy	Target	# level	Privacy	Energy	MSC				TX	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
Usense [2]	2013	Middleware	Real imp.	Both	Heterogeneous	3	X	X	★	★	☆	★		★	☆		☆
REPSense [75]	2013	Applications	Real imp.	Opportunistic	Environment monitoring	2			☆	☆	☆	☆				☆	
BeC3 [31]	2013	System	Real imp.	Opportunistic	Heterogeneous	3		X	★	☆	★	★		★	☆	★	★
McSenseFoster [26]	2013	System	Real imp.	Participatory	Transportation and Urban Sensing	3			☆	☆	☆	☆	☆	☆	☆	☆	★
NoizCrowd [128]	2013	Components	Real imp.	Participatory	Healthcare	4			☆	☆	☆	☆		★	★	☆	☆
PLUS [67]	2014	Framework	Real imp.	Participatory	Transportation and Urban Sensing	2		X	★	☆	☆	☆		☆	☆	☆	☆
MOSDEN [97]	2014	Middleware	Real imp.	Participatory	Heterogeneous	3		X	★	★	★	★		★	☆	★	☆
SenseDroid [109]	2014	Framework	Prototype/Re imp.	Both	Location Services	3		X	★	★	★	☆			★	☆	
SenSocial [83]	2014	Middleware	Real imp.	Both	Social recom-mendation	3	X	X	★	★	☆	★		★	★	★	☆
AlienVsMobile [118]	2014	Client-Server	Prototype/Re imp.	Participatory	Location Services	3	X		★	☆	☆	☆		☆	☆	☆	☆
SaaS [112]	2014	Framework	Description	Participatory	Heterogeneous	3				☆		☆		★	★	☆	☆
CUPUS [11]	2014	Middleware	Prototype/Re imp.	Both	Heterogeneous	3		X	★	☆	★	☆	☆	☆	☆	☆	☆
BLISS [11]	2014	Framework	Simulations	Participatory	Social-Budget	-			☆	☆	☆	☆				☆	☆
GPS-Less [131]	2014	System	Real imp./Simulati	Both	Transportation and Urban Sensing	2		X	☆	☆	☆	☆		★	☆	★	☆
MCS-Space [92]	2014	System	Real imp.	Participatory	Social-General	3		X	☆	☆	☆	☆		★	★	★	☆
SPREAD [57]	2014	Algorithm	Prototype	Participatory	Location Services	-			☆	☆	☆	★				☆	☆

Continued on Next Page. . .

Table 2.2 : Continued from previous page

Publication	Year	Proposal	Type	Recollection strategy	Target	# level	Privacy	Energy	MSC				TX	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
Map++ [9]	2014	System	Real imp.	Participatory	Transportation and Urban Sensing	-		X	☆	☆	☆	☆			☆	☆	☆
JoiPolices [10]	2014	System	Simulations	Participatory	Heterogeneous	3	X		☆	☆	☆	☆		★	☆	☆	☆
CrowdRecruiter [149]	2014	Framework	Real imp.	Both	Location Services	-		X		☆		☆		☆	☆	★	☆
Neighbor [51]	2014	Middleware	Simulations	Opportunistic	Heterogeneous	-		X		☆	☆	☆	★	☆		☆	☆
LineKing [21]	2014	System	Real imp.	Both	Social recommendation	2		X	★	☆	☆	☆		★	☆	☆	☆
GROPING [147]	2014	Application/F	Prototype	Participatory	Location Services	2		X	★	☆	☆	☆		☆	★	★	☆
EasyHarvest [65]	2014	Framework	Prototype	Opportunistic	Heterogeneous	2			★	☆	☆	★		★		☆	☆
WiFIScout [129]	2014	System	Real imp.	Participatory	Social recommendation	2			☆	☆	☆	☆		☆	☆		
QoS-Constrained [127]	2014	Framework	Simulations	Both	Heterogeneous	2					☆	☆		☆	☆	★	
Ecosystem [55]	2015	System	Real imp.	Participatory	Social General	4		X	★	☆	☆	★		★	☆	★	☆
MCSaaS [84]	2015	Framework	Real imp.	Both	Heterogeneous	4		X	☆	☆	★	★		★	★	★	★
COUPON [153]	2015	Framework	Simulations	Opportunistic	Heterogeneous	-		X	☆	☆	☆	☆	★	☆			
ADTS [32]	2015	Application	Simulations	Participatory	Location Services	1			☆	☆	☆	☆					
Anonymity [134]	2015	Framework	Prototype/Re imp.	Participatory	Transportation and Urban Sensing	2	X	X	☆	☆	☆	☆	★		☆	☆	☆
TYT [100]	2015	Framework	Real imp.	Participatory	Healthcare	3	X		★	☆	☆	☆		☆	☆	☆	★
QOATA [156]	2015	Application	Simulations	Participatory	Heterogeneous	-				☆	☆	☆					
NeCoRPIA [45]	2015	Protocols	Simulations	Opportunistic	Heterogeneous	-				☆		☆	★	☆		☆	
QoSMCS [41]	2015	Framework	Simulations	Opportunistic	Heterogeneous	3		X		☆	☆	☆	★	☆	★	☆	☆

Continued on Next Page. . .

Table 2.2 : Continued from previous page

Publication	Year	Proposal	Type	Recollection strategy	Target	# level	Privacy	Energy	MSC				TX	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
FlierMeet [46]	2015	Framework	Real imp.	Participatory	Transportation and Urban Sensing	3			★	☆	☆	☆		☆	☆	☆	☆
PRESM [130]	2015	Scheme	Real imp.	Participatory	Transportation and Urban Sensing	3	X		★	☆	☆	☆		☆	☆	★	☆
SmartRoad [53]	2015	Framework	Real imp.	Participatory	Transportation and Urban Sensing	2		X	★	☆	★	★		★	★	★	☆
MCSgame [132]	2015	System	Simulations	Participatory	Location Services	2		X		☆	☆	☆		★	☆	☆	☆
RemoteCloud [115]	2015	-	Simulations	Participatory	Transportation and Urban Sensing	-					☆	☆	★		☆	☆	☆
MDPPs [110]	2015	Middleware	Real imp.	Opportunistic	Location Services	3			★	☆	☆	☆		★	★	★	☆
RuPS [85]	2015	Framework	Real imp.	Participatory	Location Services	4			☆	☆	☆	☆		☆	★	☆	★
EffSense [126]	2015	Framework	Simulations	Opportunistic	Heterogeneous	-		X		☆	☆	☆	★		☆	☆	
PLP [151]	2015	Scheme	Simulations	Opportunistic	Heterogeneous	-	X			☆	☆			★	★	☆	
Sahyog [13]	2015	Middleware	Real imp.	Participatory	Heterogeneous	3	X		★	☆	☆	★		☆		☆	
Context [89]	2015	Middleware	Simulations	Opportunistic	Social recommendation	-		X		☆	☆	☆	★		☆	☆	
MoreWithLess[133]	2015	Framework	Real imp.	Participatory	Heterogeneous	4		X		☆					★	★	
Sparse[125]	2016	Framework	Real imp.	Participatory	Heterogeneous	3	X			☆					★	★	

2.3.2 Client-side analysis

In this section, we focus on the specific contributions to the MSC, which is the client side of our proposed architecture. To achieve it, in Table 2.3 we describe the features of the different proposals regarding the Client Interface Manager (CIM), the Client Data Manager (CDM), and Client Sensor Manager (CSM). Notice that we excluded the Client Communications Manager (CCM) from this section, as it will be addressed separately in section 2.3.4. Also notice that the table is split into two sections, being that proposals in the upper section are client-specific, meaning that the publication only describes the client side of the crowdsensing architecture, while proposals in the bottom section describe both client and server sides.

Concerning the CIM, we found that most of the solutions provide a graphical user interface designed for the Android operating system, thus typically adopting the Java language for development. In fact, only a few solutions such as LineKing [21], TYT [100], and DAM4GSN [96] focused on other operating systems. Also, most of the proposals allow the user to have access to an administrative interface in order to have control over sensing tasks.

With regard to the CDM, we observe that, in general, most available solutions resort to plugins or external libraries in order to simplify their processing, query and storage task on the device by reusing existing software. In particular, different techniques and algorithms are adopted mostly to support the data collection procedure including spatiotemporal area calculation and programming algorithms, among others. The spatiotemporal coverage of an area refers to the amount of time and space needed to properly sensorize that area according to the target task, being Usense [2] one of the most used. Also, we found that, although several solutions provide data analytics within the mobile device itself, such functionality is seldom combined with the use of plugins.

Among solutions integrating plugins, we would like to highlight solutions such as DAM4GSN [96], MOSDEN [97], and CAROM [146], that use open source GSN technologies for Internet of Things (IoT). In particular, CAROM [146] uses a plugin where, among other functionalities, it incorporates Open Mobile Miner (WMO), which is an open-source solution that allows performing data analysis on the mobile terminal. Similarly, SenSocial [83] uses a plugin providing an agent able to retrieve data from both Facebook and Twitter, and its process is based on joining Online Social Networks (OSNs) that provide a physical context data stream. In addition, we found that few solutions include a broker functionality. We also found that there is a balance between the approaches preferring pushing contents onto the servers, and solutions that prefer the server to pull contents instead.

With regard to data processing, we find that few solutions perform aggregation-fusion on the mobile device, as opposed to data filtering, whose support is quite common. Finally, with regard to the Client Sensor Manager, in general the different proposals available make use of generic sensors that are internal to the mobile devices, offering in few cases support for external sensors. There is also evidence of applications using external sensors or multimedia stream processing before sending the streams to the server (see, e.g., StressSense [78], and REPSense [75]).

Finally, regarding the adoption of virtual sensors, only a minority of the proposals studied do so. In particular, options such as DAM4GSN [96], MOSDEN [97], and CAROM [146] relied on an adapted version of GSN [71], while other proposals like SenseDroid [109], CUPUS [11], and SmartRoad [53] provide their own virtualization solutions.

Table 2.3: Classification of the different crowdsensing proposals according to the proposed client architecture.

	Client Interface Manager(CIM)			Client Data Manager (CDM)							Client Sensor Manager (CSM)	
				Plugins / external libraries					Dat. Process. and Store			
Publication	Interface / admin.	O.S.	Technology	Characteristic	Real time	Sensing task	Data analytics	Broker Plug-in	Filter / Aggregate	Query / Local store	Type of sensor	Virtual sensor / Preprocessing
PRISM[38]	Y/-	Windows Mobile	C# and C++	PRISM's sandbox	Sent	Push-Pull				Y/Y	Generic/ Multimedia	-
DAM4GSN[96]	Y/Y	Android/ IOS	Java	XML-Based/GSN	store and send	Push-Pull		both	Y	Y/Y	Generic and external	Y/Y
StressSense[78]	Y/-	Android	Java, C and C++	GMMs and Simulate	send	Pull		plug-in	-	-/Y	External microphone	-/Y
Usense[2]	Y/Y	Android	Java-BlueZen	XML-Based/spatiotemp	store and send	Push		plug-in	Y/Y	Y/Y	Generic/Mult	-/Y
REPSense[75]	Y/-	Android		Scheme divide/merge	store and send				Y/-	Y/Y	GPS, ambient light, air pressure and accelerometer	-/Y
PLUS[67]	-/Y	Android		Markov Predictor Model	store and send		X		Y/-	-/Y	GPS	-/Y
MOSDEN[97]	Y/Y	Android	Java	XML-Based/GSN	store and send	Push-Pull	X	both	Y/-	Y/Y	Generic and external	Y/Y
SenseDroid[109]	-	Android		Spatiotemporal	store and send			broker	Y/Y	-/Y	Generic and external	Y/Y
AlienvsMobile[11	Y/-	Android	Java	Area coverage	send	Pull			-	-	GPS, barometric pressure	-/Y
CUPUS[11]	Y/-	Android		CUPUS MIOs	store and send	Push-Pull			Y/-	Y/Y	Generic and external	Y/Y
BLISS[49]	-		Simulated	Simulated/Online learning	Send		X		Y/-	Y/-	N/A Simulated	-
SPREAD[57]	-		Simulated	Simulated/area of interest	Send				Y/-	Y/-	GPS and Simulated	-
MAP+[9]	Y/-	Android		DBSCAN	store and send	Pull			Y/-	Y/-	External GPS	-/Y

Continued on Next Page...

Table 2.3: Continued from previous page

Publication	Client Interface Manager(CIM)			Client Data Manager (CDM)							Client Sensor Manager (CSM)	
				Plugins / external libraries					Dat. Process. and Store			
	Interface / admin.	O.S.	Technology	Characteristic	Real time	Sensing task	Data analytics	Broker Plug-in	Filter / Aggregate	Query / Local store	Type of sensor	Virtual sensor / Preprocessing
ADTS[32]	-		Simulated	ADTS Algorithm	send		X		Y/-	-	Generic	-
FlierMeet[46]	Y/-	Android		STA Grouping	send	Pull			-	-	GPS, light sensor, accelerometer, magnetometer	-/Y
MDPPs [110]	-/Y	Android		Spatiotemporal coverage (MDPPs)	stored and sent	Push	X		-	Y/Y	Generic Sensor	-
EasyHarvest[65]	Y/Y	Android	Java for Android	Spatiotemporal coverage	store and send	Push Binarios	X		Y/-	Y/Y	Generic Sensor	-
Sahyog[13]	Y/-	Android	Java	Query format	store and send	Push-Pull			-	Y/Y	GPS, Accelerometer	-
WiFIScout[129]	Y/Y	Android		Read Wifi	store and send	Pull			Y/-	-/Y	GPS/WiFi	-
Ear-Phone[105]	Y/-	Symbian	Java	Spatiotemporal coverage	Send	Pull			Y/-	Y/Y	GPS Microphone	-/Y
Medusa[103]	-/Y	Android	Java SMS and MMS	MedScript XML	send	Push	X	broker	Y/-	-/Y	GPS, Multimedia	-/Y
CrowITS[6]	Y/Y	Android / IOS		Pluggin-Based	store and send	Push-Pull		plug-in	-	Y/-	GPS	-
CAROM[146]	Y/Y	Android	Java/GSNLite	XML-Based/GSN	send	Push-Pull	X	plug-in (OMM)	-/Y	Y/Y	Generic and external sensor	Y/Y
SoundOfTheCity[	Y/Y	Android	Java		store and send	Pull			-	Y/Y	GPS, Microphone	-/Y
MoPS [145]	Y/Y	Android	Java	Broker Mios		Push	X	broker	Y/-	Y/-	External Pollutions	-Y
Vita[54]	Y/Y	Android	Java	XML-Based	send	Push		broker	Y/-	Y/Y	Generic and Multimedia	Y/-
Matador[27]	Y/Y	Android		XML-Based Spatiotemporal	send	Pull	X		-/Y	-/Y	GPS	-/Y

Continued on Next Page...

Table 2.3: Continued from previous page

Publication	Client Interface Manager(CIM)			Client Data Manager (CDM)							Client Sensor Manager (CSM)	
				Plugins / external libraries					Dat. Process. and Store			
	Interface / admin.	O.S.	Technology	Characteristic	Real time	Sensing task	Data analytics	Broker Plug-in	Filter / Aggregate	Query / Local store	Type of sensor	Virtual sensor / Preprocessing
SenSocial[83]	Y/-	Android	Java	XML-Based/Pluggin OSN	send	Push-Pull	X	both	Y/Y	Y/Y	GPS, Accelerometer, microphone, Social	-
MCSinSpace[92]	Y/Y	Android			send	Push		X		Y/Y	GPS signal anlysis	X
LineKing[21]	Y/-	Android/ IOS		Wait Time Algorithm	send	Pull	X		Y/-	Y/Y	GPS, Accelerometer	-
Ecosystem[55]	Y/Y	Android	Android App	XML-Based	send	Push	X		Y/-	-/Y	Generic and external sensor	-/Y
TYT[100]	Y/Y	Android/ IOS	Android and IOS Apps		send	Pull			-	Y/Y	Heart Rate, Blood Pressure, Oxygen Saturation	-
PRESM[130]	-	Android	RSS Map	Map generation	store and send	Push	X	X		-	GPS Locations	-/Y
SmartRoad[53]	Y/Y	Android	Java	Detection and identification Learning Algorithm	store and send	Push		plug-in	-	Y/-	GPS, Power Sensor	Y/Y

2.3.3 Server-side analysis

In this section, we will focus instead on the server side, which in the scope of our proposed architecture takes the name Cloud Data Colletion Server (CDCS).

Table 2.4 describes the features of server-related proposals. Similarly to the previous section, the table is split into two parts, being that proposals in the upper part are server-specific (the publication only describes the server side of the crowdsensing architecture), while proposals at the bottom section are complete ones, describing both client and server sides; obviously, since client-related details were already presented above, in this section we only focus on server-related issues.

In general, we observe that most of the proposals provide a web interface for management and result presentation purposes, and most of them also provide data management and data sharing functionalities. The technologies used in these proposals are generally open-source solutions such as Apache, Java, and Hypertext Pre-Processor (PHP), among others, and many of them use a database manager such as MySql and PostgreSQL. Also, there is evidence that many proposals rely on a cloud infrastructure provided by Amazon [35] or Google [36].

With respect to the Server Task Manager, we find that most proposals present mechanisms to manage and deploy sensing tasks. In particular, in terms of task deployment, we find that the number of proposals adopting a push-based approach is similar to those adopting a pull-based approach.

Regarding to the language used for task definition, some solutions describe tasks using specific algorithms, while others prefer using a programming language, as is the case of Pogo [20], AnonySense [113], and Medusa [103].

With respect to Data Management at the server, most solutions perform data aggregation similarly to client-side solutions. Some of them use intelligent data analysis techniques such as Big data [145][117][84], decision making (acMCDM) [85], and three-dimensional analysis Poker Flat Incoherent Scatter Radar (PFISR) [92], and various other statistical tools. Recent research works [124, 125, 133] take advantage of the space and the time correlation between the discovered data of different sub-areas with the aim of reducing the number of tasks required for the target purposes. Wang et al. [124, 125] present a solution called sparse MCS framework that uses inference algorithms to ensure the quality of the data after being collected. Instead, Xu et al. [133] describe a framework that uses four states (data structure conversion, base training, sampling, and reconstruction). It relies on programming algorithms to create a baseline dataset using the K-SVD algorithm, while for the reconstruction the Orthogonal Matching Pursuit recovery algorithm is adopted. In both cases, the intention is to produce a global saving on detection costs (power consumption, and network resources) while ensuring the overall data quality.

As output, data can be presented in different formats, being the use of HeatMaps a representative example when sensing information is geolocated.

Table 2.4: Classification of the different crowdsensing proposal according to the proposed server architecture.

Publication	Server Interface Manager(SIM)			Server Task Manager (STM)					Server Data Manager (SDM)				
	Interface	Manager / Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing		Query and analysis	Database
										Filter	Aggre.		
AnonySense [11	Web	X/X	Ruby and small HTTP Severs	AnonyTL /Ruby	Authentica servers		X	Push/Pul	Privacy distribution	X		Query	SQLite3
Pogo [20]	Web	X/SE	JavaScript - Openfire XMPP Pub/Sub	Scripts-Rhino	Generic		X	Push	Multibroker	X	X	Query	SQL
MECA [135]	Web	SE/X		Edge Task	Generic	X		Pull	Analytics Library and Multibroker		X	Query and analysis	SN and Device
Smarctity [117]	Web	SE/X	XMPP Pub/Sub		Learning algorithm	X	X	Push	BigData, datamining Casandra/s4	X	X	Query and analysis	NoSQL
ILR [118]	Web	X	Java/J2EE and Glassfish Application Server	ILR scheme	Location reliability algorithm			Push	Simulations NS2 Real Traces	X		Query and analysis	Derby
NoiseNYC [155]					Generic				3D Tensor Kriging, heatmaps and others		X	Query and Analysis	S/N and Device
BeC3 [31]	Web	X/X	C, Java/XMPP Pub/Sub, D-LITe Cloud	D-Lite/Python BPEL WS-CDL	Generic	X	X	Push/Pul			X	Query	S/N and Device
McSense [26]	Web	X/X	Java Servlet Ajax Web - Apache Spring		Generic	X	X	Push		X		Query and Analysis	PostgreSQL
SaaS [112]	Web	X/X			Generic	X	X	Push		X	X	Query and Analysis	S/N and Device
GPS-less [131]	Web		JavaScript		Coverage model algorithm		X	Push			X	Query and Analysis	S/N and PostgreSQL

Continued on Next Page. . .

Table 2.4 – Continued from previous page

Publication	Server Interface Manager(SIM)			Server Task Manager (STM)					Server Data Manager (SDM)				
	Interface	Manager / Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing Filter	Data processing Aggre.	Query and analysis	Database
JoinPolices [10]					Budget efficiency algorithm			Push	MatLab		X	Query and Analysis	
MCSaaS [84]	Web	X/X	Java, Python / Apache Tomcat. Google Cloud Messagin (GCM)	XML Task	Generic	X	X	Pull	Cloud Broker Big data	X	X	Query and Analysis	S/N and Device
GROPING [147			Amazon Mechanical Turk(AMT)		Location estimation algorithm	X		Pull		X	X	Query and Analysis	S/N
Qoata [156]				QOATA scheme Simulated	Online learning and task allocation algorithm						X		
MCS game [132]					Q-Learning algorithm	X		Push	Nash equilibrium (NE)			Query	
RuPS [85]	Web	X/X			Generic			Pull	Multi-Criteria Decision Making (MCDM)		X	Query and Analysis	S/N and Device
NoizCrowd [128		SE/X			Spatial and temporal algorithm	X		Pull	Bigdata		X	Query and Analysis	SciDB and SPARQL
QoS-Constrained[127					Spatial temporal coverage			Pull	QoS:Min, Max, Utility	X	X		
CrowRecruiter[					Spatial temporal coverage		X	Pull		X	X		S/N
PLP [151]					Learning algorithm			Pull	RSS fingerprin	X	X	Query and Analysis	S/N

Continued on Next Page...

Table 2.4 – Continued from previous page

Publication	Server Interface Manager(SIM)			Server Task Manager (STM)						Server Data Manager (SDM)			
	Interface	Manager / Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing Filter	Data processing Aggre.	Query and analysis	Database
MoreWithLess[1					Spatial temporal correla-tions		X	Pull		X	X	Query and Analysis	
Sparse[125]					Spatial temporal correla-tions			Pull		X	X	Query and Analysis	
Ear-Phone [105]	Mobile		PHP script and Java		Generic			Pull	GPS MGRS converter		X	Query	MySQL
Medusa [103]	Web		Apache, PHP, Java and AMT	MedScript/P	Generic	X		Push	Stage Libray	X		Query	MySQL and device
CrowITS [6]	Web	X/X	Google C2DM and PHP-Google Maps Api		Generic	X	X	Push/Pul		X	X	Query and Analysis	MySQL and device
CAROM [146]	Web	X/X	AMT EC2 and GSN	Task-XML/GSN	Generic	X	X	Push/Pul	Data Mining/ FSI fussy	X	X	Query and Analysis	S/N and device
SoundOfTheCit	Web	SE/X	JavaScript		Generic			Pull	Media Streaming and encoding		X	Query and Analysis	MySQL
MoPS [145]	Web		Java and Google Cloud Messaging (GCM) Pub/Sub		Generic	X	X	Push/Pul	Data Mining		X	Query	S/N
Vita [54]	Web	X/X	Apache Tomcat and JBoss and (AWS) EC2 AND S3	Task BPEL/ BPEL4Peopl : ODE and jBPM	Genetic algorithm and K-means	X	X	Push			X	Query	S/N
Matador [27]	Web	X/X		Matador Task	Spatial temporal algorithm		X	Pull	Adaptive sampling algorithm		X	Query	S/N
SenSocial [83]	Web	SE/X	PHP script/ Pub-Sub (MQTT)	XML	Task OSN		X	Push/Pul	OSN and Filter Aggregated	X		Query and Analysis	MongoDB and device

Continued on Next Page…

Table 2.4 – Continued from previous page

Publication	Server Interface Manager(SIM)			Server Task Manager (STM)					Server Data Manager (SDM)				
	Interface	Manager / Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing Filter	Data processing Aggre.	Query and analysis	Database
MCSinSpace [92	Web	X/SE	Mapping and tomographic reconstruction algorithms		Generic	X	X	Push	PFISR, kalman filter, interpolates	X	X	Query and Analysis	S/N
LineKing [21]	Web		Apache HTTP and AWS EC2		Generic	X	X	Pull	Wait-Time estimator	X		Query	MySQL and device
Ecosystem [55]	Web	X/X	Amazon EC2 M1. Ubuntu, Apache TomCat	Task BPEL XML	Generic	X	X	Push/Pul		X	X	Query and Anality	S/N
TYT [100]	Web	X/X	PHP Lavarel		Generic	X		Pull	Algorithm TYT		X	Query	MySQL
PRESM [130]	Web	SE/X			Algorithm CS-Based and Rss		X	Push	RSS Map Generation	X		Query	S/N
SmartRoad [53]	Web		Java, Django, and Python plugin/Google Maps		Learning algorithm		X	Push	HeatMap, 3DView		X	Query and Anality	MySQL and device

2.3.4 Data communications issues

We conclude our analysis of the current crowdsensing literature by focusing on client-server communication solutions. Notice that, since communications simultaneously involve clients and servers, we address communication issues jointly in this section.

Table 2.5 summarizes the main communication characteristics associated to the different proposals. We have also surveyed the metrics used by each proposal for performance analysis, and classified them according to their scope as: generic, Quality of Service (QoS), and scalability. As *generic* performance metrics we refer to those proposals addressing network performance in terms of packet delivery ratio, end-to-end delay, and transmission overhead, among others. *QoS* issues are associated to data acquisition, and they attempt to avoid that the delivery of massive data (data without processing) directly from the source negatively impacts network traffic and the energy consumption of mobile devices. Concerning *scalability*, authors assess the capability of the infrastructure in terms of adaptability to an increasing number of sensing tasks and terminals to determine if it is able to adapt to both small and large deployments. Under the scalability concept we have also considered the elasticity of these services (middleware) to manage changes.

Notice that Table 2.5 is clustered into four different parts according to the scope of the proposal: T refers to those proposals only addressing transmission issues, C refers to proposals centered on the client side, S refers to proposals centered on the server side, and G refers to global solutions.

Concerning communication technologies used, a large number of proposals relied on WiFi and Cellular communications, although we can also find proposals that rely instead on Bluetooth due to its flexibility and low consumption features. Additionally, we find that most solutions opted for either a centralized topology or a distributed topology, with only a reduced number of proposals choosing a hybrid approach. Regarding the networking approach, most solutions adopt RESTful services based on Hypertext Transfer Protocol (HTTP), or make use of the XML format.

Focusing now on the performance metrics addressed by each proposal, most solutions made a generic performance analysis (delivery delay, data rate, etc.). However, very few solutions addressed QoS and scalability issues. For instance, we can find solutions such as Google Cloud Messaging (GCM) [6] that address scalable services in the cloud, others that address scalability in the context of the Publish/Subscriber paradigm [145], and yet others that relate it to broker collaboration [109], but none of these actually assess performance in the scalability context.

Regarding proposals evaluating Quality of Service performance, they typically perform such evaluation in terms of task allocation and coverage optimization in the target area. For instance, proposals such as JoinPolices [10] evaluate the impact and the performance of task execution based on incentive policies, while

QoSMCS [41] defines an ad-hoc method for the evaluation of QoS in the context of mobile crowdsensing services based on Petri networks.

Finally, regarding scalability, solutions such as PRISM [38] assess the performance achieved through comparison against other solutions. Neighbor [51] measures message diffusion performance between the mobile nodes and the data collection server. Lastly, Medusa [103] proposes a prototype able to measure, at runtime, the time taken to perform several individual steps associated to task executions, both on the cloud and the smartphone.

Table 2.5: Classification of the different data transmission solutions according to the proposed architecture.

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols / Format	Performance Metrics		
						Generic	QoS	Scalability
Neighbor [51]		WiFi/Bluetooth/ Cellular	Hybrid	Algorithm/One simulated		X		X
Coupon [153]		WiFi/Bluetooth	Hybrid	ERF and BSWF	TCP	X		
Anonymity [134]	T^1	WiFi	Centralized	Slot reservations stage and data submission stage	TCP	X		
NeCorpia [45]		WiFi	Distributed-Hybrid	Network Encoding Format /Gaussian Eliminations	TCP	X		
QoSMCS [41]			Hybrid	Markovian Stochastic Petri NeT	HTTP-TCP		X	
RemoteCloud [115]		WiFi/Cellular	Distributed	MANET or Device-to-Device (D2D)	TCP	X		
EffSense [126]		Bluetooth/WiFi/ Cellular	Hybrid	Publish-Subscriber	HTTP-JSON	X		
Context [89]		Bluetooth	Hybrid	simulated	TCP	X		
PRISM [38]		WiFi/Bluetooth/ Cellular	Centralized	PRISM client and Sandbox				X
DAM4GSN [96]		WiFi/Cellular	Hybrid	GSN (SOAP and RESTful Web Services)	HTTP-XML			
StressSense [78]		WiFi/Cellular	Distributed	MatLab				
Usense [2]		WiFi/Cellular	Distributed	SOAP Web Services (RPC)	HTTP-XML	X		
REPSense [75]		WiFi/Cellular	Distributed					
PLUS [67]		WiFi	Centralized	Web Services				
MOSDEN [97]	C^2	WiFi/Cellular	Distributed	GSN (SOAP and RESTful Web Services)	HTTP-XML	X		
SenseDroid [109]		WiFi/Bluetooth/ Cellular	Hybrid	Provides libraries and APIs				
AlienvsMobile [119]		WiFi	Centralized	NS2 simulated	HTTP	X		
CUPUS [11]		WiFi	Distributed	Publish-Subscriber (MQTT-MOSQUITO)	HTTP-XML	X		

Continued on Next Page. . .

Table 2.5 – Continued from previous page

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols / Format	Performance Metrics		
						Generic	QoS	Scalability
BLISS [49]				Simulated				
SPREAD [57]		S/E		Simulated				
MAP+ [9]		WiFi/Cellular	Centralized					
ADTS [32]		WiFi/Cellular	Hybrid	Simulated				
FlierMeet [46]		WiFi	Distributed	-				
MDPPs [110]		WiFi/Bluetooth/ Cellular	Distributed	SOAP Web Services	HTTP-XML			
EasyHarvest [65]		WiFi/Cellular	Centralized	RESTful Web Services	HTTP-XML			
Sahyog [13]		WiFi/Cellular	Centralized	Publish-Subscriber	HTTP-JSON	X		
WiFIScout [129]		WiFi	Centralized		HTTP			
AnonySense [113]		WiFi/Bluetooth	Hybrid	SOAP Web Services (SMTP /SSL)	HTTP(S)-XML			
Pogo [20]		WiFi/Cellular	Centralized	Publish-Subscriber (XMPP OpenFire)	HTTP-JSON	X		
MECA [135]		WiFi/Cellular	Distributed					
Smartcity [117]			Distributed	Publish-Subscriber (XMPP)	HTTP-XML			
ILR [118]		WiFi/Bluetooth	Centralized	ILR Algorithm/Simulated NS2				
NoiseNYC [155]			Distributed					
BeC3 [31]		WiFi	Distributed	RESTful Web Services (COAP)	HTTP-XML			
McSense [26]		WiFi/Bluetooth	Distributed	RESTful Web Services	HTTP-XML	X		
SaaS [112]	S 3	WiFi/Bluetooth/ Cellular	Distributed	Web Services	HTTP			
GPS-less [131]		WiFi	Distributed	Approximation Algorithms	HTTP	X		
JoinPolices [10]		Budget	Distributed	Simulated Matlab		X	X	
MCSaaS [84]		WiFi/Cellular	Distributed	RESTful Web Services	HTTP-XML	X		
GROPING [147]		WiFi	Centralized					
Qoata [156]			Centralized	Simulated				
MCS game [132]		WiFi/Cellular	Centralized					
RuPS [85]		WiFi/Cellular	Centralized	RESTful Web Services	HTTP			
NoizCrowd [128]			Distributed					

Continued on Next Page. . .

Table 2.5 – Continued from previous page

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols / Format	Performance Metrics Generic	QoS	Scalability
CrowRecruiter [149]		WiFi/Cellular	Centralized	Simulated				
QoS-Constrained[127]			Hybrid	Simulated				
PLP [151]			Hybrid					
MoreWithLess[133]			Distributed					
Sparse[125]			Distributed					
Ear-Phone [105]		WiFi/Cellular	Centralized					
Medusa [103]		WiFi/Cellular	Distributed	MedScript-SMS-MMS	HTTPS-XML	X		X
CrowITS [6]		WiFi/Cellular	Centralized	RESTful Web Services (C2DM)	HTTP-JSON			
CAROM [146]		WiFi/Bluetooth/ Cellular	Hybrid	GSN (SOAP and RESTful Web Services)	HTTP	X		
SoundOfTheCity [108]		WiFi/Cellular	Centralized	SOAP Web Services (RTMP)	HTTP-XML			
MoPS [145]		WiFi/Cellular	Distributed	RESTful Web Services (GCM)	HTTP-XML			
Vita [54]	G [4]	WiFi/Cellular	Hybrid	RESTful Web Services	HTTP-XML	X		X
Matador [27]		WiFi/Cellular	Centralized	RESTful Web Services	HTTP-XML			
SenSocial [83]		WiFi/Bluetooth/ Cellular	Centralized	Publish-Subscriber (MQTT-MOSQUITO)	HTTP-JSON	X		X
MCSinSpace [92]		WiFi/Cellular	Distributed	RESTful Web Services	HTTP			
LineKing [21]		WiFi	Centralized	SOAP Web Services	HTTP			
Ecosystem [55]		WiFi/Cellular	Distributed	RESTful Web Services	HTTP-XML	X		
TYT [100]		WiFi/Bluetooth/Cellular	Centralized	RESTful Web Services	HTTP			
PRESM [130]		WiFi/Cellular	Centralized					
SmartRoad [53]		WiFi/Cellular	Centralized	SOAP Web Services	HTTP-XML			

[1] T: proposals addressing transmission issues.
[2] C: proposals centered on the client side.
[3] S: proposals centered on the server side.
[4] G: global solutions.

2.4 Open research issues

Based on the analysis presented in the previous sections, it becomes clear that, despite the many advancements introduced in the mobile crowdsensing field in recent years, there are still several issues that should be properly addressed for solutions to become more effective, and therefore gain more widespread acceptance.

At the user side, it becomes clear that the sensing tasks should not become a burden. Thus, any external sensors, if required at all, should be small, lightweight, have a low power consumption, and have an elegant and stylish look. Ideally, additional sensors should be progressively integrated into new smartphones either directly from the manufacturer or as pluggable modules. Power and network resource consumption are also an issue, and so smart algorithms able to correctly determine the best sampling times while avoiding intensive CPU usage are required; in terms of network resources, peer-to-peer data delivery combined with smart network selection can help at avoiding to deplete radio resources and having a negative impact in terms of traffic quotas.

From a more global perspective, further studies are required in order to assess the scalability and the QoS support of the different proposals. In particular, their impact on the end-to-end communications infrastructures should be thoroughly studied. Additionally, new algorithms should be developed to improve the processes of data collection and analysis.

2.5 Summary

The results of our analysis evidence that there is still much heterogeneity in terms of technologies adopted and deployment approaches, although modular designs at both client and server elements seem to be dominant. Also, the preferred client platform is Android, while server platforms are typically web-based, and client-server communications mostly rely on XML or JSON over HTTP.

Crowdsensing solutions that benefit from smartphones are proliferating due to the multiple advantages they offer. Thus, it becomes important to provide a unified view of the different author contributions to detect the major areas of improvement. In this chapter, we addressed this challenge by providing the reader with an extensive review of existing smartphone-based solutions in the field of Mobile Crowdsensing. We start by presenting a novel reference architecture where we identify the major components at the client side, server side, and at the communications level. Based on our proposed architecture, we then proceed to classify the different proposals, focusing separately on the client, the server, and the communications part of each solution.

Our extensive literature analysis has shown that most proposals provide some degree of adaptability to different work environments. Also, we found that technologies and algorithms applicable at both client and server sides have evolved significantly, and are often available in a modular format, allowing other researchers to include them in their proposed solutions. Concerning improvements in the data capture process itself, we found that the main issues are i) the software adaptability to different types of sensors, and ii) reducing power consumption. At the server side, the most critical improvements include i) task generation language and procedures, ii) the analysis and storage of data, and iii) providing an adequate interface for task management by administrators. The communication between client and server usually makes use of technologies like SOAP and RESTful, and most solutions support Publish/Subscriber models.

Overall, we consider that mobile crowdsensing is now achieving its maturity, being a widespread adoption of crowdsensing solutions expectable in the next few years.

Chapter 3

Environmental Pollution: an Overview

In this chapter, we focus on two different types of environmental pollution affecting the life quality of citizens in urban areas: air pollution and acoustic pollution. We start by introducing these topics and motivating why they are so relevant nowadays. Then we present a review of state of the art, where we describe the efforts and projects related to these two topics that have emerged in the last few years from a technical and research point of view, and where mobile sensing is the dominant approach. Finally, we present the main conclusions derived from our analysis.

