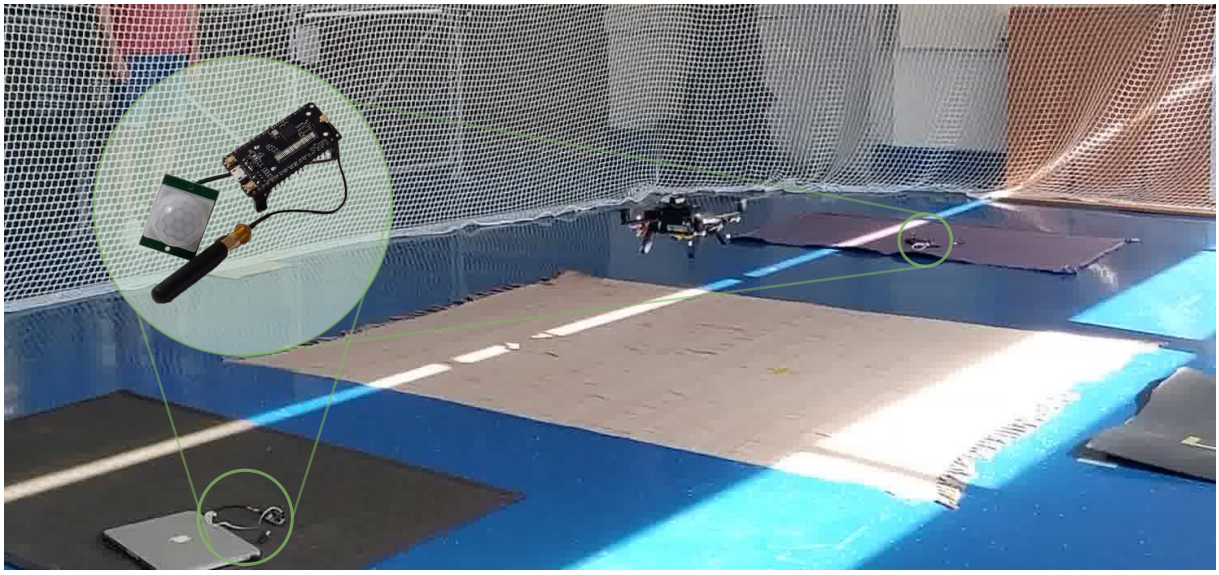
TÉCNICO LISBOA



Sensor Network Using a Mix of Cheap Sensors and Drones

Francisco Miguel Velez dos Santos

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Daniel de Matos Silvestre
Prof. Rita Maria Mendes de Almeida Correia da Cunha

Examination Committee

Chairperson: Prof. João Luís Costa Campos Gonçalves Sobrinho
Supervisor: Prof. Daniel de Matos Silvestre
Member of the Committee: Prof. David Alexandre Cabecinhas

November 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

The development of this thesis would not be possible without some important people present in my life.

I want to thank my Supervisor Professor Daniel Silvestre whose help was a big step towards completing my course. Thank you for your advising through this whole project and for giving me an opportunity to work in an area that I am most interested in. I also want to thank Professor Rita Cunha for her help in everything regarding the flight arena.

I also want to thank my friends and colleagues at IST whose friendship and support were vital in dealing with all the adversities that appeared along the past five years.

Thank you to my Parents, Brothers, close friends and family for their continued support through this journey and for always pushing for the best of me. Thank you to my cousins for their help in conducting tests for this thesis.

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020, through project PCIF/MPG/0156/2019 FirePuma and through COPELABS, University Lusófona project UIDB/04111/2020. Additional support in the means of hardware was also made available by the company Eco24, which was crucial in achieving the experimental setup.

Resumo

Esta dissertação propõe um sistema de vigilância baseado numa rede de sensores sem fios e num veículo aéreo não tripulado. Numa primeira análise, as várias tecnologias de comunicação sem fios são estudadas, resultando na escolha da tecnologia LoRa para suportar a comunicação entre sensores. Para melhor lidar com as caraterísticas da comunicação LoRa, é desenhado um protocolo, e o mesmo é testado em cenários realistas, tanto rurais como urbanos. Atendendo ao objetivo proposto aplicar um sistema de vigilância a áreas remotas para detetar intrusos numa área privada ou fogos florestais, veículos autónomos são usados para melhorar as capacidades de deteção. O sistema é implementado em pacotes de ROS e documentado para uso futuro. Em testes na arena de voo do ISR, o sistema obteve um bom erro quadrático médio de seguimento de $0.0381\ \mathrm{m}$ e inspecionou todos os eventos que ocorreram no campo de visão dos sensores de movimento com um atraso de $133\ \mathrm{ms}$.

Palavras-chave: Rede de Sensores sem Fios, LoRa, Veículo Aéreo não Tripulado, Sistema de vigilância

Abstract

This dissertation proposes a surveillance system based on wireless sensor networks and an unmanned aerial vehicle. Firstly, various wireless communication technologies are surveyed, resulting in the choice of LoRa technology to support the communication of the sensor nodes. In order to better deal with the characteristics of the LoRa communication, a protocol is designed and tested in realistic experiments in both rural and urban settings. Given the intended objective of having the surveillance system applied to remote areas to detect malicious people entering private property of forest fires, autonomous vehicles are used to further enhance the detection capabilities. The overall system is implemented in ROS packages and documented for future use. In experiments at the ISR flying arena, the system achieved a good average tracking root-mean-squared-error of $0.0381\ \mathrm{m}$ and inspected all events that crossed the motion detection sensors with a delay of $133\ \mathrm{ms}$.

Keywords: Wireless Sensor Network, LoRa, Unmanned Aerial Vehicle, Surveillance System

Contents

List of Tables

List of Figures

Acronyms

3GPP	Third Generation Partnership Project
AES	Advanced Encryption Standard
API	Application Programming Interface
BLE	Bluetooth Low Energy
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
CSS	Chirp Spread Spectrum
CSV	Comma-Separated Values
DGPS	Differential Global Positioning System
DL	Download Link
ECC	Error Correction Code
EKF	Extended Kalman Filter
GMSK	Gaussian Minimum Shift Keying
GPS	Global Positioning System
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
HITL	Hardware In The Loop
IC	Integrated Circuit
IMU	Inertial Measurement Unit
IOT	Internet of Things
ISM	Industrial, Scientific and Medical
ISR	Institute for Systems and Robotics

IST Instituto Superior Técnico

LAN Local Area Network

LPWAN Low Power Wide Area Network

LTE-A Long Term Evolution Advanced

LTE Long Term Evolution

LoRaWAN Long Range Wide Area Network

LoRa Long Range

MAC Medium Access Control

MCU Micro-Controller Unit

MEMS Micro-Electro-Mechanical Systems

MISO Master In Slave Out

MOCAP Motion Capture

MOP Maximum Output Power

MOSI Master Out Slave In

MTC Machine Type Communication

NB-IOT Narrowband Internet of things

OFDM Orthogonal Frequency Division Multiplexing

PID Proportional, Integral, Derivative

PIR Passive Infrared

QoS Quality of Service

RMSE Root-Mean-Square-Error

ROS Robot Operating System

RSSI Received Signal Strength Indicator

RTD Round-Trip Delay

SC-FDMA Single Carrier - Frequency Division Multiple Access

SCK Serial Clock

SDF Simulation Description Format

SIM Subscriber Identification Module

SITL Software In The Loop

SNR Signal-To-Noise Ratio

SPI Serial Peripheral Interface

SPN Substitution-Permutation Network

SS Slave Select

TOA Time On Air

TOA Time On Air

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

UL Upload Link

WLAN Wireless Local Area Network

WSN Wireless Sensor Network

WiFi-6 Wireless Fidelity version 6

Nomenclature

g_i	Gateway with ID i
m^i	Mission with ID i
n_i	Node with ID i
p_{n_i}	Location of node with ID i
w_i	Waypoint with ID i

Chapter 1

Introduction

1.1 Motivation

Since the past two decades, Wireless Sensor Networks (WSNs) have been gaining traction, partly due to the advancements made in the Micro-Electro-Mechanical Systems (MEMS) field which has both facilitated and motivated the development of new smart sensors characterized by their small size and low cost [1]. These networks usually consist of a large number of sensor nodes spread out across a geographic location with the goal of gathering sensor data. WSNs play a relevant role nowadays as they are widely used in fields like environment observation (e.g. pollution levels), agriculture (e.g. temperature, humidity and sunlight levels), disaster monitoring (e.g. water level in a river, wind speed and direction, amount of rain or snow) among others [2]. The power of these networks can be augmented by the introduction of autonomous vehicles (for instance, Unmanned Aerial Vehicles (UAVs)), allowing for cooperation between static and dynamic assets in numerous applications.

In the field of forest fires prevention and fighting, the integration between the UAVs and WSN allows for a better understanding of the region in study through the use of a fire likelihood map updated with the information gathered by the aforementioned technologies. In this particular case, the WSN can be composed of nodes spread around the forest or perhaps it can also be composed with citizens interacting through a mobile application. The idea is to combine the information from users with the data from the UAV which is used to scan hard-to-reach areas or other areas of increased interest in order to update the map. The map plays an important role when mobilizing assets to combat the fires and it also helps detecting fires in a premature state where they are easily put out. Given the increase in forest fires in the last few years [3], it is clear why this example motivates the study of the technologies approached in this thesis [4].

Another application is the actual project to be developed in this thesis which consists in the surveillance of a specific location for security purposes. In order to accomplish this, a WSN will be deployed in the location of interest with the nodes being equipped with appropriate sensors for the task at hand (motion sensors for instance). The data gathered by the sensors should then be transmitted and analysed by a base-station before a UAV is deployed to further investigate suspicious activity based on the

behaviour of the sensors in the network. The use of a UAV allows overcoming limitations of a typical WSN, in which nodes are in a fixed position, like following a dynamic target. Furthermore, this type of system dramatically reduces implementation costs given that a high number of surveillance cameras is not required.

1.2 Problem Definition

The proposal is to develop a surveillance system using a WSN of motion detectors together with UAVs in order to provide security for large areas. Thus, the definition of the problem at hand can be more straightforwardly explained if the problem is dissected into three main parts: the WSN, the UAV and the connection between the previous two. Moreover, the current proposal can be extended to a more general setting where the designer has access to a fleet of UAVs to cover both missions of passive surveillance (inspect sites with a high value in the uncertainty map) or active surveillance whenever there is an event generated by the WSN.

The wireless sensor network is composed of two types of devices: nodes (or end-devices) and gateways (or base-stations). Nodes are fundamentally smart sensor devices equipped with some sort of radio transceiver allowing for wireless communication. These devices can accommodate a large variety of both sensors and actuators and are equipped with a processor that can take readings from the sensors, process the values obtained and transmit them as well as following an analogous process to control actuators. Additionally, and as a constraint on this project, nodes must be able to operate on battery power so as to facilitate deployment. It is also assumed that nodes are aware of their location and that location is static. On the other hand, gateway devices are responsible for communicating with nodes, whether this means receiving the transmitted sensor values or transmitting a signal to control a node's actuator. Moreover, gateways form the bridge between the WSN nodes and the application layer (which often means the internet depending on where the application is hosted). Therefore, the gateways allow for the data collected by the WSN to be stored on a database, processed and visualized on a dashboard if necessary. Now that both types of devices available in the WSN have been defined, the network itself can be characterized. To begin with, the network will allow for bidirectional communication between the nodes and gateways, with the network topology not yet being defined due to the dependence on the chosen communications technology. Additionally, the network will be of the structured type where the nodes are deployed in a pre-determined fixed location. This choice comes from the need to have nodes in specific interest locations and comes with the added benefit of allowing coverage optimization and potentially reduce the total number of nodes required along with the associated reduction in costs and maintenance. Finally, not only motivated by the security aspect of this project but also given today's standards, all the network traffic must be encrypted to protect the data of sensitive nature.

The UAV has the main goal of carrying out missions which can be of either investigation or reconnaissance nature. In order to accomplish this, the vehicle must be equipped with hardware that allows it to monitor its surroundings (a thermal imaging camera for instance) as well as a way to transmit the captured data (some processing can be done on this data to reduce its size). Additionally, given the

autonomous nature of the vehicle, it must also be equipped with some form of an autopilot which can position the drone according to the mission specifications. The UAV should wait on the ground station (where it could be charging its batteries) until a mission is generated and it should come back to its takeoff location after completing each mission. Finally, the vehicle must be able to communicate with the ground station whilst on a mission, allowing for the mission itself or other parameters to be updated in real time.

Although both the WSN and the UAV can be studied and used independently, they must have a tight and seamless connection in this project which will be handled by the ground station, located in the site of deployment. The ground station must be able to accomplish several tasks. Firstly, it should include a gateway for the implemented WSN so as to allow for bidirectional communication with all the network nodes. Secondly, it should include some processing unit capable of converting the data gathered from the WSN into a mission for the UAV. However, the trajectory generation algorithm used to to create such mission is not within the scope of this thesis but is being developed within the context of this project. Thirdly, it needs to communicate with the UAV in order to send it the mission plan and receive the data gathered by the UAV. The ground station may also include a dashboard to display data regarding the mission and the sensors.

1.3 Literature Review

In order to contextualize the work to be done in this thesis, a review of the current state of the art is presented. In this sense, several forms of surveillance systems are briefly analysed. There is already a significant number of systems which take advantage of a network of sensor nodes to improve the capabilities of camera based surveillance systems [5–7].

In [5], the system makes use of the WSN to guide and position a rover equipped with a camera which captures the area of interest. This is useful for exploring unknown or dangerous areas. These systems that rely on the cooperation between WSNs and cameras have the advantages of being able to cover large surveillance areas with robustness and relatively low cost, which is particularly useful in visual security applications like banks, shopping malls and parking lots. Additionally, this type of surveillance is able to detect and locate objects of interest.

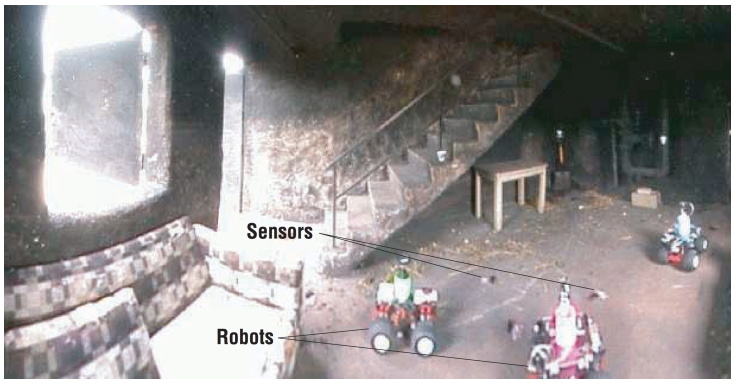


Figure 1.1: Project illustration from: Robot and sensor networks for first responders [5].

One more way WSNs have been used in surveillance systems is in fence surveillance. The work developed in [8] makes use of a WSN together with an Unmanned Ground Vehicle (UGV), an UAV, network camera (a camera on a gimbal connected to the internet) and a base-station to upgrade and improve the capabilities of traditional security cameras surveillance systems. The goal is to minimize human effort and increase automation through integration. The base-station has a control center which displays sensor status along with all the relevant monitoring information. The data from the WSN is used to position the network camera according to which sensor has been triggered and display only relevant footage on the control center. Furthermore, the UGV is used to patrol areas that are not covered by the WSN or to execute missions to specific locations. The UAV in the form of an RC-type helicopter is used to extend the communication distance between the nodes and the base-station by carrying a wireless Local Area Network (LAN) module. The paper refers that there are differences in performance from the variety of sensors that were used and tested. According to the results obtained in [8], the best performing sensor is the Passive Infrared (PIR) motion sensor, followed by the seismic, acoustic, magnetic and piezoelectric sensors in this order. It should also be noted that the system was tested with success, reporting all intruders being detected with a 5% false alarm rate and validating the importance of the UGV and UAV in improving the benefits of WSNs.

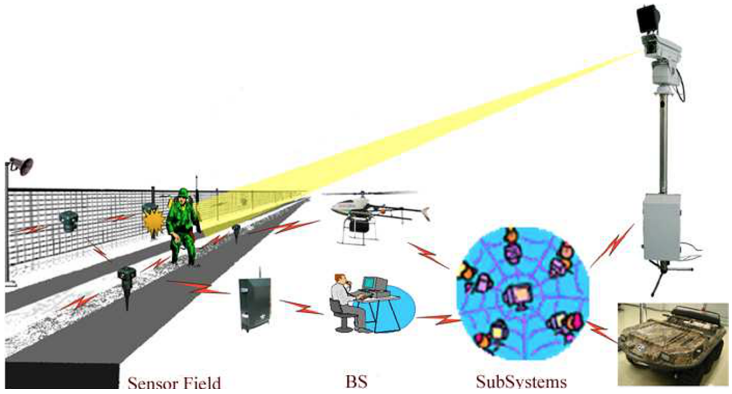


Figure 1.2: Project illustration from: Design of a fence surveillance system based on wireless sensor networks [8].

Another project [9] makes use of a WSN to collect data regarding the position of objects of interest. The data is used to control the rotation of static (in position) cameras so as to put the object of interest in their field of view. By placing cameras that are able to rotate in strategic places, it is possible to cover the same area with fewer cameras when comparing with fully static cameras. Additionally, instead of having an operator constantly monitoring video surveillance, man power is reduced by the integration with the WSN which autonomously controls the pan on the cameras and triggers a warning only when activity is detected. The result is a relevant decrease in both implementation costs and man power while maintaining, if not increasing, the level of security provided.

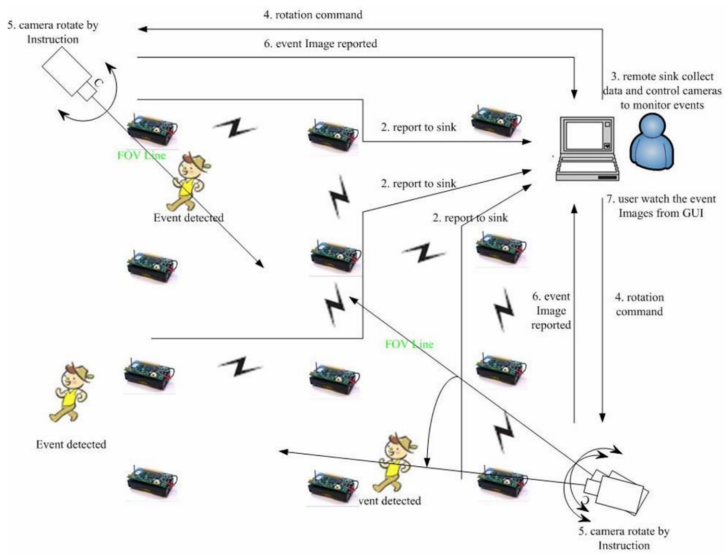


Figure 1.3: Project illustration from: Design and implementation of a real time video surveillance system with wireless sensor networks [9].

1.4 Thesis Outline

Outside the current chapter, the present thesis is organized in the following way:

- **Chapter 2 (WSN)**: begins with a state of art analyses on the currently used communication technologies followed by the choice of communication network used in the current proposal. Afterwards, a protocol that uses the LoRa modulation technology to send and receive messages is designed and implemented. Finally, the network is tested and its performance is evaluated.
- **Chapter 3 (WSN integration with the UAV)**: makes use of the developed network to implement the proposed surveillance system. After setting-up a simulation environment, a network is created using motion sensors and it is integrated with the Robot Operating System (ROS) nodes that control the autonomous vehicles. Finally, the system is tested and the results obtained are analysed.
- **Chapter 4 (Conclusion)**: sums up the work developed and provides insight into future work related to this project.

Chapter 2

Wireless Sensor Network

This chapter details how the WSN for this thesis was designed and developed as well as the theoretical background behind the decisions taken regarding the technologies used. In this way, as a starting point, an analysis is made regarding the main technologies used in wireless communications in Section 2.1. Secondly, a technology is chosen based on the requirements for the project and characteristics of the technology in Section 2.1.5. Thirdly, a protocol that uses the LoRa modulation technology to send and receive messages is designed and implemented to allow devices to communicate in a network Section 2.2. Finally, the developed network is tested in extensive scenarios and the results obtained are inspected so as to evaluate its performance and validate the fulfillment of all specifications in Section 2.4.

