Departamento de Informática de Sistemas y Computadores

Mobile Sensing Architecture for Air Pollution Monitoring

Thesis submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy in Computer Science

By

Oscar Alvear Alvear

Advisor:

Dr. Carlos Tavares Calafate

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Valencia, Spain

July, 2018

To Ana Gabriela, Óscar
Sebastian, and my parents.

They are my life.

We are what we repeatedly do;
excellence, then, is not an act,
but a habit.

Will Durant

Acknowledgements

I WANT to start by sincerely thanking my tutor, Dr. Carlos Tavares Calafate, for his friendship as well as his meticulous support and guidance along the development of this thesis.

In the same way, I want to thank the GRC (Networking Group) professors, especially Dr. Pietro Manzoni, for opening me the laboratory doors, and allowing me to work in the group; to Juan Carlos Cano for his support and corrections, and to Enrique Hernandez-Orallo for his uninterested help.

I also want to thank Dr. Enrico Natalizio and Dr. Nicola Zema from UTC (Université de Technologie de Compiègne), for receiving me during my internship, and for guiding me in the development of the research work, as well as to the Heudiasyc Group members for welcoming me as another member, and for sharing great moments.

A special thanks to GRC members, all of them, for their friendship, and for allowing me to share fantastic moments both inside and outside the laboratory, during the "Charlas On the Go" meetings, and in general on all GRC events.

To Ana Gabriela, my wife, for encouraging me to undertake new challenges, and for sharing this experience with me. To my son, Oscar Sebastian, for teaching me the most important lesson: to be a father. And to my parents for their unconditional love and support.

Finally, I want to thank the Ecuatorian Republic through the "Secretaría de Educación Superior, Ciencia, Tecnología e Innovación" (SENESCYT), for granting me the scholarship to finance my studies.

Oscar Alvear Alvear
Valencia, July 18, 2018

Abstract

INDUSTRIAL GROWTH has brought unforeseen technological advances to our society. Unfortunately, the price to pay for these advances has been an increase of the air pollution levels worldwide, affecting both urban and countryside areas. It means that air quality monitoring becomes relevant not only in cities, but also in rural environments because it directly affects citizens, crops, and the life of various animals/insects. Thus, different solutions for measuring air quality should be sought for such heterogeneous environments.

Typically, air pollution monitoring relies on fixed monitoring stations to carry out the pollution control. However, this method is too expensive, not scalable, and hard to implement in any city. So, the Mobile Crowdsensing (MCS) approach, by allowing to embed a small sensor in any sort of vehicle, becomes one of the most practical strategies due to its relatively easy and fast deployment in any place.

Concerning the widespread use of small pollution monitoring sensors embedded in mobile vehicles, the possible scenarios can be divided into two: urban scenarios, where a wide set of vehicles are available, and rural and industrial areas, where vehicular traffic is scarce and limited to the main transportation arteries.

Considering these two scenarios, in this thesis we propose an architecture, called EcoSensor, to monitor the air pollution using small sensors installed in vehicles, such as bicycles, private cars, or the public transportation system, applicable to urban scenarios, and the use of an Unmanned Aerial System (UAS) in rural scenarios.

Three main components compose our architecture: a low-cost sensor to capture pollution data, a smartphone to preprocess the pollution information and transmit the data towards a central server, and the central server, to store and process pollution information.

For urban scenarios, we analyze different alternatives regarding the design of a low-cost sensing unit based on commercial prototyping platforms such as Raspberry Pi or Arduino, and Commercial Off-the-shelf (COTS) air quality sensors.

Moreover, we analyze and propose a process to perform pollution monitoring using our architecture. This process encompasses four basic operations: data reading, unit conversion, time variability reduction, and spatial interpolation.

For rural scenarios, we propose the use of an Unmanned Aerial Vehicle (UAV) as a mobile sensor. Specifically, we equip the UAV with sensing capabilities through a Raspberry Pi microcomputer and low-cost air quality sensors.

Finally, we propose an algorithm, called Pollution-driven UAV Control (PdUC), to control the UAV flight for monitoring tasks by focusing on the most polluted areas, and thereby attempting to improve the overall accuracy while minimizing flight time. We then propose an improvement to this algorithm, called Discretized Pollution-driven UAV Control (PdUC-D), where we discretize the target area by splitting it into small tiles, where each tile is monitored only once, thereby avoiding redundant sampling.

Overall, we found that mobile sensing is a good approach for monitoring air pollution in any environment, either by using vehicles or bicycles in urban scenarios, or an UAVs in rural scenarios. We validate our proposal by comparing obtained values by our mobile sensors against typical values reported by monitoring stations at the same time and location, showing that the results are right, matching the expected values with a low error. Moreover, we proved that PdUC-D, our protocol for the autonomous guidance of UAVs performing air monitoring tasks, has better performance than typical mobility models in terms of reducing the prediction errors and reducing the time to cover the whole area.

Resumen

El crecimiento industrial ha acarreado grandes avances tecnológicos para nuestra sociedad. Lamentablemente, el precio a pagar por estos avances ha sido un aumento significativo de los niveles de contaminación del aire en todo el mundo, afectando tanto a zonas urbanas como a las zonas rurales. Hoy en día, la monitorización de la calidad del aire se ha convertido en un aspecto muy relevante no solo en las ciudades, sino también en los entornos rurales ya que afecta directamente a las personas, a los cultivos, y en general a ecosistemas y vida de diversos animales. Para abordar con éxito el problema de la contaminación se deben buscar diferentes soluciones que permitan medir la calidad del aire en todos estos entornos.

Por lo general, la monitorización de la calidad aire se realiza mediante estaciones de monitorización fijas. Sin embargo, este método es demasiado costoso, poco escalable y difícil de implementar en nuestras ciudades, las cuales están cada vez más pobladas. El uso de Mobile CrowdSensing (MCS), paradigma en el cual la monitorización se realiza por los propios usuarios, se convierte en una de las estrategias más prácticas en la actualidad debido a que se puede desplegar de manera relativamente rápida, sencilla y eficiente en cualquier lugar, simplemente instalando un pequeño sensor en cualquier tipo de vehículo.

En cuanto al uso generalizado de sensores móviles de contaminación ambiental integrados en vehículos, los posibles escenarios se pueden dividir en dos: entornos urbanos, donde hay un amplio conjunto de vehículos disponibles, y entornos rurales o industriales, donde el tráfico vehicular es escaso y está limitado a las principales arterias de transporte.

Teniendo en cuenta estos dos escenarios, esta tesis propone una arquitectura, llamada EcoSensor, que permite monitorizar la contaminación del aire utilizando pequeños sensores de bajo coste instalados en diferentes tipos de vehículos, tales como bicicletas, automóviles o autobuses del sistema de transporte público, en el caso de entornos urbanos, y en drones o UAS (Unmanned Aircraft Systems) en

entornos rurales.

La arquitectura propuesta está compuesta por tres componentes: un sensor de bajo coste para capturar datos de contaminación, un smartphone para realizar un preprocesamiento de la información y para transmitir los datos hacia un servidor central, y el servidor central, encargado de almacenar y procesar la información de contaminación ambiental.

Para entornos urbanos, analizamos diferentes alternativas con respecto al diseño de una unidad de monitorización (sensor móvil) de bajo coste basada en plataformas de prototipado comerciales como Raspberry Pi o Arduino, junto con sensores también de precio reducido.

En la tesis realizamos un análisis, y proponemos un proceso, para llevar a cabo la monitorización ambiental utilizando la arquitectura propuesta. Este proceso abarca cuatro operaciones básicas: captura de datos, conversión de unidades, reducción de la variabilidad temporal, e interpolación espacial.

Para entornos rurales, proponemos el uso de drones (UAV) como unidades de sensorización móviles. Específicamente, equipamos el drone con capacidades de monitorización a través de un microordenador Raspberry Pi y sensores de calidad del aire de bajo coste.

Finalmente, se propone un algoritmo llamado PdUC (Pollution-driven UAV Control) para controlar el vuelo del UAV con el objetivo de realizar monitorización ambiental, identificando las áreas más contaminadas, y tratando de ese modo de mejorar la precisión general y la velocidad de monitorización. Además, proponemos una mejora a este algoritmo, denominada PdUC-D, basada en la discretización del área a monitorizar dividiéndola en pequeñas áreas (tiles), donde cada tile se monitoriza una sola vez, evitando así realizar muestreos redundantes.

En general, verificamos que la monitorización móvil es una aproximación eficiente y fiable para monitorizar la contaminación del aire en cualquier entorno, ya sea usando vehículos o bicicletas en entornos urbanos, o UAVs en entornos rurales. Con respecto al proceso de monitorización ambiental, validamos nuestra propuesta comparando los valores obtenidos por nuestros sensores móviles de bajo coste con respecto a los valores típicos de referencia ofrecidos por las estaciones de monitorización fijas para el mismo período y ubicación, comprobando que los resultados son semejantes, y están acuerdo a lo esperado. Además, demostramos que PdUC-D, permite guiar autónomamente un UAV en tareas de monitorización del aire, ofreciendo un mejor rendimiento que los modelos de movilidad típicos, reduciendo tanto los errores de predicción como el tiempo para cubrir el área completa, y logrando una mayor precisión dentro de las áreas más contaminadas.

Resum

El creixement industrial ha implicat grans avanços tecnològics per a la nostra societat. Lamentablement, el preu que cal pagar per aquests avanços ha sigut un augment significatiu dels nivells de contaminació de l'aire a tot el món, que afecta tant zones urbanes com zones rurals. Avui dia, el monitoratge de la qualitat de l'aire s'ha convertit en un aspecte molt rellevant no solament a les ciutats, sinó també en els entorns rurals, ja que afecta directament les persones, els cultius i, en general, els ecosistemes i la vida de diversos animals. Per a abordar amb èxit el problema de la contaminació s'han de cercar diverses solucions que permeten mesurar la qualitat de l'aire en tots aquests entorns.

En general, el monitoratge de la qualitat aire es fa mitjançant estacions de monitoratge fixes. No obstant això, aquest mètode és massa costós, poc escalable i difícil d'implementar a les nostres ciutats, les quals estan cada vegada més poblades. L'ús de Mobile CrowdSensing (MCS), paradigma en el qual el monitoratge el duen a terme els mateixos usuaris, es converteix en una de les estratègies més pràctiques en l'actualitat a causa que es pot desplegar de manera relativament ràpida, senzilla i eficient en qualsevol lloc, simplement instal·lant un petit sensor en qualsevol tipus de vehicle.

Pel que fa a l'ús generalitzat de sensors mòbils de contaminació ambiental integrats en vehicles, els possibles escenaris es poden dividir en dos: entorns urbans, on hi ha un ampli conjunt de vehicles disponibles, i entorns rurals o industrials, on el trànsit vehicular és escàs i està limitat a les principals artèries de transport.

Tenint en compte aquests dos escenaris, aquesta tesi proposa una arquitectura, anomenada EcoSensor, que permet monitorar la contaminació de l'aire utilitzant petits sensors de baix cost instal·lats en diferents tipus de vehicles, com ara bicicletes, automòbils o autobusos del sistema de transport públic, en el cas d'entorns urbans, i en UAVs (Unmanned Aerial Vehicles) en entorns rurals.

L'arquitectura proposada està composta per tres components: un sensor de baix cost per a capturar dades de contaminació, un smartphone per a realitzar un

preprocessament de la informació i per a transmetre les dades cap a un servidor central, i el servidor central, encarregat d'emmagatzemar i processar la informació de contaminació ambiental.

Per a entorns urbans, analitzem diferents alternatives pel que fa al disseny d'una unitat de monitoratge (sensor mòbil) de baix cost basada en plataformes de prototipatge comercials com Raspberry Pi o Arduino, juntament amb sensors també de preu reduït.

En la tesi fem una anàlisi, i proposem un procés, per a dur a terme el monitoratge ambiental utilitzant l'arquitectura proposada. Aquest procés abasta quatre operacions bàsiques: captura de dades, conversió d'unitats, reducció de la variabilitat temporal, i interpolació espacial.

Per a entorns rurals, proposem l'ús de drons o Unmanned Aerial Vehicles (UAVs) com a unitats de sensorització mòbils. Específicament, equipem el dron amb capacitats de monitoratge a través d'un microordinador Raspberry Pi i sensors de qualitat de l'aire de baix cost.

Finalment, es proposa un algorisme anomenat PdUC (Pollution-driven UAV Control) per a controlar el vol del UAV amb l'objectiu de realitzar monitoratge ambiental, que identifica les àrees més contaminades i que, d'aquesta manera, tracta de millorar la precisió general i la velocitat de monitoratge. A més, proposem una millora a aquest algorisme, denominada PdUC-D, basada en la discretització de l'àrea a monitorar dividint-la en xicotetes àrees (tiles), on cada tile es monitora una sola vegada, fet que evita dur a terme mostrejos redundants.

En general, verifiquem que el monitoratge mòbil és una aproximació eficient i fiable per a monitorar la contaminació de l'aire en qualsevol entorn, ja siga usant vehicles o bicicletes en entorns urbans, o UAVs en entorns rurals. Pel que fa al procés de monitoratge ambiental, validem la nostra proposta comparant els valors obtinguts pels nostres sensors mòbils de baix cost pel que fa als valors típics de referència oferits per les estacions de monitoratge fixes per al mateix període i ubicació, i es comprova que els resultats són semblants, i estan d'acord amb el resultat esperat. A més, es demostra que PdUC-D permet guiar autònomament un UAV en tasques de monitoratge de l'aire, oferint un millor rendiment que els models de mobilitat típics, reduint tant els errors de predicció com el temps per a cobrir l'àrea completa, i aconseguint una major precisió dins de les àrees més contaminades.

Contents

List of Figures

List of Tables

Part I

Introduction and Context

Chapter 1

Introduction

In the world we live in, the continuous growth of cities, industries, automotive parks, etc., are causing air pollution to become one of the most important issues affecting people's lives. In fact, poor air quality is associated with several problems affecting people life including health issues (mainly in the respiratory tracts), climate changes, and reduced agriculture production, among others.

Air pollution mainly consists of the emission of gases or particles into the atmosphere, producing changes in its composition. It thereby affects not only humans, but life in general. In addition, such effects take place not only in industrial areas, but also in residential areas and in the countryside, meaning that lifestyle and health of people is at stake, as well as the crops, the natural cycles of animals and plants, the pollination cycle of bees, etc.

With these considerations in mind, air pollution monitoring becomes an issue of utmost importance for governments, private entities, and people in general. In fact, it is necessary to maintain pollution levels under control in any place, worldwide.

Traditionally, the air quality control is carried out by using fixed monitoring stations, which are placed throughout our cities to control pollution levels. Even though these stations have sophisticated sensors of very high accuracy, they only measure representative data for the area near to their location. In general, it becomes too hard to deploy enough stations so as to maintain a good control over an entire city, or in regions with poor accessibility such as forests, large countryside areas, or catastrophic areas. Thus, alternative solutions must be sought to fill-in for this gap.

Moreover, there are other methods to monitor air pollution such as Remote

Sensing using mainly satellites or aircrafts to capture data or, Wireless Sensor Networks. However, these methods also need an expensive infrastructure to carry out their job.

An alternative for measuring environmental pollution in a relatively cheap way is relying on mobile crowdsensing. Specifically, small low-cost devices can be installed in different types of vehicles to monitor different parts of the city at different times while moving.

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, among other problems that must be analyzed.

On the other hand, crowdsensing approaches are, so far, mostly applicable to urban areas where different mobility alternatives are available. However, in rural and industrial zones, deployment options are quite more limited.

In the case of large rural or industrial areas, a fleet of autonomous vehicles could be efficiently used to cover the vast distances associated with them. Furthermore, the use of autonomous sensor carriers is even more encouraged in this case due to the following considerations: (i) the relative absence of civilian population to be taken care of during robotic operations; (ii) stable and expectable obstacle locations; (iii) less constraints in terms of Unmanned Aerial Vehicle (UAV) flight laws; and finally (iv) safety and security concerns, as some areas could be dangerous to access for human operators. Since, in these environments, ground access is usually hindered and full of obstacles, the most feasible way to implement a fleet of mobile pollution-monitoring robots is via Unmanned Aerial Vehicles (UAVs).

1.1 Objectives

The main objective of this thesis is to propose and implement an architecture for air pollution monitoring able to offer fine-grained resolution data, and applicable in both urban and rural scenarios. To this aim the mobile sensors developed shall be installed in different types of vehicles, such as bicycles or the public transport system when operating in urban areas, or in UAVs when operating in areas with poor accessibility, like the countryside or catastrophic zones.

To achieve this main objective it becomes necessary to accomplish several specific objectives for both crowdsensing in urban areas, and UAV-based sensing in rural areas.

Crowdsensing:

- Analyze the data sampling process using mobile sensors.
- Analyze and Design a low-cost prototype sensor for air pollution monitoring adapted to mobile sensing.

- Design a sensing architecture, along with its different elements.
- Analyze the data transmission to the central server for data processing.
- Use the appropriate statistical tools for data interpolation tasks.
- Deploy a system to process the monitored data, and to show the air pollution distribution in a target region.

UAV-based sensing:

- Design a low-cost UAV able to perform air pollution monitoring tasks.
- Develop an algorithm for autonomous UAV guidance considering both air pollution aspects and UAV restrictions/capabilities.

1.2 Structure of the Thesis

The thesis dissertation follows a methodology encompassing both theoretical and practical issues. Thus, on the one hand, we do a fundamental research study (theoretical), analyzing the sensor behaviour and the air pollution measurement issue, defining the different architectural elements for air pollution monitoring based on a mobile sensing approach. On the other hand, we do applied research, developing a mobile sensor prototype, and deploying a system to monitor the air pollution in the city of Valencia.

This dissertation is organized in 10 chapters. Below, we briefly describe the contents of each part:

- **Chapter 2. Air Pollution and its Monitoring process: an overview**: we provide a review of general aspects related to air pollution monitoring.
- **Chapter 3. Mobile Sensing technologies for Air Pollution Monitoring**: we present an overview of the most relevant IoT architectures and protocols.
- **Chapter 4. The EcoSensor Platform**: we present a crowdsensing architecture applicable to urban environments that is able to offer mobile pollution sensing with high spatial resolution. The architecture includes three independent 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 using the collected data through spatial interpolation techniques.
- **Chapter 5. Mobile sensors design**: we analyze the different options for creating a small, low-cost, mobile sensor able to communicate with off-the-shelf smartphones in an Internet of Things type of environment.

- **Chapter 6. Finding the optimal measurement strategy**: We analyzed the influence of the sensor orientation in the data capture process, as well as the impact of time and spatial sampling. Moreover, we validated EcoSensor by comparing the values obtained by our mobile sensor with typical values from monitoring stations at the same dates and location.
- **Chapter 7. Mobile Pollution Data Sensing Using UAVs**: we propose the use of UAVs, specifically of the multicopter type, as an efficient solution for quickly and easily monitoring air quality in any region where ground mobility is a poor option.
- **Chapter 8. Pollution-driven UAV Control (PdUC)**: we propose the use of UAVs equipped with Commercial Off-the-shelf (COTS) devices and sensors to implement a service for air pollution monitoring that leverages the use of bio-inspired approaches as its main control strategy.
- **Chapter 9. Discretized Pollution-driven UAV Control (PdUC-D)**: we propose an optimized algorithm called PdUC-D, which is based on PdUC, but applies space discretization to substantially reduce the convergence time while achieving similar levels of accuracy.
- **Chapter 10. Conclusions & Publications**: we conclude this thesis, and present the publications related to the thesis, as well as a list of future research lines.

Chapter 2

Air pollution and its monitoring process: an overview

Environmental pollution consists of a degradation of the natural habitats such as forests, rivers, lakes, seas, and so on, by the introduction of waste or dangerous elements in the composition of soil, water and air. These elements could be in the form of physical components, biological material, chemical components, or even energy such as noise, heat, radioactivity or light, altering the normal environment behavior in a faster way than it can adapt to, thereby affecting humans and other living organisms. Usually, the environment adjusts itself to pollution through dispersion, breakdown, recycling, or storage in some harmless form, but when the pollution levels become too high, it is almost impossible for these processes to complete in a short period of time.

Despite environmental pollution can be caused by different natural sources, such as hurricanes, sandstorms, twisters, volcano eruptions, and so forth, its main source is related to human activities. Notice that, even though humans always have had an impact on nature in their day-to-day activities, in the past nature could handle their impact because the pollution levels were relatively low. However, since the industrial revolution, the number of factories increased rapidly, shifting to full-scale industries and manufacturing units that produce more pollution, and different pollutants types, becoming an overwhelming problem for the planet.

2.1 Environmental Pollution Classification

There are several types of pollution depending on its source, and their impact on our lives and the environment differs accordingly. Currently, we can classify pollution in eight basic types: (i) Air pollution, (ii) Water pollution, (iii) Soil pollution, (iv) Noise pollution, (v) Radioactive pollution, (vi) Thermal pollution, (vii) Light pollution, and (viii) Visual pollution.

Air pollution is the contamination of the atmosphere by changing the composition and chemistry of the air. It can be the excessive presence of gases, such as carbon dioxide, which cannot be removed naturally, or small particles floating suspended in the air, such as dust, PM10, or PM2.5.

Air pollution can be produced by natural sources such as volcanic eruptions, forest fires, dry soil erosion, dust storms, etc. However, nowadays, the principal cause of air pollution is human activity, being vehicular contamination, building construction, agricultural spraying, and industry very representative examples.

Depending on the air pollutant levels, several effects can be noticed in nature or the human health. For instance, they can cause an increase in smog, a higher rain acidity, crop depletion due to lack of oxygen, and higher rates of asthma for people exposed to it. Figure 2.1 illustrates an example of air pollution in urban areas.

Figure 2.1: Example of air pollution in urban areas[1]

Water pollution refers to the contamination of some water body, such as rivers, lakes, oceans, or underground reservoirs, from bacterial matter, chemicals or particles that change their composition, degrading water quality or purity.

[1]Source: Wikipedia::Pollution

Water pollution can be produced by increased sediment caused by soil erosion, improper waste disposal, and littering, leaching of soil pollution or organic material into water supplies, among others.

The main effects of water pollution could be: decreasing the amount of drinkable water available, lowering water supplies for crop irrigation, and impacting fish and wildlife populations that require good quality water. Figure 2.2 shows an example of water pollution in a port.

Figure 2.2: Example of water pollution in a port.[2]

Soil pollution refers to contamination of the land caused by some external material, such as toxins or chemicals, degrading the quality of the land surface, and thereby affecting its intended use.

Soil pollution could cause land desertification or soil erosion, reducing crop yields, loss of wildlife habitats, and so on.

The primary soil pollution sources are non-sustainable farming practices, such as the massive use of pesticides, hazardous waste and sewage spills, strip mining, deforestation, household dumping and littering, and other destructive practices. Figure 2.3 shows a example of soil pollution.

[2]Source: Wikipedia::Pollution

Figure 2.3: Example of polluted soil area (Source: Wikipedia::Pollution).

Noise pollution refers to the high levels of sound caused by human activities which affect life quality in a specific area.

It could affect people both physiologically and psychologically, including hearing loss in some extreme cases, and, more important, it can also interfere with basic activities such as sleep, rest, study, communication and socializing. On the other hand, it could affect wildlife, causing some disruption in the habit of the animals, thereby deteriorating their activities.

Noise pollution is caused mainly by traffic, railroads, airports, concerts, industrial activities, etc. Figure 2.4 shows an example of noise pollution caused by a plane in a residential area.

Figure 2.4: Example of noise pollution caused by an airplane (Source: Wikipedia::Pollution).

Light pollution refers to the over-illumination of an area that disrupts natural cycles, therefore being considered obtrusive. A good examples is the light caused by activities in large cities like New York, which almost never sleep.

Figure 2.5: Example of light pollution in urban areas (Source: Wikipedia::Pollution).

Some particular examples of light pollution could be billboards and advertising, night time sporting events, and other nighttime entertainment.

The light pollution could affect the sleep cycles in the human life, or the natural cycles in the wildlife. Figure 2.5 shows an example of light pollution caused by the illumination of the buildings in a city.

Thermal pollution refers to an artificial increase of the temperature in a particular ecosystem over long periods, altering the natural thermal cycles, and therefore causing undesirable effects.

Power plants, urban sprawl, deforestation, etc., could cause thermal pollution by affecting a specific area, but if there are a lot of similar sources, it can become a serious concern for the whole world. For instance, nowadays, global warming is one of the most significant problems humanity has to face. Figure 2.6 shows an example of thermal pollution caused by an industrial process.

Figure 2.6: Example of thermal pollution caused by industrial process (Source: Wikipedia::Pollution).

Radioactive pollution refers to the exposition of some region to radioactivity. It is rare, but extremely dangerous, possibly becoming deadly if levels are high. Thus, governments usually have a strict regulation to control or prevent radioactive pollution.

The sources of radioactivity pollution can be nuclear power plants, nuclear waste, uranium mining operations, etc.

Radioactive pollution can cause many illnesses such as congenital disabilities, cancer, sterilization, and other health problems for human and wildlife populations, becoming critical in some extreme cases such as Chernobyl or Fukushima. Figure 2.7 shows an example of radioactive pollution waste.

Figure 2.7: Example of radioactive pollution waste (Source: Wikipedia::Pollution).

2.2 Air Pollution

Industrial growth has brought unforeseen technological advances to our societies. Unfortunately, the price to pay for these advances has been an increase of air pollution worldwide associated to the different production stages.

As introduced in the previous section, air pollution basically consists in the emission of gases or particles into the atmosphere, producing changes in its composition that affect the environment, degrading people's life quality as well as wildlife, and agriculture.

Air pollution could be classified into two categories depending on the emitted pollutants:

Primary pollutants are gases or particles emitted directly into the atmosphere, such as CO or CO_2, which are emitted into the atmosphere by vehicles or industrial processes.

Secondary pollutants are gases produced by a chemical reaction between primary pollutants and some other element; in this second category we have for instance Ozone (O_3), which is produced by the combination of Nitrogen Oxides (NO_x), Oxygen (O_2), Volatic Organic Compound (VOC), and sunlight.

Even though determining the level of primary pollutants could be achieved by taking into account their sources, the level of secondary pollutants is much more

difficult to determine, as its emission and creation involves different sources.

2.2.1 Main air pollutants

According to the U.S. Environment Protection Agency (EPA), in current mega cities there are six main air pollutants:

Ozone (O_3) is a gas composed by three atoms of oxygen, which is produced in the Earth's upper atmosphere (good ozone), and at ground level (Pollutant - bad ozone).

The good ozone, called stratospheric ozone, is produced in the upper atmosphere, creating a protective layer that shields us from the sun's harmful ultraviolet rays. It has been partially destroyed by manmade chemicals, causing what is sometimes called a "ozone hole".

The bad ozone, or tropospheric ozone, is that which occurs at ground level, and that is produced by the chemical reaction between Nitrogen Oxides (NO_x), Oxygen (O_2), Volatic Organic Compound (VOC), and sunlight. The major concentrations of tropospheric ozone are near to high-density traffic, power plants, industrial boilers, refineries, or other chemical sources. Since it is produced by the reaction with sunlight, the tropospheric ozone levels are higher in sunny days during summer or spring.

It is worth mentioning that tropospheric ozone is the main ingredient of the "smog", affecting both human health and the natural environment.

Particle Matter (PM) is the presence of small solid particles floating in the air that are much smaller than dust, dirt, soot, or smoke, as the latter are large and dark enough to be seen with the human eye. In general PM particles can be detected using an electron microscope.

They can be classified in two subcategories depending on their size:

PM_{10}: Particle matter smaller than 10 micrometers.

$PM_{2.5}$: Particle matter smaller than 2.5 micrometers.

These particles are harmful because they are too small, and can easily be inhaled, causing different illness including asthma attacks, heart attacks, strokes and early death. They can be made up of hundreds of different chemicals emitted directly from sources such as construction sites, unpaved roads, fields, smokestacks or fires.

Carbon Monoxide (CO) is a gas composed by an atom of Carbon and an atom of Oxygen. It is a colorless, odorless gas that can be harmful when inhaled in large amounts. CO is produced by burning processes taking place both indoors and outdoors. When outdoors, it is mainly produced by the burning of fossil fuel by cars, trucks and other vehicles or machinery, and inside a home, it is produced by a variety of items such as kerosene and gas heaters, leaking chimneys, and furnaces.

Breathing air contaminated with Carbon Monoxide reduces the amount of oxygen that can be transported by the bloodstream to critical organs such as the

heart and the brain, causing several illnesses. Inhaling very high levels of these gases can cause dizziness, confusion, unconsciousness, and even death.

Lead (Pb) is a mineral which can be emitted onto the air by different sources. The major sources of lead are ore and metals processing, and piston-engine aircraft operating on leaded aviation fuel. Other sources include waste incinerators, utilities, and lead-acid battery manufacturers.

Exposition to lead can affect human health (nervous system, kidney function, immune system, reproductive and developmental systems, and the cardiovascular system), as well as the natural ecosystem, decreasing growth and reproductive rates in plants and animals, and having neurological effects in vertebrates.

Sulfur Dioxide (SO_2) is a gas composed by an atom of sulfur and two atoms of oxygen. Despite the fact that the entire group of Sulfur Oxides (SO_x) are dangerous, SO_2 is the component of most significant concern, being used as the indicator for the entire group. The most extensive sources of SO_2 emissions are fossil fuel combustion at power plants and other industrial facilities.

This gas can affect both human health and the environment. Short-term exposures to SO2 can harm the human respiratory system. It is more dangerous for sensitive groups such as children, the elderly, and people with asthma. Also, it contributes to the creation of small Sulfur particles (PM_{10} or $PM_{2.5}$)

Nitrogen Dioxide (NO_2) is a gas composed by an atom of Nitrogen and two atoms of oxygen, and it belongs to a group of highly reactive gases known as Nitrogen Oxides (NO_x). Nitrogen Dioxide (NO_2) is used as the indicator for the larger group of nitrogen oxides.

Nitrogen Dioxide (NO_2) primarily gets in the air from the burning of fuel by cars, trucks and buses, power plants, and off-road equipment.

This gas can react with water, oxygen and other chemicals in the atmosphere, creating acid rain.

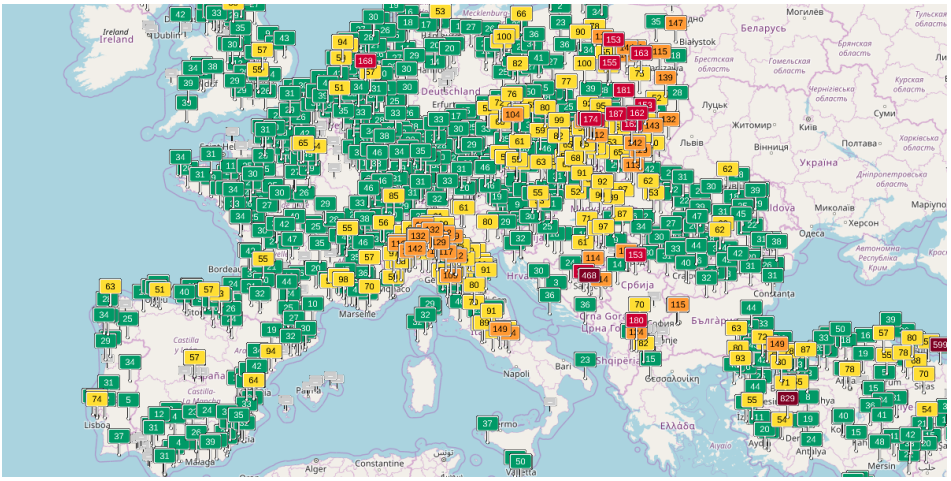
2.3 Air Pollution Monitoring worldwide

Air pollution monitoring is a critical issue, being nowadays a great concern of major cities since it has severe consequences on human health. Thus, modern cities, especially megacities, should address Air Pollution Monitoring as a prominent service.

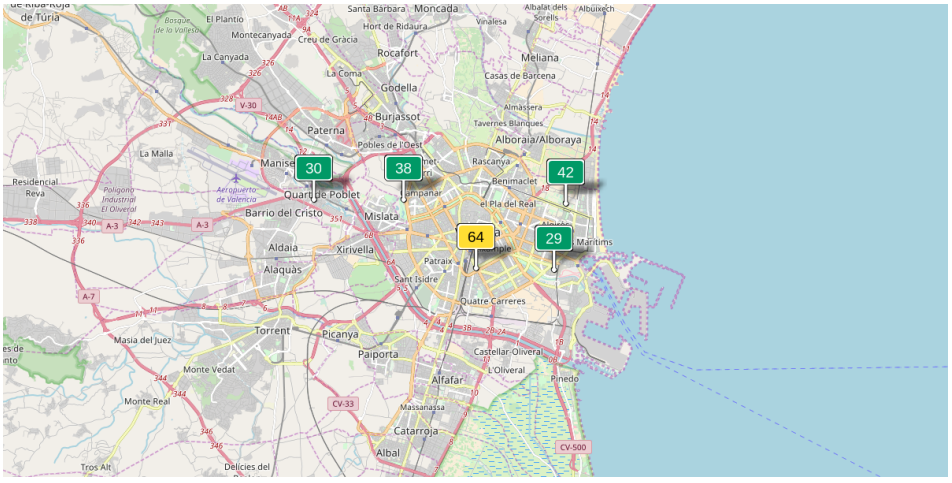
In recent decades, air pollution monitoring has gained worldwide relevance. There are many research works that study the effects of pollution on our health. Among them we can find the contributions of Chen at al. [33, 32], who analyzed the effects of ozone and particle matter on human health. Brook et al. [30] also contributed to this field by studying the relationship between the exposure to air pollution (including ozone) and cardiovascular events.

Due to these reasons, different agencies have been created worldwide. So, the United States of America has created the EPA [114], which is responsible for

tracking the evolution of environmental pollution in the whole country. Similarly, throughout 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. These stations are able to measure pollutants like O_3s, fine PMs, Carbon Dioxides (CO_2s), and COs, among others.



(a) Air pollution monitoring stations throughout Europe.



(b) Air pollution monitoring stations in Valencia, Spain.

Figure 2.8: Geographical distribution of monitoring stations. Source: http://aqicn.org

The **EPA** [114] was born in 1970 to address the raising concerns about envi-

ronmental pollution, and to carry out a variety of federal research, monitoring, standards-setting and enforcement activities to ensure environmental protection mainly in the USA.

The **European Environment Agency (EEA)** is an agency of the European Union whose task is to provide information on the environment. The EEA aims to support sustainable development by helping to achieve significant and measurable improvement in Europe's environment, through the provision of timely, targeted, relevant and reliable information to policy-making agents and the public.

The European environment information and observation network (Eionet) is a partnership network of the European Environment Agency and its member and cooperating countries. Through Eionet, the EEA brings together environmental information from individual countries, concentrating on the delivery of timely, nationally validated, high-quality data.

Figure 2.8 shows the distribution of the monitoring stations (a) in Europe and (b) in the city of Valencia, Spain.

2.3.1 Air Quality Index

Taking into account the effects of air pollution in human health, the Environmental Protection Agency has created an index to report the air quality called Air Quality Index (AQI) [2]. It defines some categories related to the amount of pollutants, and the risks for human health.

AQI defines an index between 0 to 500, where higher index values represent higher pollutant levels and, obviously, higher human health risks. There are six ranges established in this index: (i) 0 - 50, there are no risks. (ii) 51 - 100, moderate risk, (iii) 101 - 150, unhealthy for sensitive groups, (iv) 151 - 200, Unhealthy for all people, (v) 201 - 300, very high risk, very unhealthy, and (vi) over 300, hazardous.

Table 2.1 shows the description of the AQI index establishing the different levels.

Table 2.1: Air Quality Index.

Air Quality Index (AQI) Values	Levels of Health Concern	Colors
0 to 50	Good	Green
51 to 100	Moderate	Yellow
101 to 150	Unhealthy for Sensitive Groups	Orange
151 to 200	Unhealthy	Red
201 to 300	Very Unhealthy	Purple
301 to 500	Hazardous	Maroon

The AQI focuses on health effects after breathing polluted air during some period of time. The EPA calculates the AQI for five major air pollutants: Ozone (O_3), Particle Matter (PM) (both PM_{10} and $PM_{2.5}$), Carbon Monoxide (CO), Sulfur Dioxide (SO_2), and Nitrogen Dioxide (NO_2). For each of these pollutants, EPA has established national air quality standards to protect public health. Table 2.2 shows a summary of indexes related to the pollutant levels in some exposition periods.

Table 2.2: Air Quality Index for main pollutants.