3.1 Introduction

IN recent decades, air pollution monitoring has gained worldwide relevance due to the influence of air quality on our lives. Air pollution consists of the emission of gases or particles into the atmosphere, producing changes in its composition. Air pollution levels are a critical aspect to consider nowadays since it is associated to several problems affecting people's life quality, such as health issues (mainly in the respiratory tracts), climate changes, and reduced agriculture production, among others.

Many research works study the effects of pollution on our health. Among them, we can find the contributions of Chen at al. [28, 29], who analyzed the effects of ozone and related particle on human health. Brook et al. [19] also contributed to

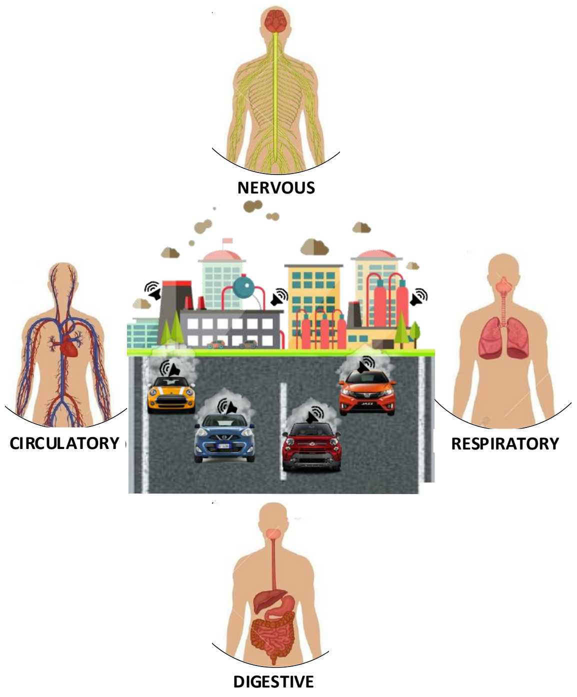


Figure 3.1: Examples of health problems caused by environmental pollution (poor air quality and noise).

this field by studying the relationship between the exposure to air pollution and cardiovascular effects.

On the other hand, noise is considered a particular type of environmental pollutant since, at certain levels, it can affect people both physiologically and psychologically and, more important, it can also interfere with basic activities, such as sleep, rest, study, communication and socializing [3, 90]. In fact, different studies [62, 144] have highlighted the importance of noise control in highly-populated areas. Similarly, the European Environment Agency has been enacting new regulations for the evaluation and control of environmental noise [93, 94], and all major cities have their own regulation. Figure 3.1 shows some of the main health problems associated to both types of pollutants.

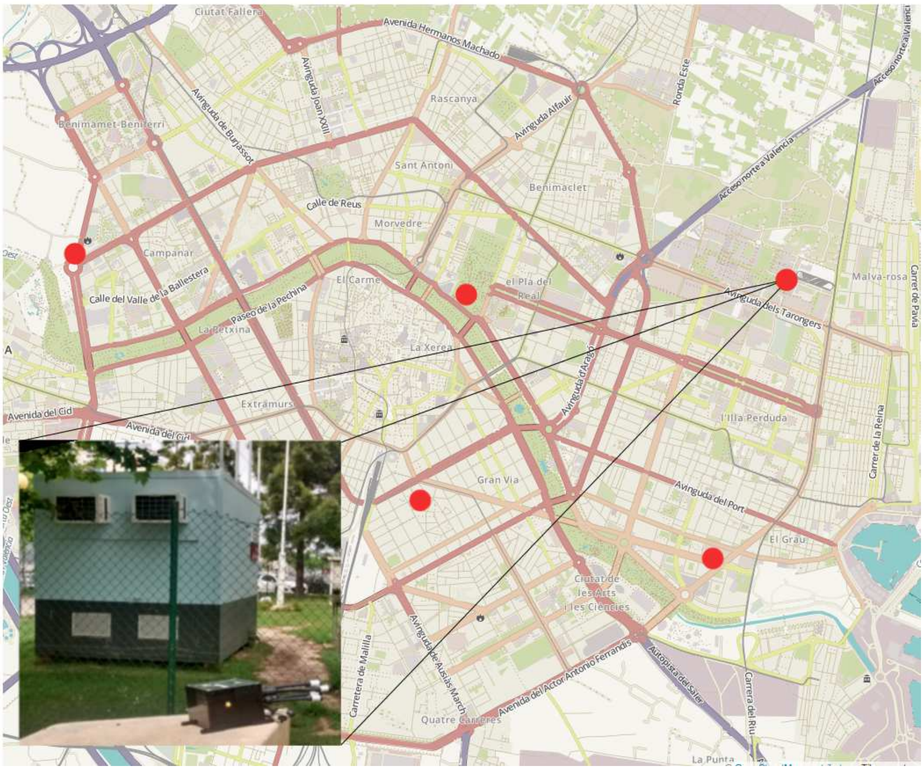
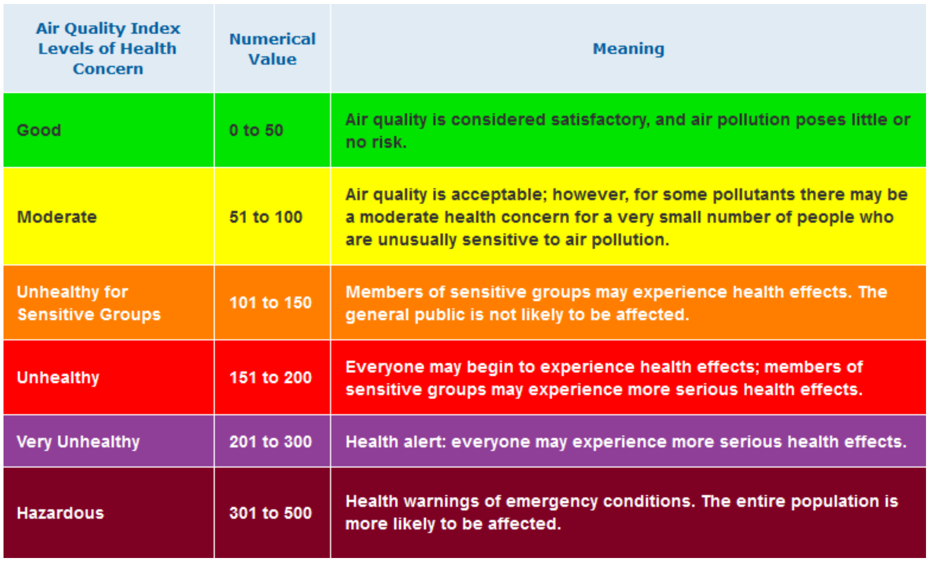


Figure 3.2: Example of an air monitoring station, and the location of the 5 stations available in Valencia, Spain.

3.2 Pollution measurement

3.2.1 Measuring air pollution

In Europe, about one thousand five hundred air monitoring stations have been deployed to control air pollution on a large scale, providing coarse-granularity pollution levels for most relevant cities. Despite this number may seem large, when focusing on a specific city, we find that these stations are quite scarce, failing to provide detailed pollution levels on a per-neighborhood basis [70]. For instance, in Valencia, the third largest city in Spain, there are only five monitoring stations, as shown in Figure 3.2. These can provide pollution levels on a large scale but, for better data granularity, and to study spatial variability with more detail, it would be necessary to have many of these stations, which becomes unfeasible due to the high costs associated.



Air Quality Index Levels of Health Concern	Numerical Value	Meaning
Good	0 to 50	Air quality is considered satisfactory, and air pollution poses little or no risk.
Moderate	51 to 100	Air quality is acceptable; however, for some pollutants there may be a moderate health concern for a very small number of people who are unusually sensitive to air pollution.
Unhealthy for Sensitive Groups	101 to 150	Members of sensitive groups may experience health effects. The general public is not likely to be affected.
Unhealthy	151 to 200	Everyone may begin to experience health effects; members of sensitive groups may experience more serious health effects.
Very Unhealthy	201 to 300	Health alert: everyone may experience more serious health effects.
Hazardous	301 to 500	Health warnings of emergency conditions. The entire population is more likely to be affected.

Figure 3.3: Relationship between Air Quality and health.

Even a small and low-cost mobile station must be endowed with several sensors able to measure different types of air pollutants. Such pollutants can be of two types: (i) primary air pollutants, which are gases or particles emitted directly into the atmosphere; in this category we have Carbon Monoxide (CO), Carbon Dioxide (CO_2), particulate matter PM smaller than 10 microns (PM10), or particulate matter smaller than 2.5 microns (PM2.5); and (ii) secondary air pollutants, which are gases produced by a chemical reaction between primary pollutants and some environment element; in this second category we have Ozone (O_3), which is produced by the combination of Nitrogen Oxides (NO_x), Oxygen (O_2), Volatile Organic Compounds (VOC), and sunlight [43].

Monitoring stations rely on sophisticated sensors, which are very accurate and introduce minimum uncertainty levels in the data capture process (e.g. Dobson spectrophotometers are used for monitoring ozone levels [14]). However, they are very expensive and hard to manage. Due to their size, they must be installed on a specific location, and the monitored value is only representative in a small surrounding area.

Concerning the different pollution levels, the Environmental Protection Agency (EPA) has defined the hazard associated to different pollutant levels in the atmosphere through the Air Quality Index (AQI) [4], which varies from safe to dangerous, as shown in Figure 3.3.

To detect air pollution, we can find a wide variety of technologies able to detect the most common pollutants (CO_2, CO, O_3, PM2.5 or PM10) at a low cost. Each

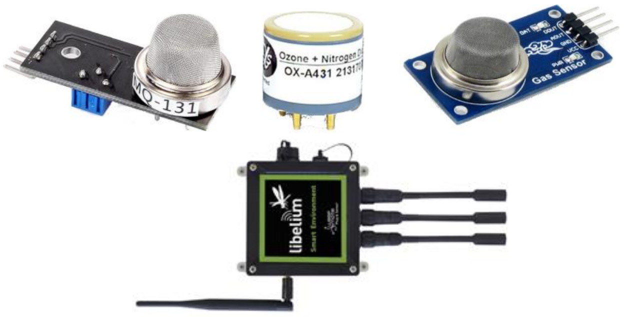


Figure 3.4: Different types of Air Pollution sensors.

technology has different properties, calibration processes, and costs, among other characteristics, and a comparison between these different technologies is required. In figure 3.4 we show examples of different air pollutant sensors. Specifically, it is worth mentioning that the Libelium Smart Environment solution (depicted in figure 3.4) is designed to monitor environmental parameters such as temperature, humidity, atmospheric pressure and some other types of gases through a set of sensors that can be incorporated (CO, O_3, among others.). Additionally, this solution has multiple radio options to communicate with other sensors such as ZigBee, 802.15.4, WiFi, RF at 868Mhz and 900Mhz, 3 Generation (3G), General Packet Radio Service (GPRS), and Bluetooth Low Energy at 2.4 GHz. Also, this equipment uses the MICS model 2610 as a resistive sensor that allows measuring the variation of ozone concentration between 10 and 1000 Particle per billion (ppb) [42].

An alternative for measuring environmental pollution is relying on mobile sensing. Specifically, small low-cost devices can be installed in various types of vehicles to monitor different parts of the city at different times. The main problem of low-end mobile sensors is that they have less accuracy than sophisticated sensors, and so they need to be regularly calibrated; besides, measurements are also weather-dependent.

3.2.2 Measuring noise pollution

The traditional approach for measuring acoustic pollution relies on professional sound level meters, which are of considerable cost and size, having high accuracy and sensitivity. Usually, these measurements are taken at a limited number of places and are then processed using different statistical techniques to generate acoustic pollution maps for well-defined target areas, thereby providing a fine

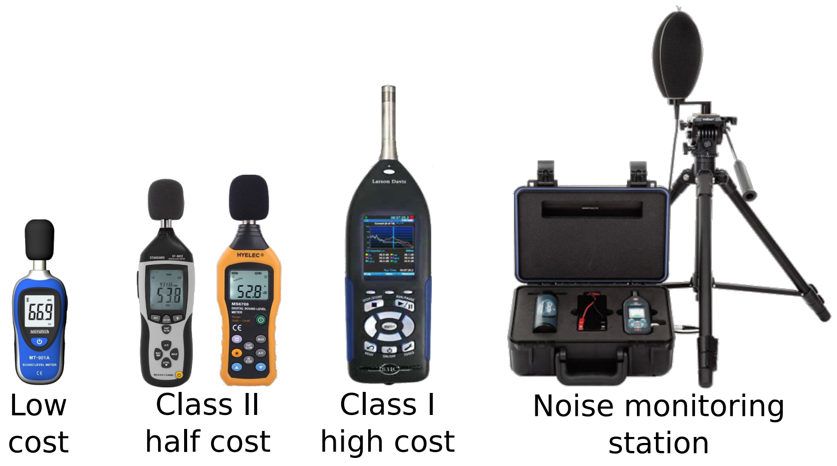


Figure 3.5: Different types of Sound Level Meter.

spatial granularity.

The International Electrotechnical Commission (IEC) 61672:2002 [69] defined two categories, namely Class I and Class 2, for Sound Level Meter (SLM) equipments, defining progressively loose tolerance levels of ±1.1 and ±1.4 decibel (dB), respectively. Thus, Class I devices are highly accurate, whereas Class II encompasses general purpose devices. Usually, based on this standard, the measured noise values are adjusted using filters. All SLM devices apply frequency weightings of Type A, B, C, D or Z [69]. In particular, A-weighting is regularly used because it offers a good correlation with the subjective human perception. In particular, the filter can be applied both in the time and frequency domains.

The basic element supporting noise level estimations is the microphone. Nowadays, smartphones incorporate one or more internal microphones, and their level of sensitivity varies according to the smartphones' brand and model. In addition, the quality of the readings they provide is directly related to characteristics such as the type of filter used (i.e., MaxRF, Enhanced Radio Frequency (RF)), and it is impedance.

In general, once the microphone is activated, it responds to vibrations in the air (sensitivity), converting them into electrical current fluctuations (sound pressure). Afterward, these electrical signals can be evaluated as a signal in the time domain or in the frequency domain. The first approach is a common one, representing the value of the signal as a function of time. The second requires a previous mathematical transformation procedure to switch to the frequency domain, thus introducing additional overhead. In both methods, the sound signal can be processed, and its loudness computed over long time intervals.

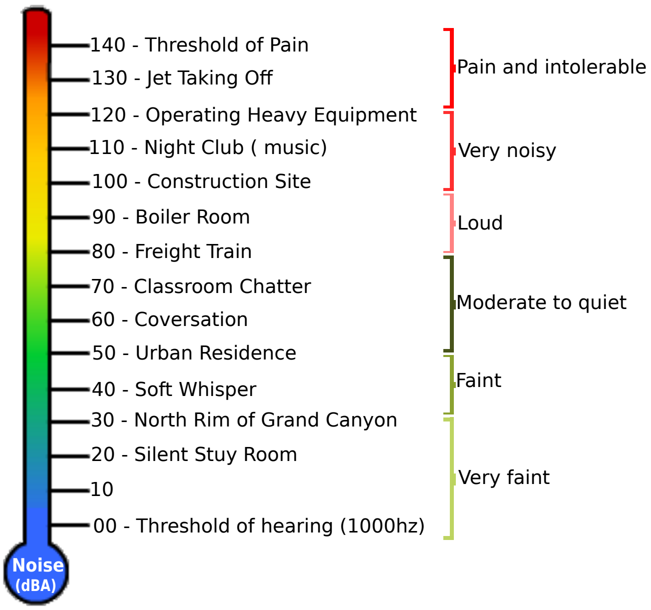


Figure 3.6: Typical Sound Levels (dBA).

Noise is measured in units of sound pressure called decibel (dB). The decibel notation is suggested any time a "sound pressure level" or "sound level" is mentioned. Decibels are measured on a logarithmic scale: a small change in the number of decibels indicates a large difference in the amount of noise and the potential damage to a person's hearing. The decibel scale is convenient because it compresses sound pressures relevant to the human hearing into a manageable scale. Figure 3.6 shows the levels that are relevant from the perspective of human hearing.

3.3 Studies addressing air pollution

In the literature, we can find several works related to crowdsensing systems applied to air monitoring.

Zualkernan et at [5] it uses a microcontroller, several air pollution sensors, a GPRS modem and a GPS integrated into the smartphone, which uses the public telephone network to transmit the data to the Server.

Hasenfratz et al. [50] show a mobile solution connected to a sensor through an RS232 interface. This application presents ozone pollution values in real time, and it relies on the Android operating system.

Mead et al. [82] analyze the behavior of electrochemical sensors to monitor air pollution in urban scenarios. They design two types of sensors (static and mobile) based on the PIC18F67J10 microchip, and they show how to deal with electrochemical sensors.

Zheng et al. [154] show how to analyze the data obtained from different sources, such as traffic levels, weather conditions, and pollution, using different Big Data techniques, and evidence how these techniques allow inferring environmental pollution levels with better granularity.

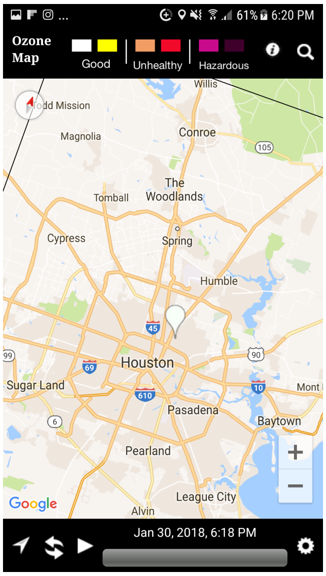
Cheng et al. [30] propose a system to monitor the concentrations of PM2.5 particles using crowdsourcing, which is an alternative to using mobile sensors. They focus on the analysis of the mechanical sensor design to optimize the air reception, as well as on data fusion techniques to analyze the data. Sensor calibration is achieved by analyzing data produced in the laboratory using neural networks.

Manna et al. [80] propose a sensor to monitor air pollution on roads, and to track vehicles which cause pollution, by using a sensor based on Arduino, an electrochemical CO gas sensor, and RFID technology.

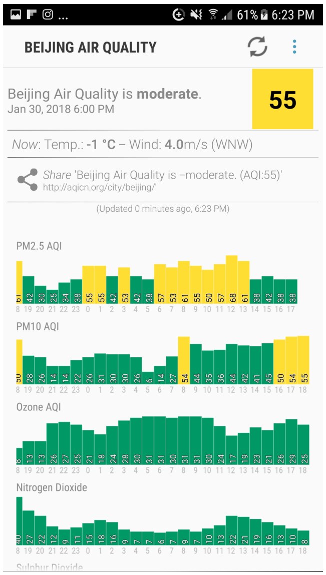
Determining the pollution distribution in a city based on a few samples requires adopting spatial interpolation techniques for estimating it, and mobile sensing is the best option to achieve it. In the literature, we can find several works adopting this approach. For instance, Brković et al. [18] propose a system to monitor environmental pollution in the city of Belgrade using Waspmote sensors [74] installed in the public transport system. Hu et al. [39] use a vehicular sensor network for air pollution monitoring. In particular, they propose to use taxis for deploying the system and mainly analyze the communication between them.

In this regard, studies such as [1] and [76] have relied on kriging interpolation techniques to predict pollution. These studies were made in the cities of Quebec and Toronto, respectively. More recently, Calafate et al. [24] combined mobile sampling techniques with kriging-based interpolation to determine the achievable accuracy when estimating the ozone distribution in a city, relying on the public transportation system for data gathering.

Additionally, some mobile applications have been developed that show information on air quality, as well as other applications (app) that serve as educational information on environmental pollution, among others. Concerning the above, in Figure 3.7 some free access mobile applications are shown through the Google Play platform. These applications are connected to one or several ground stations, and in some of these applications, the information available is not always presented in real time. Also, none of these applications allows interacting with external sensors that can be incorporated into smartphones.



(a)


BEIJING AIR QUALITY

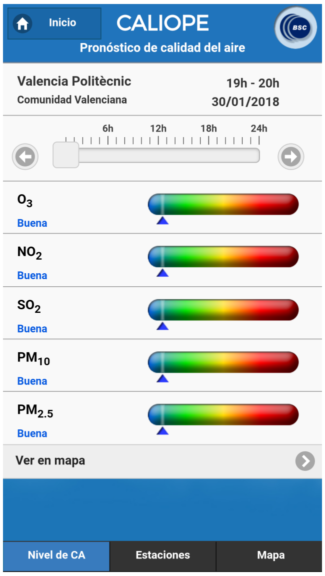
Beijing Air Quality is **moderate**. **55**
Jan 30, 2018 6:00 PM

Now: Temp.: **-1 °C** – Wind: **4.0**m/s (WNW)

Share 'Beijing Air Quality is –moderate. (AQI:55)'
http://aqicn.org/city/beijing/

(Updated 0 minutes ago, 6:23 PM)

(b)


Inicio CALIOPE
Pronóstico de calidad del aire

Valencia Politècnic
Comunidad Valenciana
19h - 20h
30/01/2018

O_3
Buena

NO_2
Buena

SO_2
Buena

PM_{10}
Buena

$PM_{2.5}$
Buena

Ver en mapa

Nivel de CA | Estaciones | Mapa

(c)

Figure 3.7: Available apps for air measurement. (**a**) OzoneMap.; (**b**) BeijingAir.; (**c**) Caliope.

3.4 Studies addressing noise pollution

In the literature, we can find several solutions where smartphones are used as mobile sensing devices to make noise pollution measurements. For instance, works such as [79, 62] propose noise sensing solutions where the developed applications include a real-time (only smartphone) sound-level data logger that also includes GPS data to generate a map. However, these works fail to provide details about the sensing task itself.

NoizCrowd [128], and Ear-Phone [105, 104] study noise levels using different spatial and temporal interpolation techniques. In particular, Ear-Phone [104] proposes an algorithm that attempts to optimize noise sampling by detecting when the phone is placed in the trouser's pocket, or in a bag. NoizCrowd [128] uses GPS to determine the users' locations, but it fails to determine when and how the collection task is performed.

SoundOfTheCity [108] is a proposal that uses several sensors to provide context-awareness. This context-awareness allows distinguishing between situations where the user is located outdoors, indoors, moving, or if the smartphone is in the user' pocket (using GPS, WiFi, and proximity) in order to determine the right instant to trigger the measurement. Later, NoiseSense [101] proposed a semi-supervised sensor completion algorithm for inferring noise levels for locations in an urban area where smartphone users are unable to provide measurements. Their work does not present details on the data collection process, focusing solely on the sending of raw data to a server.

Alternatively, Usense [2] presents a generic middleware for developing and deploying crowdsensing applications. Also, it adds a module that evaluates the precise moment of capture based on rules. This solution does not detail the noise calibration process, nor does it describe or evaluate the size of the collection window.

Not only are there solutions that study the level of noise to interpret them into an urban pollution map, but there are also proposals, such as Pryss et al. [100], that offer a crowdsourcing platform where the microphone of mobile devices is used as an aid for the medical aspects of tinnitus and its treatment. Similarly, Ren et al. [107] use fine-grained techniques to capture the behavior of breathing during sleep through smartphones. In [86], Monge-Alvarez et al. propose an automatic system for the detection of a cough based on the standard audio signal of smartphones. They use a local database of sounds of coughing for comparison, and their processing uses emotion recognition algorithms.

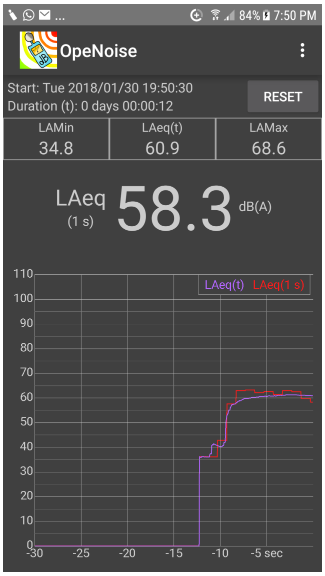
Recently, works such as [63, 88] evaluated the quality of noise level measurements of different mobile applications for both the Android and the iPhone Operating System (iOS) operating systems. In [63], the authors selected noise levels from 65 to 95 dB in 5-dB increments and generated pink noise within a 20 Hz to 20 kHz frequency range. The results showed that specific sound-measurement applications are inaccurate, thus being unreliable when assessing noise levels via smartphones, especially if the Android operating system is used. In [88], five different applications have been evaluated using an Apple iPhone 4s, where the authors evaluated different frequency ranges up to 8 kHz, showing that a large number of applications report wrong levels for loud sounds.

In [64], the authors added external calibrated microphones to the ones provided in smartphones and compared the obtained values against reference measurements. This study evidenced that values retrieved using smartphones' microphones were considerably different, inaccurate than those obtained using professional devices.

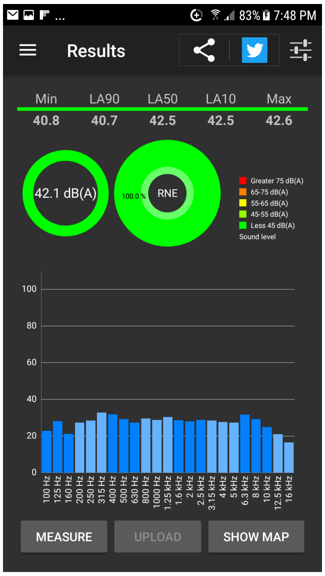
In [157] authors use two components (node-based, and crowdsourcing-based) to calibrate smartphones. The node-based component uses a lightweight algorithm to determine the offset using a linear model, doing offline calibration autonomously. Regarding the crowdsourcing-based component, it provides specific calibration parameters for different smartphone models, thereby maintaining a list of the required calibration parameters for each of them.

In Table 3.1 we show the characteristics of the main solutions found in the literature offering crowdsensing-based noise assessment. Our analysis focuses on the smartphone, transmission and task specification components.

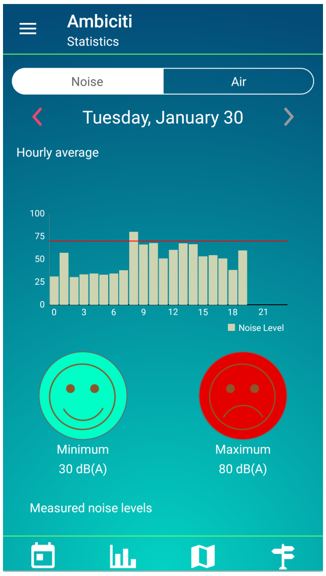
Finally, some freely available applications (Apps) allowing environmental noise to be measured using either Android-based or iOS-based mobile phones were analyzed. Some of these applications allow to visualize the measured noise in real time, and they also provide a history of the noise to be shown, including heat maps. However, most of them are not integrated into a crowdsensing solution. In Figure 3.8 we illustrate the most relevant applications studied.



(a)



(b)



(c)

Figure 3.8: Available apps for noise measurement. (**a**) OpenNoise.; (**b**) Noise Capture.; (**c**) Ambiciti.

Table 3.1: Characteristics of crowdsensing noise pollutions proposals.

Publication	Smartphones			Data Transmission Networks	Cloud Server task specification		
	Noise calibration procedure	Data collection	Inteligent context awareness		Selection area/date	Real time data task	Share maps
NoiseTube[79]	Linear interpolation	Participative interaction	No	offline	Manual	No	Yes
NoiseSPY[62]	Correlation calibrated	Participative	No	online (cache store)	Manual	No	No
NoiseMap[111]	Few details	Participative	No detail	offline	No detail	No	Yes
NoiseHound[87]	Few details	Participative	Strategy area for spatio-temporal	Offline ad hoc nework	No	No	Yes
Usense[2]	No-detail	Automatic by Server	Yes Rule-based attributes	No detail	Automatic Area and between date	Yes	No detail
WideNoise[15]	No detail	Participative	No detail	Offline	Manual area	No	Yes
NoizCrowd[128]	No detail	Participative	No	Offline	No detail task area	No	Yes
SoundOfTheCity[10	No detail	Participative	Yes	Offline (SOAP)	No	No	Yes
Sense2health[48]	Few details	Participative sensing	No detail	Yes (publish/sub-scribe RabbitMQ)	No	No	Yes
Ear-Phone[105, 104]	Regression lineal	Participative	Yes	No	No	No	Unclear
OnoM@p[159]	Cross calibration	Participatory	No detail	No detail	No details	No	Yes
GRC-Sensing	Regression lineal and others	Automatic Only Install	Yes	Online	Yes	Yes	Yes

Finally, activity recognition is also a topic closely related to smartphone-based noise assessment, as the relevance of samples taken will greatly depend on the correct understanding of the user activities, and the interaction with their smartphones. Likewise, battery depletion may become a problem for crowdsensing solutions if resource consumption is not optimized. In this context, the authors of [34] introduce a crowdsensing framework for location recognition. Specifically, they use Hidden Markov Models algorithms and Gaussian models for speech recognition and sound classification, respectively. Works such as [73] shows how classification algorithms can be used for human activity recognition systems using wearable sensors. Shoaib et al. [114] present a review of the works using recognition systems on smartphones based on their onboard sensors. The works surveyed presented some considerations when designing applications were having sensor-based activity recognition. In particular, it emphasized that, from the 30 reviewed papers, 60% of them did not address battery consumption in their analysis, and only 27% of them performed a Central Processor Unit (CPU) usage analysis.

3.5 Summary

In this chapter, we have focused on two environmental pollutants that are very relevant in the context of smart cities, clearly affecting the life quality of citizens in urban areas. In particular, we have dealt with air pollution and noise pollution. In both cases, the traditional way of measuring them (fixed infrastructure) was evidenced. Then, we introduced the most relevant research works that have contributed to the state of the art in the mobile crowdsensing context. Concerning air pollution solutions, the adoption of different interpolation techniques for processing the information gathered is indicated. Also, it was found that, in general, mobile detection solutions require adopting an external air particle detection device. Concerning noise pollution, we have found a set of works that use the microphone of smartphones for different purposes, including noise detection. In general, the solutions adopting smartphones to measure noise fail to provide details about the results of their measurements. Additionally, solutions have been found that use statistical techniques and classification algorithms to optimize noise sampling. Finally, we have found that none of the works analyzed in this context actually provides a complete mobile sensing architecture.

Chapter 4

Ecosensor: a mobile sensing architecture to monitor air quality

Mobile sensing is becoming the best option to monitor our environment due to its ease of use, high flexibility, and low price [18, 24, 30, 154]. Chapter 3 has shown that different authors have proposed several mobile monitoring solutions, but most of them merely provide an isolated analysis, failing to provide a full mobile sensing architecture.

In this chapter, we present a mobile sensing architecture able to monitor different air pollutants using low-end sensors. Although the proposed solution can be deployed anywhere, it becomes especially meaningful in crowded cities where pollution values are often high, being of great concern to both the population and authorities. Our architecture includes two modules: a mobile sensor for monitoring environmental pollutants that integrated a Android-based device for transferring the gathered data to a central server, and a central processing server for analyzing the pollution distribution using the collected data through spatial interpolation techniques.

We will start by providing an overview of the proposed architecture. Then, we will detail the procedure followed in order to obtain reliable measurements from low-cost sensing devices, and we discuss the optimal strategy for performing mobile measurements. In particular, we will focus on how to properly calibrate the sensor, and how to reduce time variability. Also, we will study the best strategy to use mobile sensors by first determining the influence of sensor orientation on the captured values, and then analyzing the influence of time and space sampling in the interpolation process. Finally, we validate the proposed architecture through comparison against infrastructure-based data.

4.1 Mobile sensing architecture overview

Our proposed architecture defines a set of elements that allow monitoring of air pollution cheaply and easily, being specially useful in very crowded cities. It combines data coming from existing air quality monitoring stations with data collected by mobile sensors to generate fine-grained reports about pollution levels. Mobile sensors can be installed in bicycles or along the public transportation system to monitor the city simply and effectively. All collected information is stored on a central cloud server for data processing, generating detailed reports afterward. The architecture of the proposed system integrates several hardware and software components. These components can be classified as mobile sensing elements or cloud elements, being the latter a set of services running on a server that analyze collected data and present detailed information. Mobile sensing elements are composed by three different components: (i) a Waspmote sensor for measuring pollution data, (ii) a Raspberry Pi [106] that acts as a gateway between the sensor and the Android-based device, and (iii) an Android-based device for showing real-time pollution status, storing the data, and transferring it to the Cloud server when network connectivity is available.

This architecture is shown in Figure 4.1. The Waspmote sensor [74] is based on an Arduino platform [12], and it measures air quality through various sensors (Ozone, CO_2, Air Pollution, and temperature). Moreover, it has has a GPS interface that allows determining the exact location of each measurement. Once data is ready, it is transferred to the Raspberry Pi via Zigbee [158]. The Raspberry Pi acts as a gateway between the Waspmote sensor and the Android-based device. It has a Raspbian operating system, and it is programmed in Python. It has two communication interfaces: ZigBee for connecting to the Arduino platform, and Bluetooth [16] for transferring data to the Android device. The Android-based device shows, in real time, the pollution level registered at a certain location and allows transferring the gathered data to the Cloud server. In particular, it uses the Bluetooth interface to receive data from the Raspberry Pi, and the Wifi or cellular network interface to transfer data to the Cloud server. The Cloud server has a web system that handles the information received from the Android device. The received data is stored in a database, which is then processed through the R Graph tool [102]. Finally, a detailed report is made available to the administrator of the web server.

4.1.1 Ecosensor applications

Ecosensor is a platform composed of two integrated modules based on our architecture proposal, (i) mobile elements responsible for capturing pollution values,

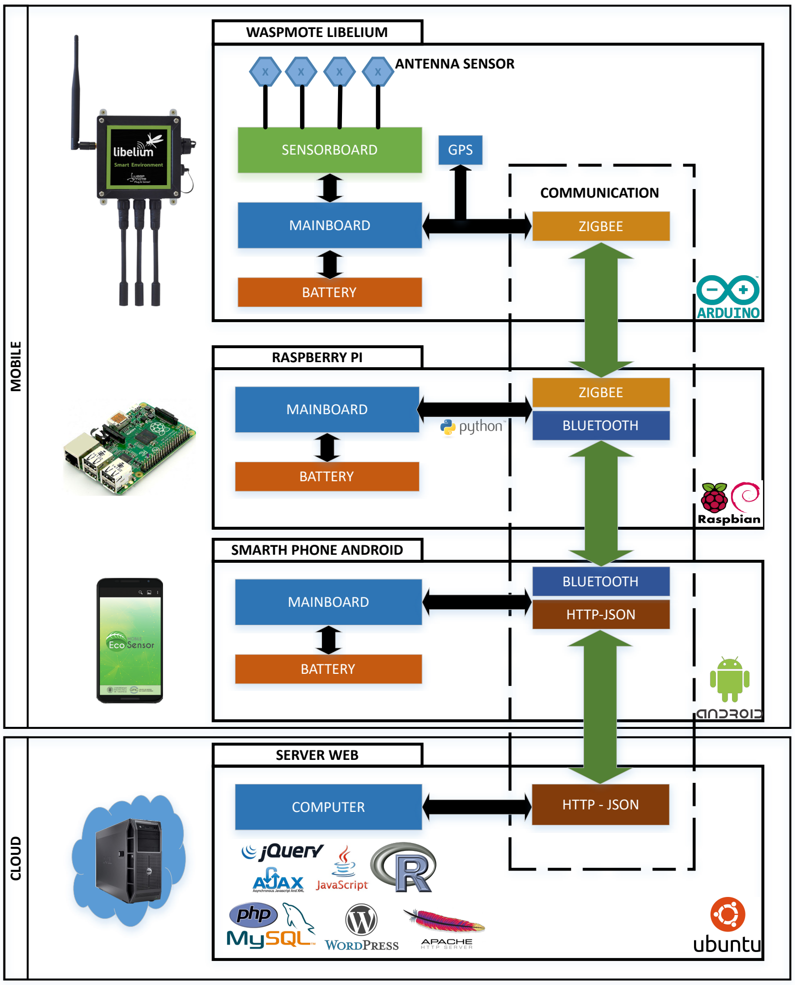


Figure 4.1: Overview of the proposed mobile sensing architecture including the main hardware components and the technologies used.

and (ii) a cloud server responsible for storing and processing the collected data. The mobile element is controlled via an Android-based application, and a Cloud Application manages the server. Below we provide a detailed overview of both applications.

Android application

The Android application was developed using the Android Studio IDE. This application allows starting or stopping a trace, viewing captured data in real-time, uploading data to the server, and performing other management tasks. Internally, the application has two parts: (i) a service that continually receives (via Bluetooth) the data sent by the sensor, and that stores it in an internal database; and (ii) a user interface that allows starting or stopping a trace data capture from the sensor. It also provides real-time feedback about pollution levels at the current location according to the AQI index [4]. Moreover, the full trace can be represented on a map showing pollution variations through different color identifiers. Once the trace is completed, the data can be sent to the server via an HTTP POST message using the JSON format. The application is shown in Figure 4.2)

Server applicationn

The cloud server provides a web interface based on Word-press, which allows the administrator to have full access to the information gathered regarding trace handling, processing, and visualization. Once logged, the administrator views all uploaded traces, being able to perform different statistical analyses on the different datasets (CO_2, Ozone, Air Pollution, and Temperature). For statistical analysis and report generation, we relied on the R Graph tool, which offers us a way to generate graphics concerning:

- Pollution level.- It shows detailed pollution levels for a specific area.
- Kriging analysis.- It presents the kriging output: (i) interpolation result, (ii) kriging error, and (iii) variogram analysis.
- Captured data filtering.- It presents the adjustment/calibration process, showing the relationship between the original data and the resulting data.
- Data variation.- It provides a boxplot showing the overall data distribution.