2.1 Literature Review

Given an application which requires some form of a communication network, there are several technologies with different trade-offs in: power consumption, cost, latency, bandwidth, range and coverage. In this section, we cover Wireless Local Area Network (WLAN) (i.e.: Wireless Fidelity version 6 (WiFi-6)), Bluetooth, Cellular (i.e.: Global System for Mobile Communication (GSM), Long Term Evolution Advanced (LTE-A), 5G) and Low Power Wide Area Network (LPWAN) (i.e.: Sigfox, Narrowband Internet of things (NB-IOT), LoRa) [10, 11], and highlight the main characteristics that showcase the most adequate network for this project. The presentation will be organized by ascending order of range (derived from the main applications of each technology). Also, it should be noted that all the specifications mentioned correspond to the ones applied in Europe and all the technologies were reviewed according to their main use cases.

2.1.1 Bluetooth

The current generation Bluetooth 5.0 has many improvements over its predecessor Bluetooth 4.2, namely a 4-fold longer range, twice the speed and support for mesh networks. However, the range is still in the magnitude of tens of meters with some devices achieving up to $100\ \mathrm{m}$ outdoors. Bluetooth 5 offers low power consumption capabilities through the introduction of Bluetooth Low Energy (BLE) [12]

and allows for a throughput of up to 1.4 Mbps [13]. The physical layer in Bluetooth 5.0 supports three modes of operation: LE 1M uncoded, LE 1M coded and LE 2M uncoded; "1M/2M" represents the symbol rate of 1 Mpbs or 2 Mbps and the "coded/uncoded" means whether the protocol is using an error coding scheme to improve sensitivity. Furthermore, Bluetooth works on the 2.4 GHz spectrum with a maximum theoretical bandwidth of 2 MHz at a Maximum Output Power (MOP) of 20 dBm [13].

2.1.2 WLAN

WiFi-6 is the new standard for WLAN communications. The new specification comes with improved speeds over the predecessors WiFi-4 and WiFi-5 as video traffic will dominate in the future with its increased throughput requirements [14, 15]. The 2.4 GHz frequency widely used in WiFi has a range in the magnitude of tens of meters [16]. WiFi is widely used in the home and office setting where short range communications with a high throughput rate are beneficial [14]. On the 5 GHz frequency the range is less than on the 2.4 GHz frequency but the throughput can be up to 9.6 Gbps. WiFi-6 is based on the IEEE 802.11ax protocol for its physical layer and has a maximum bandwidth of 40 MHz and 160 MHz for the 2.4 GHz and 5 GHz frequencies, respectively [17].

2.1.3 Cellular

Cellular networks like GSM, LTE-A and 5G offer the advantage of great coverage, with a range between base-stations of up to several kilometers. On the other hand, compatible devices are expensive when compared to alternatives and require a Subscriber Identification Module (SIM) card. Cellular networks like LTE-A are being used for Machine Type Communication (MTC) and Internet of Things (IOT) given their high speed characteristics and reliability [18]. The GSM network is based on Gaussian Minimum Shift Keying (GMSK) modulation for its physical layer and works across the frequencies 890 MHz to 915 MHz and 935 MHz to 960 MHz. With a theoretical limit of 200 KHz for the bandwidth, the throughput can reach 270 Kbps with a MOP of up to 30 dBm [19]. In 2011, a new generation of cellular communications appeared with LTE-A which utilizes the same frequencies as GSM but has a bandwidth of 100 MHz allowing for throughput levels of up to 1 Gbps on the Download Link (DL) and up to 500 Mbps on the Upload Link (UL). When it comes to the physical layer, LTE-A makes use of Orthogonal Frequency Division Multiplexing (OFDM) modulation on the DL and Single Carrier - Frequency Division Multiple Access (SC-FDMA) modulation on the UL [20, 21]. 5G is the new means of mobile communications and its main advantage against LTE-A is the increase in bandwidth (with the trade-off being the reduced range of less than 1 km). By utilizing *millimeter wave* technology, 5G can work on the 30 GHz to 300 GHz spectrum with a 500 MHz bandwidth which allows data rates of up to 1 Gbps with a MOP of 18 dBm [22, 23].

2.1.4 LPWAN

LPWAN is mainly characterized by its low power consumption, high efficiency, low cost and long range capabilities. Among the several available LPWAN technologies, there are three that stand out in

the industry: Sigfox, NB-IOT and LoRa [24].

Sigfox was developed in 2010 in France by the startup *Sigfox* and works both as a company which implements private IOT solutions and as an LPWAN operator. Sigfox uses the unlicensed Industrial, Scientific and Medical (ISM) bands which correspond to $868\ \mathrm{MHz}$ in Europe. Sigfox operates with a bandwidth of $100\ \mathrm{Hz}$ which allows for a maximum throughput of $100\ \mathrm{bps}$ using Binary Phase Shift Keying (BPSK) modulation in an ultra narrow-band. The use of the aforementioned ultra narrow-band and BPSK modulation allow for very low noise levels and power consumption as well as a notable range of up to $10\ \mathrm{km}$ in urban areas and $40\ \mathrm{km}$ in rural areas. With Sigfox, proprietary base stations are deployed by the company itself in order to increase coverage around the whole world. In 2018, Sigfox coverage was present in at least 31 countries. The communications between end-devices and base-stations are handled in the following way: the end-device sends the payload in three different frequency channels which is then received by a base-station. The need to send the payload three times comes from the lack of payload acknowledgment in the UL and is used to ensure reliability. On the downside, Sigfox will not operate as a LAN due to the proprietary base stations [24, 25]. Additionally, *Sigfox* imposes strict regulations on the communications allowed on their network. The payload length is limited to $12\ \mathrm{bytes}$ on the UL and $8\ \mathrm{bytes}$ on the DL. Moreover, the number of transmitted messages per day is also limited: 140 on the UL and 4 on the DL [25].

Third Generation Partnership Project (3GPP) released NB-IOT as a new IOT technology which is integrated into the Long Term Evolution (LTE) standard. However, the protocol stack for NB-IOT which is based on the protocol stack for LTE has been stripped of all the unused features so as to reduce device costs and increase battery life. Additionally, the NB-IOT network works on licensed frequencies where a time slotted synchronous protocol is implemented which makes this network preferable for applications that require better Quality of Service (QoS). This protocol comes with two main implications when compared to its LoRa counterpart: an increase in energy consumption due to the regular synchronization and a decrease in latency for the same reason. When it comes to range, NB-IOT can reach $35\ \mathrm{km}$ but it requires the use of LTE base stations which may not be available in more rural areas. Regarding throughput, NB-IOT has a bandwidth of either $180\ \mathrm{KHz}$ or $200\ \mathrm{KHz}$ depending on the configuration and a peak data rate of $100\ \mathrm{Kbps}$ with a MOP of up to $35\ \mathrm{dBm}$. Finally, NB-IOT has a notable spectrum cost of over $\$500\ \mathrm{million/MHz}$ and a base station cost of around $\$15000$ [26].

LoRa on the other hand, patented by *Semtech Corporation* uses a non-licensed band for its operation and works with a proprietary Chirp Spread Spectrum (CSS) modulation scheme which makes it a better choice for low power, low data rate and long range applications. The $125\ \mathrm{KHz}$ of bandwidth allows for a throughput that ranges from $250\ \mathrm{bps}$ to $50\ \mathrm{Kbps}$ with a MOP of $14\ \mathrm{dBm}$. There are several network protocols that make use of the LoRa modulation technology. One of them is the Long Range Wide Area Network (LoRaWAN) protocol which uses a star of stars topology in which the end-devices communicate with a core network though the use of gateway devices ("base-stations"). The gateway devices receive messages broadcast from the end-devices and relay them to the network server where duplicate messages are filtered in case of multiple gateways receiving the same message [26]. The network architecture employed by LoRaWAN is depicted in Figure 2.1.

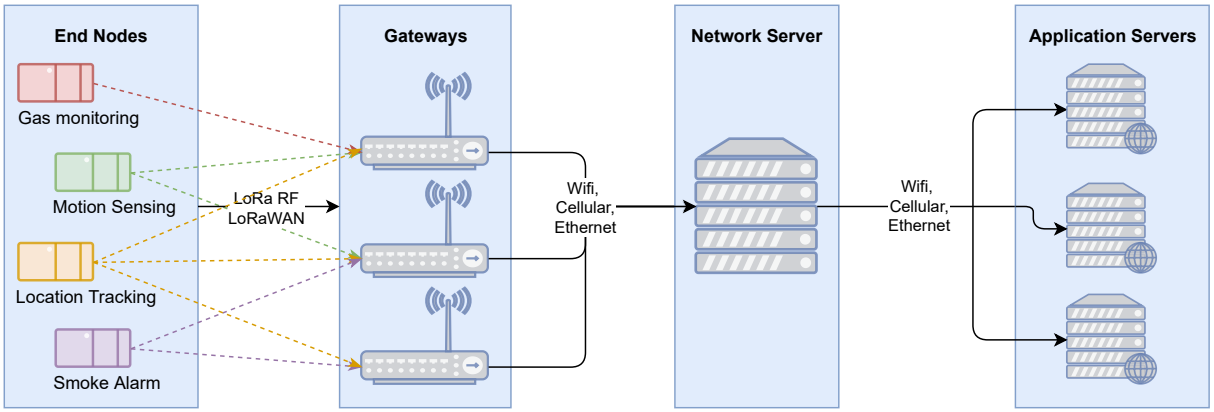


Figure 2.1: LoRaWAN Architecture Diagram.

LoRaWAN specifies three different Medium Access Control (MAC) protocols for the three different device classes created based on the trade-off between latency and power consumption [27]:

Class-A: Usually battery powered. It is the default class for an end-device and after a packet transmission it provides two receive windows before the device goes to sleep, saving battery life. Class-A devices have the best battery life with the worst latency performance.

Class-B: Usually battery powered. Built on top of Class-A, Class-B devices have extra receive windows, synchronized using beacons sent from the gateway. This allows for a balance between latency and power consumption.

Class-C: Usually wired to a power adapter. These devices can afford to always be ready to receive data when not transmitting since battery life is not a concern. Class-C devices have the best latency performance at the cost of an increase in power consumption.

LoRaWAN devices can sleep whenever instructed to do so due to the use of an asynchronous, ALOHA-based protocol where the nodes do not require the transmission of messages with regular frequency for synchronization purposes. This gives LoRa the edge on battery life when compared to NB-IOT, as expected given its worse latency performance. When it comes to range, a usual LoRa gateway can reach $15\ \mathrm{km}$ but more advanced gateways can reach up to hundreds of kilometers. LoRa components are readily available on the current market and a gateway device can usually cost from $\$100$ to $\$1000$ which is significantly less than one NB-IOT base station [26, 27].

2.1.5 Conclusions

From the aforementioned discussion regarding network technologies, it is possible to evaluate their strengths and weaknesses and find the optimal network for the envisioned application. Table 2.1 shows a list of the main specifications for all the reviewed networks. Namely, the physical layer used by the technology, the spectrum frequency in which the communications are handled, the maximum bandwidth allowed per channel, the maximum theoretical throughput achievable, the MOP and the maximum range

where packets can still be transmitted successfully. It should be noted that the MOP for WiFi-6 is strongly router-dependent which makes it a less relevant comparison factor and is therefore not mentioned in the table.

		PHY	Spectrum	Bandwidth	Throughput	MOP	Range
Bluetooth	Bluetooth 5.0	LE 1M uncoded, LE 1M coded, LE 2M uncoded	2.4 GHz	2 MHz	1.4 Mbps	20 dBm	$<$ 100 m
WLAN	WiFi 6.0	IEEE 802.11ax	2.4 GHz 5 GHz	40 MHz 160 MHz	9.6 Gbps	—	$<$ 100 m
Cellular	GSM	Based on GMSK	890-915 MHz, 935-960 MHz	200 KHz	270 Kbps	—	1 to several km
	LTE-A	DL:OFDM, UL:SC-FDMA	Same as GSM	100 MHz	DL:1 Gbps, UL:500 Mbps	30 dBm	1 to several km
	5G	5G-NR	mmWave: 30-300 GHz	500 MHz	1 Gbps	28 dBm	$<$ 1 km
LPWAN	Sigfox	BPSK	868 MHz	100 Hz	100 bps	14 dBm	$<$ 40 km
	NB-IOT	Same as LTE	Same as LTE	180/200 KHz	100 Kbps	35 dBm	$<$ 35 km
	LoRaWAN	LoRa (CSS)	867-869 MHz	125 KHz	250 bps - 50 Kbps	14 dBm	$<$ 15 km

Table 2.1: Main specifications of the networks approached in this section (all values refer to Europe specifications).

To complement the aforementioned Table 2.1, a radar graph is also presented in Figure 2.2. In this graph, a comparison is presented related to the main groups of technologies being analysed. As a result, and by using as metrics the range and coverage, throughput, latency performance, cost efficiency and battery life, this graph facilitates the reader to understand a technology choice for a certain application.

When it comes to range, the networks have been categorized according to the data in Table 2.1 where LPWAN is the technology which allows for the largest distance between devices as opposed to both Bluetooth and WLAN which usually have a maximum range of 100 meters. When talking about range, coverage should also be mentioned for the applicable networks. Cellular and NB-IOT (an LPWAN which uses the existing LTE network) as well as LoRa and Sigfox are technologies which already have substantial coverage around the world. As a result, when using these technologies, coverage must be taken into account at the site of deployment as it is required by the end-devices in order to connect to the network and this in turn reduces the importance of the maximum range. However, LoRa, which despite taking advantage from a growing network of gateways all around the world, still allows for a local deployment with a local gateway (not requiring coverage at all) as well as Cellular and WLAN which can also be deployed locally. Regarding throughput, the values shown in the graph (Figure 2.2) depict the best-case scenario for each technology and are once again a high-level view (more detailed information is shown in Table 2.1). Concerning latency, LPWANs have the worst latency performance with a transmission delay that can range from seconds up to minutes [11]; for Cellular networks the

latency values presented correspond to the ones for LTE and these usually range from 100 ms up to 6 s [28]; for WLAN the latency performance is safely the best with values in the millisecond range (usually less than 20 ms) [29]; finally, Bluetooth has a latency of around 20 ms to 200 ms as stated in a study carried out by *silabs* [30]. With respect to technology cost, an examination of the end-devices, network requirements and subscription fees was made. With Bluetooth and WLAN, there is no demand for an existing infrastructure and neither is any kind of subscription fee required. End-devices for both technologies are low cost but the star or even mesh topologies of WLAN require the use of routers which will increase the overall cost. Cellular, on the other hand, can only be used around an existing infrastructure of base-stations. Moreover, a SIM card must also be acquired and transmissions will either have a fixed price or a mandatory subscription fee (usually paid monthly). End-devices for Cellular are also expensive when compared to other options given that they require a SIM card interface which others do not. When referring to the technology cost of LPWANs, the several networks in this group have very different properties even though the end-devices have similar price characteristics and are all low cost. While Sigfox and NB-IOT require the use of an already established network infrastructure, LoRa does not. Additionally, Sigfox also requires a subscription to be paid so as to access their network of base-stations. At last, with regards to battery life, LPWAN has the best performance with devices being able to last up to ten years without changing the battery or recharging [11]; devices using Cellular technology (LTE) can last from 2 to 4 years [28] while Bluetooth devices will last up to 2 years [31] and finally, devices using WLAN have the highest power consumption and will only last for up to 1 year [32].

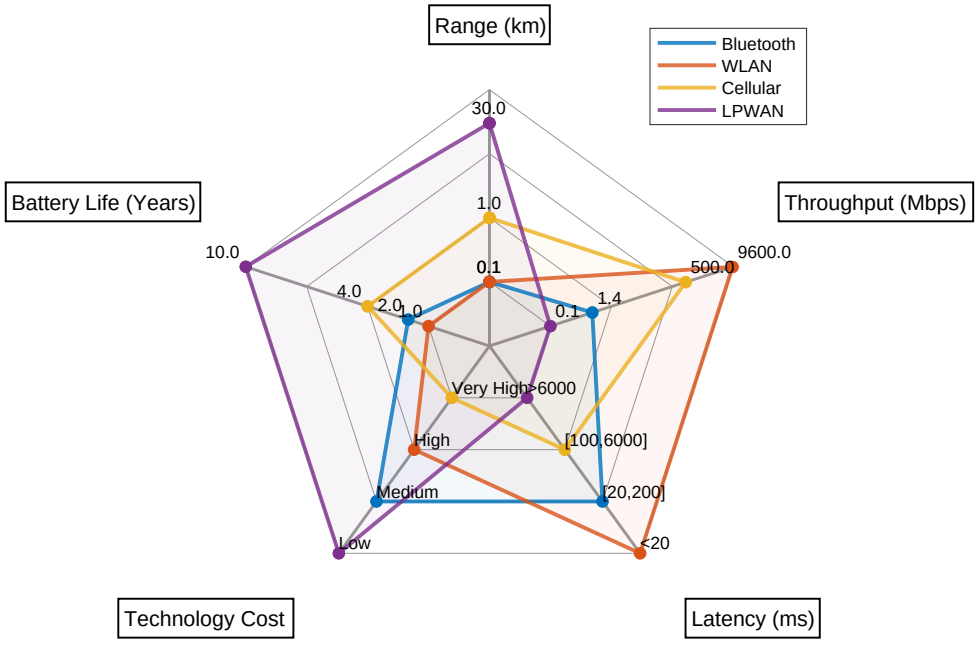


Figure 2.2: Performance Comparison across the categories of range, throughput, latency, technology cost and battery life between the main technologies approached: Bluetooth, WLAN, Cellular and LPWAN.

For the application addressed in this thesis, a network that has long range capabilities (at least 1 km between the nodes and the gateway) and low power consumption (allowing for battery operated sensor

devices) is required. Additionally, a low cost technology is preferable. Since only sensor data with low sample size (mostly binary values) and low sample rate needs to be transmitted, a network with high bandwidth capabilities is not necessary. Finally, regarding latency, no restrictions have been made so far since this can only be analysed when testing the developed network. For now, a maximum latency of 10 seconds is defined since when compared to the delay associated to the UAV time of flight, this value is negligible. From Figure 2.2, it is clear that LPWAN is the most promising network available. Within the LPWANs, Sigfox was ruled out first due to its coverage requirement and limitation on the number of transmitted messages which can be a security risk for this project since a node will be blocked out if it does not comply with the regulations. In addition, the use of Sigfox comes with a subscription fee which increases the technology cost [25]. For the remaining technologies, given that all the requirements had been met, the main decision factor was the technology cost. The implementation of a LoRa enabled network is several times more affordable than its rival NB-IOT. Additionally, a LoRa gateway can easily be deployed in any area whereas NB-IOT requires LTE coverage to function which may not be available at the deployment site for this project. Finally, a similar CSS modulation technology to the one used by LoRa had already been used by the US army to assist communications and surveillance due to its great properties of range and reliability (an application that is strongly related to this project) [26]. Hence, from the analyzed networks, the LoRa specification from the LPWANs is the choice which best fits the aforementioned requirements and it will be used to provide the communications network for this project.

2.2 LoRa based Wireless Sensor Network

In Section 2.1, LoRaWAN was mentioned as an example of a LoRa network protocol. This protocol comes with features like data encryption, device classes for different energy consumption applications and device addressing, which are useful for the WSN we are trying to build. However, LoRaWAN also has some disadvantages: it requires the use of specific LoRaWAN compatible devices, which are more expensive than just plain LoRa modems; it imposes additional regulations on the transmission duty cycle (which further decreases the allowed number of transmitted messages) and it introduces additional delays due to the use of slotted receive windows in the end-devices. These are relevant drawbacks since they can lead to nodes not being able to communicate with the gateway which is a security flaw. There was a choice made at this moment of developing a network protocol specific for the needs of this project. This decision was motivated by the disadvantages of the LoRaWAN protocol and by the hardware equipment that had already been obtained for this project (Equipment which included plain LoRa modems). In order to reference the devices the following nomenclature will be used: n_i represents a node with id i and g_i represents a gateway with id i.