AQI	O_3 *ppb* avg. 8h	PM_{10} $\mu g/m_3$ avg. 24h	$PM_{2.5}$ $\mu g/m_3$ avg. 24h	CO *ppm* avg. 8h	SO_2 *ppb* avg. 1h	NO_2 *ppb* avg. 1h
0 to 50	0 - 54	0 - 54	0 - 12	0 - 4.4	0 - 35	0 - 53
51 to 100	55 - 70	55 - 154	12.1 - 35.4	4.5 - 9.4	36 - 75	54 - 100
101 to 150	71 - 85	155 - 254	35.5 - 55.4	9.5 - 12.4	76 - 185	101 - 360
151 to 200	86 - 105	255 - 354	55.5 - 150.4	12.5 - 15.4	186 - 304	361 - 649
201 to 300	106 - 200	355 - 424	150.5 - 250.4	15.5 - 30.4	305 - 604	650 - 1244
301 to 500	–	425 - 605	250.5 - 500	30.5 - 50.4	605 - 1004	1245 - 2044

2.4 Air Pollution Monitoring technologies

Currently, controlling pollution levels is an on-going effort undertaken by most European cities, which invest a considerable amount of money in controlling the different hazards produced by poor air quality. This process mainly relies on fixed monitoring stations distributed along the cities, which is the traditional method to monitor the air pollution [70]. Alternatively, thanks to the last technological advances, there are other methods to monitor air pollution such as Remote Sensing, Wireless Sensors Networks, or Crowdsensing. All of these alternatives try to solve some inconvenient of the fixed stations.

Fixed base 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 [15]). 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.

Remote Sensing (RS) refers to the methods to acquire the information from some phenomenon such as pollution without making human contact with the object. Commonly, remote sensing is associated to the use of satellites or aircraft-based sensors to monitor different phenomena in inaccessible or poorly accessible areas like oceans, forest, catastrophic areas, etc.

Figure 2.9: Air Pollution Monitoring station located at Universitat Politecnica de Valencia, Valencia, Spain.

This technology started in military applications, but it was extended to different civil areas such as hydrology, ecology, oceanography, glaciology, geology, etc.

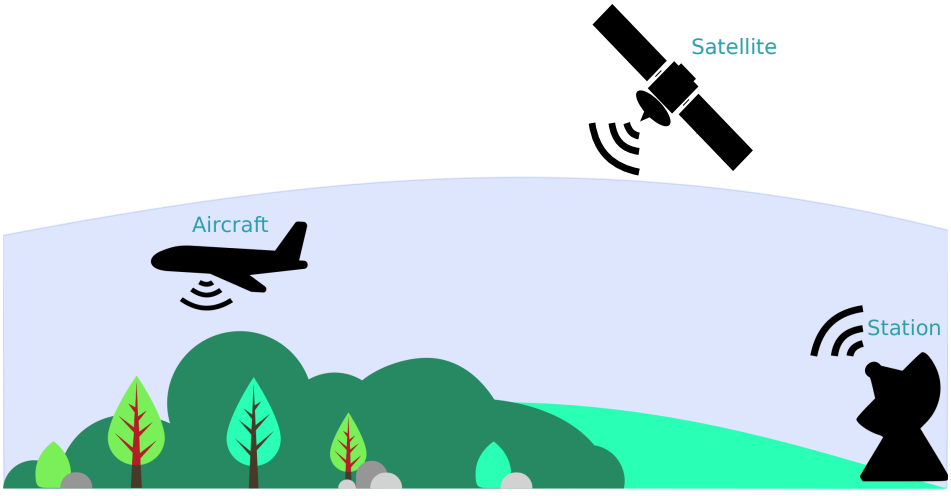


Figure 2.10: Remote Sensing approach.

Wireless Sensor Network (WSN) is a group of sensors which collaborate to measure some phenomenon, and transmit the sampled data wirelessly toward a main location, where data is processed.

In the same way than remote sensing, the WSN began with military purposes, but currently it is used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

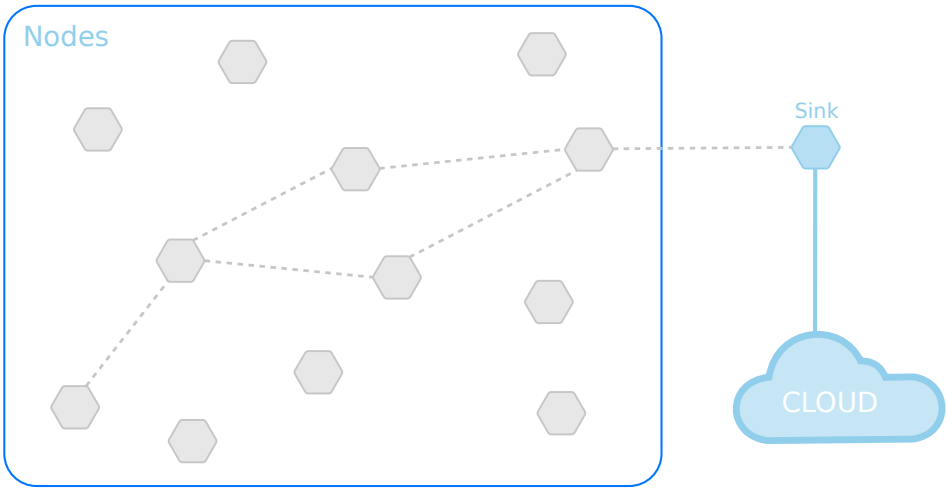


Figure 2.11: Wireless Sensor Networks example.

Crowdsensing refers to the collaborative sensing by the end users. So, a significant number of users perform collaborative sensing tasks, thereby collecting data from different populated locations while doing their daily activities. The collected data is periodically transmitted to a central server (Cloud) for data storage and processing.

An alternative for measuring environmental pollution relies 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.

Figure 2.12: Crowdsensing approach.

2.5 Summary

In this chapter, we analyzed the most important concepts related to Air Pollution and Air Pollution Monitoring. First of all, we described the pollution types, focusing next on the Air Pollution, specifically on those components considered by the EPAs as that main air pollutants. Afterward, we analyzed how air pollution monitoring is currently carried out through fixed stations. Also, we presented the Air Quality Index related to each pollutant described by the EPA. Finally, we analyzed the different alternatives to fixed stations for carrying out air pollution monitoring, including Wireless Sensor Network (WSN), Remote Sensing (RS), and Crowdsensing.

Chapter 3

Mobile Sensing technologies for Air Pollution Monitoring

Smart cities are revolutionizing our view of the world, and their functioning achieves a very high level of integration, coordination, and cooperation between ordinary objects, providing them with some degree of intelligence. This novel paradigm provides a plethora of systems and technological tools aimed at increasing our life quality, minimizing the environmental impact of everyday activities, and optimizing resource usage. Such effects are more noticeable in urban areas with millions of citizens, the so called *mega cities*, which in the near future will be more and more common [53]. The main concept behind a Smart City is the integration of the physical world with the virtual world [22]. This is achieved by providing additional capabilities such as environmental sensing and automatic behaviour to common objects, allowing to capture and to analyze data from the real world to ensure a better operation of the virtual one.

As shown by Gartner in its analysis of 2016 (see Figure 3.1), currently there are several emerging technologies to implement the concept of a Smart City. Focusing specifically in the monitoring field, different smart sensing solutions have emerged.

Figure 3.2 provides a global overview of available technologies from diverse perspectives, which cover different aspects that allow creating a Smart City. Thus, in a Smart City, all daily objects, called things, are equipped with extra capabilities, usually sensing and/or acting capabilities, along with communication capabilities, to share information, and to optimize their functional operation. This way, and from a communications perspective, the Internet of Things (IoT) [55] focuses on the intercommunication between all things, as well as on the communication be-

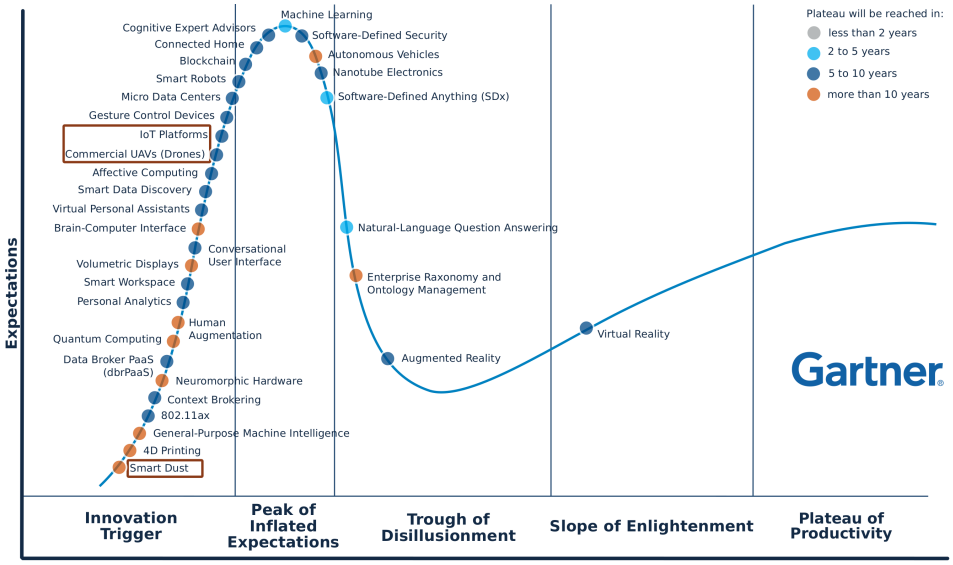


Figure 3.1: Gartner hype cycle for Emerging Technologies, 2016.

tween things and data servers (Cloud or Fog/Edge). On the other hand, from an operational perspective, Cyber Physical Systems (CPSs) focus on the integration of these physical things with the computational process [76, 120, 13] to improve its functionally. Finally, from a service perspective, Cloud Computing [98, 112] and Edge/Fog [42, 3, 126] Computing focus on the data processing, and on the structure of the Central servers or Local devices. In the remainder of this chapter, we will focus on the Internet of Things perspective, analyzing available technologies, and their adequacy in terms of implementation.

Any city has several areas of concern to the authorities. In a smart city, all of these areas must have some level of intelligence to minimize management efforts. Thus, there are various subareas of interest including Smart Governance, Smart Mobility, Smart Utilities, Smart Buildings, and Smart Environment, where the adoption of this paradigm can have a clear impact, being highly beneficial [31, 122].

Smart sensing refers to smart sensor devices monitoring some phenomenon, that are connected to a central server or to the Internet when trying to optimize the monitoring process. The sensor side, commonly called "edge computing", carries out the sensing and transmission information tasks, and the central server carries out the processing and presentation tasks. Moreover, the Internet of Things approach studies the intercommunication between smart devices.

For optimizing the monitoring tasks, we could consider the mobile sensing

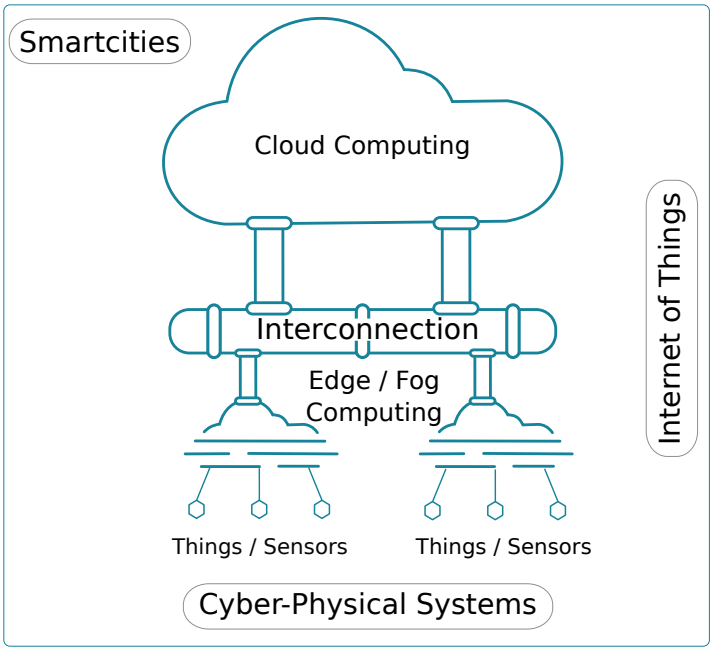


Figure 3.2: Smart city structure.

where a smart sensor is installed in some vehicle, and it moves around an area to monitor some phenomenon.

With respect to the widespread use of small monitoring sensors embedded in mobile vehicles, the possible scenarios can be divided into two main classes:

- **Urban environments**, where it is possible to embed the sensors on a wide set of vehicles like bicycles [45, 10] or cars [94];

- **Rural and industrial areas**, where vehicular traffic is scarce and limited to the main transportation arteries.

In the latter case, crowdsensing often fails to provide enough data to obtain realistic measurements having the required granularity.

Figure 3.3 shows a scheme of a Smart sensing approach focusing on mobile sensing environments.

In the next sections, we will analyze the different Internet of Things (IoT) scenarios, mobile crowdsensing in smart cities, and UAV-based sensing in rural areas.

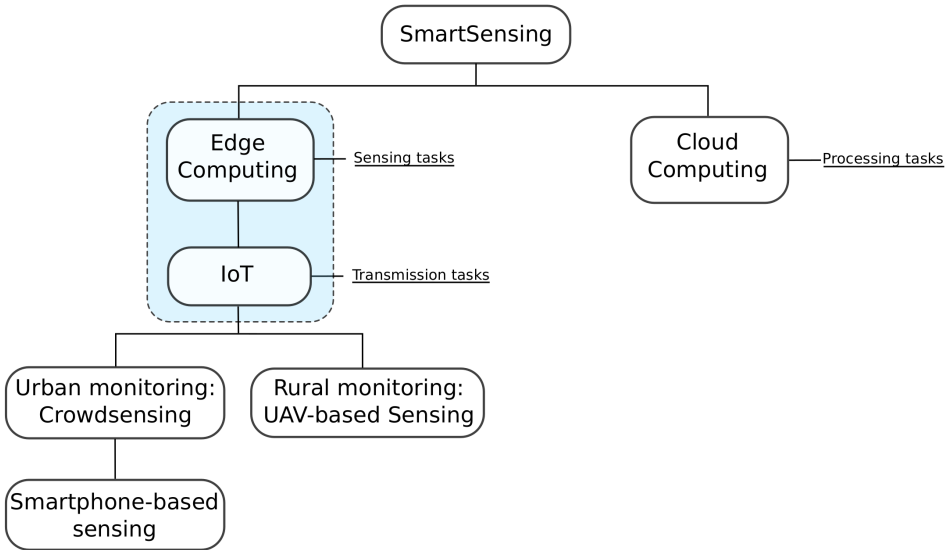


Figure 3.3: Smart sensing structure.

3.0.1 Overview of IoT Protocols

In recent years, the Internet of Things (IoT) has become one of the most challenging research topics, offering a wide range of novel solutions for Smart Cities [55, 72]. These proposals mainly analyze the intercommunication between devices, and involve a large variety of domains like home-based solutions [97], intelligent transportation systems [77], healthcare [41], safety and security [52, 62], industrial control [82], and environmental monitoring. Thus, the analysis of the sensor design must be able to cope with IoT protocols, which will be described in this section.

The main principle underlying the IoT paradigm is that all "things" are, or must be, connected to the Internet, and interact with each other to create developed areas that promote sustainability and enhancing life quality in multiple key areas [122, 12].

The main characteristic of these things is that they are constrained devices such as small sensors, hav ing restricted processing/storage capacity, restricted battery, restricted communication characteristics, i.e., Low Bandwidth, Low Data Rate, Low Coverage, etc. With this in mind, the Internet of Things has created a subset of protocols divided into various layers, similar to the traditional Internet stack, but taking into account the restrictions of the Internet of Things in terms of processing, battery capacity, and communication capabilities of embedded devices. Figure 3.4 summarizes the different layers defined for IoT, and the differences

towards the traditional Internet: (i) the Infrastructure Layer typically relies on wireless technologies, like ZigBee, Long Range Network (LoRa), or Bluetooth Low Energy (BLE); (ii) the Addressing Layer focuses on the analysis of the addressing issues to achieve compatibility with Internet protocols; (iii) the Transport Layer is the same than for the Internet Prototol Stack (TCP/IP), so either Transport Control Protocol (TCP) or User Datagram Protocol (UDP) are available, although UDP is typically used; (iv) the Messaging Layer defines protocols to transmit data towards the servers; (v) the Message Format Layer defines encoding types to store and transmit data; and (vi) the Semantic Layer defines the structure of the data.

Below, we describe the most important protocols involved in the Internet of Things at these different layers.

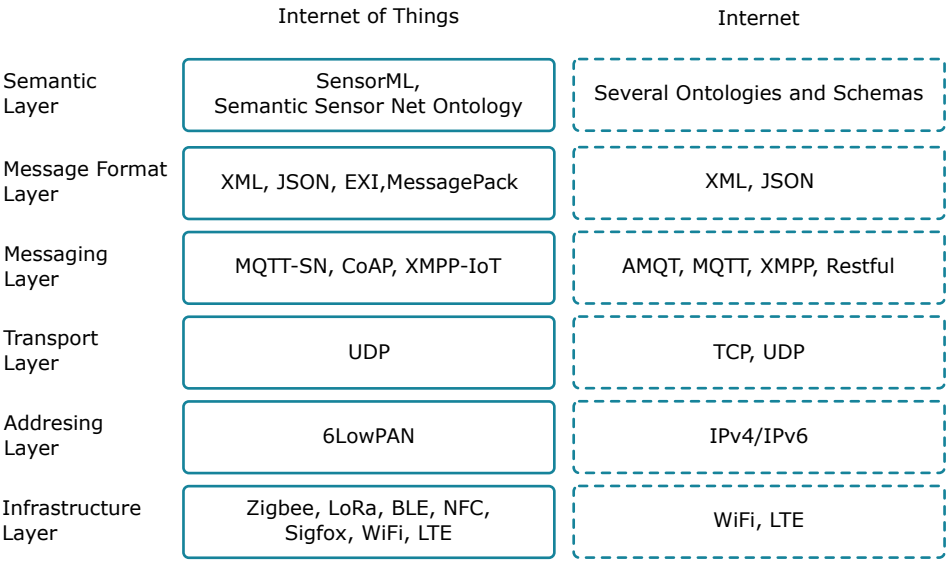


Figure 3.4: IoT protocols.

Infrastructure Layer

Currently, there are several communication technologies for the Internet of Things. Notice that, while any communications technology would allow us to create a network for IoT, not all of them take device restrictions into account. Theoretically, 5G allows a lot of possibilities to be offered in the context of the Internet of Things [46]. Similarly, new technologies such as LoRa or SIGFOX allow network sensors to remain connected due to their large coverage.

Below, we make a brief analysis of the possible wireless technologies for IoT.

Fifth Generation Network (5G) Network [57] is the fifth generation cellular network architecture, designed to support great amounts of data, high speed,

configurability, etc. from new emerging technologies such as Internet of Things. Currently, it is in the first phase, where new standards and services will be defined, but soon it shall become the default cellular network technology.

ZigBee [14] is based on the **IEEE 802.15.4** standard, and it was designed for Wireless Sensor Networks. Its main characteristics are its small size and low power consumption. Usually, the transmission range can vary from 10 to 100 m, depending on the output power. The main drawback of this technology, though, is that current smartphones are not equipped with ZigBee interfaces.

Wireless Fidelity (Wi-Fi) [11] is based on the IEEE 802.11 standard, and it was designed for Wireless Local Area Networks. The evolution of this technology provides several variants operating in the 2.4 GHz or in the 5 GHz band, being currently the 802.11n version the most widespread option. The transmission range for standard interfaces is about 100 m.

Long Range Network (LoRa) [107] is a Low Power Wide-Area Network (LPWAN) technology designed to optimize different aspects such as communication range, battery lifetime, and costs, supporting thousands of devices headed for the Internet of Things in several domains including sensing, metering, and Machine to Machine (M2M) communications.

Theoretically, LoRa achieves a transmission range of more than 15 km in rural environments, and of more than 2 km in dense urban areas. Its bandwidth ranges between 250 bps and 50 Kbps in different frequencies: 169 MHz, 433 MHz, and 868 MHz in Europe, and 915 MHz in North America.

SIGFOX [106] is an emerging technology that offers a proprietary telecommunications network to support Internet of Things solutions. It was designed for Low Power Wide Area (LPWA) networks operating in the ISM 868 MHz band, reaching distances greater than 1 km. Since the selected Industrial, Scientific and Medical (ISM) band is restricted, the communications could be of up to 12 bytes per message, and up to 140 messages per day.

Near Field Communications (NFC) [60] was designed for communications between two nearby devices (closer than 4 cm). Its main target applications are smartphone-based payments and IoT solutions such as access control, or inventory systems. However, its distance requirements and intermittent connectivity features make it a poor option for the main purpose of this thesis.

Bluetooth [21] was designed for Personal Area Networks (PANs), purposely having a maximum coverage range of 10 meters by default. Currently, it is used for transmitting information between personal devices, such as smartphones, smartwatches, and headsets.

Bluetooth Low Energy (BLE) [54], or **Bluetooth Smart**, is the name under which Bluetooth version 4 is known. Its main advantage, when compared to previous versions, is using very low power, being nowadays one of the best options for IoT applications. Similarly to previous Bluetooth specifications, the coverage range is of 10 m.

Addressing Layer

The addressing layer defines the logic address of the packets by assigning a specific address to all possible nodes. These protocols deal with the packet forwarding problem, which in the TCP/IP model is handled by Internet Protocol version 4 (IPv4), and/or Internet Protocol version 6 (IPv6).

IPv4/IPv6 [91, 40] are basic enabling protocols for the current Internet. The Internet Protocol (IP) is responsible for the per-hop relaying of datagrams, being in charge of addressing tasks (defining device addresses), being routing tasks (search/define routes to reach a destination) handled by other protocols.

There are two versions of this protocol: (i) Internet Protocol version 4 (IPv4), which uses 32 bits to specify a device address, failing to provide a pool of addresses that can serve all current and future devices, and (ii) Internet Protocol version 6 (IPv6), which uses 128 bits to specify a device address, allowing to address billions of devices. Both standards define the datagram structure, where the origin and destination are defined along with control data.

IPv6 over Low power Wireless Personal Area Network (6lowPAN) [87] allows to use IPv6 over networks based mainly in the IEEE 802.15.4 standard.

It is designed for resource-constrained devices by reducing the size of the address to 64 or 16 bits, depending on whether it is for a Local Network or a Personal Area Network, respectively; it uses default values for specifying the network.

Transport Layer

The transport layer, which is also one of the base protocols in the current Internet, is in charge of the end-to-end connection. This layer is composed by two protocols: (i) TCP, for persistent connections defining a reliable transmission channel, and (ii) UDP, for connection-less datagram delivery.

Transport Control Protocol (TCP) [93] provides reliable, ordered, and error-checked data delivery in a link based on an IP network by establishing, first at all, a transmission path, and, next, enforcing retransmission in the case of errors.

User Datagram Protocol (UDP) [92] allows sending messages (datagrams) on an IP-based link without requiring connection establishment, thereby avoiding retransmission mechanisms and reducing complexity. It is useful for applications with real-time constraints, such as most of the applications running on IoT devices.

Messaging Layer

The messaging layer defines protocols for data transmission systems considering IoT restrictions.

Representational state transfer (RESTful) [48] is a Web-based architecture to exchange or to manipulate Web resources through a textual representation

using preset stateless operations; it means that each message must have all the required information to complete the request. It follows a client/server model based on the Hyper Text Transfer Protocol (HTTP), and relies on its functions: (i) GET, to retrieve a resource; (ii) POST, to create a resource; (iii) PUT, to change the state of a resource; and (iv) DELETE, to delete a resource. The data representation typically adopts the eXtendible Markup Language (XML) or JavaScript Object Notation (JSON) formats.

Figure 3.5 presents a basic overview of a RESTful Architecture under the client/server model.

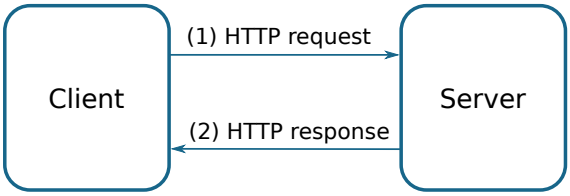


Figure 3.5: Basic RESTful architecture.

There is a specification [104] for constrained nodes and networks called Constrained RESTful Environments (CoRE) Link Format. It specifies a set of links to discover resources, and to access these resources in a M2M environment.

Message Queue Telemetry Transport (MQTT) [80] is a lightweight messaging protocol based on the publisher/subscriber scheme that runs on top of the TCP/IP protocol. It is also designed for constrained networks with limited bandwidth.

MQTT is composed by three elements: publishers, subscribers, and a message broker. A subscriber wanting to receive a message related to a specific topic, must subscribe to the message broker; next, when a publisher sends/publishes a message related to a certain topic, it is transmitted to all subscribers subscribed to this topic.

MQTT has three transmission Quality of Service (QoS) levels: (i) QoS 0: At most once. The message is sent once, but it does not check for ACKs to confirm message reception. (ii) QoS 1: At least once. The message could be sent more than once to each subscriber. (iii) QoS 2: Exactly once. The message is sent exactly once using four-way handshaking.

MQTT Sensor Network (MQTT-SN) [65] has been designed to be similar to MQTT, but considering the restrictions of wireless communication environments, such as limited bandwidth, short message length, etc., running over UDP or on Non-IP environments. For interoperating with standard MQTT environments, it needs an MQTT-SN Gateway which connects MQTT-SN nodes, such as constrained sensors, to the MQTT network. Figure 3.6 shows a basic MQTT architecture.

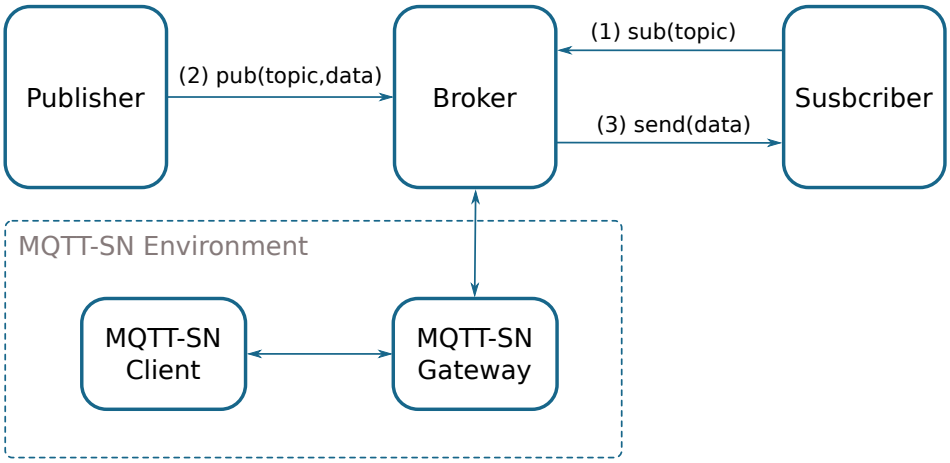


Figure 3.6: Basic MQTT(Message Queue Telemetry Transport) architecture.

Constrained Application Protocol (CoAP) [105] is a generic web protocol designed for constrained environments with restricted network capacities and restricted devices, allowing these devices to communicate with the Internet or other constrained devices. It implements a compressed subset of the RESTful model implementing GET, POST, PUT, and DELETE operations over UDP.

CoAP reduces the message header and restricts message exchange, reducing the network overhead. It is very useful for Machine to Machine (M2M) communication, and for the Internet of Things (IoT).

CoAP can be easily translated to HTTP for seamless integration with existing Web systems, while reducing network requirements.

eXtensible Messaging and Presence Protocol (XMPP) [99] is a message-oriented communications protocol based on XML. Initially, it was called Jabber, and it was designed for Instant Messaging (IM). It allows federations among various XMPP servers, and even communication with different technologies using XMPP gateways. Currently, it is also used for Voice over IP (VoIP), video, gaming, or even for IoT applications. Figure 3.7 shows the basic architecture of an XMPP-based system.

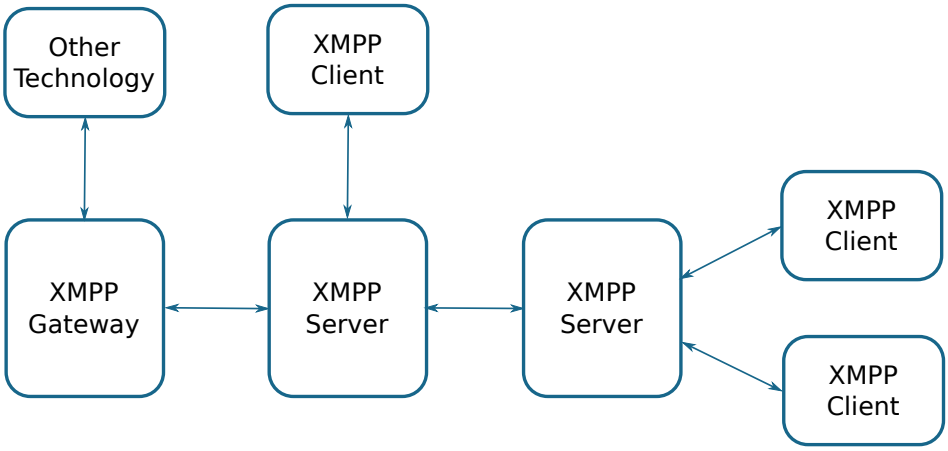


Figure 3.7: Basic XMPP (eXtensible Messaging and Presence Protocol) architecture.

The specification for IoT is XEP-0323: Internet of Things—Sensor Data [115], which provides the architecture, basic operations, and data structures for sensor data communication, including a hardware abstraction model for the interconnection of constrained devices.

Simple Measuring and Actuation Profile (sMAP) [39] is an example of how RESTful web services can be simplified, while still allowing instruments and other producers of physical information to directly publish their data in a central server.

Message Format Layer

The Message Format layer presents all data encoding types to store and transmit structured data for IoT applications.

eXtendible Markup Language (XML) [27] is a markup language for encoding documents in a text format that is understandable by both human and machines. XML is designed to store data units called entities, where all data structures and document descriptions are achieved through markups. Using these markups, it is possible to create any logical data structure in an easy way.

JavaScript Object Notation (JSON) [26] is a format notation to encode structured data (attribute-value pair or array) using human-readable text. It was designed to replace XML by reducing its complexity. It is very common in web systems, especially in AJAX-style ones. It uses pairs (object_id:object_value) and brackets to provide complex object structuring for fitting data in a text document.

Efficient XML Interchange (EXI) [100] is a binary and compact representation of XML or JSON documents. It aims at resource-constrained devices

and networks, attempting to reduce the size of the data and the computational requirements when compared to other compressors such as gzip. The EXI coder is based on events, and it follows a simplified Huffman coding to create a binary document.

MessagePack [50] is a binary serialization format that encodes messages faster, and in a more compact manner, than traditional methods such as JSON or XML. This is possible since small integer values are encoded in a single byte, while strings require only an extra byte to identify them. This characteristics simplifies the encoding process, but it has some limitations, such as the size of strings or numbers, the number of the key/value association map, etc.

Table 3.1 shows a representation of sensor data using XML and JSON encoding, and Table 3.2 shows a representation of sensor data using EXI and MessagePack encoding, respectively. We can observe that the EXI and the MessagePack encoding are much smaller (163 and 186 bytes) than the encoding achieved using XML (474 bytes) or JSON (357 bytes).

Table 3.1: Example of data representation using typical encoding types.

XML Encoding (474 bytes)

```
<?xml version="1.0" encoding="UTF-8" ?>
<trace>
  <id>trace1</id>
  <values>
    <captures>
      <latitude>39.470577</latitude>
      <longitude>-0.3336604</longitude>
      <ozone>56</ozone>
      </captures>
      <captures>
        <latitude>39.470652</latitude>
        <longitude>-0.3343365</longitude>
        <ozone>68</ozone>
      </captures>
      <captures>
        <latitude>39.470892</latitude>
        <longitude>-0.3359987</longitude>
        <ozone>59</ozone>
      </captures>
  </values>
</trace>
```

JSON Encoding (357 bytes)

```
{
  "trace": {
    "id": "trace1",
    "values": {
      "captures": [
        {
          "latitude": "39.470577",
          "longitude": "-0.3336604",
          "ozone": "56"
        },
        {
          "latitude": "39.470652",
          "longitude": "-0.3343365",
          "ozone": "68"
        },
        {
          "latitude": "39.470892",
          "longitude": "-0.3359987",
          "ozone": "59"
        }
      ]
    }
  }
}
```

Table 3.2: Example of data representation using compressed (binary) encoding types.

EXI Encoding (163 bytes)	**MessagePack Encoding (186 bytes)**
80 40 67 47 26 16 36 5a 80 24 06 d2 c9 50 08 84 3a 39 30 b1 b2 98 80 ee cc 2d 8e ac ae 75 00 48 25 8d 85 c1 d1 d5 c9 95 ce a0 00 00 96 c6 17 46 97 47 56 46 5a 80 44 2c cc e4 b8 d0 dc c0 d4 dc dc 0a 6c 6f 6e 67 69 74 75 64 65 a8 04 43 0b 4c 0b 8c cc cc cd 8d 8c 0d 00 66 f7 a6 f6 e6 5a 80 44 10 d4 d9 00 0c 02 cc ce 4b 8d 0d cc 0d 8d 4c 80 0c 83 0b 4c 0b 8c cc cd 0c cc cd 8d 40 0d 01 0d 8e 10 00 c0 2c cc e4 b8 d0 dc c0 e0 e4 c8 00 c8 30 b4 c0 b8 cc cc d4 e4 e4 e0 dc 00 d0 10 d4 e5 ea 80	81 a5 74 72 61 63 65 82 a2 69 64 a6 74 72 61 63 65 31 a6 76 61 6c 75 65 73 81 a8 63 61 70 74 75 72 65 73 93 83 a8 6c 61 74 69 74 75 64 65 a9 33 39 2e 34 37 30 35 37 a9 6c 6f 6e 67 69 74 75 64 65 aa 2d 30 2e 33 33 33 36 36 30 34 a5 6f 7a 6f 6e 65 a2 35 36 83 a8 6c 61 74 69 74 75 64 65 a9 33 39 2e 34 37 30 36 35 32 a9 6c 6f 6e 67 69 74 75 64 65 aa 2d 30 2e 33 33 34 33 33 36 35 a5 6f 7a 6f 6e 65 a2 36 38 83 a8 6c 61 74 69 74 75 64 65 a9 33 39 2e 34 37 30 38 39 32 a9 6c 6f 6e 67 69 74 75 64 65 aa 2d 30 2e 33 33 35 39 39 38 37 a5 6f 7a 6f 6e 65 a2 35 39

Semantic Layer

The Semantic layer presents all approaches that describe a logical representation of things in the IoT context. Therefore, in sensing approaches, we can find several representative examples:

SensorML [23] is an standard model based on XML encoding for describing sensors and measurement processes. It is developed by the Open Geospatial Consortium, Inc. (OGC), describing a wide range of sensors for different types of architectures, including remote sensors, in-situ sensors and dynamic sensors, among others.

Semantic Sensor Net Ontology [36] describes sensors and observations, avoiding to describe domain concepts, location, time, etc. It is developed by the W3C Semantic Sensor Networks Incubator Group (SSN-XG).

Web of Things (WoT) [68] specifies a data model to describe physical devices connected to the Web (Internet) using JSON encoding. It was created for the Mozilla project, and it was formally submitted to the World Wide Web Consortium (W3C) for discussion.

Notice that any sensor device must be able to cope with a subset of the previously described protocols to allow the exchange of data between these sensors and a central server.

3.1 Crowdsensing in Smart cities

Sensing processes are one of the most important tasks in a smart city because they allow retrieving the different parameters involved in different control pro-

cesses. Examples of such processes include transportation, energy management, air conditioning, etc. However, controlling air pollution in smart cities stands out as a key issue, as it has severe consequences on human health, thereby making environment sensing a critical task and a prominent service.

By embracing the Smart City paradigm, crowdsensing becomes a solution able to cope with air pollution monitoring since it assumes that a significant number of users perform collaborative sensing tasks, thereby collecting data from different populated locations while doing their daily activities. The collected data is periodically transmitted to a central server (Cloud) for data storage and processing. Overall, this strategy implies that the sensors used must be cheap and tiny enough for comfortable management and easy transportation. Otherwise, it becomes hard to achieve a widespread distribution and adoption. Besides, there must be a communications link for transmitting the acquired data to a cloud-based server, where data are constantly being stored and processed.

3.1.1 Mobile crowdsensing

Following the Smart City paradigm, and focusing on the data collection domain, the concept of crowdsourcing has been introduced to refer to scenarios where a large group of people, through different devices and technologies, actively participate in the data acquisition process [56]. Once data are collected, they are sent to a central server for analysis, and feedback will eventually be returned to citizens through actions and services aimed at improving their life quality.