The site is available at http://www.ecosensor.net, and its design is shown in Figure 4.3.

4.2 Monitoring Process

After defining the proposed architecture, we now focus on the most relevant issues regarding the reliability of the pollution monitoring process. Our target pollutant was ozone due to its well known negative impact on health, and also because it is more complicated to measure accurately than other pollutants due to its dependency on temperature and time of day.

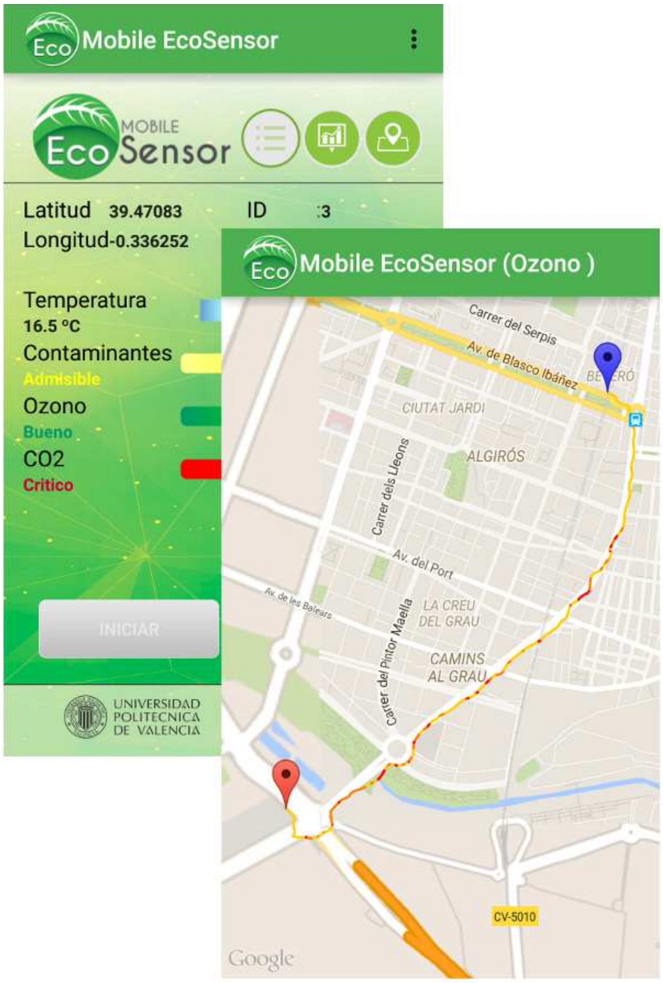
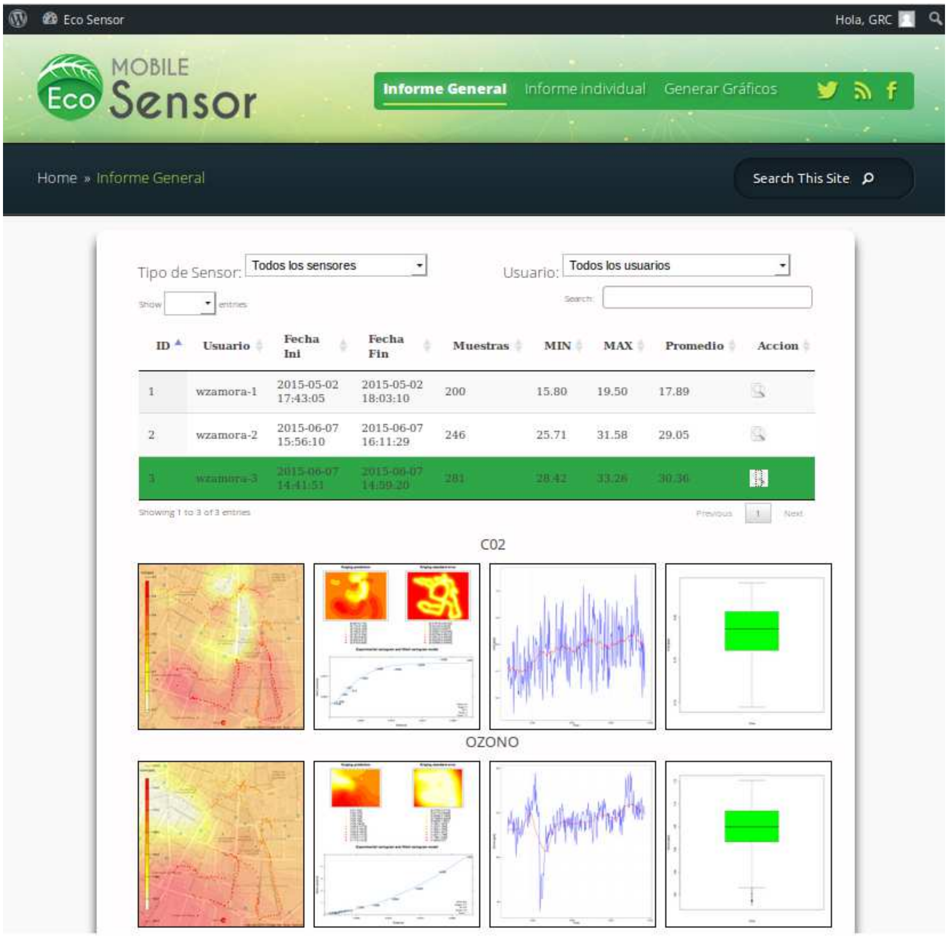


Figure 4.2: Android-based application deployed monitoring screen (back) and path of a monitoring session (front).

The issues that should be taken into account to perform accurate ozone measurements are the following:

- Sensor output data measurements are highly variable in ranges close to the real values, and so such variability should be reduced.
- The sensor outputs should be transformed into the respective units for each pollutant. In most cases, the measured resistance value must be converted



Tipo de Sensor: Todos los sensores

Usuario: Todos los usuarios

Show entries

Search:

ID	Usuario	Fecha Ini	Fecha Fin	Muestras	MIN	MAX	Promedio	Accion
1	wzamora-1	2015-05-02 17:43:05	2015-05-02 18:03:10	200	15.80	19.50	17.89	
2	wzamora-2	2015-06-07 15:56:10	2015-06-07 16:11:29	246	25.71	31.58	29.05	
3	wzamora-3	2015-06-07 14:41:51	2015-06-07 14:59:20	281	28.42	33.26	30.36	

Showing 1 to 3 of 3 entries

Previous 1 Next

Figure 4.3: Example of the cloud application web page showing some monitoring sessions, and the analysis output for two air pollutants.

into *particles per billion* (ppb).

- In order to use mobile sensors, time-dependent variability must be removed since different samples are obtained at different times.
- Using the adjusted measurements, the next phase is to apply spatial interpolation techniques for creating detailed pollution maps.

Figure 4.4 shows the different steps taken when transforming the raw sensor

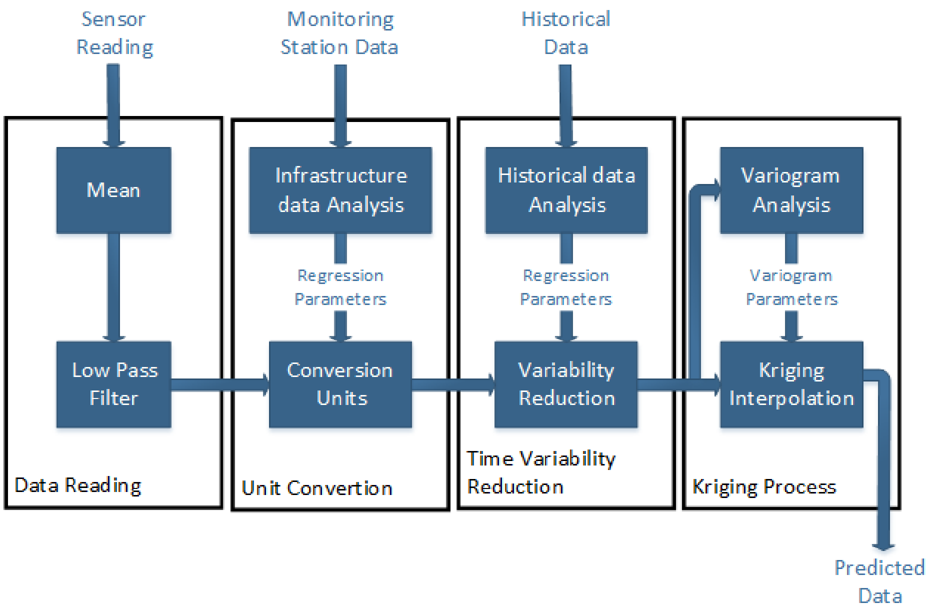


Figure 4.4: Monitoring process overview showing the different tasks associated to each step in the process.

readings in detailed air pollution maps. Also, below we discuss how each of these issues has been addressed.

4.2.1 Data reading

Low-end sensors introduce significant variability between consecutive measurements (absolute values for inter-sample differences are $\bar{x}$=6.15, σ=5.73), so data retrieval processes should eliminate these oscillations associated to noise in the sensor readings. For this purpose, we performed the following steps: first, we calculated the average value of 25 samples ($n = 25$), with an interval of $10ms$ between each consecutive sample, as shown in equation 4.1:

$$O_s = \frac{\sum_{i=1}^{n} O_i}{n} \tag{4.1}$$

In this equation O_s represents the estimated ozone level, O_i represents the ozone level sample i obtained from the sensor, and n represents the number of measurements. In this step, we slightly reduce the absolute variability ($\bar{x}$=5.39, σ=5.01). Afterward, and taking into account that the variability was still very

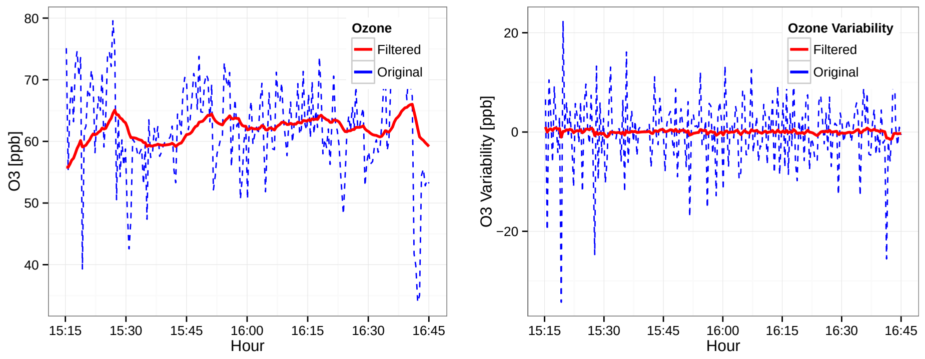


Figure 4.5: Relationship between captured data and filtered data (left), and relationship between captured data variation and the filtered data variation (right).

high, we used a low-pass filter for the data analysis process with α equal to 0.95 to further reduce this variability, as shown in equation 4.2:

$$O_i = O_r + \alpha \cdot (O_{i-1} - O_r) \tag{4.2}$$

O_i represents the current ozone level, O_{i-1} represents the ozone level in the previous measurement, O_r represents the filtered ozone value, and α represents the filter coefficient. In this step, we drastically reduce the absolute variability ($\bar{x}$=0.32, σ=0.30).

Figure 4.5 (a) shows the difference between the values of captured ozone levels and the values of ozone levels after applying the low-pass filter, and figure 4.5 (b) shows the variability after applying the mean and the low-pass filter. It shows that data variability is significantly reduced while maintaining the correct trend.

At the end of this process, we have measurements without the variability associated to noisy sampling.

4.2.2 Unit conversion

Sensors provide an electrical signal output, which needs to be transformed to a pollution level value. Specifically, the *Ozone sensor probe (MiC-2610)* has an internal resistance that varies proportionally to ozone concentration. The sensor can measure ozone variations between $10ppb$ and $1000ppb$, being that the resistance varies between $11k\Omega$ and $2M\Omega$ with a quasi-linear behavior.

Sensor specifications were made at a constant temperature of 25 degrees centigrade, and vary depending on weather conditions.

For calibrating the sensor we have done several measurements at different days, and under different weather conditions, to get a broad range of values. These data have been compared against the data obtained from the official monitoring station located at the Universitat Politècnica de València (UPV), Spain. Data obtained are shown in Table 4.1. Considering that the measurements have a dependency on both ozone levels and temperature, we obtained through regression a second-degree polynomial (see equation 4.3) that takes the temperature and the resistance obtained by the sensor into account to determine the actual ozone values.

Table 4.1: Relationship between sensor readings and monitoring station readings.

Resistance $[Ohm]$	Temperature $[^\circ C]$	Station Ozone $[ppb]$	Calculate Ozone $[ppb]$
25.0	19.0	80	63.55
31.0	16.0	65	73.10
28.0	15.5	52	55.22
35.0	13.0	83	79.72
28.0	28.0	120	117.46
23.3	23.0	70	77.58
23.5	22.0	70	73.35

$$O = \alpha + \beta_1 t + \beta_2 r + \beta_3 r^2 \tag{4.3}$$

In this equation α is a regression coefficient, β_1 is a temperature coefficient, β_2 is a sensor reading coefficient, β_3 is the reading coefficient squared, t is the measured temperature, and r is the sensor reading (measured as Resistance). The output O is the ozone level measured. The final regression obtained is shown in equation 4.4:

$$O = -29.19 + 4.79t - 3.09r - 0.13r^2 \tag{4.4}$$

The error obtained for the regression was $R^2 = 0.85$. Compared against a 1st order regression ($R^2 = 0.83$) the obtained result is better in terms of R^2. Compared against a 3rd order regression ($R^2 = 0.86$), the improvement in R^2 is minimum and the differences minor.

4.2.3 Time variability reduction

To cover large areas of land with a fine spatial granularity, we use mobile sensors, which can capture data at various points although at different time instants. So, the difference between measurements O has both times $\triangle O_t$, and spatial $\triangle O_e$

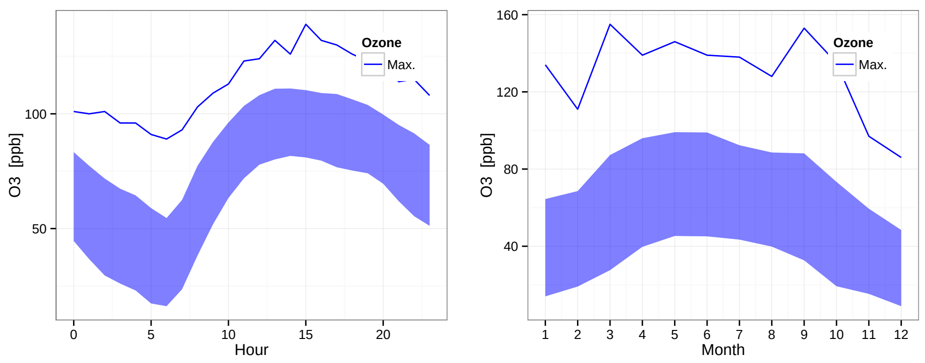


Figure 4.6: Ozone evolution in June (left) and throughout the year (right).

dependencies. Since our main goal is to determine the differences between ozone levels in a particular area, it is necessary to eliminate the time variation.

$$\triangle O = \triangle O_t + \triangle O_e \tag{4.5}$$

$$\triangle O_e = \triangle O - \triangle O_t \tag{4.6}$$

For the calculation of the ozone time variations, we analyzed data from a monitoring station located at the UPV, focusing on historical data between 2008 and 2014. In the historical data analysis, we analyzed the ozone evolution focusing on average monthly measurements between 2008 and 2014. It is noted that the values are higher from April to September, and lower for the remaining months. Figure 4.6a shows the mean values and standard deviation in the shaded area, and maximum values with the top line. The variation in ozone levels during a typical day of June was also analyzed. As shown in Figure 4.6b, ozone levels reach their lowest value at the end of the night, at about 6 am, and rise to reach maximum values at 2 or 3 pm, beginning to decline gradually afterward. The behavior for the other months of the year is analogous to the month shown.

As a result of the analysis of these data, we observe that ozone has different behavior in summer (specifically from April to September) compared to the rest of the year. During daytime, the behavior is very similar to the parabolic logarithmic distribution, with an onset of rapid growth followed by a less pronounced decline.

Based on the previous data regarding monthly average values between 2008 and 2014, taken at the monitoring station of the Technical University of Valencia, ozone level prediction relies on a parabolic logarithmic regression influenced by temperature and season of the year, one for summer, and one for winter. The expression used (in linear format) was the following:

$$\ln(O_t) = \alpha + \beta_1 s + \beta_2 t + \beta_3 \ln(h) + \beta_4 \ln(h)^2 \tag{4.7}$$

where h is time of day, s is the season, t is the temperature, and the remaining α and β_i values are regression coefficients (β_1- season coefficient, β_2- temperature coefficient, β_3- coefficient for the logarithm of the time of day, β_4- coefficient for the logarithm of the time of day squared).

$$\ln(O_t) = -7.70 + 0.03s - 0.01t + 9.23\ln(h) - 1.77\ln(h)^2 \tag{4.8}$$

$$\ln(O_t) = -15.43 + 0.12s + 0.03t + 14.42\ln(h) - 2.83\ln(h)^2 \tag{4.9}$$

The values of $\left\|R^2\right\|$ are 0.91 and 0.82 for summer and winter, respectively, showing a behavior very similar to the actual one.

The procedure followed to correct time-dependent variability was: (i) ozone values are calculated at two-time instants using equation 4.7; (ii) the difference between the values is obtained; (iii) the actual readings are reduced according to the calculated variation.

4.2.4 Interpolation data

The adjusted data is the input for creating detailed pollution maps. In the scope of this work, this is achieved by using the R graph tool. Specifically, we rely on spatial interpolation techniques known as ordinary kriging. First, a semivariogram is calculated for a specific area, and kriging parameters are determined. Next, a detailed pollution distribution is created using the obtained parameters. To easily visualize the pollution levels distribution in space, different maps are created, as shown in Figure 4.7.

The semivariogram defines the variance of the differences between two points. It determines the parameters required for the kriging interpolation, which have an influence on the distribution form.

- Sill: determines the total variance of the values.
- Nugget: determines the variance at the origin.
- Range: determines the range of influence of the model.
- Model: determines the distribution function. It can be Gaussian, Spheric, Exponential, Circular or Linear.

Figure 4.8 shows a sample semivariogram as an example.

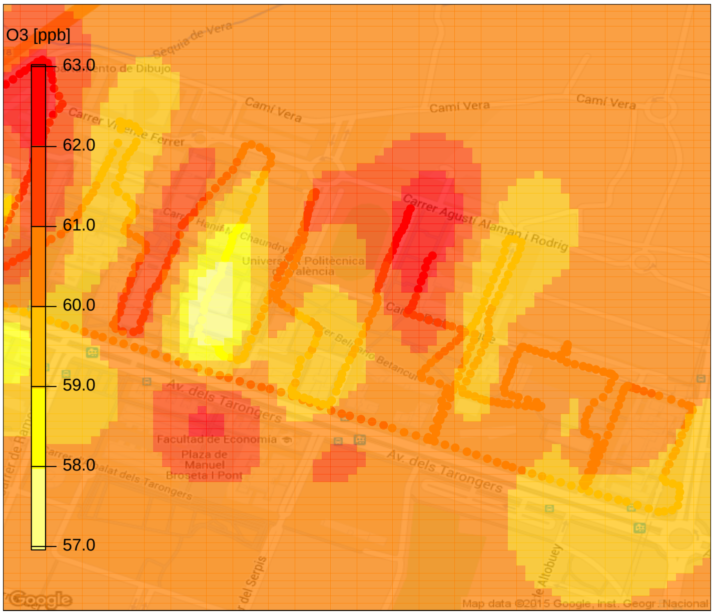

O3 [ppb]
63.0
62.0
61.0
60.0
59.0
58.0
57.0
Camí Vera
Carrer Agustí Alamán i Rodrig
Av. dels Tarongers
Facultad de Economía
Universitat Politècnica de València
Map data ©2015 Google, Inst. Geogr. Nacional

Figure 4.7: Example of an ozone distribution heatmap for the UPV using the proposed architecture.

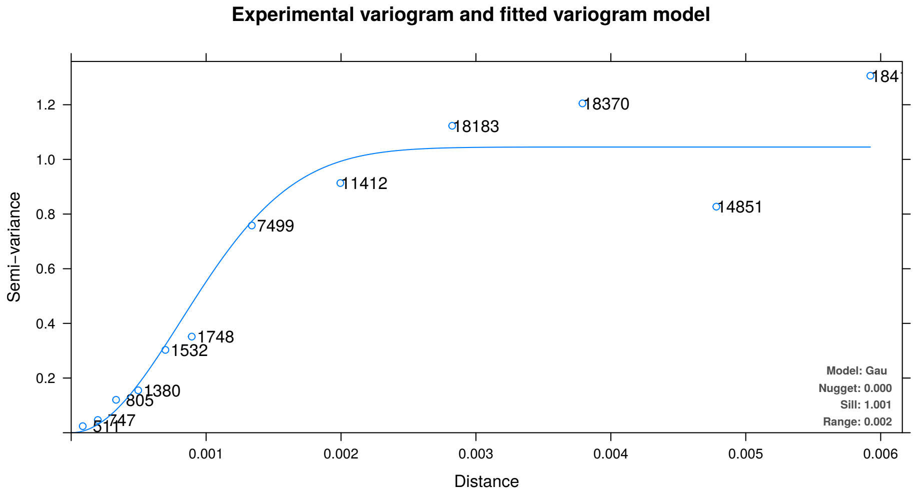


Figure 4.8: Example of a semivariogram showing a Gaussian distribution. It shows the different parameters related with the interpolation techniques (Nugget, Sill and Range).

4.3 Finding the optimal measurement strategy

After defining the architecture and the monitoring process, we now proceed to determine the optimal strategy for air pollution data collection using mobile sensors.

With this purpose, we first analyzed the impact of mobility on sensor readings by comparing static against mobile measurements. Also, we determined the influence of sensor orientation in the mobile sensing process. Our next step was to analyze the impact of reducing the sampling frequency on the kriging process accuracy under mobile scenarios. Similarly, we analyzed the impact of reducing the number of spatial samples on the kriging process accuracy. This was achieved by skipping selected streets when capturing data, progressively reducing the overall path.

4.3.1 Optimal sensor positioning

To analyze the impact of mobility on the data capture process we performed different tests, collecting ozone levels in a specific area either statically, or using a bike moving at a speed of about 20km/h. For mobility tests, we collected measurements with different sensor orientations: (i) facing forward, (ii) facing backward, and (ii) facing up. Statistics for the "mobile" case combine measurements with different sensor orientations.

Table 4.2: Statistical summary of the sensor position analysis.

Period	Mean	Std. Dev.	p-value
Static	27.39	0.85	-
Movement	27.34	1.03	0.25
Facing Forward	27.41	1.04	0.77
Facing Backwards	27.44	1.02	0.38
Facing Upwards	26.85	0.95	0.06

To have further insight on how these results are distributed, Figure 4.9 shows that mobility, at least at speed used for testing, does not have a significant impact on sensor measurements.

The results for the t-test analysis are shown in Table 4.2, revealing that we cannot find a statistically relevant difference between the static sensor ($\bar{x} = 27.39, \sigma = 0.85$) and the mobile sensor ($\bar{x} = 27.34, \sigma = 1.03$), obtaining a p-value = 0.25 and $\alpha = 0.05$, neither for the facing forward orientation ($\bar{x} = 24.41, \sigma = 1.04$, p-value = 0.77) nor for the facing backwards orientation ($\bar{x} = 27.44, \sigma = 1.02$, p-value = 0.38).

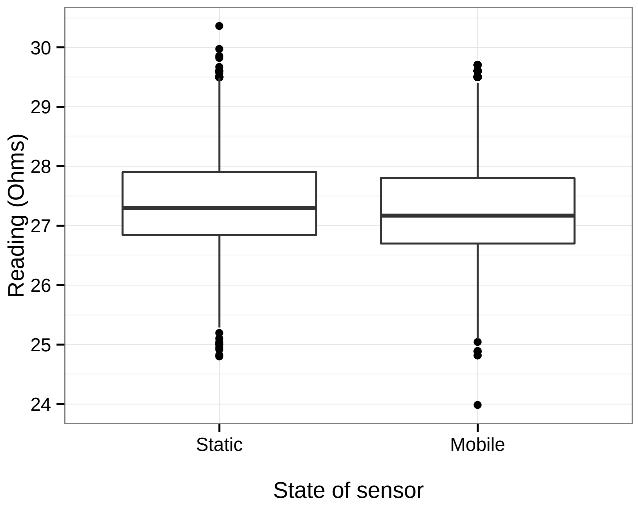


Figure 4.9: Analysis of the variability of mobile sensor readings: static vs. mobile sensor.

Figure 4.10 shows that the actual sensor orientation has little impact on the data capture process, being the differences between different orientations minimal. Anyway, the backward orientation option exhibited greater resemblance with the static measurements and was adopted for the tests that follow.

4.3.2 Impact of time sampling on geostatistical predictions

In this section, we analyze the impact of time sampling on the predicted pollution map. In particular, we want to determine if reducing the number of samples allows making similar predictions or if, on the contrary, there is a significant prediction error when generating the pollution map. For this purpose, we monitored the Technical University of Valencia campus with a mobile ozone sensor installed on a bike.

To obtain an accurate distribution of ozone levels, we monitored the entire campus by setting the sampling period to the lowest value allowed by the sensor (5 seconds). Next, we reduced the sampling frequency by setting the inter-sample period to 10, 20, 30, 40 and 80 seconds respectively. This was achieved by filtering the full trace, and retrieving datasets with 1/2, 1/4, 1/6, 1/8, and 1/16 of the data, respectively.

Next, we performed spatial interpolation through kriging for each trace, obtaining a detailed pollution distribution. We used the full trace (samples every

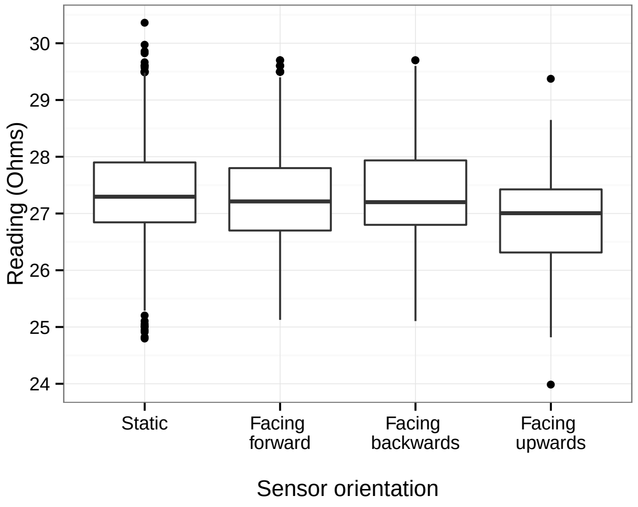


Figure 4.10: Analysis of the variability of mobile sensing for different sensor orientations.

5 seconds) as a reference and compared it against the results obtained using the other datasets.

Table 4.3: Statistical summary of the time sampling analysis.

Period	Mean	Std. Dev.	Similarity (s_i)
5 sec.	60.31251	1.140371	1
10 sec.	60.31928	1.158987	0.9734
20 sec.	60.37815	1.131514	0.9579
30 sec.	60.48890	1.118012	0.9411
40 sec.	60.36123	1.131782	0.9225
80 sec.	60.45629	1.126616	0.9181

Table 4.3 summarizes the statistical analysis for the different datasets in terms of mean, standard deviation, and relative prediction error, being the latter calculated using the initial trace (5s sampling) as a reference, as shown in equation

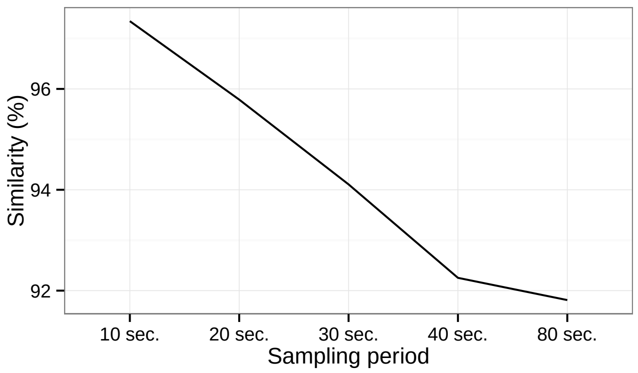


Figure 4.11: Analysis of the output similarity with respect to reference sampling period (5 s).

4.10.

$$s_i = 1 - \frac{1}{m \cdot n} \sum_{x=0}^{m} \sum_{y=0}^{n} \left| \frac{k_{i_{xy}} - k_{0_{xy}}}{\triangle k_0} \right| \tag{4.10}$$

In this equation, s_i represents the similarity index of dataset i with respect to the reference dataset, m and n represent the width and length of the target area under analysis, $k_{i_{xy}}$ represents the value calculated through kriging interpolation for dataset i at position xy, $k_{0_{xy}}$ represents the value calculated through kriging interpolation for the reference dataset at position xy, and $\triangle k_0$ represents the total variation of the predicted values for the reference dataset.

By analyzing Table 4.3 we can see that the mean and the standard deviation values are nearly the same in all cases, although the similarity index s_i varies more significantly. This information is also shown in Figure 4.11 for the sake of clarity. Notice that, despite the distribution of values is similar, the mean similarity shows an almost linear decrease. Nevertheless, the similarity values are still relatively high since the kriging interpolation process also acts as an error filter, helping to approximate the mean value when lacking enough reference values.

Detailed heat maps for some relevant traces (5 seconds, 20 seconds, and 80 seconds) are shown in Figure 4.12. By taking a look at these heat maps, built through the kriging interpolation process, we can clearly see that the level of detail experiences a degradation. In particular, we find that, although the pollution maps for inter-sample times of 5 seconds and 20 seconds are quite similar, significant differences are observed when the sampling period grows to 80 seconds; for the latter case, the ozone distribution achieved is quite different from the one used as a reference (5 seconds). Based on these maps, it becomes quite clear that little

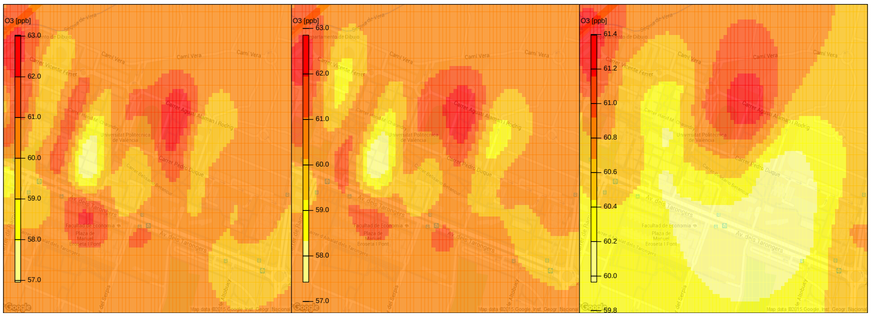


Figure 4.12: Heatmaps for the ozone distribution using different sampling periods (5, 20, and 80 seconds).

differences in terms of basic statistical analysis can represent huge differences in terms of the spatial distribution of those values.

4.3.3 Impact of spatial sampling on geostatistical predictions

In this section, we analyze the impact of spatial sampling on the predicted pollution map. In particular, we want to determine to which degree taking a shorter, less exhaustive path throughout the target area (reducing the trip time and the number of samples, accordingly) affects the accuracy of the predictions made.

To find the optimal spatial sampling strategy we produce different datasets by deleting path fragments from the first trace. In detail, starting from the full trace (100% of the data), we deleted selected paths so as to produce shorter but yet valid trips, maintaining start and end locations. As a result, we obtained traces with 72%, 54%, 50%, 46%, and 42% of the data.

Similarly to the previous section, we perform, for each dataset, a statistical analysis of the resulting data, also obtaining the pollution heatmap generated through kriging interpolation, and calculating the similarity index using equation 4.10.

Table 4.4 presents the statistical analysis results showing the mean, the standard deviation, and the similarity, being the latter calculated using the initial dataset as a reference.

Based on Table 4.4, we find that the mean value is close to the reference one (60.31) in all cases, although being in general slightly higher. This occurs because the first eliminated path showed the lowest values.

Figure 4.13 shows the decreasing trend when spatial sampling decreases. Compared to the time sampling results shown in Figure 4.11, we find that, now, the similarity values degrade much faster, meaning that reducing the route taken along the target area is prone to eliminate relevant samples, resulting in a less detailed pollution map.

Table 4.4: Statistical summary of the spatial sampling analysis.

Dataset size	Mean	Std. Dev.	Similarity (s_i)
100%	60.31251	1.140371	1
72%	60.49253	1.000335	0.9336
54%	60.62813	1.112316	0.9084
50%	60.66518	1.137273	0.8820
46%	60.66079	1.137295	0.8872
42%	60.51269	1.082692	0.8530

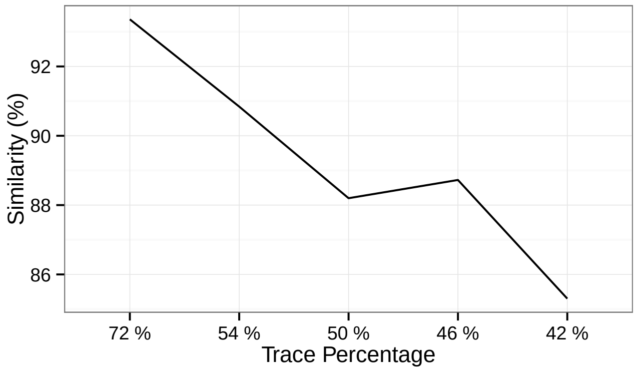


Figure 4.13: Analysis of the similarity with respect to the reference trace (100% of the data used.

Figure 4.14 shows detailed maps for datasets representing 100%, 72%, 50%, and 42% of the data. Based on these heat maps, we can indeed observe how spatial subsampling causes distortion on the spatial distribution of pollution throughout the target area.

Overall, we can conclude that the spatial sampling granularity is the most relevant factor to take into account, being time sampling granularity less but yet somehow important, and sensor orientation the factor having less impact on results.

4.4 Validation of the proposed approach

As stated at the beginning of the chapter, the current infrastructure elements allow measuring pollution levels in cities with high accuracy, although with a

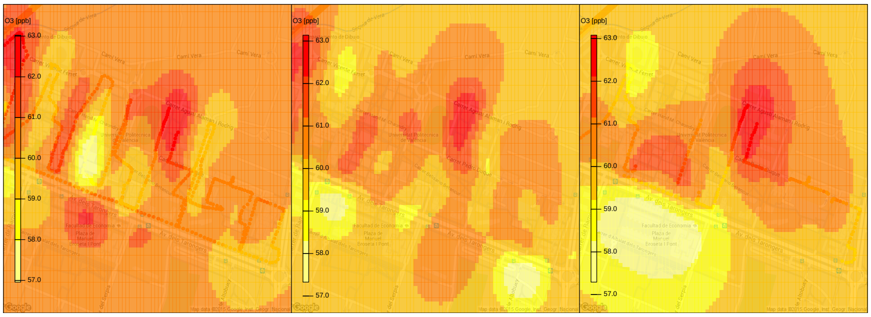


Figure 4.14: Heatmaps for the ozone distribution using different fraction of the original trace (100%, 72%, and 42%).

low spatial resolution. On the contrary, our proposed mobile sensing approach is able to achieve a much higher spatial resolution using cheap sensors. Thus, in this section, we validate our approach by first comparing captured values with the range of typical values obtained for this time of year by fixed monitoring stations, and then by comparing the ozone maps generated when relying on either infrastructure-based or mobile-based sensing.

We started by gathering data from different areas of Valencia using the proposed mobile sensors platform. Different experiments have been conducted at different times, allowing us to compare the data captured with the data from the existing public infrastructure. In particular, for each route taken, we first reduced the data variability using the proposed low-pass filter (see Equation 4.2). Next, the measurements were adjusted through Equation 4.3. Finally, the temporal dependencies of data were reduced according to Equation 4.7.

Figure 4.15 shows data for a particular route, and the common values at the date of the capture (February 16, 2015). We can see that the measured ozone levels are within the range of historical values for the monitored time, being quite close to the expected value (mean). This indicates that, using our methodology, we are able to obtain reliable data despite using low-cost sensors, allowing to focus our analysis on the spatial distribution of pollutants.

We now proceed to compare the actual heat maps for a specific date and time of day using only infrastructure data, and using only data obtained by our sensors. We can see that, by relying on our proposed architecture (see Figure 4.17) it becomes possible to observe in detail even small pollution variations, while using only infrastructure-based data (see Figure 4.16), the observed variations are much smoother, experiencing a linear increase or decay from one air quality station to the other.