2.2.1 LoRa Technology

As previously mentioned, the LoRa physical layer is based on CSS modulation technology, which is commonly used for radio applications. With CSS modulation, wide band frequency impulses which

increase or decrease over time are used to transport the encoded data [33]. The LoRa modulation process can be configured across different parameters.

Carrier Frequency

LoRa operates on the sub-GHz frequency band. For each region, a specific band is reserved for LoRa communications. In Europe, that frequency band ranges from $867\ \mathrm{MHz}$ to $869\ \mathrm{MHz}$.

Transmission Power

The transmission power value can be configured depending on the LoRa module from $2\ \mathrm{dBm}$ up to $20\ \mathrm{dBm}$. However, this value is regulated by region and in Europe the transmission power is limited to $14\ \mathrm{dBm}$ or $25\ \mathrm{mW}$.

Bandwidth

Regarding bandwidth, an increase in signal bandwidth allows the use of a higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity. Bandwidth values can range from $7.8\ \mathrm{KHz}$ up to $500\ \mathrm{KHz}$.

Coding Rate

As for the coding rate, it is a fraction between the number of raw bits over the number of encoded bits after Error Correction Code (ECC). It works with chunks of $4\ \mathrm{bits}$ and can encode them into 5, 6, 7 or $8\ \mathrm{bits}$.

Spreading Factor

The LoRa modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which the spread information is sent is referred to as the symbol rate (R_S). The spreading factor represents the number of raw bits that can be encoded per symbol as well as the number of chips contained in each symbol (2^{SF}); meaning that increasing the spreading factor will increase the number of chips per symbol which as a result allows for smaller and even negative Signal-To-Noise Ratio (SNR) values at the receiver, increasing the sensitivity, link budget and range.

Implicit Header Mode

The implicit header mode defines whether a header is included in the packet or not. The header has information regarding the payload length, the coding rate and whether a Cyclic Redundancy Check (CRC) is used in the packet.

Low Data Rate Optimization

Low Data Rate Optimization is a modem parameter which can be use to increase the robustness of a LoRa link at higher spreading factors. Its use is mandatory for a symbol time (T_S) larger than 16 ms.

Cyclic Redundancy Check

Finally, the CRC is used to enable or disable the inclusion of a CRC for the packet.

It should be noted that the aforementioned parameters often affect the receiver's sensitivity. Increasing the bandwidth will decrease the sensitivity on the receiver while increasing the spreading factor will do the opposite. Finally, (2.1) is useful to better characterize the transmission data rate.

$$R_b = \mathrm{SF} \times \frac{\mathrm{BW}}{2^{\mathrm{SF}}} \times \mathrm{CR} \tag{2.1}$$

2.2.2 Time on Air

In order to evaluate the network performance regarding the delay times, it is important to first characterize the LoRa packet Time On Air (TOA). This value represents the time it takes to transmit a single LoRa packet. To calculate the TOA for this network, the *SX1276* modem datasheet [34] was consulted. The TOA depends on a number of radio parameters: the Spreading Factor (SF), the Bandwidth (BW), the use of CRC (CRC), the Coding Rate (CR), the use of Implicit Header mode (IH) and the use of Low Data Rate Optimization (DE); as well as packet parameters like the preamble length (N_{PREAMBLE}) and payload length (PL). In this way, the symbol rate and symbol time are defined as:

$$R_S = \frac{\mathrm{BW}}{2^{\mathrm{SP}}}\,, \tag{2.2a}$$

$$T_S = \frac{1}{R_S}\,. \tag{2.2b}$$

The preamble time can now be calculated as a function of the preamble length in symbols:

$$T_{\mathrm{PREAMBLE}} = (N_{\mathrm{PREAMBLE}} + 4.25) \times T_S\,. \tag{2.3}$$

To calculate the payload time, the number of symbols in the payload must first be calculated:

$$N_{\mathrm{PAYLOAD}} = 8 + \max\left(\left\lceil \frac{8\mathrm{PL} - 4\mathrm{SF} + 28 + 16\mathrm{CRC} - 20\mathrm{IH}}{4(\mathrm{SF} - 2\mathrm{DE})} \right\rceil \times (\mathrm{CR} + 4), 0\right)\,, \tag{2.4}$$

Where $\lceil\ \rceil$ is used as a round up operation. The payload time is therefore given by:

$$T_{\mathrm{PAYLOAD}} = N_{\mathrm{PAYLOAD}} \times T_S\,. \tag{2.5}$$

Finally, the total TOA can be calculated by:

$$T_{\mathrm{PACKET}} = T_{\mathrm{PREAMBLE}} + T_{\mathrm{PAYLOAD}} \,. \tag{2.6}$$

In this case, the preamble size has the default value of 8 symbols and the payload length is of 18 bytes as explained in Section 2.2.5. Additionally, the CRC parameter has a value of 1 as CRC has been enabled for this network, the DE parameter has a value of 0 as there is no low data rate optimization for this network and the IH parameter has a value of 1 as no explicit header is being used. The remaining parameters are dependent on the implementation and configuration of the network and must be decided according to the circumstances of each implementation.

2.2.3 Custom Network Protocol

The main advantage of developing a custom protocol is the possibility to select some required features without all the unnecessary ones that other protocols offer along with their limitations. The main focus in this protocol is to have the ability for a base station to communicate with sensor devices bidirectionally mainly to gather sensor data. There is also an interest in decreasing the payload size and its corresponding TOA, which in turn saves power by decreasing the time that the modem is transmitting.

The communication protocol to be developed for the WSN defines three device types: **end-nodes** (able to be battery operated, can be equipped with a number of sensors and/or actuators), **range-extenders** (able to be battery operated, extend the range of the gateway) and **gateways** (responsible for the communication with the end-nodes and bridging the network to a base-station). Additionally, it must meet the following specifications: **Device addressing** - so communication can be made between the base station and a single node; **Data encryption** - so that the sensitive data in the payload is protected; **Bidirectional communication** - so that both UL and DL messages are supported; **Message delivery acknowledgement and retransmission** - given the use of a public frequency spectrum susceptible to interference; **Detection of Transmission Errors** - using a CRC of the data to detect corrupted packets.

Device Addressing

Each node device has a unique identifier in the form of a byte value which is defined at compile time (when the code is compiled and uploaded to the board) is what allows device addressing to work. When sending a message, this identifier is included as part of the message header (detailed in Section 2.2.5).

Data Encryption

To handle data encryption the well-known *Advanced Encryption Standard (AES)* encryption algorithm was used [35]. AES uses a fixed block size of 128-bits and a key size of 128-bits, 192-bits or 256-bits (the 256-bit variant was used in this project). This algorithm is based on the Substitution-Permutation Network (SPN) principle where several rounds of substitution and permutation operations are applied to the *plaintext* (the text to be encrypted) in order to generate the *ciphertext* (the encrypted text). The number of rounds applied depends on the size of the used key and corresponds to 14 rounds for a

256-bit key. To this extent, every node device has a unique symmetric 256-bit encryption key, which is used for both UL and DL communications. The gateway device has information of all the remaining keys at compile time. This allows for end-to-end encryption between the nodes and the gateway.

Bidirectional Communication

Communication between the nodes and the gateway is defined in the following way: nodes only send data to the gateway (corresponding to an UL message) and the gateway only sends data to nodes (corresponding to a DL message). To achieve this, a feature from the LoRa modem called *IQ inversion* is used. In sum, by enabling *IQ inversion*, the preamble consisting on a defined number of chirps, used so the LoRa modem can identify a packet, becomes inverted (the up-chirps are swapped for down-chirps). Figure 2.3 illustrates the concepts of preamble, up-chirp and down-chirp. Finally, the gateway uses no *IQ inversion* for packet reception and uses *IQ inversion* for packet transmission. In contrast, nodes use *IQ inversion* for packet reception and use no *IQ inversion* for packet transmission allowing for correct bidirectional communication according to specifications.

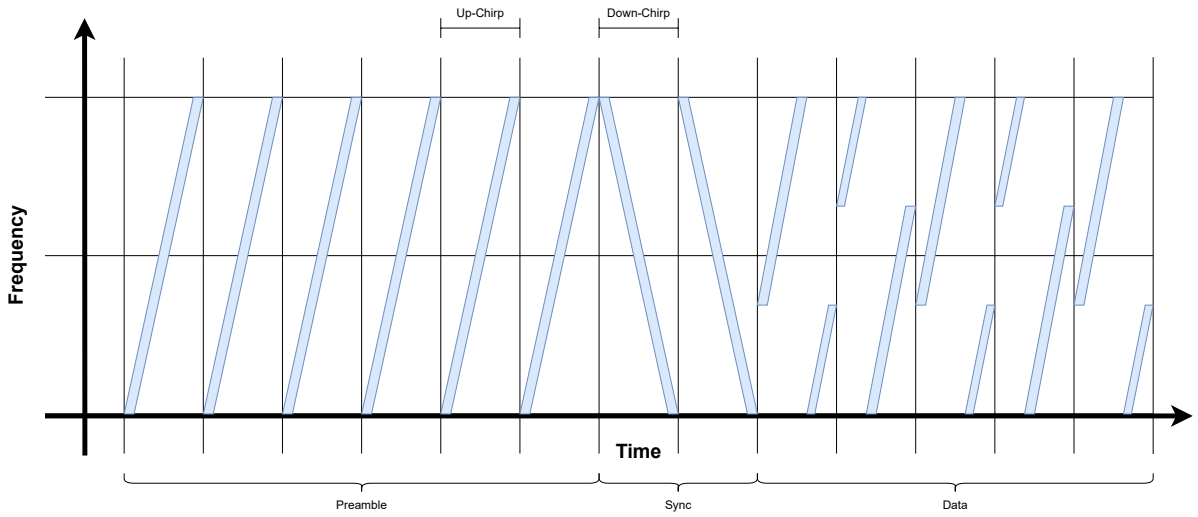


Figure 2.3: LoRa modulation on an example packet. (No *IQ inversion*.)

Message delivery acknowledgement and retransmission

In this project, it is important for a device in a network to know if a message has been correctly received as the undetected loss of messages can be a security vulnerability. To achieve this, every time a message is sent, the sender awaits for an acknowledgement confirming the correct reception. In case the acknowledge message is not received after a transmission, the sender will wait for a designated amount of time (defined at compile time, meaning that after deployment, physical presence at the node location is required in order to reprogram the node to change this value) and try to re-transmit the message as depicted in Figure 2.4. This process will be repeated up to a maximum number of retries (defined at compile time) after which the message is discarded. It should be noted that the gateway

cannot know when an UL message is dropped. However, it knows when a DL message is dropped and can inform the base-station of that occurrence. Additionally, we remark that re-transmitting a message can interfere with the process of handling the remaining messages of the queue for that node. For instance, if a message initially fails to be transmitted by a sensor node and, during its retransmission, a new trigger of a sensor occurs, then a new message will be generated and transmitted at the same time. To handle this, a message queue is implemented, which is illustrated for a practical case in Figure 2.5. In this example, a network with three nodes, where the nodes n_1 and n_2 are offline and the node n_3 is online, is used. Additionally, the network is configured with a retry timeout of $1200\ \mathrm{ms}$, a maximum number of retries of 5 and queue size of 5. This means that the gateway will only send a DL message every $1200\ \mathrm{ms}$ and, in case of failure, a message will be re-transmitted up to 5 times before being dropped. The executed test consists of sending a status request message to each node sequentially three times every three seconds. As a result, Figure 2.5 shows the queue size as a function of time where every data point represents a sent message. Each data point is labeled with the corresponding message ID. Moreover, the DL message commands sent from the server are also shown along with the responses to the status request messages.

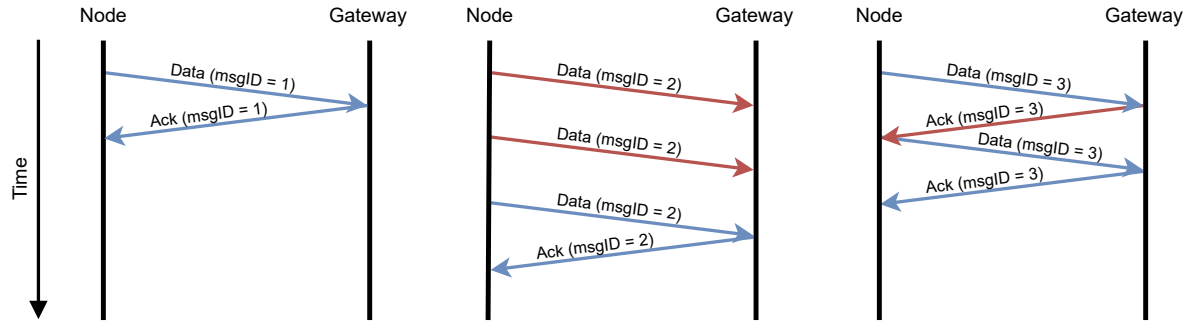


Figure 2.4: Sequence diagram for three different examples of how a message can be successfully transmitted. (A packet that was received correctly is shown in blue and a failed packet in red.)

As depicted in Figure 2.5, the DL commands from the server are represented by the orange dots and, as expected, appear separated by a three-second interval. The blue dots represent a response to a sent message, which can either be a failure to send the message or a successful response from the node. Looking at the message IDs, it can be seen that the gateway sends messages corresponding to the nodes n_1 and n_2 5 times each. This happens due to the nodes being offline and the gateway retrying to send the message before triggering a failure response message. In contrast, messages sent to the node n_3 are only sent once as the node successfully responds to the received messages. It should be noted that around the 28 second mark, the final DL message command is sent from the server. However, as shown, the queue size at that time is already equal to the maximum queue size. As a result, this message cannot be added to the queue, being immediately dropped and a failure response triggered (represented by the blue dot labeled with the message ID 9).

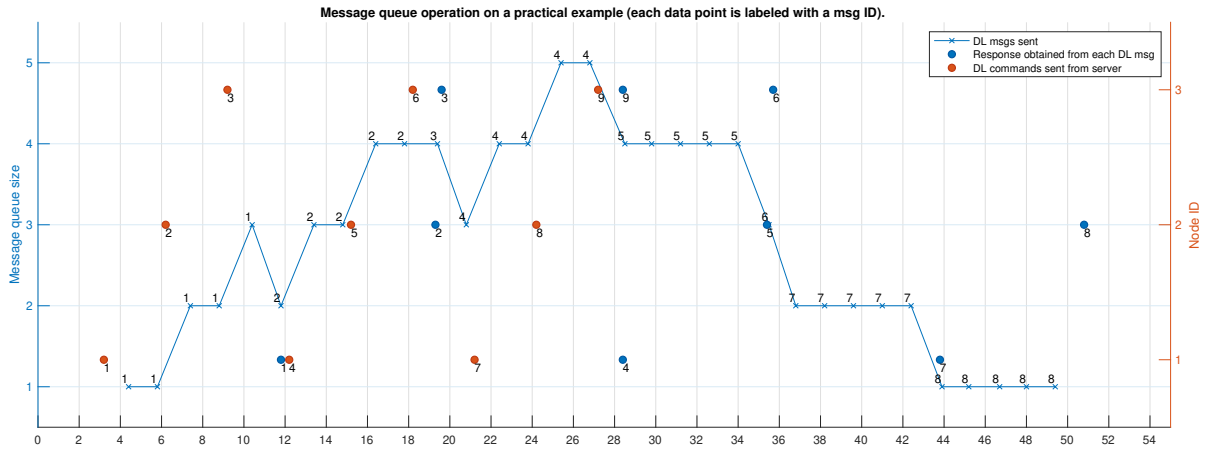


Figure 2.5: Practical example of the message queue operation and message retransmission executed on a network with three nodes where the nodes n_1 and n_2 are offline and only the node n_3 is online.

Detection of Transmission Errors

The LoRa modem used (*SX1276* Section 2.3) includes an option to enable a CRC. By doing this, a 16-bit value calculated using the remainder of a polynomial division of the payload contents is added to the packet. On reception, the same calculation is repeated and if the result is different signals a corrupted packet. In addition, the LoRa modulation process already includes cyclic ECC, which allows for the data in corrupted packets to be recovered (up to a certain extent) giving LoRa some tolerance to interference [34] (the Coding Rate parameter can be changed to increase or decrease the robustness of the error correction code).

This protocol is intended for a local deployment where only one gateway is utilized. The gateway is connected via serial to a processing unit which processes all the communications. If there are nodes located outside the range of the gateway, range-extenders can be used.

2.2.4 Network Topology

The topology of a network specifies how the devices communicate with each other. In wireless networks, there are three main topologies [36, 37]:

Star Topology - All nodes are connected to a gateway acting as middleware between the nodes;

Mesh Topology - Nodes are interconnected with each other. In a full mesh topology, there is a direct link between every pair of nodes whereas in partial meshes some links might be missing;

Cluster-tree Topology - A hybrid between star and mesh topologies. Nodes are clustered around routers or repeaters in a star topology and in turn the routers or repeaters communicate with each other and the gateway in a mesh topology.

The aforementioned topologies have different characteristics. The Cluster-tree topology is useful if there is a need to apply different topologies to different parts of the network, which is not the case in this project. The mesh topology is easier to expand since a new device only needs to be within range of any other existing device in the network. In addition, it is more robust since there is more than one path from a node to the gateway. However this topology incurs in higher power consumption since an end-device is required to be always listening, which is not ideal for battery operated devices. In contrast, the star topology offers the best power savings since devices only communicate directly to the gateway. This topology is suitable for the type of network being developed for this project and its architecture is depicted in Figure 2.6.

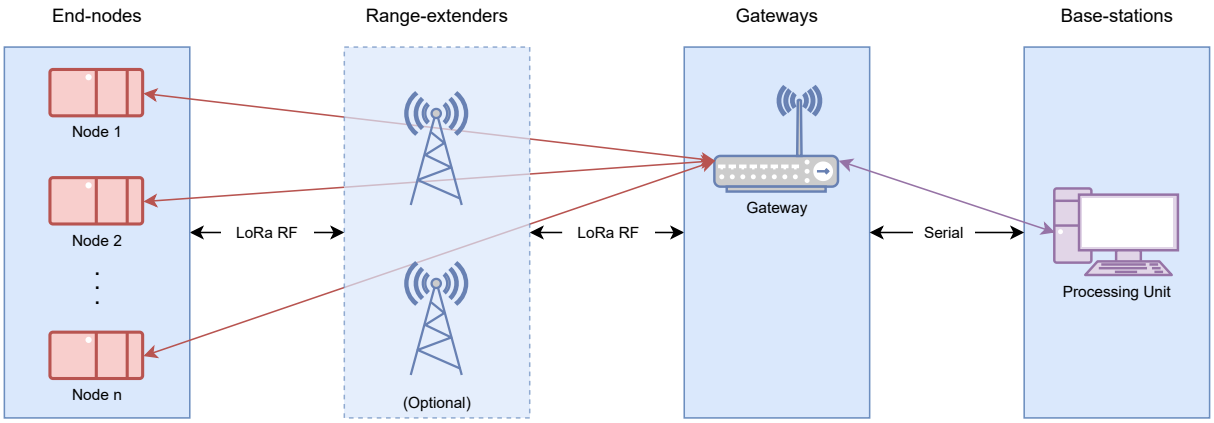


Figure 2.6: WSN Architecture Diagram.

2.2.5 Message Formatting and Data Packets

A requisite for a successful communication is the correct formatting of messages to be exchanged, which in the current protocol are:

- Status update message (UL);
- Sensor data message (UL);
- Status request message (DL);
- Control message (DL);
- Acknowledge message (UL, DL).

The aforementioned message types allow for the gateway to receive sensor data and status data from the end-devices as well as to request a status update from any node or to send a message to control an actuator. To meet all the specifications mentioned in Section 2.2.3, a custom packet format was defined with a plain header containing a network ID, necessary to distinguish different networks as well as due to the spectrum used to communicate, which is part of the public access ISM band. The network ID is therefore used to identify the messages that come from devices in this network. Additionally, the header

also includes the node ID, which represents the destination ID in case of a DL message or the sender ID in case of an UL message. The payload is encrypted using the well-known *AES* encryption algorithm [35] which only encrypts blocks of $16\ \text{bytes}$, explaining the fixed size of current messages. The entire packet format is depicted in Figure 2.7 in blue.