Crowdsensing is a subtype of crowdsourcing where sensors are the actual sources of the data gathered [51]. If air quality sensors are used, crowdsensing becomes a good alternative to traditional stationary air quality stations whereby small sensors are distributed to a large group of people that seamlessly contribute to the system while doing their everyday tasks [81].

Before the mobile crowdsensing became a viable option due to the availability of small sensors, the sensing process was made through a Wireless Sensor Network (WSN) [121, 4], which is composed by a set of nodes or sensors that collect data and send this data towards a central sink or gateway. The latter carries out processing tasks, or merely forwards collected data to a server for storage and processing. Usually, all sensors are resource-constrained devices, and the central sink or gateway has fewer restrictions, often being connected to the power network. Currently, most of the Wireless Sensor Networks are based on the IoT architecture since it allows us to work with constrained devices and restricted networks.

Figure 3.8 shows the basic structure of a Wireless Sensor Network (WSN). We can see that the communications link between the sensors and the sink/gateway is wireless, typically relying on Zigbee, and that the communications link between the sink/gateway and the central server is typically a more robust link, either wired (e.g. Ethernet) or wireless (e.g. Wi-Fi or Cellular).

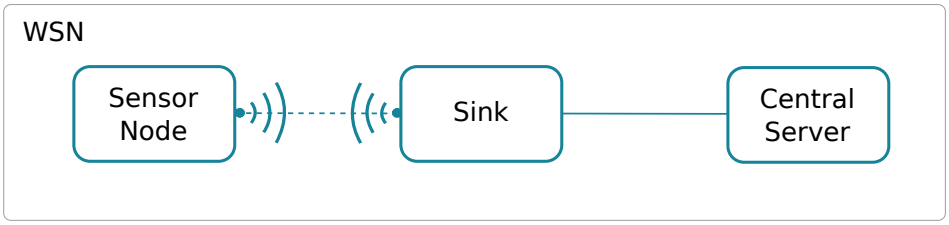
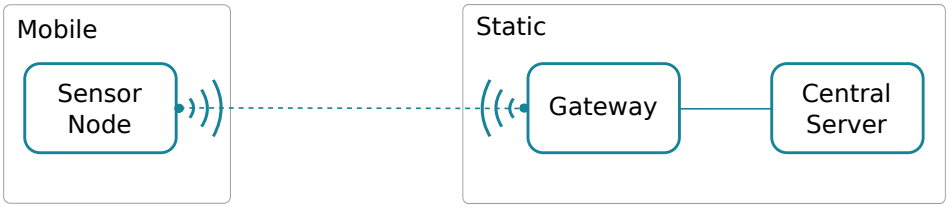
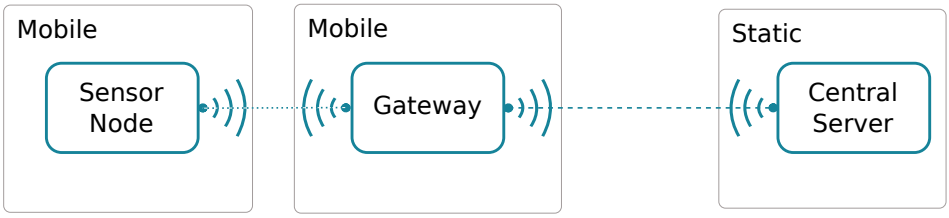


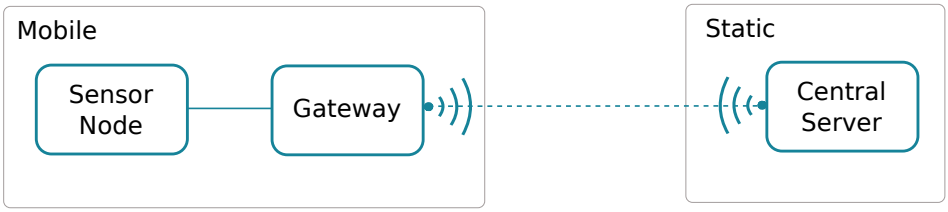
Figure 3.8: Wireless Sensor Network structure.



(a) Mobile Wireless Sensor Network with mobile sensors.



(b) Mobile Sensor Network with mobile sensors and mobile gateway.



(c) Crowdsensing architecture.

Figure 3.9: Different types of Mobile Sensor Network structures.

The support for mobility in a Wireless Sensor Network [88, 110] can be achieved through different strategies, including sensor mobility, as shown in Figure 3.9a, or by having mobility on both sensors and gateway, as shown in Figure 3.9b. Finally, we have crowdsensing architectures where the gateway and the sensor are the same, or are packed together. Commonly, the best way to implement crowdsensing is through smartphones, since nearly all people carry one with them nowadays, and they are endowed with several sensors and communication interfaces. Figure 3.10 shows an example of a crowdsensing architecture where a smartphone is used as the gateway between the sensor and the Central Server.

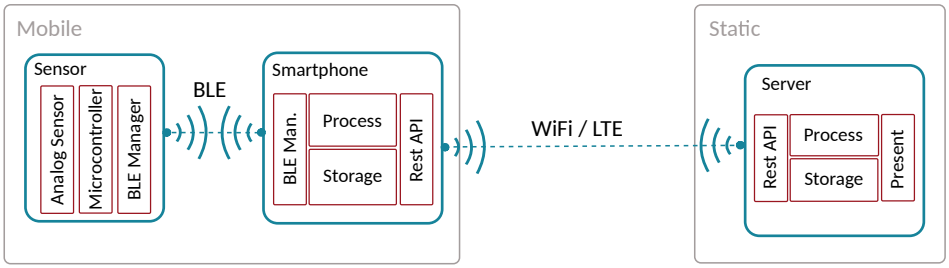


Figure 3.10: Crowdsensing architecture overview.

Crowdsensing solutions need to be widely disseminated and adopted by users to be successful. In addition, to achieve such widespread acceptance, the impact on users' everyday activities must be low. This means that any deployed application must operate in background, and they should avoid consuming excessive resources, while requiring only a minimal user intervention. Concerning the sensor itself, if external to the smartphone, it should be cheap, small, easy to use, and comfortable to carry.

Crowdsensing approaches have two basic architectural components [7]: a mobile component for the data acquisition process, and a central server for data storage and processing.

The mobile component must be able to collect environmental parameters, transmitting them towards the central server. The data acquisition process is based on smartphone sensors, or on small external sensors accessible via smartphone, and the transmission process usually relies on smartphone connectivity towards the Internet. Despite delegating transmission tasks on smartphones, external sensing devices must still be endowed with communication capabilities to transfer the collected data to the smartphone. So, the sensing device should be equipped with a wireless communications interface, being technologies like Wi-Fi, Bluetooth, Radio Frequency ID (RFID), NFC, and ZigBee good candidate solutions.

The central processing server must be able to receive the transmitted data from the sensors, store and process the data, as well as properly offer the obtained results

to system managers. Also, in some cases, they perform remote communication with the mobile devices for configuration tasks, thereby allowing to dynamically change the sensing behaviour.

Taking the aforementioned considerations into account, Figure 3.10 shows a basic hardware architecture applicable to air quality sensing applications that should include: (i) a mobile sensor, (ii) a smartphone, and (iii) a central server; the proposed architecture resembles various approaches from different authors [61, 124, 34, 7]. Moreover, as shown in Figure 3.11, the crowdsensing process basically includes five different tasks: (i) sampling process, (ii) filtering process, (iii) data transfer, (iv) data processing, and (v) results presentation. Notice that all these tasks could be done by different hardware components; for instance, the filtering process could be done by the sensor, the smartphone, or even the central server, depending on the system characteristics. Moreover, characteristics associated to sensing, filtering, and transmission tasks could be defined based on parameters obtained from the processing step.

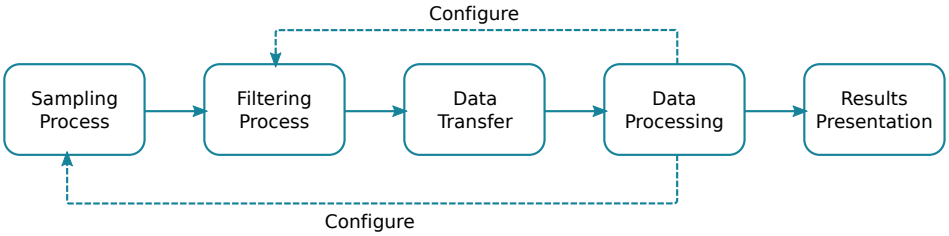
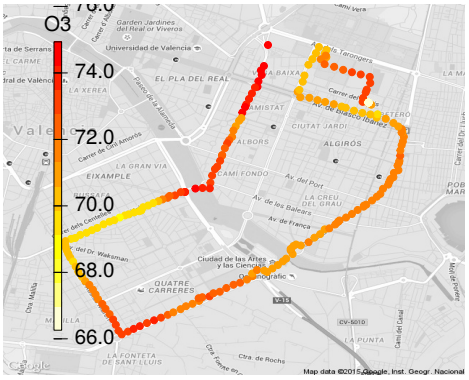


Figure 3.11: Crowdsensing steps.

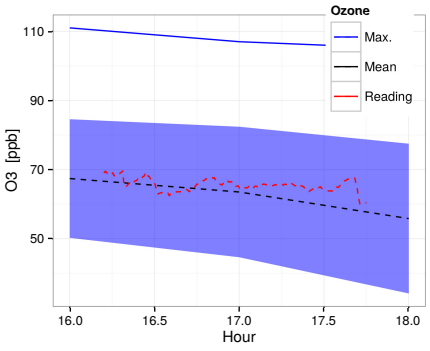
Sampling process: refers to the process of capturing pollutant measurements, including the calibration process, where electrical signals are translated to pollution units, filters, fault detection and diagnosis, etc.

Sensor calibration for Commercial Off-the-shelf (COTS) sensors, such as electrochemical ones, is a process that depends on the physical sensor characteristics, the temperature, etc. Basically, electrical outputs must be translated to pollution units, and often there is no lineal relation. The calibration is commonly made in advanced laboratories, taking into account samples taken with different pollution levels, and for different temperatures and humidity conditions [113, 69]. However, in urban scenarios, the auto calibration procedure is too complicated because all sensors are distributed among different users. Alvear et al. [6] proposed a method to calibrate off-the-shell sensors using mathematical regressions based on high-accuracy samples obtained through fixed stations deployed in a city. Once a mobile sensor is near to these stations, these samples are used to adjust the translation equation (electrical signal to pollution values).

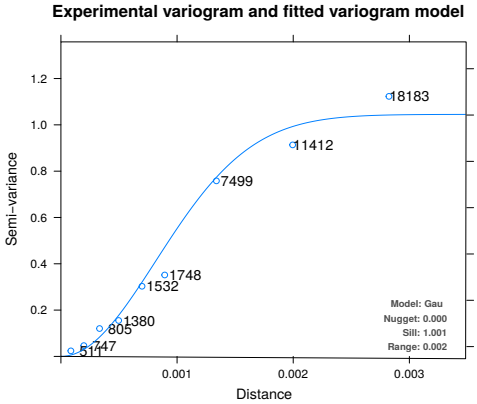
Using COTS sensors, the sampling error and its diagnostics can be a problem [6]. Nevertheless, when focusing on a Crowdsensing solution using a large number



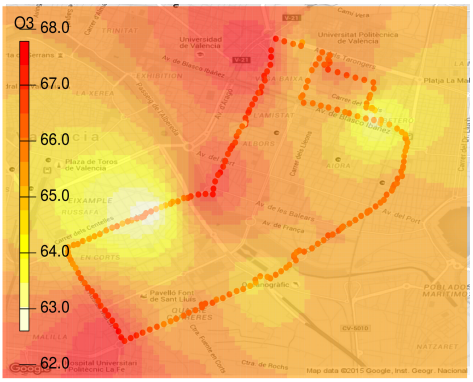
(a) Sampling process



(b) Filtering process



(c) Data processing



(d) Results presentation

Figure 3.12: Data handling process as described in [7]: (a) sampling process, (b) filtering process after adjusting the temporal variations, (c) data analysis using a semivariogram of the captured data used for interpolating the entire area using the kriging technique, and (d) pollution distribution map.

of mobile sensors (smart city scenario), this problem could be solved by accounting for redundant data and statistical analysis (i.e.: Kriging method allows us to deal with sampling error).

Filtering process: refers to deleting redundant and/or wrong measurements caused by the sensors reading oscillations. By using mobile sensors, the filtering process also has to deal with temporal variations, as described in [6, 7], adjusting samples to a same temporal fragment.

Data transfer process: refers to the upload of data from the sensor to the cloud (Central servers), including sensor-smartphone and smartphone-server

communications. It is achieved through the IoT protocols.

Data processing: refers to the interpolation technique used to recreate a pollution distribution map. It can rely on different methods (Kriging, IDW, Nearest neighbour Spatial Averaging) as described in [119]. Currently, the most used method is the kriging interpolation technique, where a semivariogram is calculated to create a complete pollution map.

Results presentation: refers to presenting the obtained results to the system administrator in an understandable way. The most useful representation is a graphical map for the target region.

Figure 3.12 presents the data handling process, as the authors described in [7], showing the four processes for handling pollution information in order to recreate a complete pollution map for a certain target area.

3.2 UAV-base monitoring in rural areas

Regarding crowdsensing approaches, projects like [29, 63, 34] relied on crowdsensing solutions to monitor pollution in urban areas. However, in rural and industrial zones, available options are quite more limited. In the particular case of large rural or industrial areas, a fleet of mobile vehicles could be efficiently used to cover the vast distances associated with them. Furthermore, the use of autonomous sensor carriers is even more encouraged in this case due to the following considerations:

- The relative absence of civilian population to be taken care of during robotic operations.
- Stable and regulated positioning of obstacles.
- Fewer constraints concerning UAV flight laws.
- Safety and security concerns, as some areas could be dangerous to access for human operators.

Since, in these environments, ground access is usually hindered and full of obstacles, the most feasible way to implement a fleet of mobile pollution-monitoring robots is via Unmanned Aerial Vehicles (UAVs) [43].

Nowadays we can find several types of UAVs having variable size, flight time, load capacity, and price. Figure 3.13 shows an example of the configuration of different UAVs.

Figure 3.13 presents a classification of UAVs depending on their size, payload capacity and endurance [58, 19, 38]:

- **Micro/Mini Aerial Vehicles (MAV)** are the smallest, cheapest and most restricted devices. They were created mainly for entertainment purposes, and have been recently adopted for different research fields. They fly at a

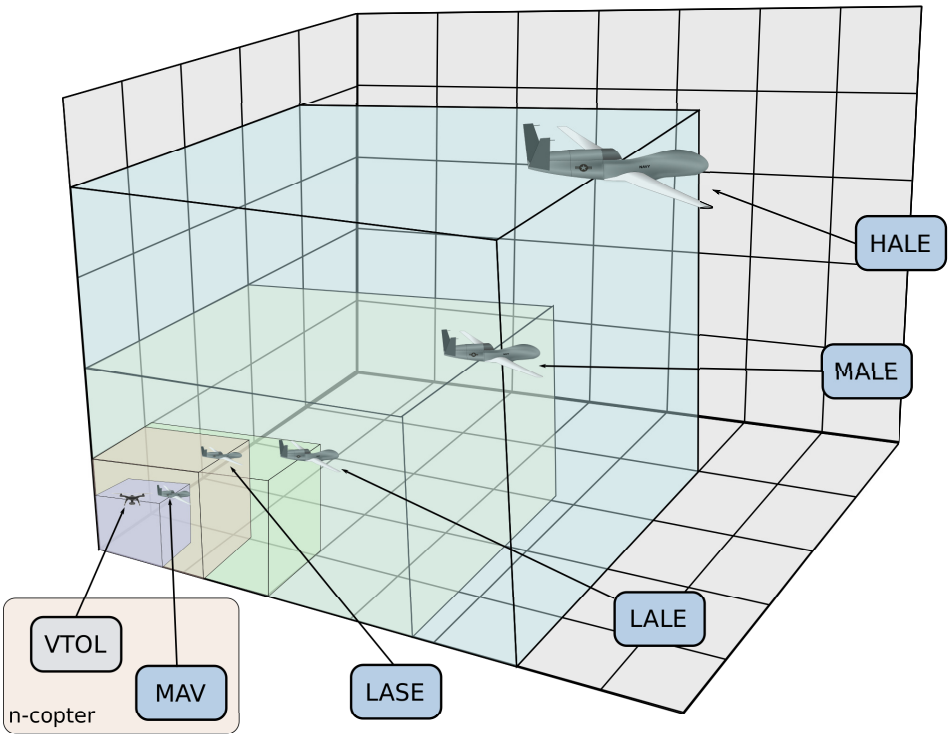


Figure 3.13: Types of UAV [118].

maximum altitude of 100/200 meters due to government restrictions, and flight time is of less than 1 hour due to battery constrains. Their price is typically less than 1000 euros.

- **Vertical Take-Off and Landing (VTOL)** devices require no takeoff or landing strip, and are therefore typically chosen in situations where terrain limitations require this specialized capability.

- **Low Altitude, Short Endurance (LASE)** are small aircraft designed for some specific activity (civil or military) with a limited payload (20 Kg), a range of a few kilometers, and an endurance of less than 2 operating hours; it is relatively cheap, with a typical cost of a few thousand euros.

- **Low Altitude, Large Endurance (LALE)** are similar to LASE systems, but with higher endurance and higher prices. They are used for Forest inventory, monitoring, etc.

- **Medium Altitude, Large Endurance (MALE)** are high performance UAVs, but also have a small size. They are used mainly in military operations and a few civil operations that require carrying less than 600 Kg. They could cost hundreds of thousands euros.

- **High Altitude, Large Endurance (HALE)** are the most expensive and offer the highest performance. They are used almost exclusive in military operations and in some NASA projects, being able to reach up to 30.000m of altitude, having an endurance higher than 24 hours, a range of 20.000 Km, and a payload higher than 600 Kg. They are very expensive (millions of euros).

Table 3.3 presents a summary of the UAV classification characteristics.

Table 3.3: UAV Classification.

Type	Payload	Altitude	Endurance	Range	Price
MAV	<2Kg	50m	<1h	<5Km	very low
LASE	<20Kg	200m	<2h	25Km	low
LALE	<150Kg	500m	<24h	50Km	medium
MALE	<600Kg	6000m	Days	Unlimited	high
HALE	>600Kg	10000m	Days	Unlimited	very high

UAV-based solutions have experienced a very substantial increase in the last decade, especially in the past five years. Back in 2004, NASA experts defined a wide set of civil applications for UAVs [37], highlighting their potential in areas such as commercial, Earth Sciences, national security, and land management. This preliminary report was ratified years later by authors such as Hugenholtz et al. [64], who explained how the use of UAVs could revolutionize research methods in the fields of Earth Sciences and remote sensing. In [90], authors display the results of a detailed study on different UAVs aspects, showing their applicability in Agriculture and Forestry, Disaster Monitoring, Localization and Rescue, Surveillance, Environmental Monitoring, Vegetation Monitoring, Photogrammetry, and so on.

If we focus specifically on research using quadrotor multicopters, authors like Gupte et al. [59], and Colomina and Molina [35] consider that, given their high maneuverability, compactness, and ease of use, different applications for these devices are being found in areas including Civil Engineering, Search and Rescue, Emergency Response, National Security, Military Surveillance, Border Patrol and Surveillance, as well as in other areas such as Earth Sciences, where they can be used to study climate change, glacier dynamics, volcanic activity, or for atmospheric sampling, among others.

In our case, we are more interested in atmospheric sampling to measure air pollution levels. In this research area, Anderson and Gaston [9] highlight the

applicability of UAVs in the field of ecology, emphasizing that the spatial and temporal resolutions of the data obtained by traditional methods often fail to adapt well to the requirements of local ecology-oriented research. Furthermore, the use of UAVs, when flying at low altitudes and speeds, offers new opportunities in terms of ecological phenomena measurements, enabling the delivery of data with a finer spatial resolution. In fact, Zhang and Kovacs [123] explain how the images taken by small UAVs are becoming an alternative to high-resolution satellite images, which are much more expensive, to study the variations in crop and soil conditions. Specifically, the use of UAVs is considered a good alternative given its low cost of operation in environmental monitoring, its high spatial and temporal resolution, and its high flexibility in the scheduling of image acquisitions. A good example of this use can be found in the work by Bellvert et al. [17], which shows how, by using a multicopter equipped with a thermal camera, it was possible to obtain a very precise map of water levels in a vineyard, thereby achieving significant advances in the field of precision agriculture.

3.3 Summary

In this chapter, we analyze the different available technologies for create a mobile sensing solution considering urban and rural monitoring in Smart Cities. In urban scenarios, a good option is a crowdsensing approach using small sensors installed in some vehicle and smartphones to monitor a certain city region. Moreover, in rural scenarios due to limited number of user or vehicles, a good approach is using UAVs.

Respecting to Crowdsensing monitoring, we make an overview of the Internet of Things protocols, which allow to deploy a Smart city solution. Next, we analyze the crowdsensing approach using mobile sensors and how to integrate smartphones to the solution, considering the different steps from capture until process and present data.

Finally, we make a brief overview of the different types of UAV and their use in different fields, next focusing in air pollution monitoring.

Part II

Crowdsensing in Smartcities

Chapter 4

EcoSensor Platform

Taking the crowdsensing approach as reference, in this chapter we propose an architecture offering mobile pollution sensing with high spatial resolution that uses a smartphone as the gateway between the mobile sensor and the central server. Our architecture includes three independent 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 using the collected data through spatial interpolation techniques. Throughout our analysis, we will focus on ozone sensing since it is more complex to estimate that other pollutants.

4.1 Architecture

Our proposed architecture defines a set of elements that allow monitoring air pollution in a cheap and easy, but effective way, being specially useful in very crowded cities. It combines information from existing air quality monitoring stations with the data collected by mobile sensors to generate fine-grained reports about spatial pollution distribution throughout a city. These mobile sensors can be installed in bicycles or the public transportation system to monitor the whole city in a simple and effective way. All collected information is stored on a central server for data processing, generating detailed reports afterward.

The architecture integrates several hardware and software components. These components are either mobile sensing elements, or the central processing server that analyzes collected data, and presents detailed information. Mobile sensing elements are composed by two different components: (i) a mobile sensor for

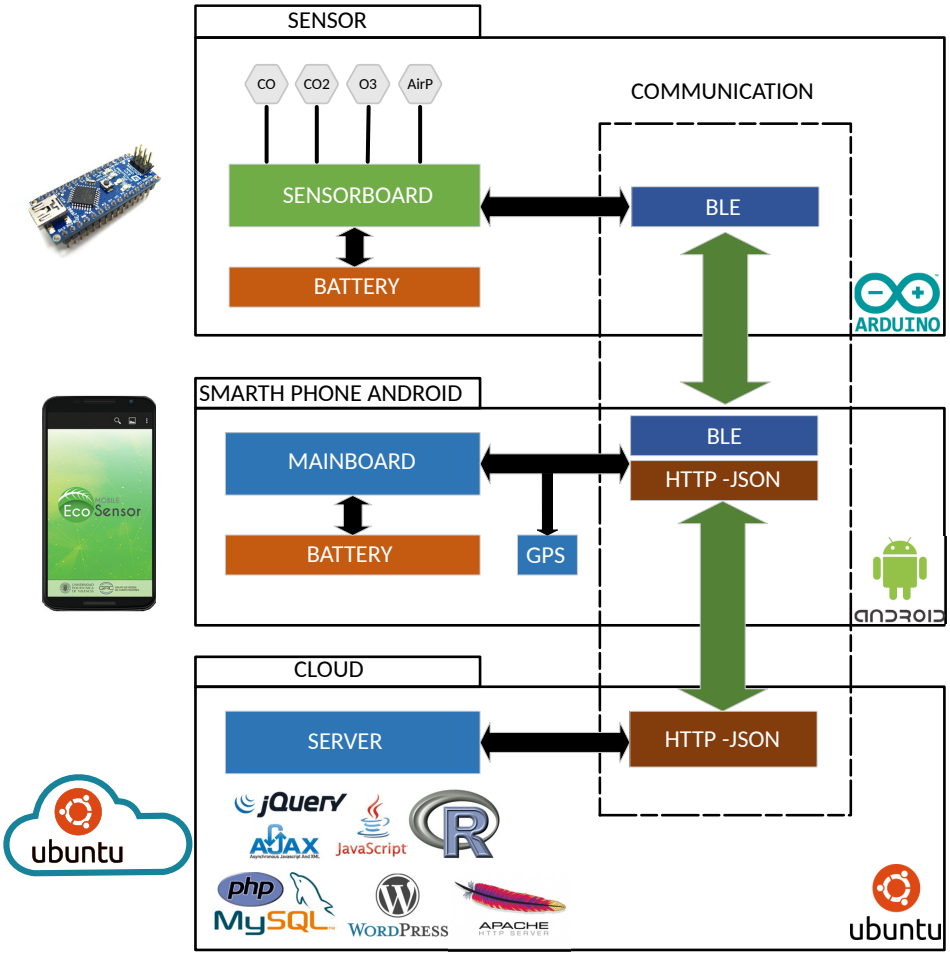


Figure 4.1: Overview of the proposed mobile sensing architecture including the main hardware components and the technologies used.

measuring pollution data, and (ii) 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. Figure 4.1 provides an overview of the proposed architecture.

The mobile sensor could be based on an Arduino, a Raspberry Pi, or similar platforms, as discussed in chapter 5. It measures environment parameters through various sensors (Ozone (O_3), Carbon Dioxide (CO_2), Air Pollution, or temperature). Once data is ready, it can be made available to the Android device via a

Bluetooth Low Energy connection.

4.2 Central processing server

The central server stores and processes all the information captured by mobile sensors. It is a web-enabled system that handles the information received from the Android device. In general terms, the central server is responsible for the following tasks:

- Receive and decode data sent by the mobile sensor.
- Store data in a database for future processing and presenting tasks.
- Process the information using different statistical procedures.
- Present detailed information to the system administrator through a web front-end.

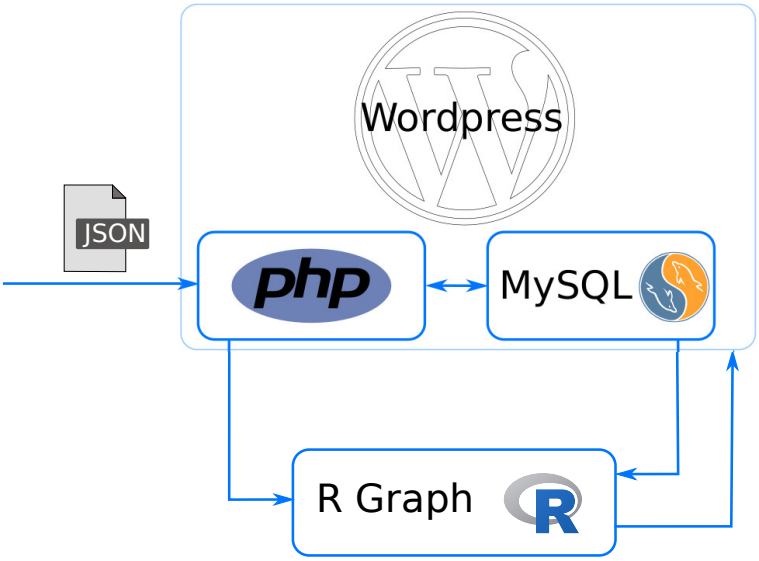


Figure 4.2: Overview of the cloud system architecture.

The cloud server was built in a PHP server using a Word-press CRM as the web interface, and MySQL as the database to store data related to the Word-press CRM and to pollution readings. In addition, the R Graph software is used both as a data processing tool, to interpolate data using kriging, and also a graphic

generator. Figure 4.2 shows the software architecture of the proposed solution, and the different elements integration.

To process the data, the following steps take place:

- First, the data is uploaded to the server in JSON format through an HTTP POST command issued by the smartphone application, and using the following format:

```
{
  "trace": {
    "device": "node1",
    "id": "trace1",
    "date": "2016/07/23",
    "values": {
      "captures": [
        {
          "latitude": "39.470577",
          "longitude": "-0.3336604",
          "time": "13:45:34",
          "ozone": "56"
          "co2": "36"
          "temperature": "32"
        },
        {
          "latitude": "39.470652",
          "longitude": "-0.3343365",
          "time": "13:50:03",
          "ozone": "68"
          "co2": "31"
          "temperature": "32"
        },
        {
          "latitude": "39.470892",
          "longitude": "-0.3359987",
          "time": "13:44:48",
          "ozone": "59"
          "co2": "39"
          "temperature": "32"
        }
      ]
    }
  }
}
```

- In the server, the data is decoded and stored in a MySQL database, as shown Figure 4.3. There are three sections: (i) the data from the sensor (Sensor Reading), (ii) the data from fixed stations (Infrastructure Data), and, (iii) the data generated, namely, the data filtered and calibrated (Results).

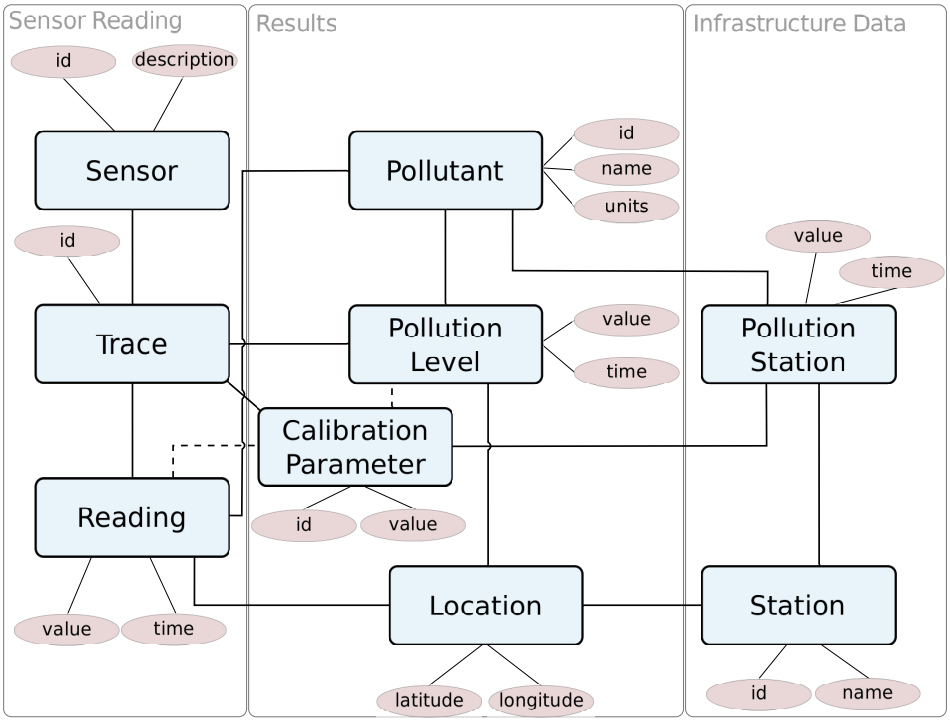


Figure 4.3: Data structure of Ecosensor platform.

- Once the data is in the database, kriging interpolation can be done using the R Graph tool through a PHP system call.

```
1 <?php
2 exec('Generate_trace.R $trace');
3 ?>
```

- For processing the data, the R Graph tool first retrieves the data from the database, and then interpolates the pollution in the target area using kriging techniques, available as part of the "automap" package, so generating different graphs afterwards to be shown to the user via the web interface.

For each trace, the system can generate different graphics such as heatmaps, boxplots, time series, and the confidence associated to the spatial interpolation process, as shown in Figure 4.4.

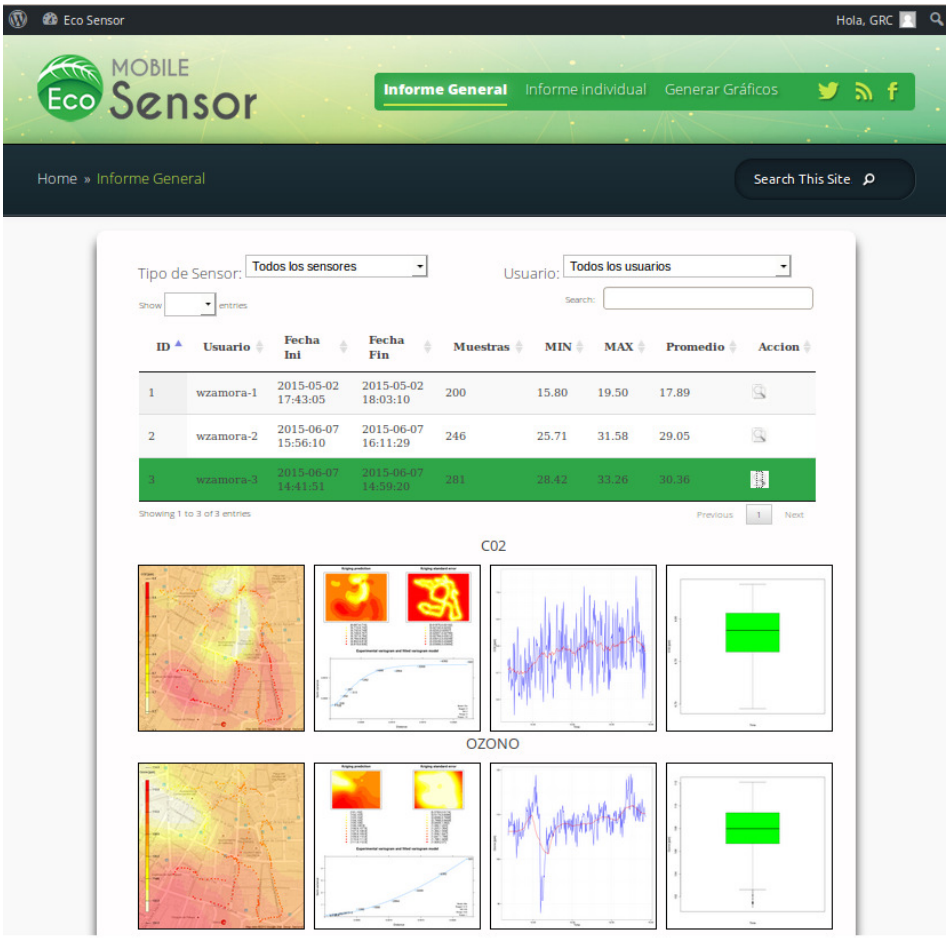


Figure 4.4: Example of the cloud application web page showing some monitoring sessions, and the output analysis for two air pollutants.

The website, available at *http://www.ecosensor.net*, has two access types: (i) administrators, who have full access to the information in terms of trace handling, processing and visualization. Once logged in, the administrator views all uploaded traces, being able to choose different statistical analyses for the different datasets (e.g. CO2, Ozone, Air Pollution and temperature). And (ii) normal user, who

only can access to previously generated information.

4.3 Android-based Application

Concerning smartphones, they are devices widely used nowadays for nearly all tasks. They are characterized by powerful computing capabilities, large amounts of memory, and several embedded sensors and communication interfaces [75]. We consider smartphones as the best gateway option for connecting mobile sensors with a central server. In addition, they can perform CPU-intensive tasks such as data filtering or data fusion, simplifying sensor requirements and design to mere data acquisition and data relaying towards the smartphone.

Since the smartphone must act as a gateway between sensors and the cloud server, it must manage at least two network interfaces: one to collect data from the sensors (Sensing middleware), and another one to upload data to a central server (Cloud middleware). Although both tasks must run independently, the data uploading process is often not made in real time, contrarily to the sensor data collection process, which is a task that should be done periodically, especially if we aim at a simplified sensor design, as shown in Figure 4.5. Moreover, modules to process and store the collected data are also needed.

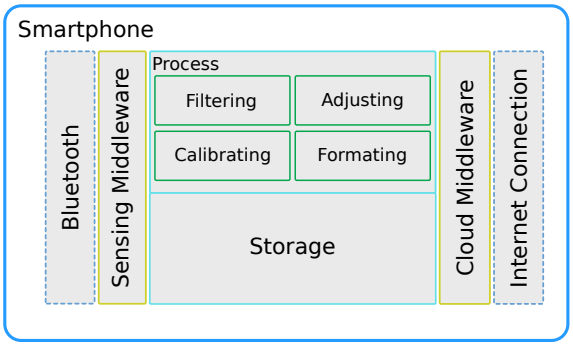


Figure 4.5: Smartphone software architecture overview.

The smartphone application was built for Android systems (see Figure 4.6). This application allows starting or stopping a trace, view captured data in real-time, upload data to the server, and perform other management tasks.

