

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Multi-technology Flying Access Network for Disaster Management Scenarios

Rúben Miguel Rei Queirós

**INTEGRATED MASTER IN ELECTRICAL AND COMPUTER
ENGINEERING**

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Rui Lopes Campos, PhD

Co-Supervisor: Hélder Martins Fontes, PhD

October 26, 2020

Abstract

First responders usually depend on a working communications infrastructure to exchange information and coordinate their efforts. Whenever a natural disaster occurs, such infrastructure may become unavailable due to power outage, broken links to the core network, physical damage to the infrastructure or overload of the network. Due to this, first response teams may have their missions compromised, possibly endangering their lives and the lives of potential victims. From this gap, raises the motivation to develop a system, consisting of a communications solution providing wireless network coverage in the affected areas.

In recent years we have been assisting to the evolution of Unmanned Aerial Vehicles (UAV), which became more reliable, more technically capable (i.e., better energy autonomy and capable of transporting heavier payloads) and more affordable. Due to this, different research projects have been targeting the deployment of flying base stations based on IEEE 802.11 (Wi-Fi) or Long Term Evolution (LTE) on board of these devices. Several trials have been done to validate the feasibility of developing such systems with objectives that are relatively similar to the scope of this dissertation, but they are based on proprietary and closed source solutions that are not cost-effective and that are also limited to one of the above-mentioned technologies.

In this dissertation we propose a Multi-technology Flying Access Network, based on an innovative payload, focused on overcoming the limitations of current solutions. This payload, to be integrated in a flying node, considers two technologies (LTE and Wi-Fi) for providing communications coverage to first responders and potential victims. The LTE technology implementation is based on Software Defined Radio and the Wi-Fi is based on Off-the-Shelf wireless modules. This solution was designed considering the limitations such as weight, size, and power consumption of a real UAV platform from the ongoing EU ResponDrone research project while maintaining low cost. It was, then, instantiated as a real communications payload prototype integrated into a test UAV, and subjected to experimental evaluation to validate its performance in terms of throughput and expected coverage.

From the developed work we concluded that the considered SDR solution alternatives and software suites for LTE implementation are not mature enough to satisfy key requirements of the project such as the required coverage area and stability, which resulted in not being recommended to integrate the communications payload prototype of the European project. Wi-Fi as the alternative solution, can be implemented with a wider variety of hardware with lower costs when compared to SDR solutions. Analysing the results, we fulfilled the stability requirements and achieved longer distances than those obtained with LTE. Our final proposed solution, despite using only Wi-Fi, still contemplates the addition of LTE technology as soon as these solutions become more mature. This work and respective conclusions had a direct impact in the design and selection of the communication payload used in the ResponDrone project in which INESC TEC is a partner.

Resumo

As equipas de salvamento geralmente dependem de uma infraestrutura de comunicações funcional para troca de informação e coordenar os seus esforços. Quando ocorrem desastres naturais, tais infraestruturas podem ficar indisponíveis devido a uma falha de energia, quebra de ligação à rede principal ou sobrecarga de utilização da infraestrutura. Devido a isto, as equipas de salvamento podem ter a sua missão comprometida, ameaçando as próprias vidas e as das potenciais vítimas. Desta lacuna nasce a motivação para desenhar e desenvolver uma solução que forneça comunicações nas áreas afetadas.

Nos últimos anos temos assistido à evolução dos veículos aéreos não tripulados (UAV) que se provaram mais confiáveis, mais capazes tecnicamente (i.e., melhor autonomia energética, maior capacidade de transporte de carga), e com preços mais acessíveis. Devido a isto, muitos projectos de investigação têm vindo a focar a implementação de estações aéreas baseadas no IEEE 802.11 (Wi-Fi) e Long Term Evolution (LTE) a bordo destes dispositivos. Vários ensaios têm vindo a ser feitos que validam a possibilidade de desenvolver tal sistema com objectivos semelhantes ao do tema desta dissertação, mas foram baseados em soluções proprietárias *closed source* que não são economicamente eficientes e que estão limitados a apenas uma das tecnologias mencionadas.

Nesta dissertação propomos uma Rede de Acesso Voadora Multi-Tecnologia baseada num *payload* inovador focado em ultrapassar as limitações das soluções atuais. Este *payload* será implementado num nó voador, considerando duas tecnologias (LTE e Wi-Fi), para fornecer a cobertura de comunicações desejada às equipas de socorro e potenciais vítimas. A implementação da tecnologia LTE é baseada em Software Defined Radio e o Wi-Fi é baseado em módulos sem fios prontos a utilizar. Esta solução foi desenhada tendo em conta limitações de peso, tamanho e consumo de energia de uma plataforma UAV real com origem no projecto de investigação europeu ResponDrone em curso, mantendo o custo o mais baixo possível. Esta solução é depois transformada para um *payload* de comunicações verdadeiro que será integrado num nó voador e sujeito a testes experimentais para validar o seu desempenho em termos de débito e alcance previsto.

Do trabalho que foi desenvolvido conseguimos concluir que as soluções alternativas SDR consideradas e *software* para implementar LTE, ainda não estão maduras o suficiente para satisfazer requisitos chave do projecto como a área de cobertura pretendida e a estabilidade, pelo que esta solução não foi recomendada para implementar o protótipo do *payload*. O Wi-Fi como solução alternativa, consegue ser implementado com uma maior variedade de *hardware* mantendo os custos abaixo das soluções SDR. Com os resultados observados conseguimos satisfazer os requisitos de estabilidade e alcances significativamente maiores do que os que foram obtidos com LTE. A nossa proposta de solução final, apesar de utilizar apenas Wi-Fi, continua a contemplar a adição da tecnologia LTE assim que as soluções baseadas em SDR se tornem mais maduras. O trabalho e as suas conclusões tiveram impacto direto no desenho e seleção do *payload* de comunicações que foi utilizado no projeto europeu ResponDrone em que o INESC TEC é parceiro.

Acknowledgments

I would like to express my gratitude for being able to contribute with the work that was developed in this dissertation, towards the ongoing ResponDrone project success. This project is funded by the European Union's H2020 Research and Innovation Programme and the Korean Government under Grant Agreement No. 833717.

I would also like to extend my gratitude to have integrated the Wireless Networks group of the Telecommunications and Multimedia Unit at INESC TEC, where I had the opportunity of meeting many helpful people whose contribution was important to the development of this work. To my supervisors Prof. Rui Lopes Campos and Dr. Hélder Martins Fontes I would like to express my gratefulness for all the support and patience throughout this work. Without the constructive discussions and help to clarify my endless questions, I would have been lost and my motivation would have disappeared. To Carlos Leocádio for the knowledge that was transmitted regarding the Ansible Software, André Filipe Coelho for the support with the OpenAirInterfaceSoftware, Filipe Borges Teixeira and Francisco Pimenta for the effort troubleshooting the RF obstacles that I came across during the work, my special thank you.

To all the teachers who fostered me the taste for knowledge and taught me how to deal with under pressure situations and strict deadlines, my sincere thanks.

I feel very fortunate for the opportunity of having met unique friends throughout my life. Especially the ones I met in the recent five years in Faculty of Engineering of the University of Porto. I had the pleasure of living unforgettable moments that an ordinary university student would not have. With them, I grew more as a person, than all the other years of my life together. My heartfelt thanks.

To my caring, supportive and loving girlfriend, my deepest gratitude for the encouragement words in the most frustrating days and the patience during my grumpy moments.

A final word must be dedicated to my family, whose support education and investment made everything I achieved possible. It was a great comfort and relief to know that they provided me the management of household activities and nourished me while I was finishing this work in the last struggling weeks.

Finally I apologize all other unnamed people who helped me in my life. Any omission in this brief acknowledgment does not mean lack of gratitude.

Rúben Miguel Rei Queirós

*“Our greatest weakness lies in giving up.
The most certain way to succeed is always to try just one more time.”*

Thomas Edison

Contents

Abbreviations	xv
1 Introduction	1
1.1 Context	1
1.2 Motivation and Problem Definition	2
1.3 Objectives	3
1.4 Document Structure	4
2 State of the Art	5
2.1 Introduction	5
2.2 IEEE 802.11	5
2.2.1 Architecture	6
2.2.2 Hardware Market Survey	7
2.3 LTE	8
2.3.1 Architecture	9
2.3.2 eNodeB	9
2.3.3 Evolved Packet Core	10
2.3.4 Interfaces	10
2.3.5 Software Defined Radio	11
2.3.6 Commercial Solutions	14
2.4 Path Loss Models	14
2.4.1 Friis Model	15
2.4.2 Two-Ray Model	15
2.4.3 Log-Distance Model	16
2.5 Related Work	16
2.6 Summary	17
3 Multi-technology Flying Access Network	19
3.1 Overview of the Proposed Solution	20
3.2 Hardware Selection	20
3.2.1 LTE Hardware	20
3.2.2 Wi-Fi Hardware	22
3.2.3 Validation of Requirements	23
3.3 LTE Solution Implementation	25
3.3.1 Overview	25
3.3.2 Ansible role Development	26
3.3.3 srsLTE Software	27
3.3.4 Open Air Interface Software	33

3.3.5	RF Front End Software	34
3.4	Wi-Fi Solution Implementation	37
3.4.1	Overview	37
3.4.2	OpenWRT	37
3.5	Preliminary Evaluation	39
3.5.1	LTE Coverage Predictions	40
3.5.2	Wi-Fi Coverage Predictions	41
4	Experimental Evaluation	47
4.1	LTE Solution	47
4.1.1	Proof of Concept	47
4.1.2	Scripts Used	52
4.1.3	Experiments in Non-controlled Environment	55
4.1.4	Laboratory Measurements	58
4.2	Wi-Fi Solution	62
4.2.1	Preparation of the Experiments	63
4.2.2	Data Processing	67
4.2.3	Ground Experiments	71
4.2.4	Discussion	79
5	Conclusion and Future Work	81
A	Images	83
A.1	Example of a Wi-Fi Implementation Setup	83
A.2	Example of a LTE Implementation Setup	84
A.3	Experiments Locations	84
B	Scripts Logs and Codes	89
B.1	OAI Script	89
B.2	Data Processing Script	89
B.3	OAI-RAN Configuration Script	94
B.4	Log of Network Cell Info Lite	100
B.5	srsEPC Report of UE Connection	100
B.6	Pullscript	101
C	CSV Files	103
C.1	TX_Rate Occurrences.	103
C.2	RSSI Statistical Data.	105
C.3	Throughput Statistical Data.	111
C.4	Throughput Packet Count and Mean per Iteration.	111
	References	115

List of Figures

1.1	WISE Concept Emergency Scenario [1].	2
2.1	IEEE 802.11 network in Infrastructure mode.	7
2.2	Evolution of mobile communications.	11
2.3	Simplified version of the LTE Architecture.	12
2.4	Two-ray ground reflection diagram [2].	15
3.1	System Architecture Diagram.	19
3.2	Proposed Solution Concept.	20
3.3	LTE Implementation Diagram.	25
3.4	Snippet of /defaults/main.yml Ansible role file.	27
3.5	Task that installs the common packages.	28
3.6	User Database file from srsEPC.	28
3.7	SIM Card and SIM reader.	29
3.8	Software Options	30
3.9	Snippet of the code result.	30
3.10	MME configuration snippet	31
3.11	HSS and SPGW configuration snippet.	32
3.12	Output of bladerf-cli with probe option.	35
3.13	Wi-Fi Implementation Diagram.	37
3.14	Overview of the OpenWRT homepage.	38
3.15	Overview of the OpenWRT Wireless interfaces.	39
3.16	Interface Configuration.	40
3.17	Device Configuration Interface.	41
3.18	Drivers Required for the Wireless USB Wi-Fi Adapters.	41
3.19	LTE Received Power Friis Prediction.	42
3.20	LTE Distance Friis Prediction.	42
3.21	LTE Received Power Log Distance Prediction.	43
3.22	LTE Distance Log Distance Prediction.	44
3.23	Wi-Fi Received Power Friis Prediction.	45
3.24	Wi-Fi Received Power Log Distance Prediction.	45
3.25	Wi-Fi Distance Friis Prediction.	46
3.26	Wi-Fi Distance Log Distance Prediction.	46
4.1	Starting phase of srsEPC.	48
4.2	Starting phase of srsENB and connection to srsEPC.	48
4.3	srsEPC logs for the srsENB association.	49
4.4	User Equipment connection to the network.	49
4.5	UE connects to srsENB.	50

4.6	Screenshot of PingTools Homepage.	51
4.7	srsENB while <i>iperf3</i> is running.	52
4.8	PingTools Reporting the Data transferred during the <i>iperf3</i>	53
4.9	srsEPC example log provided by <i>vmstat</i> tool.	53
4.10	srsENB example log provided by <i>mpstat</i> tool.	54
4.11	Snippet of the end of an <i>iperf3</i> log captured on srsEPC side.	54
4.12	Return Loss of Triband Antenna [3].	59
4.13	Return Loss of Triband Antenna A.	60
4.14	Return Loss of Triband Antenna B.	61
4.15	Return Loss and gain of BT-100.	62
4.16	Return Loss and gain of BT-200.	63
4.17	BladeRF 2.0 Micro xA9 Transmission Power with Amplifiers.	64
4.18	Friis and Log Distance Prediction considering a TX Power of -20 dBm.	65
4.19	Location of the Ground test Spots.	65
4.20	Snippet of RSSI processing.	69
4.21	Snippet of TX_Rate processing.	70
4.22	RSSI Adapter decision plots.	72
4.23	TX_Rate Adapter decision plots.	73
4.24	Throughput Adapter decision plots.	74
4.25	RSSI Channel decision plots.	76
4.26	TX_Rate Channel decision plots.	77
4.27	Throughput Channel decision plots.	78
4.28	Downlink RSSI Results using Best Solution.	80
4.29	Uplink RSSI Results using Best Solution.	80
A.1	Example of a Wi-Fi implementation setup.	83
A.2	Example of a LTE implementation setup.	84
A.3	Router Node window positioning.	85
A.4	Location A Perspective.	85
A.5	Location B Perspective.	86
A.6	Location C Perspective.	86
A.7	Location D Perspective.	87
B.1	Example Log of Network Cell Info Lite.	100
B.2	srsEPC report when new UE connects.	101
B.3	Script designed to centralize the logs in one computer.	102

List of Tables

2.1	Market Survey on wireless USB adapters.	8
2.2	Wi-Fi module transmission power and sensitivity for different data rates at 2.4 GHz (Per Chain) [4].	8
2.3	Wi-Fi module transmission power and sensitivity for different data rates at 5 GHz (Per Chain) [4].	9
2.4	MCS values for 802.11n considering one spatial stream [5].	10
2.5	RF receivers compatible with srsLTE and OAI.	12
2.6	Market Survey on LTE portable solutions compatible with UAV integration. . . .	14
2.7	Empirical values of λ [6].	16
3.1	RF Front Ends specifications.	21
3.2	PA and LNA specifications.	22
3.3	NUC and Raspberry Pi specifications.	22
3.4	Wireless USB Adapters specifications.	23
4.1	srsLTE results of the preliminary connectivity study.	56
4.2	OAI results of the preliminary connectivity study.	57
4.3	Comparison of Theoretical and Experimental RSSI.	58
C.1	TX_Rate Occurrences.	104
C.2	RSSI Statistical Data.	106
C.2	RSSI Statistical Data.	107
C.2	RSSI Statistical Data.	108
C.2	RSSI Statistical Data.	109
C.2	RSSI Statistical Data.	110
C.3	Throughput Statistical Data.	112
C.4	Throughput Packet Count and Mean per Iteration.	113

Abbreviations

3GPP	3rd Generation Partnership Program
AGC	Auto Gain Control
AP	Access Point
BLER	Block Error Rate
BPSK	Binary Phase Shift Keying
BRATE	Bit Rate
BS	Base Station
BSR	Buffer States Report
BSS	Basic Service Set
CQI	Channel Quality Indicator
CSV	Comma Separated Values
DHCP	Dynamic Host Configuration Protocol
DS	Distribution System
DSSS	Direct-Sequence Spread Spectrum
EARFCN	E-UTRA Absolute Radio Frequency Channel Number
ENB	Evolved-UTRAN Node B
EPC	Evolved Packet Core
EPS	Evolved Packet System
ESS	Extended Service Set
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
FDD	Frequency Division Duplex
FEUP	Faculty of Engineering of the University of Porto
FPGA	Field Programmable Gate Array
GI	Guard Interval
HSS	Home Subscription Server
FCC	Federal Communications Commission
FHSS	Frequency Hopping Spread Spectrum
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GTP	GPRS Tunnel Protocol
IEEE	Institute of Electrical and Electronics Engineering
IMSI	International Mobile Subscriber Identity
INESC TEC	Institute for Systems and Computer Engineering, Technology and Science
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
LNA	Low Noise Amplifier
LOS	Line of Sight

LTE	Long Term Evolution
MAC	Media Access Control
MCS	Modulation Coding Scheme
MIMO	Multiple-Input Multiple-Output
MME	Mobility Management Entity
MMS	Multimedia Messaging Service
MS	Mobile Station
NAT	Network Address Translation
OAI	OpenAirInterface
OAI-CN	OpenAirInterface Core Network
OSA	OpenAirInterface Software Alliance
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OS	Operating System
PA	Power Amplifier
PC	Personal Computer
PHR	Power Headroom Report
PHY	Physical
PGW	Packet Data Network Gateway
PLMN	Public Land Mobile Network
QAM	Quadrature Amplitude Modulation
RAN	Radio Access Network
RF	Radio Frequency
RI	Rank Indicator
RNTI	Radio Network Temporary Identifier
RSRP	Reference Signal Received Power
RSSI	Reference Signal Strength Indicator
RX	Receive
SCP	Secure Copy Protocol
SDR	Software Defined Radio
SGW	Serving Gateway
SIM	Subscriber Identity Module
SMS	Short Message Service
SNR	Signal to Noise Ratio
SQL	Structured Query Language
srsLTE	Software Radio Systems Long Term Evolution
SSH	Secure Shell
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TX	Transmit
UE	User Equipment
UAV	Unmanned Aerial Vehicle
UDP	User Data Protocol
UMTS	Universal Mobile Telecommunication System
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Context

First responders usually depend on a working communications infrastructure to exchange information and coordinate their efforts. Whenever a natural disaster occurs, such infrastructure may become unavailable due to power outage, broken links to the core network, physical damage to the infrastructure or overload of the network. Due to this, first response teams may have their missions compromised, possibly endangering their lives and the lives of potential victims.

By June 2017 in Leiria, Portugal, a wildfire flared which resulted in the most fatal natural disaster that ever occurred in the country and the 11th most fatal worldwide, since the records of year 1900 [7]. 66 people died, 254 were injured, and 53 thousand hectares were burned. Many of the victims died inside their cars or running away from the flames in a horrible way. The private national emergency network was unavailable because the corresponding infrastructures were severely damaged or destroyed during the disaster. It is believed that the delayed first response reaction time, due to the lack of communications, was one of the main causes for so many deaths. Without communications, the coordination of the first response teams was greatly impaired, and the victims were unable to request for help or receive instructions to flee the areas at risk.

The mentioned wildfires are not the only cause for emergency communications networks to fail. There are other possible incidents such as the earthquakes or the floods which can lead to similar emergency scenarios, possibly resulting in damaged or overloaded network infrastructures. Every event that affects those infrastructures can be crucial for human lives, if the communication is affected. A possible solution would be a quickly deployable fleet of Unmanned Aerial Vehicles (UAV), providing an airborne communications infrastructure, to re-establish communications for first responders and potential victims. Two independent networks could be considered: 1) a cellular private network, that would cover an area wide enough to enable first response teams to efficiently assist the people, with a better coordination between them; 2) a public network, that would function as a replacement for the commercial one that fails, now allowing for potential

victims to request for help and receive instructions. The use of such flying network infrastructure solution would improve first response operation efficiency and reduce the risk for human lives.

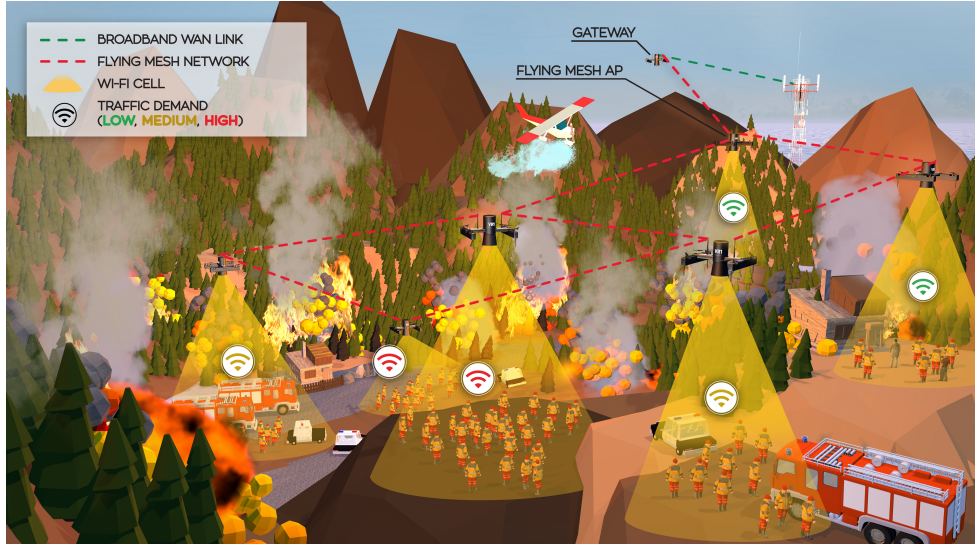


Figure 1.1: WISE Concept Emergency Scenario [1].

Another possible use case for this airborne communications infrastructure is in big events that take place in remote areas such as music festivals or rally races. In these areas, the available communications resources may be insufficient because they were not planned for the temporary high number of users, and in some cases the coverage area is insufficient or nonexistent. These events have a big seasonal factor involved because, during most time of the year, the respective sites do not have users that would justify the construction of a permanent expensive infrastructure. This solution allows the quick and easy deployment of a temporary network that would suit the users' needs during the event.

1.2 Motivation and Problem Definition

This dissertation was developed in the context of the ResponDrone project [1] to help in the design and selection of the communication payload that the Institute for Systems and Computer Engineering, Technology and Science (INESC TEC) is responsible for. INESC TEC has been working on solutions to achieve optimization of traffic routing and smart positioning of flying mesh access points, optimizing network coverage and throughput where it is most needed. Until now, the

research was based only on Wireless Fidelity (Wi-Fi or IEEE 802.11)¹, due to the characteristics of existing Long Term Evolution (LTE) solutions, such as the high cost, large size and weight, and high power consumption, which made their use unfeasible.

With the emergence of Software Defined Radio (SDR) solutions with low cost and compact equipment, it becomes affordable and technically viable to develop a fleet of unmanned aerial vehicles to deploy cellular coverage based on LTE (complementary to Wi-Fi), providing communications to first response teams or victims that are in the affected areas.

The ResponDrone project is currently responsible for the development of a fleet of UAVs that can be operated by a single pilot in emergency situations, in which INESC TEC is involved. The nodes will be providing assistance to the first responders and victims through different types of payload: Video, Communications and other cargo (such as medicines, blood and food). INESC TEC is the responsible partner for the communications payload selection and testing, motivating the work developed in this dissertation.

1.3 Objectives

The goal of this dissertation is to study and evaluate different solutions based on state of the art Wi-Fi and LTE, using commercial wireless modules to support Wi-Fi and SDR for LTE, to deploy communications coverage in emergency scenarios for first responders and victims. There are several objectives worth mentioning that must be accomplished to reach this goal:

- **Solution Design** — Proposal of the solution, which consists of a communications payload for an on-demand flying communications platform capable of providing LTE and Wi-Fi coverage to the mobile terminals used by the first response teams and victims, thus employing state of the art wireless technologies. This communications payload needs some preliminary evaluations regarding feasibility, compatibility and performance that will be relevant for the decision of whether it is viable to go forward with the proposed solution;
- **Development of the prototype** — A system capable of being assembled into an UAV taking into consideration its limitations, such as size, weight and power consumption while maintaining a low cost. This prototype should contemplate several combinations of different hardware and software that must accomplish a fully functional end-to-end connection, using both technologies (LTE and Wi-Fi), between a commercial User Equipment (UE) and itself. These combinations should then be subject to performance evaluations to conclude which is better;
- **Performance Evaluation** — Several tests between different SDR Hardware and open source LTE software suite combinations. The metrics that should be used for such performance

¹ IEEE 802.11 and Wi-Fi is used interchangeably throughout the document.

comparisons include the transmission power, the reference signal received power (RSRP), the Signal-to-Noise Ratio (SNR), throughput and power consumption, in order to come up with the best combination of SDR hardware and software to integrate the final communications prototype. Similar tests shall be carried using Wi-Fi in order to reach conclusions regarding coverage area and throughput. These metrics will be used to come up with the best wireless adapter and best channel combination, to use when implementing the communications payload.

1.4 Document Structure

This document is structured in five chapters and three appendixes. Chapter 2 focuses on the State of the art, background and related work. Chapter 3 presents the proposed solution, the hardware selection, the implementation process of both technologies ending with a preliminary evaluation of the proposed solution. Chapter 4 describes the proof of concept process, the auxiliary scripts and code developed to automatically collect and process the experimental data collected during the experiments, and the results obtained together with some conclusions of the work. Chapter 5 draws the main conclusions of this dissertation and addresses the future work. Appendix A shows images of the setups that were used and other images that were taken during the experiments. Appendix B addresses script code and logs that were used for the development of this work. Appendix C presents the complete Comma Separated Values (CSV) files from where the plots and conclusions were taken.

Chapter 2

State of the Art

2.1 Introduction

This chapter covers wireless technologies related to this dissertation, such as Wi-Fi and LTE, the most commonly used path loss models, and the related work. The research done for LTE and Wi-Fi technologies is unbalanced because the LTE technology is more complex than the Wi-Fi technology, and includes the study of SDR hardware and software suites. The chapter is divided into five main sections:

- [IEEE 802.11](#) — How it all started, the most relevant improvements and revisions to the standard. Hardware market survey of off-the-shelf wireless modules;
- [LTE](#) — Some history for context, the LTE architecture and the most relevant components. It also covers the SDR concept together with a market survey for Radio Frequency (RF) receivers, commercial LTE solutions and open source software suites needed to deploy LTE technology;
- [Path Loss Models](#) — A theoretical study regarding the prediction of the network performance in terms of coverage and throughput based on the 3 most common path loss models: Friis, Two-Ray Ground and Log-Distance Model;
- [Related Work](#) — Similar work that has been published and is relevant to be considered;
- [Summary](#) — A brief section that describes the main ideas that should be taken from the state of the art and their relevance in the scope of this dissertation.