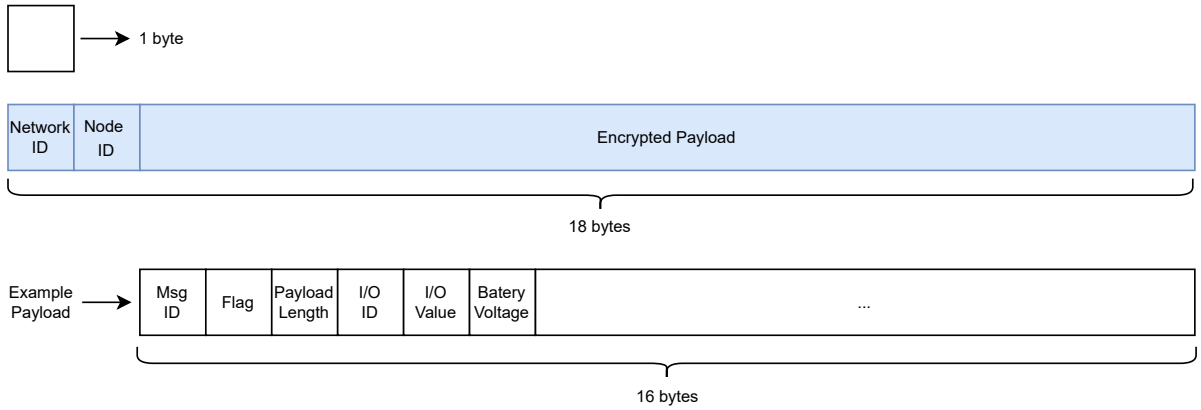


Figure 2.7: Packet and payload format.

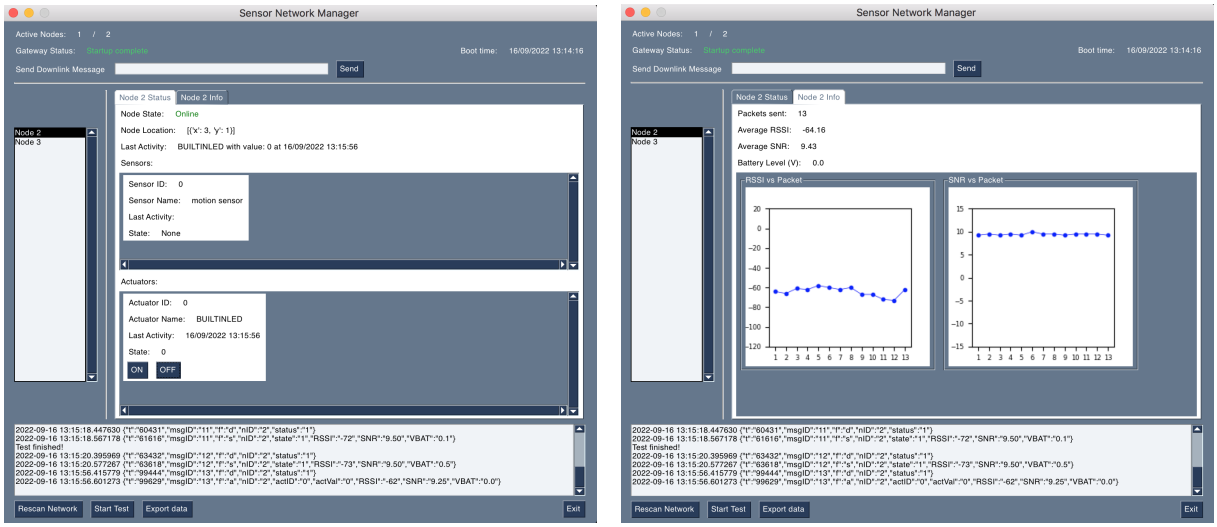
Also depicted in Figure 2.7 is a payload example containing a unique message ID used by the packet recipient to send back an acknowledgement, a flag byte to indicate the message type and a payload length byte indicating its size. Finally, there are bytes allocated for I/O (sensor ID and value in case of UL message or actuator ID and value in case of DL message) and for the battery voltage if the sender device is equipped with one. It should be noted that henceforward, since the encryption algorithm imposes a payload size of at least $16\ \text{bytes}$ there is room for additional data to be included in the payload.

2.2.6 Network Manager Interface

In order to monitor the network, a tool was developed in the form of a python Graphical User Interface (GUI) that runs on the processing unit connected to the gateway. The interface reads a *yaml* configuration file containing information regarding the network devices (including sensor and actuator information) and is updated in real-time whenever an UL message is received by the gateway.

The data monitored by the **Network Manager Interface** can be seen in Figure 2.8 and includes the following information:

- Gateway status;
- Number of active end nodes;
- Node status;
- Node location;
- Node last activity;



(a) Focus on the Info tab.

(b) Focus on the Stats tab.

Figure 2.8: Sensor Network Manager python interface.

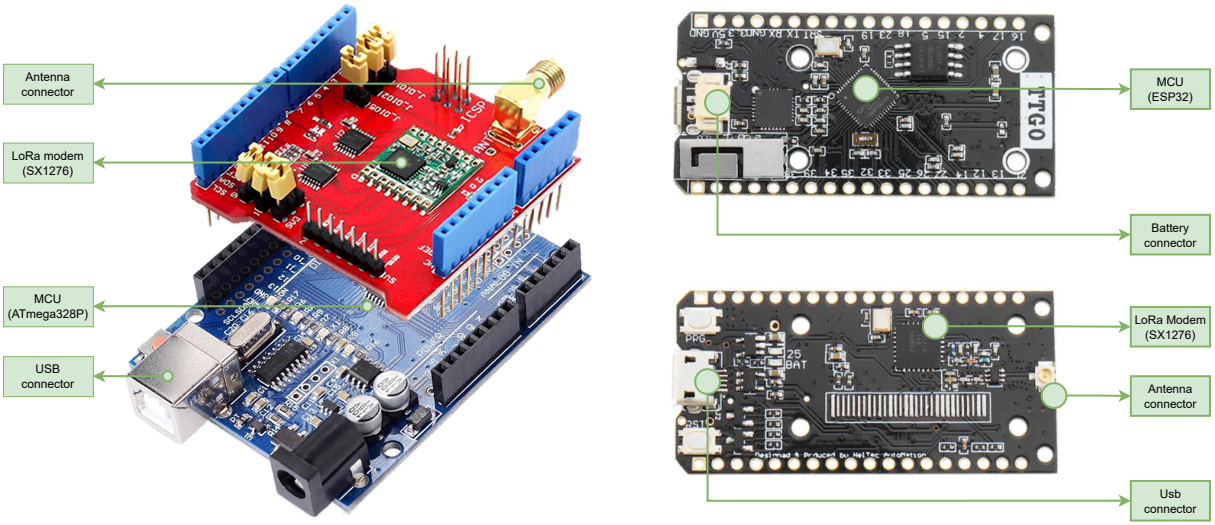
- Node sensors information;
- Node actuators information, including binary control options;
- Node statistics, including packets sent, average Received Signal Strength Indicator (RSSI), average SNR, and current battery level;
- Node RSSI and SNR plots over the packets sent;
- Terminal output for debug purposes.

Additionally, there is an input field for sending DL messages, a “Rescan Network” button that sends a status request to all nodes, an “Export Data” button that exports a Comma-Separated Values (CSV) file containing all the communications made since the interface boot time and a “Start Test” button which runs a predefined test on the network. This interface provided an important role in verifying the correct functioning of the network and in exporting all the data that is used in the performance plots shown in this chapter.

2.3 Hardware Equipment

The hardware equipment needed to deploy the WSN can be characterized by two main components: the processing unit and the radio transceiver. The processing unit is a Micro-Controller Unit (MCU) that can be the *ESP32* [38] or *Arduino Uno* [39] modules. As for the radio transceiver, the *SX1276* chip by *Semtech* was utilized [34]. The *Arduino Uno* was utilized as the gateway having the LoRa modem connected using the *Dragino LoRa Shield*. This assembly is depicted in Figure 2.9(a). Alternatively, the *TTGO ESP32 SX1276* was utilized as the platform for the nodes and can be seen in Figure 2.9(b).

The *SX1276* is equipped with the LoRa long range modem which has the main characteristics of providing ultra-long range spread spectrum communication and high interference tolerance whilst min-



(a) Arduino Uno (bottom) and Dragino Shield (top).

(b) TTGO ESP32 SX1276.

Figure 2.9: Hardware equipment used in the deployment of the network.

imizing current consumption and is ideal for applications requiring long range or robustness. It should be noted that the LoRa modem communicates using the available Serial Peripheral Interface (SPI). SPI is a synchronous serial data protocol used by MCUs for communication with one or more peripheral devices. With an SPI connection there is a master device (usually the MCU) which controls the peripheral devices. In order to do so, there are three lines common to all devices: Master In Slave Out (MISO) (slave line for sending data to the master), Master Out Slave In (MOSI) (master line for sending data to the peripherals) and Serial Clock (SCK) (the clock pulse for synchronization purposes); and one line specific to each device: Slave Select (SS) (allows each device to be controlled individually while sharing the aforementioned lines) [40].

2.4 Network Testing

In order to evaluate the performance and ensure the correct functioning of the WSN, a network was established with a gateway based on the *Arduino Uno* with the *Dragino SX1276 Shield* (Figure 2.9(a)) and 3 nodes based on the *TTGO ESP32 SX1276* (Figure 2.9(b)). It was tested the following metrics: RSSI, SNR, packet loss and packet delay. The sequence of steps consist in sending status request messages to the nodes sequentially and repeatedly. The nodes will respond to the gateway request by sending a corresponding status message. This allows the gateway to measure the RSSI and SNR values on the received messages and it allows the delay and packet loss to be calculated based on the corresponding sent and received messages.

2.4.1 Basic Experiment

The purpose of the basic test is to ensure the network works properly under ideal conditions and to establish a base-line. The first type of measurement was related to packet delay under different

circumstances. As seen previously (Section 2.2.2), packet delay is given by the TOA (T_{packet}). It should be noted that all communications in the network include an acknowledge message and therefore for each message sent, two packets are transmitted. The overall theoretical Round-Trip Delay (RTD) is therefore equal to $\text{RTD} = 2 \times T_{\text{packet}}$. Given this, the network was initialized with a bandwidth of $BW = 125$ KHz, a coding rate of $\text{CR} = 4/5$ and a spreading factor of $\text{SF} = 7$ which correspond to the default values for the *SX1276* modem (Table 2.2).

Parameter	Value
Spreading Factor	7
Bandwidth	125 KHz
Coding Rate	4/5

Table 2.2: LoRa modem initialization parameters.

The spreading factor was then varied across different values ($\text{SF} \in \{7, 9, 11\}$), as were the bandwidth ($\text{BW} \in \{125 \text{ KHz}, 250 \text{ KHz}\}$) and the coding rate ($\text{CR} \in \{4/5, 4/8\}$), with each test run having only one parameter changed from the initial configuration. Finally, the test was ran for each configuration, where 50 status request messages are sent to node n_1 (with a distance of $d = 40\text{cm}$ to the gateway) and the response status messages are received. Table 2.3 shows the results obtained.

CR	BW (KHz)	SF	Packet Loss(%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)	Theoretical RTD (ms)	RTD Diff. (ms)
4/5	125	7	0	-32.56	9.37	186.68	92.68	94.00
4/5	125	9	0	-32.84	12.81	457.54	329.72	127.82
4/5	125	11	0	-32.00	10.19	1426.18	1318.92	107.26
4/5	250	7	0	-33.26	10.13	133.46	46.34	87.12
4/8	125	7	0	-34.48	9.63	222.70	123.40	99.30

Table 2.3: Results obtained from sending the status request message to a node 50 times with different parameters. (Varied parameters shown in orange.)

As expected, the RTD follows the calculated theoretical RTD. Moreover, it can be concluded that if the range has to be extended by increasing the spreading factor, the RTD will also increase. Regarding the bandwidth, it also followed the theoretical RTD which makes sense given that an increase in bandwidth also increases the bit-rate, which in turn will decrease the delay. As for the coding rate, the results are also as expected. By including more redundancy bits in the ECC, the payload size increases and therefore the delay also increases. In this basic setup, there were no packets lost from the 50 sent messages. The RSSI and SNR are not dependent on the way the packet is transmitted as they are only affected by the conditions of the environment (like the distance to gateway, the existence of line-of-sight between the node and gateway and the amount of existing noise). Finally, there is an approximately

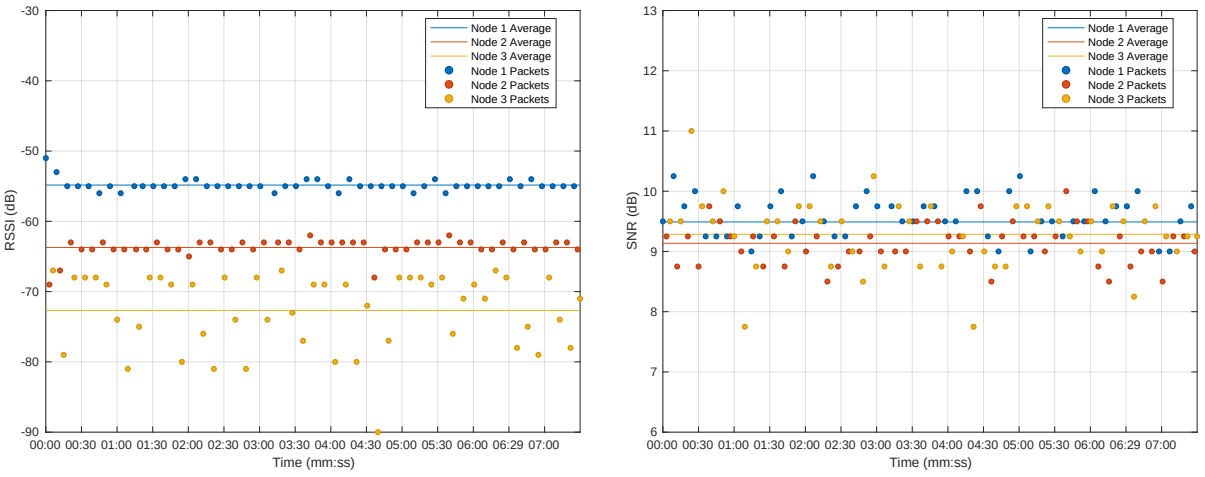
constant difference of approximately 100 ms in the measured RTD and the theoretical one due to the processing times on the MCUs.

A second test was performed with the same nodes in the network and following the parameters in Table 2.2 with 50 status request messages being sent to the nodes n_1, n_2 and n_3, sequentially and repeatedly. The goal of this test is to evaluate the behaviour of the network with multiple nodes. The results obtained are shown in Table 2.4.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	1	0	0	-54.84	9.49	204.04
2	4	0	0	-63.72	9.14	202.68
3	7	0	0	-72.68	9.29	203.34

Table 2.4: Results obtained from sending the status request message to each node 50 times sequentially and repeatedly.

Aligned with the same conclusions from the previous test, in ideal conditions, all the packets are successfully delivered to all three nodes resulting in a zero packet loss. Additionally, the RTD delay stays consistent with the value on Table 2.3 (regarding the initialization parameters) showing once again that the delay does not depend on the distance or other parameters apart from the ones used in 2.6.



(a) RSSI over time.

(b) SNR over time.

Figure 2.10: Results obtained from sending the status request message to each node 50 times sequentially and repeatedly.

One observation is the way that the RSSI value changes with distance. As the distance between a node and the gateway increases, the power of the transmitted signal is the same. However, by having to cover a larger distance in the medium, the signal loses more strength resulting in a received signal with a lower RSSI value. This can be seen in Figure 2.10(a) where the RSSI values are plotted for each node. Furthermore, as the distance increases, the RSSI deviation from the average also increases, as the

signal may be subjected to more interference. The values for the SNR are also plotted in Figure 2.10(b). Given that the SNR is a ratio between the signal to noise power, it is reasonable that for the small distances in this test we obtained high SNR. Moreover, the values in Figure 2.10(b), seem to follow a normal distribution around a constant value for each node, which can be caused by the noise interference in the medium since the signal power is constant for each node.

2.4.2 Rural Experiment

In this test, the aim is to evaluate the performance of the network in rural conditions. The nodes are located the farthest from the gateway (when compared to the basic and urban experiments) in this test so as to find the maximum range where communication is still possible. A location was selected where electromagnetic interference is minimal and where the nodes can be in line of sight of the gateway or with only some bushes or trees in the way while keeping a large distance to the gateway. In this way, the performance of the network can be evaluated through larger communication distances in both line of sight and lightly obstructed conditions.

Figure 2.11: Measurement location for the rural experiment. “A” and “B” represent gateway locations while the labels with the distances represent node locations. (Adapted from Google Maps.)

Depicted in Figure 2.11 are the device locations where the measurements took place. The letters “A” and “B” show the gateway locations and the distance values show the node locations. It should be noted that all node locations had line of sight to the gateway with exception of the closest one $(438\ \mathrm{m})$ in which a few trees were in the path.

The tests run in this experiment are similar to the ones in the basic experiment, where for each node, 50 status request messages are sent and responses are recorded. For each node location, a battery of 5 tests were run with different physical layer configurations where the relevant parameters were varied:

$\mathrm{SP} \in \{7, 9, 11\}$, $\mathrm{BW} \in \{125\ \mathrm{KHz}, 250\ \mathrm{KHz}\}$, $\mathrm{CR} \in \{4/5, 4/8\}$. Given this, and similarly to the previous experiment, the network was initialized with a bandwidth of $\mathrm{BW} = 125\ \mathrm{KHz}$, a coding rate of $\mathrm{CR} = 4/5$, and a spreading factor of $\mathrm{SF} = 7$, which match default settings for the *SX1276* modem (Table 2.2) and only one parameter is changed from the initialization per run.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	1.96	0	-98.76	8.16	186.30
2	612	3.84	0	-105.34	6.57	185.24
2	956	1.96	0	-106.42	4.66	185.20

Table 2.5: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=7.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	9.26	2	-100.18	8.70	458.39
2	612	0	0	-105.22	9.62	457.04
2	956	0	0	-106.00	5.84	456.92

Table 2.6: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=9.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	30.88	6	-100.60	8.65	1426.70
2	612	10.90	2	-106.12	7.01	1425.60
2	956	24.59	8	-106.59	3.95	1425.40

Table 2.7: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=11.

Tables 2.5 to 2.9 show in detail the results obtained from this experiment. For each node location and physical layer configuration, information is provided regarding the main performance evaluation parameters. Packet loss shows the percentage of packets that were lost during a run including any packet retransmissions. On the other hand, the message loss shows the percentage of lost messages out of the total 50. These two metrics help to measure how packet retransmission affects the overall performance of the network. Additionally, it is reported averages for the RSSI, SNR and RTD values for each test run.

Looking at the average RTD value for each run and physical layer configuration, it is noticeable that the RTD stays consistent regardless of the distance to the gateway, as expected from the basic

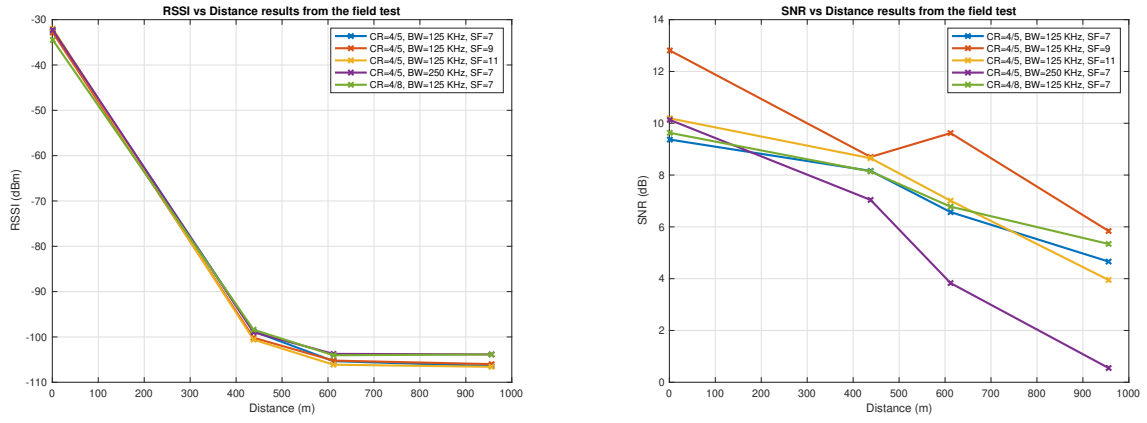
Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	0	0	-98.98	7.04	134.36
2	612	1.96	0	-103.74	3.83	133.12
2	956	1.96	0	-103.85	0.55	133.03

Table 2.8: Results obtained with the following physical layer configuration: CR=4/5, BW=250 KHz, SF=7.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	5.66	0	-98.40	8.14	223.58
2	612	0	0	-104.04	6.78	222.02
2	956	3.92	2	-103.86	5.34	222.04

Table 2.9: Results obtained with the following physical layer configuration: CR=4/8, BW=125 KHz, SF=7.

tests. Moreover, the values obtained here follow the values obtained on the basic experiment closely, reinforcing that the RTD value does not depend on the distance to the gateway but only on the physical layer configuration. Regarding the averages for RSSI and SNR, plots in Figure 2.12 help better compare the results across different physical layer configurations.