Internally, the application has two parts: (i) a service that continually receives the data sent by the sensor, and that saves it in an internal database. The service opens a Bluetooth serial communications channel with the sensor for the data transfer. And (ii) a user interface that allows starting or stopping a trace data capture from the sensor, and that also provides real-time feedback about pollution levels at the current location according to the Air Quality Index (AQI) [2].

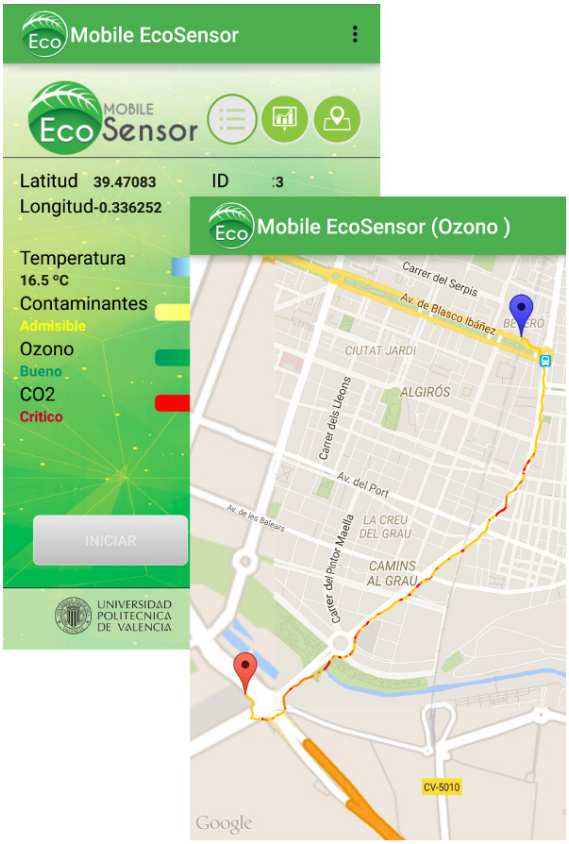


Figure 4.6: Android-based application developed: monitoring screen (back), and path followed during a monitoring session (front).

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 connection.

4.4 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 complex to measure accurately than other pollutants due to its dependency on temperature and time of day.

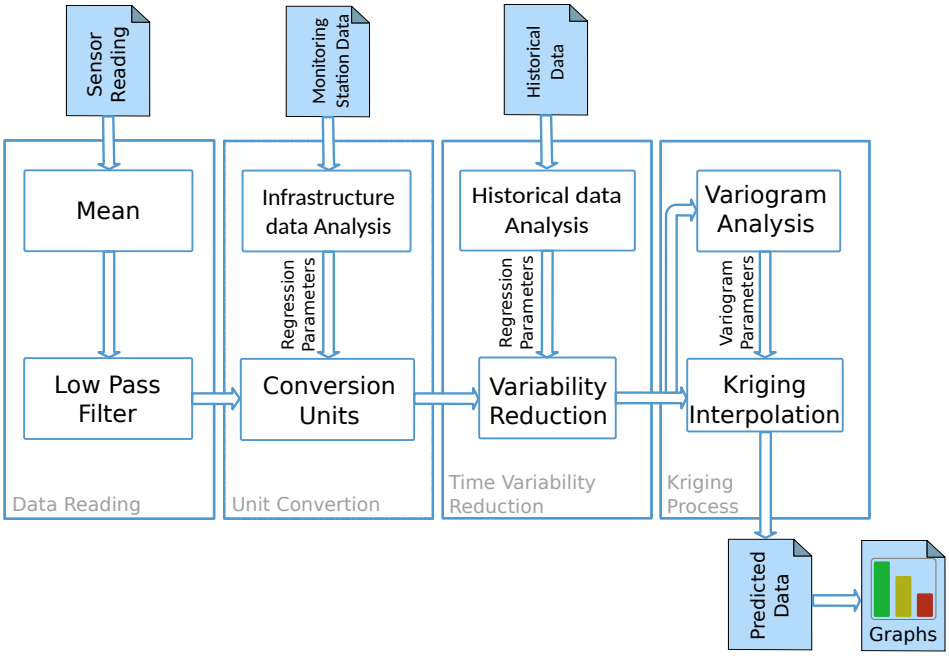


Figure 4.7: Monitoring process overview showing the different tasks associated to each step in the process.

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.
- Sensor outputs should be transformed into the respective units for each pollutant. In most cases, the measured resistance value must be converted into particle 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 4.7 shows the different steps taken when transforming the raw sensor readings into detailed air pollution maps. Also, next we detail how each of these issues has been addressed.

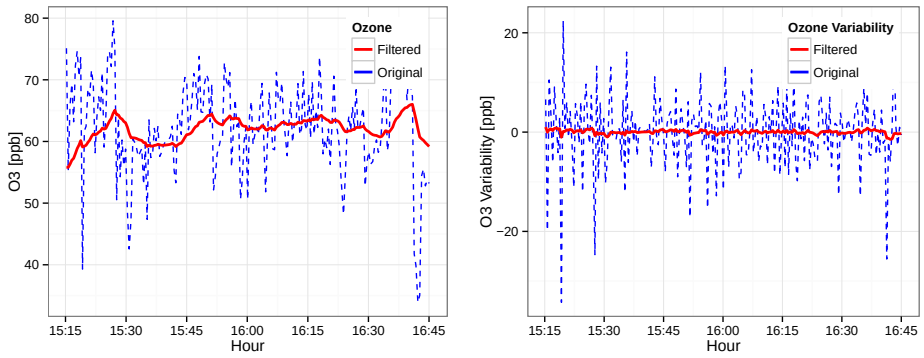


Figure 4.8: Relationship between captured data and filtered data (left), and relationship between captured data variation and the filtered data variation (right).

4.4.1 Data reading

Low-end sensors introduce significant variability between consecutive measurements (absolute values for inter-sample differences have $\bar{x}$=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 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 filtering coefficient. In this step, we drastically reduce the absolute variability ($\bar{x}$=0.32, σ=0.30).

Figure 4.8 (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 4.8 (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.

4.4.2 Unit conversion

Sensors provide an electrical signal output. It needs to be transformed to a pollution level value. Specifically, the *Ozone sensor probe (MiC-2610)* has an internal resistance, which varies proportionally to ozone concentrations. The sensor can measure ozone variations between 10*ppb* and 1000*ppb*, being that 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 Technical University of Valencia (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 $[°C]$	Station Ozone $[ppb]$	Calculated 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 the temperature coefficient, β_2 is the 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 1^{st} order regression ($R^2 = 0.83$), the obtained result is better in terms of R^2. Compared against a 3^{th} order regression ($R^2 = 0.86$), the improvement in R^2 is minimum, and so differences are reduced.

4.4.3 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 time $\triangle O_t$ and spatial $\triangle O_e$ dependencies. Since our main goal is to determine differences between ozone levels in a particular area, it is necessary to eliminate the time variation when attempting to calculate the pollution value in the same instant time.

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

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

Concerning the pollution variation throughout time, specifically the Ozone (O_3), we must know the typical behaviour during the day of this pollutant considering the known given environment conditions (instant time, temperature, season, etc.). For these purposes, we analyzed the historical data from the monitoring station located at the Technical University of Valencia in the period between 2008 to 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 4.9 shows the mean values and standard deviation in the shaded area, and the line on top represents the maximum values. The variation in ozone levels during a representative June day was also analyzed. As shown in Figure 4.9(left), 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 a different behavior in summer (specifically, from April to September) compared to the rest of the year. During day time, 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,

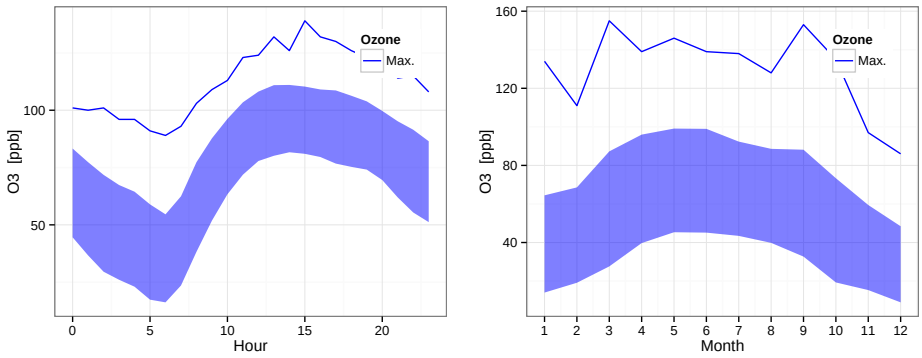


Figure 4.9: Ozone evolution in June (left) and throughout the year (right).

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.

Having a idea of how ozone levels typically evolve throughout the day in the city of Valencia, we proceed to reduce the time-related ozone variation corresponding to the time evolution observed. We could adjust the referential time to any time instant in the trace. Specifically, for our analysis, we consider the starting point as our reference.

The procedure followed to correct time-dependent variability was the following: (i) ozone values are calculated at two time instants (Starting time and Sampling time) using equation 4.7; (ii) next, the difference between the calculated values (Sampling time value minus the Starting time value) is calculated obtaining the typical ozone variation in these environmental conditions; (iii) using the previously

calculated variation, the actual readings are corrected (sampled value minus the typical variation).

4.4.4 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 4.10.

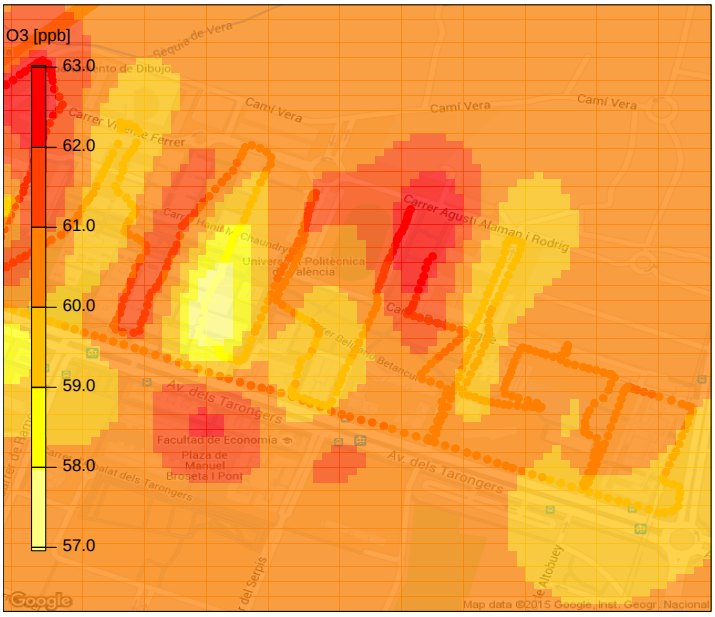


Figure 4.10: Example of an ozone distribution heatmap for the UPV using the proposed architecture.

The semivariogram defines the variance of the differences between different reference points. It determines the parameters required for the kriging interpolation, which have an influence on the final value distribution. These parameters are:

- Sill: determines the total variance of the values, namely the range of captured pollution values.

- Nugget: determines the variance at the origin, namely the oscillation of the readings. We could specify this value by knowing the usual reading oscillations of the off-the-shelf sensor used to avoid interpolation error.
- Range: determines the range of influence of the model, namely the influence of distance towards a particular sampled value.
- Model: determines the distribution function. It can be Gaussian, Spherical, Exponential, Circular, or Linear.

Figure 4.11 shows a sample semivariogram as an example.

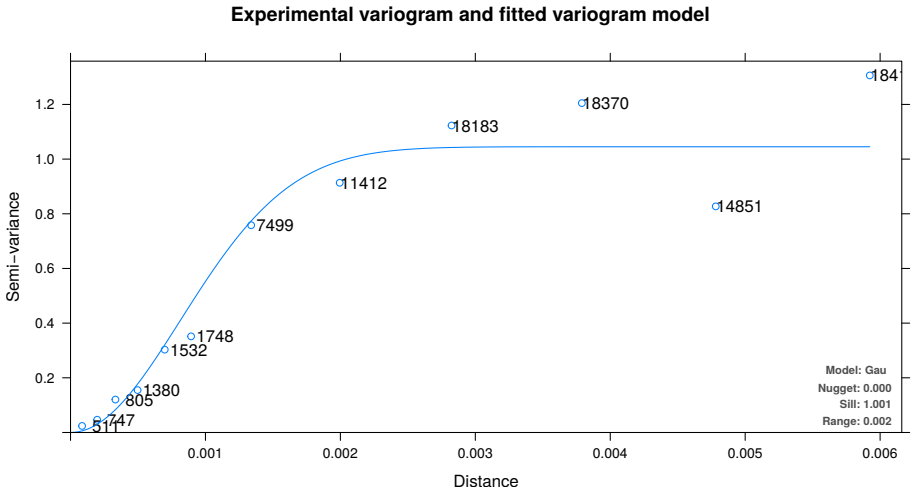


Figure 4.11: Example of a semivariogram showing a Gaussian distribution. It shows the different parameters related with the interpolation techniques (Nugget, Sill, and Range).

4.5 Summary

In this section we proposed a complete architecture for environmental monitoring that combines low-end sensors, smartphones, and cloud services to measure pollution levels with 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 was 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.

Chapter 5

Mobile Sensor Design

In the scope of Air Pollution monitoring, Mobile sensing solutions require the use of sensors able to capture the pollution level, as well as the location an the time associated to the different measurements. Using the captured data, different processes then take place to filter and process the information retrieved.

In our architecture, called EcoSensor, we define two main components: a central server to store and process the data, and a mobile node that carries out the capture process; the latter is based mainly on a smartphone as the processing element, thereby assuming an edge computing function.

5.1 Mobile sensing requirements

Despite relying on smartphones for providing system intelligence, basic mobile sensor requirements still involve:

Processing: The sensor must be able to process the measured data, perform basic filtering tasks, and transfer data to an external device. Anyway, in terms of processing power, requirements are low.

Storage: By assuming that a links towards a smartphone or a similar device is available, the sensor does not need to actually store large amounts of collected data. In fact, since data can be seamlessly relayed to the smartphone in real time, the sensor can limit its internal storage to only a few samples.

Communication: Sensors do not need to have a direct connection to the cloud server via Internet, but they still need to transfer the collected data to the smartphone. Thus, sensors require a communications link compatible with current smartphone technologies like Wi-Fi, Bluetooth, or NFC.

Autonomy: Sensors must be able to operate for long periods using a small power supply. Thus, energy optimization becomes a key requirement to take advantage of small batteries.

Size: Sensors need to be transported by users, or to be quickly installed in vehicles (e.g., bicycle, motorcycles, and cars). Thus, they must be small enough for the sake of aesthetics and comfort.

Price: To be attractive to users, and to provide scalability, sensors must be as cheap as possible. Otherwise, it becomes difficult to meet the broad dissemination requirements of crowdsensing approaches.

To fulfill the technical requirements, a basic mobile sensor should be composed of a sensor device able to monitor the differences between different pollutant levels, a communications module for transferring the data collected, and a microcontroller/microcomputer acting as a central element for managing all tasks.

Figure 5.1 shows a basic mobile sensor design, and the main characteristics to consider. As shown, the sensor hardware module must be able to connect to a microcontroller/microcomputer, a connection that typically relies on an analog/digital port. Similarly, the communications module must also be connected to the microcontroller via an UART or USB port. Thus, the microcontroller becomes a central element in the sensing module, being responsible for managing the interactions between all the elements.

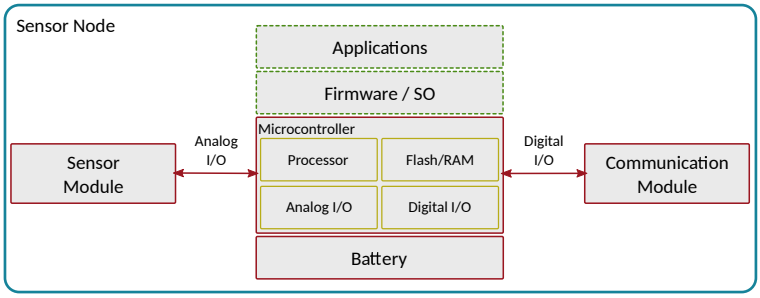


Figure 5.1: Mobile sensor design.

Overall, the sensing module must be equipped with different analog ports, UART/USB ports, a processor, and flash memory (ROM or RAM).

5.2 Overview of Available Hardware and Software

In recent years, the appearance of different embedded prototyping platforms, such as Raspberry Pi or Arduino, which are complemented by a large variety of compatible electronic components, paved the way for the creation of diverse applications related to IoT. When specifically focusing on environmental monitoring require-

ments, we find that there are different development options, including different types of sensor, and various communication interfaces.

Commercially, several companies offer small and yet powerful boards, along with a large variety of electronic components for personalizing them according to the user needs. In addition, there are various companies offering extra modules or add-ons such as Seeedstudio [102], which offers their own sensors for developing personalized frameworks based on Grove technologies. Similarly, Adafruit [1] provides different embedded platforms, as well as all kinds of electronic components, including personalized add-ons for batteries, communication modules, and sensor boards compatible with the most widely extended platforms: Raspberry Pi, Arduino, BeagleBone, Intel Edison, or Intel Galileo.

Focusing on final solutions, TST [109] offers products in the field of Smart Cities (Waste Management, Industrial Control, Light Control, etc.), basing their solutions on their own hardware platform. Likewise, Libelium [78] is a company providing various products in the field of monitoring (Environment monitoring, Agriculture, Water monitoring, etc.); most of the components, and the programming tool used, are based on the Arduino platform. However, from a crowdsensing perspective, the solutions offered are inadequate due to the relatively large sizes of the devices, being mostly oriented for public infrastructure deployment.

Based on the state of the art, we now provide an in-depth analysis of the different hardware and software components applicable to our mobile air quality sensing context.

In fact, to enable air quality data acquisition, specialized pollution sensors must be connected to a microcontroller or microcomputer via an analog or digital port. Moreover, for communication tasks, the microcontroller/microcomputer must be connected to the communications module via an USB port or UART interface. In this sense, available options will depend on the microcontroller/ microcomputer characteristics.

5.2.1 Microcontroller/Microcomputer-Based Embedded Systems

Despite the lightweight processing constraints, there are several options available for embedded systems acting as central elements in the sensor design. In fact, it is possible to use microcomputers such as Raspberry Pi, BeagleBone, or Intel Edison, which use a standard operating system, and that allow developing applications for sensing tasks in a straightforward manner. Alternatively, it is possible to use a microcontroller board such as Arduino, and develop application-specific firmware instead.

Raspberry Pi is nowadays one of the most popular microcomputers worldwide. It is a low-cost and small-sized computer that allows connecting standard PC peripherals including a monitor, a keyboard, and a mouse. It was designed to explore computing, and it supports different Operating Systems: Raspbian, which

is based on Debian, and also Ubuntu Mate or Windows 10 IoT Core, thereby allowing to use several programming languages. In addition, all Raspberry Pi versions benefit from several input/output ports operating at 5 Volts, thus being ideal for all sorts of IoT projects.

There are different versions of the Raspberry Pi, as shown Figure 5.2, being model A and type 2 the most commonly used. They have different characteristics, e.g., Type B offers better performance in terms of memory and processing, but Model A consumes much less power. Recently, Raspberry Pi 3 has appeared, offering better features than previous versions, being the major difference the integration in the main board of a Bluetooth and a Wi-Fi module, thereby facilitating communication tasks in the scope of IoT projects.

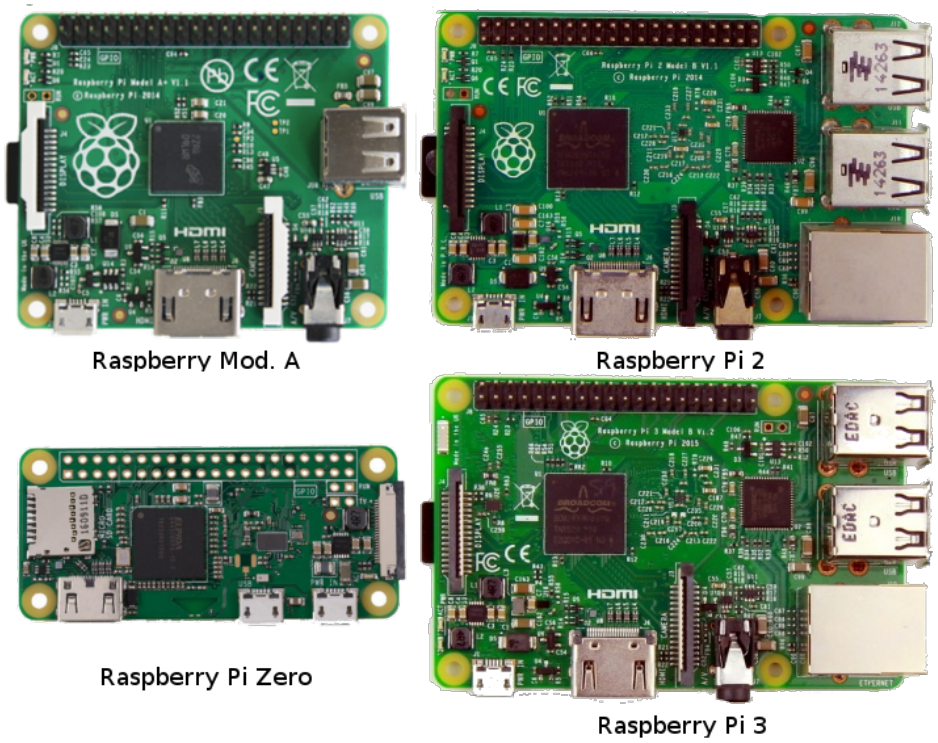


Figure 5.2: Overview of different Raspberry Pi models.

BeagleBone is a small computer running a Linux Operating System called Angstrom, and supporting various software distributions such as Android or Ubuntu. It has an USB port for connecting distinct peripherals, along with an HDMI port for video connection, allowing to use it as a regular computer. It has two 46 pin headers which operate at 3.3 V, allowing to augment the available functionalities

by connecting different digital or analog devices like sensors or actuators.

Commercially, we can find several Beaglebone versions, being Beaglebone Black the most commonly used (see Figure 5.3).

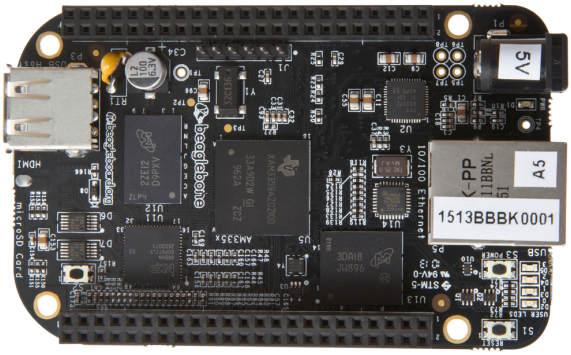


Figure 5.3: Beaglebone Black microcomputer overview.

Intel Edison is a tiny but powerful computer developed by Intel. It is designed for IoT applications, targeting at both prototypes and commercial solutions with performance constraints. It supports a modified Linux distribution (Yocto) as its Operating System, and it integrates both Wi-Fi and Bluetooth 4.0 interfaces. In addition, it can benefit from two types of expansion board: an Arduino Expansion board, and a mini breakout board (see Figure 5.4).

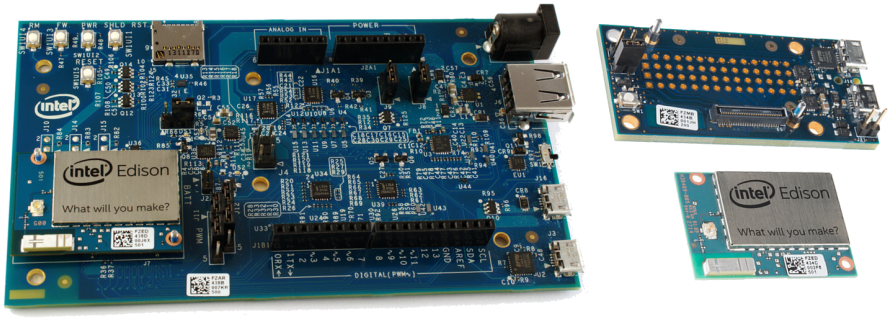


Figure 5.4: Intel Edison with two extension boards.

Pycom [95] is a microcontroller based on the ESP32 chip with 24 GPIO pins, 2 UARTs, 1 SPI and 1 I2C port, using a firmware based on micropython. It can be equipped with several communication interfaces such as Wi-Fi, Bluetooth Low

Energy, LoRA, and Sigfox. Moreover, using an expansion board, it can incorporate an SD Card, as well as different sensors such as Gyroscope, Accelerometer, or GPS (see Figure 5.5).

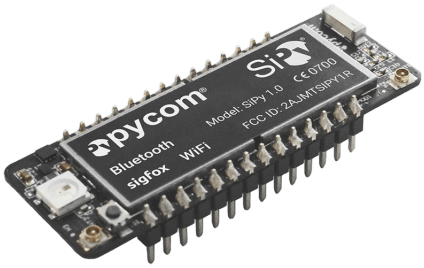


Figure 5.5: Pycom module.

Arduino Uno is an open source prototyping platform characterised by easy-to-use hardware and software. It has several analog and digital input/output pins to connect sensors, actuators or complementary boards, allowing to create a wide variety of IoT solutions. Arduino has its own programming language based on Wiring, and its own Arduino Software based on Processing [18]. It has a central microcontroller, and an USB port for programming and power supply.

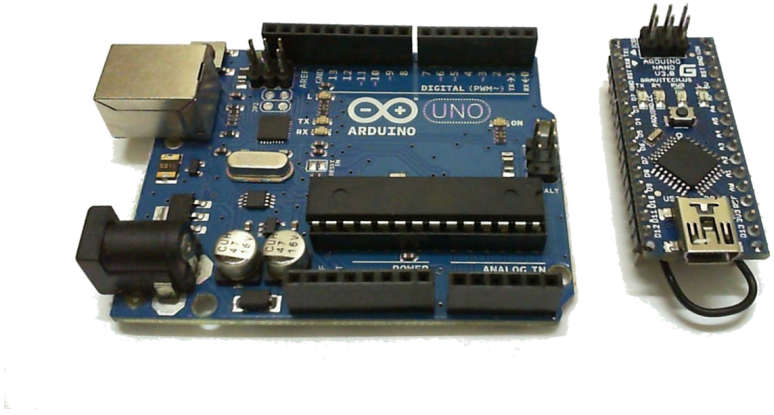


Figure 5.6: Arduino Nano module (right) and Arduino Uno module (left).

Table 5.1: Comparison of different processing module components.

Board	CPU Speed	Memory Storage	Power Comp.	Ports A/D	Size Weight	Extras	Price	Operating System
Respberry Pi Model A	700 MHz	256 MB SD (4 GB)	0.8 W	0/30	6.5×5.5 cm 100 g	1 USB port	$25	Raspbian
Respberry Pi 2	900 MHz	1 GB SD (4 GB)	1.5 W	0/30	8.4×5.5 cm 136 g	4 USB ports	$35	Raspbian Ubuntu Mate Windows 10 IoT
Respberry Pi 3	1.2 GHz	1 GB SD (4 GB)	1.8 W	0/30	8.4×5.5 cm 136 g	4 USB ports Wi-Fi Bluetooth	$40	Raspbian Ubuntu Mate Windows 10 IoT
Beagle Bone	720 MHz	512 MB SD (4 GB)	1.0 W	14/6	8.4×5.5 cm 100 g	1 USB port	$75	BeagleBone Linux
Intel Edison	500 MHz	1 GB SD (4 GB)	0.6 W	14/6	5.9×2.8 cm 82 g	Wi-Fi Bluetooth	$90	Yocto Project
Pycom	32 MHz	1 kB 32 kB	0.2 W	14/6	6.8×5.4 cm 100 g	Wi-Fi Bluetooth LoRa	$45	Micropython
Arduino Uno	32 MHz	1 kB 32 kB	0.2 W	14/6	6.8×5.4 cm 100 g		$25	Processing-based
Arduino Nano	16 MHz	0.5 KB 16 kB	0.2 W	14/6	4.2×1.8 cm 20 g		$10	Processing-based

Arduino nano is a tiny prototyping platform that maintains the Arduino Uno concept, using the same programming languages and the same libraries. It was also developed for prototyping solutions, but it is smaller than the standard Arduino, and it has less available memory (see Figure 5.6).

Table 5.1 allows comparing these five embedded systems by providing a summary of the most significant technical details.

Operating Systems for IoT Microcomputers

The choice of an adequate Operating System is a very important issue in the design of a sensor since it will shape the corresponding software architecture. Thus, we now proceed by analyzing the different operating systems available for the microcomputers referred above.

Table 5.2: Comparison of different operating systems.

Operating System	Booting Time (s)	Minim. Memory	GUI	Threading Type	Programming Languages
Raspbian	25	150	Yes	Multiple	C Java Python Scratch
Raspbian Lite	15	50	No	Multiple	C Java Python Scratch
Ubuntu MATE	30	200	Yes	Multiple	C Java Python
Yocto Project	20	150	No	Multiple	C Java Python Arduino-based
Angstrom Linux	15	100	No	Multiple	C Java Python Bonescript
Windows 10 IoT	35	250	No	Multiple	Visual Studio C
Micropython	<1	<1	No	Multiple	Python -
Processing-based	<1	<1	No	Single	Wiring -

Raspbian is the most common operating system designed for Raspberry Pi. It is supported by all Raspberry Pi versions, and it has two versions: (i) a complete version with a graphical interface and many development tools which facilitate

the development of solutions, but that consumes a lot of resources; and (ii) a lite version, without graphical interface, and with just a basic set of preinstalled software, allowing to add only those packages that are actually required.

Since it is a Linux-based operating system, it supports several programming languages like C, C++, Java, Scratch, Python, or BASH.

Ubuntu MATE is an option for Raspberry Pi microcomputers that is supported by model 2 and model 3. This operating system attempts to be simple from the end user perspective, integrating several entertainment applications, although it is also possible to add development tools.

Angstrom is a modified Linux optimized for Beaglebone microcomputers. It has no graphical interface, and it supports several programming languages such as Python, C, Java, or BASH. In addition, it has its own programming language called BoneScript, which is based on the node.js language.

Yocto Project is a complete embedded Linux development environment with tools and methods to facilitate the creation of embedded systems. It can be configured to run Arduino-based or Linux-based programs, thereby offering a great flexibility.

Micropython [86] is a Python 3.5 implementation optimised for running in microcontrollers. It allows interacting via a prompt, executing commands or running scripts in an autonomous way. It is entirely compatible with Python, and it includes modules for accessing low-level hardware.

Windows 10 IoT Core is a Windows 10 based operating system specially designed for Internet of Things projects using small devices. It is supported by Raspberry Pi 2 and 3, Arrow DragonBoard 410c, and MinnowBoard MAX. Windows 10 IoT Core relies on the rich, extensible Universal Windows Platform (UWP) API for building solutions. It can be used together with the Visual Studio environment for programming.

Table 5.2 shows a brief comparison of the different Operating Systems currently available.

5.2.2 Air Pollution Sensors

Nowadays, we can find a wide variety of sensing technologies (see Figure 5.7) for gas detection (Metal oxide semiconductor, polymer, carbon nanotubes, moisture absorbing materials, Optics, Acoustics, etc.) as shown in [79]. Each technology has different properties, calibration processes, costs, among other characteristics, and so a comparison between these different technologies is required.

To evaluate the different sensing technologies, we must consider some characteristics, especially when focusing on the design of small mobile sensors: (i) sensibility, which refers to the range of values that the sensor can measure; (ii) selectivity, which is the capability of reacting only to the target gas; (iii) linearity, which is the rate of change with respect to gas variations; (iv) response time,

Figure 5.7: Different types of air pollution sensors.

which is the time required to start measuring correctly; (v) power consumption; and (vi) price.

In the market, we can easily find various gas sensors for air pollution monitoring, being the three following types of sensors the most common: electrochemical, semiconductor, and infrared. These sensors have a wide range of prices and characteristics. Below we provide more details about each of these sensor types:

Electrochemical gas sensors measure the concentration of some air pollutant by oxidizing or reducing its internal porous membrane, thereby producing current changes. Usually these sensors behave quite linearly, allowing to make accurate measurements. They typically operate at 5 V, having a power consumption of about 600 mW; most pollution sensors in this category cost between $100 and $400.

Semiconductor gas sensors are the most common gas sensors because of their low cost and high sensitivity. They have an internal conductive material that increases their conductivity level in the presence of a specific air pollutant. These sensors are nonlinear and have a low selectivity, adding difficulty to the monitoring process. Usually they operate at 5V, having a power consumption in the 500–900 mW range, and their cost varies between $10 and $35.

Moisture absorbing material sensors are used for measuring temperature and humidity. Their dielectric constant varies according to the water content in the environment. They operate at 5V, having a power consumption of about 0.5W. In addition, they are very cheap, with a price of about $5.

Infrared sensors measure gas/element variations by detecting interferences in an infrared laser. They are specially adequate for monitoring pollutants such as fine particulate matter sized less than 10 micrometers (PM_{10}), or fine particulate matter sized less than 2.5 micrometers ($PM_{2.5}$). They usually operate at 5V,

having a power consumption of about 1W, and their cost is about $40.

Table 5.3 provides a brief summary of the most significant aspects of these different types of sensors for comparison purposes.

Table 5.3: Comparison of different sensing module components.

Sensor Type	Sensitivity	Selectivity	Linearity	Response Time	Power Comp.	Size (cm)	Price
Electrochemical	Medium	Medium	High	Medium	0.6 W	2×4	$200
Semiconductor	High	Low	Low	Low	0.5 W	2×4	$10-35
Moisture Absorbing	High	Medium	High	High	0.5 W	2×4	$5
Infrared	High	Low	Medium	High	1.0 W	15×10	$40

5.2.3 Communications Modules

Although the RFID [74] standard was developed for IoT solutions, there are currently several options available for providing communications between the sensors and the mobile terminal (usually a smartphone), as analyzed in Chapter 3. Figure 5.8 shows some communication modules examples. We now proceed to analyze the technical characteristics associated to the different available options.

The main characteristics to consider are: (i) distance, that is, the wireless coverage range; (ii) communication type, which refers to the characteristics of the channel over which messages are transmitted—usually two types are considered, i.e. serial-based, when a communications channel is open to transmit a stream of data, or message-based, when the data are transmitted via a unique message; (iii) message size, which refers to the maximum size of the message when the communication is message-based; (iv) power consumption; and (v) price.

Figure 5.8: Examples of some communication modules.

Table 5.4 shows a brief summary of the most significant aspects to consider in terms of communication modules.

Table 5.4: Comparison of different network module components.

Module	Commun. Type	Max. Msg. Size	Data Rate	Distance	Power Comp.	Price
Wi-Fi	Serial-based	-	+54 Mbps	100 m	0.5 W	$10
NFC	Message-based	32 kB	424 Kbps	0.04 m	0.1 W	$35
ZigBee	Message-based	128 bits	250 Kbps	100 m	0.1 W	$40
SIGFOX	Message-based	96 bits	140 msg/day	+1 km	0.3 W	$60
LoRa	Message-based	0.1, 1 or 10% TimeOnAir	250–5400 bps	2–15 km	0.1 W	$45
Bluetooth	Serial-based	-	+2.1 Mbps	10 m	0.2 W	$10
BLE	Message-based	160 bits	1 Mbps	10 m	0.05 W	$15

We can observe that all options are relatively cheap, with prices in the range from $40–$50, but the most widely used in the market, i.e., Wi-Fi and both Bluetooth and Bluetooth Low Energy, are the cheapest ones, with prices in the range $10–$15. In terms of power consumption, the best option is Bluetooth Low Energy (0.05 W), although ZigBee, LoRA, and NFC also exhibit low power consumption levels. In terms of bitrate, the best performing technology is Wi-Fi, being that typical wireless sensing technologies, such as LoRA, ZigBee, or SIGFOX, have a low bitrate. Regarding the communications type, there are two options: (i) Wi-Fi and Bluetooth, which open a serial communications channel for

transmitting a stream of bytes; and (ii) Bluetooth Low Energy, ZigBee, Sigfox, LoRA, and NFC, which transmit a message per iteration.

5.3 Mobile Air Pollution Sensor Design

The design of a small and cheap mobile sensor is a basic requirement for air pollution monitoring. After analyzing the main technical characteristics of hardware components available in the market, it quickly becomes evident that there are many options for creating a mobile air quality sensor for crowdsensing in the Internet of Things context.