Overall, it becomes clear that, despite having up to 5 different stationary air quality stations in the city of Valencia, they fail to capture significant details

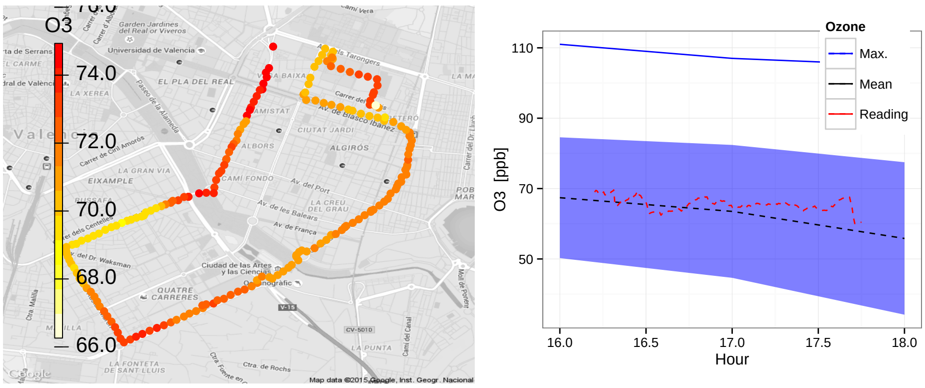


Figure 4.15: Comparison between captured data and typical values for the day/time of the year.

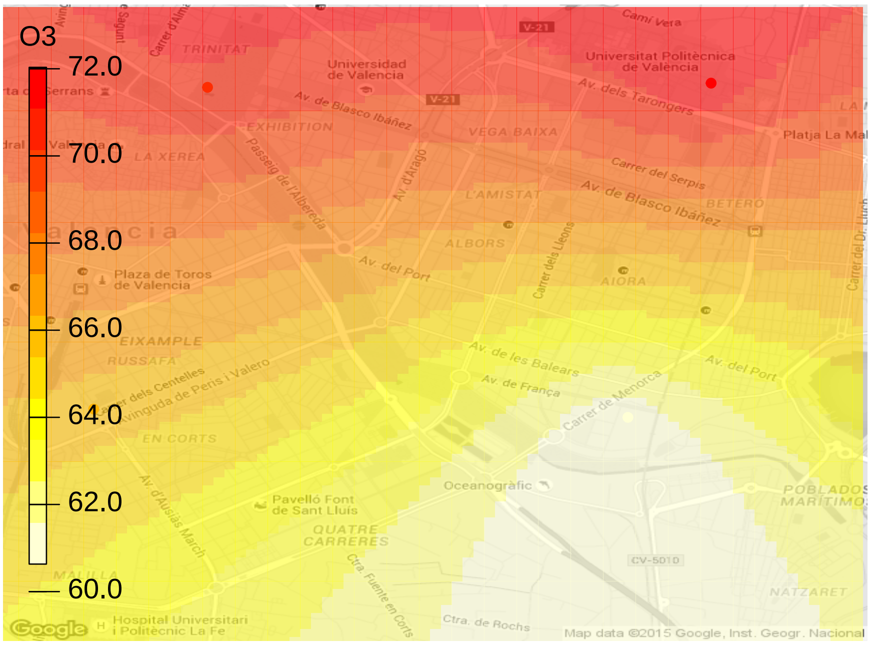


Figure 4.16: Ozone levels in the target region using infrastructure data.

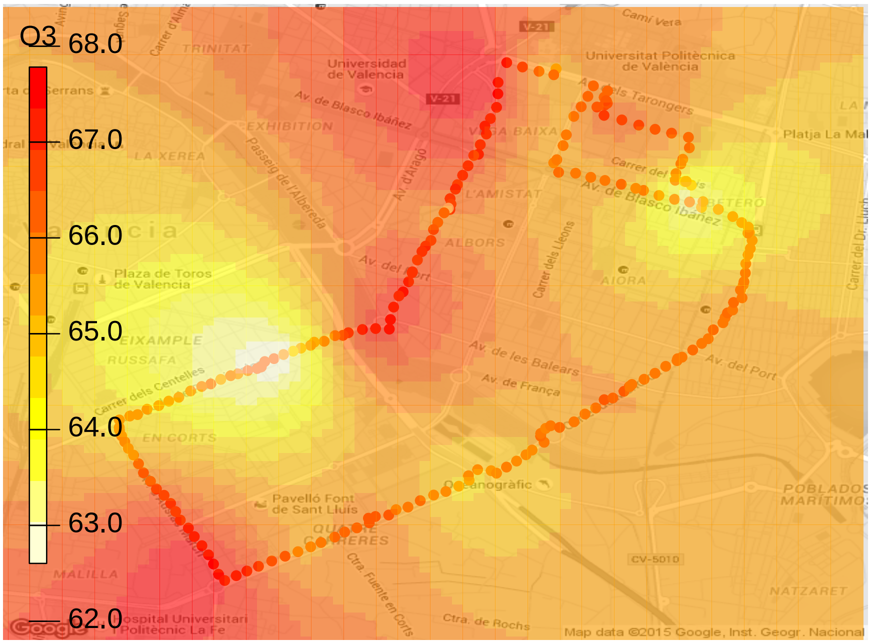


Figure 4.17: Ozone level in the target region using mobile sensing data.

that are related to areas with more traffic congestion (high pollution values) or green/windy spaces (low pollution values), thereby leading to some wrong conclusions. In contrast, our approach is able to provide a greater richness since all small variations can be perceived with great detail, thereby meeting the proposed goal.

4.5 Summary

Nowadays, environment pollution monitoring has become a fundamental requirement for cities worldwide, and there are many studies related to it. Nevertheless, only a few of them explore all sides of this problem.

In this chapter, we proposed a complete architecture for environmental monitoring that combines low-end sensors, smartphones, and cloud services to measure pollution levels with a high spatial granularity. In detail, we used a mobile sensor to provide pollution measurements, a smartphone providing real-time feedback about air quality conditions, and also acting as a gateway by uploading gathered data to the cloud server, in addition to the cloud server itself, required for data processing and visualization.

Once the architecture has been defined, we analyzed different issues related to the monitoring process: (i) Filtering captured data to reduce the variability of consecutive measurements; (ii) Converting the sensor output to actual pollution levels; (iii) Reducing the temporal variations produced by the mobile sensing process; and (iv) Applying interpolation techniques for creating detailed pollution maps.

To address the challenges associated with taking mobile measurements in a target area, we analyzed the influence of the sensor orientation in the data capture process, as well as the impact of time and spatial sampling. In particular, we varied the sampling period and the overall path length to determine the most effective monitoring strategy. Experimental results show that the sensor orientation and the sampling period, within certain bounds, have very little influence on the data captured, while the actual path taken has a greater impact on results, especially when estimating the distribution of pollutants throughout the target area.

Finally, we validated our proposal by comparing the values obtained by our mobile sensor with typical values from monitoring stations at the same dates and location. Furthermore, we compared the resulting heat maps generated using data from monitoring stations against ours, showing that our mobile-sensing approach is able to provide a much higher data granularity.

Chapter 5

GRC-Sensing: An Architecture to Measure Acoustic Pollution Based on CrowdSensing

NOISE pollution is an emerging and challenging problem for all large metropolitan areas, affecting the health of citizens in multiple ways. Thereby, obtaining a detailed and real-time map of noise in cities becomes of utmost importance for authorities to take preventive measures. Until now, noise measurements were limited to occasional sampling made by specialized companies and focusing mainly on major roads, and near airports.

In addition to the current solutions for measuring noise pollution, the pervasiveness of smartphones, together with the increasing integration of new sensors (e.g. ambient light, accelerometer, proximity, among others), has paved the way for a new paradigm called mobile crowdsensing [44, 47]. The concept of crowdsensing is that users of mobile devices participate by contributing with some environmental data obtained through their devices, being these measurements then stored and subsequently handled using data fusion and data analysis techniques. Allowing data potential with high granularity in space and time.

In the literature, we find initial solutions that use smartphones to measure ambient noise in urban areas [79, 62, 105, 128, 108]. Many of these solutions rely on a participatory sensing [72, 40] approach, detailing their architecture and integration with real-time collection solutions.

In view of the previous research studies, it is clear that, in spite of the many advances in the field of mobile crowdsensing in recent years, there are still a number of issues that must be addressed adequately for solutions to become more effective and, from the user perspective, it is clear that these tasks should not become a burden. Therefore, it is necessary to consider new crowdsensing architectures that include: (i) retrieval of accurate noise samples using smartphones; (ii) the management of context assessment to accelerate the discarding of unwanted samples; (iii) automated task management on the client side for seamless, transparent participation; and (iv) a fully-fledged server solution allowing to easily define and disseminate tasks, collect data, and visualize the resulting pollution levels. Our proposed architecture achieves these goals, empowering participating citizens by allowing them to seamlessly, and based on their context, sample the noise in their surrounding environment. This allows us to provide a global and detailed view of noise levels around the city, including places traditionally not monitored due to poor accessibility even while using their vehicles.

The main contributions of the chapter are: first, we propose an integrated architecture to measure noise pollution based on crowdsensing. Second, we detail the main characteristics of our proposed architecture. Finally, we present some implementation details of our solution for the server and client side. Our solutions differ from previous proposals because it includes minimal user intervention and allows displaying pollution data using real-time heat maps.

5.1 Crowdsensing Architecture Overview

In this section, we propose a unique crowdsensing architecture for reading ambient noise. In particular, our proposed architecture defines a set of elements that allow monitoring of the environmental noise in real time. Our design includes both the Mobile Noise-Sensing Client (MNSC), and the Cloud Data Colletion (CDC). Those two elements are connected to each other through a Data Transmission Network. The data transmission network works on the full scope of our architecture, and it provides support for iterative CDC and MNSC. In general, the MNSC is composed of smartphones that will provide sound sensing operations by capturing noise data, which will be delivered in real time to the CDC. Also, the CDC can be a single server or a server farm that allows receiving, processing, analyzing and sharing the sensed data. In Figure 5.1, the proposed architecture for noise analysis through crowdsensing is shown. In particular, we will focus on ensuring the quality of the measurements taken, and on the representativeness of samples taken by smartphones, respectively.

5.1.1 Mobile Noise-Sensing Client (MNSC)

In this section, we detail the client-side solution. Our solution allows the ambient noise to be sampled with little user intervention. Once activated, the mobile application autonomously performs noise sampling through a service running in

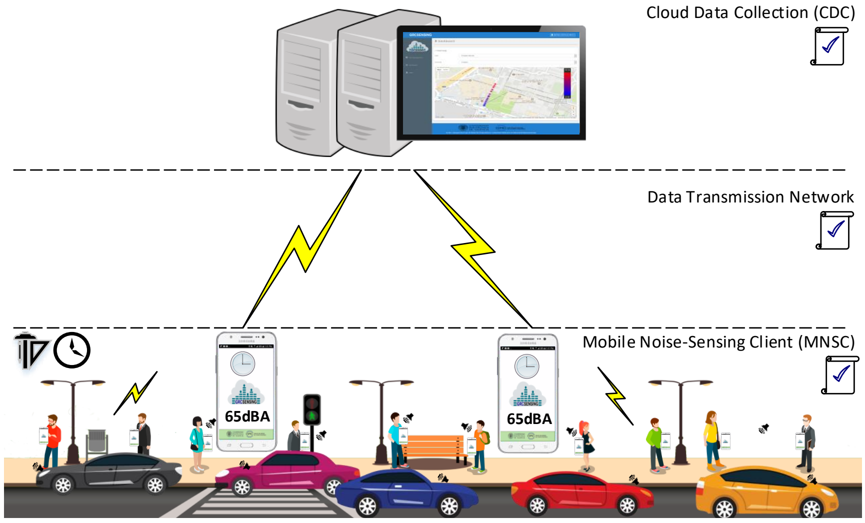

Cloud Data Collection (CDC)
Data Transmission Network
Mobile Noise-Sensing Client (MNSC)
65dBA
65dBA

Figure 5.1: General architecture for noise assessment based on crowdsensing.

the background. Besides, once the smartphone is in the desired context, the noise value retrieved is forwarded to the server for further processing. Figure 5.2 shows the main components of the MNSC. These are the Client User Interface (CUI), Client Sensor Task Manager (CSTM), the Client Data Manager (CDM), and the Client Communications Manager (CCM). Each of these four components has a controller responsible for supporting either bidirectional or unidirectional interactions between the different system elements. We now proceed to describe each of the client components in detail.

Client User Interface (CUI)

This component allows applications to interact with the user, and allows us to configure the different permissions associated with the operating system, such as: storage, audio, and location. Once the application startup processing ends, the splash screen closes and the main service is activated. Then, when the service has been activated and the necessary permissions have been granted, the application switches to background, and the graphical interface is cleared from memory.

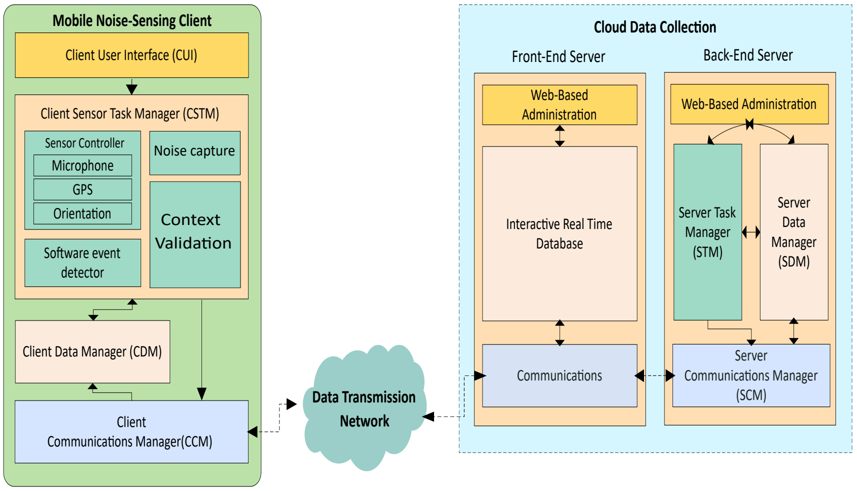

Mobile Noise-Sensing Client
Client User Interface (CUI)
Client Sensor Task Manager (CSTM)
Sensor Controller
Microphone
GPS
Orientation
Noise capture
Context Validation
Software event detector
Client Data Manager (CDM)
Client Communications Manager(CCM)
Data Transmission Network
Cloud Data Collection
Front-End Server
Back-End Server
Web-Based Administration
Interactive Real Time Database
Communications
Web-Based Administration
Server Task Manager (STM)
Server Data Manager (SDM)
Server Communications Manager (SCM)

Figure 5.2: Proposed crowdsensing architecture for mobile noise analysis.

Client Sensor Task Manager (CSTM)

The CSTM is the main component responsible for the noise sensing task. In general, the process is the following: once the CSTM is activated, and the server provides a new task, the CSTM is responsible for validating the different environments where the smartphone is located. Once this is done, the noise value is read and then sent to the server. The CSTM has three sub-components: Sensor Controller, Context Validate, and Software Event Detector. Below we describe each of its sub-components.

- *Sensor Controller.-* This component implements the access to the different sensors proposed in our architecture, thus providing access to gyroscope, accelerometer, microphone, and GPS. Also, this component allows data to be preprocessed before interacting with Validate Context. In particular, we process raw sound data in dB(A). In chapter 6 we show the details and strategies to obtain a correct and calibrated measurement.

- *Context Validate.-* This sub-component aims at determining the optimal strategy for collecting noise pollution data through smartphones. In general, it integrates an algorithm-based decision tree that interacts synchronously with the Sensor Sensor Controller (SC) and the Software Event Detector (SED). The purpose of the interaction with the CDM is to read the data of the task that is stored in the local database (SQLite). Regarding the interaction with SC and SED, its function is to obtain the previously processed values of the sensors (i.e. GPS, Orientation, etc.), and to perform calls to the operating system (i.e., determine whether the phone has the music player active), respectively. Once the answer is true for all cases, the algorithm proceeds to sample the noise. Also, the Context Validate has an algorithm that balances the use of tasks that are in the same range of date and time. Besides, to minimize energy consumption, Context Validate enforces a minimum time between trial and collected samples. Those times are given in the task from the CDC. In chapter 7, we show the implementation details of our collection strategy.

- *Sensor Event Detector.*- This component allows us to determine the different states of smartphones (i.e., playing music, speaker on, and smartphone performing active call) through a call to the operating system.

Client Data Manager (CDM)

The CDM performs two functions: (i) it allows us to store the tasks provided by the server; and (ii) it supports SQL queries to the CSTM. In particular, the tasks are received in one direction by the CCM. Also, to avoid data redundancy, the tasks are previously consulted and, if they do not exist, then they should be stored. Once the tasks are registered, they are available through a query based on the starting date and time and the final date and time for the CSTM.

Client Communications Manager (CCM)

The main purposes of this component are interaction with the server, such as receiving the tasks and forwarding the captured noise data to the server CDC. In the first task, once the listener service is activated, the tasks on the Front-End Server (Interactive real-time database) are replicated for each mobile device that is registered in the Front-End Server. Each new task is verified for later storage in the local storage. In the second case, the Front-End Server automatically offers the synchronization option between Mobile Noise-Sensing Client and Back-End Server. Figure 5.2 shows the direction of the communications that the proposed architecture components use.

5.1.2 Cloud Data Colletion (CDC)

The Cloud Data Colletion (CDC) is our server-side solution. The server provides a web interface that allows the administrator to control the tasks, and it also allows to have complete access to the information regarding trace management, processing, and visualization. Once logged in, the administrator can create a new collection task, or display the noise data provided by smartphone clients using heat maps. The architecture is composed of two servers: (i) Front-End Server, and (ii) Back-End Server, as shown in Figure 5.2. In general, the Front-End Server acts as an intermediary between the Back-End Server and the MNSC, while the Back-End Server provides the processing and delivery of collection tasks. Below, we provide more details about the specified components.

Front-End Server

The Front-End Server is responsible for carrying out the communications for sending/receiving data between the mobile devices and the Back-End Server. In general, it performs the intermediary functions between the proposed components. In particular, once the Back-End Server provides the task, it is replicated to all client devices that have the application activated. In contrast, once the clients supply the noise data, they are automatically sent to the Back-End Server. In our architecture, the Front-End Server includes three components: (i) Web-Based Administration, (ii) Interactive Real-Time Database, and (iii) Communication; these components are described below.

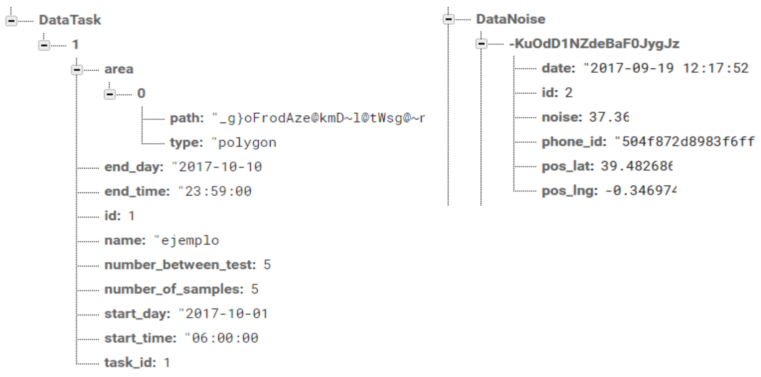
- *Web Based Administration.*- It is the administration console provided by Firebase [98]. Firebase is a Google solution that is integrated with our architecture in a simple and transparent manner through its API. Among other options (i.e., hosting, analytic, etc.), it allows us to manage the database in real time.

- *Interactive Real-Time Database.*- It is our real-time (NoSQL) database, whose format is JSON. In general, it is a gateway responsible for automatically sending / receiving sensing tasks towards smartphones. The tasks are previously defined in the Back-End Server, and are sent to this database whenever the administrator user requires it. The data sent and received are temporarily saved for the duration of the date range defined in each task. In particular, we maintain two "*DataNoise*" and "*DataTask*" objects within our JSON object. The first one stores the values of the task, and the second one stores the values of the captured noise. Figure 5.3 shows the attributes of our JSON objects.

- *Communications.*- Firebase uses a push communications model for sending data to specific recipients registered in its database. Generally, Firebase maintains a two-way open socket-based communications channel between the CDC and the MNSC.

Back-End Server

The Back-End Server provides a web interface which allows the administrator to have full access to the information gathered concerning trace handling, processing, and visualization. Additionally, it allows to define, schedule, and store the noise



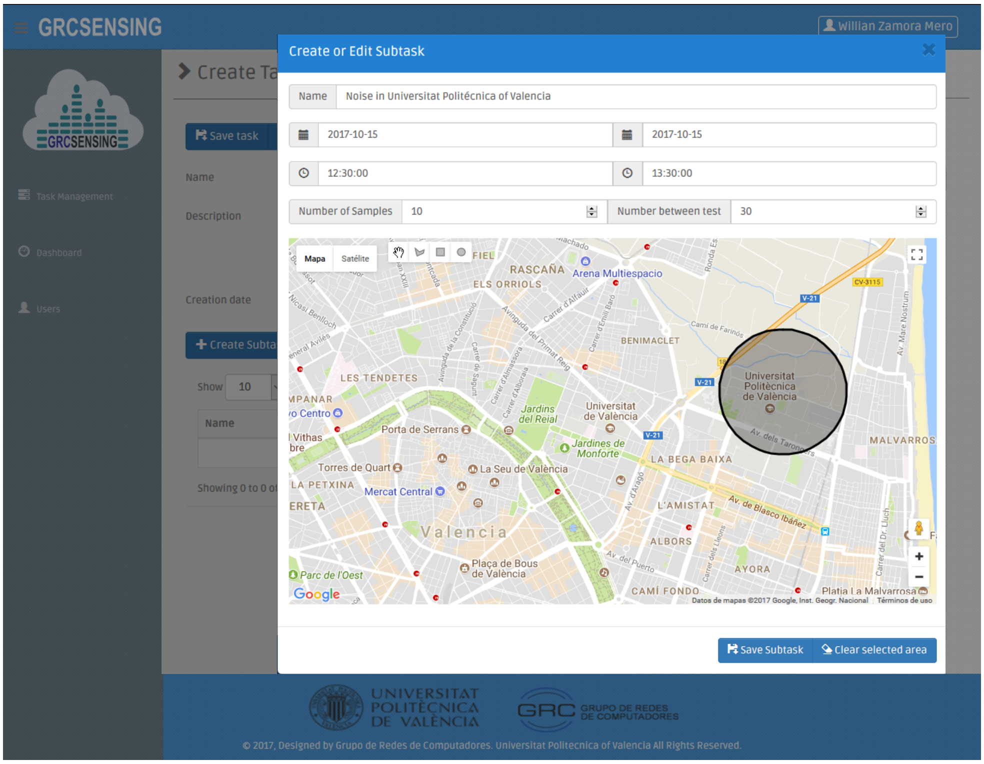
```
DataTask
  1
    area
      0
        path: "_g}oFrodAze@kmD~l@tWsg@~r
        type: "polygon
    end_day: "2017-10-10
    end_time: "23:59:00
    id: 1
    name: "ejemplo
    number_between_test: 5
    number_of_samples: 5
    start_day: "2017-10-01
    start_time: "06:00:00
    task_id: 1

DataNoise
  -KuOdD1NZdeBaF0JygJz
    date: "2017-09-19 12:17:52
    id: 2
    noise: 37.36
    phone_id: "504f872d8983f6ff
    pos_lat: 39.48268
    pos_lng: -0.34697
```

Figure 5.3: Format of a JSON data message.

sensing task. The Back-End Server has four components: Web-Based Administration, Server Task Manager (STM), Server Data Manager (SDM), and Server Communications Manager (SCM). Below we describe in more detail the different components on the server side.

- Web-Based Administration.- This component allows the user to manage and schedule noise sensing tasks interactively. It also supports the visualization of charts (heat-map) relative to the sensed data. Both functionalities are performed using a web-based graphical interface, meaning that the system manager can operate remotely. The site is available at http://www.grcsensing.net, and its design is shown in Figure Figure 5.4. , where the administrator (among other users) can create sensing tasks in specific areas, as shown in Figure 5.4.

- Server Task Manager (STM).- Task Management is one of the main components of the Back-End Server according to our proposed architecture, being responsible for scheduling planning, and pushing crowdsensing noise tasks. For the definition of the tasks, we have created two attributes: one for the waiting time between attempts, and another one for the time between samples. The purpose of these features is to minimize the consumption of resources in the tasks handled by smartphones. Also, we have enabled three types of geographic area selection procedures (polygons of n sides, rectangle, and circle) for the capture of environmental noise. Finally, once created, the tasks are stored by the SDM, and they can be forwarded to the Front-End Server when the user administrator considers it necessary. Figure 5.4 shows an example where a noise gathering task is created, including area selection i.e., using a circle in the figure.



Name: Noise in Universitat Politécnica of Valencia

2017-10-15 | 2017-10-15

12:30:00 | 13:30:00

Number of Samples: 10 | Number between test: 30

© 2017, Designed by Grupo de Redes de Computadores. Universitat Politecnica of Valencia All Rights Reserved.

Figure 5.4: GRCSensing Web-based application.

- Server Data Manager (SDM).- This component is responsible for the processing, storage, query, and analysis of the noise sensing task.

- Server Communications Manager (SCM).- The SCM is the Rest API that allows communication between the Back-End Server and the Front-End Server. In our architecture, we used an unidirectional interaction between SCM and STM, and a bidirectional one between SCM and SDM. The interaction with STM is unidirectional as we have transmission towards the Front-End Server when pushing new noise sensing tasks. The communication with the SDM is bidirectional since, when the API is notified of the existence of a new registry (data capture), it is first consulted before being inserted. Once the record has been inserted into the database, the SCM proceeds to delete the record in the Front-End Server. In particular, we have used Pyrebase [33], which is an API written in Python. Figure 5.2 shows the communication between these components.

Table 5.1: Size of the message with different selection areas and captured noise samples.

Data JSON	Description type	Message size (bytes)
DataNoise	Normal data	236
DataTask	With a polygon of 5 points	363
	With a polygon of 10 points	404
	With a polygon of 20 points	411
	With a circle (unencoded)	461
	With a rectangle (unencoded)	509

5.1.3 Data Transmission Network

This is the element responsible for the actual communication between the Cloud Data Colletion (CDC) and the Mobile Noise-Sensing Client (MNSC) devices through the establishment of end-to-end connections. Typically, reliable TCP connections are established. In particular, we use the Firebase API, which supports high-level communications, by automatically opening sockets. At the client side, Firebase establishes its communication through generic sockets, so that it guarantees compatibility with all smartphones, while at the server the connection relies on a REST service, which specifically uses the "Pyrebase" library, a simple Python wrapper for the Firebase API. Additionally, Firebase includes in its API the persistence option on disk, which means that, if the mobile device loses the network connection, Firebase will cache the captured noise values and, when the connection is available, it synchronizes the data that was previously cached with the server. Table 5.1 shows the size of the message when the clients send data (DataNoise), and when the server sends the task (DataTask).

5.1.4 Implementation

Our MNSC app, called GRCSensing, was developed using Android Studio 6.01. Besides, a set of Google dependencies (i.e. maps, Firebase) have been used for the coding and decoding details of GPS positions and Firebase APIs. Once the mobile application is installed, and the required permissions are granted, it automatically starts capturing data regarding the ambient noise. Those samples are sent in real time to the CDC. We use SQLite as the local data structure for storage.

Regarding the Server solution, specifically the Back-End Server, it is designed following the Model View Controller (MVC) pattern, and using the Django platform [99]. Also, we make use of JavaScript to add several functionalities, like those using Google maps, and to draw the different types of areas required. The database used is MariaDB [81], and the Pyrebase API [33] is used for communications. For statistical analysis and reporting, we use the R Graph tool [102], which includes the generation of heat maps for captured noise.

5.2 Summary

Currently, crowdsensing solutions have become an enabling technology for Smart Cities by empowering users to participate in the monitoring process of their environment through their mobile devices. Studies of noise pollution over densely populated areas is no an exception, with different works pointing out in this direction.

In this chapter, we proposed a complete architecture for environmental noise monitoring that combines smartphones and cloud services to measure noise pollution levels with high spatial granularity. In detail, we proposed using smartphones as mobile sensors to provide noise pollution measurements, and relying on Firebase as a gateway technology, allowing the interaction between the sending of sensing tasks at back-end servers, and the noise capture (by smartphones) at client devices. Once the task is delivered, the smartphone decides the optimum time for capturing data, and it provides real-time feedback on the given noise quality conditions; finally, the back-end server provides services for storage, processing, and data visualization. In the following chapters, we will analyze different issues related to the quality of the measurements gathered, and to the sampling process itself.

Chapter 6

Accurate Ambient Noise Assessment Using Smartphones

NOWADAYS, smartphones have become ubiquitous, and one of the main communication resources for human beings. Their widespread adoption is due to the huge technological progress, and to the development of multiple useful applications. Their characteristics have also experienced a substantial improvement as they now integrate multiple sensors able to convert the smartphone into a flexible and multi-purpose sensing unit. The combined use of multiple smartphones endowed with several types of sensors offers the possibility to monitor a certain area with fine spatial and temporal granularity, a procedure typically known as crowdsensing, and that was introduced in chapter 2.

In this chapter, we propose using smartphones as environmental noise-sensing units. In chapter 3, some related works on this topic were described. In particular, participatory solutions such as NoiseTube [79], Ear-Phone [104], and NoiseSPY [62] use smartphones to measure environmental noise levels in urban areas, generating pollution maps that use internal sensors for geo-localization. Other options are available in the Android and iOS application markets. However, these applications always introduce some error margin regarding their measurements, and many of them are not freely available for download and testing [63, 88]. The main causes of these errors are two-fold: first, the obvious hardware differences between the smartphones' microphones and the professional sound level meters; second, the specific algorithms, filters and the sound API that are used to process the sensed values. Thus, a thorough study and comparison of the noise measurement performance when using smartphones becomes necessary.

Our work differs from the previous ones since our goal is to study noise measurement accuracy when only smartphones are used. We developed our own noise measurement application considering current solutions, as well as different algorithms proposed in the literature. Then we determine the best configuration option for each of the algorithms in terms of sampling rate and block size. In addition, our study also analyzes which approach provides the best trade-off between accuracy and computational requirements, allowing one to determine the most adequate algorithm for application deployment, along with the time sampling period defined for sensing. For this purpose, we focus on the main characteristics influencing the design and implementation of reliable systems for the assessment of noise pollution levels using smartphones. We have taken our proposed architecture for crowdsensing-based noise measurements that were presented in chapter 5 as a starting point, proposing a mobile noise sensing solution based on off-the-shelf smartphones that can offer performance comparable to the one achieved with professional devices. In general, we will focus on the Noise Capture module of the Mobile Noise-Sensing Client (MNSC).

The main contributions of this chapter are as follows: first, we analyze the behavior of three different algorithms for noise measurement, determining the best approach when taking as reference a professional and fully-calibrated Class II sound level meter [77]. Second, for each algorithm, we evaluate the impact of different sampling rates and sample block sizes. Third, after selecting our candidate algorithm, we make a performance analysis using different types of smartphones, improving the results based on linear regression techniques, and then evaluating the optimal time-sampling period. Finally, we validate our proposal in typical outdoor environments.

6.1 Measuring Noise Level

In this section, we focus on a procedure that leads to accurate noise level measurements using Android-based smartphones. Such a procedure gains particular relevance within the framework of our reference crowdsensing architecture for noise data gathering. It is also important to note that, in this chapter and the following one, we will only focus on the Mobile Noise-Sensing Client (MNSC). In particular, we will focus on ensuring the quality of the measurements taken, and on the representativeness of samples taken by smartphones, respectively. This characteristic is in the "Noise capture" of the Client Sensor Task Manager (CSTM) analyzed in the previous chapter.

6.1.1 Measuring Background Noise Levels

In chapter 3 we analyzed the basic elements used to measure environmental noise. We now proceed to describe the applicable formulas. Thus, the measured value is expressed in decibels, and the equation used to derive the noise value is defined as follows:

$$L_{Ap} = 10 \log_{10} \frac{p_{rms}^2 \cdot \triangle filter}{p_{ref}^2} \tag{6.1}$$

where p_{ref} is the reference sound pressure, and p_{rms} is the root-mean-square sound pressure at a point during a given time interval. In addition, $\triangle filter$ refers to frequency-specific A-weighting coefficients [116].

In Equation (6.2), $L_{A_{eqp}}$ is the average sound level-equivalent for A-weighting applicable to a specific sample block of size BS.

$$LA_{eqp} = 10 \log_{10} \frac{1}{BS} \sum_{i=0}^{BS-1} \left[10^{\frac{LAp_i}{10}} \right] \tag{6.2}$$

The Sampling Rate (SR) or sampling frequency is the number of samples captured per unit of time, being an important factor when calculating environmental noise levels. The higher the sampling rate, the more accurate the voltage fluctuations, and, therefore, its shape more accurately resembles the original sound wave. Concerning the Block Size (BS), it is the total number of samples stored for processing in a single capture.

6.1.2 Android-Specific Issues

In the Android platform, there are two classes for managing sound resources, i.e., the AudioRecord and the MediaRecorder class. AudioRecord allows data analysis to be performed while recordings are still in progress. Such analysis is dependent on the minimal internal buffer provided by the read object. These raw data are then converted to standard noise levels. On the other hand, when the MediaRecorder is used, data are instead copied to a file, and it provides a method that returns the maximum absolute amplitude that was sampled since the last call. Both classes allow us to configure audio characteristics including sample rate, block size, input and output channels, etc. For our tests, we selected the AudioRecord class because it offers the possibility of reading directly from the buffer.

6.2 Noise Calculation Algorithms

As mentioned earlier, we can find a few free smartphone applications able to measure noise levels, although the majority suffers from large measurement errors. Some of these applications have used a method to calibrate the sampled values. A part of these algorithms also defined a fixed block size and sampling rate.

In this section, we study three different algorithms that allow measuring noise levels, with an emphasis on smartphones. In particular, we have considered three algorithms. The first one operates in the frequency domain, and the second and third algorithms operate in the time domain. All of these algorithms allow adjusting both the sampling rate and the block size, and they have been implemented in our Android-based application described later on.

Algorithm 1: dB(A) calculating using Fourier transform.

```
   Data: BufferRawData, AudioRecord
   Input: $SR(SampleRate); BS(BlockSize); T(Totaltime)$
1  $k = \frac{SR}{BS}$
2  for $c = 1$ to $T$ do
3      $dat = 0$
4      for $z = 1$ to $k$ do
5          Read current AudioRecord with Block Size BS
6          for $i = 0$ to $BS$ do
7              $w = \frac{1}{2}(1 - \cos(\frac{2\pi i}{BS-1}))$
8              $sample(i) = NormalizedRawData(i) \cdot w$
9          end
10         $FFT$(Array $sample$)
11         for $i = 0, j = 0; i < BS/2; i++, j+=2$ do
12             $mag(i) = \sqrt{FFT(j)^2 + FFT(j+1)^2}$
13             $dat = dat + 10^{\frac{L_{pmag(i)}}{10}}$
14         end
15     end
16     ToLogFile($10 \cdot \log_{10} \frac{dat}{k}$)
17 end
```

6.2.1 Fourier-Based Algorithm

This first solution, which is described in Algorithm 1, follows a procedure that consists of retrieving the raw noise measurements and transforming these values from the time to the frequency domain. Specifically, we used the Fourier transform with data previously standardized and processed using a Hamming window. Once the Fourier transform is applied, the data are evaluated and processed using Equation (6.2), applying the corresponding A-weighting coefficients for the different frequencies. Finally, after a cycle, the noise value is calculated using Equation (6.2).

6.2.2 Time Domain-Based Algorithm

This second solution, which is described in Algorithm 2, is one of the most widely known and adopted. It processes the noise levels in the time domain as follows:

values are read from a set of raw data and are then processed using Equation (6.1). In this equation, the value of the reference pressure (p_{ref}) is set to one. Once the cycle is completed, it calculates the average noise value.

Algorithm 2: dB(A) calculation using time series and non-normalized data.

```
   Data: BufferRawData, AudioRecord
   Input: $SR(SampleRate); BS(BlockSize); T(Totaltime)$
1  $k = \frac{SR}{BS}$
2  for $c = 1$ to $T$ do
3      $dat = 0$
4      for $z = 1$ to $k$ do
5          Read_AudioRecord(BS);
6          Apply equation for row data in buffer array;
7          $dat = dat + (\frac{1}{N} \sum_{i=0}^{BS-1} 10^{\frac{L_{p_i}}{10}})$
8      end
9      ToLogFile($10 \cdot \log_{10} \frac{dat}{k}$)
10 end
```

6.2.3 Normalized Time Domain Algorithm

This third algorithm shares the same basic characteristics as Algorithm 2. The main difference lies in the fact that samples are normalized before the noise estimation is made, so that the referential pressure of Equation (6.1) now becomes p_{ref} = 20 Pa (0.00002 N/m^2), which is the standard reference value for sound pressure.