(a) RSSI over distance and physical layer configuration.

(b) SNR over distance and physical layer configuration.

Figure 2.12: Results obtained from the field test.

In Figure 2.12(a), it is shown that the tested physical layer configurations barely affect the strength of the received signal. This is expected as the RSSI value is determined only by the transmitted signal output power (the power at which the the LoRa modem transmits modulated signal), the antenna gain of the transmitter and receiver and by the gain loss due to the medium. As all these parameters are constant on each test, the RSSI value is expected to remain constant as result. Due to these same reasons, the plot clearly shows a decrease in the RSSI value as the distance to the gateway increases.

This happens due to the medium gain loss. In this case, there is a drop in the RSSI value from the 438 m test to the 612 m test but from the 612 m test to the 956 m test the RSSI value remains more or less constant. This can be explained due to the few trees that block the line of sight at the 438 m test. On the other hand, the remaining tests have line of sight to the gateway and therefore the signal is only blocked by the air reducing the gain loss of the medium when compared to obstacles such as trees. When it comes to SNR values, the results shown in Figure 2.12(b) describe how the SNR value varies over distance and across physical layer configurations. As expected, as the distance increases the amount of noise that the signal is subjected to also increases and the SNR decreases as a result. However, in this case the SNR value never goes below 0, showing that there is little to no noise on this rural environment, as there are no obstacles or other signals being transmitted causing interference. Regarding physical layer configurations, all of them follow the same trend of worse SNR values over distance.

Finally, Figure 2.13 shows data regarding the network performance on packet loss. Overall, the packet loss is higher on the 438 m test. Intuitively, it would be expected that packet loss increased with distance. This is generally true but it must be taken into account that the 438 m test does not have an entirely clear line of sight to the gateway and this is most likely the reason why the packet loss is higher on this test.

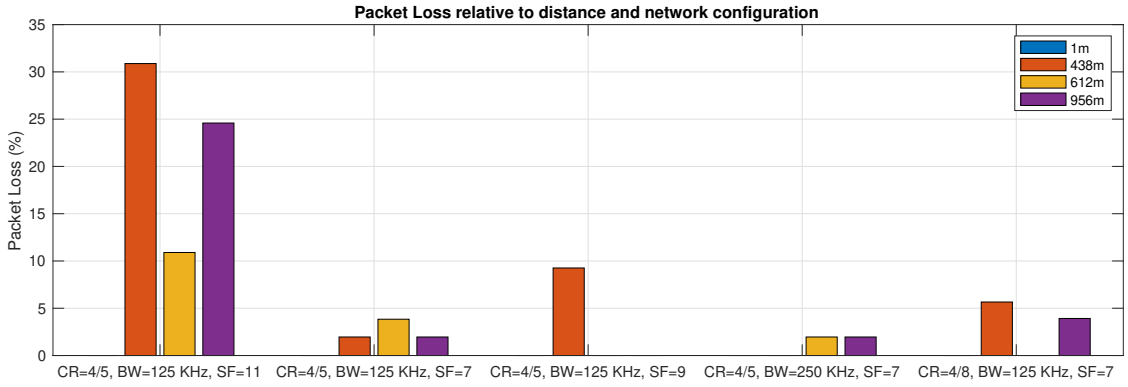


Figure 2.13: Results obtained from the field test regarding packet loss over distance and physical layer configuration.

In summary, the rural experiment showed communication to be possible with a range close to 1km while still maintaining a relatively strong signal. This points to the possibility of even larger distances being covered by the WSN. However, when compared to the advertised range of over 15 km, the results obtained fall short of that number. Additionally, an overall message loss of 1.33% was obtained given that out of the 50 messages per 3 nodes and per 5 physical layer configurations (750 in total) only 10 messages were dropped. Also in this subject, it should be noticed that the implemented packet retransmission capability plays an important role in reducing the message loss as often the message loss value is lower than the packet loss value proving how this feature improves the robustness of the network. When comparing the various tested physical layer configurations, interesting results were obtained: increasing the spreading factor actually lead to an increase in packet loss, which was not expected (the benefits of increasing the spreading factor may be more visible at larger communication

distances); additionally, changing the coding rate and bandwidth had little effect on packet loss. In this way, the configuration which best suits a rural environment is: a bandwidth of $BW = 125$ KHz, a coding rate of $\mathrm{CR} = 4/5$ and a spreading factor of $\mathrm{SF} = 7$.

2.4.3 Urban Experiment

The rural experiment is the most interesting with respect to the envisioned application of the surveillance system being designed in this thesis. Nevertheless, it is also possible that some cases of areas to be inspected can have buildings or target surveillance of events in cities. Therefore, an urban case was also experimented which provide other researchers with a complete picture in terms of communication characteristics in a real-life test of both a rural and urban scenarios. Thus, this test is conducted at the Alameda campus of Instituto Superior Técnico (IST) in the center of Lisbon. The campus counts with two 12-story towers covered in glass and smaller buildings. The gateway was placed on the 8^{th} floor of the IST North tower close to a south-oriented window with the test node being moved over 5 different locations, some of which do not have line of sight to the gateway. This information is depicted in Figure 2.14.

Figure 2.14: Measurement location for the urban experiment. “G” represents the gateway location while the other letters represent the test node location. (Adapted from Google Maps.)

Figure 2.14 shows the location used for this experiment along with the node and gateway locations where the measurements took place. It should be noted that the location point **E** is on the 5^{th} floor of the North tower while the remaining positions are at ground level. The remaining points were selected since they offer different conditions in terms of line of sight to the gateway. The location point **A** has direct line of sight to the gateway while **B** and **C** have a slightly obstructed view. Positions **D** and **E** are inside a

building and, therefore, completed obstructed.

In comparison with the rural experiment, the urban test also compares the effect of two different antennas on the node device while sending 25 messages each (maintaining a 50-message test per configuration and per location). The battery of 5 tests encompass different physical layer configurations where the relevant varied parameters are: $\mathrm{SP} \in \{7, 9, 11\}$, $\mathrm{BW} \in \{125\ \mathrm{KHz}, 250\ \mathrm{KHz}\}$, $\mathrm{CR} \in \{4/5, 4/8\}$. Given this, and similarly to the previous experiment, the network was initialized with a bandwidth of $\mathrm{BW} = 125\ \mathrm{KHz}$, a coding rate of $\mathrm{CR} = 4/5$, and a spreading factor of $\mathrm{SF} = 7$, which match the default settings for the *SX1276* modem (Table 2.2) and only one parameter is changed from the initialization per run.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-96.64	7.64	185.60
A	2	171	0	0	-102.04	5.98	184.36
B	1	73	0	0	-95.52	8.42	204.44
B	2	73	0	0	-101.80	6.95	203.16
C	1	193	0	0	-93.08	8.54	185.76
C	2	193	8	8	-103.78	6.81	184.52
D	1	110	8	8	-105.17	3.79	186.18
D	2	110	76.67	52	-110.57	-4.71	184.43
E	1	34	40	40	-103.27	0.32	185.67
E	2	34	81.25	76	-105.50	-3.50	184.50

Table 2.10: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=7.

Tables 2.10 to 2.14 show the results obtained from this experiment in a similar fashion to the rural case reporting the main performance evaluation parameters: packet loss, message loss, average RSSI, average SNR and average RTD.

Regarding RTD, the results are as expected. For each physical layer configuration the delay stays approximately constant. Moreover, the values obtained for each physical layer configuration are close to the ones obtained when testing two devices in close proximity. When comparing the RTD values for both antennas, it can be concluded that the delay times are independent as the RTD is the same. With respect to RSSI and SNR, Figure 2.15 helps visualizing the results.

In Figure 2.15(a) the behaviour of the RSSI is shown against multiple communication distances. Furthermore, to widen the scope of this analysis, two scenarios were considered: an indoors scenario (where both the gateway and the node are inside a building) and an outdoors scenario (where only the gateway is inside a building). Immediately, a few observations can be made. The RSSI values for the node with the small antenna are consistently lower than the ones for the node with the big antenna.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-97.88	7.40	457.88
A	2	171	4	4	-103.08	5.04	456.54
B	1	73	0	0	-95.12	8.96	457.50
B	2	73	3.84	0	-101.56	8.09	456.15
C	1	193	0	0	-96.52	9.15	457.44
C	2	193	0	0	-106.76	5.93	456.24
D	1	110	0	0	-104.88	3.35	457.76
D	2	110	82.14	80	-109.80	-8.80	456.00
E	1	34	48	48	-102.23	0	457.93
E	2	34	76	76	-104.67	-6.50	456.33

Table 2.11: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=9.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	7.40	0	-98.28	7.13	1781.20
A	2	171	0	0	-104.12	5.85	1765.70
B	1	73	3.84	0	-96.40	8.17	1426.20
B	2	73	0	0	-101.88	7.40	1425.10
C	1	193	0	0	-97.76	7.23	1998.20
C	2	193	3.84	0	-106.68	6.84	1998.50
D	1	110	12	12	-108.04	2.92	1795.80
D	2	110	85.19	84	-113	-3.25	1795.80
E	1	34	32	32	-104.41	0.06	1426.10
E	2	34	68	68	-105.88	-4.37	1425.10

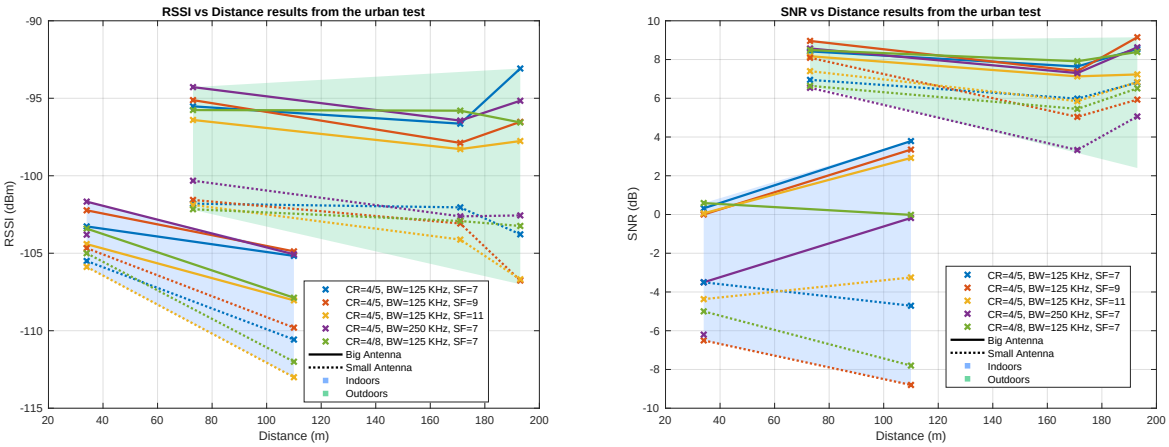
Table 2.12: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=11.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-96.44	7.30	133.76
A	2	171	4	4	-102.62	3.33	132.37
B	1	73	0	0	-94.28	8.57	133.96
B	2	73	0	0	-100.32	6.54	132.44
C	1	193	0	0	-95.16	8.63	133.72
C	2	193	0	0	-102.56	5.06	132.28
D	1	110	46.15	44	-105.0714	-0.18	133.71
D	2	110	100	100	-	-	-
E	1	34	77.78	76	-101.67	-3.50	134.33
E	2	34	89.36	80	-103.80	-6.20	132.20

Table 2.13: Results obtained with the following physical layer configuration: CR=4/5, BW=250 KHz, SF=7.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-95.80	7.91	222.80
A	2	171	0	0	-102.92	5.45	221.48
B	1	73	0	0	-95.76	8.49	222.84
B	2	73	3.84	0	-102.16	6.64	221.44
C	1	193	0	0	-96.56	8.39	222.88
C	2	193	0	0	-103.24	6.51	221.52
D	1	110	54.55	40	-107.87	-0.02	222.67
D	2	110	85.29	80	-112	-7.80	221.60
E	1	34	32	32	-103.41	0.59	222.76
E	2	34	96.15	96	-105	-5	222

Table 2.14: Results obtained with the following physical layer configuration: CR=4/8, BW=125 KHz, SF=7.



(a) RSSI over distance and physical layer configuration.

(b) SNR over distance and physical layer configuration.

Figure 2.15: Results obtained from the urban test.

Since the two nodes stay at the exact same place, the medium is the same for both of them and the difference in the RSSI values is most likely caused by different output powers. Given that the same LoRa modem transmission power is selected for both nodes, it means that the larger antenna has a higher gain. When it comes to node location, it is also clear that the indoor test runs yielded worse results in comparison with the outdoor setting. This is once again to be expected as there is no line of sight to the gateway and there are several walls and other obstacles in the signal path. Finally, the same conclusions taken from the previous experiment also apply here: RSSI has a small dependence on the physical layer configuration and overall it gets lower when the distance to the gateway increases.

As for the SNR, Figure 2.15(b) shows its behaviour for different communication distances. Similarly to the RSSI, the SNR tends to be lower for the small antenna. This should be explained by the fact that with the small antenna there is a lower signal output power resulting in a higher noise level by comparison. Again, in the indoor test runs, the results are worse. This happens due to the additional noise and interference caused by all the obstacles to the signal propagation. One interesting observation is that on the test run in location **D**, the SNR values obtained were generally better in comparison to using location **E** (even though location **E** was further away from the gateway). This could be explained by the difference in the medium through which the signal must pass. In location **E**, the node was three floors below the gateway, resulting in concrete obstacles while **D** is inside a different building (the buildings had line of sight to each other), leading to less signal interference.

Figure 2.16 shows the packet loss for all the test locations across multiple physical layer configurations. An expected trend is the higher packet loss for indoor test (34 m and 110 m) and the use of the small antenna since a lower antenna gain generates a lower RSSI. Regarding message loss, it is noted how this value is often lower than the packet loss, showing how packet retransmission is improving the network reliability.

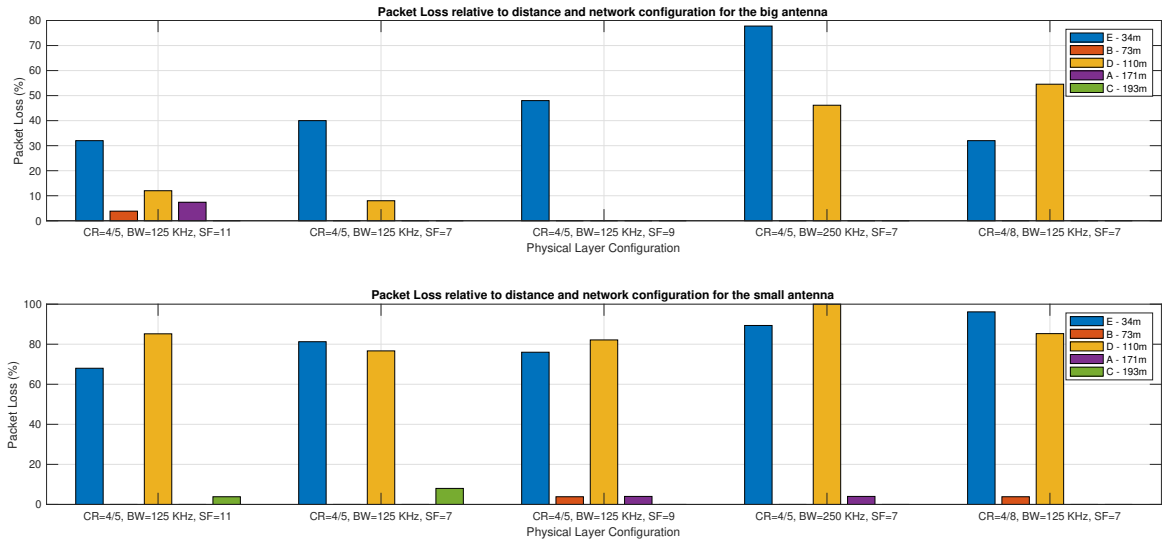


Figure 2.16: Results obtained from the urban test regarding packet loss over distance and physical layer configuration.

To sum up, the urban experiment explored the capabilities of the developed LoRa based WSN in an urban environment, which can be quite challenging for a network using LoRa as its high efficiency property makes use of a very low signal transmission power. In addition, two different antennas were also tested. The results in this experiment showed how communication is possible from devices in different buildings and how well the network performs in better conditions, such as when the node is outside and with line of sight to the gateway. Similarly to the rural experiment, the default physical layer configuration appeared to show the best results overall. With this configuration a maximum packet loss of $40\ \%$ for the big antenna and $81.25\ \%$ for the small antenna was obtained for communication between devices in two different buildings with a communication distance of up to $110\ \mathrm{m}$. For communication where only the gateway is inside a building and with a communication distance of up to $193\ \mathrm{m}$, a maximum packet loss of $0\ \%$ and $8\ \%$ was obtained for the big and small antennas respectively.

2.5 Discussion

In this chapter, after reviewing the main wireless communication technologies, LoRa stood as the most suitable for the envisioned scenario, albeit requiring the implementation of a protocol to support the type of WSN required for the surveillance system. The extensive testing of the proposed WSN is one of the main contributions of this chapter, which can further contribute to other researchers interested in the topic since the data provided is closer to a real setting than the information provided by the manufacturers. Three experiments were conducted to cover the main cases of use for the communication: a base line, a rural experiment and an urban setting. In the rural context, communication was possible for distances of over $1\ \mathrm{km}$, even when the physical layer configuration is not set up for maximum range.

Additionally, this large reach is achieved with an RTD delay of down to around $133\ \mathrm{ms}$.

Running a WSN in a rural context has the advantage of having direct line of sight for most of the devices. Whenever such a network is established in an urban environment, the LoRa modulation technology decreases in its performance. However, the data from the tests showcases promising results with successful communications even from inside different buildings at a distance of around $110\ \mathrm{m}$. Moreover, it was clarified the relationship in practice of the RSSI and SNR decrease with distance and partial or complete obstructions.

The research presented in this chapter sets forth some interesting possibilities in terms of future work, namely: the implementation of UL time slots (assigned periods of time where each device can transmit) to avoid multiple nodes trying to transmit at same time when using a single channel gateway (this is not a problem in the current sparse communication pattern in the network); the implementation of DL time slots to minimize the node up-time and decrease power consumption and exploring the possibility of a multi-channel gateway. Both developments can benefit the performance of the WSN and the power usage of nodes, leading to a more robust network suitable for deployment over remote areas where exchanging a node battery is a complicated endeavour.

Chapter 3

Wireless Sensor Network Integration with Autonomous Vehicles

In this chapter, the developed WSN is integrated with autonomous aerial vehicles in order to tackle the surveillance system with the characteristics outlined in Section 1.2. In a first step, simulations were conducted in a high-fidelity software detailed in Section 3.1. Following that, the WSN network is integrated with the ROS environment to allow conducting event-triggered missions by the vehicles with the alerts coming from the sensors with the details for the communication being presented in Section 3.2 and those related to the control task being developed in Section 3.3. Finally, the system is experimentally tested and results are analysed in Section 3.4.