If focusing on the mobile sensor/smartphone wireless connection, the best option is using Bluetooth Low Energy since it is able to fulfill all requirements (low power consumption, low price, and small-sized modules), whereas all other options have different drawbacks. For instance, the ZigBee technology, despite being the most extended technology for wireless sensor networks due to its low power consumption, it is not supported by current smartphones. The SIGFOX technology has very strict restrictions regarding the number of messages that it is possible to generate per time slot, thus having little applicability to our aims. The Wi-Fi technology, although being widely used and having a large coverage range, consumes more power, and it is typically used for Internet connectivity. Finally, the major problem of the NFC technology is its coverage range (only about 4 cm).

In terms of sensor device prototyping, the best option for creating a small and cheap mobile sensor for air pollution monitoring is the Semiconductor gas option. Notice that it is cheaper than the electrocavailable.hemical sensor and, even though the latter is more accurate, the error introduced can be mitigated by combining information from other nearby users. Concerning infrared sensors, they are only applicable to a specific type of pollutant (fine particulate), while moisture absorbing sensors are mostly used to measure temperature and humidity.

Next, we propose and evaluate some possible configurations for the different studied boards using a semiconductor gas sensor for pollution monitoring, and a Bluetooth Low Energy interface as the transmission technology. For all the proposed hardware configurations, it is possible to use an external USB charger as a power supply to offer more autonomy, while maintaining the support for mobility.

Figure 5.9 shows the four solutions we have developed to evaluate the different hardware options, and Table 5.5 shows a comparison between these proposed solutions.

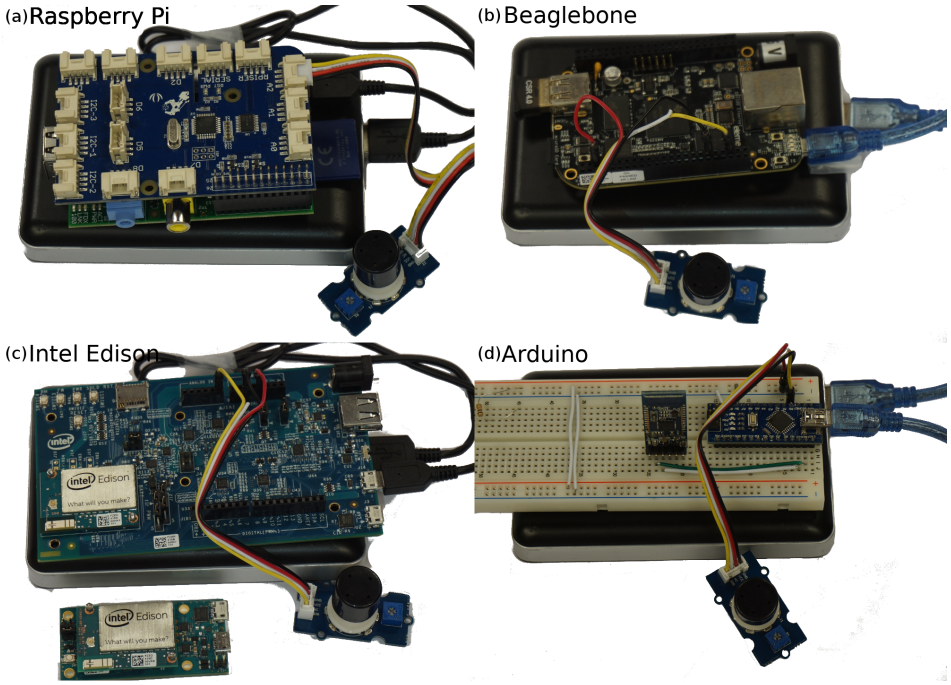


Figure 5.9: Proposed mobile sensing solutions for air quality monitoring.

Table 5.5: Comparison of the four different solutions proposed.

Module	Extras	Network	Power Comp.	Weight	Price	Flexib.	Develop.	Comput. Power
RPi 3	+converter	–	2000 mW	200 g	€90	★★★	★★★	★★★
Beaglebone	–	+BLE usb	1500 mW	150 g	€110	★★★	★☆☆	★★★
Intel Edison	+breakout +expansion	–	1000mW	100g 200g	€130	★★★	★★☆	★★★
Arduino	+circuit	+BLE uart	600mW	60g	€55	★☆☆	★★★	★☆☆

The first hardware option we propose is based on the Raspberry Pi (RPi) model 3 platform (see Figure 5.9a). Since it has no analog ports, it has to be provided with an analog/digital converter. For this purpose, we propose using GrovePi [101], which is an extension board that allows connecting several analog/digital grove ports to a Raspberry Pi in an easy way. Furthermore, since the Raspberry Pi has several USB ports, we propose using a standard USB Bluetooth module for Raspberry Pi 2, or the built-in Bluetooth module for Raspberry Pi 3. With this solution, it becomes possible to run several programming languages, as it is possible to install a Linux or a Windows 10 IoT operating system, and there are a lot of development efforts around it. This configuration has a power consumption

of about 2000 mW, a total weight of 200 g, and it costs about $90. Overall, it is the most power-hungry solution among the four proposed, but it becomes the best option for quick prototyping due to its flexibility and large community of developers.

The second hardware approach relies on the BeagleBone board (see Figure 5.9b). It has several analog ports, allowing us to connect the sensor directly to this board without any intermediate device. In addition, since the BeagleBone has an USB port, it is possible to use a standard USB Bluetooth module. It runs a Linux-based operating system, allowing to use different programming languages, but there are not too many developments or projects focusing on this solution. This configuration has a power consumption of about 1500 mW, a total weight of 150 g, and it costs about $110, thus being one of the most expensive options. It is not a very useful prototyping platform since it has characteristics that are similar to the Raspberry Pi, but it has a smaller developer community and less support.

The third hardware solution we propose is based on the Intel Edison platform (see Figure 5.9c). It has an embedded Bluetooth interface, but it does not provide an analog port to directly connect the sensor. For this purpose, it is possible to use: (i) the Arduino expansion board; or (ii) the Breakout board [67]. The first sensor connection option is very simple; however, the sensor becomes excessively large. Regarding the second option, the overall size remains small, but it is necessary to make an *ad hoc* circuit to connect the sensor. The Edison board supports a Linux-based operating system (Yocto) including the possibility to run Arduino-based scripts. For this last configuration, the power consumption is of about 1000 mW, the total weight is 200 g, and it costs about $130, making it the most expensive option. It is mostly useful for end solutions due to its price.

The last embedded solution we propose is based on the Arduino platform (see Figure 5.9d). Since it was designed for these types of solutions, it becomes easy to connect a sensor via the existing analog ports; nevertheless, USB ports are not available, and so a Bluetooth module must be connected via an UART port for both Arduino Uno and Arduino Nano boards. This solution only runs Arduino-based scripts, reducing the programming flexibility, but we can find a lot of developments using this platform.

For the Arduino Uno solution, the power consumption is about 600 mW, the total weight is 150 g, and it costs about $55. This option is improved by the Arduino Nano solution, which has a power consumption of about 600mW, a total weight of 60g, and it costs about $50. The latter option is better than all others in terms of consumption, weight, and price, having as its only drawback the limited memory/CPU resources. It is also useful for restricted environments where the power consumption is very limited.

5.4 Summary

In this chapter, we have analyzed the requirements of an embedded mobile sensor platform from a crowdsensing perspective, identifying the basic tasks the sensor must be able to perform. Besides, an analysis of the hardware architecture requirements has been done, and candidate off-the-shelf hardware components have been analyzed. Finally, several complete hardware configurations meeting all the design requirements have been developed and compared in terms of power consumption, weight, and cost. Overall, we have found that the Arduino Nano platform, despite having very limited resources, is able to fulfill the established requirements, thus being the most recommendable alternative in terms of price, weight, and power consumption features.

Regarding other hardware alternatives, microcomputers like Raspberry Pi, BeagleBone, or Intel Edison are more powerful and flexible by supporting a standard Operating System, thereby allowing to quickly deploy any application. We believe that the Raspberry Pi solution can be the best option for quick prototyping. For more professional solutions, requiring higher processing capacity, the Intel Edison becomes a better option, although imposing a higher overhead in terms of development time. Finally, Arduino becomes an option for very restricted environments.

Overall, we find that a hardware solution applicable to all IoT contexts, and meeting low size and low power requirements, along with adequate communication interfaces and battery capacity, is still missing, although in years to come many more products are expected.

Chapter 6

Finding the optimal measurement strategy

After designing the sensor hardware design, defining the overall architecture and the monitoring approach, we now proceed to determine the optimal strategy for air pollution data collection using mobile sensors.

With this purpose, we first analyze the impact of mobility on sensor readings by comparing static against mobile measurements. Also, we determine the influence of sensor orientation in the mobile sensing process. Afterward, we analyze the impact of reducing the sampling frequency on the kriging process accuracy under mobile scenarios. Similarly, we analyze 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.

6.1 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.

To have further insight on how these results are distributed, Figure 6.1 shows that mobility, at least at the speed used for testing, i.e., up to $20Km/h$, does not have a significant impact on sensor measurements.

Table 6.1: 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

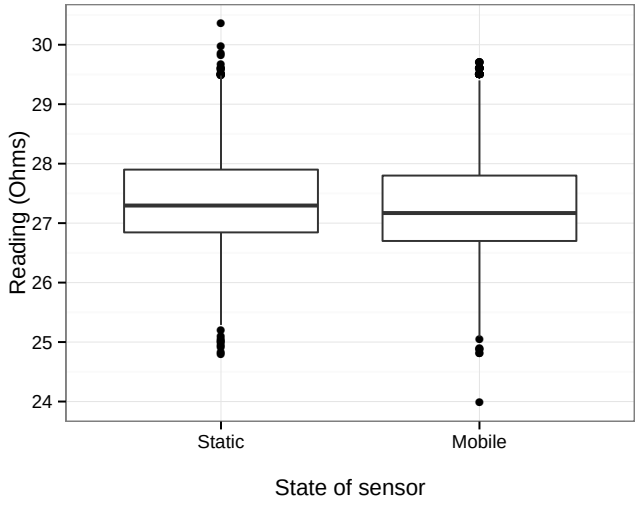


Figure 6.1: Analysis of the variability of mobile sensor readings: static vs. mobile sensor.

The results for the t-test analysis are shown in Table 6.1, 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 with an $\alpha = 0.05$, 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 6.2 shows that the actual sensor orientation has little impact on the data capture process, being the differences between different orientations minimal. Anyway, the backwards orientation option shows greater resemblance with the static measurements, and it was adopted for the tests that follow.

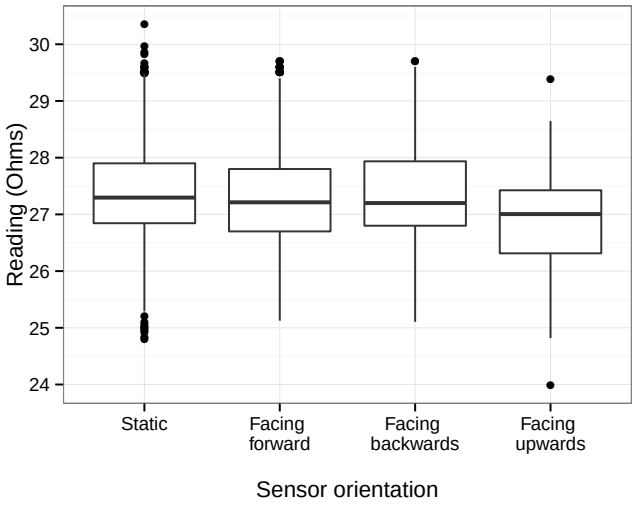


Figure 6.2: Analysis of the variability of mobile sensing for different sensor orientations.

6.2 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. This experiment was achieved by filtering the full trace, and retrieving datasets with 1/2, 1/4, 1/6, 1/8, and 1/16 of the total data, respectively.

Next, we performed spatial interpolation through kriging for each trace, obtaining a detailed pollution distribution. We used the full trace (samples every 5 seconds) as reference, and compared it against the results obtained using the other datasets.

Table 6.2 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 reference, as shown in equation

Table 6.2: 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

6.1.

$$s_i = 1 - \frac{1}{m \cdot n} \sum_{x=0}^{m} \sum_{y=0}^{n} \left| \frac{k_{i_{xy}} - k_{0_{xy}}}{\triangle k_0} \right| \tag{6.1}$$

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 6.2, we observe 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 6.3 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 6.4. 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 also 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 reference (5 seconds). Based on these maps, it becomes quite clear that little differences in terms of basic statistical analysis can represent huge differences in terms of the spatial distribution of those values.

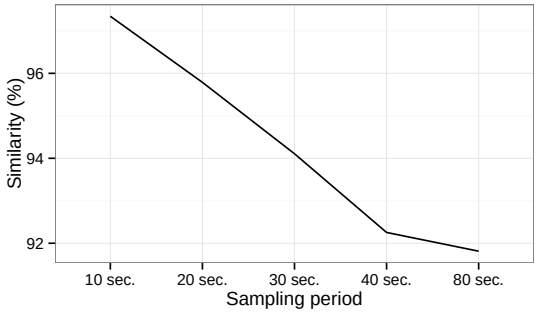


Figure 6.3: Analysis of the output similarity with respect to reference sampling period (5 s).

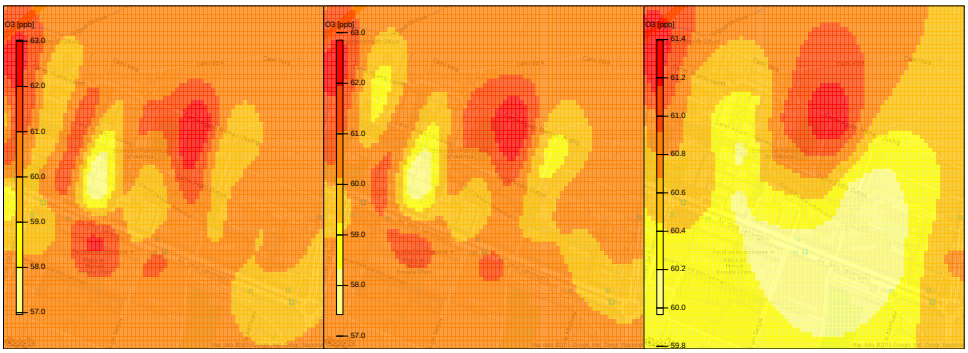


Figure 6.4: Heatmaps for the ozone distribution using different sampling periods (5, 20, and 80 seconds).

6.3 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 initial 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 total data.

Similarly to the previous section we perform, for each dataset, a statistical analysis of the resulting data, also obtaining the pollution heatmap generated

Table 6.3: 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

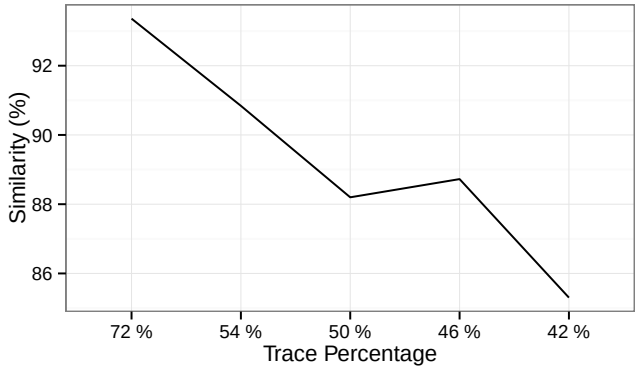


Figure 6.5: Analysis of the similarity with respect to the reference trace (100 % of the data used).

through kriging interpolation, and calculating the similarity index using equation 6.1.

Table 6.3 presents the statistical analysis results showing the mean, the standard deviation, and the similarity, being the latter calculated using the initial dataset as reference.

Based on Table 6.3, 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 6.5 shows a decreasing trend when spatial sampling decreases. Compared to the time sampling results of Figure 6.3, 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.

Figure 6.6 shows detailed maps for datasets representing 100%, 72%, 50% and 42% of the data. Based on these heatmaps, we can clearly conclude how spatial

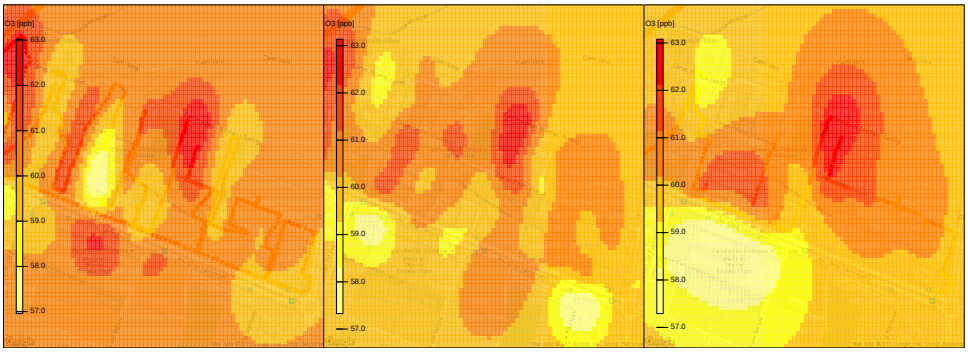


Figure 6.6: Heatmaps for the ozone distribution using different fraction of the original trace (100%, 72% and 42%).

subsampling causes a 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 obtained results.

6.4 Validation of the proposed approach

As stated at the beginning of the paper, current infrastructure elements allow measuring pollution levels in cities with high accuracy, although with a 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 concerning to the time of year, and then by comparing the ozone maps generated when relying on either infrastructure-based or mobile-based sensing.

We started by gathering data in different areas of Valencia using the proposed mobile sensors. Different experiments have been conducted at different times, allowing 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 6.7 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,

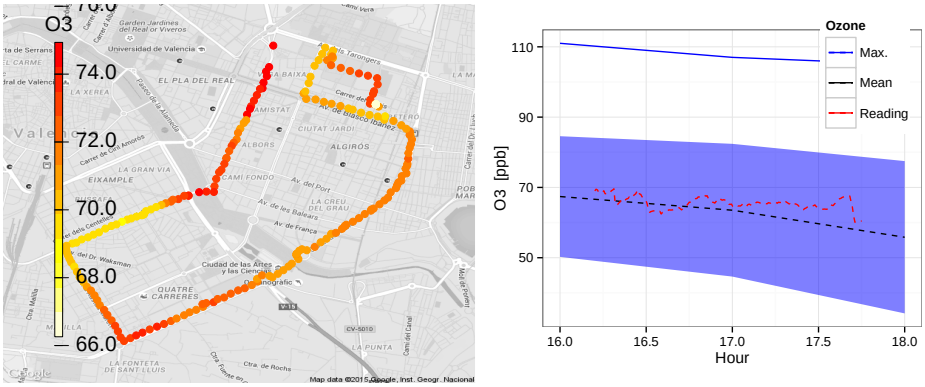


Figure 6.7: Comparison between captured data and typical values for the day/time of the year.

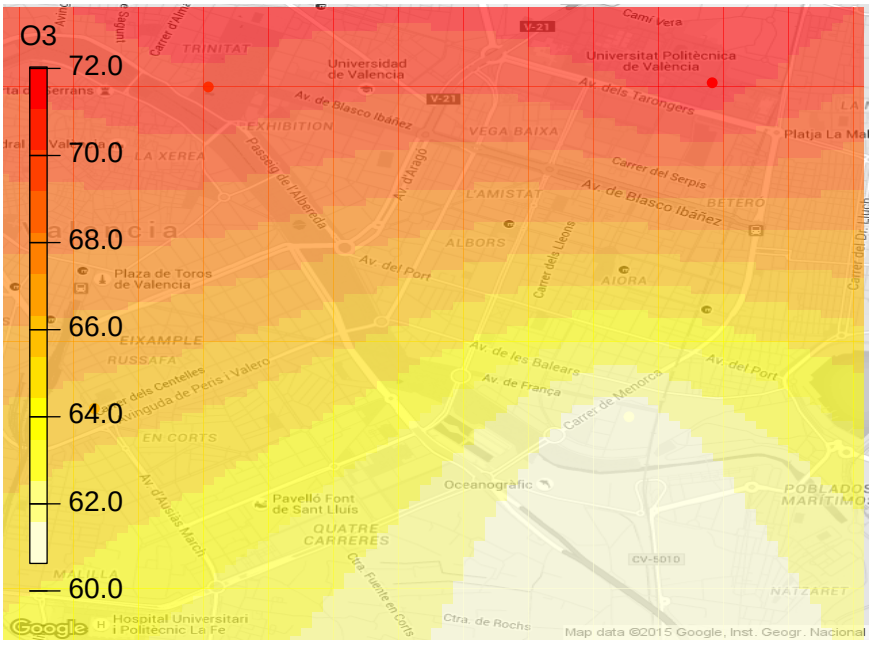


Figure 6.8: Ozone levels in the target region using infrastructure data.

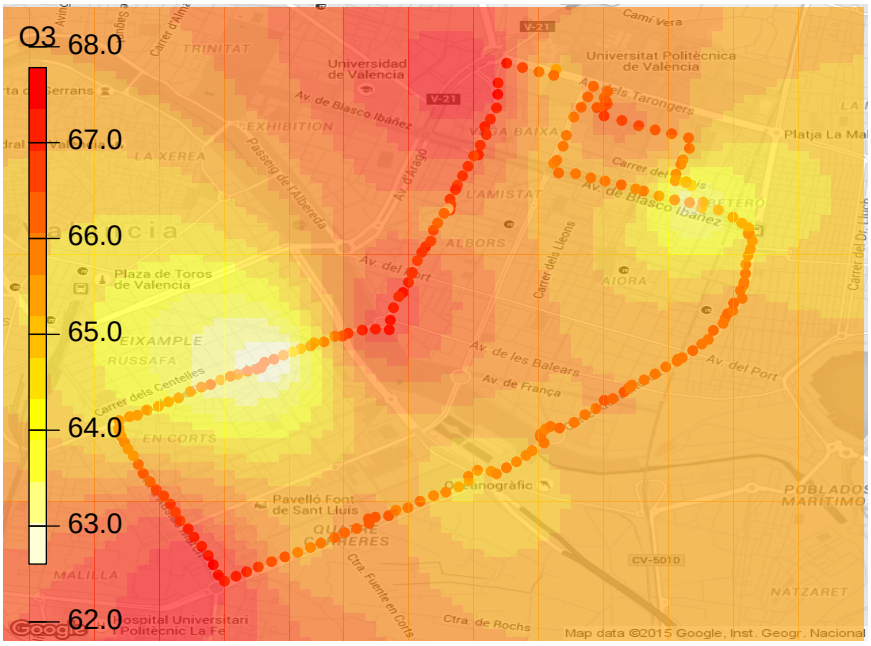


Figure 6.9: Ozone level in the target region using mobile sensing data.

we are able to obtain reliable data despite using low-cost sensors, allowing to focus our analysis on the spatial variations of pollutants.

We now proceed to compare the actual heat maps for a specific date and time of day using only infrastructure data, and only data obtained by our mobile sensor. We can see that, by relying on our proposed architecture (see Figure 6.9) it becomes possible to observe in detail even small pollution variations, while using only infrastructure-based data (see Figure 6.8), 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 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 even small variations can be perceived with great detail, thereby meeting the proposed goal.

6.5 Summary

Based on the architecture defined, and having developed the different system elements, in this chapter we analyzed the optimal strategy for air pollution data collection using mobile sensors.

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 showed that the sensor orientation and the sampling period, within certain bounds, have very little influence on the data captured, while the actual path taken had 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 platform 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 the proposed mobile-sensing approach is able to provide a much higher data granularity.

Part III

UAV-based sensing in rural areas

Chapter 7

Using UAV-based systems to monitor air pollution in areas with poor accessibility

Air quality monitoring is relevant not only for the people living in urban areas, but also in rural environments because it directly affects crops and different animals/insects [83]. Thus, different solutions for measuring air quality should be sought for such environments.

Regarding mobile sensing, the crowdsensing approach can deal with air pollution monitoring in urban areas, as analyzed in previous chapters using the EcoSensor platforms (Chapter 4), by installing small sensors in bicycles or the public transport system. However, in the countryside, or in areas with poor accessibility, the usage of ground vehicles can be quite limited.

In this chapter we propose the use of UAVs, specifically of the multicopter type, as an efficient solution for quickly and easily monitoring air quality in any region where ground mobility is a poor option. To meet the proposed goal, we plan mounting a computing unit endowed with pollution sensors on the UAV to create a flying air quality station. Our solution allows programming the UAV directly from a smartphone by defining the target region to monitor. The UAV then flies autonomously throughout the target area and returns with the desired data. Again using a smartphone, data is retrieved from the UAV and uploaded to a server for storage and analysis, similarly to the approach adopted in the EcoSensor platform.

7.1 Methodology and Proposed Architecture

To achieve the desired solution, the proposed methodology consists of: (i) clearly defining the design requirements of the target solution; (ii) proposing a high-level overview of the global system architecture; (iii) defining the desired information flow in the scope of the proposed architecture; and (iv) developing an initial prototype meeting all previous requirements and conditions.

7.1.1 Design requirements

We have defined five basic requirements in order to properly fulfill our objective of developing an UAV able to monitor the air pollution in a target area.

Requirement 1: Open multicopter design Regarding the multicopter design, we seek an open platform that is easily scalable and reusable by other researchers, and whose cost is reduced. To achieve this objective, the different components used to manufacture the multicopter should be standard components easily found in the market, and that are consistent with the "open-source" philosophy. This will allow any researcher to modify or replace any multicopter component without a negative impact on the overall performance of the device. To achieve this goal, the following strategy is proposed: (i) Basic multicopter components, such as the frame, engines, propellers, Electronic Speed Controllers (ESCs) and batteries, shall be chosen among models available in the market, easily accessible, and avoiding expensive elements. (ii) The flight controller should be open to enable possible modifications, and shall have a serial communications interface following a well-known protocol. (iii) An element offering high-accuracy Global Positioning System (GPS) and compass is required for navigation, and it should be fully compatible with the flight controller. These elements are particularly important to allow the controller to know the location, speed and altitude of the device at any time, and also to fly to predefined positions autonomously. (iv) The remote control used should be universal to allow adapting it to any type of multicopter, programmable to associate different functions to each of the controls on demand, while integrating aspects such as telemetry on the same radio frequency link.

Requirement 2: Integration of an embedded system to provide intelligence to the multicopter In order to perform additional tasks not related to flight and navigation, the existing multicopter hardware should be extended through an embedded, low-power device that is able to operate as a fully-featured computer in practical terms. Preferably, it should support the Linux operating system, along with all associated tools to provide a platform that is well known and widely used by the open source research community. This embedded system should be linked to the flight controller to manage flights.

Requirement 3: Integration of a complete communications system The communications system of an UAV is especially critical considering that any failure can cause losing the control of the device, usually resulting in it being crashed or become lost. For this reason, the communications systems adopted should be resilient against attacks, and so its design should target security and robustness. To allow the UAV to be controlled by mobile devices such as smartphones or tablets, it should act as a wireless access point, creating a secure wireless network to which these mobile devices can connect to in order to control the UAV remotely. This network will be complemented by the UAV's native communications system, including remote control and wireless telemetry.

Requirement 4: Integration of pollution monitoring sensors The multicopter should include different types of sensors, including temperature, humidity, and different pollutants such as ozone, carbon monoxide, and nitrogen oxides, among others. These sensors should preferably be connected to the embedded system integrated in the multicopter.

Requirement 5: Cloud-based storage, processing and display of data The data collected by the UAV should be retrieved by a mobile terminal and uploaded to the cloud, where they will be stored along with temporal and geographical information. The server is responsible for data processing tasks such as performing linear unbiased estimations, statistical processes commonly referred to as ordinary Kriging, and cartographic visualization of the data through color maps. This cloud system will allow us, for example, to study the pollutant levels in a given area at different times, on different days, and at different altitudes.

7.2 Overview of the proposed solution

To implement a solution for air pollution monitoring using UAVs we have to consider, like in any cyber-physical system, two main aspects: (i) the hardware configuration, and (ii) the control process for controlling the system behavior.

By following these guides, our proposal can also be split into two parts: (i) the physical configuration of the UAV and the environmental sensors, which are analyzed in this chapter, and (ii) the algorithm used to control the UAV for automatically monitoring a specific area (PdUC), analyzed in chapter 8.

7.2.1 Physical UAV Configuration

We have designed a scheme to dynamically drive the UAV by connecting the UAV control module to a Raspberry Pi [49], and connecting the latter to the set of pollution sensors via an analog converter. The scheme is shown in figure 7.1.

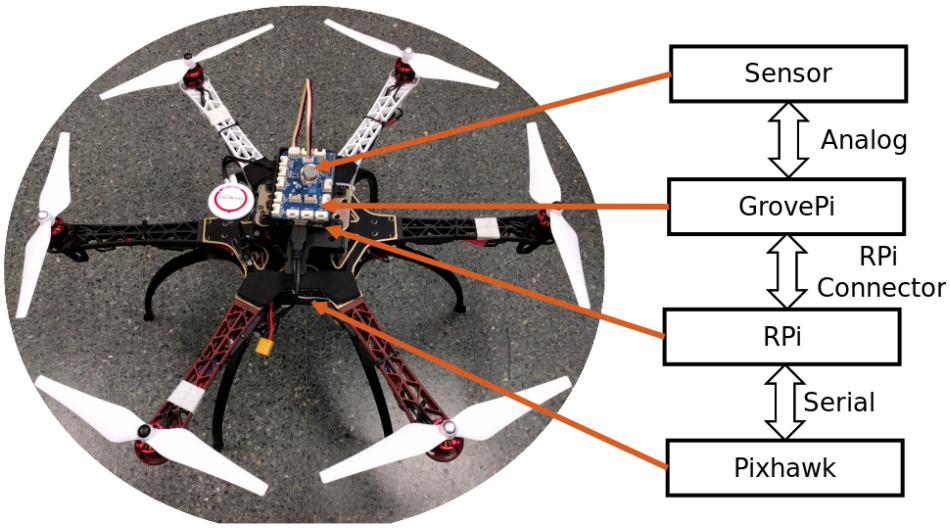


Figure 7.1: Proposed UAV with air pollution sensors.

The UAV is driven using a Pixhawk Autopilot [84, 85], which controls its physical functioning. The Raspberry Pi is mounted over the UAV chassis, and it is connected to the Pixhawk through a serial port. The sensors are connected to the Raspberry Pi using a Grove Raspberry Hat (GrovePi) [66], which allows connecting different kinds of COTS sensors easily. Specifically, we are using:

- **Pixhawk Autopilot:** a high-performance flight control module suitable for several types of autonomous vehicles including multi-rotors, helicopters, cars, boats, and fixed-wind aircrafts. It is developed under the independent, open hardware Pixhawk project, and it has two main components: (i) an Autopilot hardware that provides an industry standard autopilot module designed as a 168 MHz Cortex M4F CPU with 3D ACC/Gyro/MAG/Baro sensors, microSD slot, 5 UARTs, CAN, I2C, SPI, ADC, etc.; (ii) an Autopilot software that includes a Real-Time Operating System (RTOS) with a POXIS-style environment to control the drone.
- **Raspberry Pi:** one of the most popular Single Board Computers (SBCs) worldwide. It is a low-cost and small-sized piece of hardware that allows exploring computing, and that supports different Operating Systems. The most popular of them is Raspbian, which is based on Debian, although Ubuntu Mate or Windows 10 IoT Core can also be installed, thereby allowing developers to use different programming languages. Besides, all Raspberry Pi versions benefit from several input/output ports operating at 5V, thus being ideal for all sorts of IoT projects.

- **GrovePi:** extension board that allows to connect several analog/digital *grove* ports to a Raspberry Pi in an easy way. It has several Grove ports: seven digital ports, three analog ports, three I2C ports, one serial port to the GrovePi, and a serial connector to the Raspberry Pi.
- **Grove Sensors:** sensors which use a grove-standardized connector, providing an easy connection to different boards like GrovePi. There are several COTS environmental sensors such as CO2, CO, or Alcohol, among others. Specifically, we mostly focus on the Ozone sensors (MQ131).

Figure 7.2 shows the closed-loop control scheme of our proposal. The Pixhawk autopilot is responsible for the physical control system of the UAV (Lower level), while the Raspberry Pi is in charge of the Guidance system (Higher Level), determining the path to be followed according, in our case, to the pollution variations detected.

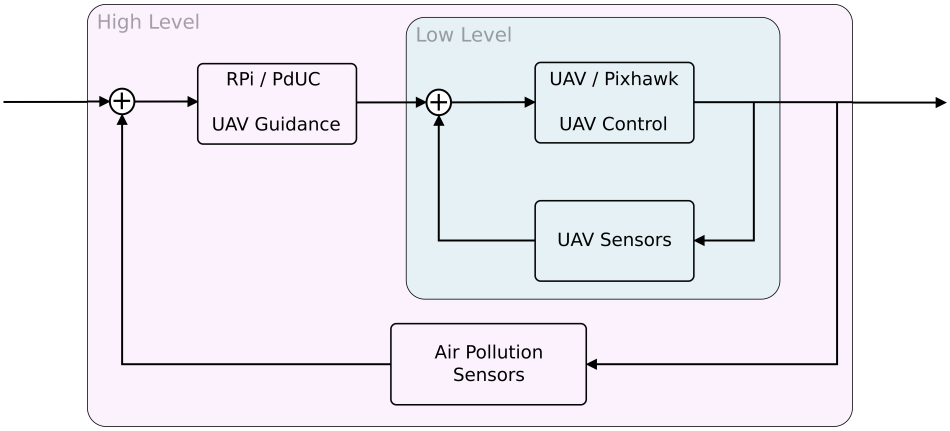


Figure 7.2: UAV control loop.

7.2.2 Information flow analysis

Figure 7.3 illustrates the information flow of the proposed prototype.

To start monitoring a particular area, the UAV route should be introduced through the Graphical User Interface (GUI) of the Android-based application (step 1). The user defines different route points and the initial point, in addition to speed and active sensor settings. Once the Android device receives the route parameters, it sends all entered settings to the Raspberry Pi (step 2) using the Connection Manager module (via Wi-Fi). The Raspberry Pi then configures the UAV with the received guidance parameters through the UAV Guidance module (step 3).

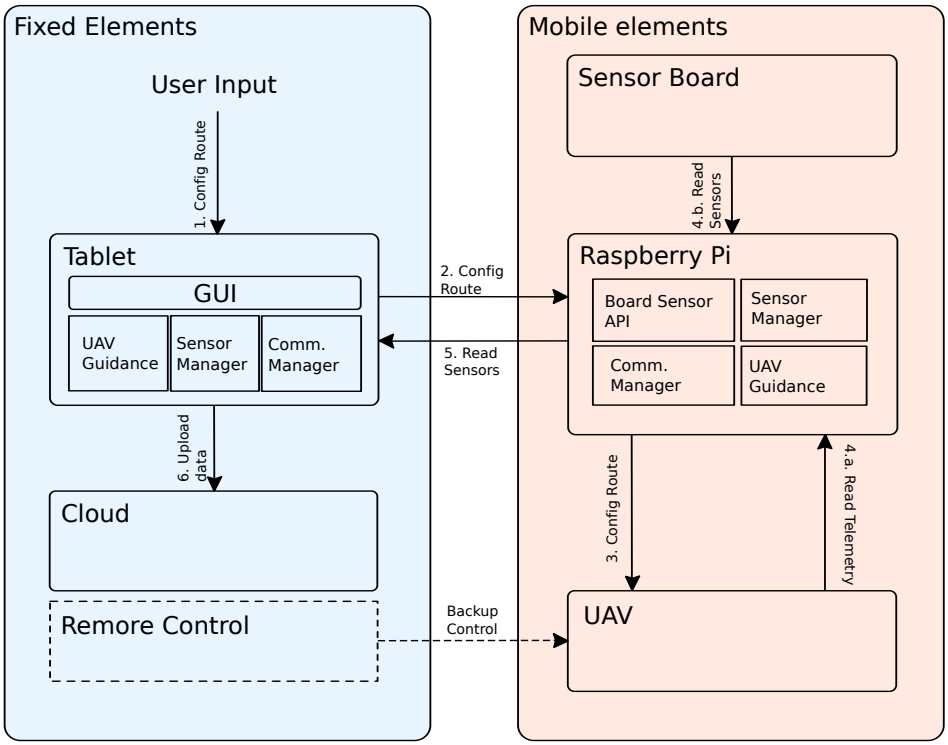


Figure 7.3: Information flow diagram.

During the flight, the Raspberry Pi receives telemetry information from the UAV, again through the UAV Guidance module (step 4.a), and sensor data from the different sensors via the Sensor Manager module, which in turn relies on the Board Sensor API (step 4.b), storing all received information for future diffusion. If at any time during the flight manual intervention is required, the remote control helps to recover the control over the UAV.

When the UAV returns to the origin, all the collected information is sent to the Android device (step 5) using the Connection Manager module (via the Wi-Fi connection). Then, when the smart device has Internet connectivity, either Wi-Fi or Cellular, it sends all the collected information to the Cloud-based server via the Sensor Manager module (step 6).