6.3 Calibration Procedure

Our experimental scenario analyzes the accuracy of the three algorithms mentioned above in actual smartphones, comparing them with the measurements obtained by a professional sound level meter. Correctly, we used the PCE-322A sound level meter [77] for smartphone calibration. This device has a condenser microphone, and it presents a response to sound pressure in the range between 20 Hz and 20 kHz. It achieves a precision level of ±1.4 dB, complying with the IEC61672-1 standard for Class 2 devices. Concerning the smartphone used for the testing, we selected a Samsung Galaxy S7 Edge (Model SM-G935F) running the Android 6.0.1 operating system. This smartphone incorporates two microphones on the top and the bottom, respectively. The main microphone is of good quality, integrating a low-noise input buffer and active noise cancellation [59]. It is worth pointing out that the microphones included in both professional sound level meters and smartphones are omnidirectional.

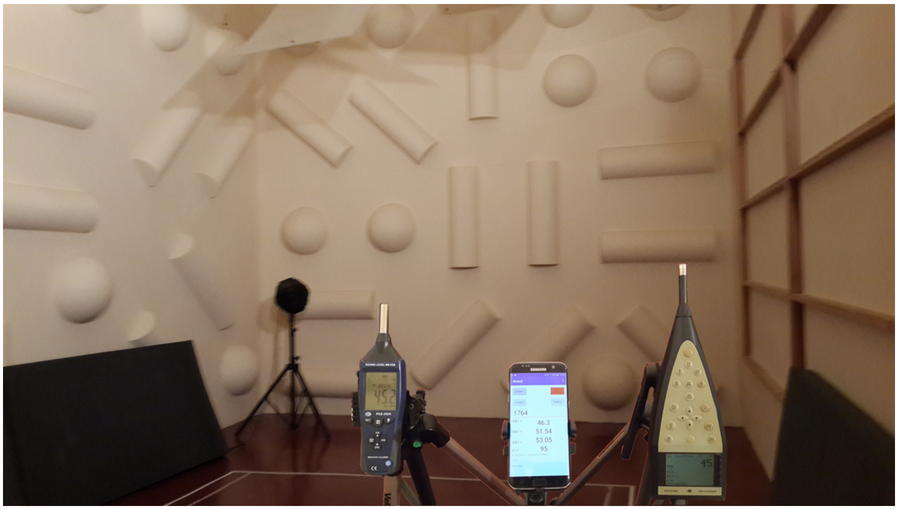


Figure 6.1: Noise calibration using professional devices. Location: reverberant acoustic chamber, at the Universitat Politècnica de València.

This PCE-322A sound level meter was calibrated using a Class 1 sound level meter in a reverberant acoustic chamber to guarantee the highest measurement accuracy, both of which are institutional resources with limited access and availability. Our purpose was to make sure that both sound level meters provide similar readings, as shown in Figure 6.1.

In our experiments, the sound source was aligned opposite to the smartphone and the sound level meters. A Creative Lab T30 2.0 speaker generated the noise. In particular, we injected pink noise generated by the Audacity software package in the range from 35 to 85 dB (A) in 10-dB steps, which is the typical dynamic range of microphones embedded in smartphones. The sound levels injected were calibrated using the PCE-322A device. Then, in our study, we implemented and tested the three algorithms defined in the previous section using different sampling rates (8 kS/s, 11 kS/s, 16 kS/s, 22 kS/s, and 44 kS/s), as well as different block sizes.

For each noise level being evaluated, we took a total of 30 short samples lasting 1 s each, and prior to each sampling procedure, there was an initial warm-up period of 15 s to achieve a stable measurement level.

Finally, it is worth mentioning that the algorithms were developed in Android Studio 2.1.3 using the AudioRecord class, and the 16-bit audio format was the one used by default.

6.4 Experimental Results

Once the algorithms were implemented and tested from a functional point of view, our next goal was to determine the impact of the sampling rate and the block size in the noise estimation accuracy. It should be noted that the sampling rate and the block size affect the accuracy of noise calculations due to two factors: frequency resolution, and filtering. The frequency resolution improves as the acquisition time increases. If the acquisition time is fixed, as in our case, a higher sampling rate will allow, according to the Nyquist theorem, to capture the behavior at higher frequencies, being that the maximum measurable frequency is half the sampling rate. This means that frequencies above this threshold are filtered out. However, if the sampling rate is increased, but the block size (number of samples acquired) is maintained, it will correspond to an overall lower acquisition time, and so the representativeness of measurements will be reduced, again resulting in a lower spectral resolution.

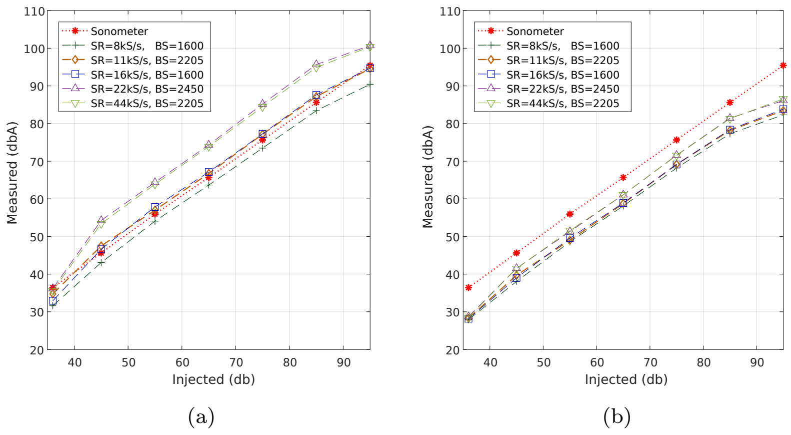
It is also worth pointing out that, in our case, we are not targeting at a very high spectral resolution, nor at encompassing very high frequencies; instead, our goal is to achieve representative noise measurements that mimic the sound level meter results with high accuracy, meaning that lower frequencies are the most representative in terms of noise assessment from the human ear perception system, and so this becomes our ultimate goal.

With this purpose in mind, we analyzed the results obtained by the different algorithms against the values stored by the PCE-322A device. In parallel, we analyzed the computation time associated with each algorithm to allow determining the best trade-off between estimated error and processing overhead. Also, once the candidate algorithm was obtained, we studied the variability when using different types of smartphones, and optimized results using linear regression techniques. Finally, we studied the sampling period that offers the best trade-off between sample length and accuracy.

6.4.1 Sampling Rate Analysis

In this section, we analyze the impact of the SR on each of the tested algorithms. In particular, we want to determine whether the actual sampling rate significantly affects the noise estimation. For this purpose, we carried out different tests where noise levels were collected from readings using slightly different block sizes (BS), where the block size refers to the number of samples gathered so that, combined, the ratio between SR and BS returns an integer value. As a result, BS values are in the range between 1500 and 2500 samples. Then, for each sampling rate and each block size combination, we determined the average noise value for all of the sampling periods, in dBA.

Figure 6.2 shows the impact of the selected sampling rates on each algorithm. In general, the three tested algorithms show a linear increase as the input noise



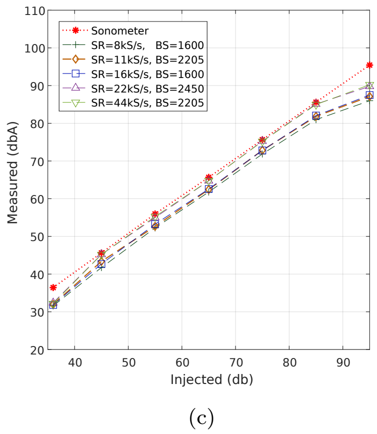


Figure 6.2: Sampling rate analysis. (**a**) Algorithm #1.; (**b**) Algorithm #2.; (**c**) Algorithm #3.

level varies. This is desirable, as a non-linear curve within the dynamic microphone range will mean that its quality is poor, possibly being unsuitable for this task.

Concerning the impact of the SR on noise estimations, we find that, in general, values tend to increase when increasing the SR. Moreover, we find that Algorithms #2, and #3 show values close to the reference ones when SR = 22 kS/s and SR = 44 kS/s, respectively, while Algorithm #1 shows better results for SR = 11 kS/s. In particular, it can be clearly observed that Algorithm #1 exhibits a significant difference concerning the reference value when SR > 16 kS/s. Overall, we find that

Algorithm #3 is the one showing the best match towards the reference values.

6.4.2 Block Size Analysis

In this section, we determine the impact of the BS on noise estimations. To this end, we fixed the sampling rate at 22 kS/s and made several tests with characteristics similar to the previous ones. Specifically, we used six different block sizes to evaluate each algorithm.

Figure 6.3 shows the behavior achieved for the noise range under test using different block sizes. In general, the three algorithms mostly present a linear behavior when the injected noise values increase. We find that Algorithm #3 best fits the reference value when BS = 882 samples. Regarding Algorithm #1, we find that the best result achieves for BS = 450 samples. Also, Algorithm #1 is the only algorithm that experiences a direct relationship between the block size and the estimated noise values (dBA). For Algorithms #2, and #3, this relationship was not so clear, although the lowest BS size still provides the lowest noise estimations.

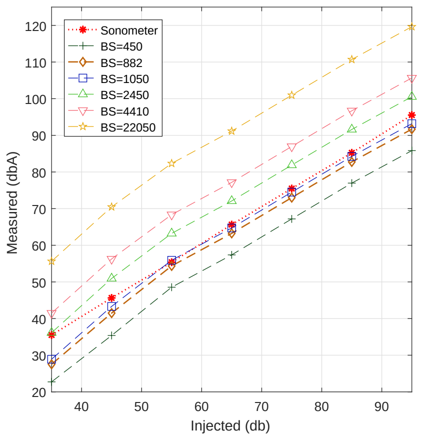
6.4.3 Algorithms Comparison when Fixing the Block Size

In this section, we want to extend our evaluation to the analysis of the impact of using different sampling rates when a same block size is adopted for all cases. In particular, we have assigned a default block size of 2012 samples to all measurement setups, which approximately corresponds to the mean size of the blocks evaluated in the previous section. In addition, we have discarded Algorithm #2 since the values it returned in the previous section were the most distant ones to our reference SLM values.

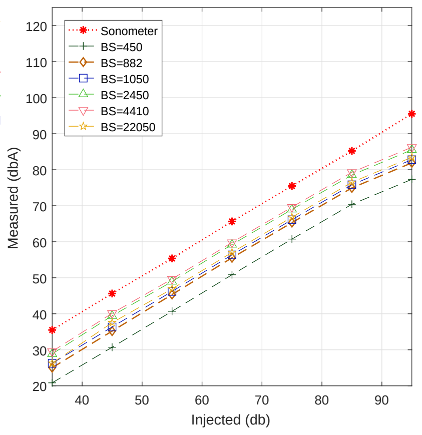
We consider that it is interesting to compare the behavior of Algorithms #1 and #3 due to their different noise calculation approaches, that is Fourier transform vs. time domain analysis, respectively. The samples were obtained with the same smartphone used for the previous tests, and the average value corresponding to 30-second periods was calculated for each noise level (35 to 95 dB) injected. Figure 6.4 shows the impact of the sampling rate on the noise estimation accuracy. We find that, in general, there is an increasing behavior with SR, meaning that increasing SR values causes the measured noise values to increase as well. We also noticed that for Algorithm #1, a nearly optimal curve adjustment achieved when using a sampling rate of SR = 11 kS/s, whereas for Algorithm #3, all of the returned values are slightly below the reference line provided by the professional sound meter.

6.4.4 Estimation Error (ϵ_s) vs. Computational Overhead

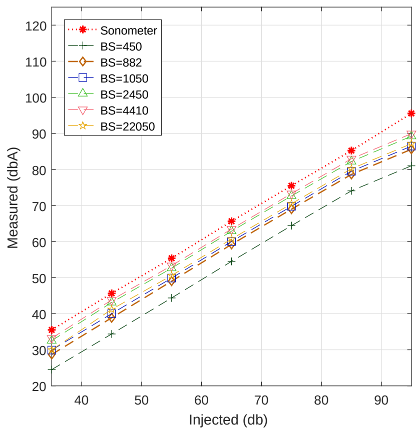
We now proceed to analyze which algorithm exhibits the lowest error with regard to the reference value, as well as their impact on the computational overhead. To



(a)

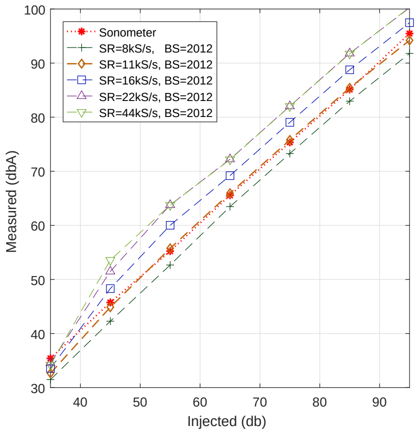


(b)

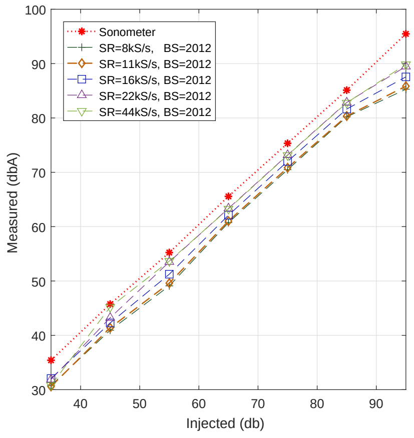


(c)

Figure 6.3: Block size analysis. (**a**) Algorithm #1.; (**b**) Algorithm #2.; (**c**) Algorithm #3.



(a)



(b)

Figure 6.4: Sampling rate analysis when fixing the block size. (**a**) Algorithm #1; (**b**) Algorithm #3.

calculate the error, we used a total period $T = 30$ s, with $\frac{SR}{BS}$ sample blocks per second, adopting the same characteristics as in the experiments detailed above.

Concerning the error value, it was calculated using Equation (6.3):

$$\epsilon_s = \mid \frac{i_s - R_s}{R_s} \mid \tag{6.3}$$

In this equation, ϵ_s represents the relative average error for the sample s with regard to the set of reference data; i_s represents the value calculated by the algorithm for sample s; and R_s represents the reference value of the sound level meter for sample s.

Concerning the calculation of the computation time, we obtained the average value corresponding to 1000 independent executions for each particular algorithm under the sampling rate and block size defined.

Table 6.1 presents a summary of the mean and maximum estimation errors achieved, along with the corresponding computational overhead. This table is completed based on the results obtained in the previous evaluations for different sampling rates and block sizes, but we have filtered this table showing the three best results obtained in each algorithm. Notice that Algorithms #1 and #3 have a mean error value of about 1 to 4%, and that they introduce a similar overhead. On the other hand, the computational overhead for Algorithm #1 is higher due to the fact that they need to perform the Fourier transform prior to determining the error. Additionally, we find that the maximum error for Algorithms #3 and #2 is higher (10%) when compared to the first algorithm.

Table 6.1: Estimation error and computational overhead achieved with the different algorithms (best overall configurations).

Sampling Rate (kS/s)	Block Size	Algorithm	$\bar{\epsilon_s}$	$max(\epsilon_s)$	Overhead (ms)
11	2012	1	0.01931	0.07796	25.1533
11	2205	1	0.02420	0.04589	25.3667
16	1600	1	0.03236	0.09625	25.7133
44	2205	3	0.03146	0.11664	25.4101
22	2450	3	0.03154	0.11048	16.0122
44	2012	3	0.04579	0.12980	25.0178
44	2205	2	0.09326	0.21741	25.3252
22	2450	2	0.09334	0.21124	16.1289
11	2205	2	0.12582	0.22328	11.1083

To have a greater insight into the different trade-offs faced, Table 6.1 shows the *SR/BS* combinations returning the lowest average error concerning the sample rates evaluated, along with the respective computation overhead. There is evidence that Algorithm #3 offers the smallest error margins, achieved for SR = 44 kS/s and SR = 22 kS/s, when having BS = 2205 and BS = 2450, respectively, achieving an error of less than approximately 4%. Algorithm #2 presented errors more than 4%. Finally, regarding Algorithm #1, it is indeed the one offering the best combination by achieving both low error values and low computational overhead, and so it is considered as the best candidate solution.

6.4.5 Analysis Using Different Smartphone Models

Once Algorithm 1 was found to be the best option, the next task was to evaluate the effectiveness of the selected algorithm when obtaining samples using different smartphone models. Figure 6.5 shows the smartphone models evaluated. All smartphones run the Android 6.0 operating system, and all them are able to run the developed application correctly.

In the experiment, we performed a procedure similar to that followed in previous tests. In particular, we injected pink noise in the range from 35 to 95 dB, and each test lasted 30 s.

Figure 6.6a shows the obtained results. In general, we find that, except for smartphone model BQ Aquaris (AQ), all other smartphones models (S4, J5, and S7) present a linear behavior. Regarding result accuracy, though, we find that only the results for the Samsung S7 device are near to the reference values. Such near-optimal accuracy is expected since the experiments performed earlier on relied on this same device. Thus, we find that the results achieved using the proposed

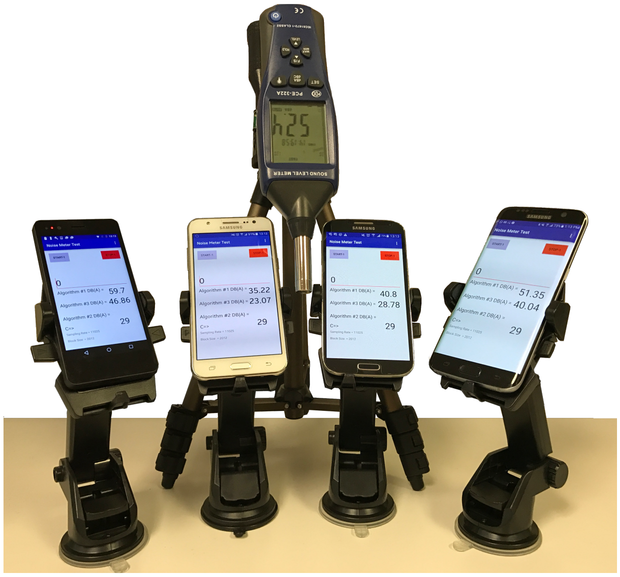


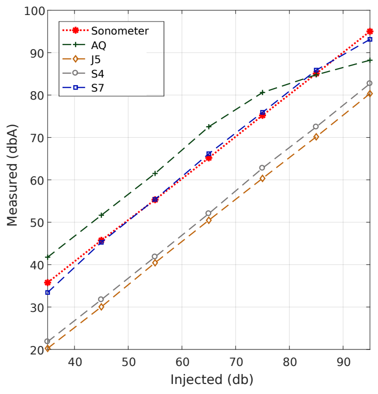
Figure 6.5: Smartphone models used for testing. From left to right: BQ Aquaris, Samsung J5, Samsung S4 and Samsung S7 Edge.

algorithm show model-specific variations, which are in general expected due to hardware differences.

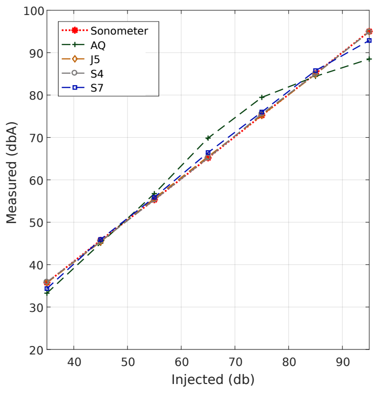
To solve the problem detected, our next step was to apply linear regression techniques with respect to the reference dataset. Our goal was to adjust the results achieved with the different smartphone models so that they resemble the reference ones as much as possible. The results of this curve adjustment process are shown in Figure 6.6b. It quickly becomes evident that the output results for most of the models fully agree with the reference value, with more pronounced differences for the BQ Aquaris smartphone case in the range from 65 to 75 dB (A). Finally, Figure 6.6c shows that, after the adjustment procedure, the error is less than 2% in Samsung phones, the BQ Aquaris model being the one showing the highest error values. Anyway, this error is always less than 8%, which is a reasonable value.

6.4.6 Analysis for the Same Smartphone Model

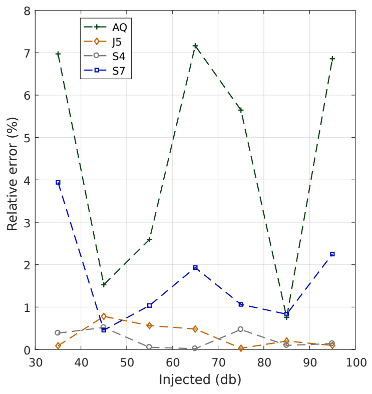
In this section, we compared the differences between smartphones of the same model/provider to determine the differences between them. Notice that differences between smartphones of the same model are expected, especially for low-range market devices, where cheaper hardware is used. For our tests, we picked four BQ



(a)



(b)



(c)

Figure 6.6: Estimation accuracy for the different smartphone models with and without linear regression. (**a**) Algorithm #1: default sampling; (**b**) Algorithm #1: values adjusted using linear regression; (**c**) Algorithm #1: estimation error.

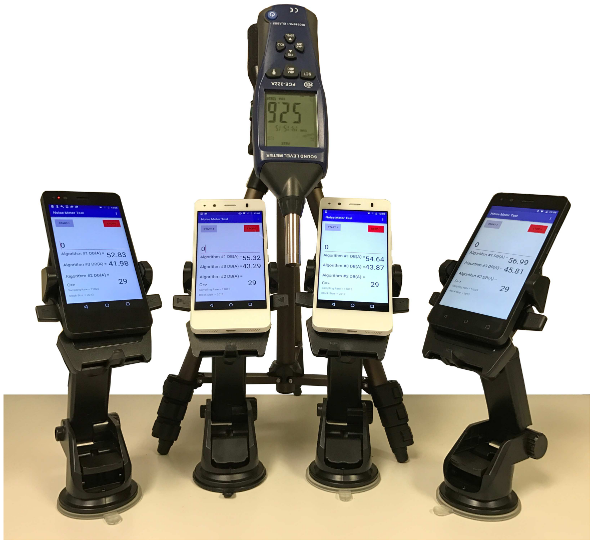


Figure 6.7: Smartphones of a same model used for testing.

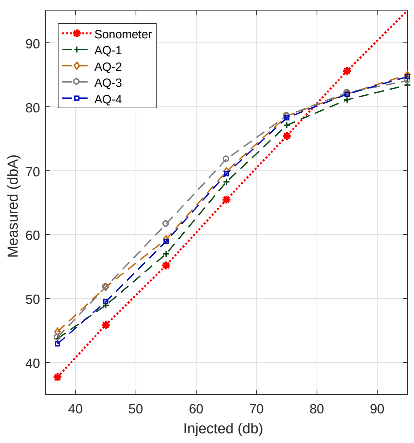
Aquaris smartphones since these are the cheapest ones used. Figure 6.7 shows the smartphones of a same model being evaluated.

Figure 6.5a shows the results for our noise sensing tests before applying any curve adjustment. The experiment was performed under the same conditions used before. We can see that there are some differences between smartphones, although the shape of the curve is similar in all cases, with a loss of linearity for values above 75 dB.

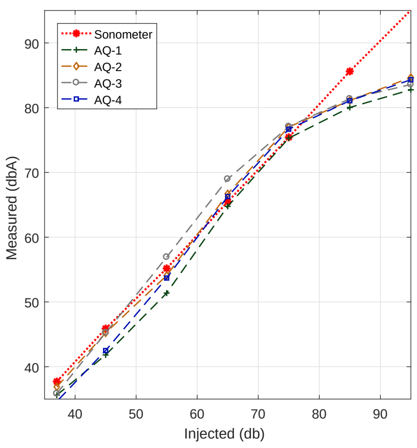
Figure 6.8b shows the output after performing the linear regression procedure. We can now see how values tend to resemble the reference values better, showing differences for inputs of 85 dB and above due to the non-linearity detected earlier. Finally, Figure 6.8c shows the value of the estimation errors, which are below 8% in most of the cases.

6.4.7 Impact of Reducing the Sampling Period

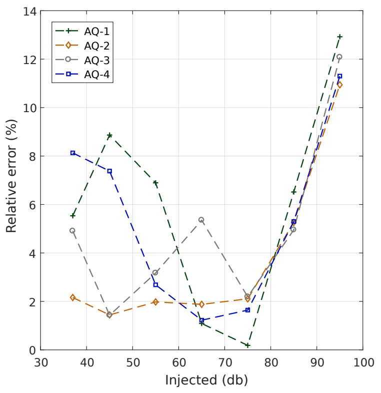
Our next goal was to determine the optimal duration of the data collection process. This is a critical issue when aiming at minimizing the resource usage of smartphones. With this purpose, we have evaluated the impact of the sampling time for two smartphone models: the low-end BQ Aquaris, and the high-end Samsung S7 Edge. For both devices, we have obtained 30 samples for each sampling period tested. In all cases, the block size used is 2012, and the sampling rate is 11 kS/s, as defined in previous sections. The injected noise was fixed at the intermediate



(a)



(b)



(c)

Figure 6.8: Estimation error analysis when using similar smartphones. (**a**) Before regression; (**b**) after regression; (**c**) estimation error.

value of 65 dB for all the tests. Figure 6.9 shows the sample estimations achieved with both smartphones when varying the sampling period, along with the associated error values. In particular, Figure 6.9 shows that the Samsung S7 edge smartphone introduces a measurement bias detectable when the interval is very low, meaning that a filter is introduced that causes values to converge to the final value slowly. This effect is not noticeable in the BQ Aquaris smartphone, as shown in Figure 6.9, which is much more linear. In both cases, we find that, as expected, higher sampling periods are associated with a lower result variability, although the margin of variation typically remains below ± 0.5 dB in all cases. Figure 6.9 shows the error towards the reference values for the two smartphones under test. Clearly, the S7 smartphone introduces an error that only becomes negligible when the overall sampling period is greater than 10 s. Therefore, it is important to determine the required stabilization time for the different smartphone models.

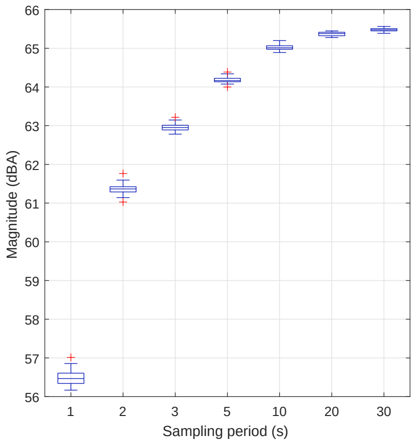
Based on the prior analysis, we decided to broaden our study to gain further insight into the sound capture process for the initial samples taken during the first second and thereby determine the duration of the transient period detected above. This procedure was repeated 30 times.

Figure 6.10 shows that, during this initial second of sampling, sample values gradually tend to increase, initially taking values below the reference value (65 dB), and then becoming slightly higher. The main problem detected occurs for the first sample taken with the Samsung S7 smartphone, where the values registered are more than 40 dBs below the reference value, thus causing a substantial bias when averaging these different samples. Given these circumstances, at least the first sample taken should be discarded in order to increase the reliability of the results and to achieve meaningful measurements in a short period.

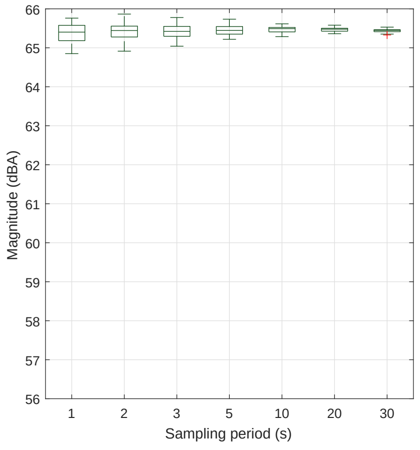
Figure 6.11a, b shows the new results when varying the sampling rate and discarding the first sample during this initial second. We find that there is a substantial prediction improvement, especially for the Samsung S7 device, which was the one experiencing the problem. We also find that the readings do not depend significantly on the period of time chosen.

Figure 6.11c now shows that the margin of error is below one percent for most of the cases. The causes of these results varies according to each specific smartphone; in particular, the microphone of the Samsung S7 device presents a filtering at the time of being activated, reason why we consider appropriate to discard the first samples. The worst result takes place when using a sampling time of one second using the BQ Aquaris model; anyway, the average error remains below 2%, which is considered acceptable for these types of measurements.

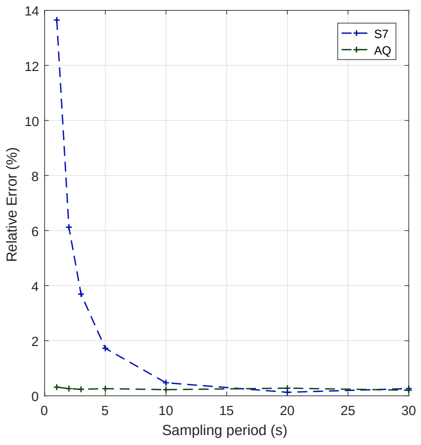
Overall, based on the results obtained, we can conclude that Algorithm #1 is an optimal candidate when used together with the following conditions: discard the first sample; adopt a sampling rate of 11 kS/s; select a block size of 2012 (evaluated); and apply a curve adjustment (linear regression) specific to each smartphone model. This strategy allows reducing the sampling error below 1% in most cases, even when limiting the sampling period to a very short interval of 1



(a)



(b)



(c)

Figure 6.9: Sampling period analysis. (**a**) S7 edge; (**b**) BQ Aquaris; (**c**) estimation error.

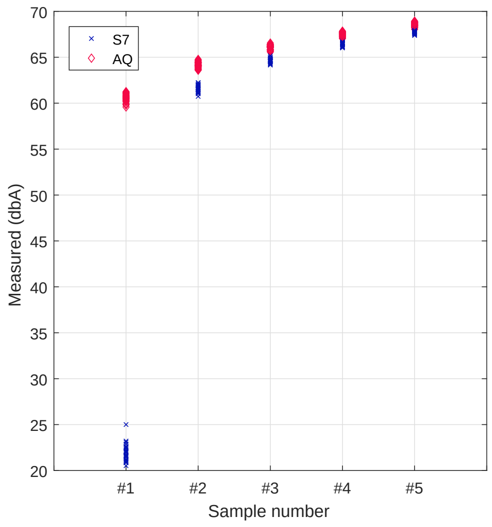


Figure 6.10: Analysis sampling S7 and Aquaris for 1 s.

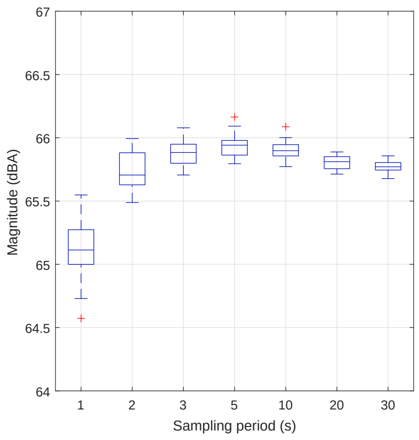
Table 6.2: Sampling period analysis (when removing the first sample).

Smartphone Type	**Sampling Period (s)**	**Mean (dBA)**	$\bar{\epsilon_s}$	**Std. Dev.**
Samsung S7 Edge	1	65.1229	0.0052	0.2355
	2	65.7344	0.0056	0.1382
	3	65.8856	0.0078	0.0979
	5	65.9405	0.0097	0.0890
BQ Aquaris	1	66.5757	0.0172	0.2270
	2	65.9599	0.0091	0.2128
	3	65.7659	0.0059	0.1880
	5	65.6615	0.0054	0.1506

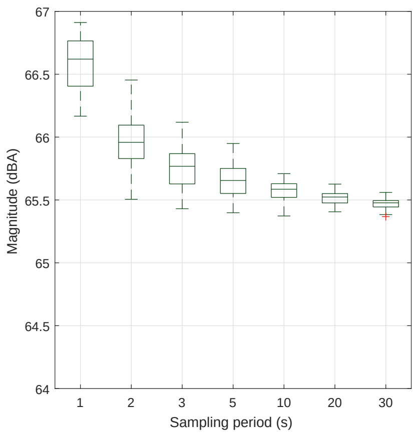
to 2 s. Finally, by analyzing Table 6.2, we show that the mean value obtained is close to the reference value generated for testing (65.5 dBA). We also find that, on both smartphones, the standard deviation experienced a moderate decrease when increasing the sampling time.

6.5 Validation in Real Outdoor Environments

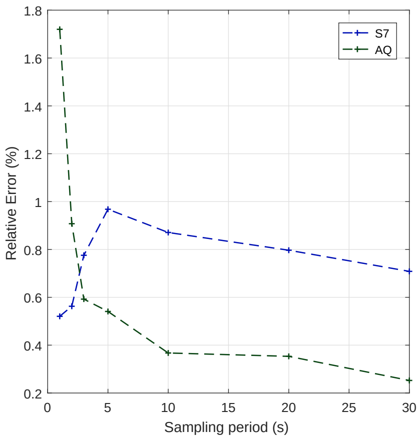
All of the results mentioned above were obtained in a controlled noise environment. However, field noise measurement results may experience significant differences due to environmental effects, such as humidity, temperature, and the stability of



(a)



(b)



(c)

Figure 6.11: Sampling period and error analysis when removing the first sample. (**a**) S7 Edge; (**b**) Aquaris; (**c**) error analysis.

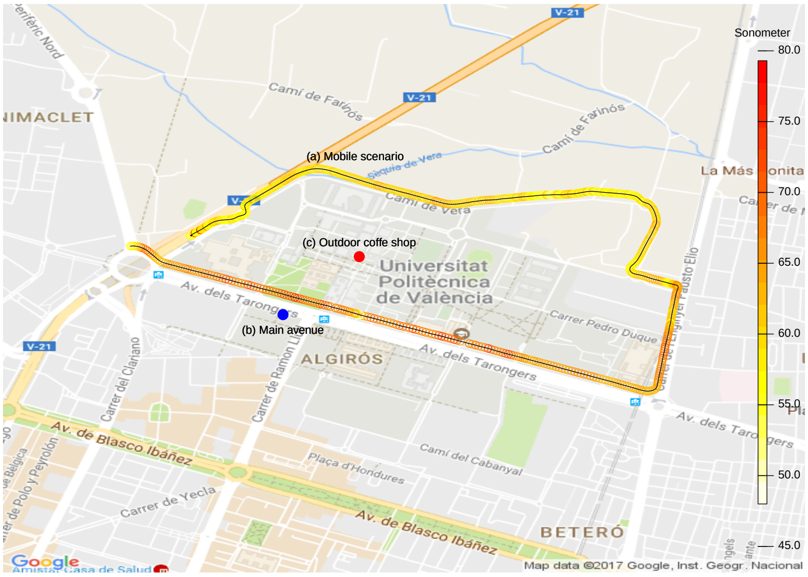


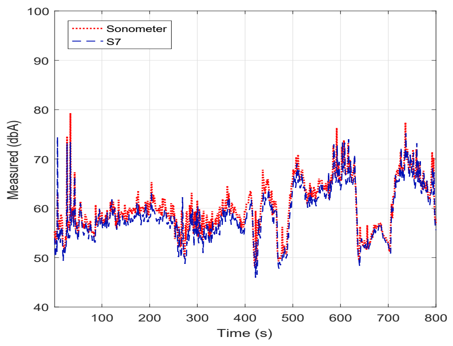
Figure 6.12: Analysis for the mobile scenario.

the device, among others. Therefore, to complete our study, our next goal was to validate the behavior of the candidate algorithm in real environments, comparing the results obtained with those of our professional noise level meter.

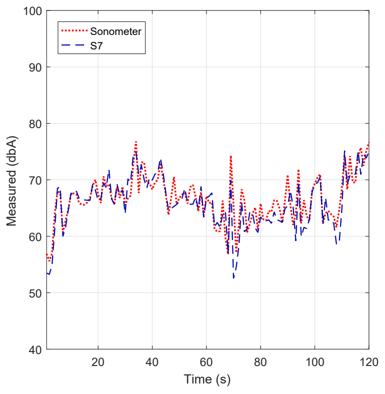
We decided to perform our study in three different outdoor conditions: the first experimental results were obtained under mobility with the support of a vehicle; the second set of results was obtained when statically positioned near the main avenue; finally, the third set of results was obtained at a crowded outdoor coffee shop. Figure 6.12 shows the locations associated with the three scenarios we used for validation.

In detail, the first set of results was obtained by having the noise level meter and the Smartphone S7 in a car with the windows down, both devices being statically positioned with an appropriate holding arm. We then followed the path shown in Figure 6.12 at an average speed of 20 km/h, the obtained results being shown in color in Figure 6.12, and graphically in Figure 6.13a. The latter results showed a behavior close to the ones obtained with the sound level meter, with a margin error of 0.0308 percent (see Table 6.3).