2.2 IEEE 802.11

It all started by 1985 when the United States Federal Communications Commission (FCC) released several portions of wireless spectrum for unlicensed use. Amongst them, the industrial, scientific and medical radio bands also known as Industrial, Scientific and Medical (ISM) band were released for unlicensed use. This IEEE 802.11 working group purpose, is to create an open

standard that defines an over the air interface for how the wireless stations communicate with each other. They establish protocols for Wireless Local Area Networks (WLANs), mainly in the Media Access Control (MAC) and Physical (PHY) Layers that provide interoperability in between wireless systems.

The original version, 802.11 legacy was released in 1997 and there have been several corrections, improvements and new features ever since. The most relevant for the context of this dissertation are the following:

- **802.11** — Provided 1 or 2 Mbit/s of data rate in the 2.4 GHz band using three different physical layer technologies. Diffuse Infrared, Frequency-Hopping Spread Spectrum (FHSS) and Direct-Sequence Spread Spectrum (DSSS);
- **802.11a** — Commonly called "Wi-Fi 2" it uses Orthogonal Frequency Division Multiplexing (OFDM) as the physical layer interface, instead of the ones mentioned previously, which accomplished 54 Mbit/s in the 5 GHz band;
- **802.11b** — This standard managed to raise the maximum data rate up to 11 Mbit/s using only DSSS on the 2.4 GHz band;
- **802.11g** — Known as "Wi-Fi 3", it works in the 2.4 GHz band like the one stated previously, however it uses as physical interface, OFDM (same as 802.11a) which increased the data rate up to 54 Mbit/s in the 2.4 GHz band;
- **802.11n** — The 4th version of Wi-Fi extended the data rate up until 600 Mbit/s on both frequency bands, by adding Multiple-Input Multiple-Output (MIMO) Antennas. Channels of 40MHz, four spatial streams and 64 Quadrature Amplitude Modulation (QAM);
- **802.11ac** — "Wi-Fi 5" builds on 802.11n increasing the width of channels to 80 or 160 MHz, increased number of spatial streams, up to eight and the addition of Multi User MIMO. This results in an up to 1300 Mbit/s total data rate considering all the spatial streams, in the 5 GHz band;
- **802.11ax** — "Wi-Fi 6" the successor of 802.11ac, is currently in development (Draft 6.0 [8]) and will use up to 1024-QAM increasing the data rates by 35%. The physical layer interface will be based on Orthogonal Frequency Division Multiple Access (OFDMA) for overhead reduction and latency. Higher sensibility and robustness for better functioning with the Received Signal Strength Indicator (RSSI).

2.2.1 Architecture

IEEE 802.11 supports two different modes of operation, Infrastructure and Ad-Hoc, however this section will only address the first one mentioned which is relevant for the scope of this dissertation. The diagram presented in Figure 2.1 will be used to explain the architecture.

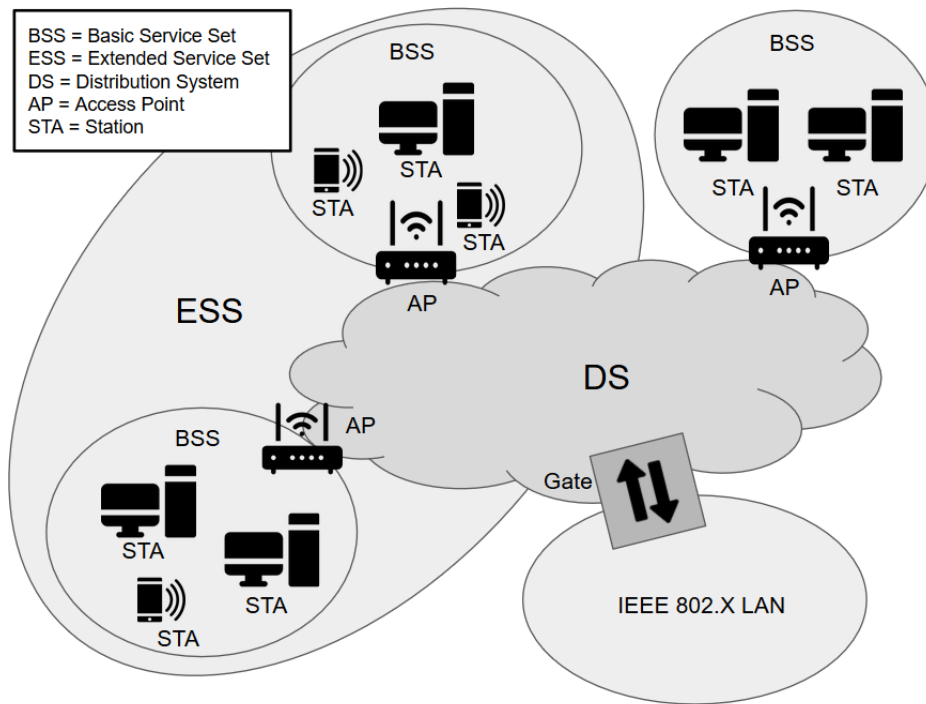


Figure 2.1: IEEE 802.11 network in Infrastructure mode.

When considering this type of architecture, the Basic Service Set (BSS) consists on a set of stations that are capable of communicating directly through the Access Point (AP). This AP provides some network services, such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP) and it may act as a gateway to the Distribution System (DS), if the stations considered in the BSS want to communicate to another BSS. An Extended Service Set (ESS) is the name given to a set of BSS that are connected through the same DS. This technology offers connectivity compatibility with other 802.X networks through a Gate also known as "Portal".

2.2.2 Hardware Market Survey

To understand the limitations we could have with the selection of wireless modules, a market survey was conducted. We reached the conclusion that commercial solutions were not expensive and are very similar within each other. The most relevant wireless modules are presented in Table 2.1.

The TX power was not advertised by the manufacturers of the wireless Universal Serial Bus (USB) adapters with the exception of the model *WLE200NX* whose transmission powers and sensitivity for different data rates are presented in Table 2.2 and Table 2.3.

This module uses different Modulation Coding Schemes (MCS) that range from BPSK up to 64-QAM modulations depending on the receiver sensitivities and other factors (i.e. it differs for different frequencies and chains). This factor is important to take into consideration due to performance comparisons that are meant to be done later in the development phase.

	WLE200NX	PAU09 N600	TL-WN722N
Frequency	2.4 GHz and 5 GHz	2.4 GHz and 5 GHz	2.4 GHz
Wi-Fi Standard	802.11n	802.11ac/n	802.11n
Backward compatibility	802.11b/g	802.11b/g	802.11b/g
Bus Interface	Mini PCIe	USB 2.0	USB 2.0

Table 2.1: Market Survey on wireless USB adapters.

	Data Rate	TX (dBm)	RX (dBm)
802.11b	1 MBps	18	-94
	11 MBps	18	-92
802.11g	6 MBps	17	-94
	54 MBps	15	-79
802.11n 20 MHz	MCS 0	17	-94
	MCS 7	10	-73
802.11n 40 MHz	MCS 0	17	-90
	MCS 7	9	-72

Table 2.2: Wi-Fi module transmission power and sensitivity for different data rates at 2.4 GHz (Per Chain) [4].

The MCS Index mentioned in Tables 2.2 and 2.3 is used to refer to the different modulations used and respective coding rates for both Transmission (TX) and Reception (RX). The data rate at which the communications work also takes into consideration the Guard Interval (GI) between each transmitted symbol to prevent multi path interference. The data rate corresponding to each MCS index and GI, can be found in Table 2.4.

2.3 LTE

It all started in the 1980s with the first generation of mobile communication. Back then, the telecommunications were analog and the transfer speed was extremely low, when compared to nowadays technology. With the introduction of Global System for Mobile Communications (GSM) technology in 2G, by the 1990s, the architecture suffered some improvements and was based on circuit switching. This meant that all services were digital and transported over circuit switches, namely the short message service (SMS) and other data.

The years have passed and the needs for higher transfer speeds have increased with the years. The third generation of mobile communications also known as Universal Mobile Telecommunication System (UMTS) evolved from the General Packet Radio Service (GPRS) developed for the

	Data Rate	TX (dBm)	RX (dBm)
802.11a	6 MBps	17	-94
	54 MBps	12	-77
802.11n 20 MHz	MCS 0	17	-93
	MCS 7	9	-74
802.11n 40 MHz	MCS 0	17	-90
	MCS 7	9	-73

Table 2.3: Wi-Fi module transmission power and sensitivity for different data rates at 5 GHz (Per Chain) [4].

2G. This technology brought significant improvements of up to 2 MBit/s and it marks the time of when packet-switching was added to the circuit switching. This option added more flexibility and efficiency to the network. The circuit domain was still used to transport voice and short message services, however data was now being sent through the packet switching domain.

2.3.1 Architecture

Long Term Evolution is the project name that resulted in the 4th generation of technology (4G). The 3rd Generation Partnership Program (3GPP) community decided to use Internet Protocol (IP) to transport all services, therefore, the circuit switch domain was not needed anymore. This led to some consequences on the architecture itself because the traditional circuits that were used to carry voice or other types of data (e.g. SMS and Multimedia Messaging Service (MMS)) needed to be replaced by IP solutions.

The solution was named as the Evolved Packet System (EPS), that was based on the Evolved UMTS Terrestrial Radio Access Network (E-UTRAN) architecture that was improved from the 3G Radio Access Network (RAN) as a flat type architecture with their two main components. The Evolved Node B (eNB) as the base station, and the Evolved Packet Core (EPC), which has many functionalities, one of them being the gateway to the Internet. Figure 2.3 shows the simplified version of the architecture that will be used to explain briefly the concept behind it.

2.3.2 eNodeB

This is the base station that communicates between the User Equipment (UE) and the EPC of the architecture. Its functions are based on Radio Resource Management, IP header compression and encryption of user data stream. The selection of a Mobility Management Entity (MME) at UE attachment and the Routing of the User Plane data towards the Serving Gateway is also its responsibility.

MCS Index	Type	Coding Rate	Spatial Streams	Data Rate (Mbps) with 20 MHz Channel		Data Rate (Mbps) with 40 MHz Channel	
				800 ns	400 ns	800 ns	400 ns
0	BPSK	1/2	1	6.5	7.2	13.5	15
1	QPSK	1/2	1	13	14.4	27	30
2	QPSK	3/4	1	19.5	21.7	40.5	45
3	16-QAM	1/2	1	26	28.9	54	60
4	16-QAM	3/4	1	39	43.3	81	90
5	64-QAM	2/3	1	52	57.8	108	120
6	64-QAM	3/4	1	58.5	65	121.5	135
7	64-QAM	5/6	1	65	72.2	135	150

Table 2.4: MCS values for 802.11n considering one spatial stream [5].

2.3.3 Evolved Packet Core

The data that comes from eNB base stations is then sent over an interface named S1 to the EPC. It was decided to separate the user data (user plane) from the signaling (control plane) to make the scaling independent and allow operators to dimension and adapt their network easily. Inside EPC, we can decompose it in five network elements.

- **Serving Gateway** — Responsible node for the connection between each UE to the EPC by using a tunneling protocol called GPRS Tunnel Protocol (GTP). It is connected to the eNB base stations and the Packet Data Network Gateway (PGW);
- **Packet Data Network Gateway** — This is the node that connects the EPS Network with the external network (commonly known as the Internet). One of its purposes is to assign an IP address to a remote destination from each UE;
- **Mobility Management Entity** — This node's function is to handle user mobility and hand-over execution;
- **Home Subscription Server** — In this node, there is user profile information that is used to authenticate users in the Evolved Packet System Network;
- **Policy Control and Charging Rules Function** — Control plane node that manages the Quality of Service inside the Evolved Packet System Network and performs pricing actions.

In the context of the dissertation, the User Data Plane will be the main focus. With that in consideration, the Serving Gateway (SGW) and PGW are the important components here.

2.3.4 Interfaces

The communication between each network component inside the architecture is made with interface protocols. The ones that are related to the User Data Plane are the Following:

- **X2-UP** — It's main function is to tunnel end-user packets between LTE eNB stations. This tunneling function supports identification of packets and packet loss management, however

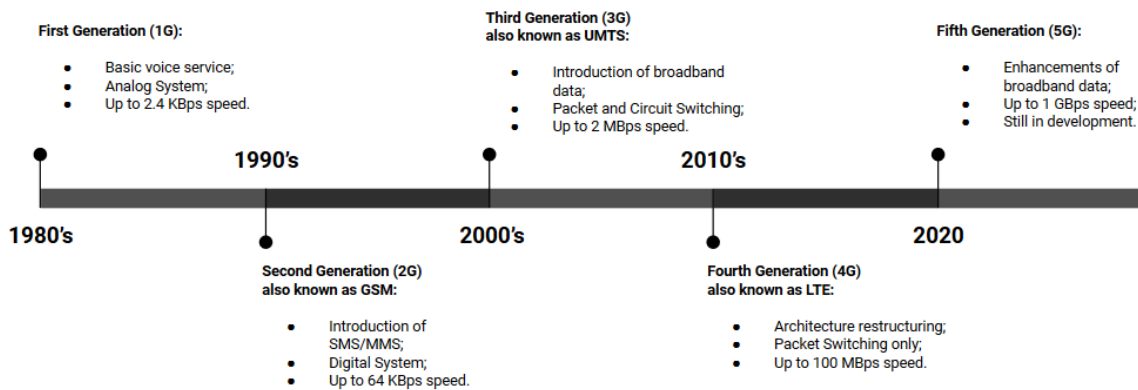


Figure 2.2: Evolution of mobile communications.

there is no guarantee on the delivery of user plane data packet units. It has an identical protocol stack to the S1-UP protocol stack;

- **S1-UP** — This interface's responsibility is to deliver user data between the eNB and the Serving Gateway. It uses GTP over User Datagram Protocol (UDP) or IP (User data encapsulation) and it does not guarantee the delivery of user plane data packet units. As stated before, S1-UP protocol stack is the same as X2-UP.
- **S5/S8** — Provides User plane tunneling and tunnel management between the serving gateway and the packet data network gateway. It is used for SGW relocation due to UE mobility. Used if SGW needs to connect to an non collocated PGW for an established Packet Data Network connectivity. S8 is the roaming version of S5 (inter Public Land Mobile Network (PLMN))

2.3.5 Software Defined Radio

2.3.5.1 SDR Concept

A standard radio system uses dedicated embedded circuits designed to perform signal processing for a specific matter. This hardware, after being designed and produced, has close to none possibility of reconfiguration, making it hard to adapt for any other purpose than what it was designed for.

Within this context, it comes the motivation to develop the SDR. The main concept is to provide a system where the processing that was previously done by the dedicated hardware, was now meant to be done by software.

The main advantage of this system is the possibility of easily changing the running software on which the SDR is using, making it possible to adapt the hardware to anything within the limitations provided by the RF Front End. There are however some disadvantages regarding this concept, such

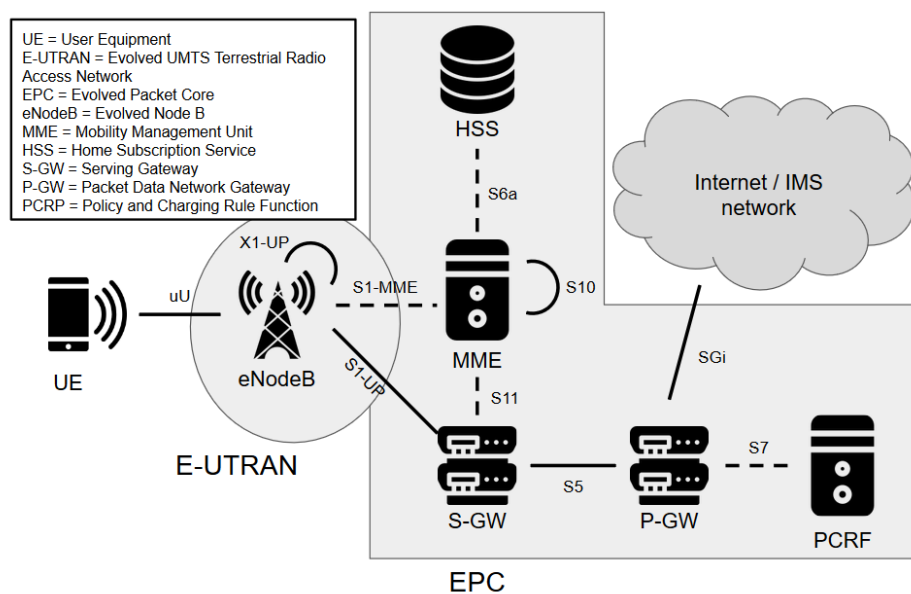


Figure 2.3: Simplified version of the LTE Architecture.

as the bottleneck of the USB bus between the SDR and the PC, which limits the amount of Mega samples that can be processed per second and the transmission power that is significantly lower when compared to dedicated hardware.

2.3.5.2 Hardware Market Survey

A market survey was needed in order to satisfy some hardware requirements and software compatibility. The RF receiver must be able to communicate through a duplex communication system, which means that it must be able to be transmitting and receiving information. It also must be able to tune to the frequency of operation of a LTE system. The higher the transmitting power, better performance the system will have in terms of range and SNR.

	bladeRF x40	bladeRF 2.0 Micro xA9	USRP B210	USRP X300	LimeSDR	XTRX
Tuning Range	300 MHz to 3.8 GHz	47 MHz to 6 GHz	70MHz to 6GHz	DC to 6GHz	30MHz to 3.8 GHz	30MHz to 3.7GHz
Duplex	Full SISO	Full MIMO (2x2)	Full MIMO (2x2)	Full MIMO (2x2)	Full MIMO	Full MIMO
Max Sample Rate	40 MSps	61.44 MSps	61.44 MSps	200 MSps	61.44 MSps	120 MSps or 90 MSps
Max Bandwidth	28MHz	56MHz	56MHz	160MHz	61.44MHz	120MHz
Channels	1	2	2	2	2	2
Transmit Power	+6 dBm	+8 dBm	+10 dBm	+10 dBm	0 to +10 dBm	0 to +10 dBm
Bus Interface	USB 3	USB 3	USB 3	1 or 10 GBit PCIe / USB 3	USB 3	PCIe x2 / USB 3

Table 2.5: RF receivers compatible with srsLTE and OAI.

Only RF receivers that were compatible with the software suite that is going to be used were considered in Table 2.5 in order to, later on, have a controlled environment to establish performance comparisons between hardware and software during the testing phase of the dissertation.

2.3.5.3 SDR LTE software suite

In order to implement the communication system, OpenAirInterface (OAI) and Software Radio Systems LTE (srsLTE) were the software suites that were chosen to the development of the work. This software suites are installed and configured in a Linux Machine, one at a time, and are compatible with the previously mentioned RF receivers.

2.3.5.4 OpenAirInterface

OpenAirInterface Software Alliance (OSA) is a French non-profit organization funded by corporate sponsors. The mission of OSA is to provide software and tools for 5G Wireless Research and Product Development.

It provides a subset release 10 LTE for every relevant architecture element of LTE (UE, eNB, MME, Home Subscription Server (HSS), SGW and PGW). The software is freely distributed by the Alliance under the terms stipulated by the OSA license model and it is implemented on a standard Linux-based machine. It will be used together with the already mentioned RF receivers.

The OAI has many relevant features to the development of the dissertation. Amongst them, the most important ones are the following:

- **Allows different configurations** — The possibility of simulating UE, eNB or EPC as Commercial or OAI based;
- **LTE Release** — LTE release 8.6 compliant, with a subset of Release 10;
- **Frequency Division Duplex (FDD) and Time Division Duplex (TDD) configurations** — Configurable in 5, 10 and 20 MHz bandwidth;
- **Transmission Modes** — Single-Input Single-Output (SISO) and 2x2 MIMO.

2.3.5.5 Software Radio Systems LTE

Software Radio Systems is an Irish company specialized in the development of modular and portable software for wireless systems in the lower layers. It was founded in 2012 by a team from the research centre CONNECT from Trinity College in Dublin.

The product that we will be using throughout the development of the dissertation is srsLTE. It is a high-performance LTE library designed for SDR applications. It is available under both commercial and open-source licenses. The most relevant features are the following:

- **Software** — srsLTE can be complemented with srsENB, srsUE to simulate eNB and UE;
- **LTE Release** — LTE release 15 compliant;
- **FDD configuration** — Supported bandwidths 1.4, 3, 5, 10, and 20 MHz;
- **Transmission Modes** — Up to MIMO (2x2).

2.3.6 Commercial Solutions

Chapter 1 presented the problem that raised the motivation for the development of this dissertation. After a preliminary market survey, some already existing commercial solutions were identified as having the similar objective of solving the problem previously mentioned. However these solutions proved to be expensive which would risk the scale factor of this work.

A market survey was carried out to evaluate which commercial solutions were available on the market capable of deploying LTE technology in a portable way, to use as the payload of an UAV. The results of this survey are presented in Table 2.6

	FeatherLite	OCTBTS8000	OCTBTS8500	Nutaq Pico LTE
Frequency Range	700 MHz to 3.6 GHz	700 MHz to 2.7 GHz	400 MHz to 6 GHz	70 MHz to 6 GHz
Supported Bands	13,14 & Other major frequency bands	1-10, 12, 17, 19	-	-
Bandwidth	5, 10 & 20 MHz FDD/TDD	-	-	1.4, 3, 5, 10, 15 and 20 MHz
TX Power	1 W	2 W (Max)	4 W (Max)	15 dBm
Power Supply	56W Battery & Solar	Battery up to 2 Hr	9V - 30V DC	60-90 W
Weight	1.2 kg	700 g	1.7 kg	2.75 kg
Price	49.5k USD	25k - 40k USD	25k - 40k USD	30k USD

Table 2.6: Market Survey on LTE portable solutions compatible with UAV integration.

These alternatives are relatively heavier but they do not require a computer when compared to the SDR solutions. However, they are significantly more expensive which is a key factor when considering that we are aiming towards a fleet of UAV.

2.4 Path Loss Models

Path Loss is the reduction in power density of an electromagnetic wave as it propagates through space. The sum of factors that are involved in that attenuation of the signal is of major importance in order to design the link budget of our system. An in-depth study of some path loss models will be presented here, to attempt a later on correlation, between the results of the dissertation objectives and a prediction based on theoretical models.

2.4.1 Friis Model

The Friis model is the simplest way of describing the path loss between two antennas without taking into consideration any shadowing or refraction effects that would be caused by the environment. The received power (P_r) is obtained based on the transmitted power (P_t), the Gain of both transmitting and receiving antennas (G_t and G_r respectively) when compared with an isotropic antenna (theoretical model that radiates equally in all directions), the wavelength (λ) of the signal (considering $v = c = 3 \times 10^8 \text{ m/s}$) and the distance between antennas (d) in meters.

$$[ht]P_r = P_t \frac{G_t G_r \lambda^2}{4\pi d^2} \quad (2.1)$$

The Equation 2.1 refers to the Friis model when considering transmitting and receiving power in Watts and the antenna power gain unitless.

$$[ht]P_r = P_t + G_t + G_r + 20 \log_{10} \left(\frac{\lambda}{4\pi d} \right) \quad (2.2)$$

The Equation 2.2 considers dBm as the measuring unit for transmission and reception power and dBi as the measuring unit for the antenna gains.

2.4.2 Two-Ray Model

The two-ray model is used when the effect of reflection is considered in the system. The path loss is divided into two components. The first component is the line of sight (LOS) contribution and the other component is the contribution of the reflection on the ground to the antenna. This effect can be beneficial or harmful to the signal, depending on the phase differences of the LOS and Reflected signals. The original equation is complex and computationally heavy to calculate, so it will be considered the simplified equation stated below.

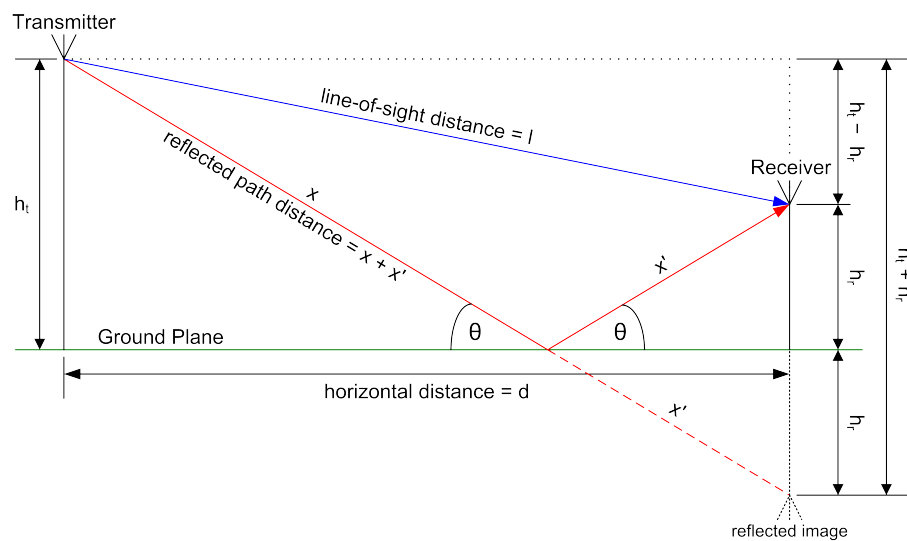


Figure 2.4: Two-ray ground reflection diagram [2].

The Path Loss ($PL(dB)$) is obtained with the distance between antennas (d), the height of the transmitting antenna (h_t) and the height of the receiving antenna (h_r) in meters according to the Diagram 2.4.

$$[ht]PL = 20\log_{10}\left(\frac{d^2}{h_th_r}\right) \quad (2.3)$$

2.4.3 Log-Distance Model

The Log-Distance model is another common model to describe the path loss in electromagnetic waves as they propagate through space. However, this model takes into consideration shadowing effects. The Path Loss ($PL(dB) = P_t(dBm) - P_r(dBm)$) for a given distance (d) is obtained based on a reference Path Loss (PL_0) with a reference distance (d_0) and a path-loss exponent (γ).

$$[ht]PL = PL_0 + 10\lambda\log_{10}\left(\frac{d}{d_0}\right) \quad (2.4)$$

The values given to the path-loss exponent stated in Equation 2.4 are estimated by known empirical values. The values may be later on adjusted to suit each test environment since the values stated in the table below, may not be accurate enough.

Environment and the respective λ	
Free Space	2
Urban area cellular Radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
Inside a Building - LOS	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factory	2 to 3

Table 2.7: Empirical values of λ [6].

2.5 Related Work

Over the last years, there have been several approaches that combine UAVs with SDR for a wide variety of applications.

In [9] SDR was implemented in a UAV to characterize the radio limits of the technology. The objective was to understand if it was possible to keep the required link quality of service, using GNUradio software and USRP SDRs, stressing the channel capacity with the use of multiple UAV, which was accomplished. However, the authors of this work used laptops instead of real UAVs. This is an important factor to consider because when assembling SDR with an UAVs, the Space

and payload weight, processing power, energy consumption and the resulting performance of the communication are relevant factors that are not considered consideration when using a Laptop.