3.1 Simulation Environment

The setup of the simulation environment was guided by the real flying arena as well as the previous work by Oliveira et al. [41, 42]. The complete architecture of the simulation environment is depicted in Figure 3.1. Prior to explaining the several utilized simulation modules, the working environment itself should be characterized. The operating system consists of the *Ubuntu 18.04LTS* version with the *melodic* version for the ROS installation [43]. In the remainder of this section, each block shown in Figure 3.1 will be analysed to better convey the simulation flow.

3.1.1 Gazebo Simulator

Gazebo [44] is a robotics simulator with a physics engine modeling the configurable dynamic and kynematic properties for each vehicle. Furthermore, it allows for multiple types of sensors and actuators to be included through plugins. In essence, *Gazebo* computes the motion of the vehicle along with the new sensor readings, from the inputs given to the vehicle. The main advantage of *Gazebo* is its integration with the ROS middleware, which in turn offers a modular definition of the overall system by defining each entity as a separate package and enabling communication through the concept of

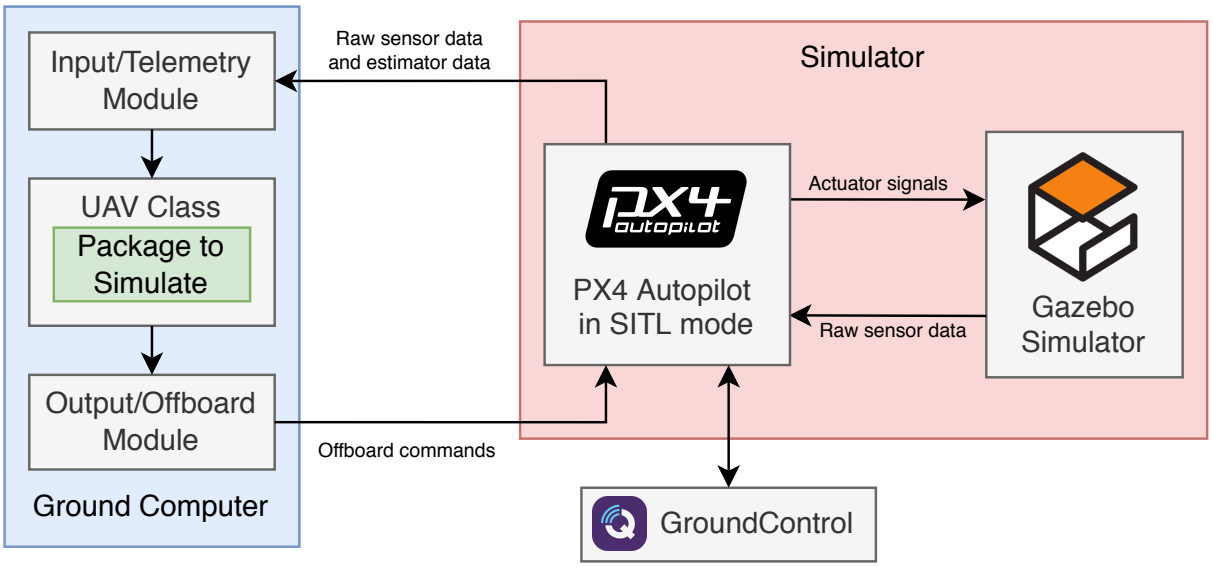


Figure 3.1: Architecture of the simulation environment.

publishing and subscribing messages to topics and/or services.

3.1.2 *PX4* Autopilot in SITL mode

PX4 [45] is an autopilot firmware which can run on board of the vehicle both as Hardware In The Loop (HITL) or Software In The Loop (SITL) modes. *PX4* acts as an interface between the offboard modules and the actuators of the vehicle providing sensor readings, actuators control, autonomous controllers and estimators, safety features and communication with external systems through the lightweight *MAVLink* messaging protocol [46]. In this block, *PX4* provides raw sensor and estimator data (i.e.: pose estimation from the use of an Extended Kalman Filter (EKF) filter with the data from the onboard Inertial Measurement Unit (IMU)) and accepts offboard commands to control the vehicle. Additionally, *PX4* already has a set of available quadrotor models, including the *Iris* [47] quadrotor that will be used in the simulations. It should be noted that, if using the *PX4* autopilot capabilities, the simulations can be transferable to other UAVs as long as they have the same *PX4* firmware. Finally, the translation layer *MAVROS* [48] was used as an interface between the autopilot and the ROS middleware.

3.1.3 Ground Computer and QGroundControl

The Ground Computer stack (shown inside the blue rectangle in Figure 3.1) is what allows the user to test different control algorithms and run missions with the vehicles. The **UAV class** represents a vehicle and is available due to the input and output modules developed by Oliveira [41] and adapted by Jacinto [49]. The input or telemetry module is responsible for subscribing to the information published by the *PX4* and the Motion Capture (MOCAP) system and making such information available to the user in a standardized way. The output or offboard module deals with storing the methods that allow the user to send offboard commands and control references to the *PX4* [41].

Moreover, the *QGroundControl* [50] application is a GUI that provides full flight control and vehicle setup through the *MAVLink* communication protocol. It has features like mission planning for autonomous flight, flight map display showing vehicle position, flight track, waypoints and vehicle instruments and video streaming capabilities.

3.1.4 Running Simulations

In order to run simulations, there are a number of launch files that run all the necessary services. These launch files allow services like *Gazebo* to be started within the ROS environment, which enables further access to variables related to the state of the world and simulation. Depicted in Figure 3.2 is the hierarchic process of running a simulation with all called launch files.

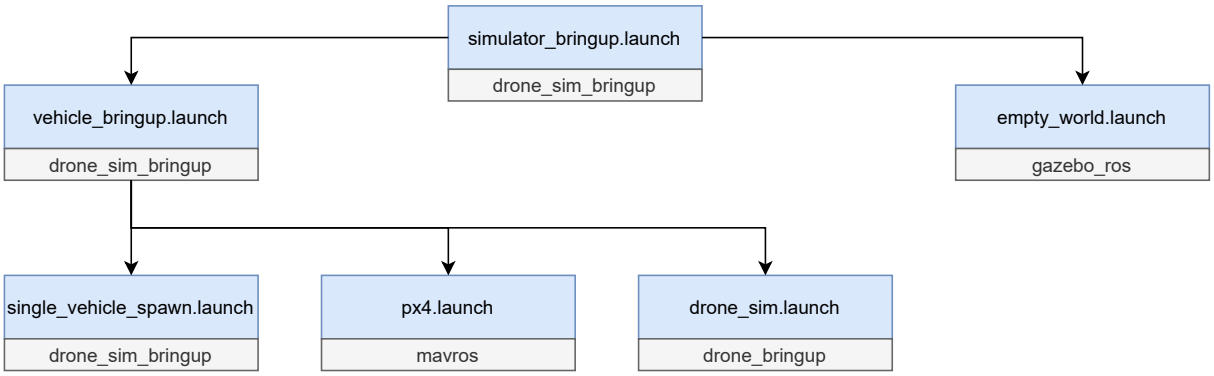


Figure 3.2: The hierarchic process of running a simulation. Each launch file belongs to the packet named below file name.

Using a top-down approach, the **simulator_bringup.launch** is the main initialization file. When called with the command "roslaunch drone_sim_bringup simulator_bringup.launch", it will open *Gazebo* and spawn the world and one UAV instance.

The **simulator_bringup.launch** executes two additional launch files: one to spawn the world in gazebo and another to populate it with a vehicle. The former is the **empty_world.launch**, which is the standard simulation launcher provided by the ROS Application Programming Interface (API) for *Gazebo* and accepts multiple arguments to control its different configurations. The main ones to highlight are: the binary argument **gui** selects whether to start *gzclient* (its GUI) when initializing the mandatory *gzserver* (its physics engine); and, **world** which is the model of the world to be spawned. The latter is the **vehicle_bringup.launch** which is responsible for spawning the vehicle in *Gazebo* as well as starting a *PX4* instance for that vehicle. In order to accomplish this, it resorts to three additional launch files. **The single_vehicle_spawn.launch** file i) generates a Simulation Description Format (SDF) model (SDF is used to describe objects and environments for robotics simulators) for the vehicle, starts a *PX4* instance with the passed arguments and correct configurations: **px4_config** and **px4_name_config**; and, ii) spawns the vehicle in *Gazebo* in the desired position and attitude. It should be noted that this file can be called individually to spawn more UAV instances. The **px4.launch** file from the mavros package is responsible for the *MAVROS* configurations and the **drone_sim.launch** is the file that loads all the user defined

configurations in YAML format, a human-friendly data serialization language, (i.e.: names for the ROS topics and values for controller constants) and runs the package which the user intends to simulate.

To this extent, the simulation process is as follows: the ROS simulation package itself is developed; the drone_bringup package is updated by changing the necessary configurations in the config.yaml files and updating the drone_sim.launch to call the desired package; finally, the simulation is started with the command “roslaunch drone_sim_bringup simulator_bringup.launch”.

3.2 Wireless Sensor Network for Surveillance and ROS Integration

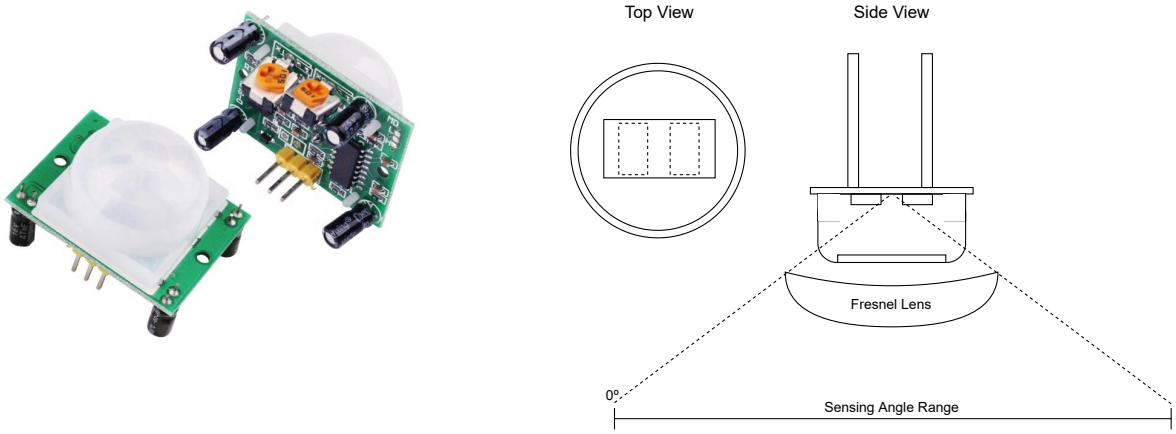
Following the work developed in Chapter 2, the surveillance network consists of a gateway (g_1) and three node devices (n_1, n_2 and n_3). Similarly to the testing section in the previous chapter, the hardware used for the gateway is an *Arduino Uno* module and the *Dragino LoRa Shield*. For the node devices, the *TTGO ESP32 SX1276* modules were used.

To detect events to be subject of surveillance using the UAV, each node is equiped with a *HC-SR501* PIR motion sensor, given its adequacy for battery operated devices and included automatic control module. The main specifications for this sensor are detailed in Table 3.1. Moreover, this sensor main applications include security products, which is closely related to the surveillance aspect of this project. The *HC-SR501* [51] module consists of the *BISS0001* [52] Micro Power PIR Motion Detector Integrated Circuit (IC) (specifically developed to process the signal from PIR motion sensors) and the *LHI778* [53] pyroelectric sensing element as depicted in Figure 3.3. This type of sensor works based on the principle that every body with temperature will emit infrared radiation in the form of heat. PIRs use two sensing elements to read the infrared levels in the environment. When a body enters the sensor’s field of view, it first causes a positive differential between the two sensing elements due to the difference in temperature. Analogously, when the body leaves the sensing area, the reverse happens leading to a negative differential. In this way, motion is detected through these differentials in the pyroelectric sensing elements readings.

Parameter	**Value**
Operating Voltage	4.5-20 V
Power Consumption	65 mA
Sensor Angle	110 ° cone angle
Max. Sensing Distance	3-7 m
Delay Time	5-300 s

Table 3.1: Main specifications for the *HC-SR501* motion sensor.

The sole task of each node is to generate an UL message every time motion is detected and sent it through the LoRa network (depicted in Figure 3.4) to the base station corresponding to a *Nvidia Jetson AGX Xavier* developer kit [54] referenced in Figure 3.4. (It should be noted that this board was used



(a) *HC-SR501* image.

(b) PIR motion sensor sensing element diagram.

Figure 3.3: Image of the *HC-SR501* motion sensor and diagram of its sensing element.

since it was already available but cheaper alternatives like the *Nvidia Jetson Nano* are more suitable given the low cost approach of this project.) This board is essentially a small computer running an 8-core ARM CPU and a 512-core Volta GPU, allowing to run the WSN software and a process for the ROS environment used to control the UAV. Given its size, energy consumption and processing power, make it adequate to serve as a base station for this surveillance system.

3.2.1 Sensor Testing and Calibration

To ensure the correct operation of the motion sensor, it has been tested under different configurations and calibrated to ensure the desired behaviour. The *HC-SR501* includes two potentiometers capable of adjusting the sensing distance as well as the delay between motion triggers. In this way, a program was created to read the output of the sensor as well as to measure the delay of the sensor output change from 1 to 0.

Parameter	**Value (5 measurement average)**
Max. Sensing Distance (min.)	9 m (center), 6 m (edge)
Max. Sensing Distance (halfway)	9 m(center), 7 m (edge)
Max. Sensing Distance (max.)	10 m (center), 9 m (edge)
Delay Time (min.)	3.84 s
Delay Time (halfway)	107.32 s
Delay Time (max.)	210.54 s
Angle Range	$\sim$106 °
10s Continuous Motion Delay Time (min.)	14.56 s

Table 3.2: Testing results for the *HC-SR501* motion sensor.

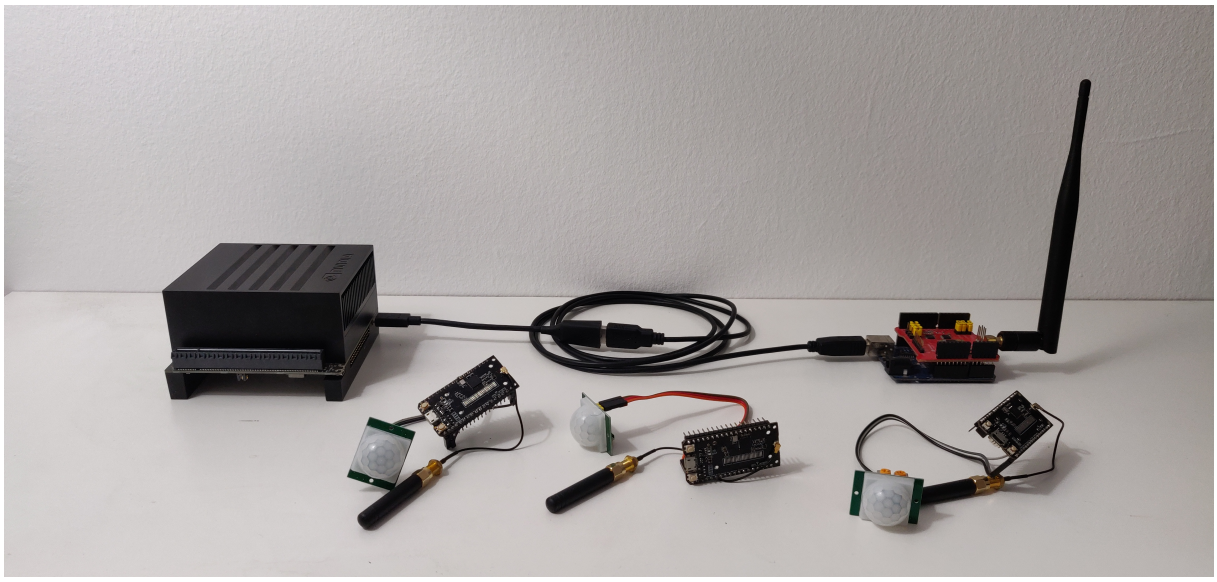


Figure 3.4: Image of the hardware used by the WSN. At the front there are the three node devices used. At the back there is the gateway on the right and the *Nvidia Jetson AGX Xavier* developer kit on the left.

Represented in Table 3.2 are the results obtained from the motion sensor testing. When it comes to sensing distance, the results were quite different from the specifications, which could be caused by the manufacturing tolerances. Even when dialed to the minimum setting, the maximum sensing distance was $6\ \mathrm{m}$, in contrast to the specified $4\ \mathrm{m}$. In the same way, with maximum sensing distance selected, a value of $10\ \mathrm{m}$ was recorded, in contrast to the specified $7\ \mathrm{m}$. Moreover, different values were acquired depending on the location of the movement in terms of angle to the sensor. When movement was made in the center of the sensing field, the maximum sensing distance was larger than when the movement was made at the edge. With respect to delay, the results were closer to the specifications but still had some deviation, especially in the maximum delay. The minimum delay time can be set to approximately $4\ \mathrm{s}$ and the maximum delay time can be set to approximately $211\ \mathrm{s}$ as opposed to a specification of $5\ \mathrm{s}$ and $300\ \mathrm{s}$, respectively. Regarding the angle range, the results were also close to the specification with a measured difference in range of approximately 4° which is within the measurement errors. Finally, it was also tested when movement continues past the sensor triggering time to understand how this situation is managed and the conclusion is that the delay is restarted whenever motion is detected.

With the aforementioned information, the desired behavior for the setup of the sensor must be within the specific bounds found. It should be remarked that the sensor requires around 1 minute to initialize and adjust to the ambient infrared levels when powered up. Moreover, the sensor is affected by the wind (given that big gusts of wind can affect the infrared temperature levels) and light levels in the environment and there is also a fixed blocking time of $2.5\ \mathrm{s}$ after the sensor output goes from 1 to 0 where motion cannot be detected.

3.2.2 ROS Integration

To allow for communication between ROS nodes and the gateway, the previously network manager GUI application was adapted into a ROS node. Additionally, it was modified so that all received UL and DL messages are relayed into two distinct ROS topics, allowing WSN data to become available by other nodes on the ROS environment.

3.3 Final System and Mission Flow

The final surveillance system is composed of the integration between the WSN and the UAV. More specifically, the WSN nodes equipped with motion sensors and the ROS package stack that reads the data and creates a mission as a sequence of waypoints to be inspected by the UAV. The architecture of this system is described in Figure 3.5.

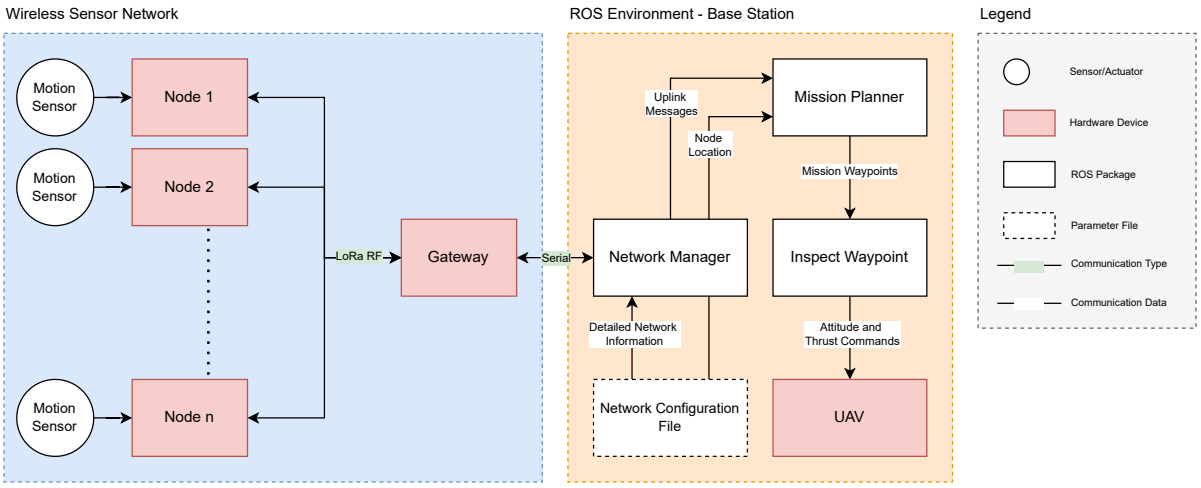


Figure 3.5: Final System Architecture Diagram.