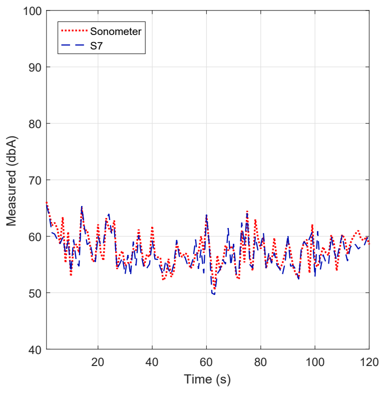
Regarding the second set of tests, we installed the equipment (noise level meter and smartphones) on one side of the main avenue (Avenida Tarongers nearly at the Universitat Politècnica de València), and in such a way that it remained static for the whole test duration. The measurements were taken at 17:00, a time when the traffic density was moderate. The results are shown in Figure 6.13b and Table



(a)



(b)



(c)

Figure 6.13: Analysis of noise pollution for three outdoor scenarios. (**a**) Mobile scenario; (**b**) main avenue; (**c**) outdoor coffee shop.

Table 6.3: Error in the three outdoor scenarios.

Scenario	$\bar{\epsilon_s}$	$max(\epsilon_s)$
(a) Mobile scenario	0.0308	0.2275
(b) Main avenue	0.0260	0.1989
(c) Outdoor coffee shop	0.0258	0.1151

6.3. Again, we can find a high resemblance when compared to the results produced by the professional sound level meter.

Concerning the third set of tests, the test equipment was installed on the table of an outdoor coffee shop within our university campus following a similar procedure. The results are shown in Figure 6.13c and Table 6.3. In particular, we find that the error margin was even lower compared to the previous results, with an error value of only 0.0257 percent.

Overall, we consider that the results obtained are quite satisfactory, and they evidence the adequateness of the proposed solution, as well as of the methodology adopted.

6.6 Summary

The widespread adoption of smartphones and the various sensors they integrate offers unlimited potential waiting to be unleashed. In particular, the combination of multiple smartphones acting as mobile sensing devices allows achieving massive monitoring, a process known as mobile crowdsensing.

In this chapter, we proposed using the microphones of smartphones as sensors to create noise level maps for metropolitan areas. In particular, we address the issue of measurement accuracy and representativeness when using the microphones of commercial smartphones. To this aim, we implemented and compared different algorithms typically used to obtain Type A noise levels, assessing the impact of the sampling rate and the buffer size on noise measurement accuracy. We also evaluated the impact that different issues could have over the measurement process: (i) the impact of testing with different smartphones models in the same conditions; (ii) the impact of same testing devices in the same conditions; (iii) the model-specific tuning by applying linear regression techniques; and (iv) the analysis of the trade-off between sampling period duration and noise estimation error.

Experimental results showed that both the sampling rate and the selected buffer size can have a significant impact on the accuracy of noise level estimations, being that estimation errors can vary from 1% to 12% in the best cases. However, although statistically representative, we consider that errors between 1 and 2 dB are acceptable for noise measurements in the majority of the scenarios.

More important, we find that, if an adequate selection is made, it is possible to combine low noise-level errors with a low computational overhead, a situation that is ideal in crowdsensing contexts. In addition, we show that low-end smartphones are prone to introduce a higher error than high-end smartphones on average, although the additional filters included in the latter may require some of the initial measurements to be discarded. Overall, we were able to demonstrate that it is possible to accurately assess noise levels by taking relatively short samples (from 1 to 3 s), while introducing a minimal estimation error.

We consider that it is important to take into account the differences between mobile phones from different manufacturers, which are mainly noticeable in terms of microphone quality and the presence of filters; the latter can cause unwanted effects, such as initial transient periods in the measurements obtained. Similarly, the choice of a specific configuration in terms of sampling rate and block size will have an effect on the noise measurements made, and should also be taken into account. Finally, the algorithms developed should try to compensate for non-linearities in noise measurement over the different frequency ranges.

In the next chapter, we integrate this algorithm in our proposed client-side application. Additionally, we evaluate how other smartphones sensors (i.e., gyroscope, accelerometer, orientation, GPS) can help us to determine the ideal time to capture environmental noise in urban areas.

Chapter 7

Determining the Right Time to Obtain Noise Samples

In the previous chapter we studied the viability of using commercial smartphones to determine noise levels, showing that, if a suitable calibration procedure is adopted, such an option is feasible.

In addition, some of the works reviewed in chapter 2 highlight that determining the sampling rate used and the right time to gather samples (by accounting for the context of smartphones) are both critical issues to consider when aiming at a widespread user adoption, as an excessive resource usage would make any application worthless. In fact, the sampling rate directly affects the use of the hardware resources in the smartphone, while smartphone context awareness is critical to determine the right time and place to capture the noise, avoiding samples with little or no representativeness. For example, it would be incorrect to measure noise when the smartphone user is playing music using the loudspeakers, or when the user is talking on the phone.

Also, another factor to consider in the design of smartphone applications is battery consumption. By avoiding taking noise samples at times where conditions are inadequate, it is possible to achieve significant energy savings, thereby extending the battery lifetime. Considering the aforementioned issues, crowdsensing solutions for ambient noise assessment require new approaches to data collection that can reduce user intervention and maximize energy efficiency.

In view of the previous research studies, it is clear that, in spite of the many advances in the field of mobile crowdsensing in recent years, there are still a number of issues that must be addressed adequately for solutions to become more effective and, from the user perspective, it is clear that these tasks should not become a burden. Therefore, it is necessary to consider the management of context assessment to accelerate the discarding of unwanted samples. The consumption of energy and network resources is also a problem, and so there is a need for intelligent algorithms capable of correctly determining the best sampling times, while simultaneously avoiding CPU-intensive tasks.

In this chapter, we focus on application design issues; specifically, we attempt to detect which is the most appropriate time and context to obtain these noise samples, while simultaneously minimizing power consumption. To this purpose, we define different contexts to determine whether adequate environment sampling conditions are met, and we then apply classification algorithms to generate the most accurate decisions trees automatically. An analysis of resource consumption requirements associated with the different decision trees [66] obtained shows that, despite their high accuracy, the resource consumption levels were prohibitive for this kind of applications. Thus, we propose an alternative decision tree that maintains the accuracy levels of automatically generated trees, while significantly reducing the resource consumption introduced by the latter.

7.1 Classification Techniques

During the mobile app design process, the developer defines different procedures to determine, e.g., when the sensors should be activated. Besides, certain conditions should be considered when using smartphones as a noise measurement tool. For instance, several situations make it inadequate to sense environmental noise, as user intervention is affecting the overall result. Examples of such situations include talking on the phone, listening to music using the loudspeaker, or keeping the smartphone in a pocket/purse.

In this chapter, we continue by optimizing the process taking place at the Mobile Noise-Sensing Client (MNSC). In particular, we will focus on how to determine the right time and context to capture meaningful noise samples, while avoiding excessive resource usage. This architecture is shown in Figure 5.2. In our architecture, we define a module (Context Validation) that is responsible for analyzing the right time to undertake the sensing task. So, once a task notification is received from the server, it will evaluate the user context to determine if the required conditions are met and if so then proceed with the sensing task. In our proposed architecture, this module belongs to a set of modules that is responsible for managing smartphone sensors, and that is part of our general proposal, detailed in chapter 5.

In this section, we define a series of context analysis and actions undertaken at the smartphone at the time of assessing whether the moment and context are adequate for noise-capturing. Also, we define the algorithms used for classification, and then we present the obtained results.

7.1.1 Attribute context classification

Table 7.1: Attribute candidate by categories.

Categories	Attribute
Task	TaskDate, TaskArea
Sound	Speaker, Music, ActiveCall, MicroPhone, ActiveApplication
Status	PhoneStatus, Blocked, Camera, Keyboard, Location

In this section, we seek to assess the adequacy of a particular situation to take environmental noise samples. Specifically, we want to optimize the sequence followed to access the different built-in states and sensors (i.e., GPS, accelerometer) for such an assessment to be made. We define a series of attributes that contribute to determining whether the ideal conditions for environmental noise sampling are met. These attributes are shown in Table 7.1, and were classified according to the conditions and characteristics of the smartphone. For the first category, we consider the tasks as a set of actions to be taken by the smartphone at a specific time and location with the aim of measuring noise pollution (e.g., measure the noise level in the downtown area of Valencia, on Sunday the 11th of October, from 12:00 to 13:00). In particular, it defines the sensing task. Regarding the second category, it refers to the attributes that produce a sound phenomenon, and that can cause the noise readings to be inaccurate. The third category refers to the conditions of the smartphones (e.g. the smartphone can be active or idle). These last two categories refer to the optimal instant to perform the noise measurement. In Table 7.1 we present the different attributes considered. The proposed task-related attributes are the following:

- TaskDate. This attribute allows you to validate the existence of a new sensing task pushed by the server, and to check the range of dates and times associated with a particular task.

- TaskArea. This attribute allows determining whether the smartphone is within any of the target areas considered of interest to the task. In this study, we have defined the target area as a general polygon of n sides, or as a circle.

The proposed sound-related attributes are the following:

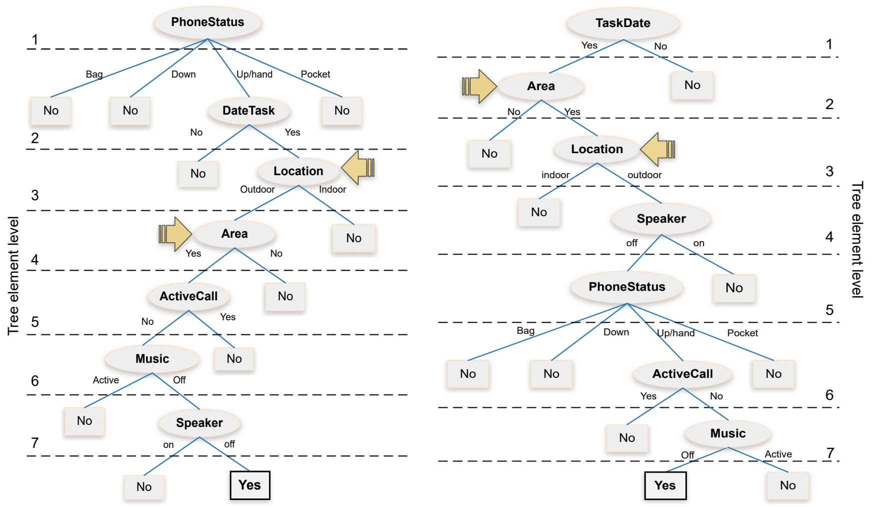
- Speaker. This attribute allows checking whether the smartphone's built-in speakers are on or off. We make use of a call to the system audio administrator to obtain this state information.

- Music. This attribute allows determining when the smartphone is playing music. This information is made available through a call to the operating system. Such playback can be triggered by an event produced by WhatsApp, Spotify, Youtube, or similar applications.
- ActiveCall. This attribute allows determining whether the smartphone has an active call through the telephony management API, which allows determining the specific state of the device.
- PhoneStatus. This attribute refers to the four main states of a smartphone: on the hand, on a flat surface facing upwards/downwards, in a pocket, and in a bag. If the smartphone is being held, or if the smartphone is on a flat surface facing upwards, this is considered a favorable context. We have discarded the options where the smartphone is inside a bag, in a pocket, and over a flat surface facing downwards.
- ActiveApplication. This context allows determining whether the smartphone is making use of some type of social network application or similar (i.e., WhatsApp, Instagram, Facebook, etc.).

The proposed status-related attributes are the following:

- Blocked. This attribute allows determining whether the smartphone's screen is locked or unlocked.
- Microphone. This attribute allows determining whether the microphone integrated into the smartphone is activated or not.
- KeyBoard. This attribute allows determining whether the screen keyboard is activated or not. In general, this can be an indicator that the user is actively using an application, and it can be a good moment to obtain a noise sample.
- Camera. This attribute allows determining whether the smartphone's camera is activated or not.
- Location. This attribute allows determining whether the smartphone is indoors or outdoors, as only outdoor measurements are targeted. This is assessed by accounting for the satellite visibility, which is quite reduced when indoors.

Overall, a smart combination of the attributes mentioned above should allow an adequate assessment of the adequacy of sampling conditions. Also, notice that the first two attributes (TaskDate, and TaskArea) are in fact defined by system managers themselves using the cloud server application. So, each time a user is notified about a new crowdsensing task, the application should collect these



(a) J48 algorithm. (b) Random Tree algorithm.

Figure 7.1: Visualization of the automatically generated trees.

attributes, validating and processing them. Based on these 12 nominal attributes defined above, we have created a list of all possible combinations (16384 cases), being all of them tagged as admissible or not from the perspective of environmental noise assessment.

7.1.2 Weka-based automatic classification

Using as input the 16384 cases referred above, we relied on the Weka tool [121] to provide an automatic classification of these different cases. As output, Weka generated two decision trees, one using the J48 algorithm [95], and another one using the RandomTree algorithm [17].

Figure 7.1 shows the obtained decision trees. Regarding their accuracy, the J48 algorithm achieves a 100% accuracy, while for the RandomTree algorithm, the accuracy achieved is slightly reduced (99.89%), with an absolute error of 0.001. Overall, it is worth mentioning that attributes *ActiveApplication, Block, Microphone, Keyboard*, and *Camera* have been discarded, as they are considered unnecessary by both algorithms.

From a resource consumption perspective, we can observe that the location-related attributes are positioned in the fourth node of both trees, as signalled by the biege arrows. This leads us to think that the resource consumption associated with these decision trees can be excessively high. Additionally, for our study, the *"TaskDate"* attribute should be considered at the beginning of each tree since this is a basic requirement to check for the existence of a new task. In short, these two proposed decision trees offer a viable theoretical solution, but they are not optimal from a software design perspective, and they are not at all optimal regarding time and energy consumption. So, in the next section, we will analyze the different issues involved to optimize the decision process properly; in particular, we will propose an alternative decision tree that is more resource efficient than its automatically generated counterparts (see Figure 7.4).

7.2 Task sequencing optimization

Once the candidate trees were obtained, our next objective was to determine the optimal strategy for collecting noise pollution data through smartphones. To achieve this purpose, in this section we will analyze the computation time associated with each tree element, as well as its level of accuracy. Secondly, we will analyze the energy consumption required. Finally, we will present our proposal for a balanced tree in terms of computation time and energy savings.

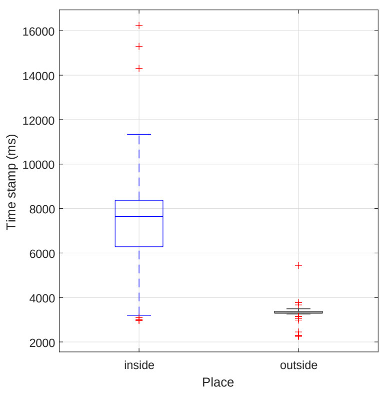
7.2.1 Computation Time

A specific application was developed to analyze the computation time associated with each particular task. This application allows us to evaluate each attribute individually, and follows a repeatable and reliable procedure. In general, 100 independent readings were obtained for each attribute to be measured, and we took their average value. A Samsung S7 Edge model running the Android 7.0.2 operating system was used for testing. Below we detail some relevant characteristics of the most critical attributes:

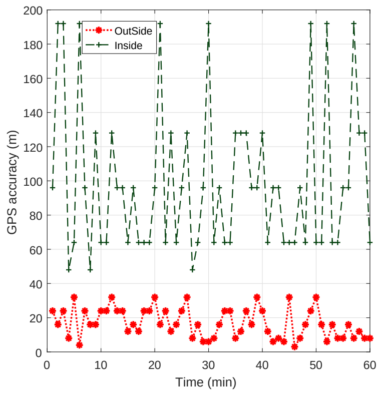
For the *TaskDate, and TaskArea* attributes, the developed application reads the data from an internal database (i.e., SQLite), and then these values were instantiated in a class for later use. We assume that the server application had previously sent these tasks to the smartphone, and so they are available for processing. In particular, the TaskArea attribute was implemented as a class that compares a polygon of n sides with the current position given by the GPS sensor. This class returns true if the smartphone is located inside such a polygon.

Regarding the *Speaker, Music,* and *ActiveCall* attributes, the developed application works by making calls to the corresponding API offered by the operating system. The implementation of the *PhoneStatus* attribute was carried out through a service that reads the proximity, light, and accelerometer sensors. In particular, this service ends when the results are obtained.

The *location* attribute is considered a critical factor because of its high consumption and processing time. To evaluate this attribute, we implemented a service where we read the latitude and longitude of the GPS sensor embedded in the smartphone. Also, we recorded the prediction accuracy to have greater



(a) GPS accuracy analysis: indoor vs. outdoor measurements.



(b) GPS error range.

Figure 7.2: GPS analysis.

control of the positions obtained. A time-stamp was recorded when the GPS obtained the first coordinate. In particular, the design of our solutions aims at outdoor locations alone, meaning that we will also use the GPS to discriminate between both cases (indoor vs. outdoor). To assess the ability of the GPS sensor to differentiate between both environments, we first analyzed the accuracy results achieved inside a building (near to a window to get worst-case conditions), as well as outdoors, in an open environment. For each case, 30 records were taken at two different times: mid-morning, and mid-afternoon. Our goal is checking whether the obtained readings through our application show differentiating features for these environments.

Figure 7.2 (a) shows that, in outdoor environments, 99% of the location measurements were obtained in less than 4000 ms, with just sporadic values found in the four to six second's duration range. Besides, to ensure that the smartphone is indeed in an outdoor environment, Figure 7.2 (b) shows that a GPS accuracy of 40 meters or less is typically only obtainable outdoors, while indoors the accuracy is typically much more reduced, thus allowing to discriminate between both context scenarios clearly. Hence, based on these results, we can validate the attribute location in the scope of our tree, and we will set its duration to 4000 ms, as it provides the necessary trade-off between consumption and accuracy. Finally, Figure 7.3 shows the computation times associated with each key element of the tree (excluding the *Location* parameter). In this figure, it is noticeable that the *PhoneStatus* attribute is the one consuming more resources in our tree, i.e., lasting for 185 ms, followed by the *ActiveCall* attribute, that needs about 9 ms.

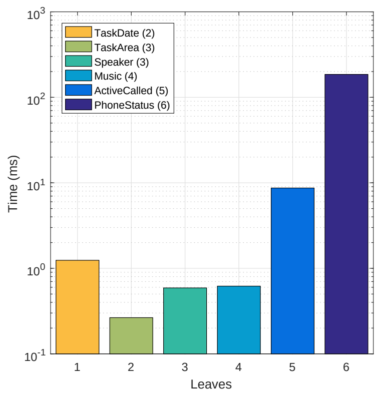


Figure 7.3: Processing time for the different tree elements.

7.2.2 PhoneStatus

Regarding the PhoneStatus attribute, our goal was to determine, with a certain level of accuracy, whether the phone is on the user's hand (phone either in the normal vertical position, or with the left or right turn), or in a desk, but with the front facing upwards. As we said earlier, we have considered these options as the ideal moment to capture environmental noise. The idea of the different states is that there are particular user preferences when used in those positions. The capture of our training dataset was produced as shown in Table 7.2, and our main goal was to determine which was the phone position: on the hand, or at a desk facing upwards. We have developed an application to capture the different proposed states in a supervised way. The application reads the sensors: calibrated gyroscope, proximity sensor, linear acceleration, and light sensor. The capture frequency was of three samples per second for one minute. After completing our learning set, the data were extracted, and we used Matlab as the tool for the treatment and validation of the data.

In particular, we proceeded with the following methodology: (i) we used the K-means algorithm to classify the output from the linear acceleration sensor and the light sensor into three groups. For the linear acceleration, a single value was taken for its three axes. (ii) Once the previous classifications were obtained, a single matrix was made along with the gyroscope values in "x", "y", "z", and the proximity sensor; (iii) Finally, our resulting set was processed using three different classification algorithms and using the k-fold cross-validation on ten observations. Regarding their accuracy, MatLab shows that the Decision Tree achieves a 99.70% accuracy, while for the linear support vector machines algorithm, the accuracy achieved is slightly reduced (86.20%). The same performance occurs when using the algorithm of discrimination with a 67.90% accuracy. Table 7.3 shows the confusion matrix when using the Decision Tree. We can clearly recognize that the different states that we want to validate using our tree are clearly differentiated.

Table 7.2: Details of the training dataset.

Label	Status	Orientation	Movement	Response	Total dataset
Hand(1)	Left, Vertical, Right	North, South, West, East	Static/ Walking	x	4320
Pocket and bag(2)	Up/Down; Vertical/Horizontal	North	Satic / Walking		1440
Desk up (3)	Up	North, South, West, East	Static	x	720
Desk down (4)	Down	North, South, West, East	Static		720

Table 7.3: Confusion matrix associated to phone status recognition.

Label	Recognized value			
	(1)	(2)	(3)	(4)
Hand(1)	4310	9	1	-
Pocket and bag(2)	5	1434	-	1
Desk up (3)	2	-	718	719
Desk down(4)	-	2	-	718

7.2.3 Energy consumption

After determining the computation times associated with each decision attribute, we now proceed to analyze the energy consumption of the different decisions trees. Our methodology relies on Event-based models [52]. Specifically, a background service was implemented on the smartphone that is periodically reading the different attributes of the proposed tree; for all cases, we set the sampling period to 4 seconds. Before each test, we checked that the smartphone's battery is at 100%, and that Internet access is disabled. To obtain representative results, the evaluation lasted for 1 hour. The smartphone used for testing is the Samsung S7 Edge, having a battery capacity of 3600 mAh. In general, three different types of readings were made for comparison, all of them having the smartphone in the suspended mode. The situations under comparison were: i) without the application installed; ii) with the developed application running and testing the different attributes (but without activating the location attribute), and iii) with the application running, but only the attribute that activates the GPS (location) is enabled. The idea of separating the location attribute from the rest one is to have a better insight about the values associated with the different elements. Notice that the high amount of time and resources associated with the location attribute would blur the values related to the other attributes (typically lasting less than 200 ms), thus making such measurements less reliable and representative.

Experimental results show that the one-hour consumption estimation without the application running is of 36 Milliampere-hour (mAh). Then, when turning on our developed application and running it in the background, energy consumption increases to 108 mAh. Finally, if the GPS value is obtained through the location attribute, energy consumption further raises to 180 mAh. Based on the measurements made, it was possible to assess the energy consumed (μAh) by each attribute in our tree. The equation used for this purpose is the following:

$$E_c = \frac{t_l}{\sum t_l} \cdot \frac{E_r - E_o}{N} \tag{7.1}$$

In this equation, E_c represents the energy consumed during 1 second, t_l represents the time overhead associated with each tree attribute, and E_r and E_o represent the reference value for the energy consumed with and without the application, respectively. N is the total number of occurrences recorded in an hour. Table 7.4 presents a summary of the energy consumption estimation associated with each attribute on the tree. In particular, we can observe that the attributes corresponding to the GPS and PhoneStatus present the highest energy consumption values.

Table 7.4: Energy consumption estimation for each leaf of the tree.

Tree element	Random Tree (μAh)	J48 (μAh)
1	0.0354	5.2761
2	0.0076	0.0354
3	160.00	160.00
4	0.0168	0.0076
5	5.2761	0.2479
6	0.2479	0.0176
7	0.0176	0.0168

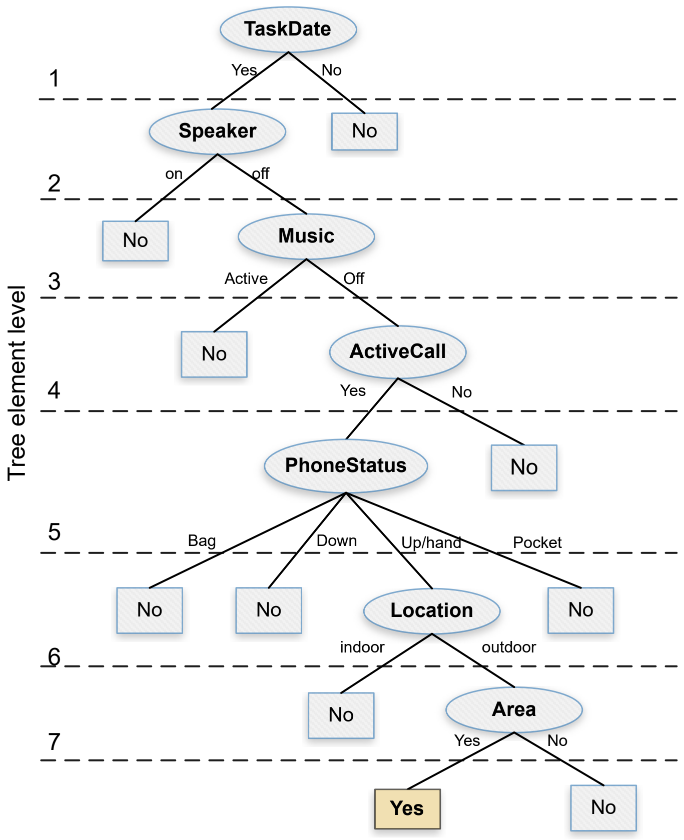
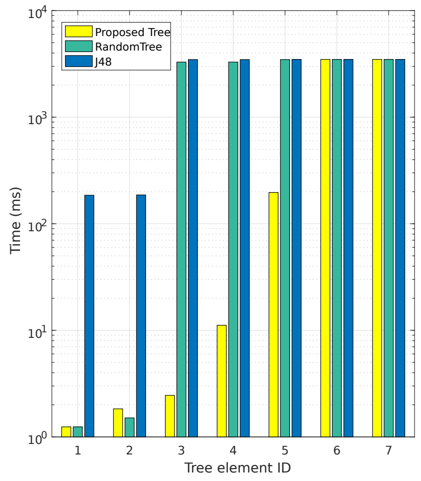


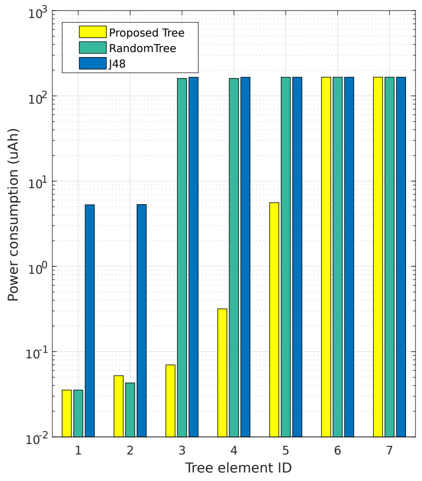
Figure 7.4: Proposed resource-efficient decision tree.

7.2.4 Proposed tree and performance improvement

Once we obtained the computation time overhead and the energy consumption associated with each attribute, our next goal was to propose an alternative decision tree that is more resource efficient. For this purpose, we designed a tree in such a way that its elements are organized and balanced according to the desired objective of reducing time and energy overhead. In particular, for the *TaskDate*, *Speaker*, *Music*, *ActiveCall*, *PhoneStatus*, and *Location* attributes, we followed a sequential order by considering the computation time calculated earlier. Figure 7.4 shows the proposed decision tree which, similarly to the J48 algorithm, can achieve a decision accuracy of 100%. Notice that, in this alternative tree, the location attribute is located near the tree bottom, thereby optimizing the overall system resources whenever a previous attribute allows discarding the noise sam-



(a) Time overhead analysis.



(b) Energy consumption analysis.

Figure 7.5: Time and energy consumption corresponding to the different tree levels.

pling procedure by not meeting the required conditions. Besides, we can observe that the area attribute remains just below the location attribute due to its direct dependency, being this an attribute with a lower computation time, but nevertheless highly relevant in terms of the final decision.

To gain further insight into the performance gains achieved, Figure 7.5 shows a comparison of the accumulative computation time and energy consumption for both our proposed tree and the automatically generated trees. Increasing tree element Ids correspond to progressing along the tree, from top to bottom. In particular, Figure 7.5 (a) shows that our proposal presents a much lower time overhead compared to the two previous candidate trees, being that high periods of activity only take place whenever it becomes indispensable (near the bottom of the tree); specifically, the first tree elements introduce a lower time overhead compared to the others. In Figure 7.5 (b) we find a behavior that is similar to the previous one, although it now represents the overall energy consumption associated with the proposed and automatically generated trees.

Finally, Table 7.5 summarizes the performance benefits introduced by our alternative decision tree. In particular, it shows both the accumulated and average values for the time overhead and energy consumption associated with the three decision trees being compared. We find that our tree is 57.81% and 58.70% lower than the RandomTree algorithm regarding computation time and average energy consumption, respectively. For the J48 tree, improvements are further boosted by 60%, while maintaining the same decision accuracy.

Table 7.5: Estimation of computational requirements and energy consumption for the different decision trees.

Total	Algorithm	CPU time (ms)		Energy(μAh)	
# elements		$\sum$	$\bar{\sum}$	$\sum$	$\bar{\sum}$
7	Proposed tree	3483.89	897.72	165.60	42.16
7	RandomTree	3483.89	2127.84	165.60	102.09
7	J48	3483.89	2244.44	165.60	105.41

Overall, we consider that the process followed in this paper to achieve a tree that is both precise and resource efficient is critical to enable the development of a crowdsensing application aiming at a widespread adoption and usage, a problem to be discussed in the next research steps.

7.3 Summary

In this chapter, we proposed to use smartphones as environmental noise sensors, as part of our complete crowdsensing architecture. In particular, we addressed the issue of optimizing the decision process that precedes actual noise sampling by determining whether the required conditions are met or not. With this purpose, we first defined a set of smartphone contexts, and then, through different algorithms, we automatically generated two decision trees able to meet the decision requirements. However, we found that a theoretical classification does not necessarily provide a balanced tree in terms of computation overhead and energy consumption. Through a detailed analysis of these two performance factors, we determined which attributes had the most negative impact on resources, and then proposed a tree redesign so as to improve resource consumption as much as possible.

Experimental results show that our proposal obtained a relative saving of nearly 60% in terms of both energy consumption and computational overhead, while maintaining the same accuracy as the best decision tree obtained through automated systems (J48).

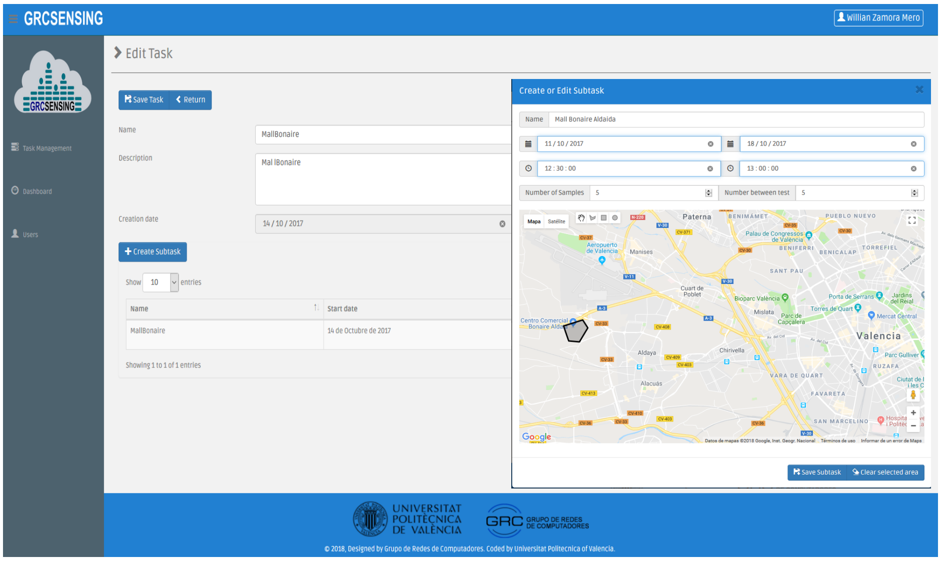
In the next chapter, we integrate the proposed client-side solution in our architecture, and we will proceed to validate our architecture in different scenarios to achieve distributed noise measurements.

Chapter 8

Validation of the Proposed Architecture

The current noise-sensing infrastructure, based on professional sonometers, allows measuring noise pollution levels in cities with high accuracy, although with a very low time and spatial resolution due to the limited number of devices available. In contrast, when adopting a crowdsensing approach, we can achieve much higher time and spatial resolution by relying on the microphones of commercial off-the-shelf smartphones.

The main contribution of this thesis was a complete architecture for environmental noise analysis based on the crowdsensing paradigm. To this goal, in chapter 5 we presented our full architecture, and in chapters 6 and 7 we detailed how to achieve meaningful and accurate noise measurements using commercial smartphones. In this chapter, we proceed to validate the effectiveness of our complete proposal in two representative scenarios: (i) a shopping mall, and (ii) our university campus. The idea of the first scenario is to show that our solution can be used in an environment that is far from the nearest city, and for which there is in general no data regarding noise levels. Concerning the second test scenario, we want to demonstrate that the noise values obtained within our UPV campus, in Valencia, Spain, may differ from those delivered by the Valencia City council, which takes no actual measurements inside the campus.



Edit Task

Name: MallBonaire

Description: Mal lBonaire

Creation date: 14 / 10 / 2017

Show 10 entries

Name	Start date
MallBonaire	14 de Octubre de 2017

Showing 1 to 1 of 1 entries

Create or Edit Subtask

Name: Mall Bonaire Aldaida

11 / 10 / 2017 — 18 / 10 / 2017

12 : 30 : 00 — 13 : 00 : 00

Number of Samples: 5 — Number between test: 5

© 2018, Designed by Grupo de Redes de Computadores. Coded by Universitat Politecnica of Valencia.

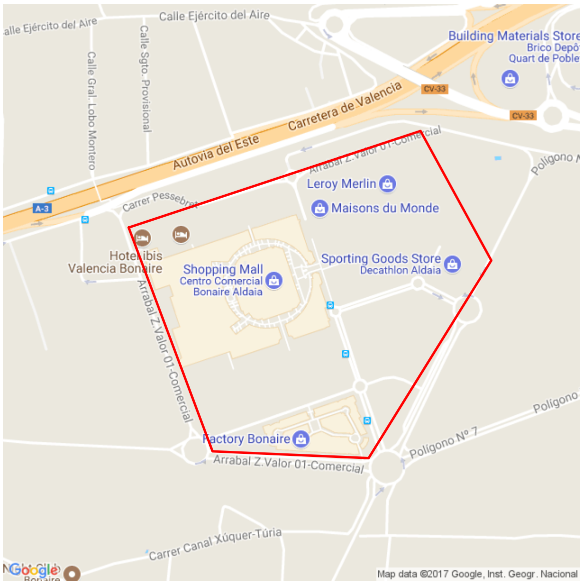
Figure 8.1: Task specification at the back-end server for noise distribution analysis at the Bonaire shopping mall.

8.1 Evaluation of our proposal in an isolated shopping area

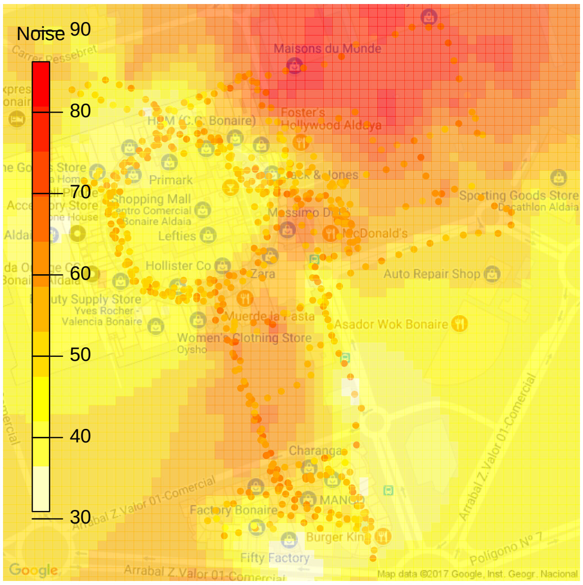
In the first scenario, we evaluated our proposal in a shopping mall characterized by having free pedestrian circulation and open aisles. To perform this test, we first defined the task on the server, and set it so that the noise-capturing takes place during the weekend, specifically on a Saturday from 12:30 to 13:00 PM. The definition of the task is shown in Figure 8.1. Figure 8.2 (a) shows the coverage area for our first test scenario. Concerning users, this first validation was made with four people who took a random and simultaneous tour on the interior facilities (hallway) and outside of the shopping mall. Each of them used different smartphones with the application installed and activated. Figure 8.2 (b) shows the obtained values of processing the results at the server to obtain a heat map. We can see that the places where the highest noise values are detected correspond to vehicular parking areas, while in interiors much lower values are measured. Such results are expectable since combustion-based vehicles are known to introduce more noise compared to people walking around and talking.