In [10] a combination of SDR and a UAV was also proposed to measure and gather data related to the cellular coverage of a GSM Network. This work considered a remotely controlled UAV, for collecting geo-referenced (GPS coordinates) RSSI measurements from a single GSM Base Station. Even though this solution was proposed, it was never implemented and tested in a real world scenario. Instead, it was simulated with a moving car, not considering the previously stated important limitations of payload integration into a UAV.

In [11] SDR and a UAV were combined to deploy a mobile GSM Network. During these tests, the authors tested the connectivity between the Base Station and Mobile Station by sending an SMS between them. This idea was originally meant to be deployed on a small UAV however it did not happen, since they used an Ettus Research E100 USRP with RFX900 TX/RX board to implement the GSM Base Station only. This solution design required a big sized and heavy SDR, which affected considerably the power consumption of the UAV.

In a more similar approach to the objectives of this dissertation, the project described in [12] has the objective to use SDR to accurately locate victims in disaster management scenarios. This is done by detecting devices that have survived and have access to a GSM network. The initial idea was also to consider state of the art 802.11, however this would require the device to have it activated, which does not need to happen when considering user equipments compatible with the GSM network. The system was used to locate 4 different smartphones spread across an area of 18x18 meters where the UAV would be flying over, while capturing RSSI measurements. Then, with the help of the Mobile anchor (which represents a reference data set collected during the experiments) these RSSI measurements are processed to give a localization also known as Mobile Assisted Localization. The results turned out to be quite good with an average localization error of 3.22 meters which is low considering that the objective is to locate people that may be lost in much larger areas.

2.6 Summary

This chapter was structured with the objective of explaining every technology and concept related to this work, with the due depth, so that the reader is capable of fully understanding the work developed for this dissertation.

The chapter started with the IEEE 802.11 standard, one of the two communications technologies widely available in mobile terminals carried by first responders and victims. We referred to the basics of the Wi-Fi technology and highlighted the most recent updates on 802.11 standards as well as the limitations inherent to the corresponding hardware.

LTE was also approached in this chapter in more depth since it is far more complex to implement and requires knowledge about the architecture of such communication system, the SDR and respective Software suites that can be used to implement it.

The theoretical study of Path Loss Models that will be used to predict the performance of our system was presented next. This is important to give us a perception of what to expect and how to scale our system to suit our requirements. The Friis model estimates the path loss in a simple way, without taking into consideration shadowing effects, while the Log-Distance model is more accurate when using a path loss exponent value suitable for the experimental scenario.

Finally, we presented some examples of related work. From the survey that was performed, we can conclude that there are no solutions that implement both Wi-Fi and LTE technologies, or that takes into account the real limitations of the UAV payload, such as the ones that exist in this work regarding the ResponDrone project.

Chapter 3

Multi-technology Flying Access Network

As stated before, the main objective of this work is to provide network coverage to first response teams and victims of disaster scenarios (e.g., wildfires, floods and earthquakes) or other use cases, such as music festivals and other temporary crowded events that may require network coverage by quickly deploying a temporary fleet of UAVs with communications payload, as presented in Figure 3.1. In this figure we see a ground node that is responsible for the handover and communicates with every UAV stations, and also the UAV stations providing communications coverage to the first response teams. This chapter presents the proposed solution that will be considered during the development of this dissertation. It starts by an overview of the proposed solution, and then it presents a deeper view of the implementation of the two radio technologies that integrate the solution, together with the reasoning behind every decision made throughout the development of this work.

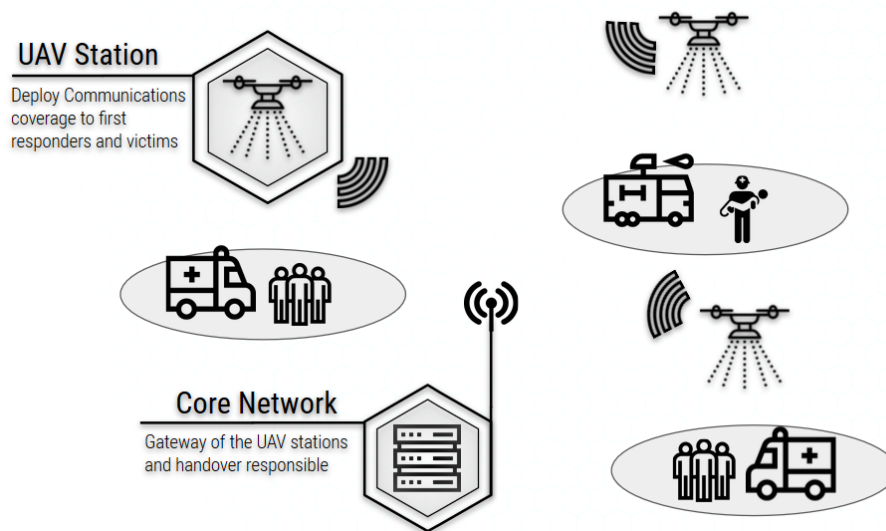


Figure 3.1: System Architecture Diagram.

3.1 Overview of the Proposed Solution

The objective is to design and implement a communications payload with two radio technologies capable of granting LTE and Wi-Fi network coverage. This payload will have restrictions related to its integration with the UAV platform, such as weight, size and power consumption. On top of these, the payload selection will also have complementary restrictions regarding the cost of the hardware.

The concept of the communications payload is presented in the diagram of Figure 3.2. The payload is composed of a Mini Personal Computer (PC) running a Linux based Operative System (OS) that grants compatibility with both: the used LTE Software suite and therefore the chosen SDR Front End software; the Wi-Fi Software suite and the off-the-shelf Wi-Fi cards, accordingly.

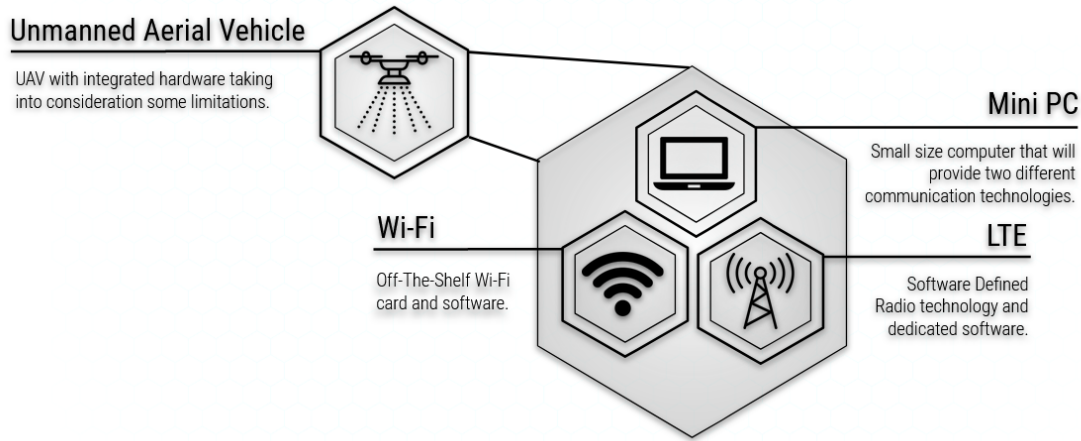


Figure 3.2: Proposed Solution Concept.

3.2 Hardware Selection

The ResponDrone project set some restrictions regarding the specifications of the communications payload we are developing. In terms of maximum weight supported by the UAV, we are limited to 3 kg. The dimensions must not exceed 15x20x7 cm and regarding the power consumption no more than 60-70 Watt depending on the Autopilot system consumption that will be used in the UAV that is not specified yet. Regarding the cost of the payload, 5.000 € was determined as the target price for each payload unit, considering the available budget and the objective of being a low cost and scalable solution.

3.2.1 LTE Hardware

Considering the limitations presented in the previous section, the commercial solutions to deploy LTE technology presented in Table 2.6 were already excluded from being selected since the cost

is close to or above the maximum allowed by the project. We decided to use SDR solutions since they were both lightweight and small sized and also considerably less expensive when compared with the commercial solutions available.

The SDR Front Ends we have chosen consisted of 4 different alternatives from 3 different manufacturers whose specifications, regarding the parameters of the established restrictions, are presented in Table 3.1. The power consumption was not announced by the manufacturers, so we considered the maximum possible available current in USB 3.0 (900 mA) multiplied by the 5 Volts, i.e., the maximum power consumption of an RF Front End is 4.5 W.

- BladeRF x40 by Nuand [13]
- BladeRF 2.0 Micro xA9 by Nuand [14]
- USRP B210 by Ettus Research [15]
- LimeSDR by Lime Microsystems [16]

	BladeRFx40	BladeRF 2.0 Micro xA9	USRP B210
Weight	80 g	90 g	350 g
Dimensions	8.7 x 13.1 x 1.8 cm	6.3 x 10.2 x 1.8 cm	9.7 x 15.5 x 1.5 cm
Power Consumption	(4.5 W)	(4.5 W)	(4.5 W)
Price	352.8 €	604.8 €	1321 €

Table 3.1: RF Front Ends specifications.

Although the 4 alternatives proved to be compatible with the software, the ResponDrone project was leaning towards the BladeRF options. The reason was because Nuand, the manufacturer, also produces Low Noise Amplifiers (LNA) and Power Amplifiers (PA) designed for their RF frontends at a low price when compared with other signal amplifiers. Taking this into consideration, the first two options would give us the best results regarding cost. Those Amplifiers were the BT-100 and BT-200 for the BladeRF 2.0 Micro xA9 and the XB-300 for the BladeRFx40 whose specifications are presented in Table 3.2. The antennas [3] used were also from Nuand since they had a wide frequency range that would fit our purpose.

The Mini PCs used in the implementation are listed in Table 3.3. For the LTE approach we used 2 NUCs. One is hosting the eNB software and is connected to a RF Front End through a USB 3.0 interface and the other hosts the EPC and communicates with the eNB through a wireless interface, however we are only considering the one that will be flying in the UAV since the EPC computer can be communicating from the ground to multiple eNB stations and is not part of the communications payload. The current price of this specific model is unknown because

	BT-100	BT-200	XB-300
Weight	8 g	8 g	80 g
Dimensions	3.8 x 2.54 x 1.27 cm	3.8 x 2.54 x 1.27 cm	8.26 x 12.7 x 3.175 cm
Power Consumption	200 mW	300 mW	-
Price	25.2 €	25.2 €	125.16 €

Table 3.2: PA and LNA specifications.

it was discontinued, but the prices of equivalent newer models (with equivalent size, weight and consumption) range from 200€ to 1000€ depending on the on-board processing performance required for the application scenario. We estimate its price to be approximately 600€ for budget purposes.

3.2.2 Wi-Fi Hardware

For the Wi-Fi technology approach, we used a Raspberry Pi 3 Model B+ (c.f. specifications presented in Table 3.3) to host the OpenWRT OS. To deploy the network we used two different Wireless USB adapters from two distinct manufacturers, TP Link and Panda Wireless (c.f. the specifications presented in Table 3.4). The values of power consumption were not found, so we considered the maximum possible available current in USB 2.0 (500 mA) multiplied by the 5 Volts concluding that the maximum power consumption of a Wireless USB adapter is 2.5 W.

	NUC Intel 5I5MYBE	Raspberry Pi 3B+
Weight	210 g	42 g
Dimensions	10.2 x 10.2 x 2.8 cm	8.5 x 5.6 x 1.7 cm
Power	12-19 V DC	1.7 - 2.0 W
Price	600 € (Discontinued)	35 €

Table 3.3: NUC and Raspberry Pi specifications.

The Panda N600 from Panda Wireless, Inc. is a dual band (2.4 GHz and 5 GHz) wireless USB adapter that can achieve a data rate of up to 300 Mbit/s with support to 802.11n, 802.11g, 802.11b and 802.11a standards. It uses a Ralink RT5572 Chipset and the package came with two 5 dBi detachable omnidirectional antennas that were tested for compatibility with the 2.4 GHz frequency band in INESC TEC lab and proven adapted for the scenario.

The TL-WN722N is a single band wireless adapter (2.4 GHz) that goes up to 150 Mbit/s when using the 802.11n standard. Similar to the Panda N600, this wireless USB adapter also supports 802.11g and 802.11b standards. The omnidirectional antenna provided together with the adapter has an antenna gain of 4dBi and it was also checked in INESC TEC lab for compatibility with the 2.4 GHz frequency band and proven adapted for the scenario.

	TL-WN722N	Panda N600
Weight	36 g	56 g
Dimensions	9.4 x 2.6 x 1.1 cm	6.1 x 2.3 x 0.9 cm
Power	(2.5W)	(2.5W)
Price	12.6 €	21 €

Table 3.4: Wireless USB Adapters specifications.

3.2.3 Validation of Requirements

As stated in the beginning of this section, the limitations set by the ResponDrone project were:

- **Weight** – Limited to 3 kg;
- **Dimensions** – 15 x 20 x 7 cm;
- **Power Consumption** – 60-70 W;
- **Price** – Maximum 5.000 €;

The hardware presented in this section will now be used to validate that the communications payload, as a whole, matches the requirements set by the project. Not every UAV needs to be deploying both technologies simultaneously so we can consider three different types of setup: a LTE solution setup; a Wi-Fi solution setup; and a Multi-Technology solution setup. It is important to mention that all the power supply hardware is already considered in the weight of the UAV and does not count towards our maximum payload weight limit. Also, the antenna and power cable dimensions, weight and price are negligible so they are not considered as well.

To implement the LTE solution we will use a NUC 5I5MYBE mini PC with an USB 3.0 cable to stack on one of the RF Front Ends together with the respective amplifier (if it has one). Considering every possible combination for this setup, including the stacking of the NUC and the SDR Front Ends accounting for the 7 cm payload combined height limit, we have:

- NUC with USRP B210:
 - Weight: $210 + 350 = 560$ g;
 - Dimensions: 10.2 x 15.5 x 4.3 cm;
 - Power Consumption: 65 W (considering NUC is drawing maximum current from power supply);
 - Price: $600 + 1321 = 1921$ €.
- NUC with BladeRF Micro 2.0 xA9 and both BT-100 and BT-200:

- Weight: $210 + 90 + 8 + 8 = 316$ g;
 - Dimensions: $10.2 \times 14 \times 4.6$ cm;
 - Power Consumption: 65 W;
 - Price: $600 + 604.8 + 25.2 + 25.2 = 1255.2$ €.
- NUC with BladeRFx40 and XB-300:
 - Weight: $210 + 90 + 80 = 380$ g;
 - Dimensions: $10.2 \times 13.1 \times 7.78$ cm;
 - Power Consumption: $65 + 4.5 = 69.5$ W (the XB-300 current was drawn from the BladeRFx40 board, therefore it is part of the mentioned 4.5)
 - Price: $600 + 352.8 + 125.16 = 1077.96$ €.

Every configuration regarding the LTE Solution satisfies all the requirements that were established, with the exception of the usage of XB-300 on the BladeRFx40 that is 0.78 cm above the limit. The Weight is definitely not a problem since we are never above 600g in total, and every price is below the 5000€ budget limit established. The power consumption is close to the limit because we are assuming that both the NUC and the RF Front Ends will be drawing max current permanently.

To implement the Wi-Fi solution we used a Raspberry Pi 3B+ with an USB 2.0 cable to stack on one of the presented USB adapters which led us to two possible combinations:

- Raspberry Pi with TL-WN722N:
 - Weight: $42 + 36 = 78$ g;
 - Dimensions: $9.4 \times 5.6 \times 2.8$ cm;
 - Power Consumption: $2 + 2.5 = 4.5$ W;
 - Price: $35 + 12.6 = 47.6$ €.
- Raspberry Pi with Panda N600:
 - Weight: $42 + 56 = 98$ g;
 - Dimensions: $8.5 \times 5.6 \times 2.6$ cm;
 - Power Consumption: $2 + 2.5 = 4.5$ W;
 - Price: $35 + 21 = 56$ €.

In summary, to implement the Wi-Fi technology none of the restrictions are a problem since in every specification the Wi-Fi hardware is within the requirements.

For the Multi-Technology solution, as multiple combinations of hardware components are possible, we considered the worst case scenario in terms of weight, size, and power consumption of each specification requirement:

- NUC with USRP B210 and Panda N600:
 - Weight: $210 + 350 + 56 = 616$ g;
 - Dimensions: $12.5 \times 15.5 \times 4.3$ cm;
 - Power Consumption: 65 W;
 - Price: $600 + 1321 + 21 = 1942$ €.

Considering a NUC with multiple USB ports we could have both technologies running at a single computer since the computational power of one unit is enough to run both LTE and Wi-Fi softwares. All the dimensions previously mentioned assume minimal cabling between the components to comply with the limited volume available.

3.3 LTE Solution Implementation

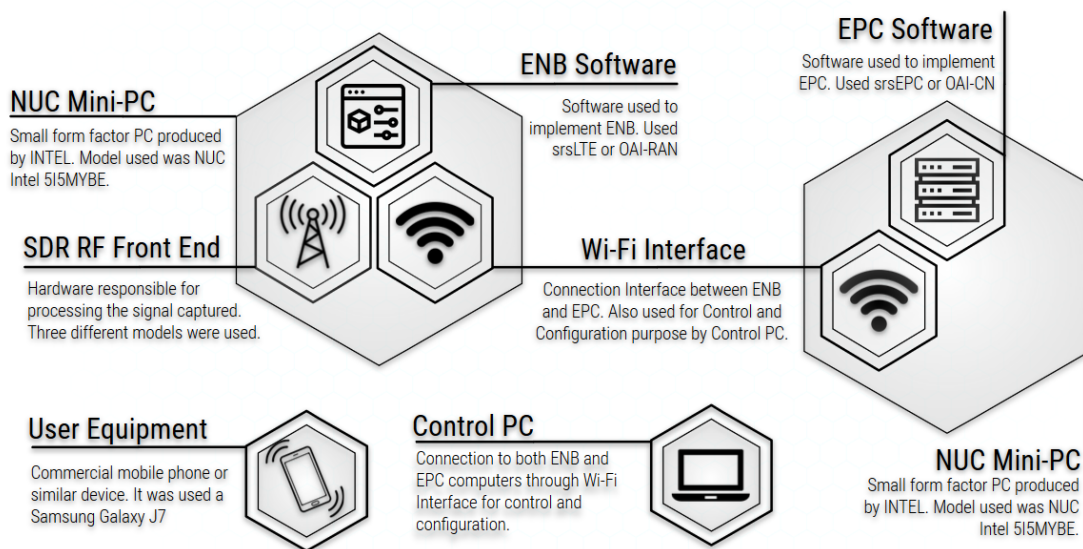


Figure 3.3: LTE Implementation Diagram.

3.3.1 Overview

The diagram presented in Figure 3.3 shows the information regarding the components of hardware needed and the function they had in the implementation of the LTE technology. There were two different NUCs, and each of them played different roles in the implementation. The first one represented the payload to be integrated in the UAV, hosting the eNB software and carrying the RF Front End hardware. The other NUC represented the ground infrastructure running the EPC software and communicating to the eNB station through an Wi-Fi link. In this implementation, we also considered an User Equipment representing the end terminals that would connect to the deployed network, and finally a control PC. This Personal Computer was used to remotely configure

both NUCs through Secure Shell (SSH) connections and to collect the data from tests for further processing.

At first, we installed the recommended OS, Ubuntu 16.04 and we did the installation of dependencies and manually built every software module. We quickly realised it would be hard to make the system scalable if we kept doing this setup manually every time we needed a new node. That reason led us to develop an Ansible role which only requires a computer with the OS installed and an SSH client. After having those 2 requirements, running the Ansible role makes the machine fully operational in 10 minutes on average. The configuration files could also be configured through the Ansible role, however for this work we did it manually since we wanted to keep editing them to explore alternative configurations.

3.3.2 Ansible role Development

Our original objective was to use Ansible to fully automate the process of installing everything required by both software suites from the moment it had the OS installed. This was first developed for srsLTE and later, for OAI, it proved not to be necessary since the installation script was already checking for missing dependencies and installing the respective packages, if needed, in a fully automated way. Therefore, we ended up just developing the role exclusively for the machines we wanted to install srsLTE on.

Figure 3.4 shows a snippet of the file in which every package and variable we used are listed to automate this process. The *packages_common* list contains the main packages used by both softwares and every RF Front End. Below comes the packages exclusively for srsLTE together with the variables required. These variables are basically the links of repositories and important folder locations. Also, in this snippet we can see the packages and variables required to install the bladeRF Front End. Finally, in this file there was information about the SoapySDR which is the software required to use LimeSDR, the UHD for the Ettus research hardware, USRP B210 and also some OAI.

Figure 3.5 shows an example of a simple task that is used to install the common packages previously listed. The *name* field is used so that in case something goes wrong we can debug what happened and where it failed or crashed. The *Become* field is the way Ansible understands that the commands must be used with elevated permissions, which means there is another file in the role which contains the users and their password accordingly. *Apt* is the name of the command that is going to run. Along with the name of the command there is a list of available options to specify what we want. In this case the *name* field point to the list variable which contains every package that we want to install. The *state* field refers to the desired package state. We are using *present* because we just want to check whether it is installed or not, since downloading the latest version every time we had a new node could break the system.

Besides this task whose objective was to install the common packages, there were also task files designed to install packages libraries and software of all the RF Front Ends and also the srsLTE software. These files follow the same logic already presented, however they are bigger

```

srslte-oai_ansible > defaults > ! main.yml
1  ---
2  # defaults file for srslte-oai_ansible
3  #Packets required and main variables used throughout this role
4  #Common Packets
5  packages_common:
6  - git
7  - cmake
8  - build-essential
9  - python-mako # from here and below, required for VOLK
10 - python-six
11 - libboost-all-dev
12
13 #Requisites Packages and Variables for srsLTE install and configuration
14 packages_srs_lte:
15 - libfftw3-dev
16 - libmbedtls-dev
17 - libboost-program-options-dev
18 - libconfig++-dev
19 - libsctp-dev
20 - lksctp-tools
21 - libpcsc-lite-dev
22
23 srs_lte_repo: https://github.com/srsLTE/srsLTE.git
24 srs_lte_source_dir: "/home/{{ ansible_user }}/srsLTE"
25 srs_lte_build_dir: "{{ srs_lte_source_dir }}/build"
26 install_dir: /usr/local/share/srslte #this variable should be retrieved from cmake info
27 conf_dir: /etc/srslte
28
29 #####
30
31
32 #Requisites Packages and Variables for bladeRF install and configuration
33 packages_bladeRF:
34 - libusb-1.0-0-dev
35
36 bladeRF_repo: https://github.com/Nuand/bladeRF.git
37 bladeRF_source_dir: "/home/{{ ansible_user }}/bladeRF"
38 bladeRF_build_dir: "{{ bladeRF_source_dir }}/build"
39

```

Figure 3.4: Snippet of /defaults/main.yml Ansible role file.

and a little more complex, since the *apt* command has multiple parameters that were required to perform the desired task.

In order to run this Ansible role, an *install* *playbook* file was created together with an *uninstall* file, for debug purpose. This *playbook*'s function is to tell Ansible the tasks from which role to run and it is where we can configure the hosts to which the role will be applied. In this scenario, we only stated one IP but it could scale in a simple way.

3.3.3 srsLTE Software

3.3.3.1 srsEPC

After having everything installed it was time to understand how to configure and run the software. srsEPC is a lightweight version of an Evolved Packet Core. The setup of the srsEPC was concen-

```

srslte-oai_ansible > tasks > ! install_common.yml
1  ---
2  # installer for common packages
3
4  - name: Install packages required during commissioning procedure
5    become: yes
6    apt:
7      name: "{{ packages_common }}"
8      state: present

```

Figure 3.5: Task that installs the common packages.

trated in two configuration files. The *user_db.csv* file which is represented in Figure 3.6 contains all the UE's information stored in srsEPC's HSS. In order to make the commercial mobile phone authenticate successfully to the LTE network, we had to flash a Subscriber Identity Module (SIM) card with information that would match this database file from srsEPC. Where it was possible, we kept every parameter with his default value, so we only ended up changing the Authentication algorithm used by the UE, UE's International mobile subscriber identity (IMSI) value and the UE's Sequence number.

```

1 #
2 # .csv to store UE's information in HSS
3 # Kept in the following format: "Name,Auth,IMSI,Key,OP_Type,OP,AMF,SNQ,QCI,IP_alloc"
4 #
5 # Name:      Human readable name to help distinguish UE's. Ignored by the HSS
6 # IMSI:      UE's IMSI value
7 # Auth:      Authentication algorithm used by the UE. Valid algorithms are XOR
8 #            (xor) and MILENAGE (mil)
9 # Key:       UE's key, where other keys are derived from. Stored in hexadecimal
10 # OP_Type:   Operator's code type, either OP or OPc
11 # OP/OPc:    Operator Code/Cyphered Operator Code, stored in hexadecimal
12 # AMF:       Authentication management field, stored in hexadecimal
13 # SNQ:       UE's Sequence number for freshness of the authentication
14 # QCI:       QoS Class Identifier for the UE's default bearer.
15 # IP_alloc:  IP allocation strategy for the SPGW.
16 #           With 'dynamic' the SPGW will automatically allocate IPs
17 #           With a valid IPv4 (e.g. '172.16.0.2') the UE will have a statically assigned IP.
18 #
19 # Note: Lines starting by '#' are ignored and will be overwritten
20 ue1,xor,001010123456789,00112233445566778899aabbccddeeff,opc,63bfa50ee6523365ff14c1f45f88737d,
   9001,000000000474,7,dynamic
21 ue2,mil,208920123456780,00112233445566778899aabbccddeeff,opc,63bfa50ee6523365ff14c1f45f88737d,
   8000,00000000033d3,7,dynamic

```

Figure 3.6: User Database file from srsEPC.

- **Authentication algorithm:** mil - Milenage was the first successfully implemented authentication algorithm in the commercial UE used with the help of the Open Cells software.
- **IMSI:** 208920123456780 - The first 3 numbers are the Mobile Country Code (MCC), the next 2 numbers are the Mobile Network Code (MNC) and the rest is the Mobile Subscription Identification Number (MSIN). For this scenario we used 208 as the MCC and 92 as the MNC that is associated to the Association *Plate-forme Telecom* in France. It was also tested the 001 MCC and 01 MNC that are considered Test Network Codes however the commercial

mobile phone did not recognise them so we decided to use the ones that were set by default which worked just fine in the proof of concept test.

- **SN:** 0000000033d3 - This value is updated everytime there is an authentication of the UE in the HSS.



Figure 3.7: SIM Card and SIM reader.