According to Figure 3.5, there are three ROS packages implemented. The **Network Manager**, as previously described, is responsible for monitoring the network as well as for the bidirectional communication to the gateway. The **Mission Planner** is tasked with receiving UL messages relayed from the **Network Manager** and transforming them into a mission for the UAV. Finally, the node location is sent to the **Inspect Waypoint** package that instructs the UAV. Thus, the mission flow can be summarized as follows: i) the UAV is ready to takeoff and awaiting for instructions; ii) a motion sensor is triggered; iii) an UL message is sent to the gateway; iv) the gateway relays that message to the **Network Manager**; v) the **Network Manager** injects the received message into the relevant ROS topic; vi) the **Mission Planner** receives the UL message on the relevant ROS topic, gets the node location from the configuration file and sends a waypoint mission to the **Inspect Waypoint** package via a service call; vii) finally, the **Inspect Waypoint** package receives a mission and visits all the waypoints before returning to the charging station and landing.

3.4 Experimental Validation and Results

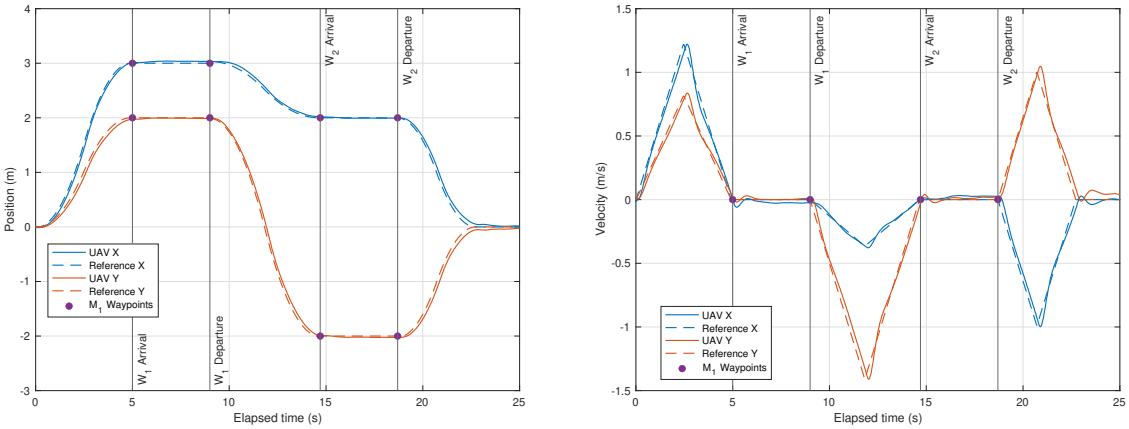
In order to assess the performance and characteristics of the surveillance system, both computer simulations and real tests were conducted. The network was implemented according to Section 2.4 and initialized with a bandwidth of $\mathrm{BW} = 125$ KHz, a coding rate of $\mathrm{CR} = 4/5$ and a spreading factor of $\mathrm{SF} = 7$. This configuration was chosen as the flying arena is not a very large space (so a lower SF is beneficial) and given it had already been tested with good performance. As for the UAV, the *Iris* [47] quadrotor was used for the simulations and the *Aero* [55] quadrotor was used for the real trials.

3.4.1 UAV Waypoint Following Simulation

In order to prepare the experiments in the flight arena, simulations involving the UAV were run using the environment described in Section 3.1. Given that the focus of this thesis is on the WSN and on its integration with autonomous vehicles using the ROS middleware, a simple package for waypoint visiting was written relying on the *PX4* autopilot. In this test, the **Inspect Waypoint** package defines the ith waypoint denoted by w_i in the format $w_i = \{x_i, y_i, \tau_i\}$ where x_i and y_i are the site i coordinates and a staying time τ_i in seconds for the UAV to inspect w_i. A mission can be executed by sending a message to the service **/$uav_id/topics/services/start_mission** (where $uav_id is a string identifying the UAV) with a vector m with the concatenation of all n waypoints, i.e., $m = [w_0, \cdots, w_n]$. Summarizing, the mission starts with the UAV taking off, visiting each waypoint in m and waiting the requested amount of time, and then returning to the take-off location and landing. If the **Inspect Waypoint** sends a new mission, the current one is aborted and the UAV starts executing the new list of waypoints. In order to execute these waypoint missions the internal capabilities of the *PX4* autopilot could be used but with the goal of later having more sophisticated trajectories implemented, it is useful to include a controller in the developed ROS package. The UAV controller is a Proportional, Integral, Derivative (PID) from the examples in [41] with a reference trajectory fed to the controller using uniform accelerated motion expressed in (3.1), where s is the displacement, v is the velocity, a is the desired acceleration and t is the time

$$\begin{aligned} s &= s_0 + v_0 \cdot t + \frac{1}{2} a_0 \cdot t^2, \\ v &= v_0 + a_0 \cdot t, \\ a &= a_0. \end{aligned} \tag{3.1}$$

The UAV follows a straight line between the starting position and the waypoint with acceleration in the first half segment connecting the two and decelerating otherwise. In the simulations shown, an acceleration value of $a_0 = 0.5\ \mathrm{m/s^2}$ was used. The first case is a multi-point test and executes the mission $m^1 = [\{x_0 = 3, y_0 = 2, \tau_0 = 4\}, \{x_1 = 2, y_1 = -2, \tau_1 = 4\}]$. The results obtained are depicted in Figures 3.6(a) and 3.6(b) by showing the position and velocity over time and in Figure 3.8(a) by showing the executed trajectory.

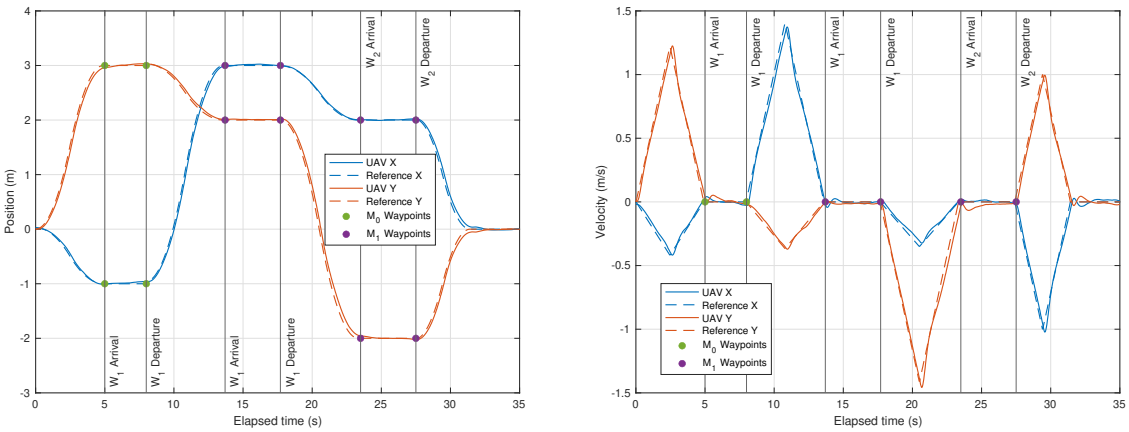


(a) Plot of the UAV position and reference over time.

(b) Plot of the UAV velocity and reference over time.

Figure 3.6: Results obtained from running the multi-point test mission.

On the other hand, the second simulation shows the behaviour of the UAV when a new mission is requested amid a current operation. The missions objects were defined as a single-point test $m^0 = [\{x_0 = -1, y_0 = 3, \tau_0 = 3\}]$ followed by the multi-point test mission m^1. This results in the UAV changing its current mission and head for the new next waypoint. The results obtained are once again depicted in Figures 3.7(a) and 3.7(b) by showing the position and velocity over time and in Figure 3.8(b) by showing the executed trajectory.



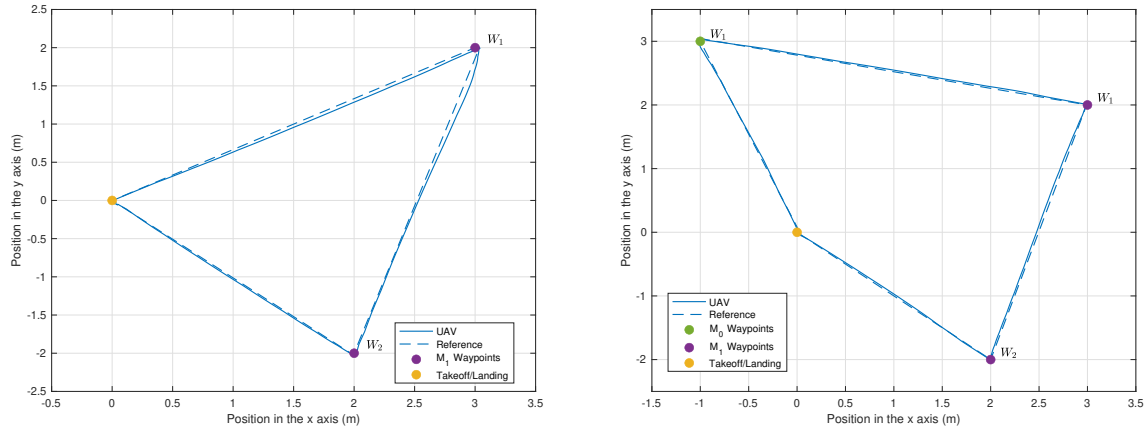
(a) Plot of the UAV position and reference over time.

(b) Plot of the UAV velocity and reference over time.

Figure 3.7: Results obtained from running the multi-point test shortly after running the single point test.

A few observations can be made regarding the time evolution of the position and velocity. Firstly, it is noticeable how the UAV converges on the waypoints and stays there for the designated amount of time. Additionally, there is no noticeable overshoot or oscillation present in the UAV trajectory when compared to the reference trajectory. This happens due to the use of an already tested controller and a reference trajectory using uniform accelerated motion expressed in (3.1), where a constant acceleration of $a_0 = 0.5\ \mathrm{m/s^2}$ allows a smooth evolution of the position.

When it comes to the trajectory plots, these show the position values of both the reference and the UAV trajectories. The plots for both simulations (Figures 3.8(a) and 3.8(b)) show consistent information, with the UAV following the reference trajectory with precision and visiting all the mission waypoints.



(a) Results obtained from running the multi-point test mission.

(b) Results obtained from running the multi-point test shortly after running the single point test.

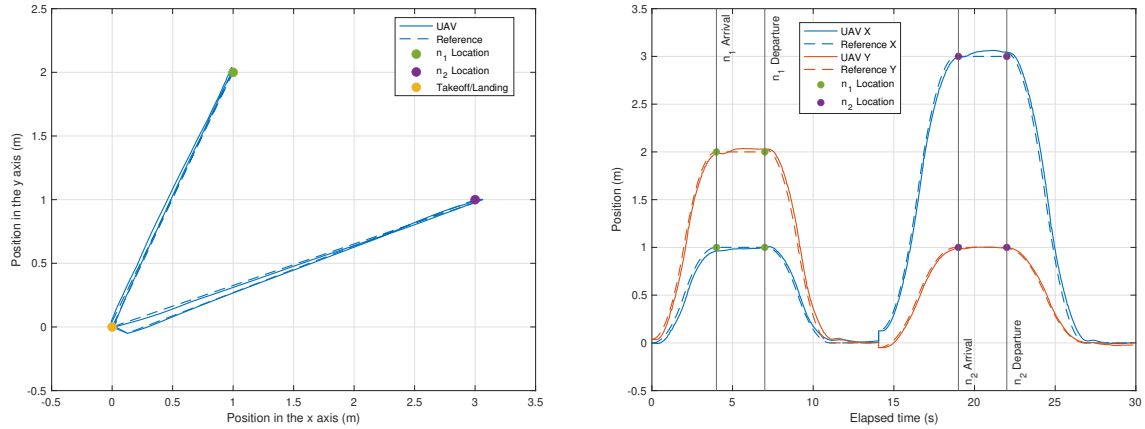
Figure 3.8: Plot of the UAV position, reference and waypoints.

This simulation shows that the UAV has a high accuracy and can change missions during their execution without large deviations from the reference.

3.4.2 Final Surveillance System Simulation

This simulation consists on the integration of the sensor events with the UAV mission. It should be noted that beforehand the created motion sensor network was tested within the ROS environment to ensure it was working properly at this stage. The goal is to trigger a motion sensor and observe the simulated UAV trajectory to the node position. This is an important test to ensure the correct functioning of both the WSN and the UAV while working together under the ROS environment. In this way, two node devices were used and the "physical layer configuration File" was updated with the node locations. Let p be the location of a node defined as $p = [x, y]$. Node n_1 was placed at location $p_{n_1} = [1, 2]$ and node n_2 was placed at $p_{n_2} = [3, 1]$.

The test procedure consisted in triggering the motion sensor on node n_1, waiting for the UAV to inspect and return to the takeoff location and land, followed by a trigger in n_2. The results concerning the UAV position is depicted in Figure 3.9(a). In more detail, Figure 3.9(b) shows the UAV flying to the location of n_1, then to the takeoff area followed by a visit to n_2 and back.



(a) Plot of the UAV position, reference and nodes location.

(b) Plot of the UAV position and reference over time.

Figure 3.9: Results obtained from the simulation with the motion sensors.

3.4.3 Final Surveillance System Experiment on the Flight Arena

In order to experimentally validate the proposed surveillance system, we conducted a real trial in the Institute for Systems and Robotics (ISR) flight arena at the IST *TagusPark* facilities using the *Intel Aero Ready to Fly* quadrotor (Appendix D). The flight arena has an area of approximately $7\ \mathrm{m}$ by $4\ \mathrm{m}$, which poses some difficulties to test a larger WSN. In this experiment, node n_1 was placed in the location $p_{n_1} = [-2, 0]$, node n_2 was placed in the location $p_{n_2} = [2, 0]$ and the UAV was placed at the origin. The test procedure consisted in a test subject passing as an "intruder" to follow a path that intersected the detecting zones of both motion sensors. This procedure is described through a diagram in Figure 3.10 (footage from the trials can be found at [56]).

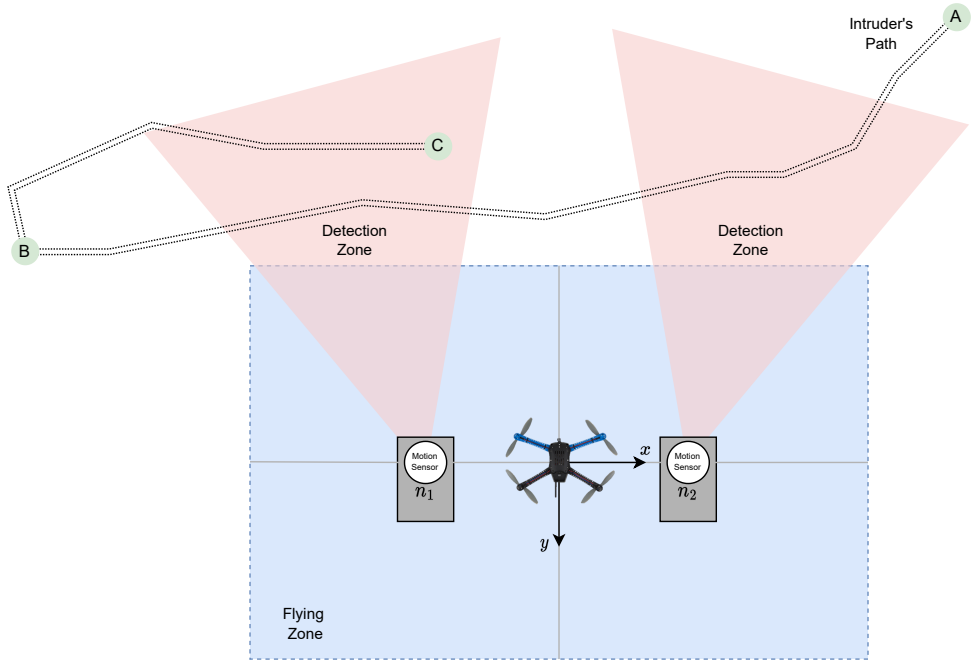
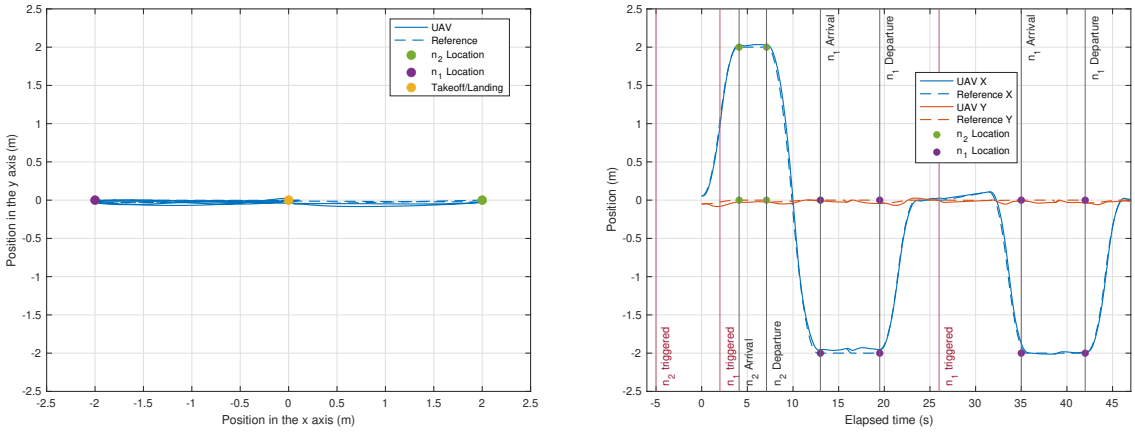


Figure 3.10: Diagram of the trial executed in the flight arena.

With Figure 3.10 in mind, the intruder walks from point **A** to point **B**, waiting for the UAV to land and then proceeds to point **C**. The goal of this path is to test the system in multiple scenarios: firstly when the system is idling, secondly when the UAV is currently on a mission and thirdly after a mission has finished. In this way, a mission is executed where the node n_2 is triggered and shortly after the node n_1 is triggered. After the UAV returns to the takeoff position, the node n_1 is triggered again, demonstrating how the UAV is still ready after executing a mission.



(a) Plot of the UAV position, reference and nodes location.

(b) Plot of the UAV position and reference over time.

Figure 3.11: Results obtained from the real trial with the motion sensors.

Figure 3.11 shows the trajectory of the UAV during the experiment in Figure 3.11(a) and the evolution over time in Figure 3.11(b) to better convey the position error involved. As expected from the simulations, the UAV was able to fly to the locations of the nodes n_1 and n_2 when these were activated. Furthermore,

Figure 3.11(b) shows timestamps for all the major events, including arriving and leaving the location of a node and when a motion sensor trigger message is received. According to the test procedure, the two sensors are triggered in quick succession in the first mission and later with node n_1 being triggered again. Moreover, in the first mission, the UAV flies to the location of both nodes sequentially as designed. The mission planner was configured to command the UAV to inspect node n_1 for 3 s and node n_2 for 5 s. When it comes to the delay between a sensor being triggered and the UAV arriving at its location the main factors at play are the communication delay associated with the WSN and the time it takes the UAV to arrive at its destination, where the former should be negligible for most missions where the vehicle travel time is much bigger than any communication delay. With this in mind, the values for this delay are approximately 9 s, 12 s and 10 s for the three sensor trigger events respectively. In the first and last case, the values are very similar since the UAV was disarmed in both cases and at the same distance from its destination. In the second case, the delay was bigger since that when the sensor was triggered, the UAV was still completing the previous mission and it was further away from its destination.

3.5 Discussion

In this chapter, the WSN developed in Chapter 2 was complemented with autonomous aerial vehicles to further enhance the capabilities of the proposed surveillance system. In order to validate the characteristics of the autonomous inspection, we first prepared a simulation environment with a reproduction of the steps in Appendix B along with those detailed in this chapter. Its main contribution is to allow others to use this simulation environment in preparation for trials in the real ISR flight arena. Additionally, a network was assembled using motion sensors and the WSN developed in Chapter 2, and ROS packages were developed to allow the UAV to carry out inspection missions.