8.2 Evaluation of our proposal in an urban area

In the second scenario, we evaluated the noise at the UPV campus. The idea of this evaluation is to compare the results achieved using our proposed architecture

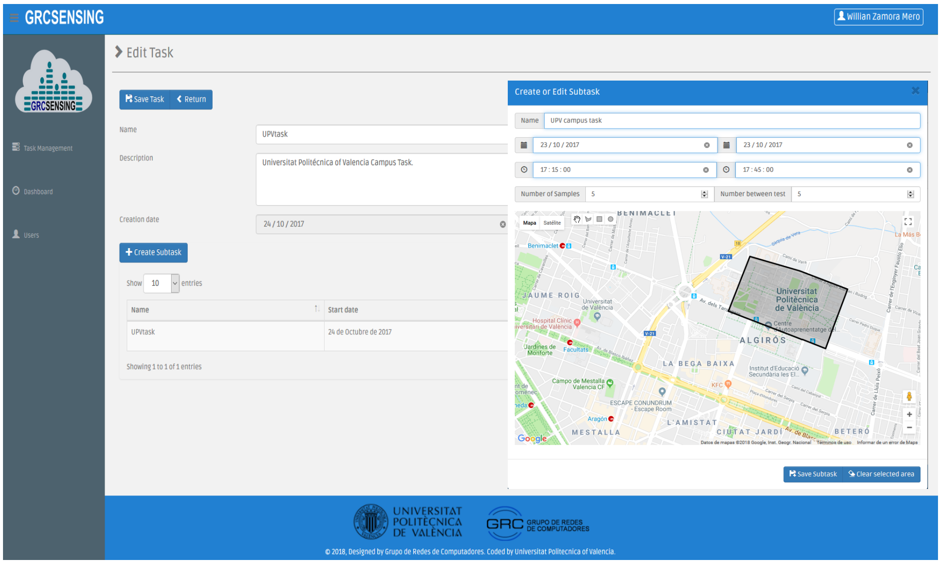


(a)



(b)

Figure 8.2: Analysis of the environmental noise in a shopping mall. (a) Task specification at the back-end server; (b) Noise distribution of the shopping mall (Bonaire).


GRCSENSING
Willian Zamora Mero
Task Management
Dashboard
Users
Save Task
Return
Create Subtask
Save Subtask
Clear selected area
Mapa
Satélite
Universitat Politècnica de València
© 2018, Designed by Grupo de Redes de Computadores. Coded by Universitat Politecnica of Valencia.

Edit Task

Name	UPVtask
Description	Universitat Politécnica of Valencia Campus Task.
Creation date	24 / 10 / 2017

Show 10 entries

Name	Start date
UPVtask	24 de Octubre de 2017

Showing 1 to 1 of 1 entries

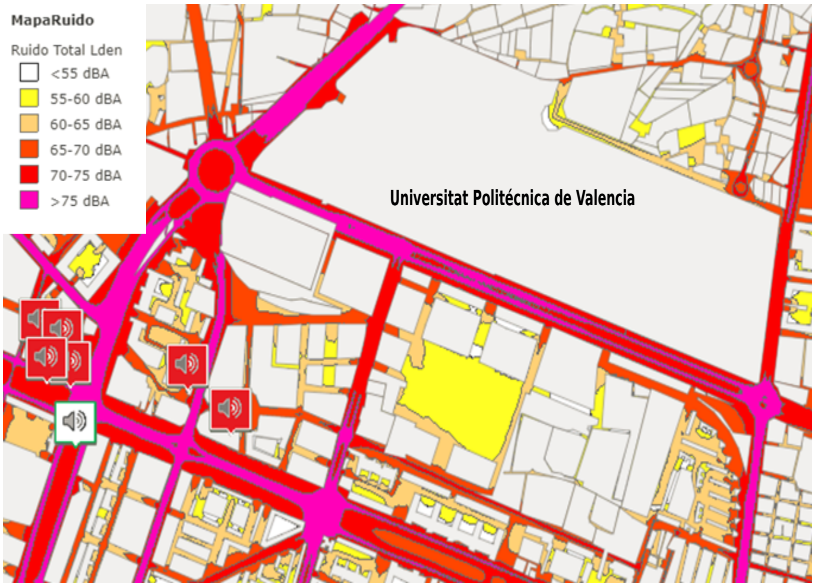
Create or Edit Subtask

Name	UPV campus task		
	23 / 10 / 2017		23 / 10 / 2017
	17 : 15 : 00		17 : 45 : 00
Number of Samples	5	Number between test	5

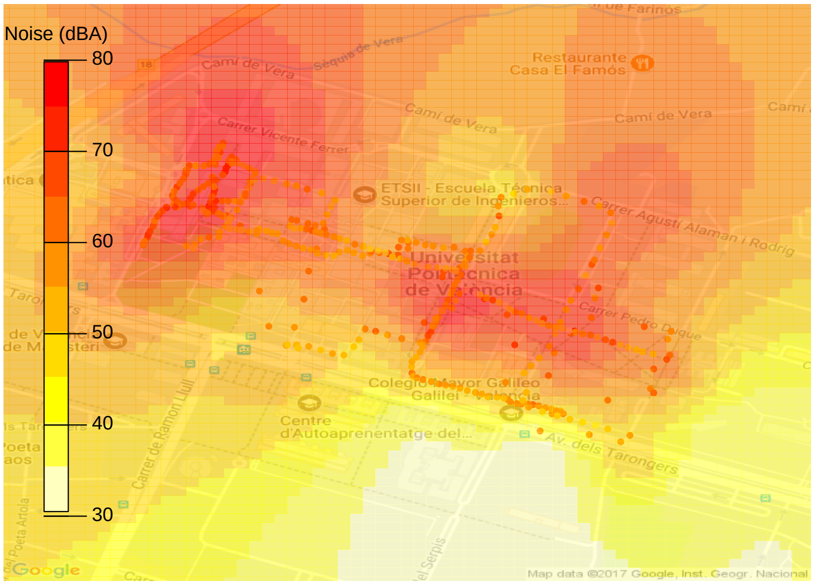
Figure 8.3: Specification of task at the back-end server for analysis noise distribution of the campus UPV.

against the noise analysis provided on the website of the City Hall of Valencia [37]. In Figure 8.4 (a) we show the noise map for the area surrounding the campus, as provided by the City Hall. In general, it is noticeable that high noise levels are located in places of high traffic, like main avenues. We can also see that this Figure represents with white color (<50 dBA) the facilities of our University. Also, the static noise monitoring systems available, which provide constant measurements, are depicted in the figure. As we can see, they are scarce and located at strategic places only.

In view of the above, we proceed to generate a sensing task on the server covering an hour during which many students gather to take a break or coffee (between 17:15 and 17:45). Figure 8.3 shows the task created for this purpose. Specifically, we define a target area sized 800 x 600 square meters. In particular, this test was performed with two people simultaneously walking, one at each end of the map, allowing to cover the interior parts of the campus. The smartphones used were Samsung S7 devices. Figure 8.4(b) shows the results obtained. In general, when compared with Figure 8.4 (a), there is a very noticeable difference concerning the noise values obtained during that hour of the day, especially regarding data within the campus. Also, the heat map is shown in Figure 8.4(b) indicates a high-level noise zone in areas near the coffee shop and restaurants around the campus. Regarding the actual amount of data read on smartphones, the Samsung G930F had a 99.94% effectiveness regarding noise sampling opportunities, while



(a) Analysis of the noise distribution by the city council.


Noise (dBA)
80
70
60
50
40
30
Camí de Vera
Carrer Vicente Ferrer
ETSII - Escuela Técnica Superior de Ingenieros...
Universitat Politècnica de València
Restaurante Casa El Famós
Carrer Agustí Alamán i Rodrigo
Carrer Pedro Duque
Colegio Mayor Galileo Galilei - Valencia
Centre d'Autoaprenentatge del...
Av. dels Tarongers
Carrer de Ramon Llull
Google
Map data ©2017 Google, Inst. Geogr. Nacional

(b) Analysis of the noise distribution by our proposed

Figure 8.4: Comparison of the distribution of environmental noise according to official data, and using our architecture.

the Samsung G935F device had an effectiveness of 94.5%. In the latter, the value varied because in a small part of his test the user walked through areas covered with a ceiling, thereby failing GPS accuracy tests. Overall, the total number of samples was 382.

Finally, these results show that our crowdsensing solution for environmental noise analysis can benefit different to both public or private institutions by providing a real-time data with high spatial and temporal granularity. Based on the information obtained, authorities can detect locations where noise limits are not within the bounds regulated by local laws, and better plan the development in areas of high environmental risk, including the distribution of bus lines, the location of leisure areas, among others. Also, our solution provides an easy interface for the administrator to define tasks, and visualize the outcome of such tasks in the form of heat maps, allowing to see how noise evolves throughout time, and thus assess if any corrective measures taken were effective, and to which extent.

8.3 Summary

Crowdsensing solutions have become an enabling technology for Smart Cities by empowering users to participate in the monitoring process of their environment through their mobile devices. In this scope, studies of noise pollution over densely populated areas are no an exception, with different works pointing in this direction.

In this chapter, we evaluated our complete architecture for environmental noise monitoring that combines smartphones and cloud services to measure noise pollution levels with high spatial granularity. In detail, we used smartphones as mobile sensors to provide noise pollution measurements, and relied on Firebase as a gateway technology, allowing the interaction between the sending of sensing tasks at back-end servers, and the noise capture (by smartphones) at client devices. Once the task is delivered, the smartphone decides the optimum time for capturing data, and it provides real-time feedback on the given noise quality conditions; finally, the back-end server provides services for storage, processing, and data visualization.

Experimental results evidence that, compared to static solutions for environmental noise monitoring, our solution can provide a much higher time and spatial granularity.

Chapter 9

Conclusions, Publications and Future Work

CROWDSENSING is a novel sensing paradigm with unlimited potential waiting to be unleashed. In particular, the combination of multiple smartphones acting as mobile sensing devices allows extensive monitoring of a target area with minimal costs regarding both sensors and infrastructure. Besides, added-value services can be provided to users to promote their participation following data aggregation and data fusion processes in the cloud, where data intelligence can extract relevant information for these users. In this scope, studies of noise and air pollution over densely populated areas are no an exception, with different works pointing in this direction.

Below we briefly summarize the most relevant contributions of this thesis.

- We studied state of the art to determine the most relevant contributions in the crowdsensing area. We addressed this challenge by providing the reader with an extensive review of existing smartphone-based solutions in the field of Mobile Crowdsensing. Our analysis showed that most of the proposals offer some degree of adaptability to different work environments, being that contributions focus on the three main elements: client side (smartphone), server side, and transmission. For each of them, we showed the evolution of the technologies and algorithms used. Also, the main problems described regarding the data capture process, the task generation procedures, and the energy consumption issues.

- Based on an extensive analysis of crowdsensing proposals, we propose a generic crowdsensing architecture based on a client-server approach, and we then classify the different proposals by focusing on the client, the server, and communications components separately. Our proposed classification allows, to a certain degree, match the different proposals analyzed to different categories. It is also evidenced that the adoption of crowdsensing solutions will be generalized in the coming years.

- We proposed a complete solution for environmental monitoring (specifically, ozone levels) that combines low-end sensors, smartphones, and cloud services to measure pollution levels with a high spatial granularity. Our approach was based on using a mobile sensor to provide pollution measurements, a smartphone to provide real-time feedback about air quality conditions, also acting as a gateway to upload gathered data to the cloud server. Additionally, we also provide a cloud server for data processing and visualization. Experimental results show that the sensor orientation and the sampling period, within certain bounds, have very little influence on the data captured. On the contrary, the actual path taken has a significant impact on results, especially when estimating the distribution of pollutants throughout the target area.

- We have studied the feasibility of using commercial smartphones as noise level measurement units by determining their accuracy when compared to a professional noise measurement unit. For this purpose, we focused our study on analyzing the behavior of three different algorithms for noise measurement, determining the best approach when taking as reference a professional and fully-calibrated Class II sound level meter [14].

- We have determined the best sampling strategy and algorithm used to retrieve noise level samples using different types of smartphones. For this purpose, we evaluated, for each algorithm, the impact of different sampling rates and sample block sizes. Then, by selecting our candidate algorithm, we make a performance analysis using different types of smartphones, improving the results based on linear regression techniques, and then evaluating the optimal time-sampling period. Overall, we were able to demonstrate that it is possible to accurately assess noise levels by taking relatively short samples (from 1 to 3 s) while introducing a minimal estimation error. Finally, we validated our proposal in typical outdoor environments.

- We have determined an adequate strategy for capturing environmental noise using smartphones by accounting for their context while minimizing energy consumption. With this purpose, we first defined a set of contexts for the smartphones, and then, through different algorithms, we automatically generated two decision trees able to meet the decision requirements. However, we found that a theoretical classification does not necessarily provide a balanced tree in terms of computation overhead and energy consumption. Through a detailed analysis of these two performance factors, we determined which attributes had the most negative impact on resources and then proposed a tree redesign to improve resource consumption as much as possible. Experimental results showed that our proposal obtained relative savings of nearly 60% in terms of both energy consumption and computing overhead while maintaining the same accuracy as the best decision tree obtained through automated systems (J48).

- We developed an Android application for noise sensing offering reliable and meaningful measurements. Our solution adopts the best sampling strategy, as well as the proposed decision algorithm. Also, our solution uses Firebase technology, which allows real-time communication between servers and smartphones. Besides, we develop a data center for central management of crowdsensing tasks accounting for spatial and temporal restrictions and offering transparent task dissemination to the system users.
- Finally, we validated the proposed noise sensing architecture in real environments. We have shown that, compared to static solutions for environmental noise monitoring, it can provide a much higher time and spatial granularity. In general, we also show that our solution can be adopted in any place, easily reaching those areas where traditional approaches fail or are too expensive.

Overall, we consider that the objectives defined for this thesis have been satisfied by proposing and developing a full crowdsensing solution for the monitoring of noise pollution. Also, its correct functioning was verified by performing different tests taking place in the city of Valencia and nearby areas.

9.1 Publications

This section lists the publications that have been produced as a result of this thesis, as well as some other collaborations and non-directly related publications we published during this time.

9.1.1 International Journals

- O. Alvear, W. Zamora, C. Calafate, J.-C. Cano, and P. Manzoni. “An Architecture Offering Mobile Pollution Sensing with High Spatial Resolution”. In: *Journal of Sensors* 2016.i (2016), pp. 1–13. ISSN: 1687-725X. DOI: 10.1155/2016/1458147. Impact Factor: 1.704 (JCR Q2).

  In this paper, we proposed a mobile sensing architecture able to monitor different pollutants using low-end sensors. Although the proposed solution can be deployed everywhere, it becomes especially meaningful in crowded cities where pollution values are often high, being of great concern to both population and authorities. Our architecture is composed of three different modules: a mobile sensor for monitoring environment pollutants, an Android-based device for transferring the gathered data to a central server, and a central processing server for analyzing the pollution distribution. Moreover, we analyze different issues related to the monitoring process: (i) filtering captured data to reduce the variability of consecutive measurements; (ii) converting the sensor output to actual pollution levels; (iii) reducing temporal variations produced by the mobile sensing process; and (iv) applying interpolation techniques for creating detailed pollution maps. In addition, we study the best strategy to use mobile sensors by first determining the influence of sensor orientation on the captured values, and then analyzing the influence of time and space sampling in the interpolation process.

- W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Survey on Smartphone-Based Crowdsensing Solutions". In: *Mobile Information Systems* 2016 (2016), pp. 1–26. ISSN: 1574-017X. DOI: `10.1155/2016/9681842`. Impact Factor: 1.462 (JCR Q2).

  In this paper, we provided a survey of smartphone-based crowdsensing solutions that have emerged in the past few years, focusing on 64 works published in top-ranked journals and conferences. To properly analyze these previous works, we first define a reference framework based on which we classify the different proposals under study. The results of our survey evidence that there is still much heterogeneity in terms of technologies adopted and deployment approaches, although modular designs at both client and server elements seem to be dominant. Also, the preferred client platform is Android, while server platforms are typically web-based, and client-server communications mostly rely on XML or JSON over HTTP. The main detected pitfall concerns to the performance evaluation of the different proposals, which typically fail to make a scalability analysis despite this being a critical issue when targeting very large communities of users.

- W. Zamora, C. Calafate, J.-C. Cano, and P. Manzoni. "Accurate Ambient Noise Assessment Using Smartphones". In: *Sensors* 17.4 (Apr. 2017), p. 917. ISSN: 1424-8220. DOI: `10.3390/s17040917`. Impact Factor: 2.677 (JCR Q1).

  In this paper, we proposed using smartphones as environmental noise-sensing units. For this purpose, we focus our study on the sound capture and processing procedure, analyzing the impact of different noise calculation algorithms, as well as in determining their accuracy when compared to a professional noise measurement unit. We analyze different candidate algorithms using different types of smartphones, and we study the most adequate time period and sampling strategy to optimize the data-gathering process. In addition, we perform an experimental study comparing our approach with the results obtained using a professional device.

- W. Zamora, E. Vera, C. T. Calafate, J.-C. Cano, and P. Manzoni. “GRC-Sensing: An Architecture to Measure Acoustic Pollution Based on Crowdsensing”. In: *Sensors* 18.8 (2018). ISSN: 1424-8220. DOI: `10.3390/s18082596`. Impact Factor: 2.475 (JCR Q1).

  In this paper, we proposed an alternative approach to this problem based on crowdsensing. Our proposed architecture empowers participating citizens by allowing them to seamlessly and based on their context, sample the noise in their surrounding environment. This allows us to provide a global and detailed view of noise levels around the city, including places traditionally not monitored due to poor accessibility, even while using their vehicles. Also, we detail how the different relevant issues in our architecture, i.e., smartphone calibration, measurement adequacy, server design, and client-server interaction, were solved, and we have validated them in real scenarios to illustrate the potential of the solution achieved.

9.1.2 International Conferences

- O. Alvear, W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “EcoSensor: Monitoring environmental pollution using mobile sensors”. In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, June 2016, pp. 1–6. DOI: `10.1109/wowmom.2016.7523519`. (CORE A)

  In this paper we have developed EcoSensor, a solution to monitor air pollution through mobile sensors. It is deployed with off-the-shelf hardware such as Waspmote (based on the Arduino platform), low-end sensors, and Raspberry Pi devices. EcoSensor collects air pollution using embedded sensors and transfers the captured data to an Android-based device, which displays to the user the air pollution levels in real time. EcoSensor also stores the different pollution traces in a Cloud-based server to analyze the pollution distribution. The cloud server uses the uploaded data, together with highly-accurate data made available by the existing air monitoring infrastructure, to create detailed pollution distribution maps using kriging-based spatial prediction techniques. To optimize the usage of our system, we analyze the impact of sensor orientation in the presence of mobility. Also, we analyze the best time and space sampling strategies to determine the most effective data capturing strategy. Experimental results show that the sensor orientation and the sampling period have a lot less impact on created maps than the actual path taken.

- W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Smartphone tuning for accurate ambient noise assessment". In: *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia - MoMM2017.* ACM Press, Dec. 2017, pp. 115–122. DOI: `10.1145/3151848.3151854`. (CORE B)

  In this paper we show how to tune smartphones so that they become accurate noise-sensing units. In particular, our focus is on the processes of sound capture and sound processing, determining the impact of different noise calculation approaches on the noise estimation accuracy; the latter is determined through comparison against a professional noise sensing device. Through real experiments we find that, with a proper tuning, it is possible to obtain noise measurements in the entire dynamic range of typical smartphones (35 to 95 dB SPL) with an accuracy comparable to that of professional devices.

- W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Noise-Sensing Using Smartphones: Determining the Right Time to Sample". In: *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia - MoMM2017.* ACM Press, Dec. 2017, pp. 196–200. DOI: `10.1145/3151848.3151868`. (CORE B)

  In this paper we determine the precise context for the capture of environmental noise through smartphones. We perform an analysis of the impact that the sensing task collection will have on energy consumption. To this purpose, we defined different contexts to determine whether adequate environment sampling conditions are met, and we then apply classification algorithms to generate the most accurate decisions trees automatically. An analysis of resource consumption requirements associated with the different trees obtained shows that, despite their high accuracy, the resource consumption levels were prohibitive for this kind of applications. Thus, we proposed an alternative decision tree that maintains the accuracy levels of automatically generated trees, while significantly reducing the resource consumption. Experimental results show that our proposed decision tree can reduce the energetic impact of our target application by about 60% when compared to the optimum theoretical tree generated through automatic classification procedures.

9.1.3 National Conferences

- W. Zamora, O. Alvear-Alvear, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Monitorización de la Contaminación Ambiental Mediante Sensores Móviles". In: *Actas Jornadas Sartecto 2016.* Salamanca, Spain: Ediciones Universidad de Salamanca, Sept. 2016, pp. 645–651. ISBN: 978-84-9012-626-4.

- W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Evaluación del Ruido Ambiental Utilizando Teléfonos Inteligentes”. In: *Actas Jornadas Sartecto 2017*. Málaga, Spain: Ediciones Universidad de Málaga, Sept. 2017, pp. 651–657. ISBN: 978-84-697-4835-0.
- W. Zamora, O. E. Vera, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Detección del Ruido Mediante Teléfonos Inteligentes: Determinación del Momento Adecuado para el Muestreo”. In: *Actas Jornadas Sartecto 2018*. Teruel, Spain: Ediciones Sarteco 2018, Sept. 2018, pp. 583–589. ISBN: 978-84-09-04334-7.
- E. Vera, W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Arquitectura Servidora para la Gestión deSoluciones Basadas en Crowdsensing”. In: *Actas Jornadas Sartecto 2018*. Teruel, Spain: Ediciones Sarteco 2018, Sept. 2018, pp. 591–596. ISBN: 978-84-09-04334-7.

9.2 Future work

The results accomplished in this doctoral thesis represent an advance in the state of the art of research in the area of crowdsensing solutions and architectures for environmental monitoring. We consider that the contributions made offer a new starting point that opens a wide range of possibilities regarding research work. In detail, we believe that this thesis can be extended through the following lines of work:

- Integrate algorithm and service proposed in this thesis to platforms of Smartcity is currently available in cities around the world.
- In the mobile application, enhance the proposed algorithms used for the optimization and adaptability of the data collection process.
- Regarding communications, new mechanisms could be exploited to optimize task assignments for different collection scenarios.
- Concerning the server, design new functionality that includes minimizing the number of users required to cover specific areas and as well as exploiting Firebase or other technology for real-time communications.
- Add a new type of task planner that simplifies the generation of tasks and provides warning about tasks about to expire.
- Finally, use our platform for other types of applications, such as measuring or collecting data from other sensors, including temperature and light sensors among others.

Part I

Appendices and References

Appendix A

Acronyms

ITU	International Telecommunication Union	5
WiFi	Wireless Fidelity	15
3G	3 Generation	49
4G	4 Generation Telecommunication	5
GPS	Global Positioning System	5
MCS	Mobile Crowd-Sensing	7
4W1H	What, When, Where, Who, and How	7
MCSC	Mobile Crowd Sensing and Computing	8
SC	Sensor Controller	84

SED Software Event Detector 84

MSC Mobile Sensing Client 9

CIM Client Interface Manager 10

CSM Client Sensor Manager 10

CDM Client Data Manager 11

CCM Client Communications Manager 10

SR Sampling Rate 93

BS Block Size 93

MNSC Mobile Noise-Sensing Client 82

CUI Client User Interface 20

CSTM Client Sensor Task Manager 83

CDCS Cloud Data Colletion Server 9

CDC Cloud Data Colletion 82

SIM Server Interface Manager 10

STM Server Task Manager 10

SDM Server Data Manager 16

SCM	Server Communications Manager	16
MVC	Model View Controller	89
UPV	Universitat Politècnica de València	67
EPA	Environmental Protection Agency	48
IEC	International Electrotechnical Commission	50
SLM	Sound Level Meter	50
AQI	Air Quality Index	48
GUI	Graphical User Interface	11
I/O	Input/Output	14
API	Application Programming Interface	15
IP	Internet Protocols	22
TCP	Transmission Control Protocol	20
P2P	Peer-To-Peer	15
SOAP	Simple Object Access Protocol	15
RESTful	Representational State Transfer	15
JSON	JavaScript Object Notation	15

PHP	Hypertext Pre-Processor	33
XML	Extensible Markup Language	15
HTTP	Hypertext Transfer Protocol	38
AP	Access Point	22
MAC	Media Access Control	22
IoT	Internet of Things	28
WMO	Open Mobile Miner	28
dB	decibel	50
RF	Radio Frequency	50
iOS	iPhone Operating System	54
CPU	Central Processor Unit	58
HTML	Hypertext Markup Language	15
TXT	Plain Text	15
AWS	Amazon Web Services	16
EC2	Amazon Elastic Compute Cloud	16
S3	Amazon Simple Cloud Storage Service	16

GCM Google Cloud Messaging 38

GSN Global Sensor Networks 15

GPRS General Packet Radio Service 49

SQL Structured Query Language 18

JDBC Java Database Connectivity 19

FSI Fluid Structure Interaction 19

OSN Online Social Network 28

QoS Quality of Service 38

mAh Milliampere-hour 124

Tx Transmit 23

NO_x Nitrogen Oxides 48

CO Carbon Monoxide 48

CO_2 Carbon Dioxide 48

O_2 Oxygen 48

O_3 Ozone 48

VOC Volatile Organic Compounds 48

PM Particle Matter

ppb Particle per billion 49

PFISR Poker Flat Incoherent Scatter Radar 33

Bibliography

[1] A. Adam-poupart, A. Brand, M. Fournier, M. Jerrett, and A. Smargiassi. “Spatiotemporal modeling of ozone levels in Quebec (Canada): a comparison of kriging, land-use regression (LUR), and combined Bayesian Maximum Entropy–LUR approaches.” In: *Environmental Health Perspectives* 970.January 2013 (2014), pp. 1–19. ISSN: 1552-9924. DOI: `10.1289/ehp.1306566` (cited on p. 52).

[2] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal. “USense – A Smartphone Middleware for Community Sensing.” In: *2013 IEEE 14th International Conference on Mobile Data Management.* Vol. 1. IEEE, June 2013, pp. 56–65. ISBN: 978-0-7695-4973-6. DOI: `10.1109/MDM.2013.16` (cited on pp. 22, 25, 28, 30, 40, 54, 57).

[3] E. E. Agency. *Noise European Environment Agency.* Available online: `http://www.eea.europa.eu/themes/noise/intro`. (accessed on 10 November 2016). Sept. 2016 (cited on pp. 1, 46).

[4] U. S.E. P. Agency. *Air Quality Index.* Available online: `http://cfpub.epa.gov/airnow/index.cfm?action=aqibasics.aqi`. (accessed on 10 Janaury 2015). 2015 (cited on pp. 48, 62).

[5] A. R. Al-Ali, I. Zualkernan, and F. Aloul. “A Mobile GPRS-Sensors Array for Air Pollution Monitoring.” In: *IEEE Sensors Journal* 10.10 (Oct. 2010), pp. 1666–1671. ISSN: 1530-437X. DOI: `10.1109/JSEN.2010.2045890` (cited on p. 51).

[6] K. Ali, D. Al-Yaseen, A. Ejaz, T. Javed, and H. S. Hassanein. “CrowdITS: Crowdsourcing in Intelligent Transportation Systems.” In: *2012 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Apr. 2012, pp. 3307–3311. ISBN: 978-1-4673-0437-5. DOI: `10.1109/WCNC.2012.6214379` (cited on pp. 8, 24, 31, 36, 38, 42).

[7] O. Alvear, W. Zamora, C. Calafate, J.-C. Cano, and P. Manzoni. “An Architecture Offering Mobile Pollution Sensing with High Spatial Resolution.” In: *Journal of Sensors* 2016.i (2016), pp. 1–13. ISSN: 1687-725X. DOI: `10.1155/2016/1458147` (cited on p. 137).

[8] O. Alvear, W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “EcoSensor: Monitoring environmental pollution using mobile sensors.” In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, June 2016, pp. 1–6. DOI: `10.1109/wowmom.2016.7523519` (cited on p. 139).

[9] H. Aly, A. Basalamah, and M. Youssef. “Map++: A Crowd-Sensing System for Automatic Map Semantics Identification.” In: *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, June 2014, pp. 546–554. ISBN: 978-1-4799-4657-0. DOI: `10.1109/SAHCN.2014.6990394` (cited on pp. 26, 30, 41).

[10] C. M. Angelopoulos, S. Nikoletseas, T. P. Raptis, and J. D. P. Rolim. “Characteristic Utilities, Join Policies and Efficient Incentives in Mobile Crowdsensing Systems.” In: *2014 IFIP Wireless Days (WD)*. IEEE, Nov. 2014, pp. 1–6. ISBN: 978-1-4799-6606-6. DOI: `10.1109/WD.2014.7020795` (cited on pp. 26, 35, 38, 41).

[11] A. Antonic, K. Roankovic, M. Marjanovic, K. Pripuic, and I. P. Arko. “A Mobile Crowdsensing Ecosystem Enabled by a Cloud-Based Publish/Subscribe Middleware.” In: *2014 International Conference on Future Internet of Things and Cloud*. IEEE, Aug. 2014, pp. 107–114. ISBN: 978-1-4799-4357-9. DOI: `10.1109/FiCloud.2014.27` (cited on pp. 19, 25, 29, 30, 40).

[12] Arduino. *Arduino*. `https://www.arduino.cc/`. Jan. 2015 (cited on p. 60).

[13] G. Bajaj and P. Singh. “Sahyog: A Middleware for Mobile Collaborative Applications.” In: *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, July 2015, pp. 1–5. ISBN: 978-1-4799-8784-9. DOI: `10.1109/NTMS.2015.7266518` (cited on pp. 27, 31, 41).

[14] R. E. Basher. *Review of the Dobson spectrophotometer and its accuracy.* Springer, 1985 (cited on p. 48).

[15] M. Becker, S. Caminiti, D. Fiorella, et al. “Awareness and Learning in Participatory Noise Sensing.” In: *PLoS ONE* 8.12 (Dec. 2013). Ed. by N. Lebedev, e81638. ISSN: 1932-6203. DOI: `10.1371/journal.pone.0081638` (cited on p. 57).

[16] Bluetooth. *Bluetooth.* Available online: `http://www.bluetooth.com/`. (accessed on 10 Janaury 2015). 2015 (cited on p. 60).

[17] H. Bouali and J. Akaichi. “Comparative Study of Different Classification Techniques: Heart Disease Use Case.” In: *2014 13th International Conference on Machine Learning and Applications* (Dec. 2014), pp. 482–486. DOI: `10.1109/icmla.2014.84` (cited on p. 119).

[18] M Brković and V Sretović. “Urban Sensing–Smart Solutions for Monitoring Environmental Quality: Case Studies from Serbia. Congress Proceedings.” In: *48th ISOCARP World Congress. : http://bit. ly/Z5AarU [15, 2013].* 2012 (cited on pp. 52, 59).

[19] R. D. Brook. “Inhalation of Fine Particulate Air Pollution and Ozone Causes Acute Arterial Vasoconstriction in Healthy Adults.” In: *Circulation* 105.13 (Mar. 2002), pp. 1534–1536. ISSN: 00097322. DOI: `10.1161/01.cir.0000013838.94747.64` (cited on p. 45).

[20] N. Brouwers and K. Langendoen. “Pogo, a Middleware for Mobile Phone Sensing.” In: *Lecture Notes in Computer Science.* Middleware ’12. Springer Berlin Heidelberg, Dec. 2012, pp. 21–40. ISBN: 978-3-642-35169-3. DOI: `10.1007/978-3-642-35170-9_2` (cited on pp. 24, 33, 34, 41).

[21] M. F. Bulut, M. Demirbas, and H. Ferhatosmanoglu. “LineKing: Coffee Shop Wait-Time Monitoring Using Smartphones.” In: *IEEE Transactions on Mobile Computing* 1233.c (2014), pp. 1–1. ISSN: 1536-1233. DOI: `10.1109/TMC.2014.2384032` (cited on pp. 8, 26, 28, 32, 37, 42).

[22] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. “Participatory sensing.” In: *Center for Embedded Network Sensing* (2006) (cited on p. 6).

[23] F. Calabrese, L. Ferrari, and V. D. Blondel. “Urban Sensing Using Mobile Phone Network Data: A Survey of Research.” In: *ACM Computing Surveys*

47.2 (Nov. 2014), pp. 1–20. issn: 03600300. doi: 10.1145/2655691 (cited on p. 8).

[24] C. T. Calafate and B. Ducourthial. "On the use of mobile sensors for estimating city-wide pollution levels." In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2015 International*. IEEE. 2015, pp. 262–267. doi: 10.1109/iwcmc.2015.7289093 (cited on pp. 52, 59).

[25] A. T. Campbell, S. B. Eisenman, N. D. Lane, et al. "The Rise of People-Centric Sensing." In: *IEEE Internet Computing* 12.4 (July 2008), pp. 12–21. issn: 1089-7801. doi: 10.1109/MIC.2008.90 (cited on p. 6).

[26] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola. "Fostering participaction in smart cities: a geo-social crowdsensing platform." In: *IEEE Communications Magazine* 51.6 (June 2013), pp. 112–119. issn: 01636804. doi: 10.1109/mcom.2013.6525603 (cited on pp. 8, 25, 34, 41).

[27] I. Carreras, D. Miorandi, A. Tamilin, E. R. Ssebaggala, and N. Conci. "Matador: Mobile Task Detector for Context-Aware Crowd-Sensing Campaigns." In: *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. March. IEEE, Mar. 2013, pp. 212–217. isbn: 978-1-4673-5077-8. doi: 10.1109/PerComW.2013.6529484 (cited on pp. 24, 31, 36, 42).

[28] T.-M. Chen, W. G. Kuschner, J. Gokhale, and S. Shofer. "Outdoor air pollution: ozone health effects." In: *The American Journal of the Medical Sciences* 333.4 (Apr. 2007), pp. 244–248. issn: 0002-9629. doi: 10.1097/maj.0b013e31803b8e8c (cited on p. 45).

[29] T.-M. Chen, W. G. Kuschner, J. Gokhale, and S. Shofer. "Outdoor Air Pollution: Particulate Matter Health Effects." In: *The American Journal of the Medical Sciences* 333.4 (Apr. 2007), pp. 235–243. doi: http://dx.doi.org/10.1097/MAJ.0b013e31803b8dcc (cited on p. 45).

[30] Y. Cheng, X. Li, Z. Li, S. Jiang, Y. Li, J. Jia, and X. Jiang. "AirCloud: a cloud-based air-quality monitoring system for everyone." In: *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems - SenSys '14*. ACM. 2014, pp. 251–265. isbn: 9781450331432. doi: 10.1145/2668332.2668346 (cited on pp. 52, 59).

[31] S. Cherrier, I. Salhi, Y. M. Ghamri-Doudane, S. Lohier, and P. Valembois. “BeC 3: Behaviour Crowd Centric Composition for IoT applications.” In: *Mobile Networks and Applications* 19.1 (Feb. 2014), pp. 18–32. ISSN: 1383-469X. DOI: `10.1007/s11036-013-0481-8` (cited on pp. 16, 25, 34, 41).

[32] M. H. Cheung, R. Southwell, F. Hou, and J. Huang. “Distributed Time-Sensitive Task Selection in Mobile Crowdsensing.” In: *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing - MobiHoc ’15*. New York, New York, USA: ACM Press, 2015, pp. 157–166. ISBN: 9781450334891. DOI: `10.1145/2746285.2746293` (cited on pp. 26, 31, 41).

[33] J. Childs-Maidment. *Pyrebase A simple python wrapper for the Firebase API*. `https://pypi.python.org/pypi/Pyrebase`. July 2017 (cited on pp. 88, 89).

[34] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao. “Automatically characterizing places with opportunistic crowdsensing using smartphones.” In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp*. ACM Press, 2012, p. 481. ISBN: 9781450312240. DOI: `10.1145/2370216.2370288` (cited on p. 58).

[35] A. company. *AWS | Elastic compute cloud (EC2)*. Available online: `https://aws.amazon.com/ec2/`. Oct. 2016 (cited on pp. 16, 33).

[36] G. Company. *Google Cloud Platform*. Available online: `https://cloud.google.com/`. Oct. 2016 (cited on pp. 16, 33).

[37] V. C. council. *Heatmaps noise of Valencia, Spain*. `https://aytovalencia.maps.arcgis.com/apps/webappviewer/index.html`. July 2017 (cited on p. 132).

[38] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. “PRISM: Platform for Remote Sensing using Smartphones.” In: *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*. New York, New York, USA: ACM Press, 2010, p. 63. ISBN: 9781605589855. DOI: `10.1145/1814433.1814442` (cited on pp. 8, 22, 24, 30, 39, 40).

[39] H. B. Deng and L. Zhang. “Design on ZigBee Wireless Sensor Network Node.” In: *Key Engineering Materials* 474-476 (Apr. 2011), pp. 283–286.

ISSN: 1662-9795. DOI: `10.4028/www.scientific.net/kem.474-476.283` (cited on p. 52).

[40] E. D'Hondt, M. Stevens, and A. Jacobs. "Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring." In: *Pervasive and Mobile Computing* 9.5 (Oct. 2013), pp. 681–694. ISSN: 15741192. DOI: `10.1016/j.pmcj.2012.09.002` (cited on p. 81).

[41] S. Distefano, F. Longo, and M. Scarpa. "QoS Assessment of Mobile Crowdsensing Services." In: *Journal of Grid Computing* 13.4 (Dec. 2015), pp. 629–650. ISSN: 1570-7873. DOI: `10.1007/s10723-015-9338-7` (cited on pp. 26, 39, 40).