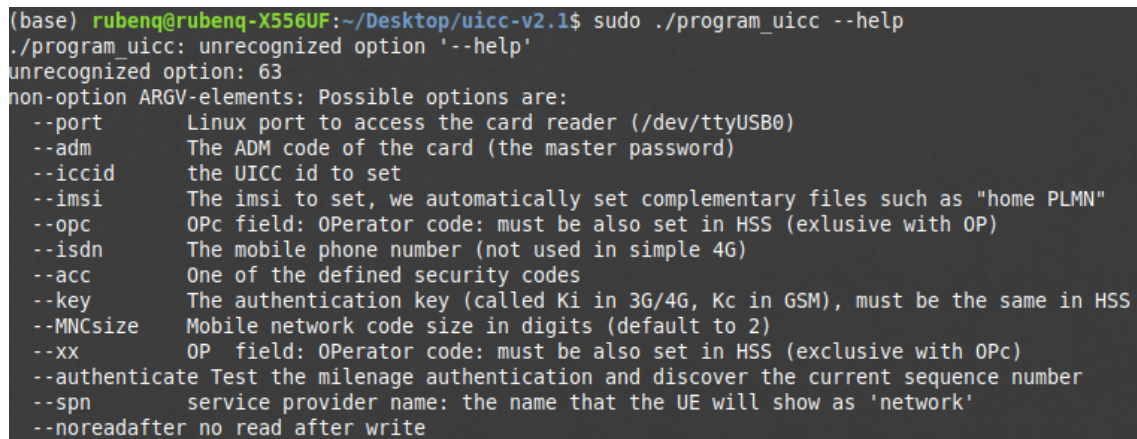
The SIM cards were obtained from the Open Cells program together with a card reader/programmer presented on Figure 3.7. Figure 3.8 shows every possible option to configure the SIM card. In order to successfully overwrite the data, the option *ADM* must be used using any combination of 8 figures. We have also needed set the IMSI to the value previously mentioned, together with the OPC field and authentication Key that would match srsEPC's HSS configuration file, the *user_db.csv*. The authenticate option was also important so we could know what SQN value HSS was going to expect from this specific SIM card, as you can see in Figure 3.9. The SQN had the value of 192 however since we tried the milenage authentication we would need to change the *user_db.csv* file to expect the SQN value of 224 so we could successfully authenticate to the EPC

with our flashed SIM card. This snippet of the terminal however refers to one example SIM card which still had the MCC and MNC set as 001 and 01 accordingly.

The command used initially to set everything accordingly was the one presented in Listing 3.1:

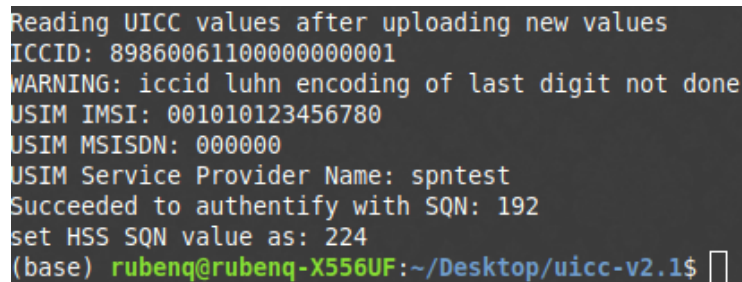
Listing 3.1: Command used to configure the SIM card.

```
sudo ./program_uicc --adm 00000000 --imsi 208920123456789
--opc 63bfa50ee6523365ff14c1f45f88737d
--key 00112233445566778899aabbccddeeff
--spn Test_srsLTE --authenticate
```



```
(base) rubenq@rubenq-X556UF:~/Desktop/uicc-v2.1$ sudo ./program_uicc --help
./program_uicc: unrecognized option '--help'
unrecognized option: 63
non-option ARGV-elements: Possible options are:
--port      Linux port to access the card reader (/dev/ttyUSB0)
--adm       The ADM code of the card (the master password)
--iccid      the UICC id to set
--imsi       The imsi to set, we automatically set complementary files such as "home PLMN"
--opc        OPc field: OPerator code: must be also set in HSS (exclusive with OP)
--isdn       The mobile phone number (not used in simple 4G)
--acc        One of the defined security codes
--key        The authentication key (called Ki in 3G/4G, Kc in GSM), must be the same in HSS
--MNCsize    Mobile network code size in digits (default to 2)
--xx         OP field: OPerator code: must be also set in HSS (exclusive with OPc)
--authenticate Test the milenage authentication and discover the current sequence number
--spn        service provider name: the name that the UE will show as 'network'
--noreadafter no read after write
```

Figure 3.8: Software Options



```
Reading UICC values after uploading new values
ICCID: 89860061100000000001
WARNING: iccid luhn encoding of last digit not done
USIM IMSI: 001010123456780
USIM MSISDN: 000000
USIM Service Provider Name: spntest
Succeeded to authenticify with SQN: 192
set HSS SQN value as: 224
(base) rubenq@rubenq-X556UF:~/Desktop/uicc-v2.1$
```

Figure 3.9: Snippet of the code result.

The other srsEPC file is the *srsEPC.conf* which contains the changeable variables to configure the EPC. In a similar way to how we did for the database, we decided to leave every parameter as *default* except the ones needed for it to work.

Figure 3.10 shows the part of the *epc.conf* file where the MME is configured. In this file we left everything with their default values except for the MCC and MNC where we changed them to 208 and 92 and the mme bind address whose value was the IP of the Ethernet interface of the NUC that was hosting the srsEPC. Figure 3.11 contains the other relevant part of the configuration file

```
#####
#                               srsEPC configuration file
#####

#####
# MME configuration
#
# mme_code:      8-bit MME code identifies the MME within a group.
# mme_group:     16-bit MME group identifier.
# tac:           16-bit Tracking Area Code.
# mcc:           Mobile Country Code
# mnc:           Mobile Network Code
# apn:           Set Access Point Name (APN)
# mme_bind_addr: IP bind addr to listen for eNB S1-MME connections
# dns_addr:      DNS server address for the UEs
# encryption_algo: Preferred encryption algorithm for NAS layer
#                (default: EEA0, support: EEA1, EEA2)
# integrity_algo: Preferred integrity protection algorithm for NAS
#                (default: EIA1, support: EIA1, EIA2 (EIA0 not support))
# paging_timer:  Value of paging timer in seconds (T3413)
#
#####
[mme]
mme_code = 0x1a
mme_group = 0x0001
tac = 0x0007
mcc = 208
mnc = 92

#mme_bind_addr = 192.168.0.100
mme_bind_addr = 192.168.1.85

apn = srsapn
dns_addr = 8.8.8.8
encryption_algo = EEA0
integrity_algo = EIA1
paging_timer = 2
```

Figure 3.10: MME configuration snippet

regarding the HSS and SPGW of the core network. The only variable presented to configure the hss was the file name that srsEPC would access to have information about the User Equipments data that we already went through and SPGW has all the default variables except the *gtpu_bind_addr* that matches the Ethernet interface of the NUC hosting the srsEPC, which is the same of the mme bind address.

This configuration file also contains variables regarding the packet capture files and log files, however we did not want to go deep through the control layer of the srsEPC so we left everything with default values as well.

```
#####
[hss]
db_file = user_db.csv

#####
# SP-GW configuration
#
# gtpu_bind_addr:   GTP-U bind address.
# sgi_if_addr:      SGi TUN interface IP address.
# sgi_if_name:      SGi TUN interface name.
# max_paging_queue: Maximum packets in paging queue (per UE).
#
#####

[spgw]
gtpu_bind_addr   = 192.168.1.85
#gtpu_bind_addr  = 192.168.0.100
sgi_if_addr      = 172.16.0.1
sgi_if_name      = srs_spgw_sgi
max_paging_queue = 100
```

Figure 3.11: HSS and SPGW configuration snippet.

3.3.3.2 srsENB

The srsENB was installed in another NUC machine that was configured in the same Local Area Network (LAN) of the srsEPC. The only configuration file is the *enb.conf* which follows the same format as the configuration files of the srsEPC. In this file we changed the MCC and MNC to match the HSS and the SIM card. We also set the *mme_addr* to the NUC hosting the srsEPC and we set the *gtp_bind_addr* and the *s1c_bind_addr* to the IP of the NUC hosting srsENB.

This file also has configuration variables referring to the PHYS layer of the LTE system.

The *n_prb* (number of resource blocks) can have the following values 6, 15, 25, 50, 75, 100 which refers to occupying a channel bandwidth of 1.25, 2.5, 5, 10, 15, 20 MHz accordingly. There is also a *dl_earfcn* that sets the RX and TX frequency values following a known table of values. *dl_earfcn* is the value that matches the Downlink frequency of the table, but it has an uplink value associated with a given offset.

In this case, we used 1575, 6300 and 3100 for the values of *dl_earfcn*. 1575 was the central channel for band 3, which meant 1842.5 MHz of downlink frequency and 1747.5 MHz of uplink frequency 3100 was the central channel for band 7, which meant 2655 MHz of downlink frequency and 2535 MHz of uplink frequency 1575 was the central channel for band 20, which meant 1842.5 MHz of downlink frequency and 847 MHz of uplink frequency. This formula of converting between the E-UTRA Absolute Radio Frequency Channel Number (EARFCN) and the corresponding frequencies of TX and RX results in a table that was consulted in [17].

Another configuration possible was the RF and TX gain, whose range of values could vary depending on the used hardware. We disable the RX value in order to configure the Auto Gain

Control (AGC) later on through the RF Front End software. Nevertheless, we set the TX gain to the maximum supported by the hardware.

In the RF configuration parameters we still had *device_args* parameter that we left as "auto" in order to detect the first RF Front End connected. Since we only had 1 RF Front End connected at a time, that would not be an issue. The "other" parameters were left with default value or commented.

Finally, in this configuration file there was the parameter to enable a GUI to pop up which was used during the tests to have a better feeling of what was happening and to better interpret the results better. Other than this parameters, everything was left untouched with their default values.

3.3.4 Open Air Interface Software

The Open Air Interface software was initially proposed as the alternative open source software to srsLTE for deploying LTE technology. However, due to its complexity and high level detail, it was instead used to solve some of the problems and obstacles found along the development of this work. With the use of Open Air Interface we could prove that some obstacles had their origin in software bugs, but we also managed to prove that some of the obstacles were hardware-related. Taking this into consideration, and also because both software suites were designed to implement the same network, the level of detail in the explanation of the software components and proof of concept using open air interface software is not as high as the level used to explain srsLTE, otherwise it would consist of duplicate concepts implemented through a different method.

3.3.4.1 OAI-CN

The updated version of the OpenAirInterface Core Network (OAI-CN) was of complex installation and required virtual machines, software to implement a distributed virtual multilayer switch and many dependencies, which raised obstacles that were not possible to overcome due to insufficient documentation. To overcome this situation we used a legacy EPC version [18] of the software. OAI-CN is a complex version of the EPC component, since it had a configuration file for each of the component: MME, HSS, SPGW and his own database in Structured Query Language (SQL) that had to be properly populated with the SIM card info through scripts in order to successfully connect the UE to the LTE network.

3.3.4.2 OAI-RAN

This Software was downloaded from EURECOM gitlab server [19] using version 1.2.1 released in February 2020. After downloading it the building process was not direct and required a command with multiple options as arguments.

Listing 3.2: compileOAIScript.sh

```
#!/bin/bash
```

```
cd ~/openairinterface5g
source oaienv
#valid options USRP and BLADERF
./cmake_targets/build_oai -w USRP -c --eNB
```

The Listing 3.2 shows the script used to build the software, where the `-w` option was to specify the RF Front End hardware that was going to be used with the OAI-RAN software. It would install the dependencies and drivers required to run such hardware without any issue. The `-C` option was to delete all the previously installed software related to OAI-RAN in order to make a clean build from scratch. The `-eNB` option was to make the LTE softmodem, the software that emulates the eNB station.

Similar to srsLTE, the EPC-RAN sums up the configuration to one file presented in Listing B.3 and at a glance it has a big amount of configuration options even though we only used a few of them and left the rest with their default values. We configured the MCC and MNC to match the ones previously configured with the SIM Card and also the IP addresses for multiple components of the network: The MME IP address, that was set with the NUC who hosted OAI-CN, and both S1U and S1_MME network interfaces that were configured with the IP address of the NUC that hosted the OAI-RAN software.

Besides this initial configuration that was required for the communication inbetween the eNB station and the EPC, this file also allowed us to configure the *eutra_band* we wanted to deploy our network with (in our case the bands 3, 7 and 20) however we still needed to manually configure the downlink frequencies of those bands together with the uplink frequency offset that is calculated from the downlink frequency. The number of the physical resource blocks was configured with the variable *N_RB_DL* and the only values that were supported by the software were 25, 50, 75 and 100 meaning 5, 10, 15 and 20 MHz of bandwidth accordingly.

Still in this configuration file, it was possible to set the values of the RX and TX gain, whose values would vary depending on the used hardware. However, in OAI-RAN the RX value could not be disabled in order to allow the usage of AGC. To the best of our knowledge, this feature is not supported by the OAI software in the version we used which was an issue when testing the software because it made the debugging job harder. The TX gain, similar to the logic adopted in the srsENB configuration file, was set to the maximum value supported by the hardware.

3.3.5 RF Front End Software

The bladeRF hardware needed the *bladerf-cli* interface that was used several times through the development of the dissertation. Initially this software was used as the only way we had of finding out if the hardware was successfully installed together with their dependencies and libraries. With the option *probe* we were able to get a report from the device such as the one shown in Figure 3.12.

It was through this interface that we would load the latest version of the Field-programmable gate array (FPGA) image into the FPGA in order to use the SDR. Since this step had to been done

```
bladerf1@bladerf1-desktop:~$ bladeRF-cli -p
Description:  Nuand bladeRF 2.0
Backend:      libusb
Serial:       349c27daeb7d47f8aa73c0549e24fdfb
USB Bus:      3
USB Address:  2
```

Figure 3.12: Output of bladerf-cli with probe option.

every single time we power the SDR we found two ways of skipping this manual step. The first one would be by specifying a path to the location of the image file, that the host device would look for and run after powering the SDR. The other option was not dependent on the host PC since it was written directly into the SDR SPI flash. This last method was slower but it was the chosen one since we could plug the SDR in any computer and having it ready to use in about 10 to 20 seconds. Special precautions were also taken into consideration such as having the antennas always connected to the RF Front End in order to prevent internal damage to the equipment.

Other feature that proved useful early on the development was related to the calibration of the device. This calibration was made to both the RX and TX modules and the way it worked was basically by checking values for given frequencies and gains, and comparing them to table based values. It would then adjust the offsets and retry so that the difference between the real values and the table ones would be near zero.

The print option of the *bladerf-cli* interface was also very important so we could keep track of the most important information of this device. Among all the information reported by the print option, the ones worth pointing since they were used in debugging process of the system issues were:

- **RX and TX bandwidth** - Set by the Resource Blocks in the configuration file of the eNB;
- **RX and TX frequency** - Set by the EARFCN value in the configuration file of the eNB;
- **AGC mode** - We configured it to be enabled by default in case the software would not have enough permission to set it;
- **TX and RX gains** - These values are printed together with their range values and they were useful to find out the max allowed for each different hardware. The TX gain would be set to the max possible value and the RX would change through the Auto Gain Control Mode;
- **Bias Tee RX and TX** - Disabled by default but it can be set to Enable through the interface. If the PA and LNA are plugged but they are disabled through the interface, the signal bypasses them. If they are enabled, a Green/Blue LED depending if we are referring to the LNA or the PA, will turn on;

- **Power Source** - USB Bus would power the SDR if no Amplifiers were being used. However it was advised to use an external Power Unit and that information would tell us if the device has the correct power source.

Unfortunately, this interface that communicates directly with the SDR hardware can only have one instance open at a time. This means that, whenever we are running any software that access the SDR through this interface, we cannot open another interface to check in real time what is happening with the hardware. This would have been useful to assure some of the previously mentioned information about the SDR was correctly configured during the software execution.

Auto Gain Control plays an important role in the RF and RX component on the system. The energy density of electromagnetic radiation decreases with the square of distance which means that the signal power will drastically change when the receiver is close to the transceiver. In order to prevent damage to the hardware, AGC controls the signal power to desired values when both scenarios occur. Either high amplitude scenarios where AGC lowers the gain, or when the signal is too weak and the AGC will increase the gain values. This changes however will not be effective if the variation of signal power is too quick because the AGC was not designed to have a fast response time. Otherwise it would be susceptible to noise or intentional variations of the amplitude such as amplitude modulations, for example.

3.4 Wi-Fi Solution Implementation

3.4.1 Overview

To implement the Wi-Fi technology we used a Raspberry Pi 3 Model B+ running an OpenWRT OS compatible with the chosen Raspberry Pi model. For better understanding, the Figure 3.13 shows all the components used to successfully implement the technology into the Raspberry Pi. With the use of OpenWRT we configured two different Wi-Fi Interfaces. The first one used the on-board network Wi-Fi interface of the Raspberry Pi, whose objective was to access the Raspberry Pi through SSH from a control PC to configure the system and control the scripts that we wanted to use. The other Wi-Fi network, which is the relevant one for the scope of this implementation, is deployed with the usage of a Wireless USB adapter and configured through OpenWRT.

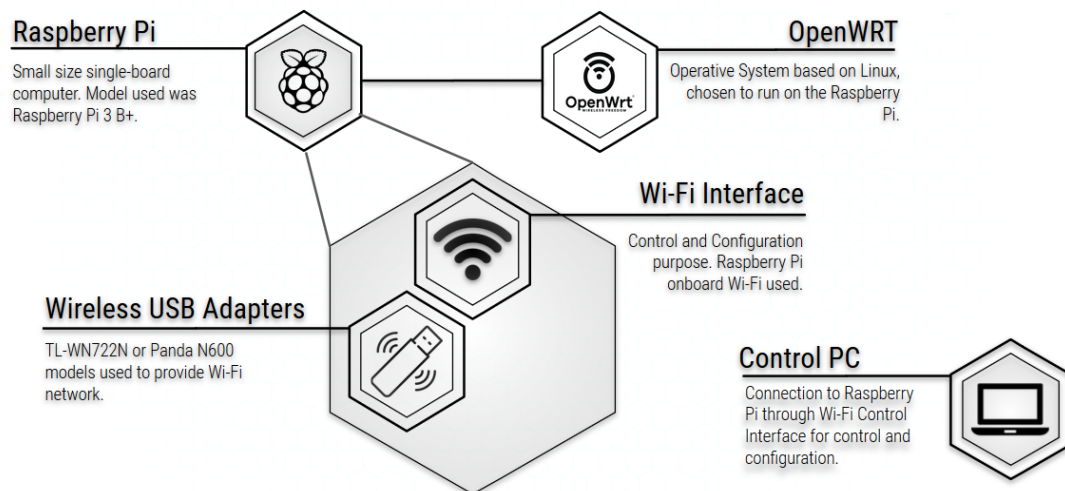


Figure 3.13: Wi-Fi Implementation Diagram.

3.4.2 OpenWRT

The OpenWRT version used was 19.07.3 bcm2710 [20] and it was flashed into the memory card used in the Raspberry Pi with a microSD Card Writer. After plugging the memory card and powering the Raspberry Pi it was ready to run since we could configure everything through an user friendly Web based Interface. In Figure 3.14 we can see that the system does not show any signs of memory starvation and the CPU usage of the processes that are always running in the background is also not a problem which means that the Raspberry Pi has enough computational power to withstand the OS. However, this screenshot was taken while the system was in idle mode, therefore we can not assume yet if it can be used to deploy a Wi-Fi network without any resource problems, before having it under load first.

Status

System

Hostname	OpenWrt
Model	Raspberry Pi 3 Model B Plus Rev 1.3
Architecture	?
Firmware Version	OpenWrt 19.07.3 r11063-85e04e9f46 / LuCI openwrt-19.07 branch git-20.136.49537-fb2f363
Kernel Version	4.14.180
Local Time	2020-08-29 19:46:00
Uptime	8h 39m 13s
Load Average	0.18, 0.05, 0.01

Memory

Total Available	869.91 MB / 923.13 MB (94%)
Used	36.18 MB / 923.13 MB (3%)
Buffered	924.00 KB / 923.13 MB (0%)
Cached	9.50 MB / 923.13 MB (1%)

Figure 3.14: Overview of the OpenWRT homepage.

3.4.2.1 Wireless Configurations

Through the web interface we configured the Wireless Networks that we needed for the tests that we were going to do. Figure 3.15 shows the interfaces we got available by the time the screenshot was taken. The *radio0* refers to the wired physical interface that was being used to connect to a personal LAN used during the development of the dissertation whose Service Set Identifier (SSID) is Vodafone-27E652. The other physical interface which is the relevant one for the work, is *radio1* that has 2 interfaces associated with it.

The first interface, that operates as the Access Point (AP) of a network operating in *infrastructure mode*, configured through the interface presented in Figure 3.16a is the one that the clients will connect to. The *monitor mode* interface configured through the interface presented in Figure 3.16b was needed for packet tracing purposes and to capture the router side radio packets which were relevant metrics to evaluate the performance of the network. This interface would not interfere with the *master mode* interface that would connect to the client. They are both set to the same network interface otherwise the *monitor mode* would not be monitoring the interface we wanted to. The SSID and security are set as seen in the figure to ease the configuration of the network and are easily changeable for a real world scenario where the security is a must in this type of networks.

Figure 3.17 represents the device configuration interface where we can check the status of the device through different variables. In this menu we configured the network to operate exclusively

Wireless Overview		
	Generic MAC80211 802.11bgnac Channel: 1 (2.412 GHz) Bitrate: ? Mbit/s	<button>Restart</button> <button>Scan</button> <button>Add</button>
	SSID: Vodafone-27E652 Mode: Client BSSID: B8:27:EB:6D:B5:BA Encryption: -	<button>Disable</button> <button>Edit</button> <button>Remove</button>
	Generic 802.11abgn Channel: 13 (2.472 GHz) Bitrate: ? Mbit/s	<button>Restart</button> <button>Scan</button> <button>Add</button>
	SSID: OpenWrtBlack Mode: Master BSSID: 9C:EF:D5:FD:D7:3E Encryption: None	<button>Disable</button> <button>Edit</button> <button>Remove</button>
	SSID: mon_mode Mode: Monitor BSSID: 9C:EF:D5:FD:D7:3F Encryption: -	<button>Disable</button> <button>Edit</button> <button>Remove</button>

Figure 3.15: Overview of the OpenWRT Wireless interfaces.

in N mode but allowing legacy 802.11b rates and to have permanently maximum available transmit power which was 20 dBm. Neither one of the USB adapters reported the noise level which would be important for the results discussion. It was also in this menu that we configured the Channel that the Router would use and the bandwidth that was fixed at 20 MHz.

3.4.2.2 Complementary Software

Besides the original software packages that come installed with the OpenWRT version used, there was the need to install the packages containing drivers for the Wireless USB adapters presented previously. The packages installed are presented in Figure 3.18.

Besides the driver packages, it was also required to install *tcpdump* and *iperf3* through the *opkg* (lightweight packet manager used by Openwrt). These two software are used to help evaluate the performance of the network and the hardware. *iperf3* tool will be used to generate the traffic between the OpenWRT (router side) and the client side, and *tcpdump* is the tool supported by the OS to capture and analyze every packet transmitted/received throughout each *iperf3* test.

3.5 Preliminary Evaluation

For a preliminary evaluation of the proposed solution, some simulations have been made considering the path loss models described in Section 2.4. The simulations presented in the following subsection assume no antenna gains and the empirical values of gamma, when predicting log distance based path loss values, were chosen based on the fact that we were going to test this prototype in a Shadowed Urban Cellular environment.

The following plots help understanding what will be the limitations and bottlenecks of our system and help predicting the results that we will obtain in the real world scenarios. There is one different section for each of the technologies presented before since there are different values for each variable.

Interface Configuration

General Setup
Wireless Security
MAC-Filter
Advanced Settings

Mode
Access Point

ESSID
OpenWrtBlack

Network
wifiblack:

Choose the network(s) you want to attach to this wireless interface or fill out the *custom* field to define a new network.

Hide ESSID
☐

WMM Mode
☒

(a) Configuration of the AP.

Interface Configuration

General Setup
Advanced Settings

Mode
Monitor

ESSID
mon_mode

Network
wifiblack:

Choose the network(s) you want to attach to this wireless interface or fill out the *custom* field to define a new network.

(b) Configuration of the Monitor mode interface.

Figure 3.16: Interface Configuration.

3.5.1 LTE Coverage Predictions

The following simulations were done considering we are using the combination of hardware that provides the higher transmission power, therefore the combination of the BladeRF 2.0 Micro xA9 with the use of the BT-100 and BT-200. We assumed that the Amplifiers had a fixed gain of 10 dBm independently of the bands considered and that the RF Front End was transmitting permanently at the maximum TX power setting, granting a total of 18 dBm of transmission power.

The Figure 3.19 is based on the Friis path loss model and it represents the received power at the user terminal considering that the signal is being transmitted at +18 dBm from the eNB station within a range of 100 meters. We did the simulations for the three bands that will be considered later in this dissertation.

The Figure 3.21 has three different plots one for each of the considered bands. The path loss model considered for this predictions was the Log Distance and three different predictions were calculated, considering different empirical values of gamma based in the Table 2.7.

The Figure 3.20 and Figure 3.22, are based on the Friis and Log Distance path loss models respectively, and they were used to draw conclusions regarding the maximum distance between the user terminal and the base station while maintaining connectivity. All the simulations were

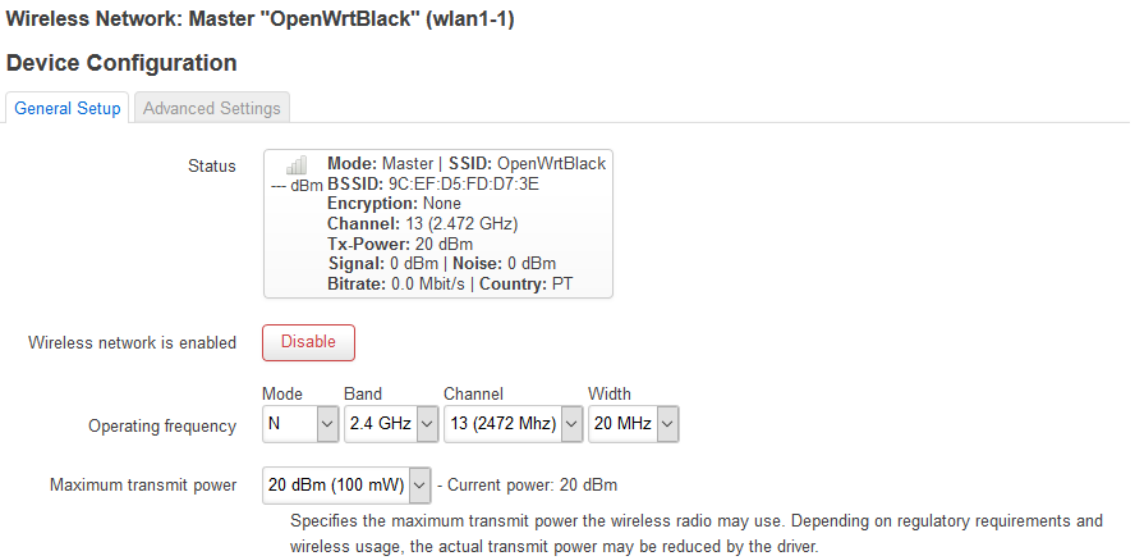


Figure 3.17: Device Configuration Interface.