Finally, an application running on the cloud analyses the information gathered, and offers a detailed report concerning air pollution levels in the target area.

7.3 Summary

In this chapter, we propose equipping UAVs with pollution sensors to create a low-cost mobile air monitoring station that is especially useful in environments with poor ground accessibility.

Specifically, we propose a physical configuration for the UAV-based sensing unit, which integrates a Pixhawk-based controller that is connected to a Raspberry Pi. This way, the Rasberry Pi can act as the brain of the device, being equipped with both sensing and controlling capabilities, thereby allowing to deploy a wide set of applications related with UAV-based sensing.

Chapter 8

PDUC: Pollution-driven UAV Control

In chapter 5 we proposed a physical architecture to enhance an UAV by endowing it with sensing capabilities. In this chapter, we propose PdUC, a Pollution-driven UAV Control algorithm that allows to automatically monitor a specific area by combining a Chemotaxis metaheuristic with Particle Swarm Optimization and Adaptive Spiral mobility.

We show that, by using our approach applied to UAV path control, it is possible to achieve faster and more accurate estimations about the location of the most polluted areas with respect to classical area-search approaches. Our analysis also takes into account uncertainty-based considerations in the sensor sampling operations.

8.1 UAV mobility control analysis

Focusing on our topic, despite the presence of several works related to air pollution monitoring using Unmanned Aerial System (UAS), the majority of these works involve, mainly, swarm creation or communication interaction between them. An example of such work is [125], where authors propose a mobility model for a group of nodes following "Virtual Tracks" (highways, valley, etc.) operating in a predefined "Switch Station" mode, where nodes can split or merge with another group of nodes.

If we analyze works related to mobility models for UAS mobility control that could be used for air pollution monitoring tasks, we observe that, basically, no

previous work focuses on the coverage improvement for a certain area.

For instance, in [20], authors propose a mobility model based on the Enhanced Gauss-Markov model to eliminate or limit the sudden stops and sharp turns that the random waypoint mobility model typically creates. Also, in [117], authors present a semi-random circular movement (SRCM) based model. They analyze the coverage and network connectivity by comparing results against a random waypoint mobility model.

The authors of [89] compare their models against random waypoint-based, Markov-based, and Brownian-motion-based algorithms to cover a specific area, analyzing the influence of the use of collision avoidance systems in the time to achieve a full area coverage. The work in [73] compares the results of using the "Random Mobility Model" and the "Distributed Pheromone Repel Mobility Model" as direction decision engines (next waypoint) in UAV environments. The authors of [116] propose an algorithm to cover a specific area; it selects a point in space along with the line perpendicular to its heading direction, and then it drives the UAV based on geometric considerations.

There are works focusing on using UAVs for specific tasks involving autonomous movements. An example is [111], where authors present a mobility model for the self-deployment of an Aerial Ad Hoc Network in a disaster scenario in order to create a flexible aerial communications infrastructure that victims can rely upon for communication. The mobility model proposed is mainly based on the Jaccard dissimilarity metric to control the deployment of UAVs that create the network. A similar work is presented in [28], where instead an in-network density analysis is used to select the physical areas that need to be visited by a flying robot.

Focusing solely on existing proposals addressing mobility models, we can find works such as [24] where authors propose the Paparazzi Mobility Model (PPRZM) by defining five types of movements - Stay-On, Way-Point, Eight, Scan, and Oval - following a defined state machine with different probabilities to change between states. There are even studies following animal-based navigation patterns. An example of such work is [16], where authors investigate the UAV placement and navigation strategies with the end goal of improving network connectivity, and using local flocking rules that aerial living beings like birds and insects typically follow.

The use of UAVs for air pollution monitoring in a specific area using multi-rotor drones is, however, still not present in scientific literature, and this work can be seen as one of the first approaches in this direction. Our contribution in this chapter is the deployment of a protocol called PdUC (Pollution-driven UAV Control) to automatically track a target area by focusing on the most polluted regions.

8.2 Autonomous Driving approach

To deploy an algorithm for automatically monitoring a specific area we first analyzed the different existing possibilities that could be useful to our goals.

So, to elaborate the proposed PdUC solution, we have used specific techniques such as the metaheuristics and optimization algorithms described below.

Chemotaxis meta-heuristic

The use of rotary-wing UAVs, equipped with chemical sensors and tasked to survey large areas, could follow chemotactic [25] mobility patterns, since their flight behavior could easily implement the following two-phase algorithm: first, read a pollution concentration while hovering; next, follow a chemotactic step.

Chemotaxis meta-heuristics are based on bacteria movement. In this model, the microorganisms react to a chemical stimulus by moving towards areas with a higher concentration of some components (e.g. food) or moving away from others (e.g. poison). In our system, we have considered the following adaptation of the chemotaxis. Let us consider an agent i moving on a Euclidean plane, located at position $\overrightarrow{P}^i_j$ from an absolute reference axis, and moving along time in sequential steps j. For every chemotactic step, a new position $\overrightarrow{P}^i_j$ is calculated based on the previous one, defined by x^i_{j-1} and y^i_{j-1}, plus a step size d^i applying a random direction θ^i_j, as specified in equation 8.1.

$$\overrightarrow{P^i_j} = \begin{pmatrix} x^i_{j-1} \\ y^i_{j-1} \end{pmatrix} + \begin{pmatrix} d^i \times \cos(\theta^i_j) \\ d^i \times \sin(\theta^i_j) \end{pmatrix} \tag{8.1}$$

$$\theta^i_j = \begin{cases} \theta^i_{j-1} + \alpha^i_j & p^i_j \geq p^i_{j-1} \\ -\theta^i_{j-1} + \beta^i_j & p^i_j < p^i_{j-1} \end{cases} \tag{8.2}$$

The direction θ^i_j, as shown in equation 8.2, is calculated on the basis of the concentration value of a certain chemical component, sampled by an agent i at step j: p^i_j. With respect to the previously sampled value p^i_{j-1}, the following two types of movements are contemplated: *Run* and *Tumble*. In the former, Run, when the component concentration is increased with respect to the previous sample, the movement continues to follow the same direction as before (θ^i_{j-1}), plus a random angle α^i_j. Regarding the latter, Tumble, when the concentration is decreasing, the movement takes a turn in the opposite direction $-\theta^i_{j-1}$, plus a random angle β^i_j. Notice that both α^i_j and β^i_j are used to introduce variability and to maximize the gradient, allowing to reach the most polluted areas faster.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a technique introduced in [44] where a solution to a problem is represented as a particle p^i moving in a D-dimensional space at a time t; each particle p^i maintains its position p_t^i, and its best performance position p_b^i. To determine the next position p_{t+1}^i, PSO calculates the stochastic adjustment in the direction of the previous best local position of i's p_b^i element, along with the general best position of any element p_b^g, as shown in equation 8.3:

$$p_{t+1}^i = \alpha \cdot p_t^i + U(0,\beta) \cdot (p_b^i - p_t^i) + U(0,\beta) \cdot (p_b^g - p_t^i) \tag{8.3}$$

where α and β are constants used to calibrate the algorithm, and $U(0,\beta)$ is a random number in the range $[0,\beta]$.

8.3 Proposed Autonomic Solution

To consistently drive the UAVs, so as to achieve the desired area coverage goals, we have devised the following algorithm, which incorporates a chemotactic approach.

8.3.1 PdUC Algorithm

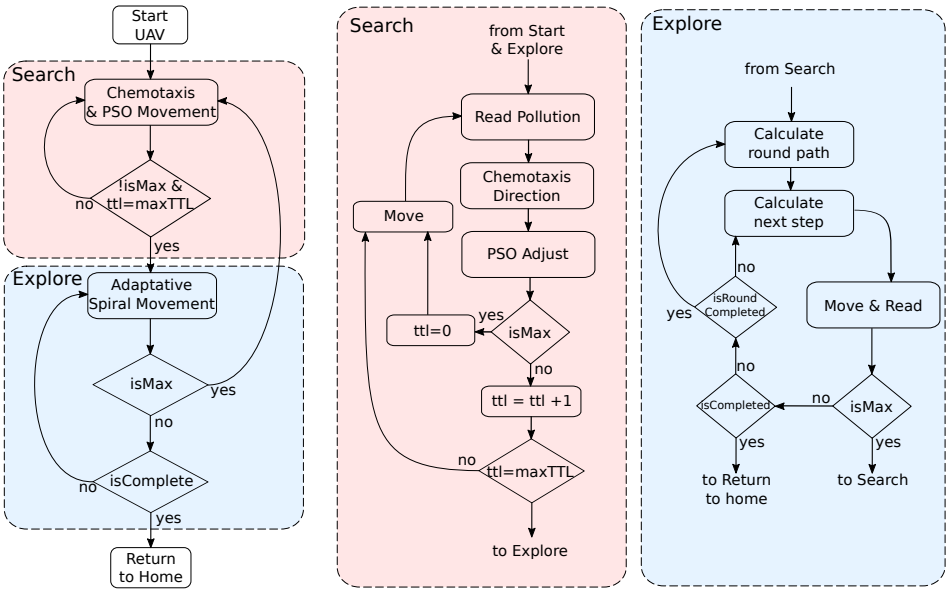


Figure 8.1: Overview of different states associated to the PdUC algorithm.

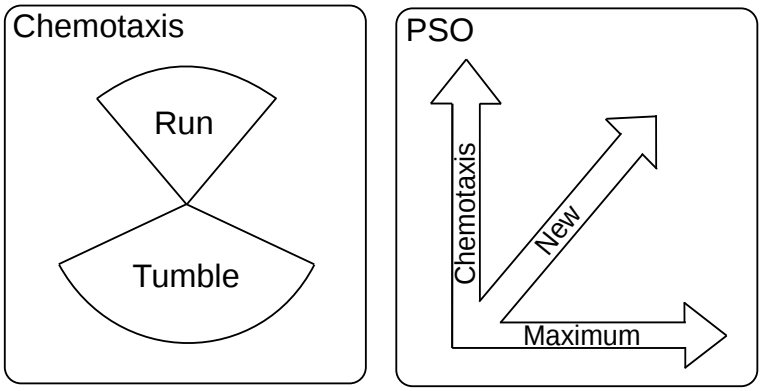


Figure 8.2: PdUC algorithm: calculation of a new direction.

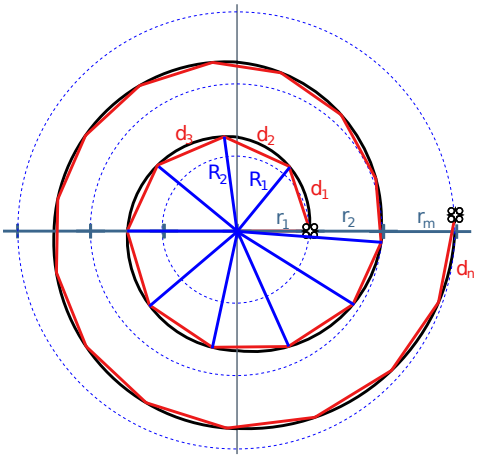


Figure 8.3: PdUC algorithm: exploration phase.

In this context, we have developed an algorithm called Pollution-driven UAV Control (PdUC), based on the chemotaxis metaheuristic concept, to search an area for the highest pollution concentration levels. Once this pollution hotspot is found, the flying drone covers the whole area by following a spiral movement, starting from the most polluted location.

Our algorithm is composed of two phases: (i) A search phase, in which the UAV searches for a globally maximum pollution value, and (ii) An exploration phase, where the UAV explores the surrounding area, following a spiral movement, until one of the following conditions occurs: it covers the whole area, the allowed flight time ends, or it finds another maximum value, in which case it returns to the

search phase, as shown in Figure 8.1.

The exploration phase is mainly based on two previously described techniques: a chemotaxis metaheuristic, and a local Particle Swarm Optimization algorithm. As detailed in Algorithm 1, initially, before the UAV starts its first movement, it samples the pollution value and inserts it in a buffer. For each chemotactic step, it starts to hover, collects another sample, and compares it with the previous one. If the sampling variation is positive (increasing), the UAV follows a "Run" chemotaxis direction, with a random α_j^i of $[-30, 30]$ degrees. Otherwise, if the sampling variation is decreasing, the UAV calculates the "Tumble" chemotaxis direction in the reverse orientation with a random β_j^i of $[-150, 150]$ degrees, although modified by the actual maximum value reached (md^i), as shown in figure 8.2. Equation 8.4 denotes the formula to calculate the new direction, and γ specifies the weight of the md^i, which must be between 0 and 1.

Algorithm 1 PdUC Search Phase.

```
1:  while isSearching do
2:      p_2^pollution ← CurrentPollution(p_2)
3:      ∇poll ← p_2^pollution − p_1^pollution
4:      p_1 ← p_2
5:      if ∇poll > 0 then
6:          ttl ← 0
7:          p_2 ← Run(p_1)
8:          p_max ← p_2
9:      else
10:         p_2 ← Tumble(p_1)
11:         ttl ← ttl + 1
12:     p_2 ← AdjustPSO(p_2, p_max)
13:     if isInsideArea(p_2) then
14:         MoveTo(p_2)
15:     else
16:         p_2 ← Tumble(p_1)
17:     if ttl > ttl_max then
18:         isSearching ← false
19:         isExploring ← true
20: end
```

To determine when our PdUC has found a maximum local value, we use a TTL (Time-to-live) counter. When PdUC finds a maximum value, the TTL is reset and then it starts increasing until a new maximum pollution value is found again, or until the maximum TTL value is reached. In this case, PdUC reverts to the exploration phase since it considers that a new local maximum value has been

found.

$$\theta_j^i = \begin{cases} \theta_{j-1}^i + \alpha_j^i & \text{Run} \\ (1-\gamma)(-\theta_{j-1}^i + \beta_j^i) + \gamma m d^i & \text{Tumble} \end{cases} \tag{8.4}$$

Once a maximum value is reached, the next phase is to explore the surrounding area. As shown in algorithm 2, this is achieved by following an Archimedean spiral similar to the one depicted in Figure 8.3. Starting from the maximum value, it covers the surrounding area by applying a basic step size d_j^i, and changing it depending on the detected pollution variations, a procedure that is similar to the *finding* phase. If the variation is less than a preset value c^i, the step size increases until reaching $3 \times d_j^i$; otherwise, it decreases until d_j^i is reached. If a maximum pollution value is found, PdUC automatically returns to the exploration phase. Finally, once the whole area is covered, the UAV changes to a *return-to-base* (RTB) mode to finish the exploration.

8.3.2 Algorithm optimization

Next, analyzing the overall behavior, we have introduced some modifications to optimize the performance of the proposed PdUC algorithm.

Spiralling with alternating directions

As shown in Figure 8.4, to avoid large steps in the exploration phase when the spiral center is next to a border, the direction of the spiral will alternate for each round to allow minimizing the length of some of the steps. To this purpose, for each spiral round, we calculate the direction adopted as being the opposite direction with reference to the previously used one. The system can get the general size of the area to search, as well as its borders, before starting the mission. This procedure takes place in line 4 of Algorithm 2. In detail, it basically follows equations 8.5 and 8.6:

$$\theta_{s,r} = \begin{cases} \alpha + \beta_{s,c} & \text{if } r \text{ is even} \\ \alpha - \beta_{s,c} & \text{if } r \text{ is odd} \end{cases} \tag{8.5}$$

$$p_{s,r} = \begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} x_c + R_s \times \cos(\theta_{s,r}) \\ y_c + R_s \times \sin(\theta_{s,r}) \end{pmatrix} \tag{8.6}$$

where $\theta_{s,r}$ defines the angle in round r and step s, α is the initial angle, and β_s is the angle in step s. Using the angle $\theta_{s,r}$, the next point p_s is calculated using as reference the coordinates for the spiral center (x_c and y_c) and radius R_s.

Algorithm 2 PdUC Exploration Phase.

1: **while** isExploring **do**
2: $round \leftarrow round + 1$
3: $round_{size} \leftarrow 2\pi \cdot \frac{round + round_{next}}{2}$
4: $round_{direction} \leftarrow -previous_{direction}$
5: $angle_{count} \leftarrow \frac{2\pi}{\frac{round_{size}}{d}}$
6: $step \leftarrow 0$
7: $angle \leftarrow 0$
8: **while** $step < round_{size}$ and $isExploring$ **do**
9: **if** $isInsideArea(p2)$ and $isNotMonitored(p2)$ **then**
10: $p_2^{pollution} \leftarrow CurrentPollution(p_2)$
11: **if** $p2 > p_{max}$ **then**
12: $store(p_{max})$
13: $store(round)$
14: $isExploring = false$
15: $isSearching = true$
16: $p_{max} = p2$
17: **else**
18: $\nabla poll \leftarrow p2_{pollution} - p1_{pollution}$
19: $MoveTo(p_2)$
20: $step \leftarrow step + d$
21: $angle \leftarrow angle + angle_{count} \times round_{direction}$
22: $p1 \leftarrow p2$
23: $p2 \leftarrow NextPoint(p1, angle, step)$
24: $previous_{direction} \leftarrow round_{direction}$
25: **end**

Skipping Previously Monitored Areas

As shown in Figure 8.5, to avoid monitoring the same area multiple times, all samples, which were taken within the monitored area during the exploration phase, are internally stored. For this purpose, PdUC maintains a list containing the location of the central position of all spirals with their respective radius to determine the monitored areas (as a circumference determined by a center and a radius). Next, in the exploration phase, all points inside these circles are omitted for the sake of celerity, as shown in line 9 of Algorithm 2.

8.4 Validation & Simulation

To validate our protocol, we have run several simulations with different configurations implemented in the OMNeT++ simulation tool, as shown in figure 8.6.

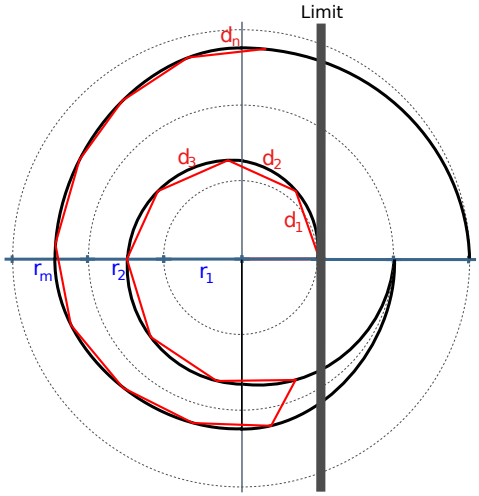


Figure 8.4: PdUC algorithm: alternating spiral direction.

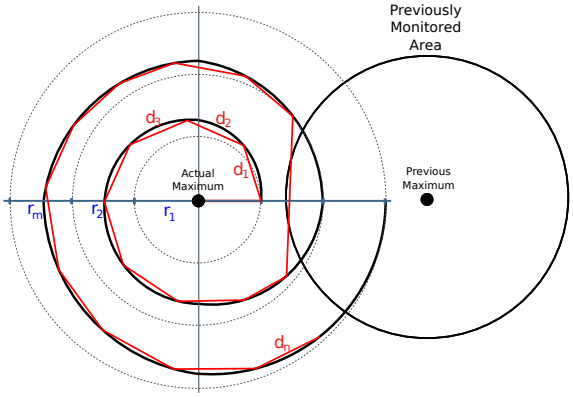


Figure 8.5: PdUC algorithm: skipping monitored areas

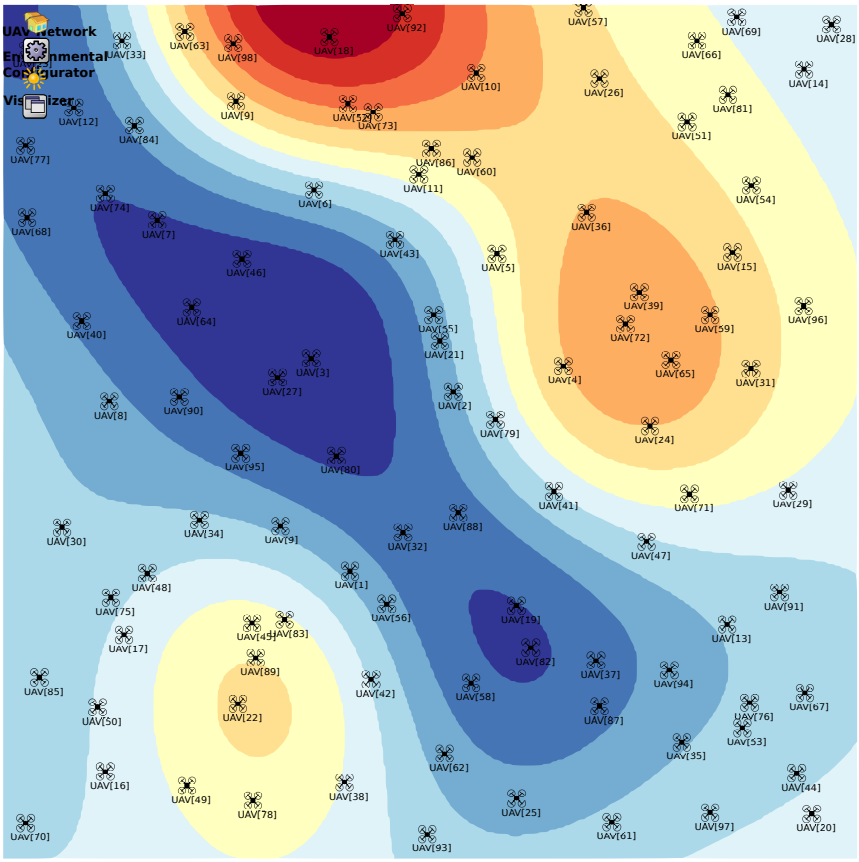


Figure 8.6: Example simulation scenario showing possible initial UAV positions over a randomly generated pollution map.

To prepare a suitable data environment, we have created various synthetic pollution distribution maps representing ozone levels to be used as inputs for testing. These pollution maps were generated using the *R Graph* tool [96], and following a Kriging-based interpolation [108]. In particular, a Gaussian distribution is used to adjust the parameters coming from random data sources of ozone concentration. The actual values range between 40 and 180 ppb, thereby providing a realistic ozone distribution.

Figure 8.7 shows some samples of the created maps, which have the highest pollution concentration (areas in red) located at completely different positions due to the stochastic scenario generation procedure adopted.

Using the previously created data as input, we have run several simulations

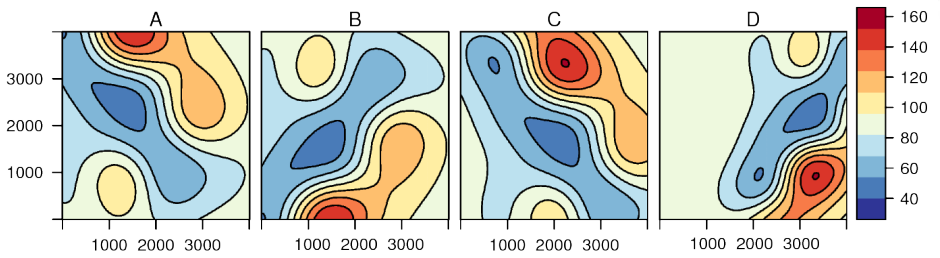


Figure 8.7: Pollution distribution examples used for validation.

using OMNeT++, comparing our protocol against both the Billiard and Spiral mobility patterns. In the simulator, we have created a mobility model implementation of PdUC. In addition, to simulate the sampling process, we have configured OMNeT++ to periodically perform measurements taken from the pollution distribution map defined for the test.

Figure 8.8 shows an example of the path followed by an UAV using the PdUC algorithm as a guidance system. As expected, the UAV starts a search process throughout the scenario until it locates a position with the highest degree of pollution (local maximum). Afterward, it follows a spiral pattern to gain awareness of the surrounding gradients. If, while following the spiral-shaped scan path, it finds a higher pollution value, the algorithm again switches to the search phase. Finally, when the entire target area has been sampled, the algorithm finishes.

To compare the three options under study, we recreate, using the R Graph tool, the pollution distribution maps using the simulation output as the input for the Kriging-based interpolation. In this way, we obtain new pollution maps for comparison against the ones used as reference.

Table 8.1 summarizes the parameters used in the simulations.

Table 8.1: Simulation parameters.

Parameter	**Value**
Area	4x4 Km
Pollution range	[40 - 180] ppb
Sampling error	10 ppb
Max. speed	20 m/s
Sampling time	4 seconds
Step distance	100 m
Mobility models	Billiard, Spiral and PdUC

Since we are proposing the PdUC algorithm for rural environments, the simulation area defined is sized 4×4 Km. As indicated above, the pollution distribution

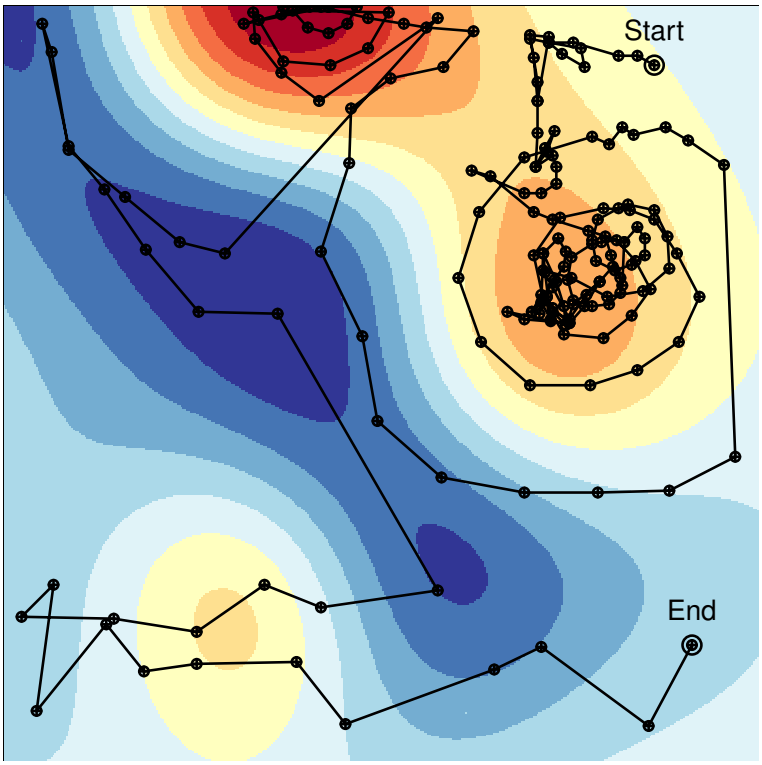


Figure 8.8: Example of an UAV path when adopting the PdUC mobility model.

relies on synthetic maps that are generated by combining a random Kriging interpolation following a Gaussian model, with values between 40 and 180 units, based on the Air Quality Index (AQI) [2]. Since samples are taken using off-the-shelf sensors, which are not precise, we introduce a random sampling error of ± 10 ppb based on real tests using the MQ131 (Ozone) sensor. In our simulation, we set the maximum UAV speed to 20 m/s, a value achievable by many commercial UAVs. The step distance defined between consecutive samples is 100 meters. Once a new sampling location is reached, the monitoring time per sample is defined equal to 4 seconds.

The mobility models used are Billiard, Spiral, and PdUC. These models have different assumptions regarding the initial UAV position. In the *Billiard* model, the UAV starts in a corner of the target area, and it then covers the whole area by "bouncing" when reaching the borders. The *Spiral* model starts at the center of the area to cover, and it then gradually moves to the periphery of the scenario following a spiral pattern. Finally, PdUC is set to start at a random position

within the target area.

We now proceed by analyzing the time required to cover the entire area using each of the approaches being tested. For this purpose, we defined 100 simulations for each model (Billiard, Spiral, and PdUC), and calculated the required time to cover the whole area, estimating the pollution map afterward.

For each run, the starting position of the UAV is randomly set on the map, as shown in Figure 8.6.

Figure 8.9 shows the Cumulative Distribution Function relative to the time required to cover the whole area for the three mobility models. It can be seen that the Billiard and Spiral models do not depend on the start position, spending a nearly constant time (5600 and 2600 seconds, respectively) for each configuration defined. In the case of the PdUC mobility model, since it reacts to air pollution, the time required to cover the complete area varies between 1800 and 4300 seconds, depending on the start position.

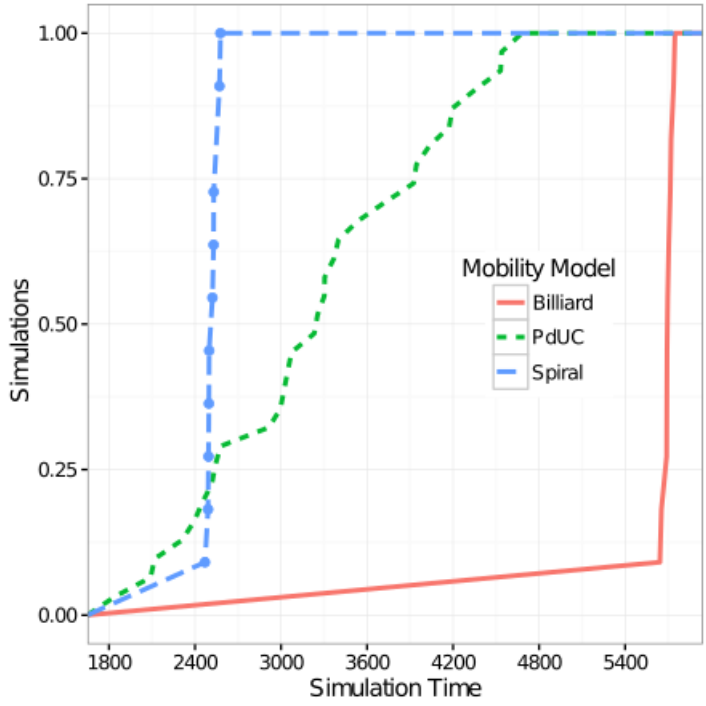


Figure 8.9: Cumulative Distribution Function of the time spent at covering the complete area for the Billiard, Spiral and PdUC mobility models.

Due to battery restrictions, it is interesting to analyze how fast each mobility

model discovers the most polluted areas, and how accurately does it recreate the pollution distribution. For this purpose, we analyze the relative error for the three mobility models at different time instants (600, 1200, 1800, 2400, 3000 and 6000 seconds); this error is defined by equation 8.7:

$$e_t = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \left| \frac{s_{x,y,t} - b_{x,y}}{\triangle b} \right|}{m \cdot n} \tag{8.7}$$

where, e_t is the relative error at time t; $s_{x,y,t}$ is the recreated pollution value at position (x, y) using the samples taken during simulation until time t, $b_{x,y}$ is the reference pollution value at position (x, y), and n and m are the dimensions of the target area, respectively.

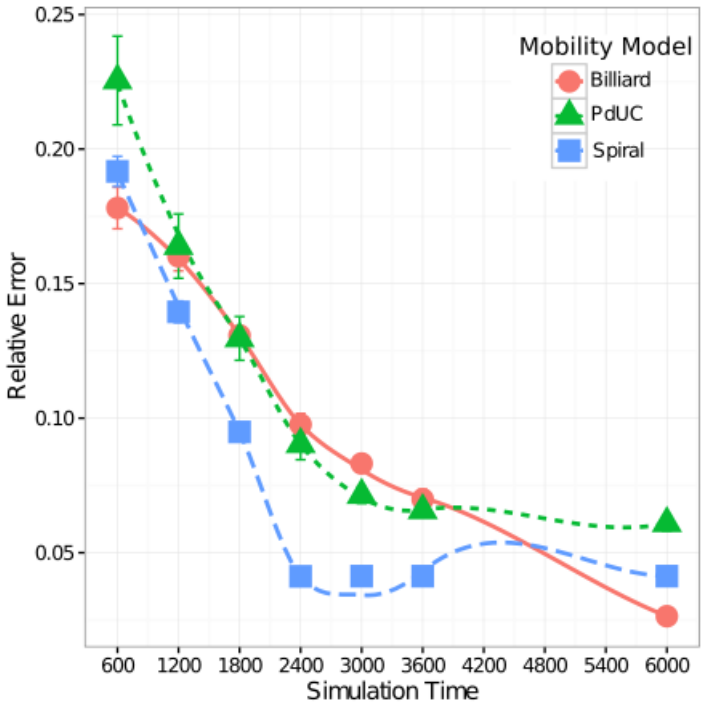


Figure 8.10: Relative error comparison between the PdUC, Billiard, and Spiral mobility models at different times, when analyzing all the values.

Figure 8.10 shows the temporal evolution of the relative error between the three mobility models (Billiard, Spiral, and PdUC) and the original one. We can observe that all mobility models have roughly the same behavior: they start with a high relative error, which is foreseeable since we are using Kriging interpolation

to recreate the pollution distribution, and it tends to the mean value when the number of samples is not enough. Then, as more samples become available, the spatial interpolation process quickly becomes more precise.

Although the three mobility models are similar, the spiral approach achieves a better performance in terms of relative error reduction. However, if we analyze only the most polluted regions, that is, regions characterized by values higher than a certain threshold (120 and 150 ppm in our case, based on AQI [2]), we find that PdUC clearly provides better results.

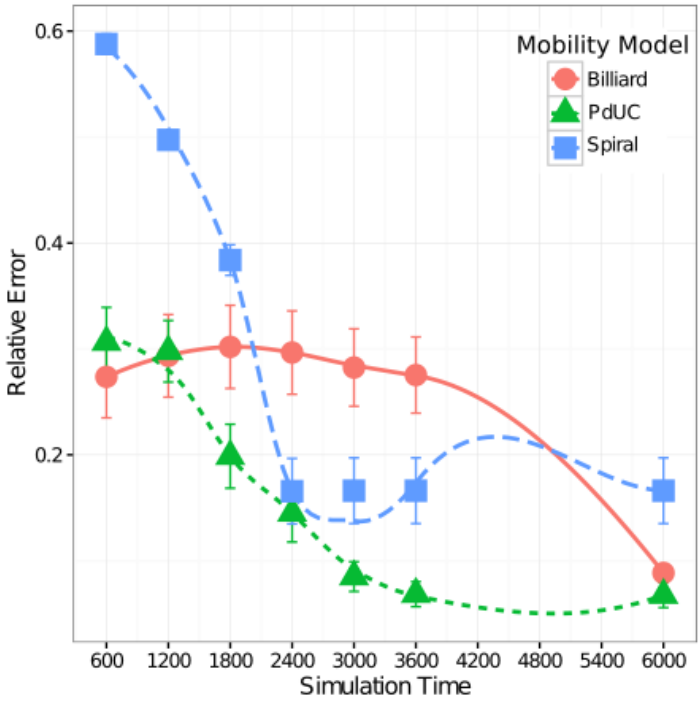
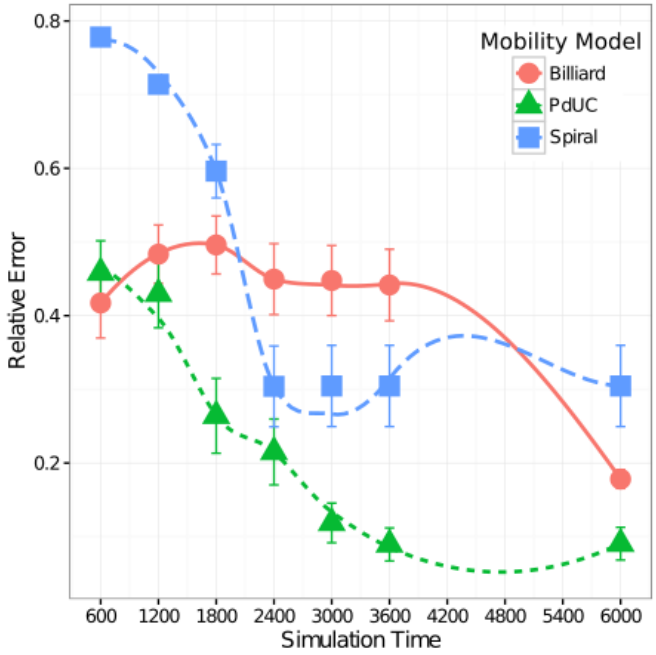


Figure 8.11: Relative error comparison between PdUC, Billiard, and Spiral mobility models at different times when only considering values higher than 120 ppb.

Figures 8.11 and 8.12 show the comparison between the Billiard, Spiral, and PdUC mobility models at different times when only focusing on air pollution values higher than 120 and 150 ppb, respectively. These results show that PdUC clearly provides better results than the Billiard and Spiral movement patterns, outperforming their accuracy from nearly the beginning of the experiment (1200 seconds), reaching the lowest relative error values in just 3600 seconds, with these



5

Figure 8.12: Relative error comparison between PdUC, Billiard, and Spiral mobility models at different times when only considering values higher than 150 ppb.

two other mobility approaches more than doubling the error values in the same scenarios. In particular, the Billiard mobility pattern requires about 6000 seconds to achieve a similar degree of accuracy (120 ppb case), while the Spiral approach is not able to achieve values as low as PdUC in any of the cases. This occurs because PdUC focuses on the highest values in the chemotaxis-based phase. PdUC always prioritizes the most polluted areas in detriment of less polluted ones, thus allowing to obtain, at least, details about the region with the highest pollution values.