The results obtained in the real experiments showed great reliability as the system flew the UAV to the sensor location every time it was triggered. Moreover, by commanding the UAV to stop at the inspection point, the surveillance task can be carried out (i.e.: by using a camera pointed at the inspection zone). Lastly, the UAV was able to follow its reference trajectory closely, with an overall average position Root-Mean-Square-Error (RMSE) of 0.0381 m. Moreover, the proposed modular architecture comprised of sub-systems working over a communication network can be extended to use other ground vehicles, contact authorities, add additional sources of data, etc.

Chapter 4

Conclusion

In summary, this thesis addresses the concept of a surveillance system using a wireless network of motion sensors and UAVs. An overview on the current state of art similar surveillance systems was presented and analysed in terms of structure. The current dissertation main contribution is the full integration between a WSN with a communication protocol capable of mitigating the loss of messages and aerial vehicles that can respond to requests of surveillance.

In Chapter 2, the main technologies used for wireless communications are gathered and analysed so as to provide comparison data and understand which technology is the most appropriate for the envisioned surveillance scenario. By comparing the requirements of the proposed project against the abilities of each technology, the systems is designed on top of LoRa technology with the design and implementation of a communication protocol to have the nodes work as a WSN. As to offer a full characterization of the capabilities of the developed network, the nodes were tested in both a rural and challenging urban setting with very promising results of achieving communication over $1\mathrm{Km}$ in the rural setup and communications within and between buildings in the urban case.

In Chapter 3, the surveillance system itself was developed by integrating a UAV to further inspect problematic zones. As to validate the solution, we first simulated the UAV resorting to ROS and *Gazebo* with the package written to allow visiting waypoints as a mission. The complete WSN was integrated with a network of wireless nodes with motion sensors that connect to the ROS environment running the UAV package. The overall surveillance system was validated in experiments at ISR flying arena with good reference error and visit of all locations generating events in the WSN.

Overall, the proposed surveillance system fulfills the requirements for this project described in Section 1.2 with the following experimental measures of performance: i) network nodes were able to communicate at more than $1\ \mathrm{km}$ with a delay value of around $133\ \mathrm{ms}$. This strengthens the idea that it is possible to cover large areas without reliability issues; ii) the inspecting UAV followed the reference trajectory with an average RMSE of $0.0381\ \mathrm{m}$ and during testing in the ISR flight arena, the system worked correctly in all trials (correct identification of a motion event and inspection of its location).

4.1 Future Work

The research developed in this document points towards interesting points of future work in order to improve the efficiency and reliability even further:

- Implementation of DL time slots to minimize the node up-time and decrease power consumption, allowing for nodes to be deployed in areas of difficult access where it is costly to change batteries;
- Integrating the developed network with other types of vehicles such as UGVs, giving the system additional means of dynamic sensing. Cooperation between the UAV and UGV could also be studied;
- Adding more UAVs to the mission poses interesting research problems in terms of overall battery management in order to have continuous operation in case automatic charging stations are available and optimization of routes to inspect all event;
- Testing the surveillance system in a deployment scenario with a malicious human with knowledge of the architecture as to assess its weaknesses.

Bibliography

[1] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52 (12):2292–2330, 2008.

[2] M. Dong, K. Ota, M. Lin, Z. Tang, S. Du, and H. Zhu. Uav-assisted data gathering in wireless sensor networks. *The Journal of Supercomputing*, 70(3):1142–1155, 2014.

[3] Wildfire statistics. (accessed: 29.12.2021). URL `https://crsreports.congress.gov/product/pdf/IF/IF10244`.

[4] Firepuma project. (accessed: 28.12.2021). URL `https://welcome.isr.tecnico.ulisboa.pt/projects/forest-fire-prevention-through-uncertainty-minimization-in-surveillance/`.

[5] V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *IEEE Pervasive computing*, 3(4):24–33, 2004.

[6] V. A. Petrushin, G. Wei, O. Shakil, D. Roqueiro, and V. Gershman. Multiple-sensor indoor surveillance system. In *The 3rd Canadian conference on computer and robot vision (CRV'06)*, pages 40–40. IEEE, 2006.

[7] Y.-C. Tseng, Y.-C. Wang, and K.-Y. Cheng. An integrated mobile surveillance and wireless sensor (imouse) system and its detection delay analysis. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 178–181, 2005.

[8] Y. Kim, D. Kim, P. Chong, J. Kang, E. Kim, and S. Seo. Design of a fence surveillance system based on wireless sensor networks. In *2nd International Conference on Autonomic Computing and Communication Systems*. 2nd International Conference on Autonomic Computing and Communication Systems, 2008.

[9] W.-T. Chen, P.-Y. Chen, W.-S. Lee, and C.-F. Huang. Design and implementation of a real time video surveillance system with wireless sensor networks. In *VTC Spring 2008-IEEE Vehicular Technology Conference*, pages 218–222. IEEE, 2008.

[10] S. Kim, H. Lee, and S. Jeon. An adaptive spreading factor selection scheme for a single channel lora modem. *Sensors*, 20(4):1008, 2020.

[11] U. Raza, P. Kulkarni, and M. Sooriyabandara. Low power wide area networks: An overview. *ieee communications surveys & tutorials*, 19(2):855–873, 2017.

[12] K.-H. Chang. Bluetooth: a viable solution for iot?[industry perspectives]. *IEEE Wireless Communications*, 21(6):6–7, 2014.

[13] J. Yin, Z. Yang, H. Cao, T. Liu, Z. Zhou, and C. Wu. A survey on bluetooth 5.0 and mesh: New milestones of iot. *ACM Transactions on Sensor Networks (TOSN)*, 15(3):1–29, 2019.

[14] D. López-Pérez, A. Garcia-Rodriguez, L. Galati-Giordano, M. Kasslin, and K. Doppler. Ieee 802.11 be extremely high throughput: The next generation of wi-fi technology beyond 802.11 ax. *IEEE Communications Magazine*, 57(9):113–119, 2019.

[15] A. Zreikat. Performance evaluation of 5g/wifi-6 coexistence. *International Journal of Circuits, Systems, and Signal Processing, NAUN*, pages 904–913, 2020.

[16] M. Hidayab, A. H. Ali, and K. B. A. Azmi. Wifi signal propagation at 2.4 ghz. In *2009 Asia Pacific Microwave Conference*, pages 528–531. IEEE, 2009.

[17] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi. A tutorial on ieee 802.11 ax high efficiency wlans. *IEEE Communications Surveys & Tutorials*, 21(1):197–216, 2018.

[18] K. A. Ogudo, D. Muwawa Jean Nestor, O. Ibrahim Khalaf, and H. Daei Kasmaei. A device performance and data analytics concept for smartphones' iot services and machine-type communication in cellular networks. *Symmetry*, 11(4):593, 2019.

[19] M. Rahnema. Overview of the gsm system and protocol architecture. *IEEE Communications magazine*, 31(4):92–100, 1993.

[20] B. A. Yasir, G. Su, and N. Bachache. Range expansion for pico cell in heterogeneous lte—a cellular networks. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, pages 1235–1240. IEEE, 2012.

[21] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas. Lte-advanced: next-generation wireless broadband technology. *IEEE wireless communications*, 17(3):10–22, 2010.

[22] Z. Pi, J. Choi, and R. Heath. Millimeter-wave gigabit broadband evolution toward 5g: Fixed access and backhaul. *IEEE Communications Magazine*, 54(4):138–144, 2016.

[23] T. Kuwabara, N. Tawa, Y. Tone, and T. Kaneko. A 28 ghz 480 elements digital aas using gan hemt amplifiers with 68 dbm eirp for 5g long-range base station applications. In *2017 IEEE Compound Semiconductor Integrated Circuit Symposium (CSICS)*, pages 1–4. IEEE, 2017.

[24] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer. Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot. In *2018 ieee international conference on pervasive computing and communications workshops (percom workshops)*, pages 197–202. IEEE, 2018.

[25] Sigfox device etsi mode white paper. (accessed: 28.12.2021). URL https://support.sigfox.com/docs/sigfox-device-etsi-mode-white-paper.

[26] R. S. Sinha, Y. Wei, and S.-H. Hwang. A survey on lpwa technology: Lora and nb-iot. *Ict Express*, 3(1):14–21, 2017.

[27] P. San Cheong, J. Bergs, C. Hawinkel, and J. Famaey. Comparison of lorawan classes and their power consumption. In *2017 IEEE symposium on communications and vehicular technology (SCVT)*, pages 1–6. IEEE, 2017.

[28] R. Ratasuk, N. Mangalvedhe, and A. Ghosh. Overview of lte enhancements for cellular iot. In *2015 IEEE 26th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*, pages 2293–2297. IEEE, 2015.

[29] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne. Reducing mac layer handoff latency in ieee 802.11 wireless lans. In *Proceedings of the second international workshop on Mobility management & wireless access protocols*, pages 19–26, 2004.

[30] (accessed: 09.11.2021). URL https://www.silabs.com/wireless/multiprotocol/mesh-performance.

[31] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *2012 IEEE wireless communications and networking conference workshops (WCNCW)*, pages 232–237. IEEE, 2012.

[32] H. V. Sampaio, A. L. C. de Jesus, R. do Nascimento Boing, and C. B. Westphall. Autonomic iot battery management with fog computing. In *International Conference on Green, Pervasive, and Cloud Computing*, pages 89–103. Springer, 2019.

[33] A. S. Kurji, A. H. Al-Nakkash, and O. A. Hussein. Lora in a campus: Reliability and stability testing. In *IOP Conference Series: Materials Science and Engineering*, volume 1105, page 012034. IOP Publishing, 2021.

[34] Semtech sx1276 datasheet. (accessed: 30.12.2021). URL https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE.

[35] D. Selent. Advanced encryption standard. *Rivier Academic Journal*, 6(2):1–14, 2010.

[36] What is network topology and types of network topology? (accessed: 14.07.2022), . URL https://afteracademy.com/blog/what-is-network-topology-and-types-of-network-topology.

[37] Wireless topologies. (accessed: 14.07.2022), . URL https://www.emerson.com/documents/automation/training-wireless-topologies-en-41144.pdf.

[38] Esp32. (accessed: 15.07.2022). URL https://www.espressif.com/en/products/socs/esp32.

[39] Arduino uno. (accessed: 30.12.2021). URL https://store.arduino.cc/products/arduino-uno-rev3/.

[40] Arduino spi. (accessed: 30.12.2021). URL https://www.arduino.cc/en/reference/SPI.

[41] Isr flying arena. (accessed: 02.01.2022). URL https://github.com/dsor-isr/isr-flying-arena.

[42] T. Oliveira, P. Trindade, D. Cabecinhas, P. Batista, and R. Cunha. Rapid development and prototyping environment for testing of unmanned aerial vehicles. In *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 191–196. IEEE, 2021.

[43] Ros melodic. (accessed: 02.01.2022). URL http://wiki.ros.org/melodic.

[44] Gazebo simulator. (accessed: 02.01.2022). URL http://gazebosim.org/.

[45] Px4 autopilot. (accessed: 02.01.2022). URL https://px4.io/.

[46] Mavlink - micro air vehicle communication protocol. (accessed: 02.01.2022), . URL https://mavlink.io/en/.

[47] Iris quadcopter uav. (accessed: 02.01.2022), . URL http://www.arducopter.co.uk/iris-quadcopter-uav.html.

[48] Mavros package. (accessed: 02.01.2022), . URL http://wiki.ros.org/mavros.

[49] M. Jacinto. Cooperative motion control of aerial and marine vehicles for environmental applications. Master's thesis, Instituto Superior Técnico, December 2021.

[50] Qgroundcontrol. (accessed: 02.01.2022). URL https://docs.qgroundcontrol.com/master/en/index.html.

[51] Hc-sr501 pir motion detector. (accessed: 25.07.2022). URL https://www.mpja.com/download/31227sc.pdf.

[52] Biss0001 pir controller. (accessed: 25.07.2022). URL https://datasheetspdf.com/pdf/990807/SilvanChip/BISS0001/1.

[53] Lhi778 dual element detector. (accessed: 25.07.2022). URL https://datasheetspdf.com/pdf/63027/PerkinElmerOptoelectronics/LHI778/1.

[54] Jetson agx xavier developer kit. (accessed: 23.11.2022). URL https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit.

[55] Intel aero ready to fly drone. (accessed: 23.11.2022). URL https://docs.px4.io/v1.9.0/en/complete_vehicles/intel_aero.html#intel-aero-ready-to-fly-drone.

[56] Collection of digital files concerning this thesis. (accessed: 23.11.2022). URL https://github.com/hardtekpt/Thesis.

[57] Repository for the wsn modules. (accessed: 18.11.2022). URL https://github.com/hardtekpt/sensor_network.

[58] Repository for the simulation ros modules. (accessed: 18.11.2022), . URL https://github.com/hardtekpt/iris_simulator.

Appendix A

WSN Documentation

This appendix acts as the documentation for the installation and usage of the developed WSN. The code referenced here can be found in the *sensor network* repository [57] and a more detailed documentation can also be found therein.

A.1 Install Guide

This section covers the necessary steps to compile the code for the devices in the WSN. In order to set-up such a wireless sensor network, the following software is required:

- Arduino IDE;
- Python3.

Additionally, the following Arduino libraries are required:

- LoRa.h;
- cppQueue.h;
- aes256.h.

After ensuring all the aforementioned requirements are met, this repository can be cloned and the contents opened in an editor like VScode:

```
$ git clone https://github.com/hardtekpt/sensor_network
$ cd sensor_network
$ code .
```

The packages need to be configured according to the hardware available. The network has been tested with both the Arduino Uno platform using the SX1276 Dragino Shield and with the TTGO LoRa ESP32 modules. To this extent, the pin map for the LoRa radio must be configured for all devices:

- On the Gateway edit the file gateway_serial_definitions.h;

- On every Node use the node_definitions folder and create a file for each node. This file is referenced by node_definitions.h.

Additionally, every node needs a unique $32\ \mathrm{byte}$ encryption key. This key must also be added to the gateway_serial_definitions.h file. Finally, every node needs a unique hexadecimal ID. The gateway has the encryption keys of all the nodes in an array indexed by the node ID.

Regarding the Network Manager, the wsn_config.yaml file must be edited to include:

- The serial port where gateway is attached;
- Information regarding each node:
  - ID;
  - Geographic location;
  - Sensors;
  - Actuators;

It should be noted that example configurations for each file are included in the repository. The last thing to do is to attach any sensors and actuators to the nodes and upload the code.

Appendix B

Flying Arena Simulation Environment Documentation

As there were some problems when following the installation instructions provided in the flying arena repository [41], this appendix describes the steps followed in the installation environment used in this thesis. It should be noted that these instructions are based on the work done by Tiago [41] and Marcelo [49]. Firstly, this guide assumes a fresh installation of *Ubuntu 18.04LTS*.

B.1 Packages Installation

Run the following script to install Gazebo, ROS and *MAVROS*:

```
$ wget https://raw.githubusercontent.com/PX4/Devguide/master
/build_scripts/ubuntu_sim_ros_melodic.sh
$ source ubuntu_sim_ros_melodic.sh
$ sudo apt-get install ros-melodic-mavros
```

Install the following Python libraries:

```
$ sudo apt-get install libgstreamer-plugins-base1.0-dev python3-empy
python3-jinja2 python3-toml python3-pip
```

Install the *PX4* Firmware:

```
$ git clone --recursive https://github.com/PX4/PX4-Autopilot.git
$ cd PX4-Autopilot
$ git fetch --all --tags
$ git checkout v1.12.3 -b latest
$ git submodule update --init --recursive
$ make px4_sitl gazebo
```

At this point, the make command will probably fail and missing packages might have to be installed. The file /.ignition/fuel/config.yaml must be changed so that the server url is “https://fuel.ignitionrobotics.org”.

Change your */.bashrc* to include these lines but put in your own username.

```
alias killros="killall -9 rosmaster gzserver gzclient"
alias s="source  /.bashrc"
export CATKIN_WORKSPACE= /catkin_ws
export GAZEBO_PLUGIN_PATH=$GAZEBO_PLUGIN_PATH:/home/username/
PX4-Autopilot/build_gazebo
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/home/username/
PX4-Autopilot/Tools/sitl_gazebo/models
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/username/
PX4-Autopilot/build_gazebo
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/home/username/
PX4-Autopilot:/home/username/PX4-Autopilot/Tools/sitl_gazebo
source /home/username/PX4-Autopilot/Tools/setup_gazebo.bash
/home/username/PX4-Autopilot
/home/username/PX4-Autopilot/build/px4_sitl_default
```

Next, create a python workspace:

```
$ sudo apt-get install python3-catkin-pkg-modules
python3-rospkg-modules
$ mkdir -p  /catkin_ws_python/src && cd  /catkin_ws_python
&& catkin_make && source  /catkin_ws_python/devel/setup.bash
&& wstool init&& wstool set -y src/geometry2 --git
https://github.com/ros/geometry2 -v 0.6.5
&& wstool up && rosdep install --from-paths src --ignore-src -y -r
&& catkin_make --cmake-args -DCMAKE_BUILD_TYPE=Release
-DPYTHON_EXECUTABLE=/usr/bin/python3
-DPYTHON_INCLUDE_DIR=/usr/include/python3.6m
-DPYTHON_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython3.6m.so
```

After that install *MavRouter*:

```
$ sudo apt install python-future python3-future libtool autoconf
$ git clone -b v1 https://github.com/intel/mavlink-router
$ cd  /mavlink-router
$ git submodule update --init --recursive
$ ./autogen.sh && ./configure CFLAGS='-g -O2' --sysconfdir=/etc
--localstatedir=/var --libdir=/usr/lib64 --prefix=/usr
$ make
$ sudo make install
```

If an error occurs during the submodule update due to *pymavlink* not found, the package can be manually downloaded from here “https://github.com/ArduPilot/pymavlink”. Finally, the *make px4_sitl gazebo* command can be ran again from the PX4-Autopilot folder.

B.2 Environment Configuration

Now that all the necessary packages have been installed, the simulation environment itself must be configured. Included in this repository are the cpp ROS modules that make up this environment [58]. As explained in Section 3.1, the system is based on ROS launch files. They are responsible for spawning the world and the vehicle and for running the simulation package. To control the UAV, the UAV class is used from the ISR flying arena. The files from the repository are ROS packages and should be place in the /catkin_ws/src folder. A build command can then be ran.

```
$ catkin build
```

To launch the simulator run:

```
$ roslaunch drone_sim_bringup simulator_bringup.launch
```

Appendix C

Vehicle Description - 3DR Iris Quadrotor

This appendix describes the UAV used in simulations regarding Chapter 3. To this extent, the quadrotor used was the the commercially available *Iris* model from the company 3DR (3D Robotics) [47] depicted in Figure C.1. For this quadrotor, a gazebo model is available from the PX4 Autopilot SITL plugin.

Figure C.1: Iris quadrotor (from [47]).

The vehicle features a motor to motor dimension of $550\ \mathrm{mm}$ with an approximate weight of $1.5\ \mathrm{Kg}$. It contains various sensors as a barometer, a magnetometer, an IMU and a Differential Global Positioning System (DGPS) unit. The vehicle's moment of inertia is given by: $I_x = 0.029\ \mathrm{Kgm}^2$, $I_y = 0.029\ \mathrm{Kgm}^2$, $I_z = 0.055\ \mathrm{Kgm}^2$ and the thrust curve used in simulation follows a quadratic of the form $T(T_{in}) = aT_{in}^2 + bT_{in}$, with $a = 34$, $b = 7.2$ and $T_{in} \in [0, 1]$ (a normalized input used by the vehicle's motor mixer) [49].

Appendix D

Vehicle Description - Intel Aero Quadrotor

This appendix targets the Intel Aero Ready to Fly quadrotor used in the real trials during this thesis. It includes the Intel Aero Compute Board, running Linux on a quad-core CPU. This vehicle has a motor to motor dimension of $360\ \mathrm{mm}$ and weights $865\ \mathrm{g}$ without the battery. An image of this quadrotor is shown in Figure D.1.

Figure D.1: Intel Aero quadrotor (from [55]).

The Aero comes with several sensors included. Global Positioning System (GPS), compass, magnetometer, IMU, altitude sensors and a RealSense camera are amongst them.