Package name	Version
ath9k-htc-firmware	20190416-1
kmod-ath	4.14.180+4.19.120-1-1
kmod-ath9k-common	4.14.180+4.19.120-1-1
kmod-ath9k-htc	4.14.180+4.19.120-1-1

(a) TL-WN722N Driver Packages.

Package name	Version
rt2800-usb-firmware	20190416-1
kmod-rt2800-lib	4.14.180+4.19.120-1-1
kmod-rt2800-usb	4.14.180+4.19.120-1-1
kmod-rt2x00-lib	4.14.180+4.19.120-1-1
kmod-rt2x00-usb	4.14.180+4.19.120-1-1

(b) Panda N600 Driver Packages.

Figure 3.18: Drivers Required for the Wireless USB Wi-Fi Adapters.

performed using the same conditions mentioned in the previous study and assuming that the end user terminal has a sensitivity level of -100 dBm. As we can see, in an ideal scenario where we have no shadow and fading effects we can achieve distances from up to 24 km. We also presented the Log Distance predictions with three different gamma values in order to get a realistic prediction of what we can expect in a real world environment.

3.5.2 Wi-Fi Coverage Predictions

The following simulations were done considering we are using a Wi-Fi card that can output a transmission power of +20 dBm.

Figure 3.23 uses the Friis path loss model to predict the received power at the user terminal considering that the signal is being transmitted at +20 dBm from the access point location up to

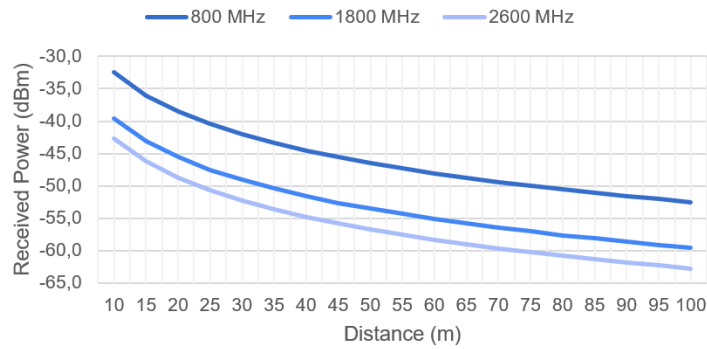


Figure 3.19: LTE Received Power Friis Prediction.

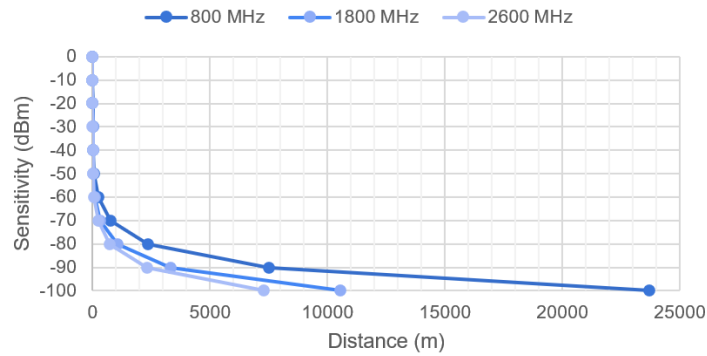
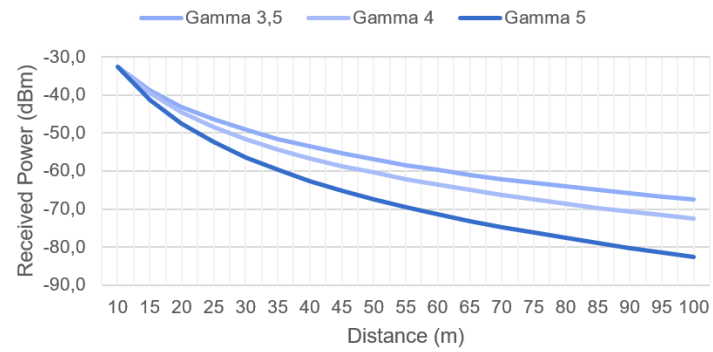


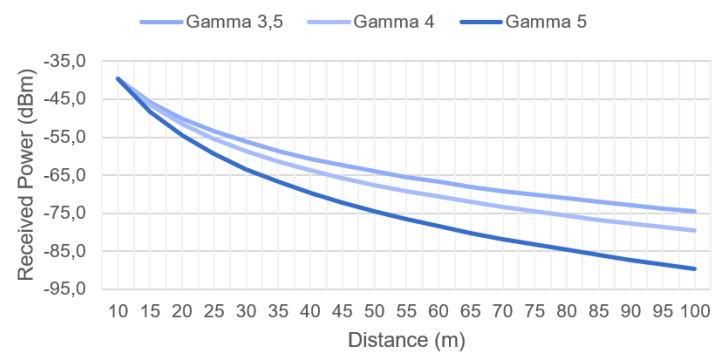
Figure 3.20: LTE Distance Friis Prediction.

a range of 100 meters from it. The simulations were carried out considering both 2.4 GHz and the 5 GHz Wi-Fi frequency bands. The Figure 3.24 also considers both Wi-Fi frequency bands, now using the log distance module exploring different values of γ) to produce more realistic predictions of the received power levels at a distance of up to 100 meters.

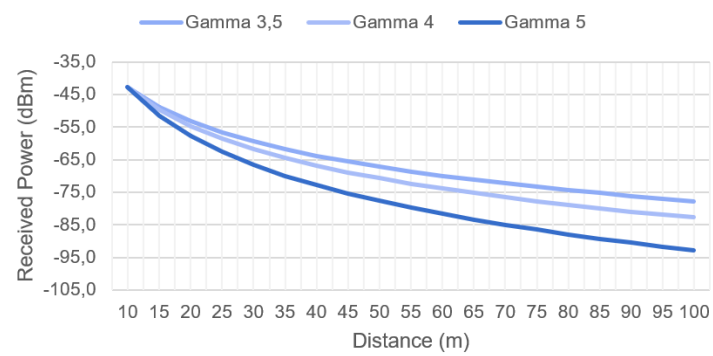
Figure 3.25 and Figure 3.26 follow the same logic that was explained in Section 3.5.1 where we considered a scenario where the terminal user has a sensitivity level of -100 dBm. The plots represent the prediction of the maximum distance at which we can keep a connection between the access point and the client, using Friis path loss model when considering an ideal scenario or Log Distance for a more realistic prediction.



(a) Band 20 (800 MHz).

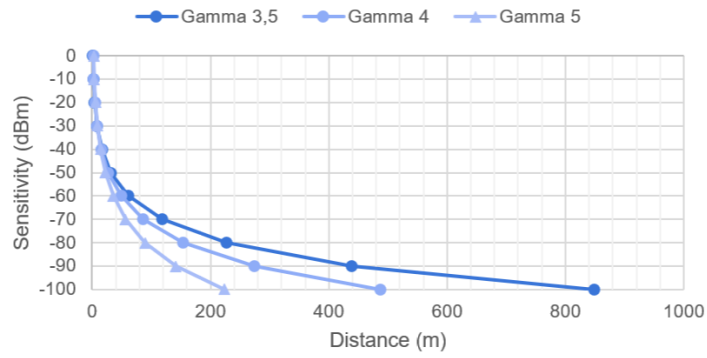


(b) Band 3 (1800 MHz).

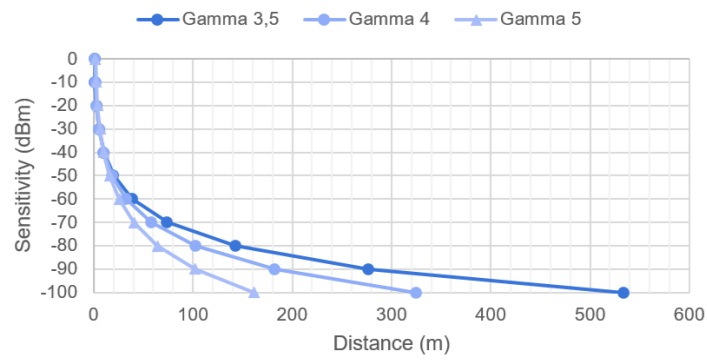


(c) Band 7 (2600 MHz).

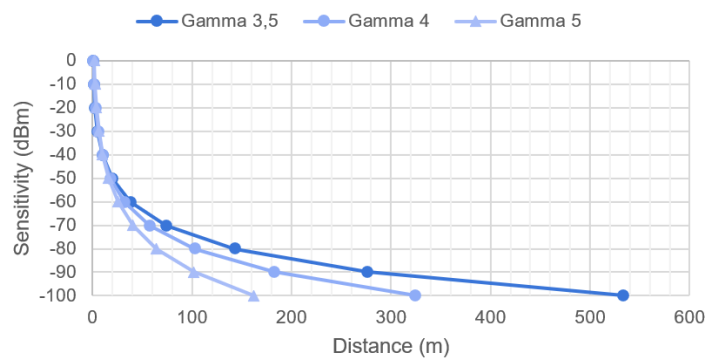
Figure 3.21: LTE Received Power Log Distance Prediction.



(a) Band 20 (800 MHz).



(b) Band 3 (1800 MHz).



(c) Band 7 (2600 MHz).

Figure 3.22: LTE Distance Log Distance Prediction.

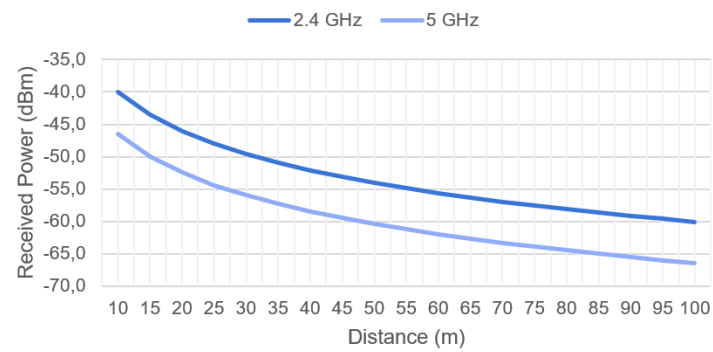
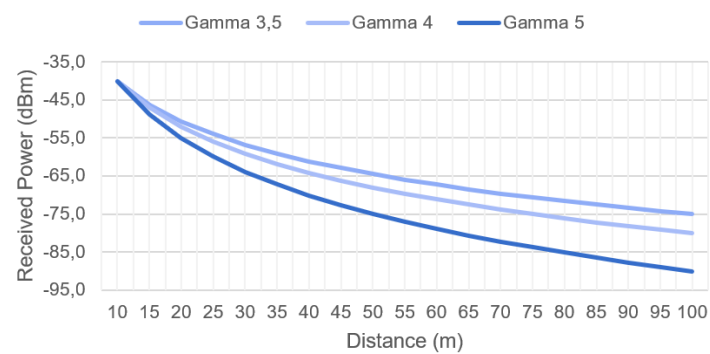
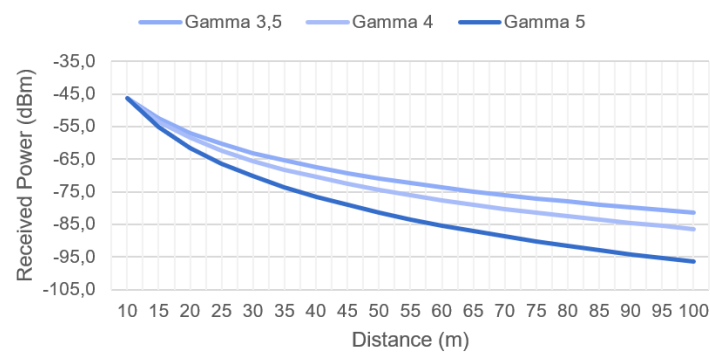


Figure 3.23: Wi-Fi Received Power Friis Prediction.



(a) 2.4 GHz.



(b) 5 GHz.

Figure 3.24: Wi-Fi Received Power Log Distance Prediction.

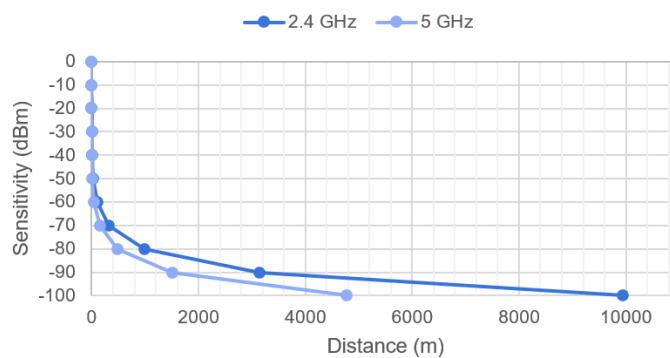
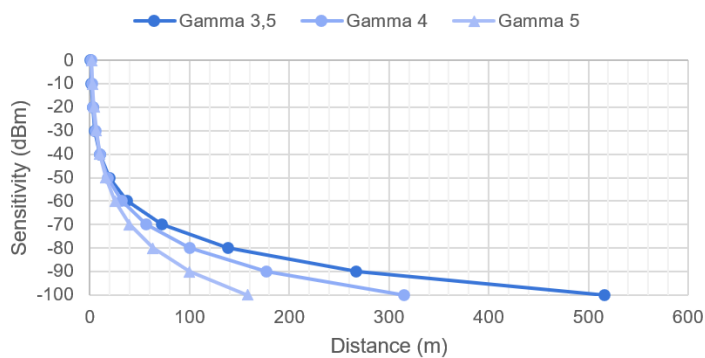
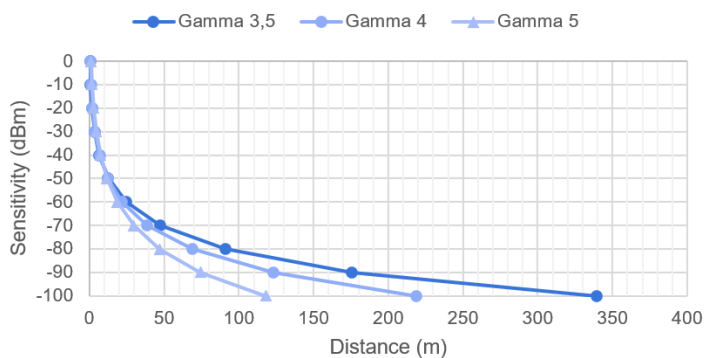


Figure 3.25: Wi-Fi Distance Friis Prediction.



(a) 2.4 GHz.



(b) 5 GHz.

Figure 3.26: Wi-Fi Distance Log Distance Prediction.

Chapter 4

Experimental Evaluation

This chapter has the objective of showing the whole process of proving the concept of our proposed solution and how well it performs given different environments and hardware configurations.

The LTE development and testing was done at home due to the Covid-19's situation which resulted in a non-controlled environment and an increased difficulty of finding and solving the issues that happened throughout the development. Later, when the confinement was no longer mandatory and we were allowed to use the INESC TEC labs under strict rules, it was possible to get a better understanding of what was happening in the system.

Both the LTE and Wi-Fi technology development required preliminary tests before the flying tests, with the objective of finding the best combination of hardware and the configuration that held the best practical results, confronting them with the theoretical predictions calculated in Chapter 3.

4.1 LTE Solution

4.1.1 Proof of Concept

The proof of concept explained in this subsection is primarily focused on srsLTE due to the complexity associated with OAI software referred in Section 3.3.4.

With the srsEPC running on a NUC, (c.f. Figure 4.1) we had the confirmation that everything was running as expected since every component of the EPC was initialized without any reported issue.

A second NUC (NUC B) was connected to the first one (NUC A) through a LAN connection using a router. NUC A ran the srsENB and looked for an EPC station to connect to. It then finds srsEPC (NUC B) and the association was made as reported in the Figure 4.2. After reading the configuration file referred in Section 3.3.3.2, the srsENB starts looking for the first available RF Front End (because we did not set a specific one to use since we were using multiple Front Ends). It first tries to find an UHD based Front End, from Ettus Research but it fails to find one because the one plugged to the NUC by the time the screenshot shown in Figure 4.2 was taken was the

```

bladerf2@bladerf2-desktop:~$ sudo srsepc
[sudo] password for bladerf2:

Built in Release mode using commit d045213 on branch master.

--- Software Radio Systems EPC ---

Reading configuration file /etc/srslte/epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf208, MNC: 0xff92
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.

```

Figure 4.1: Starting phase of srsEPC.

BladeRF 2.0 Micro xA9. Before setting the parameters from the srsENB configuration file, it also reads a configuration file for the bladeRF itself that basically sets AGC and both bias-tee amplifiers ON in order to prevent errors when srsENB tries to set them itself. At the same time, in the EPC side, we can see in Figure 4.3 the srsEPC receiving the request from srsENB to attach with some of its parameters being shown for better understanding of what is happening.

```

bladerf1@bladerf1-desktop:~$ sudo srsenb

Built in Release mode using commit d045213 on branch master.

--- Software Radio Systems LTE eNodeB ---

Reading configuration file /etc/srslte/enb.conf...
Opening 1 RF devices with 1 RF channels...
[INFO] [UHD] linux; GNU C++ version 5.4.0 20160609; Boost_105800; UHD_3.15.0.0-release
[INFO] [LOGGING] Fastpath logging disabled at runtime.
Opening USRP with args:
Error opening UHD: code 11
Opening bladeRF...
Set RX sampling rate 1.92 Mhz, filter BW: 1.92 Mhz
Setting frequency: DL=2655.0 Mhz, UL=2535.0 Mhz
set TX frequency to 2655000064
set RX frequency to 2535000062
Setting Sampling frequency 5.76 MHz

==== eNodeB started ====
Type <t> to view trace
Set RX sampling rate 5.76 Mhz, filter BW: 4.61 Mhz

```

Figure 4.2: Starting phase of srsENB and connection to srsEPC.

The system is now fully working and providing a full-fledged LTE communications network. That is when we turn on our commercial mobile phone, which has an UMTS SIM configured to match the MME database that srsEPC uses to authenticate the User Equipment. In Figure 4.4 we

```

SPGW S11 Initialized.
SP-GW Initialized.
Received S1 Setup Request.
S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b
S1 Setup Request - MCC:208, MNC:92, PLMN: 194601
S1 Setup Request - TAC 7, B-PLMN 0
S1 Setup Request - Paging DRX 2
Sending S1 Setup Response

```

Figure 4.3: srsEPC logs for the srsENB association.

can observe the mobile phone changing from flight mode to looking up for available networks, where it will find the srsLTE network and successfully authenticate to it. Turning flight mode on and off was the fastest way we had knowledge of to force the mobile phone to look for new networks. This was important because some of the configurations were really unstable and we needed to assure the UE was searching for new networks so that we could debug where the source of the issues was.



(a) UE with flight mode.



(b) UE connected to the srsLTE network.

Figure 4.4: User Equipment connection to the network.

In Figure 4.5 (and in Appendix B.2) we can see the output of the software when a new UE is connected to the network and successfully authenticated. The srsEPC is highly verbose. We will not go into much detail herein because the control component of the software is outside of the scope of this work and will not be analysed during the development of the system, as long as it functions as expected. The srsENB feedback, however, has lots of metrics associated that we used to better understand what was happening and if the system was working without problems. The values of Figure 4.5 only represent control messages that were being transmitted between both parts.

These are the metrics that are reported by the srsUE:

- **CQI** - Channel Quality Indicator is a metric reported by the UE and it ranges from value 1 to value 15 being 15 the best indicator;
- **RNTI** - Radio Network Temporary Identifier is the UE identifier which is valued 46 in this scenario. Everytime the UE disconnects and reconnects it might be given a different RNTI;
- **RI** - Used when we have multiple antennas (MIMO) it is a Rank Indicator reported by the UE. It is always 0 because we are not using MIMO which means that this metric has no interest for this scenario. However it represents how well the antennas are working together, in terms of correlation to each other [21];

- **MCS** - Value represents the number of spatial streams being used, the modulation type as well as the coding rate being used in real time. This combination of values will result on a Data Rate in Mbit/s depending on the Bandwidth used and Guard Interval time. Ranges from 0 to 28 in srsLTE and it has a value for Downlink communication and a different one for the Uplink;
- **BRATE** - Similar to MCS it also has a different value for downlink and uplink and it shows the Bit Rate in bits/sec;
- **BLER** - Block Error Rate is another metric to measure the quality of the communication. It is represented as a percentage and having it with a low value means that communication is in-sync;
- **SNR** - Signal to Noise Ratio (dB);
- **PHR** - Power Headroom Report Metric that shows how much power left can the UE transmit until it reaches the UE Max transmission power [22]. When the value is negative it means that it has surpassed the limit allowed to transmit. It is measured in dB;
- **BSR** - Buffer States Report tells us the data waiting to be transmitted at a give time, as reported by the UE. The value is expressed in bytes.

```

==== eNodeB started ===
Type <t> to view trace
Set RX sampling rate 5.76 Mhz, filter BW: 4.61 Mhz
t
Enter t to stop trace.
RACH: tti=4341, preamble=47, offset=20, temp_crnti=0x46

-----DL-----UL-----
rnti cqi  ri  mcs  brate  bler  snr  phr  mcs  brate  bler  bsr
46  0.100  0 0.100  0.0  0%  0.0  0.0  0.0  56.0  0%  0.0
User 0x46 connected
46  14.4  0 0.100  2.84k  0%  0.0  19.2  0.0  31.5k  0%  0.0
46  9.68  0 0.100  0.0  0%  0.0  19.0  0.0  8.53k  0%  0.0
46  12.1  0 0.100  0.0  0%  0.0  20.5  0.0  8.53k  0%  0.0
46  11.8  0 0.100  0.0  0%  0.0  21.0  0.0  8.53k  0%  0.0
46  13.3  0 0.100  0.0  0%  0.0  0.0  0.0  0.0  0%  0.0
46  13.8  0 0.100  0.0  0%  0.0  0.0  0.0  0.0  0%  0.0
46  13.3  0 0.100  0.0  0%  0.0  21.7  0.0  12.8k  0%  0.0
46  13.8  0 0.100  0.0  0%  0.0  22.5  0.0  8.53k  0%  0.0
46  14.2  0 0.100  0.0  0%  0.0  0.0  0.0  0.0  0%  0.0
46  11.8  0 0.100  0.0  0%  0.0  0.0  0.0  0.0  0%  0.0

```

Figure 4.5: UE connects to srsENB.

After establishing the connection the next step was to do an *iperf3* test to assure we had a connection capable of sending and receiving data. On the User Equipment side we used a mobile app

named *PingTools* that is capable of hosting an *iperf3* server that we connected from the srsEPC. In Figure 4.6 we can see the homepage screenshot of the *PingTools* app. It shows three different windows, each of them giving different information regarding the network the UE is connected to. The first window shows 1) the IP that was given to the User Equipment by the srsEPC DHCP server and 2) the IP address of the gateway; the last IP address, marked as Unavailable, is the public IP address of the network that was not configured because we did not want the network to be connected to the Internet. The second window shows a real time graph that plots the signal strength of the connected network. The last window also plots a real time graph that gives information regarding the amount of data that is being downloaded and uploaded per second. This screenshot was taken before starting the *iperf3* server, which means that the data being sent or received were control messages.

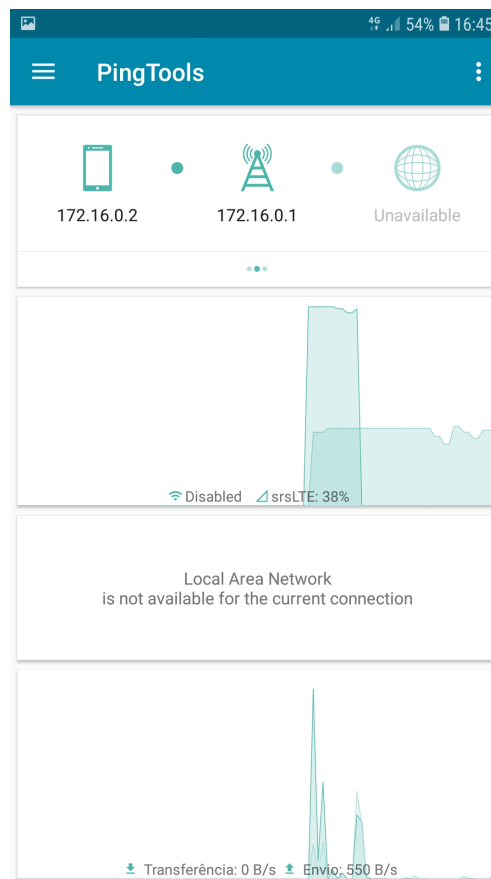


Figure 4.6: Screenshot of PingTools Homepage.

We started a proof of concept test running the *iperf3* server on the PingTools App on reverse mode which means that it was the server sending data to the client. In this scenario the server is the User Equipment and the client is the srsEPC. In Figure 4.7 we present an example log of the srsENB while the *iperf3* test is running and how it looks on the User Equipment side in Figure 4.8. Since the *iperf3* information was being registered both in the UE side and the srsEPC side, it was

decided to save the log of the srsEPC side only to avoid duplicated information that would give no extra value to the captured data.

-----DL-----						-----UL-----					
rnti	cqi	ri	mcs	brate	bler	snr	phr	mcs	brate	bler	bsr
47	11.6	0	0.100	142k	0%	0.0	16.0	0.0	6.04M	0%	64.3k
47	10.9	0	0.100	146k	0%	0.0	15.1	0.0	6.04M	0%	76.4k
47	10.8	0	0.100	143k	0%	0.0	14.4	0.0	6.04M	0%	66.0k
47	11.0	0	0.100	140k	0%	0.0	14.5	0.0	5.33M	0%	55.3k
47	11.4	0	0.100	115k	0%	0.0	12.7	0.0	4.56M	0%	44.2k
47	9.80	0	0.100	126k	3.3%	0.0	12.0	0.0	5.19M	0%	34.4k
47	11.6	0	0.100	114k	0.4%	0.0	17.2	0.0	5.62M	0%	39.8k
47	11.2	0	0.100	175k	1.6%	0.0	22.0	0.0	6.04M	0%	65.8k
47	10.2	0	0.100	155k	0%	0.0	19.1	0.0	6.04M	0%	67.1k
47	10.1	0	0.100	122k	0%	0.0	18.3	0.0	6.04M	0%	62.5k
47	10.1	0	0.100	122k	0%	0.0	18.3	0.0	6.04M	0%	75.1k

Figure 4.7: srsENB while *iperf3* is running.

With OAI the process of proving that the system was functional was similar to the one already described with srsLTE. However, it was more difficult to explain with the help of screenshots and snippets of the terminal logs since they were overly verbose and specific. This turned out to be an advantage to troubleshooting but that is not within the scope of this section.