[42] E2V. *O3 Sensor*. Available online: `https://www.cdiweb.com/datasheets/e2v/mics-2610.pdf`. (accessed on 7 July 2016). 2016 (cited on p. 49).

[43] J Fishman, V Ramanathan, P. Crutzen, and S. Liu. "Tropospheric ozone and climate." In: *Nature* 282.5741 (1979), pp. 818–820 (cited on p. 48).

[44] R. Ganti, F. Ye, and H. Lei. "Mobile Crowdsensing: Current State and Future Challenges." In: *IEEE Communications Magazine* 49.11 (Nov. 2011), pp. 32–39. ISSN: 0163-6804. DOI: `10.1109/mcom.2011.6069707` (cited on pp. 7, 8, 81).

[45] C. Greco, M. Kieffer, and C. Adjih. "NeCoRPIA: Network Coding with Random Packet-Index Assignment for mobile crowdsensing." In: *2015 IEEE International Conference on Communications (ICC)*. IEEE, June 2015, pp. 6338–6344. ISBN: 978-1-4673-6432-4. DOI: `10.1109/icc.2015.7249334` (cited on pp. 26, 40).

[46] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang. "FlierMeet: A Mobile Crowdsensing System for Cross-Space Public Information Reposting, Tagging, and Sharing." In: *IEEE Transactions on Mobile Computing* 14.10 (Oct. 2015), pp. 2020–2033. ISSN: 1536-1233. DOI: `10.1109/tmc.2014.2385097` (cited on pp. 27, 31, 41).

[47] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou. "Mobile Crowd Sensing and Computing." In: *ACM Computing Surveys* 48.1 (Aug. 2015), pp. 1–31. ISSN: 03600300. DOI: `10.1145/2794400` (cited on pp. 8, 81).

[48] S. Hachem, V. Mallet, R. Ventura, A. Pathak, V. Issarny, P.-G. Raverdy, and R. Bhatia. “Monitoring Noise Pollution Using the Urban Civics Middleware.” In: *2015 IEEE First International Conference on Big Data Computing Service and Applications*. IEEE, Mar. 2015, pp. 52–61. ISBN: 978-1-4799-8128-1. DOI: `10.1109/BigDataService.2015.16` (cited on p. 57).

[49] K. Han, C. Zhang, and J. Luo. “BLISS: Budget Limited Robust Crowd-Sensing through Online Learning.” In: *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, June 2014, pp. 555–563. ISBN: 978-1-4799-4657-0. DOI: `10.1109/SAHCN.2014.6990395` (cited on pp. 8, 30, 41).

[50] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. “Participatory Air Pollution Monitoring Using Smartphones.” In: *2nd International Workshop on Mobile Sensing* (Apr. 2012), pp. 1–5. DOI: `978-1-4503-1227` (cited on p. 51).

[51] T. Higuchi, H. Yamaguchi, T. Higashino, and M. Takai. “A Neighbor Collaboration Mechanism for Mobile Crowd Sensing in Opportunistic Networks.” In: *2014 IEEE International Conference on Communications (ICC)*. IEEE, June 2014, pp. 42–47. ISBN: 978-1-4799-2003-7. DOI: `10.1109/ICC.2014.6883292` (cited on pp. 26, 39, 40).

[52] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma. “Modeling, Profiling, and Debugging the Energy Consumption of Mobile Devices.” In: *ACM Computing Surveys* 48.3 (Dec. 2015), pp. 1–40. ISSN: 03600300. DOI: `10.1145/2840723` (cited on p. 123).

[53] S. Hu, L. Su, H. Liu, and H. Wang. “SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification.” In: *ACM Transactions on Sensor Networks* 11.4 (July 2015), pp. 1–27. ISSN: 15504859. DOI: `10.1145/2770876` (cited on pp. 27, 29, 32, 37, 42).

[54] X. Hu, T. H. S. Chu, H. C. B. Chan, and V. C. M. Leung. “Vita: A Crowdsensing-Oriented Mobile Cyber-Physical System.” In: *IEEE Transactions on Emerging Topics in Computing* 1.1 (June 2013), pp. 148–165. ISSN: 2168-6750. DOI: `10.1109/tetc.2013.2273359` (cited on pp. 8, 16, 24, 31, 36, 42).

[55] X. Hu, X. Li, E. Ngai, V. Leung, and P. Kruchten. “Multidimensional Context-Aware Social Network Architecture for Mobile Crowdsensing.” In:

IEEE Communications Magazine 52.6 (June 2014), pp. 78–87. ISSN: 01636804. DOI: `10.1109/mcom.2014.6829948` (cited on pp. 16, 21, 26, 32, 37, 42).

[56] C. to connecting the world ITU. *ICT Facts and Figures, The world in 2015.* Jan. 2016 (cited on p. 5).

[57] L. G. Jaimes, I. Vergara-Laurens, and A. Chakeri. "SPREAD, A Crowd Sensing Incentive Mechanism to Acquire Better Representative Samples." In: *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS).* IEEE, Mar. 2014, pp. 92–97. ISBN: 978-1-4799-2736-4. DOI: `10.1109/PerComW.2014.6815171` (cited on pp. 25, 30, 41).

[58] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij. "A Survey of Incentive Techniques for Mobile Crowd Sensing." In: *IEEE Internet of Things Journal* 2.5 (Oct. 2015), pp. 370–380. ISSN: 2327-4662. DOI: `10.1109/jiot.2015.2409151` (cited on pp. 7, 8).

[59] D. J. Jim Morrison Ray Fontaine and D. Yang. *Samsung Galaxy S7 edge SM-G935T Complimentary Teardown Report with Additional Commentary.* Available online: `http://www.chipworks.com/about-chipworks/overview/blog/samsung-galaxy-s7-edge-teardown`. (accessed on 08 October 2016). Mar. 2016 (cited on p. 95).

[60] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu. "Quality of Information Aware Incentive Mechanisms for Mobile Crowd Sensing Systems." In: *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing - MobiHoc '15.* MobiHoc '15. ACM Press, June 2015, pp. 167–176. ISBN: 978-1-4503-3489-1. DOI: `10.1145/2746285.2746310` (cited on p. 7).

[61] H. Jin, L. Su, H. Xiao, and K. Nahrstedt. "Inception: Incentivizing privacy-preserving data aggregation for mobile crowd sensing systems." In: *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing.* ACM. 2016, pp. 341–350 (cited on p. 7).

[62] E. Kanjo. "NoiseSPY: A Real-Time Mobile Phone Platform for Urban Noise Monitoring and Mapping." In: *Mobile Networks and Applications* 15.4 (Aug. 2010), pp. 562–574. ISSN: 1383-469X. DOI: `10.1007/s11036-009-0217-y` (cited on pp. 46, 54, 57, 81, 91).

[63] C. A. Kardous and P. B. Shaw. “Evaluation of smartphone sound measurement applications.” In: *The Journal of the Acoustical Society of America* 135.4 (Apr. 2014), EL186–EL192. DOI: 10.1121/1.4865269 (cited on pp. 54, 91).

[64] C. A. Kardous and P. B. Shaw. “Evaluation of smartphone sound measurement applications (apps) using external microphones A follow-up study.” In: *The Journal of the Acoustical Society of America* 140.4 (Oct. 2016), EL327–EL333. DOI: 10.1121/1.4964639 (cited on p. 55).

[65] M. Katsomallos and S. Lalis. “EasyHarvest: Supporting the Deployment and Management of Sensing Applications on Smartphones.” In: *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*. IEEE, Mar. 2014, pp. 80–85. ISBN: 978-1-4799-2736-4. DOI: 10.1109/percomw.2014.6815169 (cited on pp. 16, 26, 31, 41).

[66] G. Kesavaraj and S. Sukumaran. “A study on classification techniques in data mining.” In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)* (July 2013), pp. 1–7. DOI: 10.1109/icccnt.2013.6726842 (cited on p. 116).

[67] A. Khan, S. K. A. Imon, and S. K. Das. “An Energy Efficient Framework for Localization and Coverage in Participatory Urban Sensing.” In: *39th Annual IEEE Conference on Local Computer Networks*. IEEE, Sept. 2014, pp. 193–201. ISBN: 978-1-4799-3780-6. DOI: 10.1109/LCN.2014.6925772 (cited on pp. 8, 25, 30, 40).

[68] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. “Mobile Phone Sensing Systems: A Survey.” In: *IEEE Communications Surveys & Tutorials* 15.1 (Jan. 2013), pp. 402–427. ISSN: 1553-877X. DOI: 10.1109/surv.2012.031412.00077 (cited on pp. 7, 8).

[69] M. Kisan, S. Sangathan, J. Nehru, and S. G. Pitroda. *Electroacoustics - Sound level meters, Part 1: Specification*. Available online: https://webstore.iec.ch/publication/5708. Jan. 2005 (cited on p. 50).

[70] S. Król, B. Zabiegała, and J. Namieśnik. “Monitoring VOCs in atmospheric air I. On-line gas analyzers.” In: *TrAC Trends in Analytical Chemistry* 29.9 (2010), pp. 1092–1100 (cited on p. 47).

[71] E. Kwan and J. R. Getta. “Design and Implementation of Data Stream Processing Applications.” In: *Dissertation* 4611 (2010), pp. 1–142 (cited on pp. 15, 29).

[72] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. “A survey of mobile phone sensing.” In: *IEEE Communications Magazine* 48.9 (Sept. 2010), pp. 140–150. issn: 0163-6804. doi: 10.1109/mcom.2010.5560598 (cited on pp. 6–8, 81).

[73] O. D. Lara and M. A. Labrador. “A Survey on Human Activity Recognition using Wearable Sensors.” In: *IEEE Communications Surveys & Tutorials* 15.3 (2013), pp. 1192–1209. issn: 1553-877X. doi: 10.1109/surv.2012.110112.00192 (cited on p. 58).

[74] Libelium. *Sensor boards*. Available on: http://www.libelium.com/products/waspmote/sensors/. (accessed on 01 November 2016). (cited on pp. 52, 60).

[75] G. Liu, M. Iwai, Y. Tobe, and K. Sezaki. “REPSense: On-line sensor data reduction while preserving data diversity for mobile sensing.” In: *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, Oct. 2013, pp. 584–591. isbn: 978-1-4799-0428-0. doi: 10.1109/WiMOB.2013.6673417 (cited on pp. 8, 25, 29, 30, 40).

[76] L.-J. Liu and A. Rossini. “Use of kriging models to predict 12-hour mean ozone concentrations in Metropolitan Toronto—A pilot study.” In: *Environment International* 22.6 (Jan. 1996), pp. 677–692. issn: 01604120. doi: 10.1016/s0160-4120(96)00059-1 (cited on p. 52).

[77] P. I. U. Ltd. *Noise meter PCE-322A*. Available online: http://www.industrial-needs.com/technical-data/datalogging-sound-level-meter-sl322.htm. (accessed on 07 June 2016). June 2013 (cited on pp. 92, 95).

[78] H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury. “StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones.” In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp ’12*. New York, New York, USA: ACM Press, 2012, p. 351. isbn: 9781450312240. doi: 10.1145/2370216.2370270 (cited on pp. 8, 24, 29, 30, 40).

[79] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels. “NoiseTube: Measuring and mapping noise pollution with mobile phones.” In: *Environmental Science and Engineering (Subseries: Environmental Science)*. Ed. by P. Golinska, M. Fertsch, and J. Marx-Gómez. Vol. 3. Environmental Science and Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 215–228. ISBN: 978-3-642-19535-8. DOI: `10.1007/978-3-540-88351-7-16` (cited on pp. 54, 57, 81, 91).

[80] S. Manna, S. S. Bhunia, and N. Mukherjee. “Vehicular pollution monitoring using IoT.” In: *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*. May 2014, pp. 1–5. DOI: `10.1109/ICRAIE.2014.6909157` (cited on p. 52).

[81] MariaDB. *MariaDB Foundation.* Available on: `https://mariadb.org/`. (accessed on 01 November 2017). (cited on p. 89).

[82] M. Mead, O. Popoola, G. Stewart, et al. “The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks.” In: *Atmospheric Environment* 70 (May 2013), pp. 186–203. DOI: `https://doi.org/10.1016/j.atmosenv.2012.11.060` (cited on p. 52).

[83] A. Mehrotra, V. Pejovic, and M. Musolesi. “SenSocial: A Middleware for Integrating Online Social Networks and Mobile Sensing Data Streams.” In: *Proceedings of the 15th International Middleware Conference on - Middleware '14*. New York, New York, USA: ACM Press, 2014, pp. 205–216. ISBN: 9781450327855. DOI: `10.1145/2663165.2663331` (cited on pp. 8, 22, 25, 28, 32, 36, 42).

[84] G. Merlino, S. Arkoulis, S. Distefano, C. Papagianni, A. Puliafito, and S. Papavassiliou. “Mobile Crowdsensing as A Service: A Platform for Applications on Top of Sensing Clouds.” In: *Future Generation Computer Systems* 56 (Mar. 2015), pp. 623–639. ISSN: 0167739X. DOI: `10.1016/j.future.2015.09.017` (cited on pp. 16, 21, 23, 26, 33, 35, 41).

[85] J. Mohite, Y. Karale, P. Gupta, S. Kulkarni, B. Jagyasi, and A. Zape. “RuPS: Rural participatory sensing with rewarding mechanisms for crop monitoring.” In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)* (Mar. 2015), pp. 378–383. DOI: `10.1109/PERCOMW.2015.7134067` (cited on pp. 27, 33, 35, 41).

[86] J. Monge-Alvarez, C. Hoyos-Barcelo, P. Lesso, J. Escudero, K. Dahal, and P. C. de-la Higuera. “Effect of Importance Sampling on Robust Segmentation of Audio - cough Events in Noisy Environments.” In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 38. Orlando, FL, USA: In Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Aug. 2016, pp. 3740–3744. isbn: 9781457702204. doi: `10.1109/embc.2016.7591541` (cited on p. 54).

[87] L. A. Muratori, P. Salomoni, and G. Pau. “Feeling the pack: Strategies for an optimal participatory system to sense and recognize noise pollution.” In: *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*. IEEE, Sept. 2011, pp. 17–21. isbn: 978-1-4577-0233-4. doi: `10.1109/icce-berlin.2011.6031816` (cited on p. 57).

[88] D. Nast, W. Speer, and C. Le Prell. “Sound level measurements using smartphone "apps": Useful or inaccurate?” In: *Noise and Health* 16.4 (Apr. 2014), pp. 251–256. doi: `10.4103/1463-1741.140495` (cited on pp. 54, 55, 91).

[89] P. Nguyen and K. Nahrstedt. “Context-Aware Crowd-Sensing in Opportunistic Mobile Social Networks.” In: *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, Oct. 2015, pp. 477–478. isbn: 978-1-4673-9101-6. doi: `10.1109/MASS.2015.80` (cited on pp. 27, 40).

[90] NIOSH. *CDC-NIOSH Publications and Products - Occupational Noise Exposure (98-126)*. Available online: `https://www.cdc.gov/niosh/docs/98-126`. (accessed on 30 June 2016). June 2013 (cited on p. 46).

[91] T. O. Oshin and S. Poslad. “Improving the Energy-Efficiency of GPS Based Location Sensing Smartphone Applications.” In: *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, June 2012, pp. 1698–1705. isbn: 9780769547459. doi: `10.1109/trustcom.2012.184` (cited on p. 22).

[92] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter. “Mobile crowd sensing in space weather monitoring: the mahali project.” In: *IEEE Communications Magazine* 52.8 (Aug. 2014), pp. 22–28. issn: 0163-6804. doi: `10.1109/mcom.2014.6871665` (cited on pp. 8, 25, 32, 33, 37, 42).

[93] E. Parliament. “Legislation - Directive 2002/49/EC of the European Parliament and of the Council of 25 June 2002 relating to the assessment and

management of environmental noise." In: *Official Journal of the European Communities* 45.L189 (July 2002), pp. 12–25. ISSN: 0378-6978 (cited on pp. 1, 46).

[94] E. Parliament and C. of the European Union. "Directives - Commission Directive (EU) 2015/996 of 19 May 2015 establishing common noise assessment methods according to Directive 2002/49/EC of the European Parliament and of the Council." In: *Official Journal of the European Union* 58.L168 (July 2015), p. 1. ISSN: 1977-0677 (cited on pp. 1, 46).

[95] T. R. Patil and Sherekar. "Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification." In: *International Journal Of Computer Science And Applications* 6.2 (2013). ISSN: 0974-1011 (cited on p. 119).

[96] C. Perera, A. Zaslavsky, P. Christen, A. Salehi, and D. Georgakopoulos. "Capturing sensor data from mobile phones using Global Sensor Network middleware." In: *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*. IEEE, Sept. 2012, pp. 24–29. ISBN: 978-1-4673-2569-1. DOI: `10.1109/pimrc.2012.6362778` (cited on pp. 24, 28–30, 40).

[97] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen. "MOSDEN: An Internet of Things Middleware for Resource Constrained Mobile Devices." In: *2014 47th Hawaii International Conference on System Sciences*. IEEE, Jan. 2014, pp. 1053–1062. ISBN: 978-1-4799-2504-9. DOI: `10.1109/HICSS.2014.137`. arXiv: `arXiv:1310.4038v1` (cited on pp. 8, 23, 25, 28–30, 40).

[98] G. C. Platform. *Firebase*. Available online: `https://firebase.google.com/products/`. 2017 (cited on p. 86).

[99] D. projects. *Django the web framework for perfectionists with deadlines*. Available on: `https://www.djangoproject.com/`. (accessed on 01 November 2017). (cited on p. 89).

[100] R. Pryss, M. Reichert, B. Langguth, and W. Schlee. "Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy, and Research." In: *2015 IEEE International Conference on Mobile Services*. Vol. 32. 2. IEEE, June 2015, pp. 352–359. ISBN: 978-1-4673-7284-8. DOI: `10.1109/mobserv.2015.55` (cited on pp. 8, 26, 28, 32, 37, 42, 54).

[101] Z. Qin and Y. Zhu. "NoiseSense: A Crowd Sensing System for Urban Noise Mapping Service." In: *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, Dec. 2016, pp. 80–87. ISBN: 978-1-5090-4457-3. DOI: 10.1109/icpads.2016.0020 (cited on p. 54).

[102] R-Foundation. *R Project*. Available on: https://www.r-project.org. (accessed on 01 November 2015). (cited on pp. 60, 89).

[103] M.-r. Ra, B. Liu, T. F. La Porta, and R. Govindan. "Medusa: A Programming Framework for Crowd-Sensing Applications." In: *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*. Section 2. New York, New York, USA: ACM Press, 2012, p. 337. ISBN: 9781450313018. DOI: 10.1145/2307636.2307668 (cited on pp. 8, 16, 24, 31, 33, 36, 39, 42).

[104] R. Rana, C. T. Chou, N. Bulusu, S. Kanhere, and W. Hu. "Ear-Phone: A context-aware noise mapping using smart phones." In: *Pervasive and Mobile Computing* 17.PA (Feb. 2015), pp. 1–22. ISSN: 15741192. DOI: 10.1016/j.pmcj.2014.02.001. arXiv: 1310.4270 (cited on pp. 54, 57, 91).

[105] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. "Ear-Phone: An End-to-End Participatory Urban Noise Mapping System." In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN '10*. June. New York, New York, USA: ACM Press, 2010, p. 105. ISBN: 9781605589886. DOI: 10.1145/1791212.1791226 (cited on pp. 8, 24, 31, 36, 42, 54, 57, 81).

[106] Raspberry-Pi. *Raspberry-Pi*. Available online: https://www.raspberrypi.org/. 2015 (cited on p. 60).

[107] Y. Ren, C. Wang, J. Yang, and Y. Chen. "Fine-grained sleep monitoring: Hearing your breathing with smartphones." In: *Proceedings - IEEE INFOCOM* 26 (2015), pp. 1194–1202. ISSN: 0743166X. DOI: 10.1109/INFOCOM.2015.7218494 (cited on p. 54).

[108] L. Ruge, B. Altakrouri, and A. Schrader. "SoundOfTheCity - Continuous noise monitoring for a healthy city." In: *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. March. IEEE, Mar. 2013, pp. 670–675. ISBN: 978-1-4673-5077-8. DOI: 10.1109/percomw.2013.6529577 (cited on pp. 8, 24, 31, 36, 42, 54, 57, 81).

[109] S. Sarma, N. Venkatasubramanian, and N. Dutt. “Sense-making from Distributed and Mobile Sensing Data.” In: *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference - DAC ’14*. New York, New York, USA: ACM Press, 2014, pp. 1–6. ISBN: 9781450327305. DOI: `10.1145/2593069.2596688` (cited on pp. 23, 25, 29, 30, 38, 40).

[110] S. Sathe, T. Sellis, and K. Aberer. “On Crowdsensed Data Acquisition Using Multi-Dimensional Point Processes.” In: *2015 31st IEEE International Conference on Data Engineering Workshops*. IEEE, Apr. 2015, pp. 124–128. ISBN: 978-1-4799-8442-8. DOI: `10.1109/ICDEW.2015.7129562` (cited on pp. 27, 31, 41).

[111] I. Schweizer, T. Darmstadt, F. Probst, et al. “NoiseMap - Real-time participatory noise maps.” In: *World* (2011), to appear (cited on p. 57).

[112] X. Sheng, J. Tang, X. Xiao, and G. Xue. “Sensing as a Service: Challenges, Solutions and Future Directions.” In: *IEEE Sensors Journal* 13.10 (Oct. 2013), pp. 3733–3741. ISSN: 1530-437X. DOI: `10.1109/JSEN.2013.2262677` (cited on pp. 16, 25, 34, 41).

[113] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos. “AnonySense: A System for Anonymous Opportunistic Sensing.” In: *Pervasive and Mobile Computing* 7.1 (Feb. 2011), pp. 16–30. ISSN: 15741192. DOI: `10.1016/j.pmcj.2010.04.001` (cited on pp. 8, 22, 24, 33, 34, 41).

[114] M. Shoaib, S. Bosch, O. Incel, H. Scholten, and P. Havinga. “A Survey of Online Activity Recognition Using Mobile Phones.” In: *Sensors* 15.1 (Jan. 2015), pp. 2059–2085. ISSN: 1424-8220. DOI: `10.3390/s150102059` (cited on p. 58).

[115] C. Song, M. Liu, and X. Dai. “Remote Cloud or Local Crowd: Communicating and Sharing the Crowdsensing Data.” In: *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*. IEEE, Aug. 2015, pp. 293–297. ISBN: 978-1-4673-7183-4. DOI: `10.1109/bdcloud.2015.68` (cited on pp. 27, 40).

[116] M Stevens. “Community memories for sustainable societies: The case of environmental noise [Doctoral Dissertation].” PhD thesis. Vrije Universiteit Brussel, Bruxelles, Belgium, June 2012 (cited on p. 93).

[117] R. Szabo, K. Farkas, M. Ispany, et al. “Framework for Smart City Applications Based on Participatory Sensing.” In: *2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*. IEEE, Dec. 2013, pp. 295–300. ISBN: 978-1-4799-1546-0. DOI: `10.1109/CogInfoCom.2013.6719260` (cited on pp. 8, 21, 24, 33, 34, 41).

[118] M. Talasila, R. Curtmola, and C. Borcea. “Improving location reliability in crowd sensed data with minimal efforts.” In: *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, Apr. 2013, pp. 1–8. ISBN: 978-1-4673-5616-9. DOI: `10.1109/wmnc.2013.6549016` (cited on pp. 24, 25, 34, 41).

[119] M. Talasila, R. Curtmola, and C. Borcea. “Alien vs. Mobile User Game: Fast and Efficient Area Coverage in Crowdsensing.” In: *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*. ICST, 2014, pp. 65–74. ISBN: 978-1-63190-024-2. DOI: `10.4108/icst.mobicase.2014.257779` (cited on pp. 11, 30, 40).

[120] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. “VTrack : Accurate , Energy-aware Road Traffic Delay Estimation Using Mobile Phones.” In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems - SenSys 09*. ACM Press, Nov. 2009, pp. 85–98. ISBN: 9781605587486. DOI: `10.1145/1644038.1644048` (cited on p. 22).

[121] University of Waikato. *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. 2017 (cited on p. 119).

[122] E. Vera, W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Arquitectura Servidora para la Gestión deSoluciones Basadas en Crowdsensing.” In: *Actas Jornadas Sartecto 2018*. Teruel, Spain: Ediciones Sarteco 2018, Sept. 2018, pp. 591–596. ISBN: 978-84-09-04334-7 (cited on p. 141).

[123] J. Wang, J. Tang, D. Yang, E. Wang, and G. Xue. “Quality-Aware and Fine-Grained Incentive Mechanisms for Mobile Crowdsensing.” In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, June 2016, pp. 354–363. DOI: `10.1109/icdcs.2016.30` (cited on p. 7).

[124] L. Wang, D. Zhang, A. Pathak, C. Chen, H. Xiong, D. Yang, and Y. Wang. “CCS-TA: Quality-guaranteed Online Task Allocation in Compressive Crowdsensing.” In: *Proceedings of the 2015 ACM International Joint*

Conference on Pervasive and Ubiquitous Computing. UbiComp '15. ACM Press, Sept. 2015, pp. 683–694. ISBN: 978-1-4503-3574-4. DOI: `10.1145/2750858.2807513` (cited on p. 33).

[125] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed. "Sparse mobile crowdsensing: challenges and opportunities." In: *IEEE Communications Magazine* 54.7 (July 2016), pp. 161–167. DOI: `10.1109/MCOM.2016.7509395` (cited on pp. 27, 33, 36, 42).

[126] L. Wang, D. Zhang, Z. Yan, H. Xiong, and B. Xie. "effSense: A Novel Mobile Crowd-Sensing Framework for Energy-Efficient and Cost-Effective Data Uploading." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.12 (Dec. 2015), pp. 1549–1563. ISSN: 2168-2216. DOI: `10.1109/TSMC.2015.2418283` (cited on pp. 27, 40).

[127] Z. Wang, D. Huang, H. Wu, Y. Deng, A. Aikebaier, and Y. Teranishi. "QoS-constrained sensing task assignment for mobile crowd sensing." In: *2014 IEEE Global Communications Conference*. IEEE, Dec. 2014, pp. 311–316. ISBN: 978-1-4799-3512-3. DOI: `10.1109/GLOCOM.2014.7036826` (cited on pp. 26, 35, 42).

[128] M. Wisniewski, G. Demartini, and A. Malatras. "NoizCrowd: A Crowd-Based Data Gathering and Management System for Noise Level Data." In: *Mobile Web Information Systems: 10th International Conference, MobiWIS 2013, Paphos, Cyprus, August 26-29, 2013. Proceedings*. Vol. 8093 LNCS. New York, NY, USA: Springer Berlin Heidelberg, 2013, pp. 172–186. ISBN: 9783642402753. DOI: `10.1007/978-3-642-40276-0-14` (cited on pp. 8, 21, 25, 35, 41, 54, 57, 81).

[129] F.-J. Wu and T. Luo. "WiFiScout: A Crowdsensing WiFi Advisory System with Gamification-Based Incentive." In: *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*. 1. IEEE, Oct. 2014, pp. 533–534. ISBN: 978-1-4799-6036-1. DOI: `10.1109/MASS.2014.32` (cited on pp. 26, 31, 41).

[130] X. Wu, P. Yang, S. Tang, X. Zheng, and Y. Xiong. "Privacy Preserving RSS Map Generation for A Crowdsensing Network." In: *IEEE Wireless Communications* 22.4 (Aug. 2015), pp. 42–48. ISSN: 1536-1284. DOI: `10.1109/MWC.2015.7224726` (cited on pp. 8, 27, 32, 37, 42).

[131] Xiang Sheng, Jian Tang, Xuejie Xiao, and Guoliang Xue. "Leveraging GPS-Less Sensing Scheduling for Green Mobile Crowd Sensing." In: *IEEE Inter-*

net of Things Journal 1.4 (Aug. 2014), pp. 328–336. issn: 2327-4662. doi: 10.1109/JIOT.2014.2334271 (cited on pp. 25, 34, 41).

[132] L. Xiao, J. Liu, Q. Li, and H. V. Poor. “Secure Mobile Crowdsensing Game.” In: *2015 IEEE International Conference on Communications (ICC)*. Vol. 22. 1. IEEE, June 2015, pp. 7157–7162. isbn: 978-1-4673-6432-4. doi: 10.1109/ICC.2015.7249468 (cited on pp. 27, 35, 41).

[133] L. Xu, X. Hao, N. D. Lane, X. Liu, and T. Moscibroda. “More with Less: Lowering User Burden in Mobile Crowdsourcing Through Compressive Sensing.” In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp ’15. ACM Press, Sept. 2015, pp. 659–670. isbn: 978-1-4503-3574-4. doi: 10.1145/2750858.2807523 (cited on pp. 27, 33, 36, 42).

[134] Y. Yao, L. T. Yang, and N. N. Xiong. “Anonymity-Based Privacy-Preserving Data Reporting for Participatory Sensing.” In: *IEEE Internet of Things Journal* 2.5 (Oct. 2015), pp. 381–390. issn: 2327-4662. doi: 10.1109/JIOT.2015.2410425 (cited on pp. 22, 26, 40).

[135] F. Ye, R. Ganti, R. Dimaghani, K. Grueneberg, and S. Calo. “MECA: Mobile Edge Capture and Analysis Middleware for Social Sensing Applications.” In: *Proceedings of the 21st international conference companion on World Wide Web - WWW ’12 Companion*. New York, New York, USA: ACM Press, 2012, p. 699. isbn: 9781450312301. doi: 10.1145/2187980.2188184 (cited on pp. 8, 24, 34, 41).

[136] W. Zamora, O. Alvear-Alvear, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Monitorización de la Contaminación Ambiental Mediante Sensores Móviles.” In: *Actas Jornadas Sartecto 2016*. Salamanca, Spain: Ediciones Universidad de Salamanca, Sept. 2016, pp. 645–651. isbn: 978-84-9012-626-4 (cited on p. 140).

[137] W. Zamora, C. Calafate, J.-C. Cano, and P. Manzoni. “Accurate Ambient Noise Assessment Using Smartphones.” In: *Sensors* 17.4 (Apr. 2017), p. 917. issn: 1424-8220. doi: 10.3390/s17040917 (cited on p. 138).

[138] W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “A Survey on Smartphone-Based Crowdsensing Solutions.” In: *Mobile Information Systems* 2016 (2016), pp. 1–26. issn: 1574-017X. doi: 10.1155/2016/9681842 (cited on p. 138).

[139] W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Evaluación del Ruido Ambiental Utilizando Teléfonos Inteligentes.” In: *Actas Jornadas Sartecto 2017*. Málaga, Spain: Ediciones Universidad de Málaga, Sept. 2017, pp. 651–657. ISBN: 978-84-697-4835-0 (cited on p. 141).

[140] W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Noise-Sensing Using Smartphones: Determining the Right Time to Sample.” In: *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia - MoMM2017*. ACM Press, Dec. 2017, pp. 196–200. DOI: 10.1145/3151848.3151868 (cited on p. 140).

[141] W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Smartphone tuning for accurate ambient noise assessment.” In: *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia - MoMM2017*. ACM Press, Dec. 2017, pp. 115–122. DOI: 10.1145/3151848.3151854 (cited on p. 140).

[142] W. Zamora, E. Vera, C. T. Calafate, J.-C. Cano, and P. Manzoni. “GRC-Sensing: An Architecture to Measure Acoustic Pollution Based on Crowdsensing.” In: *Sensors* 18.8 (2018). ISSN: 1424-8220. DOI: 10.3390/s18082596 (cited on p. 139).

[143] W. Zamora, O. E. Vera, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Detección del Ruido Mediante Teléfonos Inteligentes: Determinación del Momento Adecuado para el Muestreo.” In: *Actas Jornadas Sartecto 2018*. Teruel, Spain: Ediciones Sarteco 2018, Sept. 2018, pp. 583–589. ISBN: 978-84-09-04334-7 (cited on p. 141).

[144] P. H. T. Zannin, A. M. C. Ferreira, and B. Szeremetta. “Evaluation of Noise Pollution in Urban Parks.” In: *Environmental Monitoring and Assessment* 118.1 (2006), pp. 423–433. ISSN: 1573-2959. DOI: 10.1007/s10661-006-1506-6 (cited on p. 46).

[145] I. P. Zarko, A. Antonic, and K. Pripužic. “Publish/subscribe middleware for energy-efficient mobile crowdsensing.” In: *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp 13*. New York, New York, USA: ACM Press, 2013, pp. 1099–1110. ISBN: 9781450322157. DOI: 10.1145/2494091.2499577 (cited on pp. 8, 24, 31, 33, 36, 38, 42).

[146] A. Zaslavsky, P. P. Jayaraman, and S. Krishnaswamy. “ShareLikesCrowd: Mobile Analytics for Participatory Sensing and Crowd-Sourcing Applica-

tions." In: *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, Apr. 2013, pp. 128–135. ISBN: 978-1-4673-5304-5. DOI: `10.1109/ICDEW.2013.6547440` (cited on pp. 8, 19, 24, 28, 29, 31, 36, 42).

[147] C. Zhang, K. P. Subbu, J. Luo, and J. Wu. "GROPING: Geomagnetism and Crowdsensing Powered Indoor NaviGation." In: *IEEE Transactions on Mobile Computing* 14.2 (Feb. 2015), pp. 387–400. ISSN: 1536-1233. DOI: `10.1109/TMC.2014.2319824` (cited on pp. 26, 35, 41).

[148] D. Zhang, L. Wang, H. Xiong, and B. Guo. "4W1H in mobile crowd sensing." In: *IEEE Communications Magazine* 52.8 (Aug. 2014), pp. 42–48. ISSN: 0163-6804. DOI: `10.1109/MCOM.2014.6871668` (cited on pp. 7, 8).

[149] D. Zhang, H. Xiong, L. Wang, and G. Chen. "CrowdRecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint." In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*. New York, New York, USA: ACM Press, 2014, pp. 703–714. ISBN: 9781450329682. DOI: `10.1145/2632048.2632059` (cited on pp. 26, 35, 42).

[150] L. Zhang, J. Liu, H. Jiang, and Y. Guan. "SensTrack: Energy-Efficient Location Tracking With Smartphone Sensors." In: *IEEE Sensors Journal* 13.10 (Oct. 2013), pp. 3775–3784. ISSN: 1530437X. DOI: `10.1109/jsen.2013.2274074` (cited on p. 22).

[151] S. Zhang, Q. Ma, T. Zhu, K. Liu, L. Zhang, W. He, and Y. Liu. "PLP: Protecting Location Privacy Against Correlation-Analysis Attack in Crowdsensing." In: *44th International Conference on Parallel Processing* (Sept. 2015), pp. 111–119. DOI: `10.1109/icpp.2015.20` (cited on pp. 27, 35, 42).

[152] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao. "Incentives for Mobile Crowd Sensing: A Survey." In: *IEEE Communications Surveys & Tutorials* c (2015), pp. 1–1. ISSN: 1553-877X. DOI: `10.1109/COMST.2015.2415528` (cited on pp. 7, 8).

[153] D. Zhao, H. Ma, S. Tang, and X.-Y. Li. "COUPON: A Cooperative Framework for Building Sensing Maps in Mobile Opportunistic Networks." In: *IEEE Transactions on Parallel and Distributed Systems* 26.2 (Feb. 2015), pp. 392–402. ISSN: 1045-9219. DOI: `10.1109/TPDS.2014.2308178` (cited on pp. 26, 40).

[154] Y. Zheng, F. Liu, and H.-p. Hsieh. “U-Air: when urban air quality inference meets big data.” In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’13*. ACM. 2013, pp. 1436–1444. ISBN: 9781450321747. DOI: `10.1145/2487575.2488188` (cited on pp. 52, 59).

[155] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang. “Diagnosing New York city's noises with ubiquitous data.” In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*. New York, New York, USA: ACM Press, Sept. 2014, pp. 715–725. ISBN: 9781450329682. DOI: `10.1145/2632048.2632102` (cited on pp. 8, 24, 34, 41).

[156] C. Zhou, C.-k. Tham, and M. Motani. “QOATA: QoI-Aware Task Allocation Scheme for Mobile Crowdsensing Under Limited Budget.” In: *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. April. IEEE, Apr. 2015, pp. 1–6. ISBN: 978-1-4799-8055-0. DOI: `10.1109/ISSNIP.2015.7106953` (cited on pp. 26, 35, 41).

[157] Y. Zhu, J. Li, L. Liu, and C.-K. Tham. “iCal: Intervention-free Calibration for Measuring Noise with Smartphones.” In: *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. Vol. 2016-Janua. IEEE, Dec. 2015, pp. 85–91. ISBN: 978-0-7695-5785-4. DOI: `10.1109/icpads.2015.19` (cited on p. 55).

[158] Zigbee. *Zigbee*. Available online: `http://www.zigbee.org/`. 2015 (cited on p. 60).

[159] R. Zůvala, E. Fišerová, and L. Marek. “Noise mapping based on participative measurements.” In: *Open Geosciences* 8.1 (Jan. 2016), pp. 140–156. ISSN: 2391-5447. DOI: `10.1515/geo-2016-0023` (cited on p. 57).