To complete our study, figure 8.13 presents an example of the evolution of predicted pollution values for the whole target area, and for the three algorithms under analysis (Spiral, Billiard, and PdUC), at different times (1200s, 2400s, 3600s and 6000s). We can observe that PdUC is able to quickly find the most polluted areas, while the effectiveness of other approaches highly depends on the actual location of pollution hotspots in order to detect them at an early stage.

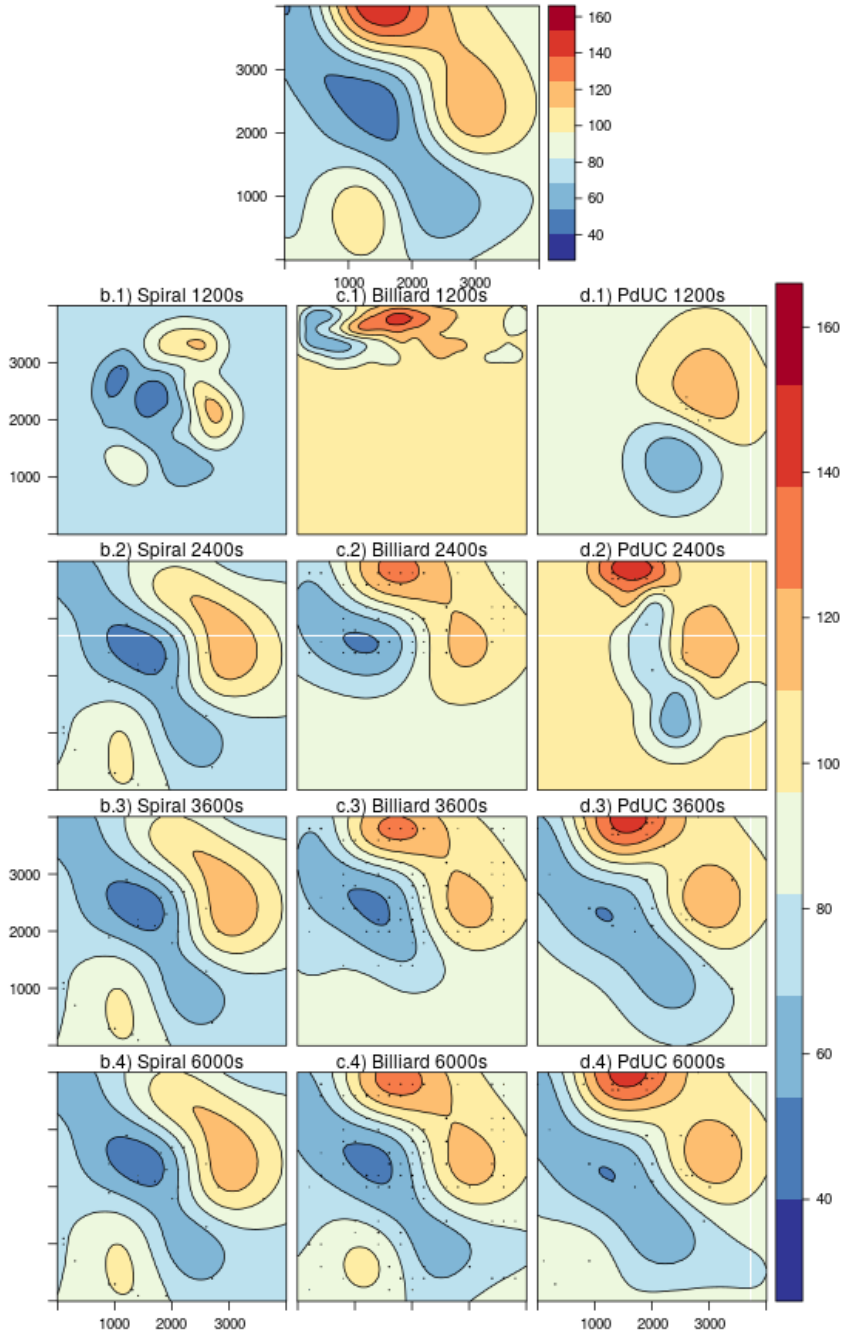


Figure 8.13: Visual representation of the estimation output for the PdUC, Billiard, and Spiral mobility models at different times.

8.5 Summary

In this chapter, we proposed PdUC (Pollution-driven UAV Control), an algorithm to guide an UAV monitoring a specific area by focusing on the most polluted zones.

PdUC is based on a Chemotaxis metaheuristic, a local Particle Swarm Optimization (PSO), and Adaptive Spiral concepts, creating an algorithm able to quickly find areas with high pollution values, and to cover the surrounding area as well, thereby obtaining a complete and detailed pollution map.

To validate our proposal, we compared our approach against the Billiard and Spiral mobility models through simulations implemented in OMNeT++. Simulation experiments show that PdUC offers significantly better performance at reducing prediction errors, especially concerning high-value pollution range.

Chapter 9

PdUC-D: Discretized Pollution-driven UAV Control

When focusing on UAV control systems for air pollution monitoring tasks, we have found that there are no systems optimized for such purposes. So, in chapter 8 we proposed PdUC, a solution that puts the focus on the most polluted regions by combining a chemotaxis metaheuristic with adaptive spiral mobility patterns to automatically track pollution sources and surrounding pollution values in a given target area. We showed that PdUC achieves better performance than standard mobility approaches, like the Spiral and the Billiard patterns, in terms of discovering the most polluted areas in a shorter time span. In this chapter, we extend PdUC by proposing an optimized algorithm, called PdUC-D, that applies space discretization to substantially reduce the convergence time while achieving levels of accuracy similar to those of PdUC.

9.1 PdUC-D: Optimizing the PdUC protocol through discretization

Despite PdUC is more effective than other mobility patterns (Spiral and Billiard) in terms of polluted area monitoring times, finding the most highly polluted locations earlier, as shown in chapter8, PdUC still spends too much time focusing on small variations (e.g. produced by sensor errors, or minimal pollution variations) in nearby areas, which are not useful when obtaining the global pollution map; on the contrary, the Spiral and Billiard models present simpler mobility patterns that, by themselves, avoid such redundant sampling. So, in this work, we attempt

Figure 9.1: Example of a discretized area, calculating the tiles and their center to restrict movements.

to avoid redundant movements (sampling) by discretizing the target area, dividing it into small tiles.

The main idea is to optimize PdUC by discretizing the whole target area. Specifically, we will create a grid composed of small tiles. Notice that discretization is one of the most efficient mathematical approaches to optimize a system by transforming a continuous domain into its discrete counterpart [47], as shown in Figure 9.1. This way, the UAV can only move to the center of each tile, and each tile can only be monitored once, thereby reducing redundant sampling, which in turn reduces the full coverage time significantly.

PdUC-D, just like PdUC, combines a *chemotaxis* metaheuristic with an adaptive spiral, the difference being that both these mechanisms are adapted to operate within discretized space environments. Therefore, PdUC-D starts by first search-

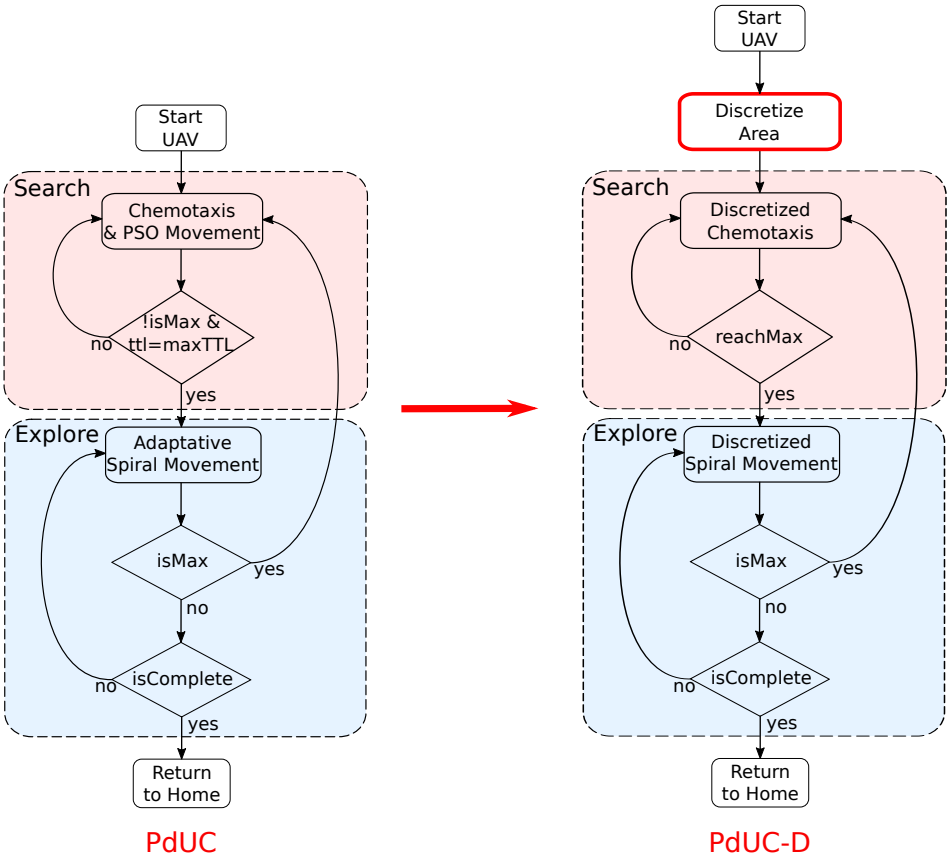


Figure 9.2: Differences between the PdUC and the PdUC-D algorithms.

ing the tile with the highest pollution level (Search phase). Next, it covers the surrounding area by following an adaptive spiral until all the area is covered, or until it can find another tile with a higher pollution value (Explore phase), thereby switching back to the Search phase.

We modify the PdUC phases by adapting its functionality to a discretized space, as shown in Figure 9.2. So, the first step involves splitting the target area into small tiles and calculating the center positions of these tiles (actual locations where monitoring takes place). Next, the search and the explore phase are modified to operate in the discretized space as well.

The search phase is based on a chemotaxis mobility pattern and an adaptation of the Particle Swarm Optimization (PSO) [71] algorithm. Figure 9.3 graphically shows the changes introduced for the chemotaxis and the PSO algorithm. Re-

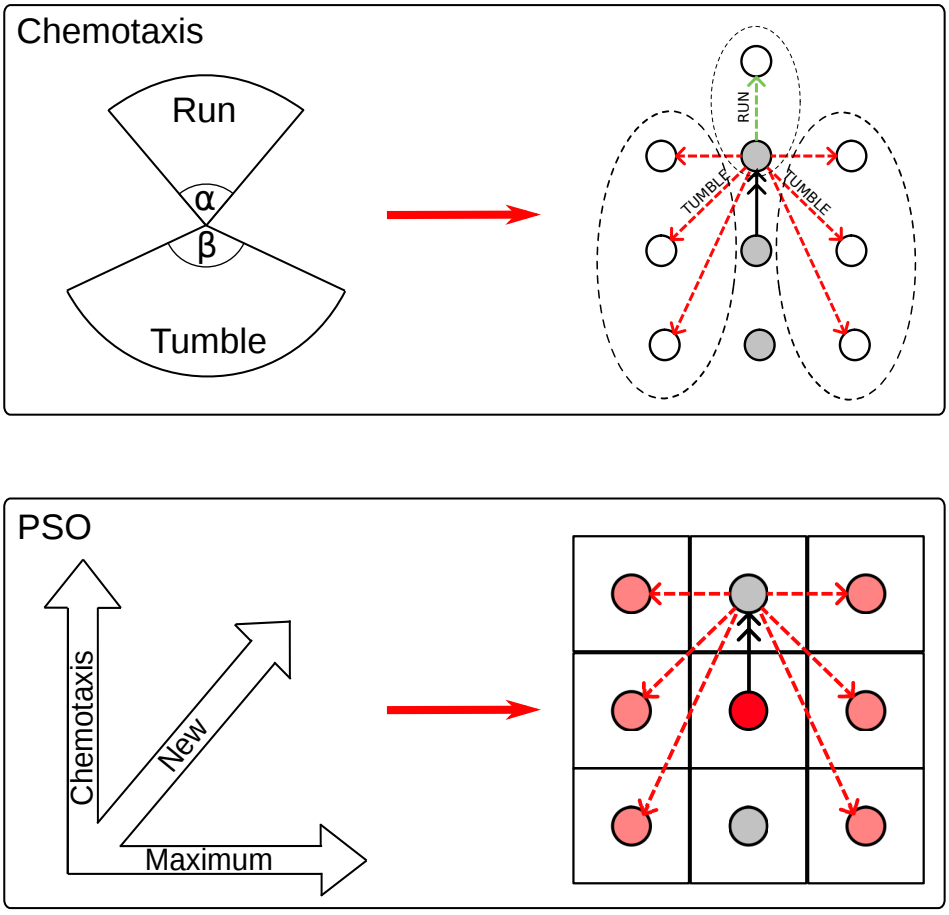


Figure 9.3: Differences between the PdUC to PdUC-D algorithms regarding the search phase: Chemotaxis (top) and PSO (bottom).

garding the chemotaxis-based movement, a particle moving in an euclidean plane between two tiles, and following a specific direction, moves towards the next tile in the same direction (*Run* move) if the pollution variation is increasing along its path. Otherwise, if the pollution variation is decreasing, it moves around the tile with higher previously monitored pollution values, assigning higher priority to nearer tiles (*Tumble* move); namely, it chooses the nearest tile. The procedure of moving around the maximum monitored value is an adaptation of the PSO,

which takes the maximum value into account. If all tiles around the one with the highest detected value have already been monitored, the algorithm switches to the *Explore* phase, just like PdUC does.

The *Explore* phase is based on an adaptive spiral modified to a discretized space environment, as shown in figure 9.4. The three main types of movements involved are: first (see figure 9.4, top), starting at the tile with the highest monitored pollution value, it follows a square spiral. For each round in the spiral, it skips an increasing number of tiles. Namely, in the first round, it has a radius of 3 tiles and it skips 1 tile; in the second round, it has a radius of 5 tiles and skips 2 tiles, and so on. Next, to avoid excessively long steps, if the spiral radius reaches a scenario border, or previously monitored areas, the direction of the spiral is changed by alternating the movement direction so as to rotate in the opposite direction, as shown figure 9.4 (middle). Finally, for controlling previously monitored areas (see figure 9.4, bottom), we consider as an already monitored area the whole square created at the end of each spiral round.

With regard to movement control, and to avoid re-visiting previously monitored areas, we use two matrixes - $P_{m,n}$ and $B_{m,n}$ - to store the sampled values and the monitored tiles, respectively. Notice that $n \times m$ represents the size of the grid (rows and columns, respectively).

$$P_{m,n} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,n} \end{pmatrix}$$

$$B_{m,n} = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,n} \end{pmatrix}$$

First, both matrices are initialized, P with NaN (null), and B with 0's. In the *search* phase, when monitoring a tile $t_{i,j}$ (i and j are the row and the column position, respectively), the pollution values are stored in $P_{i,j}$, and $B_{i,j}$ values are set to 1. When monitoring a tile either in the *explore* phase or in the *search* phase, both P and B values are stored. However, when completing a spiral round, all tiles inside the square are set as visited in B, thereby avoiding to monitor the same area again in the future.

9.2 Validation

We have implemented PdUC-D in the *R Graph* tool [96], and we have performed a wide set of simulations with different configurations. Figure 9.5 shows a screenshot

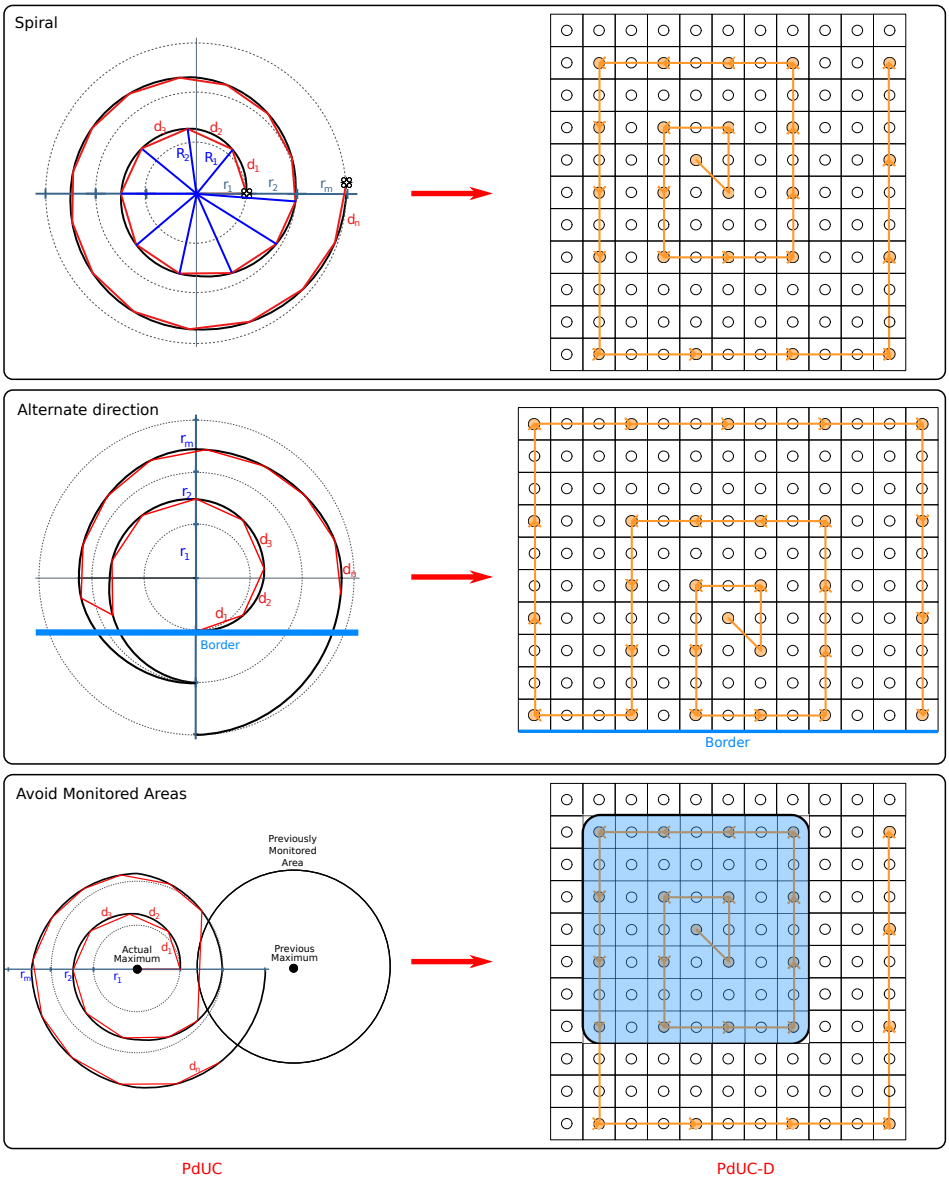


Figure 9.4: Differences between PdUC and PdUC-D algorithms in the Explore phase: Adaptive Spiral (top), Alternate direction (middle), and Avoidance of previously monitored areas (bottom).

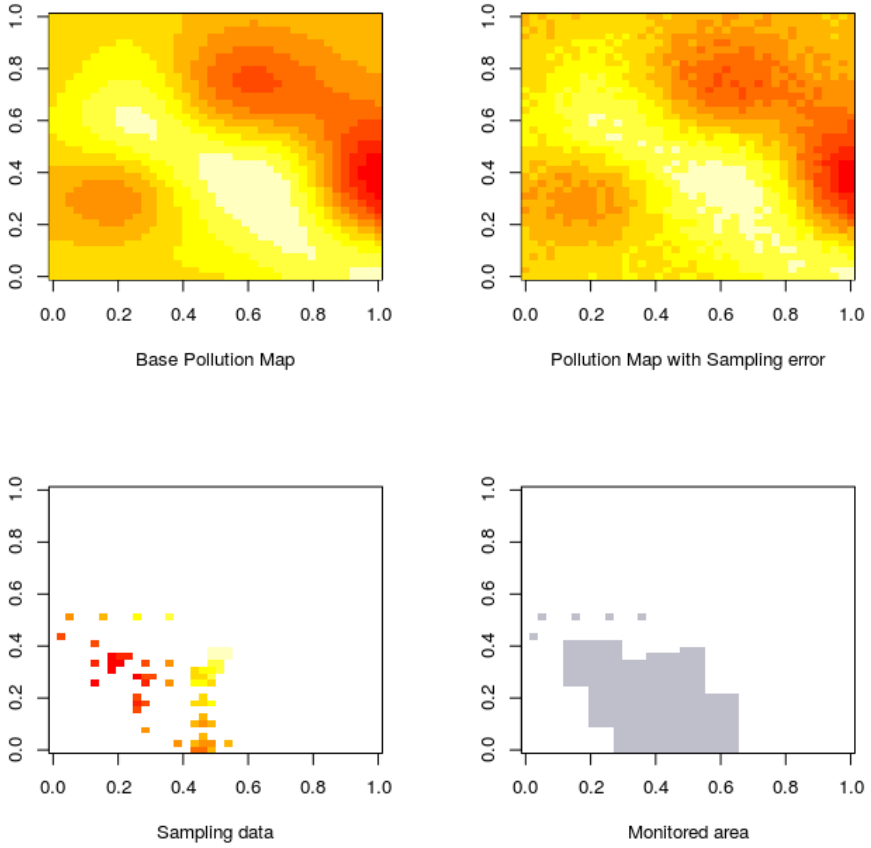


Figure 9.5: Screenshot of the different elements involved in the R implementation of PdUC-D: Initial pollution map (top-left), Pollution map after introducing sampling errors (top-right), Sampled data/P Matrix (bottom-left), and Area considered as already monitored/B matrix (bottom-right). The values on both axes correspond to the ratio with respect to the total area (0 to 1).

of the simulation script output, which includes four images: the base pollution map created based on kriging interpolation, the pollution map created when introducing a random sampling error of 10 ppb (parts-per-billion) for each point, the sampled data (P matrix values), and the areas marked as already monitored (B matrix values).

To prepare a suitable data environment, we have created various synthetic pollution distribution maps representing ozone levels to be used as inputs for

testing. These pollution maps were also generated using the *R Graph* tool following Kriging-based interpolation [108]. In particular, a Gaussian distribution is used to adjust the parameters coming from random data sources of ozone concentration. The actual values range between 40 and 180 ppb, thereby providing a realistic ozone distribution.

Obtained data using PdUC-D was compared against previous results obtained using PdUC [5]. Figure 9.6 shows an example of the path followed by an UAV using (a) PdUC and (b) PdUC-D as a guidance system. As expected, both algorithms have, in general, a similar behaviour: the UAV starts a search process throughout the scenario until it locates a position with the highest degree of pollution (local maximum). Afterward, it follows a spiral pattern to gain awareness of the surrounding gradients. If, while following the spiral-shaped scan path, it finds a higher pollution value, the algorithm again switches to the search phase. Finally, when the entire target area has been sampled, the algorithm finishes. When adopting PdUC-D, though, we can clearly see that it achieves better performance by reducing the monitoring time while avoiding redundant sampling.

To analyze PdUC-D, we used the same simulation parameters as the ones adopted for validating PdUC [8]. Table 9.1 summarizes the parameters used in the simulations.

Table 9.1: Simulation parameters.

Parameter	**Value**
Area	4x4 Km
Pollution range	[40 - 180] ppb
Sampling error	10 ppb
Max. speed	20 m/s
Sampling time	4 seconds
Step distance	50m, 100m, 200m, and 400m

Similarly to PdUC, we are proposing the PdUC-D algorithm for rural environments, and so the simulation area defined is quite large: 4×4 Km. The pollution distribution relies on synthetic maps that are generated through a random Kriging interpolation following a Gaussian model with values between 40 and 180 units, which are representative of different realistic conditions according to the Air Quality Index (AQI) [2]. Since samples are taken using off-the-shelf sensors, which are not precise, we introduce a random sampling error of $\pm$10ppb based on real tests using the MQ131 (Ozone) sensor [103]. In our simulation, we set the maximum UAV speed to 20 m/s, a value achievable by many commercial UAVs. The step distance defined between consecutive samples is 100 m since it offers a good trade-off between granularity and flight time. Once a new sampling location is reached, the monitoring time per sample is defined to be 4 seconds.

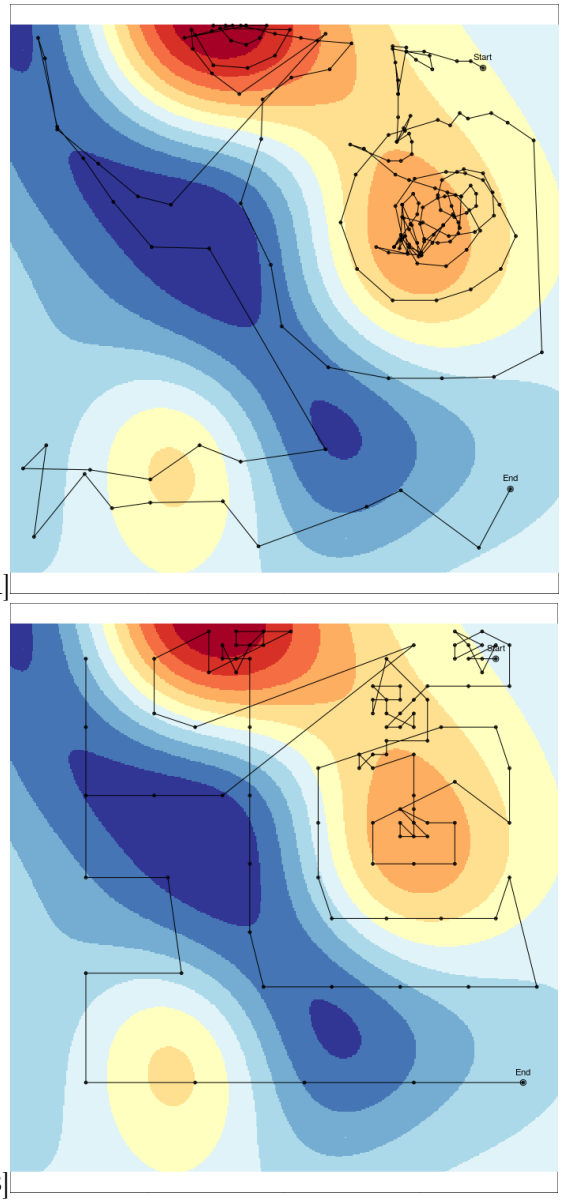


Figure 9.6: Example of a path followed by an UAV guided by the PdUC (A) and PdUC-D (B) protocols.

First of all, we analyze the impact of varying the tile size between 50m and 400m. Our goal is to determine which is the best size considering our restrictions.

Figure 9.7 shows the Cumulative Distribution Function (CDF) relative to the flight time required to cover the whole area for four tile sizes (50m, 100m, 200m, and 400m). It can be seen that, as expected, the smaller the tile size, the higher the coverage time, reaching values that range from 2400 to 4600 seconds for a tile size of 50m, while for tiles size of 200m and 400m these coverage times are in the range from 800 to 1500 seconds. For a tile size of 100m, the coverage time moves between 1500 and 2400 seconds, which represents 40 minutes of flight time. Such flight time is realistic, being achieved by many of the current UAVs that are commercially available.

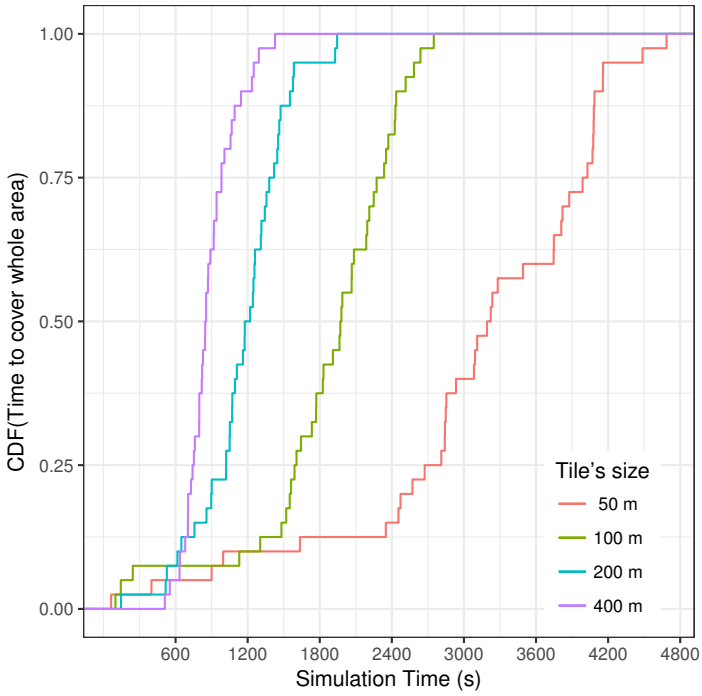


Figure 9.7: Cumulative Distribution Function of the time spent by PdUC-D at covering the complete area for different tile sizes: 50, 100, 200, and 400 meters, respectively.

To gain further insight into the goodness of the proposed algorithm, we also analyze the relative error for all cases at different time instants (600, 1200, 1800,

2400, 3000, and 6000 seconds); this error is defined by equation 9.1:

$$e_t = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|\frac{s_{x,y,t}-b_{x,y}}{\triangle b}\right|}{m \cdot n} \tag{9.1}$$

where, e_t is the relative error at time t, $s_{x,y,t}$ is the recreated pollution value at position (x, y) using the samples taken during simulation until time t, $b_{x,y}$ is the reference pollution value at position (x, y), and n and m are the dimensions of the target area, respectively.

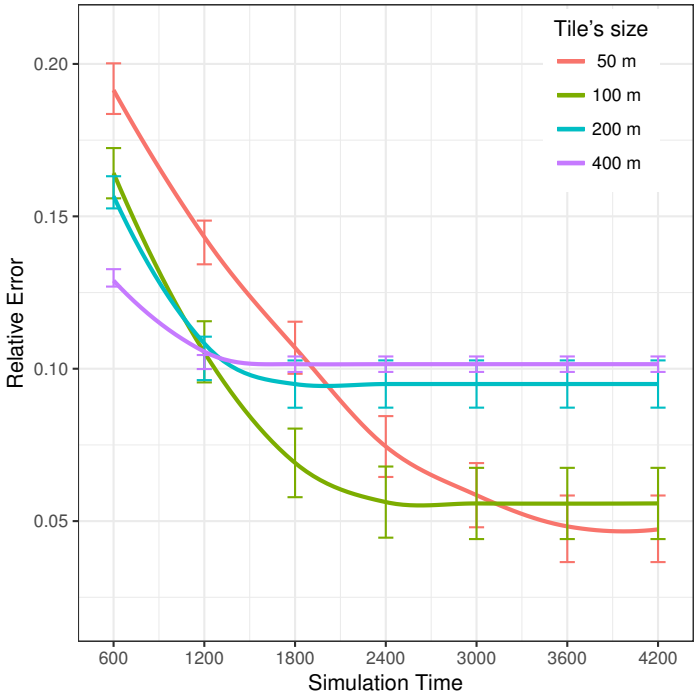


Figure 9.8: Relative error comparison for PdUC-D when adopting different tile sizes: 50, 100, 200, and 400 meters, respectively.

Figure 9.8 shows the temporal evolution of the relative error for different tile sizes (50m, 100m, 200m, and 400m), as well as the reference values. We can observe that, even though at the end a smaller relative error (4.8%) is achieved for a tile size of 50m, the time to reach this value is too long (more than 4000 seconds). On the other hand, in the 200m and 400m cases, they reach their minimum relative error faster, but it can be considered too high (almost 10%) when compared with the other cases. In the 100m case, although the final relative error is only a bit

higher (almost 6%) than the 50m case (4.8%), the time to reach this error is still manageable. For these reasons, we consider that the tile size offering the best trade-off between flight time and accuracy is 100 meters.

To further emphasize on the benefits of using PdUC-D, we now proceed to compare it against the PdUC, Spiral, and Billiard [8] strategies. We use the same simulation parameters as defined above, and we adopt the optimum calculated tile size (100 meters).

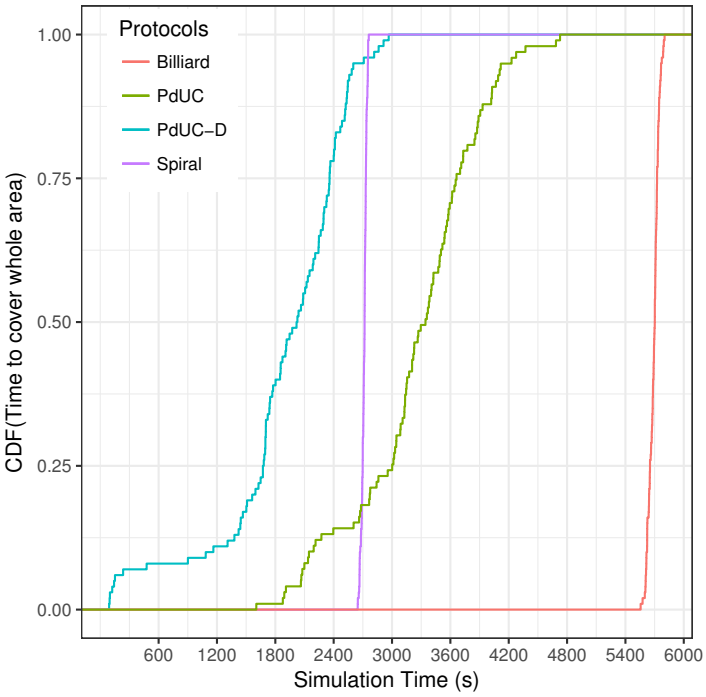


Figure 9.9: Cumulative Distribution Function (CDF) of the time spent at covering the complete area for the PdUC, PdUC-D, Billiard and Spiral mobility models.

Figure 9.9 shows the Cumulative Distribution Function relative to the time required to cover the whole area for PdUC, Billiard, Spiral, and PdUC-D mobility models. It can be seen that the PdUC-D model spends much less time (1500-3000 seconds) than the PdUC model (1800-4300 seconds) to achieve the same goal. Moreover, it spends less time than the Spiral approach in nearly all cases, and it clearly outperforms the Billiard mobility pattern.

Figure 9.10 shows the temporal evolution of the relative error between the model-based predictions (using PdUC, Spiral, Billiard, and PdUC-D) and the

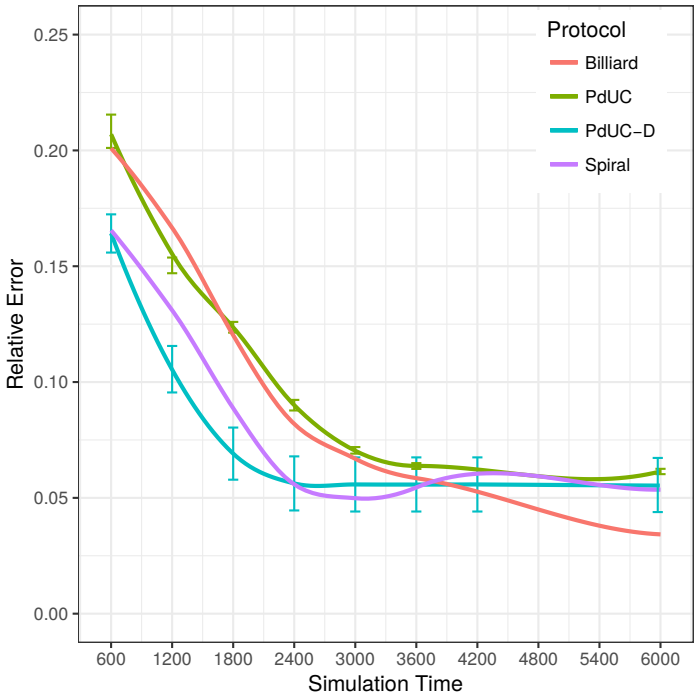


Figure 9.10: Relative error comparison between the PdUC, PdUC-D, Billiard and Spiral mobility models at different times.

reference values. We can observe that all mobility models roughly show the same behavior: they start with a high relative error, which is foreseeable since we are using Kriging interpolation to recreate the pollution distribution, and it gradually decreases towards the mean error value as the number of samples increases. Then, as more and more samples become available, the spatial interpolation process quickly becomes more precise. Moreover, we can observe that, even in this analysis, PdUC-D still obtains better results than the other approaches by significantly reducing the relative error at different times.