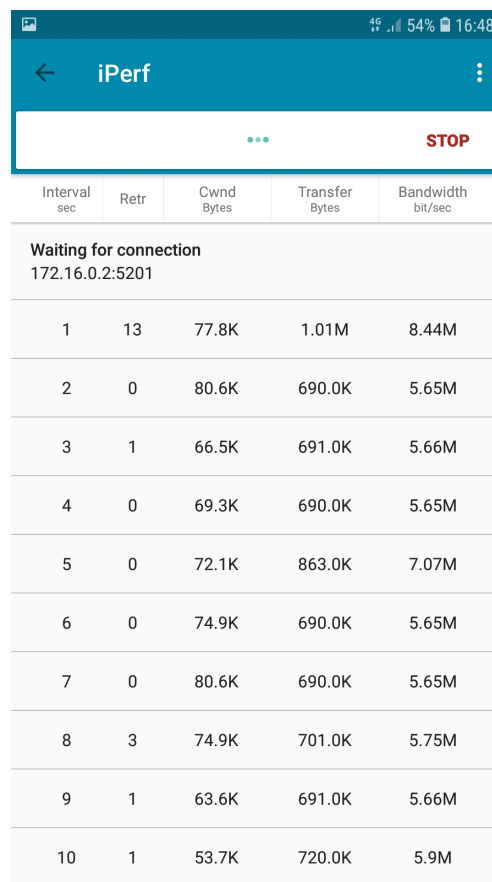
4.1.2 Scripts Used

With the need of multiple steps in order to have the system fully operational to do an *iperf3* test, there was the need to develop some scripts that helped to automate the process of getting the system up and running and to make it faster, thus lowering chances of human error.

Listing 4.1: Script used at the start of srsEPC.

```
#!/bin/bash
vmstat -wtn 1 80 > sysEPC.log &
mpstat -P ALL 1 > cpuEPC.log &
sudo srsepc | ts | tee logEPC.log
\label{fig:srsepc_start_script}
```

Listing 4.1 shows the short bash script designed to capture the desired logs while having the srsEPC software running. All the different logs that were captured in srsEPC host computer, were also captured in the srsENB side using a similar bash script. The first one, named *vmstat*, was the tool chosen to monitor the system performance in terms of memory usage. It also reports other metrics like the CPU usage as we can see in Figure 4.9 but we wanted to observe how the CPU load was distributed throughout the different CPUs, so we had to use a different tool for that, named *mpstat*. The options used were to leave the output in a log style format where we could



The screenshot shows the iPerf application interface. At the top, there's a status bar with '4G', signal strength, '54%', and '16:48'. Below it, a blue header bar contains a back arrow, the text 'iPerf', and a menu icon. A white box below the header has three dots and a red 'STOP' button. The main area displays a table with the following columns: Interval sec, Retr, Cwnd Bytes, Transfer Bytes, and Bandwidth bit/sec. The table content is as follows:

Interval sec	Retr	Cwnd Bytes	Transfer Bytes	Bandwidth bit/sec
Waiting for connection 172.16.0.2:5201				
1	13	77.8K	1.01M	8.44M
2	0	80.6K	690.0K	5.65M
3	1	66.5K	691.0K	5.66M
4	0	69.3K	690.0K	5.65M
5	0	72.1K	863.0K	7.07M
6	0	74.9K	690.0K	5.65M
7	0	80.6K	690.0K	5.65M
8	3	74.9K	701.0K	5.75M
9	1	63.6K	691.0K	5.66M
10	1	53.7K	720.0K	5.9M

Figure 4.8: PingTools Reporting the Data transferred during the *iperf3*.

synchronize timestamps between other logs to ease debugging. There was a 1 second interval between reports that would run for the specified time duration in the command, 80 seconds in this example. That log would be redirected to a file that we named *sysEPC.log*.

```
procs -----memory----- --swap-- --io-- --system-- -----cpu----- -----timestamp-----
r  b      swpd      free      buff      cache    si  so    bi  bo    in  cs  us  sy  id  wa  st      WEST
2  0          0    7122148    157520    495916    0  0     9   1   19  30  0  0 100  0  0 2020-05-18 14:14:27
1  0          0    6906620    157528    496080    0  0     0  88  568 2598 6  5  89  0  0 2020-05-18 14:14:28
0  0          0    6906704    157528    496040    0  0     0   0  63  123 0  0 100  0  0 2020-05-18 14:14:29
0  0          0    6906704    157528    496040    0  0     0   0  44  72  0  0 100  0  0 2020-05-18 14:14:30
0  0          0    6906704    157528    496040    0  0     0   0  33  59  0  0 100  0  0 2020-05-18 14:14:31
0  0          0    6905464    157528    496040    0  0     0   0  77  129 0  0 100  0  0 2020-05-18 14:14:32
0  0          0    6905464    157536    496036    0  0     0   0  64  37  69  0  0 100  0  0 2020-05-18 14:14:33
0  0          0    6905464    157536    496036    0  0     0   0  27  54  0  0 100  0  0 2020-05-18 14:14:34
```

Figure 4.9: srsEPC example log provided by vmstat tool.

The *mpstat* command was used with the option P followed by ALL that means we wanted every processor to have statistics reported. This resulted in a report such as the example on Figure 4.10. It also has a timestamp of the system and a different line for each of the CPUs of the NUC. In this case we used an example from the srsENB host but the srsEPC is also capturing a similar

one. Both this tools were useful to understand how would the systems behave during *iperf3* tests, during idle times with the software running or after long periods of time.

Linux 4.4.0-178-lowlatency (bladerf1-desktop) 18-05-2020 _x86_64_ (2 CPU)											
14:14:29	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
14:14:30	all	5,00	0,00	6,00	0,00	0,00	0,00	0,00	0,00	0,00	89,00
14:14:30	0	7,92	0,00	11,88	0,00	0,00	0,00	0,00	0,00	0,00	80,20
14:14:30	1	2,97	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	97,03
14:14:30	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
14:14:31	all	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00
14:14:31	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00
14:14:31	1	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00

Figure 4.10: srsENB example log provided by *mpstat* tool.

We would then start the srsEPC with elevated privileges and then pipe it to *ts* tool so that the output of srsEPC was then sent as an input for *ts*, whose job is to insert a timestamp on the srsEPC output so we could synchronize every file captured. This log was then redirected to another file named *logEPC.log*.

As stated in the previous section, after successfully attaching srsENB to the srsEPC, in order to start testing we run an *iperf3* server on the commercial user equipment through PingTools. The *iperf3* server then waited for a connection to his own IP address given by the srsEPC itself. On the srsEPC side we started the client, connecting to the srsUE and injecting Transmission Control Protocol (TCP) packets at the maximum rate it can achieve, during a given period of time. The results are also logged into a new log file such as the one shown in Figure 4.11, which also contains the timestamp in order to synchronize with other files, if needed.

```

Abr 13 15:13:13 [ 4] 177.00-178.00 sec 1.93 MBytes 16.2 Mbits/sec 0 102 KBytes
Abr 13 15:13:13 [ 4] 178.00-179.00 sec 1.43 MBytes 12.0 Mbits/sec 2 77.8 KBytes
Abr 13 15:13:13 [ 4] 179.00-180.00 sec 1.93 MBytes 16.2 Mbits/sec 0 97.6 KBytes
Abr 13 15:13:13 - - - - -
Abr 13 15:13:13 [ ID] Interval Transfer Bandwidth Retr
Abr 13 15:13:13 [ 4] 0.00-180.00 sec 307 MBytes 14.3 Mbits/sec 401 sender
Abr 13 15:13:13 [ 4] 0.00-180.00 sec 306 MBytes 14.3 Mbits/sec receiver
Abr 13 15:13:13
Abr 13 15:13:13 iperf Done.
```

Figure 4.11: Snippet of the end of an *iperf3* log captured on srsEPC side.

On the UE side there was no need of capturing the same report from *iperf3* through PingTools because it would have the same values. However, we used another app named Network Cell Info Lite whose objective was to log from time to time (not a fixed value) some metrics on the UE side, such as the ones mentioned in Figure B.1. We could not find other app that would do the same with the option to export to a CSV file and in this one we could not control the frequency at which the APP would report the values. The app would report values at different intervals of time so we could not synchronize properly with the srsEPC or srsENB. It was still helpful to realise the system limitations and have a feeling of how it was changing for different distances. In this

scenario, the network type we were using was the LTE type, and the relevant values we took from this different app was the timestamp with an RSRP associated. In a similar way, if the test was being run on the OAI software, we would capture every previously mentioned log file except in this case, the *oaiEPC* log files would be split into 3 different files being them the *logHSS*, *logMME* and *logSPGW*.

After having everything prepared from the capture, we needed to capture every single file that was written, so we developed the *pullscript* bash script, shown in Figure B.3, that asks what software you are using and it then transfers through Secure Copy Protocol (SCP) to the control computer. To keep track of what was configured and used while running that test, we also transferred the *enb.conf* and *epc.conf* files. Due to the big number of tests done and capture files, there was a need to develop a Unique Identifier code for each one of them, making it easier to process the information and organize the folders and files better:

- **Hardware/Software:** Ranges from 1 to 6. The first 3 refer to tests done using srsLTE software and the last 3 to the ones done with OAI software. Each of the 3 numbers refer to tests done with BladeRFx40, BladeRF 2.0 micro xA9 and USRP B210 accordingly;
- **LTE Frequency Band:** Band 3, 7 and 20 configured with the EARFCN parameter in srsENB configuration file;
- **Number of Resource Blocks:** 15, 25, 50 and 75 configured with the *n_prb* parameter in srsENB configuration file;
- **Reverse Mode:** 1 (DL) or 0 (UL) depending if we are doing a DownLink test or Uplink making use of Reverse option of *iperf3*;

4.1.3 Experiments in Non-controlled Environment

After making an end to end connection test we were able to start debugging and figuring out what kind of configurations did not work and why it was happening. We realised that there were multiple problems happening and we needed to understand the source of them.

We developed a work plan for the LTE testing phase that was done during the Covid-19 confinement situation at home, in a non-controlled environment. This work plan involved testing all combinations possible considering the following variables:

- **Software:** srsLTE, OAI;
- **RF Front End Hardware:** BladeRFx40, BladeRF 2.0 Micro xA9 and USRP B210;
- **LTE Band:** 20 (800 MHz), 3 (1800 MHz) and 7 (2600 MHz);
- **Number of Resource blocks (NRB):** 6 (1.4 MHz BW), 15 (3 MHz BW), 25 (5 MHz BW), 50 (10 MHz BW) , 75 (15 MHz BW) and 100 (20 MHz BW) - OAI did not support 6 and 15 NRBS.;

- **Mode:** Uplink or Downlink;
- **Distance:** 1 Meter and 3 Meters with line of sight (Max achievable with Home environment limitation);
- **Iteration:** 5 Tests per configuration during different days and times of the day.

Besides these variables, the duration of each test was considered in order to understand how the system would perform during a larger test duration like, for example, from 10 minutes up to 2 hours. However, the combination previously presented was used primarily to perform a preliminary connectivity study to reach some conclusions, so we used test duration of 3 minutes.

Table 4.1 and Table 4.2 present an overall scenario of what was achieved with srsLTE and OAI considering all bands and bandwidths. In the tables we use the letters C, N, and S with the following meanings:

- **C** : Connected - Used when the User Equipment successfully connects to the network. This indicator does not mean that the connection had no issues whatsoever.
- **N** : No Connection - Used for cases when the eNB software was running but the User Equipment could not detect any network and there was no feedback from the eNB logs;
- **S** : Synchronization Problems - This is combined with either C to indicate that the connection has synchronization problems and is not stable at all.

It is worth mentioning that some of the obstacles that we came across with the development of this work could not be simply addressed with a letter and will be later discussed in this section.

RF Front End	Band	6 NRBs	15 NRBs	25 NRBs	50 NRBs	75 NRBs	100 NRBs
BladeRFx40	800 MHz	N	C	C	C	C/S	N
	1800 MHz	N	C	C	C	C/S	N
	2600 MHz	N	N	C	C	C/S	N
BladeRF 2.0 Micro xA9	800 MHz	N	N	C	N	N	N
	1800 MHz	N	N	C	N	N	N
	2600 MHz	N	N	C	N	N	N
USRP B210	800 MHz	N	N	N	N	N	N
	1800 MHz	N	C	C	C	C/S	N
	2600 MHz	N	C	C	C	C/S	N

Table 4.1: srsLTE results of the preliminary connectivity study.

There are some conclusions that we can draw from the observed results. The first tests happened with the use of srsLTE software and after looking at what we could achieve with it, we were more selective and focused on what we wanted to test with OAI software.

In Table 4.1 we can see that from 75 NRBs to 100 NRBs no connection was established without problems. In those scenarios, there were 2 different situations that occurred. For the 75 NRBs configurations, the amount of information was too much and it resulted in synchronization

RF Front End	Band	25 NRBs	50 NRBs	100 NRBs
BladeRFx40	800 MHz	N	N	N
	1800 MHz	N	C	N
	2600 MHz	N	C	N
BladeRF 2.0 Micro xA9	800 MHz	C	C	N
	1800 MHz	N	N	N
	2600 MHz	C	C	N
USRP B210	800 MHz	N	N	N
	1800 MHz	N	C/S	N
	2600 MHz	N	C/S	N

Table 4.2: OAI results of the preliminary connectivity study.

problems between the components of the network. When considering even higher bandwidth with 100 NRBs (20 MHz), there was no feedback from any of the components and it simply did not have any logs to explain what was happening.

We did not reach any conclusion regarding the fact that lower NRBs also had problems with the connection simply not showing any feedback of what was happening. The BladeRF 2.0 Micro xA9 did not connected to the eNB with any configurations using more or less than 25 NRBs.

When considering specific bands like the band 20 (800 MHz) we could observe that the USRP B210 had issues and did not attach to the UE when using such configuration.

There were configurations that worked for OAI software and did not work for srsLTE. As an example, the BladeRF 2.0 Micro xA9 only connected using srsLTE with 25 NRBs but when we used OAI we got a stable connection with 50 NRBs. On the other hand, with 50 NRBs, the same configuration for USRP B210 had synchronization problems with OAI interface and when testing it with srsLTE it showed no problems. This is also worth mentioning because there were some unidentified issues with the software as well.

For the OAI testing we only used 25, 50 and 100 NRBs because the version used only supported 5, 10, and 20 MHz of bandwidth.

After finishing the preliminary connectivity study, we still needed to verify how the system would function with longer test durations and different distances between the eNB and the commercial user equipment such that we could reach some predictions on what we could get on an open scenario testbed.

In a brief note, while doing tests that took between 10 min to 2 hours, there were several times the UE would disconnect with problems of reconnecting. Sometimes it was needed to manually force looking up for networks, otherwise the UE would not connect and it seemed that the RSRP reported by the UE was not stable, which could mean that the RF Front Ends were not transmitting at a constant transmitting power. During those tests we also noticed that the CPU load and RAM load would remain low, which meant that the NUC performance was not the main issue.

The results when increasing the distance from 1 to 3 meters were also not as expected. The quality and rigour of the scenario where the testing took place was in poor conditions (indoor with lots of obstacles and some movement in the scenario while tests were running) since we could not

RF Front End	Band	RSRP (dBm)	RSSI (dBm)	Friis (dBm)	Difference
BladeRFx40	800 MHz	-74.8	-50	-24.5	25.5
	1800 MHz	-89.6	-64.8	-31.5	33.3
	2600 MHz	-88.9	-64.1	-34.7	29.4
BladeRF 2.0 Micro xA9	800 MHz	-82.3	-57.6	-22.5	35.1
	1800 MHz	-86.7	-61.9	-29.5	32.4
	2600 MHz	-94.2	-69.4	-32.7	36.7
USRP B210	800 MHz	-	-	-20.5	-
	1800 MHz	-85.0	-60.23	-27.5	32.7
	2600 MHz	-90.5	-65.73	-30.7	35.0

Table 4.3: Comparison of Theoretical and Experimental RSSI.

have a proper testbed during the pandemic. However, it is not justifiable, with a strong line of sight component, having synchronization problems and connectivity issues when increasing the range from 1 meter to 3 meters.

Table 4.3 was created with the intent of showing the RSSI differences of tests with a specific configuration using Friis path loss model for the lowest measured distance (1 meter). It was chosen the srsLTE software with 25 NRBs (5 MHz bandwidth) considering both uplink and downlink tests for every frequency band since it was the configuration that most of the RF Front Ends were compatible with.

The RSRP value is the average of all tests collected with the UE software for that given configuration. The value is then converted to an RSSI Value where we made the assumption that at 1 Meter distance, the connection was on Max load and with a high SNR which would result in the expression 4.1 in which NRB stands for the Number of Resource blocks used in the configuration.

$$RSSI (dBm) = RSRP (dBm) - 10\log_{10}(12NRB) \quad (4.1)$$

The difference column refers to the difference between the RSSI values that were experimentally obtained and the RSSI values expected for those conditions. As we can see, the values are much lower than what we expected, which could explain the synchronization and lack of connectivity for the problems at higher but still small distances.

4.1.4 Laboratory Measurements

With the lifting of restrictions imposed by the Covid-19 situation, we were able to use INESC TEC Lab, and a Vector Network Analyzer (VNA), for a better understanding of what was happening in our system that would justify such low results.

The first problem that we thought could be responsible for it was the quality and performance of the antennas. We measured them by a Return Loss (RL) across the frequency spectrum of interest (800 MHz up to 2600 MHz). The return loss is calculated following Equation 4.2 where RL stands for Return Loss, P_i is the incident power, and P_r represents the reflected power. It is desired to

have a return loss of atmost -10 dB, which means that only 10% of the input power is lost in order to assume that an antenna is tuned for the frequency of operation.

$$RL(dB) = 10 \log_{10} \left(\frac{P_i}{P_r} \right) \quad (4.2)$$

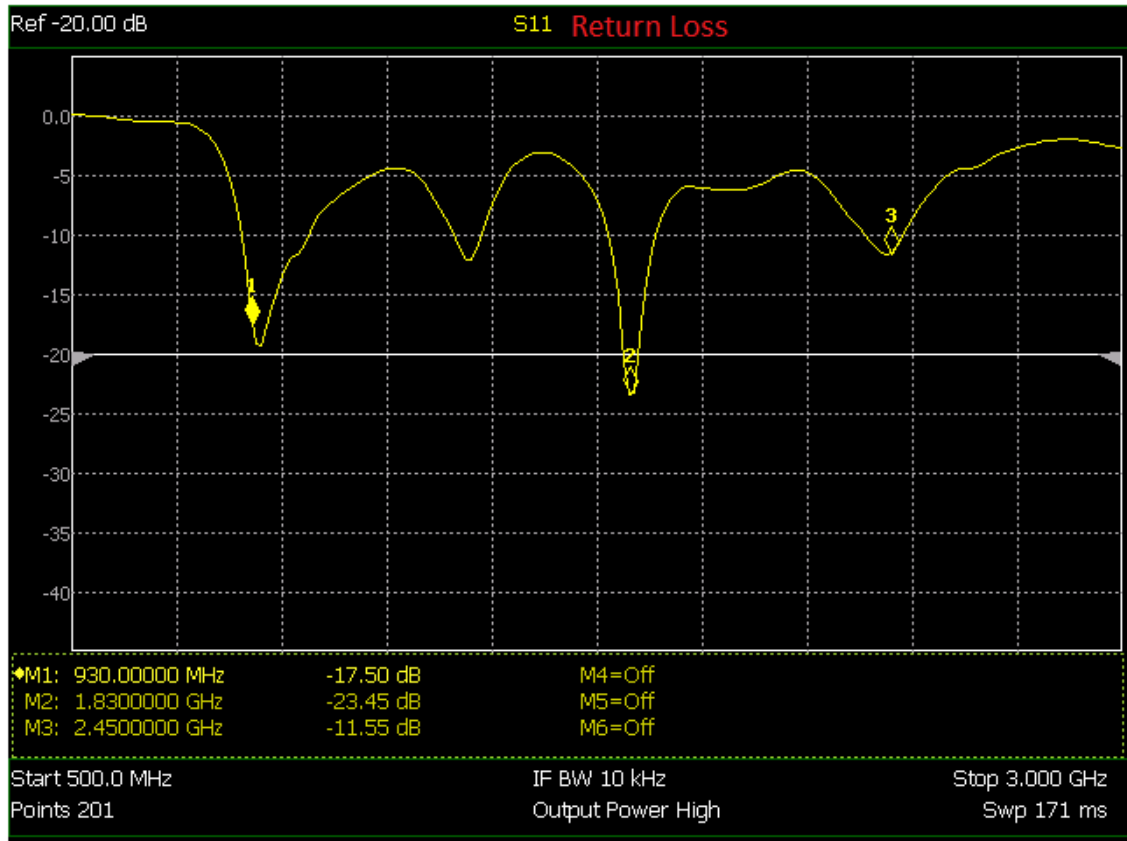


Figure 4.12: Return Loss of Triband Antenna [3].

Figure 4.12 is advertised in the manufacturer website in the description section, and as we can observe, for the three measured frequencies (930 MHz, 1830 MHz and 2450 MHz) the antenna has values below the desired threshold of atmost -10 dB of return loss. Figure 4.13 and Figure 4.14 are return loss plots of two different units (named Antenna A and B respectively) of the same advertised antenna in [3].

There are two main aspects that are worth explaining in order to draw conclusions. The first one is that in the manufacturer's website it is stated that the tri-band antenna is a general purpose antenna that covers the range of frequencies between 700 MHz and 2600 MHz. However, there is no horizontal axis in any of the presented plots that could help us understand the scale associated to it. To overcome the problem, for the frequencies that were chosen for Downlink and Uplink, we marked them in the presented figures. As we can see, none of the markers are below the desired -10 dB mark of return loss, with exception of marker 5 (1841 MHz) in Figure 4.14 with a return

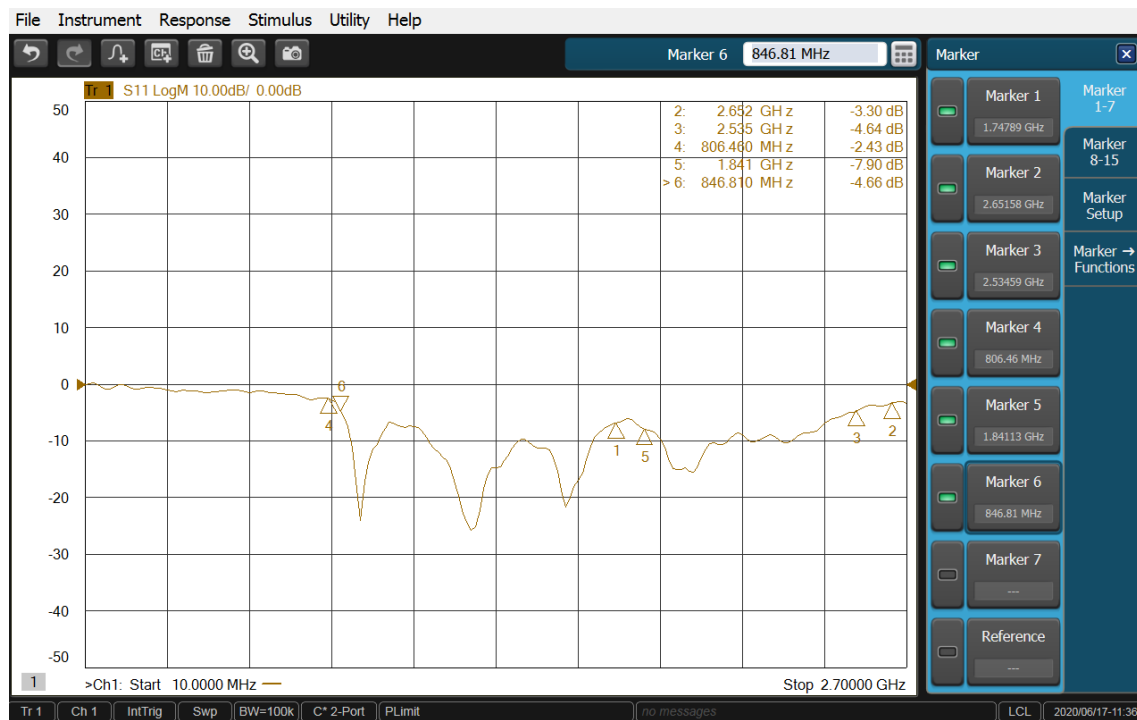


Figure 4.13: Return Loss of Triband Antenna A.

loss value of -10.83 dB.

The second one is that the return loss curve of Antenna A is different than the curve of Antenna B which is also different from the advertised performance plot in the manufacturer website. Besides that, during these tests, we also found out that one of the antennas had some bad connection in the terminal.

In summary, we concluded that the antenna also had a significant contribution to the results achieved, due to the poorly tuned factor and bad connectivity found in one of the purchased antennas.

Since we found problems associated with Nuand hardware, we also decided to test the PA and LNA referred in Chapter 3. In Figure 4.15 and Figure 4.16 we can observe different markers considered for a better understanding of what we could achieve with the amplifiers.

For the PA BT-100 it is advertised in the manufacturer website [23] a typical gain of +15.6 dB when considering a frequency of 200 MHz and a typical gain of +10.7 dB at 2.4 GHz. According to 4.15 we obtained for those frequencies +14.46 dB and +9.60 dB accordingly, which were values within our expectations. Unfortunately, for band 3 and 7 we observe gain values below the +10 dB mark.

For the LNA BT-200 it is advertised in the manufacturer website [24] a typical gain of +20.2 dB when considering a frequency of 200 MHz and a typical gain of +16 dB at 2.4 GHz. Following the same logic used for the PA measurements, in Figure 4.16 we obtained for those frequencies

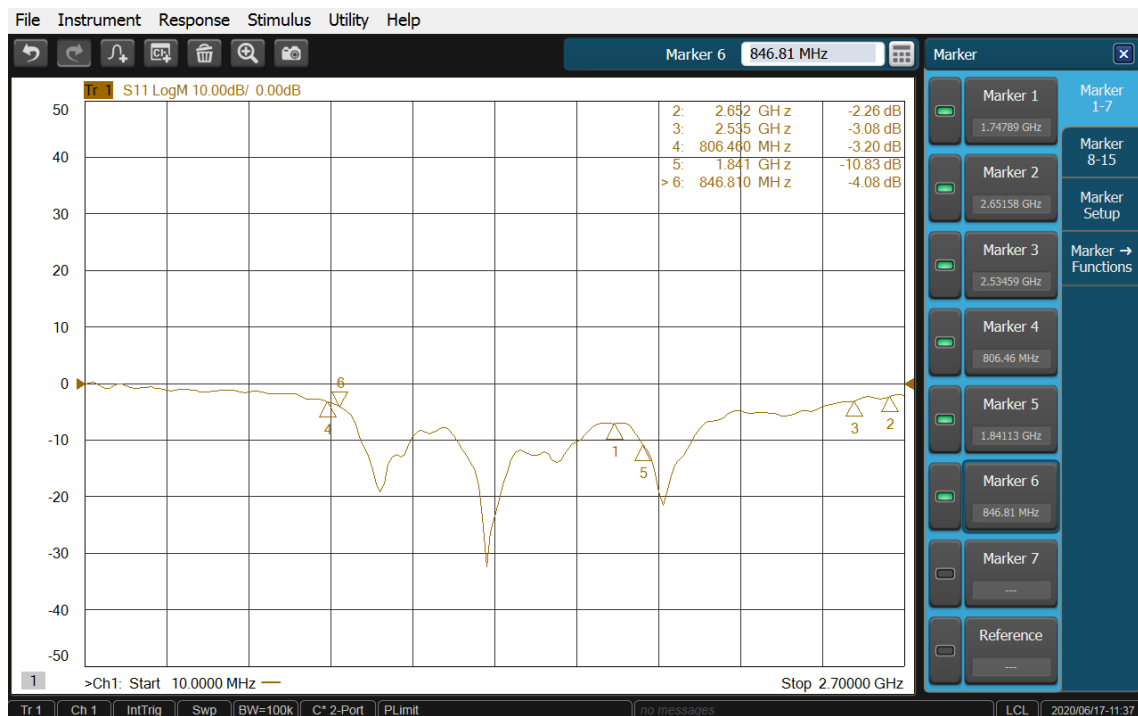


Figure 4.14: Return Loss of Triband Antenna B.

+17.46 dB and +15.58 dB accordingly. This LNA did not have the same performance problems that were noticed for the PA since the lowest gain within the 3 LTE Bands considered was +14.66 dB which is still good for what we expected regarding the link budget.

As a sidenote, the XB-300 Amplifier presented previously for the BladeRFx40 had a wideband LNA with a bandwidth from 500 MHz up to 5 GHz, but the PA component was limited from 2.4 GHz up to 2.5 GHz, which did not meet the requirements to be considered for the chosen LTE frequencies, thus no tests were carried out for that amplifier.

In summary, the PA and LNA that were used with the BladeRF 2.0 Micro xA9 were well designed and had no relevant problems that could explain the results obtained.

Figure 4.17 represents the TX power observed when directly connecting the TX connector of the BladeRF 2.0 Micro xA9 with a plugged PA to the VNA. This screenshot of the VNA was taken when we were running srsENB software to deploy the network in band 20 considering the downlink frequency of 806 MHz. The TX power configured was the maximum allowed, and the number of resource blocks used were 6, in order to have the maximum TX power possible within the software limits which resulted in a low throughput network. Taking this into consideration, the plot shown in Figure 4.17, we can see that within the frequencies of operation there is an average power of -20 dBm, when it should be around +18, considering the maximum TX power is being transmitted with the help of the amplifier ($+8 + 11.83 = +19.83$ dBm). This is almost a 40 dBm difference from what was expected with this kind of hardware.

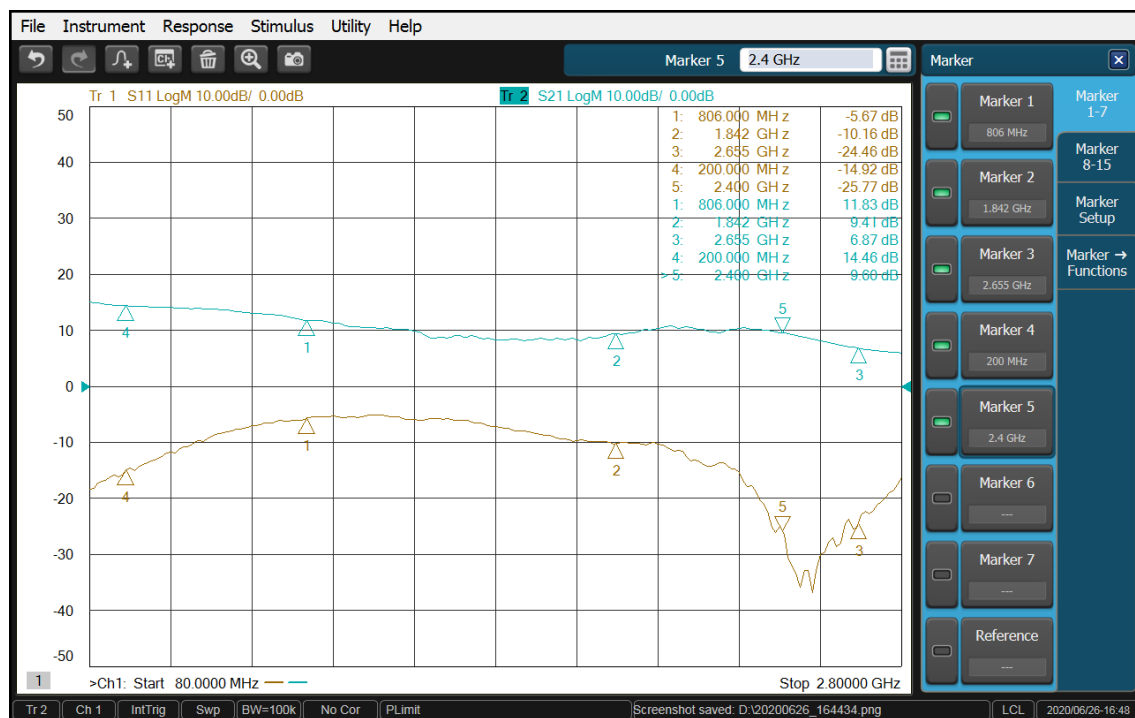


Figure 4.15: Return Loss and gain of BT-100.

When we repeat the Friis and Log Distance predictions for Band 20 (800 MHz), done previously with the new calculated TX power that we can obtain in the best conditions we obtain the plot presented in Figure 4.18. In this figure we can see that within a 100 meter range and using the log distance model with a gamma value of 3.5 we surpass the considered user equipment sensitivity of -100 dBm at 70 meters.

In summary, this technology proved not to be mature enough yet for implementation. The proposed solution is based on pillars such as the range and stability of the network communications, which could not be observed in the experimental evaluation.

4.2 Wi-Fi Solution

Before doing the flying tests, there was the need to prove that the system was well prepared and predict what would happen during those flying tests. The idea was to do tests with the Router and Client on the ground to reach some conclusions regarding the Wireless USB adapter that best performed, together with the channel that was better given the location. The location where the ground tests were carried out, was in front of the INESC TEC headquarters in Porto, inside the Faculty of Engineering of the University of Porto (FEUP) Campus.

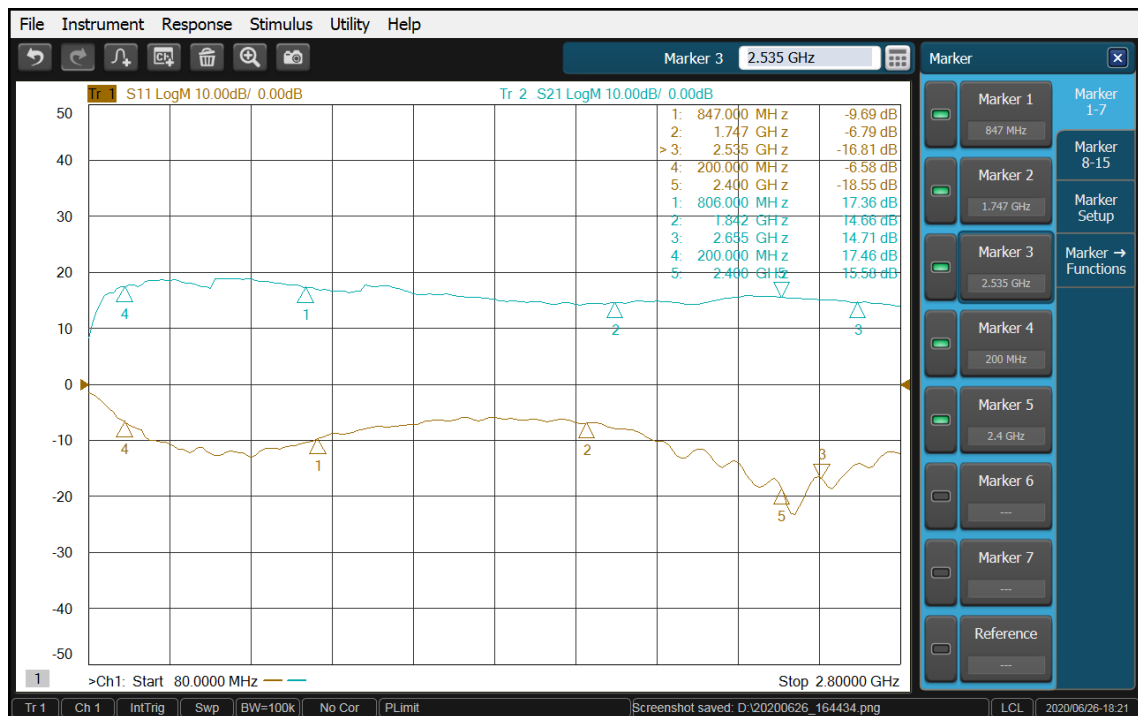


Figure 4.16: Return Loss and gain of BT-200.

4.2.1 Preparation of the Experiments

In Figure 4.19 we can see a representation of Google Earth 3D view of where the ground tests took place. The grey star marker is an approximation of the ground projection of the Router node, but the Router Node was actually located in a window at height of 12 meters from the ground level.

The Client Node was placed in 4 different spots of increasing distance marked with the colors: Blue, Yellow, Red and Green in order to reach some conclusions regarding distance limits, how the throughput and RSSI vary with the distance in this specific scenario, and other obstacles that could come in the way when doing tests with a flying router node.

We considered two different wireless USB adapters and 3 different channels. The tests were done with both Downlink and Uplink packet injections using *iperf3* and repeated five times each, for each different configuration, to gather enough data that could lead to some relevant statistical values that we could use to draw conclusions. On the client side, we used a Personal Computer ASUS X556UF running Linux Mint whose wireless adapter was a Qualcomm Atheros AR9565. We have no information regarding the antenna or its location inside the computer.

A packet analyzer software was running on both sides. In the Client Node it was used *tshark*, a terminal based version of the wireshark software, and the router used *tcpdump* since OpenWRT packet manager did not have a *tshark* package compatible with it.

The color lines of Figure 4.19 represent the ground distances of the four locations considered, colored by the respective marker's color. When considering the 12 meters high router node, the distances become:

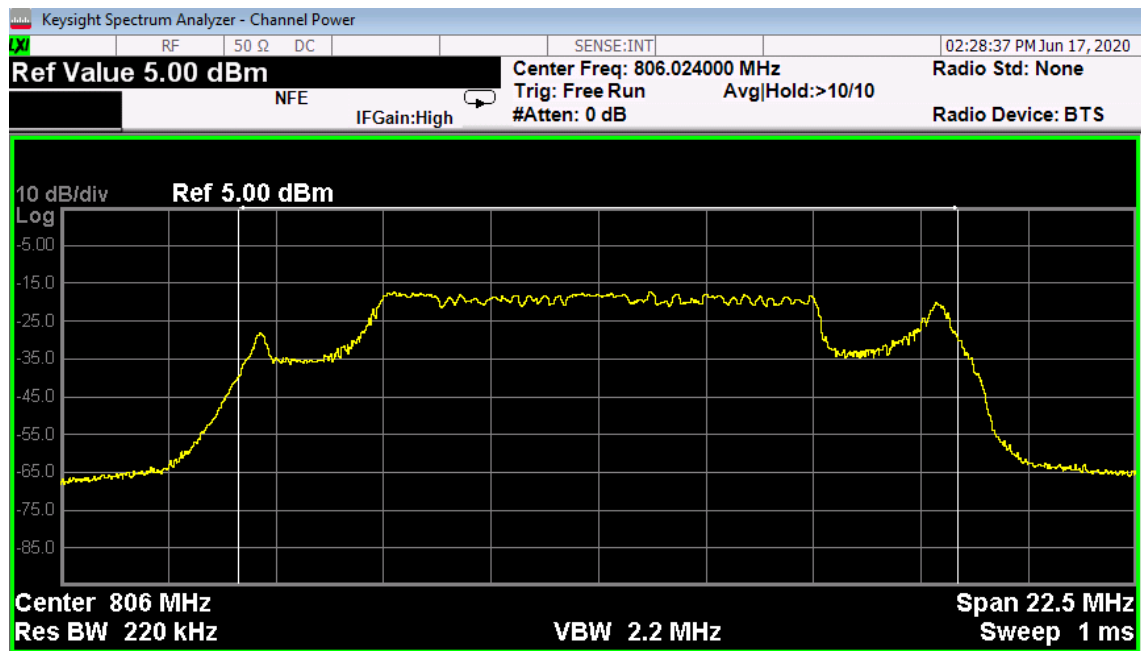


Figure 4.17: BladeRF 2.0 Micro xA9 Transmission Power with Amplifiers.

- Router Node - Location A: 17.7 Meters;
- Router Node - Location B: 34.6 Meters;
- Router Node - Location C: 65.1 Meters;
- Router Node - Location D: 89 Meters;

Before starting the ground tests, there was still some configurations needed to prepare the Client Node. Listing 4.2 is the script that was run to virtualize a network interface in *monitor mode*. It only needs to be run once, each time the computer restarts or whenever we turn off the network interface.

Listing 4.2: Client Node script to start Monitor Interface.

```
#!/bin/bash
iw phy phy0 interface add if_monitor type monitor
ifconfig if_monitor up
```

The objective of having a virtualized network interface is to successfully understand the 802.11 Radio Information of the captured packets. This protocol gives us many of radio related fields, however for the scope of this dissertation, the main ones we talked about and the ones we will use are listed below:

- **PHY TYPE:** mainly 802.11n but it may fall back to 802.11b or 802.11g;
- **MCS Index:** Only goes up to 7 since we are using SISO;

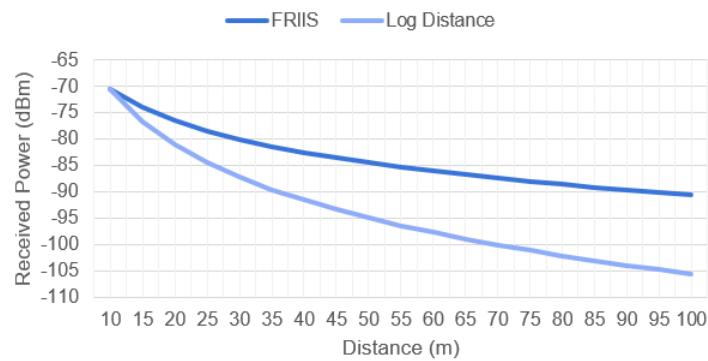


Figure 4.18: Friis and Log Distance Prediction considering a TX Power of -20 dBm.

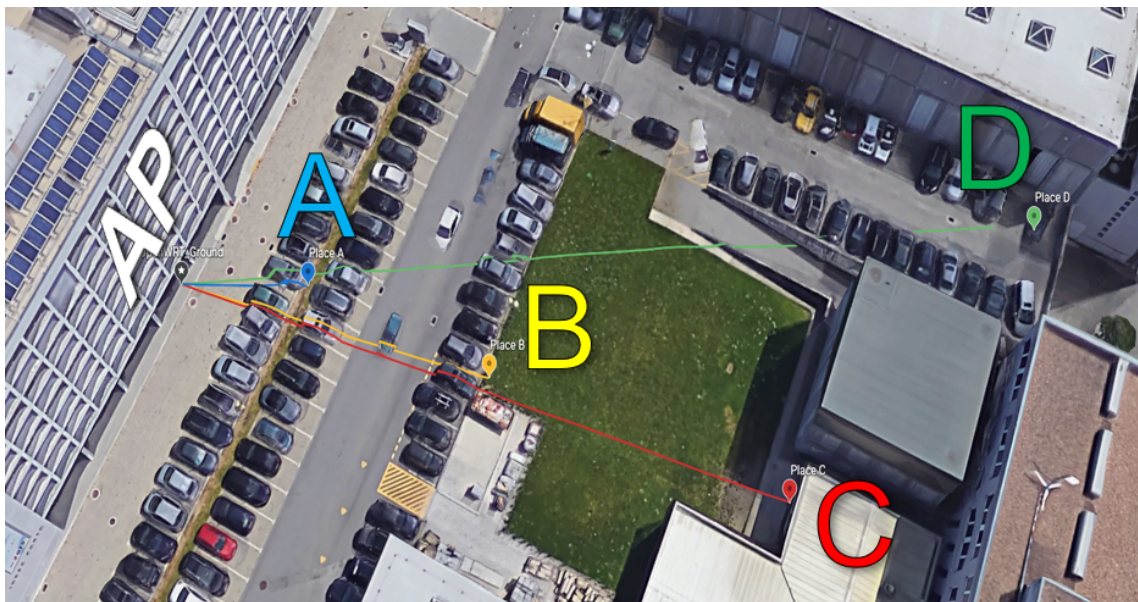


Figure 4.19: Location of the Ground test Spots.

- **Bandwidth:** Fixed at 20 MHz but could be fix to 40 MHz or auto;
- **Short GI:** Boolean (false means ts Long GI and no value means its not 802.11n);
- **Data Rate:** Raw Values measured in MBit/s;
- **Channel:** Only 1, 7 and 13 in this trials;
- **Frequency:** Frequency of the given channel;
- **Signal Strength (dBm):** RSSI value.

Listing 4.3 shows the script used in the Client Node to start the capture. It creates a file where the data will be written, and it starts an *iperf3* server on the *wlp3s0* wireless interface (it waits

for a client to connect), while running a *tshark* command. This *tshark* command captures on the *if_monitor* virtualized interface for a duration of 60 seconds. There is also an important parameter passed in this command that tells *tshark* to capture only the first 120 bytes of each packet. The objective is to avoid saving the payload information that is downloaded/uploaded using *iperf3* software without compromising the header informations we need, making the data smaller sized and easier to save and process.

Listing 4.3: Client Node script to start Capturing data.

```
#!/bin/bash

test_time=60
touch client_capture.pcap
chmod 666 ./client_capture.pcap

iperf3 -s &
tshark -i if_monitor -s 120 -a duration:$test_time
-w client_capture.pcap
kill $!
```

In the Router Node, similar to the script presented previously, we have a script whose job is to run an *iperf3* client connecting to the server and starting a TCP packet injection. In the meantime there was a *tcpdump* command running, capturing the first 120 bytes of each packet in the monitor interface *wlan1* for 60 seconds, just like *tshark* but adapted to the OpenWRT OS. Listing 4.4 shows the example for the Downlink script, however it was also used a script adapted for Uplink tests, where the *iperf3* would work with option for Reverse mode making the traffic flow from the server (Client Node) to the client (Router Node).

Listing 4.4: Router Node Downlink script to start Capturing data.

```
#!/bin/sh

iperf_runtime=60
interface=wlan1
IP_HOST=192.168.50.1
IP_ADDR=$1

tcpdump host $IP_HOST -i $interface -s 120 -ne
-w /root/capture.pcap &
iperf3 -c $IP_ADDR -t $iperf_runtime | tee iperf3.log
kill $!
```

After each test finished, the packet capture files needed to be transferred to the node where the data was going to be processed, which was the personal computer used as the Client Node. The

script below asks the user what Test is being done and saves the ID to name folders accordingly. Then, we use SCP to transfer the files from the Router Node and move the Client Node files to the previously created ID folder. The whole script looked like Listing 4.5.

Listing 4.5: Script to centralize the information in Client computer.

```
#!/bin/bash
IP_PI=192.168.50.1
HOSTNAME1=root

echo "Capture ID_?"
read id
cd /home/rubenq/WIFI_CAPTURES && mkdir id_$id && cd id_$id
cwd=$(pwd)

scp $HOSTNAME1@$IP_PI:/root/capture.pcap $cwd
scp $HOSTNAME1@$IP_PI:/root/iperf3.log $cwd
mv ../client_capture.pcap $cwd

echo "Finished!"
```

4.2.2 Data Processing

Due to the high number of tests done and captured files, there was a need to develop a Unique Identifier code for each one of them, making it easier to process the information and to better organize the folders and files:

- **Antenna Identifier:** B or P depending if we used the TL-WN722N or the Panda N600 respectively;
- **Reverse Mode:** DL or UL depending if we are doing a DownLink test or Uplink making use of Reverse option of *iperf3*;
- **Testing Channel:** 1,7 or 13 configured in the OpenWRT Web Interface;
- **Location:** A, B, C or D depending on the locations used for the test;
- **Iteration:** Goes from 1 up to 5 for each unique configuration to give better and more accurate statistics on the captured data.

For example, the ID PUL13C3 represents a test where the black antenna was used, uplink mode of *iperf3*, Wi-Fi channel 13 at the location C and it was the 3rd test done with that specific configuration. Other example is BDL1D4 where the white antenna was used, downlink mode, wi-fi channel 1, location D and 4th test.

This Unique ID coding was important to develop the *csv_maker.sh* script whose objective was to automate a way of going in every single folder and filter every file and write the relevant data to a comma separated value file.

The way this filtering was done, depended on whether it was a downlink test or an uplink test. At first, the script creates the files and gives permission for *tshark* to write in them. After that *tshark* will read the previously captured file with specific filters and capturing specific parameters only. If we are processing a Downlink test, we use *tshark* to read twice the capture of the client side.

The first *read*, will filter the packets that have the ip source of the Router Node and the packets whose radio signal strength in dbm is lower than 0. This is done because when analyzing the captures individually we noticed that the aggregation of some TCP packets resulted in only one of the aggregated packets having the data rates and signal power so we assumed that the value of that packet would be the average of the total packets in that aggregation process. The columns captured for this first read are the data rate, the short GI indicator and the radio signal power in dBm.

For the second *read*, and since the *iperf3* log did not work properly in some of the tests, we decided to calculate the throughput from the wireshark capture, considering every packet with IP source of the Router Node (in a downlink test) and with frame length higher than 1400 bytes (otherwise considered a control packet). The columns captured here are the timestamp relative to the first packet captured and the length of the packets in bytes.

From the capture file of the Router Node, similarly to the first read done with the capture file of the Client Node, we filter the packets with ip source of the Client Node, and the packets whose radio signal strength in dBm is lower than 0. The only column saved was the one with the RSSI values, because there was no interest in knowing the TX rate of control packets that were sent from the Client Node.

The Output of this script, are 3 different CSV files with data from the 48 test folders. Summing up for each test we have 3 comma separated values files being generated:

- *RSSI and TX rate File* - Captured in the side receiving the data. Raspberry Pi if it was an uplink test or the client computer if it was a downlink test;
- *RSSI File* - Captured in the opposite side following the logic presented in the previous item. This file is mainly composed of the control packets and their RSSI value;
- *Throughput File* - Captured in the side receiving the data.

In order to reduce the number of files, every RSSI and data rates information from each of the 5 tests using the same configuration were concatenated into two different files. The throughput CSV files still needed to be processed individually. From the 240 initial folders we reduced those to 48 adapting the Unique ID code based on the initial one designed but now ending in "F" instead of the numbers from 1 to 5.

Listing B.2 presents the developed script to process the raw CSV data that resulted from the *csv_maker.sh* script mentioned previously. The final output of this script are four condensed CSV files with all the information from the 48 tests previously performed. It used three different python libraries: Pandas, an open source data manipulation tool used to organize the data from every file in *dataframes*; NumPy, a requisite to use Pandas that helps working with arrays and other mathematical functions; Counter from Collections, used to obtain the number of occurrences of each TX_Rate value in each file. There are four big final *dataframes* containing the information of every test regarding the metrics that are going to be analyzed.

The Script starts by accessing every file that resulted from the *csv_maker.sh* script already explained. Each time a new file is being processed, it saves some information variables regarding whether the control data packets, the TX_Rate, RSSI and throughput come from the client side or the router side. These variables are later used to organize the data in the final CSVs at the cost of more columns.

The first metric that was analyzed are the RSSI values and the snippet of code that represents this methodology is presented in Figure 4.20. We use pandas library to read the CSV file and convert the information to a Pandas *dataframe*. The *dataframe* may contain (if we are processing the files from the side that received the packets) information regarding the TX_Rate values of the test, so we selected only the relevant column to process. This *dataframe* is then updated using the *dataframe.describe()* method, which calculates statistical data like the number of RSSI values processed, the percentile, the mean and the standard deviation of the data presented in the *dataframe*. At this point, the information regarding the RSSI of this specific test is only missing two identifier columns, and it is then ready to be appended to the *finalRSSI dataframe*. These identifiers are the unique ID of the test that was being processed and an extra identifier that tells us if the RSSI values of this test are from the Client Node or the Router Node. The *finalRSSI dataframe* contains one row per test processed and the columns contain the statistical data and the identifiers.

```

73
74 #--- Processing RSSIandTX file ---#
75
76         df= pd.read_csv('./id_'+fileID+'/'+'fileRSSIandTX, delimiter='\t', names=['TX_RATE', 'SHORTGI', 'RSSI'])
77 #-RSSI-#
78         dfrSSI=df['RSSI']
79         dfrSSI=dfrSSI.to_frame()
80         dfrSSI=dfrSSI.describe()
81         dfrSSI=dfrSSI.rename(index={'count': 'RSSIcount', 'mean': 'RSSImean', 'std': 'RSSIstd', 'min': 'RSSImin',
82                                   '25%': 'RSSI25', '50%': 'RSSI50', '75%': 'RSSI75', 'max': 'RSSImax'})
83         dfrSSI=dfrSSI.transpose()
84         dfrSSI.insert(0, "ID", fileID)
85         dfrSSI.insert(0, "who", WHOorssiandtx)
86         finalRSSI = finalRSSI.append(dfrSSI, ignore_index=True)
87
88 #-END_OF_RSSI-#

```

Figure 4.20: Snippet of RSSI processing.

The second metric that was analyzed are the TX_Rate values, and the snippet of code that represents this methodology are presented in Figure 4.21. We selected the relevant columns from the initial RSSI and TX_Rate file. This columns were the TX_Rate of the packet and a Boolean

variable that was used to give us information whether the packet used a Short Guard Interval (GI) or a Long GI. This guard intervals only refers to the 802.11n protocol and since the Wireless USB adapters used in the tests, supported falling back automatically to 802.11b or 802.11g, some of the captured packets, when under a bad quality link environment, there were processed packets that did not have this boolean variable. Bearing in mind that this additional organization of information implied more *dataframe* or columns regarding the type of TX_Rate value, it was decided to simplify the *dataframe* to a single *dataframe* with every possible value from 802.11b/g/n standards considering a 20 MHz channel. Due to this decision, and checking Table 2.4, there are some cases where there is a collision of values: for example, the MCS Index 7 considering Long GI has the same Data Rate of the MCS Index 6 with Short GI. We then converted the *dataframe* with the selected data to a Python list so we could make use of the method *Counter()* whose function is to keep track of how many times equivalent values are added. This would give us the number of occurrences of each TX_Rate packet so we could organize it in a Histogram. After that, we add the Identifier column and append all the data processed to the finalHIST *dataframe*, which contains all the occurrences of each TX_Rate value for every performed test.

```

90 #-TX-#
91         dfTX= df[['TX_RATE', 'SHORTGI']]
92         dfTX['TX_RATE']=dfTX['TX_RATE'].str.replace(",",".")
93         dfTX['TX_RATE']= dfTX['TX_RATE'].astype(float)
94         dfTX=dfTX['TX_RATE']
95         lst_BIG=dfTX.tolist()
96
97         results=Counter(lst_BIG)
98         finalHIST=finalHIST.append({'ID':fileID, '1':results[1], '2':results[2], '5.5':results[5.5],
99                                     '6':results[6], '6.5':results[6.5], '7.2222':results[7.2222],
100                                    '9':results[9], '11':results[11], '12':results[12],
101                                    '13':results[13], '14.4444':results[14.4444], '18':results[18],
102                                    '19.5':results[19.5], '21.6667':results[21.6667],
103                                    '24':results[24], '26':results[26], '28.8889':results[28.8889],
104                                    '36':results[36], '39':results[39], '43.3333':results[43.3333],
105                                    '48':results[48], '52':results[52], '54':results[54],
106                                    '57.7778':results[57.7778], '58.5':results[58.5],
107                                    '65':results[65], '72.2222':results[72.2222]}, ignore_index=True)
108
109 #-END_OF_TX-#
110 #--- End of Processing RSSIandTX file ---#

```

Figure 4.21: Snippet of TX_Rate processing.