9.3 Summary

In this chapter we described PdUC-D (Discretized Pollution-driven UAV Control), an algorithm for air pollution monitoring tasks that improves upon our previous proposal (PdUC). In particular, it operates as an UAV guidance system to move towards the most polluted areas, creating pollution maps for the surrounding area afterward. PdUC-D is based in the Chemotaxis and Adaptive Spiral principles, but

its functionality was modified to work in a space-discretized area, thereby making it much more optimal in terms of coverage time and by reducing prediction errors.

We have analyzed the optimum tile size taking onto account our flight time restrictions, and compared four tile sizes (50m, 100m, 200m, and 400m), finding that a tile sized 100×100 m is the best option.

We have compared PdUC-D against PdUC, as well as against standard mobility models (Billiard and Spiral), by creating several simulations using the R Tool, and comparing these results with the previously obtained ones. Experimental results show that PdUC-D outperforms PdUC in all aspects, reducing the time to cover a same area, and reducing the monitoring error.

Part IV

Conclusions

Conclusions, Publications and Future Work

Air Pollution monitoring has become a fundamental requirement for cities worldwide, and there are many studies related to it. Commonly, air pollution monitoring relies on fixed monitoring stations to carry out the pollution control. However, this method is too expensive and hard to implement in any city. On the other hand, the fast growth of Smart Cities has brought a set of technologies to improve different aspects of urban environments.

Based on Smart City technologies, in this thesis we have proposed an integral solution to monitor air pollution in both rural and urban areas considering their differences, using a Mobile Crowdsensing (MCS) approach focusing on smartphones as the fundamental piece of our architecture.

In urban areas, we deploy a platform to monitor the city using a mobile sensor installed on common vehicles such as bicycles or the public transport system, integrated with a smartphone to accelerate the retrieval of samples and their uploading to the cloud, and a cloud-based system for data processing.

Moreover, analyzing the requirements of the rural areas, we consider the use of UAVs to monitor the air pollution in a specific area. Specifically, we proposed a physical architecture to embed the desired air quality sensor on the UAV and monitor the air pollution. Afterward, we proposed an algorithm to control the UAV, and monitor a certain area in an autonomous manner by focusing on the most polluted areas.

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

- **EcoSensor Platform**. 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 was 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 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 better data granularity.

- **Low Cost Sensor Design**. We presented the general design of an off-the-shelf mobile environmental sensor able to cope with air quality monitoring requirements; we explore different hardware options to develop the desired sensing unit using readily available devices, discussing the main technical issues associated with each option.

  Regarding other hardware alternatives, microcomputers like Raspberry Pi, BeagleBone, or Intel Edison are more powerful and flexible by supporting a standard Operating System, thereby allowing to quickly deploy any application. We believe that the Raspberry Pi solution can be the best option for quick prototyping. For more professional solutions, requiring higher processing capacity, the Intel Edison becomes a better option, although imposing a higher overhead in terms of development time. Finally, Arduino becomes an option for very restricted environments.

  Overall, we find that a hardware solution applicable to all IoT contexts, and meeting low size and low power requirements, along with adequate communication interfaces and battery capacity, is still missing, although in years to come many more products are expected.

- **UAV design for Air Pollution Monitoring tasks**. We proposed a solution where we equip an UAV with COTS sensors for monitoring tasks, using a Pixhawk Autopilot for UAV control, and a Raspberry Pi for the sensing and storage of environmental pollution data.

- **Pollution-driven UAV Control (PdUC)**. To automatically analyze pollution values within a target area, we proposed an adaptive algorithm for autonomous navigation called PdUC. This algorithm allows an UAV to autonomously monitor a specific area by prioritizing the most polluted zones. In particular, PdUC combines different concepts including a Chemotaxis metaheuristic, a local Particle Swarm Optimization (PSO), and an Adaptive Spiralling technique, to create an algorithm able to quickly search for hotspots having high pollution values, and to cover the surrounding area as well, thereby obtaining a complete and detailed pollution map of the target region.

  To validate our proposal, we compared the proposed PdUC solution against the Billiard and Spiral mobility models through simulations implemented in OMNeT++. Simulation experiments show that PdUC offers significantly better performance at reducing prediction errors, especially regarding the accuracy achieved for the high-values range.

- **Discretized Pollution-driven UAV Control (PdUC-D)**. We improved the PdUC algorithm by proposing a discretized version called PdUC-D (Discretized Pollution-driven UAV Control). PdUC-D has the same phases as the original PdUC proposal (Search and Explore), and it is based on the same principles (Chemotaxis and Adaptive Spiral), but its functionality was modified to work in a space-discretized area, avoiding redundant sampling, thereby making it much more optimal.

  We have compared PdUC-D against PdUC using the R Tool, and comparing these results with the previously obtained ones. Experimental results show that PdUC-D has much better performance than PdUC in all aspects, reducing the time to cover a same area, and reducing the estimation error.

Having accomplished all our objectives, the original goal of this thesis has been achieved, and so this dissertation can now be concluded. The next section enumerates the publications related to this thesis. The last section of this chapter, refers to some open issues for the future.

Publications

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

International Journals

- Alvear, O., Zamora, W., Calafate, C., Cano, J. C., Manzoni, P. (2016). An architecture offering mobile pollution sensing with high spatial resolution. Journal of Sensors, 2016. **I.F. 2016: 1,704; JCR: Q2 Category**.

- Alvear, O., Zema, N. R., Natalizio, E., Calafate, C. T. (2017). Using UAV-based systems to monitor air pollution in areas with poor accessibility. Journal of Advanced Transportation, 2017. **I.F. 2016: 1,813; JCR: Q2 Category**.

- Alvear, O., Calafate, C. T., Cano, J. C., Manzoni, P. (2018). Crowdsensing in Smart Cities: Overview, Platforms, and Environment Sensing Issues. Sensors, 18(2), 460. **I.F. 2016: 2,677; JCR: Q1 Category**.

- Alvear, O. A., Calafate, C. T., Zema, N. R., Natalizio, E., Hernandez-Orallo, E., Cano, J. C., & Manzoni, P. (2018). A discretized approach to air pollution monitoring using UAV-based sensing. Mobile Networks and Applications, 2018. **I.F. 2016: 3,259; JCR: Q1 Category**.

International Conferences

- Alvear, O., Zamora, W., Calafate, C. T., Cano, J. C., & Manzoni, P. (2016, June). EcoSensor: Monitoring environmental pollution using mobile sensors. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016 IEEE 17th International Symposium on A (pp. 1-6). IEEE. **Core A**.

- Alvear, O., Calafate, C. T., Hernández, E., Cano, J. C., & Manzoni, P. (2015, December). Mobile pollution data sensing using uavs. In Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia (pp. 393-397). ACM. **Core B**.

- Alvear, O., Calafate, C. T., Cano, J. C., & Manzoni, P. (2015, October). Calibrating low-end sensors for ozone monitoring. In International Internet of Things Summit (pp. 251-256). Springer, Cham.

- Alvear, O. A., Zema, N. R., Natalizio, E., & Calafate, C. T. (2017, June). A chemotactic pollution-homing UAV guidance system. In Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International (pp. 2115-2120). IEEE. **Core B**.

- Calafate, C. T., Cicenia, K., Alvear, O., Cano, J. C., & Manzoni, P. (2017, March). Estimating rainfall intensity by using vehicles as sensors. In Wireless Days, 2017 (pp. 21-26). IEEE.

- Alvear, O., Calafate, C. T., Cano, J. C., & Manzoni, P. (2014, June). VEWE: A Vehicle ECU Wireless Emulation Tool Supporting OBD-II Communication and Geopositioning. In International Conference on Ad-Hoc Networks and Wireless (pp. 432-445). Springer, Cham.

- Alvear, O., Calafate, C. T., Cano, J. C., & Manzoni, P. (2015, January). Validation of a vehicle emulation platform supporting OBD-II communications. In Consumer communications and networking conference (CCNC), 2015 12th Annual IEEE (pp. 880-885). IEEE.

- Alvear, O. A., Calafate, C. T., Zema, N. R., Natalizio, E., Hernandez-Orallo, E., Cano, J. C., & Manzoni, P. (2017, November). PdUC-D: A discretized UAV guidance system for air pollution monitoring tasks. In 3rd EAI International Conference on Smart Objects and Technologies for Social Good (GoodTech), 2017.

- Alvear, O. A., Herrera-Tapia, J., Calafate, C. T., Hernandez-Orallo, E., Cano, J. C., & Manzoni, P. (2017, November). Assessing the Impact of Mobility on LoRa Communications. In 3rd EAI International Conference on Interoperability in IoT (InterIoT), 2017.

National Conferences (Spain)

- Alvear, O., Calafate, C. T., Cano, J. C., & Manzoni, P.(2014). Bringing the vehicle to the lab: ECU emulation for quick application development. In XXV Jornadas de Paralelismo, 2014.

- Herrera-Tapia, J., Hernández-Orallo, E., Manzoni, P., Calafate, C. T., Cano, J. C., & Alvear, O. Difusión de mensajes en redes oportunısticas en espacios con alta concentración de personas. In XXV Jornadas de Paralelismo, 2014.

- Alvear, O., Calafate, C. T., Cano, J. C., Manzoni, P., Hernandez-Orallo, E., & Herrera-Tapia, J. (2015). Metodologıa para la Monitorización de Ozono en Valencia mediante Sensores de gama baja. In VI Jornadas de Computación Empotrada (JCE 2015).

- Herrera-Tapia, J, Hernandez-Orallo, E., Manzoni, P., Alvear, O., Calafate, C. T., & Cano, J. C. (2017). Estudio de la aplicabilidad de UAVs para monitorización ambientalt. In XXVIII Jornadas de Paralelismo. Jornadas SARTECO 2017.

- Alvear, O., Calafate, C. T., Cano, J. C., & Manzoni, P.(2017). Evaluación del uso de redes inalámbricas sub-GHz en la difusión de mensajes en redes oportunistas. In XXVIII Jornadas de Paralelismo. Jornadas SARTECO 2017.

Future Work

In this thesis we proposed relying on the mobile crowdsensing paradigm to monitor air pollution monitoring by focusing on Ozone, evaluating all work phases associated to the monitoring of this pollutant. As future work, we could analyze in more detail other pollutants such as Particle Matter, Carbon Monoxide, Carbon Dioxide, and Lead.

Moreover, we used a smartphone as the gateway between the sensor (Edge) and the central server (Cloud). As future work we could analyze the use of LPWAN technologies, such as LoRa, to transmit the data to the central server.

Since technologies such as LoRa are restricted in terms of message size and transmission speed, we could analyze the best option to encode the data, the best transmission frequency, etc.

Concerning to rural pollution monitoring, in the thesis we have only considered operations limited to a single UAV. The next step in our research could be to introduce multiple UAV operations, and the associated cooperation schemes. The following aspects need to be addressed when following this research line:

- **Cooperation:** to maximize the effectiveness, and to reduce mapping times, it is advisable to have several UAVs that cooperate with each other to achieve a same task, thereby accelerating the whole process, and avoiding battery exhaustion before completing the monitoring process.
- **Collision Avoidance:** since the different UAVs are expected to have some degree of autonomy regarding their mobility pattern, a correct coordination between nearby UAVs is required to avoid collisions when flying at a close range.
- **Communications:** to achieve the aforementioned goals of cooperation and collision avoidance, communications between UAVs, and between UAVs and a central management unit, are required.

On the other hand, using mobile sensors installed on UAVs introduces new issues to the sensing process that should also be addressed:

- **Altitude:** despite currently most pollution studies are made at a ground level, the use of UAVs allows us to determine the concentration of pollutants at different heights, thus allowing to determine if there are layers of pollutants that can cause health problems in rugged mountain sides.
- **Influence of the wind:** the sampling procedure includes sensors that are sensitive to the wind conditions. In addition, wind causes the overall pollution map to be more dynamic. In this context, both issues deserve more scrutiny.

Part V

Acronyms and References

Acronyms

5G Fifth Generation Network . . . 27
6lowPAN IPv6 over Low power Wireless Personal Area Network . . . 29
AQI Air Quality Index . . . 17
SBC Single Board Computer . . . 94
BLE Bluetooth Low Energy . . . 27
CO Carbon Monoxide . . . 14
CO_2 Carbon Dioxide . . . 16
CoAP Constrained Application Protocol . . . 31
CoRE Constrained RESTful Environments . . . 30
COTS Commercial Off-the-shelf . . . 6
CPS Cyber Physical System . . . 24
CRM Customer Relationship Management
EEA European Environment Agency . . . 17
Eionet European environment information and observation network . . . 17
EPA U.S. Environment Protection Agency . . . 14
ESC Electronic Speed Controller . . . 92
EXI Efficient XML Interchange . . . 32
GPS Global Positioning System . . . 92
HALE High Altitude, Large Endurance . . . 42
HTTP Hyper Text Transfer Protocol . . . 30
IM Instant Messaging . . . 31

RESTful Representational state transfer 29
RFID Radio Frequency ID 37
RPi Raspberry Pi 76
RS Remote Sensing 18
RTOS Real-Time Operating System 94
sMAP Simple Measuring and Actuation Profile 32
SSN-XG Semantic Sensor Networks Incubator Group 34
SO_2 Sulfur Dioxide 15
SO_x Sulfur Oxides 15
TCP Transport Control Protocol 27
TCP/IP Internet Prototol Stack 27
UART Universal Asynchronous Receiver-Transmitter
UAS Unmanned Aerial System 99
UAV Unmanned Aerial Vehicle 4
UDP User Datagram Protocol 27
USB Universal Serial B
UWP Universal Windows Platform 71
VOC Volatic Organic Compound 13
VoIP Voice over IP 31
VTOL Vertical Take-Off and Landing 41
W3C World Wide Web Consortium 34
Wi-Fi Wireless Fidelity 28
WoT Web of Things 34
WSN Wireless Sensor Network 20
XML eXtendible Markup Language 30
XMPP eXtensible Messaging and Presence Protocol 31

Bibliography

[1] Adafruit. *Adafruit Available: https://www.adafruit.com/*. 2017 (cited on p. 65).

[2] U. S. E. P. Agency. *Air Quality Index Available: http://cfpub.epa.gov/airnow/index.cfm? action=aqibasics.aqi*. 2015 (cited on pp. 17, 53, 110, 113, 124).

[3] A. Ahmed and E. Ahmed. "A survey on mobile edge computing". In: *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE. 2016, pp. 1–8 (cited on p. 24).

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey". In: *Computer networks* 38.4 (2002), pp. 393–422 (cited on p. 35).

[5] O. A. Alvear, N. R. Zema, E. Natalizio, and C. T. Calafate. "A chemotactic pollution-homing UAV guidance system". In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*. IEEE. 2017, pp. 2115–2120 (cited on p. 124).

[6] Ó. Alvear, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Calibrating low-end sensors for ozone monitoring". In: *International Internet of Things Summit*. Springer. 2015, pp. 251–256 (cited on pp. 38, 39).

[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 (2016) (cited on pp. 37–40).

[8] O. Alvear, N. R. Zema, E. Natalizio, and C. T. Calafate. "Using UAV-based systems to monitor air pollution in areas with poor accessibility". In: *Journal of Advanced Transportation* 2017 (2017) (cited on pp. 124, 128).

[9] K. Anderson and K. Gaston. "Lightweight unmanned aerial vehicles will revolutionize spatial ecology". In: *Frontiers in Ecology and the Environment* 11.3 (2013), pp. 138–146 (cited on p. 42).

[10] M. André. "The ARTEMIS European driving cycles for measuring car pollutant emissions". In: *Science of the total Environment* 334 (2004), pp. 73–84 (cited on p. 25).

[11] S. Asherson, P. Kritzinger, and P. Pileggi. "Wireless Standards and Mesh Networks". In: *Cape Town: sn* (2007) (cited on p. 28).

[12] L. Atzori, A. Iera, and G. Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805 (cited on p. 26).

[13] R. Baheti and H. Gill. "Cyber-physical systems". In: *The impact of control technology* 12 (2011), pp. 161–166 (cited on p. 24).

[14] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu. "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards". In: *Computer communications* 30.7 (2007), pp. 1655–1695 (cited on p. 28).

[15] R. E. Basher. *Review of the Dobson spectrophotometer and its accuracy.* Springer, 1985 (cited on p. 18).

[16] P. Basu, J. Redi, and V. Shurbanov. "Coordinated flocking of UAVs for improved connectivity of mobile ground nodes". In: *Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE.* Vol. 3. IEEE. 2004, pp. 1628–1634 (cited on p. 100).

[17] J. Bellvert, P. Zarco-Tejada, J. Girona, and E. Fereres. "Mapping crop water stress index in a 'Pinot-noir' vineyard: comparing ground measurements with thermal remote sensing imagery from an unmanned aerial vehicle". In: *Precision Agriculture* 15.4 (2014), pp. 361–376 (cited on p. 43).

[18] C. R. Ben Fry. *Processing.org Available: https://www.processing.org/.* 2016 (cited on p. 68).

[19] M. d. F. Bento. “Unmanned aerial vehicles: an overview”. In: *Inside GNSS* 3.1 (2008), pp. 54–61 (cited on p. 40).

[20] J.-D. M. M. Biomo, T. Kunz, and M. St-Hilaire. “An enhanced Gauss-Markov mobility model for simulations of unmanned aerial ad hoc networks”. In: *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*. IEEE. 2014, pp. 1–8 (cited on p. 100).

[21] S. Bluetooth. “Bluetooth core specification version 4.0”. In: *Specification of the Bluetooth System* (2010) (cited on p. 28).

[22] E. Borgia. “The Internet of Things vision: Key features, applications and open issues”. In: *Computer Communications* 54 (2014), pp. 1–31 (cited on p. 23).

[23] M. Botts, A. Robin, J. Greenwood, and D. Wesloh. “OGC sensorML: model and XML encoding standard”. In: *Technical Standard* 2.12-000 (2014) (cited on p. 34).

[24] O. Bouachir, A. Abrassart, F. Garcia, and N. Larrieu. “A mobility model for UAV ad hoc network”. In: *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE. 2014, pp. 383–388 (cited on p. 100).

[25] I. BoussaıD, J. Lepagnot, and P. Siarry. “A survey on optimization metaheuristics”. In: *Information Sciences* 237 (2013), pp. 82–117 (cited on p. 101).

[26] T. Bray. “The javascript object notation (json) data interchange format”. In: (2014) (cited on p. 32).

[27] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. “Extensible markup language (XML).” In: *World Wide Web Journal* 2.4 (1997), pp. 27–66 (cited on p. 32).

[28] O. Briante, V. Loscri, P. Pace, G. Ruggeri, and N. R. Zema. “COMVIVOR: An Evolutionary Communication Framework Based on Survivors’ Devices Reuse”. In: *Wireless Personal Communications* 85.4 (2015), pp. 2021–2040. ISSN: 1572-834X. DOI: 10.1007/s11277-015-2888-y (cited on p. 100).

[29] M. Brković and V. Sretović. “Urban Sensing–Smart Solutions for Monitoring Environmental Quality: Case Studies from Serbia”. In: *Congress Proceedings. 48th ISOCARP-International Society of City and Regional Planners World Congress: Fast Forward: Planning in a (hyper) dynamic*

urban context. Perm, Russia. Retrieved from http://www. isocarp. net/-Data/case_ studies/2215. pdf. 2012 (cited on p. 40).

[30] R. D. Brook, J. R. Brook, B. Urch, R. Vincent, S. Rajagopalan, and F. Silverman. "Inhalation of fine particulate air pollution and ozone causes acute arterial vasoconstriction in healthy adults". In: *Circulation* 105.13 (2002), pp. 1534–1536. ISSN: 00097322. DOI: 10.1161/01.CIR.0000013838.94747.64 (cited on p. 15).

[31] I. Celino and S. Kotoulas. "Smart Cities [Guest editors' introduction]". In: *IEEE Internet Computing* 6 (2013), pp. 8–11 (cited on p. 24).

[32] T. M. Chen, J. Gokhale, S. Shofer, and W. G. Kuschner. "Outdoor air pollution: Particulate matter health effects". In: *American Journal of the Medical Sciences* 333.4 (2007), pp. 235–243 (cited on p. 15).

[33] T.-M. Chen, J. Gokhale, S. Shofer, and W. G. Kuschner. "Outdoor air pollution: ozone health effects." In: *The American journal of the medical sciences* 333.4 (2007), pp. 244–8. ISSN: 0002-9629. DOI: 10.1097/MAJ.0b013e31803b8e8c (cited on p. 15).

[34] 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. 38, 40).

[35] I. Colomina and P. Molina. "Unmanned aerial systems for photogrammetry and remote sensing: A review". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 92 (2014), pp. 79–97 (cited on p. 42).

[36] M. Compton, H. Neuhaus, K. Taylor, and A. Parashar. "Semantic Sensor Network Ontology". In: *Web Semantics: Science, Services and Agents on the World Wide Web* (2012), pp. 25–32 (cited on p. 34).

[37] T. H. Cox, C. J. Nagy, M. A. Skoog, I. A. Somers, and R. Warner. *Civil UAV Capability Assessment* (cited on p. 42).

[38] K. Dalamagkidis. "Classification of uavs". In: *Handbook of unmanned aerial vehicles*. Springer, 2015, pp. 83–91 (cited on p. 40).

[39] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. "sMAP: a simple measurement and actuation profile for physical information". In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2010, pp. 197–210 (cited on p. 32).

[40] S. E. Deering. "Internet protocol, version 6 (IPv6) specification". In: (1998) (cited on p. 29).

[41] H. Demirkan. "A smart healthcare systems framework". In: *It Professional* 15.5 (2013), pp. 38–45 (cited on p. 26).

[42] H. T. Dinh, C. Lee, D. Niyato, and P. Wang. "A survey of mobile cloud computing: architecture, applications, and approaches". In: *Wireless communications and mobile computing* 13.18 (2013), pp. 1587–1611 (cited on p. 24).

[43] M. Dunbabin and L. Marques. "Robots for environmental monitoring: Significant advancements and applications". In: *IEEE Robotics & Automation Magazine* 19.1 (2012), pp. 24–39 (cited on p. 40).

[44] R. Eberhart and J. Kennedy. "A new optimizer using particle swarm theory". In: *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE. 1995, pp. 39–43 (cited on p. 102).

[45] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. "BikeNet: A mobile sensing system for cyclist experience mapping". In: *ACM Transactions on Sensor Networks (TOSN)* 6.1 (2009), p. 6 (cited on p. 25).

[46] W. Ejaz, A. Anpalagan, M. A. Imran, M. Jo, M. Naeem, S. B. Qaisar, and W. Wang. "Internet of Things (IoT) in 5G wireless communications". In: *IEEE Access* 4 (2016), pp. 10310–10314 (cited on p. 27).

[47] U. Fayyad and K. Irani. "Multi-interval discretization of continuous-valued attributes for classification learning". In: (1993) (cited on p. 118).

[48] R. T. Fielding. "REST: architectural styles and the design of network-based software architectures". In: *Doctoral dissertation, University of California* (2000) (cited on p. 29).

[49] R. P. Foundation. *Raspberry Pi*. https://www.raspberrypi.org/, Accessed: October 10, 2015 (cited on p. 93).

[50] S. Furuhashi. "MessagePack: It's like JSON. but fast and small, 2014". In: *URL http://msgpack. org* () (cited on p. 33).

[51] R. K. Ganti, F. Ye, and H. Lei. "Mobile crowdsensing: current state and future challenges." In: *IEEE Communications Magazine* 49.11 (2011), pp. 32–39 (cited on p. 35).

[52] W. Gao, Y. Tian, T. Huang, S. Ma, and X. Zhang. "The IEEE 1857 standard: Empowering smart video surveillance systems". In: *Intelligent Systems, IEEE* 29.5 (2014), pp. 30–39 (cited on p. 26).

[53] A. Glasmeier and S. Christopherson. *Thinking about smart cities*. 2015 (cited on p. 23).

[54] C. Gomez, J. Oller, and J. Paradells. "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology". In: *Sensors* 12.9 (2012), pp. 11734–11753 (cited on p. 28).

[55] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future Generation Computer Systems* 29.7 (2013), pp. 1645–1660 (cited on pp. 23, 26).

[56] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou. "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm". In: *ACM Computing Surveys (CSUR)* 48.1 (2015), p. 7 (cited on p. 35).

[57] A. Gupta and R. K. Jha. "A survey of 5G network: Architecture and emerging technologies". In: *IEEE access* 3 (2015), pp. 1206–1232 (cited on p. 27).

[58] S. G. Gupta, M. M. Ghonge, and P. Jawandhiya. "Review of unmanned aircraft system (UAS)". In: *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 2.4 (2013), pp–1646 (cited on p. 40).

[59] S. Gupte, P. Mohandas, and J. Conrad. "A survey of quadrotor Unmanned Aerial Vehicles". In: *Southeastcon, 2012 Proceedings of IEEE*. Mar. 2012, pp. 1–6 (cited on p. 42).

[60] E. Haselsteiner and K. Breitfuß. “Security in near field communication (NFC)”. In: *Workshop on RFID security*. 2006, pp. 12–14 (cited on p. 28).

[61] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. “Participatory air pollution monitoring using smartphones”. In: *Mobile Sensing* (2012), pp. 1–5 (cited on p. 38).

[62] C. Hewitt. “Security Without IoT Mandatory Backdoors”. In: (2016) (cited on p. 26).

[63] S.-C. Hu, Y.-C. Wang, C.-Y. Huang, and Y.-C. Tseng. “Measuring air quality in city areas by vehicular wireless sensor networks”. In: *Journal of Systems and Software* 84.11 (2011), pp. 2005–2012 (cited on p. 40).

[64] C. H. Hugenholtz, B. J. Moorman, K. Riddell, and K. Whitehead. “Small unmanned aircraft systems for remote sensing and earth science research”. In: *Eos, Transactions American Geophysical Union* 93.25 (2012), pp. 236–236 (cited on p. 42).

[65] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. “MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks”. In: *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*. IEEE. 2008, pp. 791–798 (cited on p. 30).

[66] D. Industries. *GrovePi*. http://www.dexterindustries.com/grovepi/, Accessed: October 10, 2015 (cited on p. 94).

[67] Intel-Corp. *Intel Edison Compute Module Available: http://www.intel.eu/content/ www/ eu/ en/ do-it-yourself/ edison.html*. 2016 (cited on p. 77).

[68] K. Kajimoto, R. Matsukura, J. Hund, M. Kovatsch, and K. Nimura. “Web of Things (WoT) Architecture (Unofficial Draft)”. In: *WoT W3C Interest Group: Cambridge, MA, USA* (2016) (cited on p. 34).

[69] M. Kamionka, P. Breuil, and C. Pijolat. “Calibration of a multivariate gas sensing device for atmospheric pollution measurement”. In: *Sensors and Actuators B: Chemical* 118.1-2 (2006), pp. 323–327 (cited on p. 38).

[70] P. S. Kanaroglou, M. Jerrett, J. Morrison, B. Beckerman, M. A. Arain, N. L. Gilbert, and J. R. Brook. “Establishing an air pollution monitoring network for intra-urban population exposure assessment: A location-allocation ap-

proach". In: *Atmospheric Environment* 39.13 (2005), pp. 2399–2409 (cited on p. 18).

[71] J. Kennedy. "Particle swarm optimization". In: *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766 (cited on p. 119).

[72] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. "Mobile phone sensing systems: A survey". In: *Communications Surveys & Tutorials, IEEE* 15.1 (2013), pp. 402–427 (cited on p. 26).

[73] E. Kuiper and S. Nadjm-Tehrani. "Mobility models for UAV group reconnaissance applications". In: *2006 International Conference on Wireless and Mobile Communications (ICWMC'06)*. IEEE. 2006, pp. 33–33 (cited on p. 100).

[74] J. Landt. "The history of RFID". In: *Potentials, IEEE* 24.4 (2005), pp. 8–11 (cited on p. 73).

[75] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. "A survey of mobile phone sensing". In: *Communications Magazine, IEEE* 48.9 (2010), pp. 140–150 (cited on p. 53).

[76] E. A. Lee. "Cyber physical systems: Design challenges". In: *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on*. IEEE. 2008, pp. 363–369 (cited on p. 24).

[77] D. Lenior, W. Janssen, M. Neerincx, and K. Schreibers. "Human-factors engineering for smart transport: Decision support for car drivers and train traffic controllers". In: *Applied ergonomics* 37.4 (2006), pp. 479–490 (cited on p. 26).

[78] Libelium. *Libelium Available: http://www.libelium.com/*. 2017 (cited on p. 65).

[79] X. Liu, S. Cheng, H. Liu, S. Hu, D. Zhang, and H. Ning. "A survey on gas sensing technology". In: *Sensors* 12.7 (2012), pp. 9635–9665 (cited on p. 71).

[80] D. Locke. *MQTT v3. 1 Protocol Specification*. Tech. rep. Technical Report, International Business Machines Corporation (IBM) and Eurotech, 2010, 42p, 2010 (cited on p. 30).

[81] H. Ma, D. Zhao, and P. Yuan. "Opportunities in mobile crowd sensing". In: *Communications Magazine, IEEE* 52.8 (2014), pp. 29–35 (cited on p. 35).

[82] D. McFarlane, V. Giannikas, A. C. Wong, and M. Harrison. "Product intelligence in industrial control: Theory and practice". In: *Annual Reviews in Control* 37.1 (2013), pp. 69–88 (cited on p. 26).

[83] Q. McFrederick, J. Kathilankal, and J. Fuentes. "Air pollution modifies floral scent trails". In: *Atmospheric Environment* 42.10 (2008), pp. 2336–2348 (cited on p. 91).

[84] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. "Pixhawk: A system for autonomous flight using onboard computer vision". In: *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE. 2011, pp. 2992–2997 (cited on p. 94).

[85] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision". In: *Autonomous Robots* 33.1 (2012), pp. 21–39. DOI: 10.1007/s10514-012-9281-4 (cited on p. 94).

[86] Micropython. *MycroPython Available:https://micropython.org/*. 2017 (cited on p. 71).

[87] G. Mulligan. "The 6LoWPAN architecture". In: *Proceedings of the 4th workshop on Embedded networked sensors*. ACM. 2007, pp. 78–82 (cited on p. 29).

[88] S. A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, and J. Ma. "Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing". In: *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*. Vol. 2. IEEE. 2007, pp. 113–120 (cited on p. 37).

[89] D. Orfanus and E. P. de Freitas. "Comparison of UAV-based reconnaissance systems performance using realistic mobility models". In: *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE. 2014, pp. 248–253 (cited on p. 100).

[90] G. Pajares. "Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs)". In: *Photogrammetric Engineering & Remote Sensing* 81.4 (2015), pp. 281–329 (cited on p. 42).

[91] C. E. Perkins. “IP mobility support for IPv4, revised”. In: (2010) (cited on p. 29).

[92] J. Postel. *User datagram protocol*. Tech. rep. 1980 (cited on p. 29).

[93] J. Postel. “Transmission control protocol”. In: (1981) (cited on p. 29).

[94] M. Pujadas, J. Plaza, J. Teres, B. Artıñano, and M. Millan. “Passive remote sensing of nitrogen dioxide as a tool for tracking air pollution in urban areas: the Madrid urban plume, a case of study”. In: *Atmospheric Environment* 34.19 (2000), pp. 3041–3056 (cited on p. 25).

[95] Pycom. *Pycom Available: https://pycom.io/*. 2017 (cited on p. 67).

[96] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2016 (cited on pp. 108, 121).

[97] V. Ricquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge. “The smart home concept: our immediate future”. In: *E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on*. IEEE. 2006, pp. 23–28 (cited on p. 26).

[98] B. P. Rimal, E. Choi, and I. Lumb. “A Taxonomy and Survey of Cloud Computing Systems.” In: *NCM* 9 (2009), pp. 44–51 (cited on p. 24).

[99] P. Saint-Andre. “Extensible messaging and presence protocol (XMPP): Core”. In: (2011) (cited on p. 31).

[100] J. Schneider, T. Kamiya, D. Peintner, and R. Kyusakov. “Efficient XML interchange (EXI) format 1.0”. In: *W3C Proposed Recommendation* 20 (2011) (cited on p. 32).

[101] Seeed-Studio. *Grovepi Extension Board Available: http://www.seeedstudio.com/depot/ GrovePi-Starter-Kit-for-Raspberry-Pi-p-2240.html*. 2016 (cited on p. 76).

[102] Seeedstudio. *Seeedstudio Available: http://www.seeedstudio.com/*. 2017 (cited on p. 65).

[103] M. O. Sensor. *Datasheet: http://www.sensorsportal.com/ DOWNLOADS/MQ131.pdf*. 2017 (cited on p. 124).

[104] Z. Shelby. "Constrained RESTful environments (CoRE) link format". In: (2012) (cited on p. 30).

[105] Z. Shelby, K. Hartke, and C. Bormann. "The constrained application protocol (CoAP)". In: (2014) (cited on p. 31).

[106] Sigfox. *Sigfox Available: http://www.sigfox.com.* 2017 (cited on p. 28).

[107] N. Sornin and M. Luis. *LoRa MAC Specification.* 2015 (cited on p. 28).

[108] M. L. Stein. *Statistical Interpolation of Spatial Data: Some Theory for Kriging.* New York: Springer, 1999 (cited on pp. 108, 124).

[109] T. Systems. *Adafruit Available: http://www.tst-sistemas.es/en/.* 2017 (cited on p. 65).

[110] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. "A taxonomy of wireless micro-sensor network models". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 6.2 (2002), pp. 28–36 (cited on p. 37).

[111] S. Toral, D. Reina, F. Barrero, et al. "A Self Organising Aerial Ad Hoc Network Mobility Model for Disaster Scenarios". In: *Developments of E-Systems Engineering (DeSE), 2015 International Conference on.* IEEE. 2015, pp. 35–40 (cited on p. 100).

[112] W.-T. Tsai, X. Sun, and J. Balasooriya. "Service-oriented cloud computing architecture". In: *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on.* IEEE. 2010, pp. 684–689 (cited on p. 24).

[113] W. Tsujita, A. Yoshino, H. Ishida, and T. Moriizumi. "Gas sensor network for air-pollution monitoring". In: *Sensors and Actuators B: Chemical* 110.2 (2005), pp. 304–311 (cited on p. 38).

[114] *United States Environmental Protection Agency (EPA).* http://www.epa.gov/, Accessed: October 3, 2015 (cited on pp. 15, 16).

[115] P. Waher. "XEP-0323: Internet of Things—Sensor Data". In: *Experimental Standard, version 0.3* 4 (2014) (cited on p. 32).

[116] Y. Wan, K. Namuduri, Y. Zhou, and S. Fu. "A smooth-turn mobility model for airborne networks". In: *IEEE Transactions on Vehicular Technology* 62.7 (2013), pp. 3359–3370 (cited on p. 100).

[117] W. Wang, X. Guan, B. Wang, and Y. Wang. "A novel mobility model based on semi-random circular movement in mobile ad hoc networks". In: *Information Sciences* 180.3 (2010), pp. 399–413 (cited on p. 100).

[118] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley. "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use". In: *Remote Sensing* 4.6 (2012), pp. 1671–1692 (cited on p. 41).

[119] D. W. Wong, L. Yuan, and S. A. Perlin. "Comparison of spatial interpolation methods for the estimation of air quality data". In: *Journal of Exposure Science and Environmental Epidemiology* 14.5 (2004), pp. 404–415 (cited on p. 40).

[120] G. Yang and X. Zhou. "Cyber-physical systems". In: (2013) (cited on p. 24).

[121] J. Yick, B. Mukherjee, and D. Ghosal. "Wireless sensor network survey". In: *Computer networks* 52.12 (2008), pp. 2292–2330 (cited on p. 35).

[122] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. "Internet of things for smart cities". In: *Internet of Things Journal, IEEE* 1.1 (2014), pp. 22–32 (cited on pp. 24, 26).

[123] C. Zhang and J. Kovacs. "The application of small unmanned aerial systems for precision agriculture: a review". In: *Precision Agriculture* 13.6 (2012), pp. 693–712. ISSN: 1385-2256 (cited on p. 43).

[124] 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 p. 38).

[125] B. Zhou, K. Xu, and M. Gerla. "Group and swarm mobility models for ad hoc network scenarios using virtual tracks". In: *Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE*. Vol. 1. IEEE. 2004, pp. 289–294 (cited on p. 99).

[126] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi. "Improving web sites performance using edge servers in fog computing architecture". In: *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*. IEEE. 2013, pp. 320–323 (cited on p. 24).