The third and final metric that was analyzed are the Throughput values. Two different CSV files were generated from this part of the script. The first one is a *dataframe* containing the mean and packet count number of the five different tests done for each configuration, together with the total mean and count when all of them were considered. The second one contains statistical data similar to the one obtained for the RSSI metric regarding the total of five tests for each configuration.

The process behind the first output file was to check the timestamp of the first previously filtered packet and the last one, in order to get the duration of the test. After that, we would iterate through every row of the initial CSV to find the total sum of every length of the packet transferred during that same period of time. Dividing the total data by the period of time and multiplying the result would return us the mean throughput in bits/sec of that given test. The total mean throughput

was calculated following the same logic, using the values from the five individually calculated throughputs. All this data is then appended to the *finalTHROUGHPUT dataframe* together with the unique ID of the test.

The process behind the second output file is a little bit more complex since it required a different methodology to obtain the required statistical data. If we were to consider the 5 throughput values from every test the quality and relevance of the statistics would be low since the population number is not enough. The idea to solve this problem was to calculate throughputs based on each second of testing, making a total of 60 samples per test, or 300 per configuration, if every test had exactly 60 seconds of duration. The way the throughput per second was obtained required additional variables that would store the information while the difference of timestamps was below one second. The throughput was then calculated and stored on an auxiliary array that was independent of the Iteration number of the unique identifier. After having every throughput based on each second of testing, we used the *dataframe.describe()* frame to calculate the statistical data. After that, it was inserted to the *dataframe*, the unique identifier of the test that was processed and then appended to the *finalSECTHROUGHPUT dataframe* that contains every statistical data regarding each second of each test.

After having each *dataframe* prepared, we used the method *dataframe.to_csv()* whose job is to export the *dataframe* into a CSV file. This CSV file was then imported into an Excel file, making use of a GUI, where the plots and final conclusion were taken.

4.2.3 Ground Experiments

The ground experiments were done with the objective of finding the best solution out of the hardware we used and the channel to transmit the signal. The graphical data that presented in this subsection was generated through the processed CSV files that resulted from the previously explained Python script.

Every conclusion that is taken is always based on the averages obtained after sorting the tests by certain specifications that will be explained. The RSSI and throughput statistical values considered for the wireless adapter and channel decisions were the means, and for the TX_Rate we considered an weighted average, multiplying each occurrence of a TX_rate value for the respective value, summing them all and dividing by the total number of occurrences.

4.2.3.1 Wireless Adapter decision

To take the decision of choosing what wireless adapter was better for our final solution, we used the average RSSI values and faced the results using each wireless adapter. As a reminder, "Adapter B" stands for the TL-WN722N model and "Adapter P" for the Panda N600.

In Figure 4.22 we have three different plots built from the same data but sorted in different ways to get a better perception of what happened during the ground tests. The RSSI values analyzed here consists on the side that received the data, which means if it is a Downlink test being

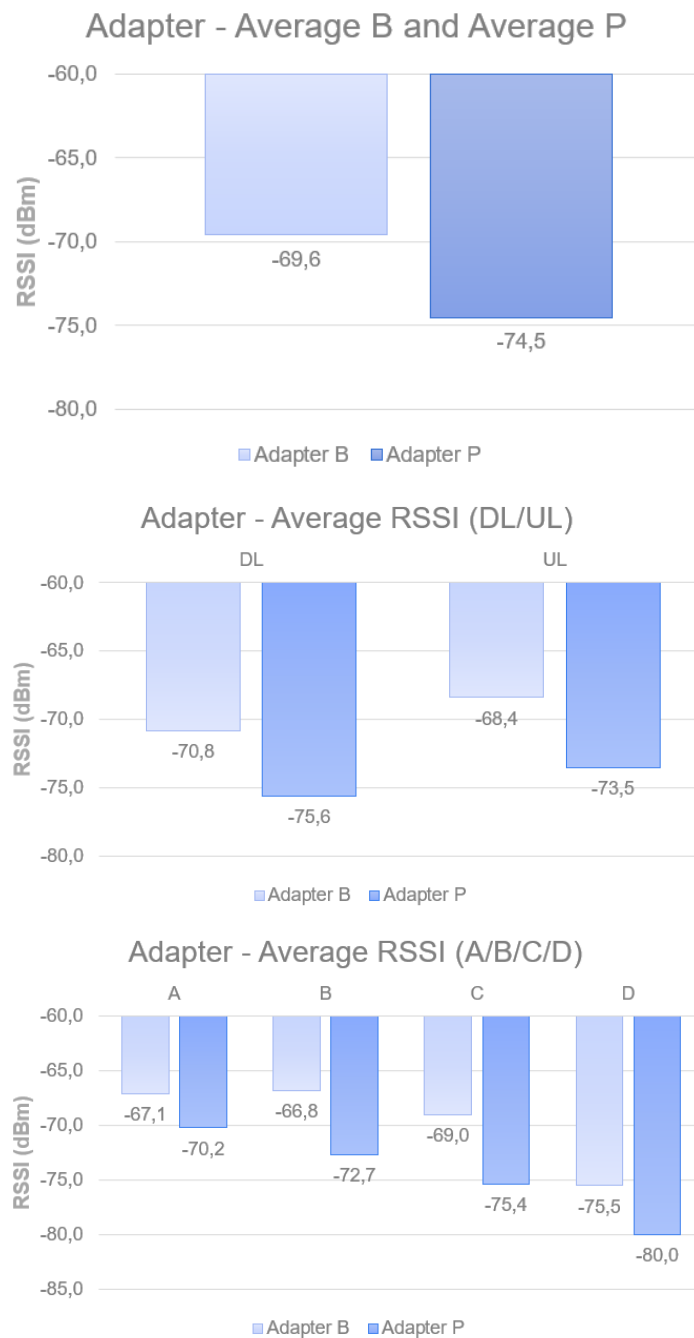


Figure 4.22: RSSI Adapter decision plots.

processed, the RSSI values are from the Client Node, and if it as an Uplink the values are from the Router Node.

The first plot refers to the average value of RSSI measured considering every test done with Adapter B and Adapter P individually. As we can see Adapter B is on average almost 5 dBm better than the Adapter P. In the following plots of the same figure we face Adapter B and P average data sorting it by Downlink and Uplink tests in the first plot and by location A to D in the last one. We

can conclude that no matter the scenario the tests are compared the average values of Adapter B do not differ much from the initial plot, which means that considering the RSSI values, Adapter B is better than Adapter P.

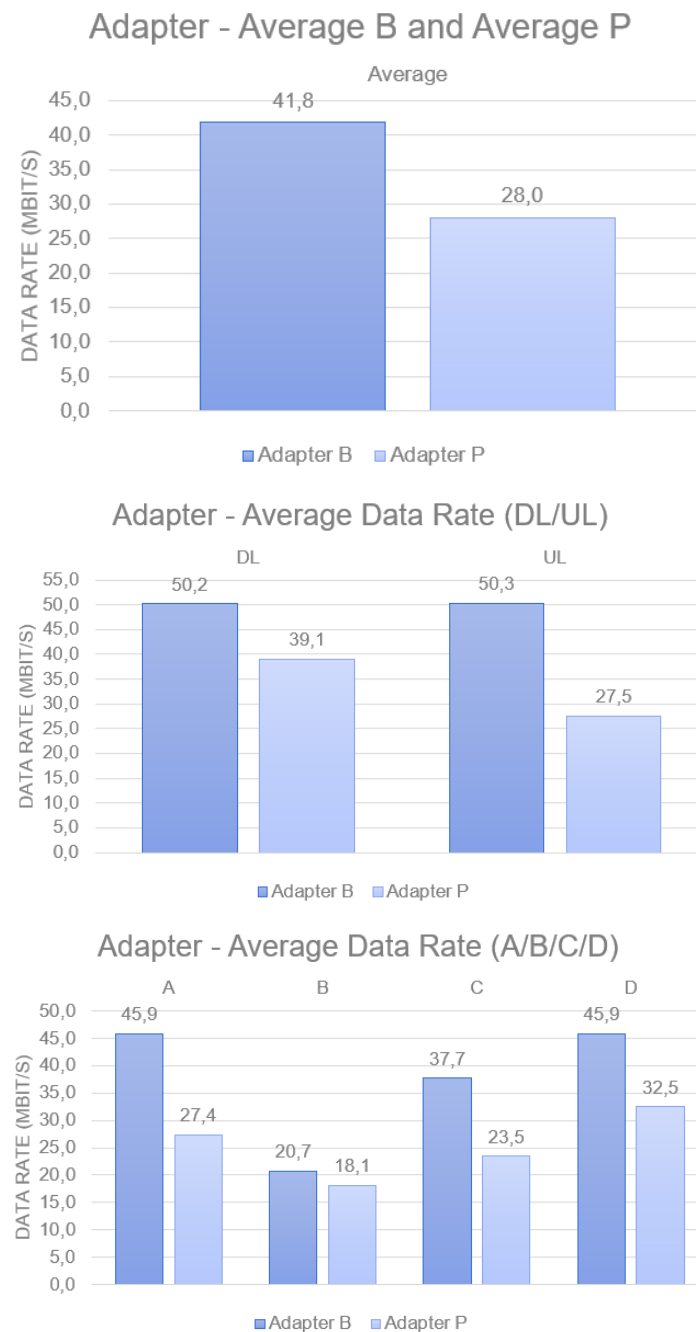


Figure 4.23: TX_Rate Adapter decision plots.

Following the same logic presented with the RSSI values, the TX_Rate plots, presented in Figure 4.23 are also divided into the same three different sortings. Considering the average TX_Rate values of the first plot, we see Adapter B with an average of 41.8 MBit/s and Adapter P with an

average of 28 MBit/s. Considering the Uplink Tests and Downlink tests the Adapter B still shows better results than Adapter P and with the increasing distance of locations A to B this difference remains high (averaging a difference of 12.1 MBit/s between Adapter B and Adapter P) with the lowest difference being noticed in the tests that took place in place B. This means that, considering the TX_Rate values that were observed during the tests, we can conclude that Adapter B is better than Adapter P.

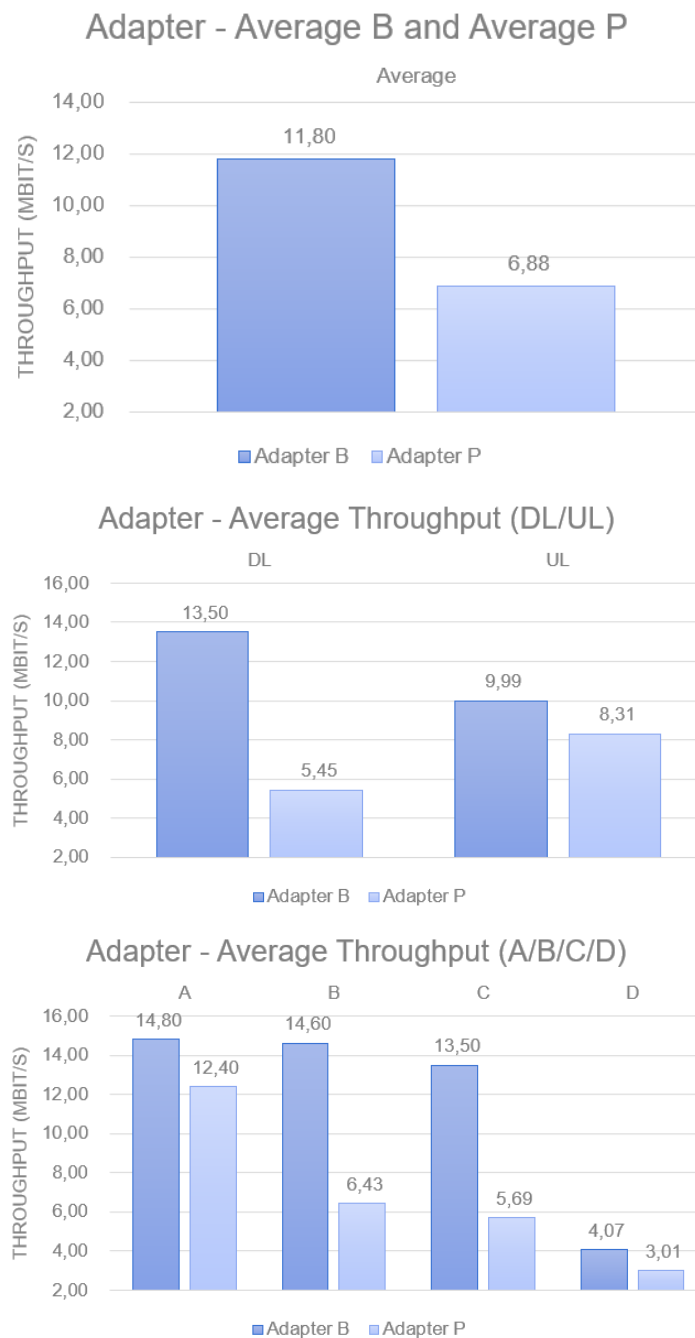


Figure 4.24: Throughput Adapter decision plots.

The last set of plots analyzed to understand whether Adapter B is better than the Adapter P is represented in Figure 4.24. The difference in throughput considering the tests performed with the Adapter B and Adapter P is close to 5 Mbit/s. When looking at the second plot represented we see that the difference between them is higher when considering the Downlink tests. If we look at the plot that sorts the tests by location, we can see that Adapter B performance withstands for longer distances when compared to Adapter P, as we notice an increasing difference between them from Location A to C, but once we look at the tests performed in location D, it is noticeable that both had low throughput values not much different from each other. Taking into consideration the analysis of all these throughput plots it is clear that the Adapter B performance is better overall scenario.

Since every metric analyzed proved that Adapter B was better in every scenario when compared to Adapter P, it was a clear decision to consider Adapter B tests from now on with the objective of finding the best channel.

4.2.3.2 Channel decision

The channel decision was done analyzing the same data containing the average values of RSSI, TX_Rate and Throughputs, however this time, plotting the information sorting it by channel.

Starting with the RSSI values, presented in Figure 4.25, the first plot was chosen to confirm that considering the values from Adapter B tests, we would obtain better results in every channel when compared to the average of Adapter B and Adapter P. We can also observe that every channel has little difference RSSI values between each other. However, for this sorting, Channel 7 shows the best performance of the three considered channels. When considering the graphich that sorts the data by Downlink tests, we observe that Channel 1 has the best RSSI value. Still, it has the lowest value when considering the Uplink average. Considering the tests done in location A, the best average value is for Channel 1, but in locations B and C the best channel is Channel 13. At the longest distance, in location D the best channel is Channel 13.

From this metric we can see that choosing the best channel was not as simple as deciding the adapter for the best solution.

Following the same methodology used previously, the first TX_Rate plot presented in Figure 4.23 was to prove that using the average of the tests with Adapter B we obtained a higher value than the average of the tests considering both adapters. From this plot we can also observe that Channel 13 holds the higher average of the 3 channels, even though the difference for the other 2 plots is low (3.6 MBit/s and 2.9 MBit/s). When considering the Uplink tests, the Channel is the Channel 1 and for Uplink tests it is the Channel 13. When considering the 4 different locations, with the exception of location B where the best channel was Channel 7, the channel 13 is the has higher averages in every scenario.

To reach the conclusion about the best Channel to use, the last metric that is going to be analyzed is the throughput, shown in Figure 4.27. At a first glance, Channel 13 has the best throughput values for every configuration, with the exception the third plot, at location A where

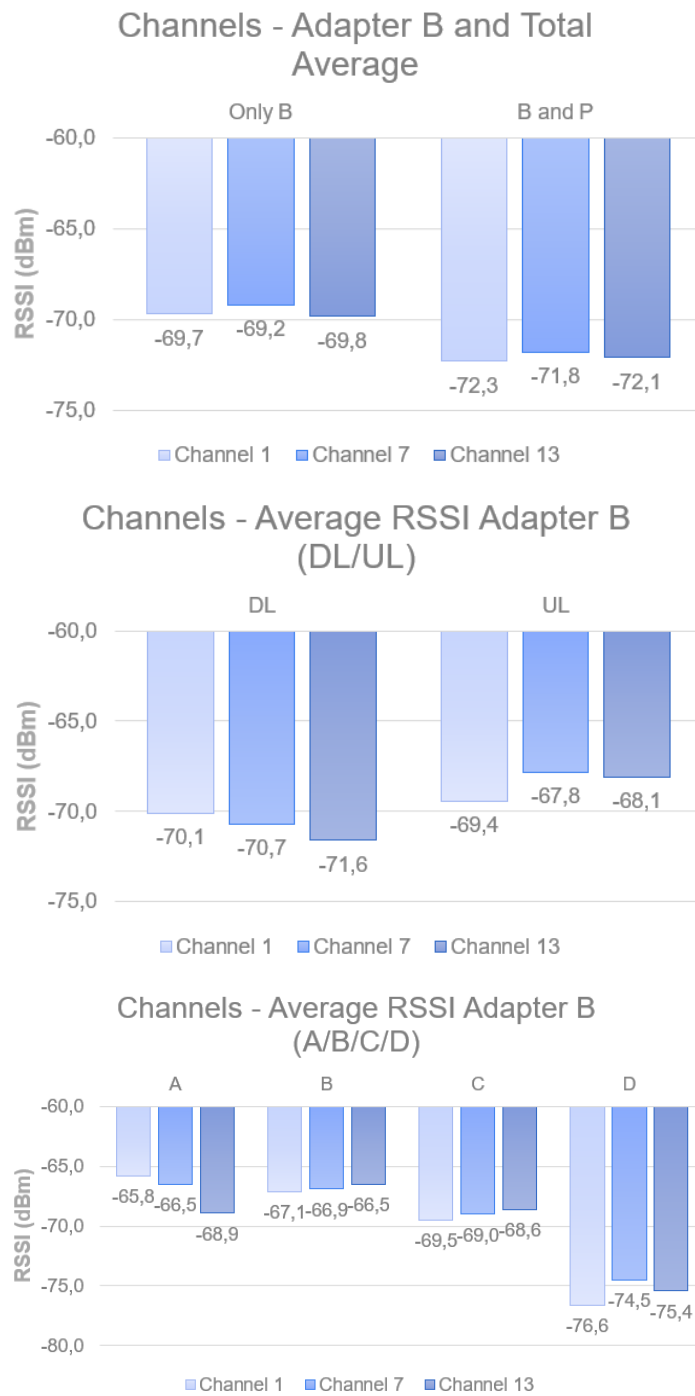


Figure 4.25: RSSI Channel decision plots.

Channel 7 and Channel 1 have higher average throughput. Still, in this metric, we can prove that Adapter B has higher throughput values than the average of Adapter B and P together, and that the Throughput Downlink averages are higher than the Uplink averages.

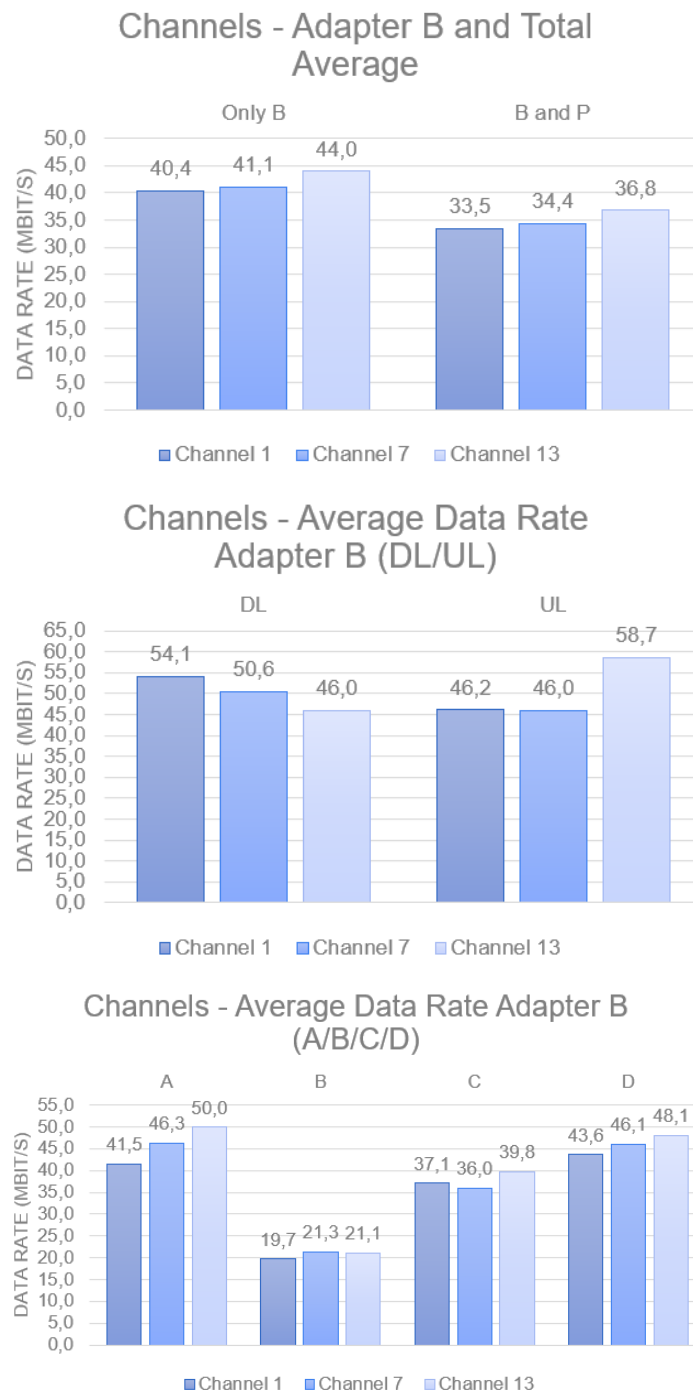


Figure 4.26: TX_Rate Channel decision plots.

Considering every metric analyzed, the best channel is not as clear as it was the best adapter, since there are different channels that are better in different configuration scenarios. However, the channel that was chosen was Channel 13. The total RSSI average difference considering the Adapter B, between Channel 13 and 7 is 0.6 dBm and when considering TX_Rate and Throughput, channel 13 is the channel that better withstands the distance factor which was a decisive factor for

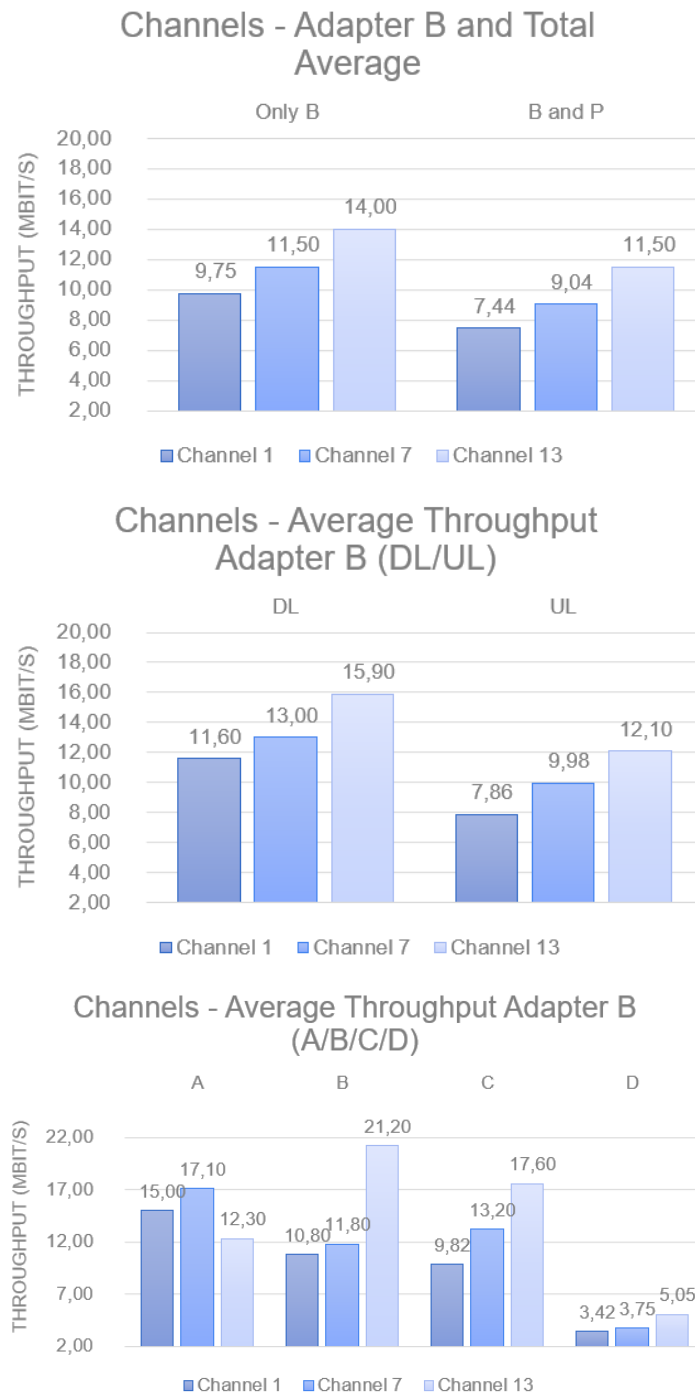


Figure 4.27: Throughput Channel decision plots.

the final conclusion. However, to the best of my knowledge, we can assume that in this scenario, did not impact as much as the Wireless USB adapter.

4.2.4 Discussion

After deciding that the best solution was using the Wireless USB Adapter TL-WN722N configured for Channel 13, we were ready to use this configuration for the flying tests. In the following sections we present the test plan designed for the flying tests followed by a section dedicated to the discussion of what was achieved with our best solution.

4.2.4.1 Flying Tests

The test plan for flying test was to stabilize the height of the UAV from 10 to 10 meters at a maximum height of 100 meters and do 5 test of 60 seconds duration per height in order to achieve relevant statistical data. The captured data would be captured using the scripts prepared previously for the ground tests, and the processing scripts would adjust for the new scenarios accordingly. With these data we would make plots together with the predictions previously calculated in order to understand if our system was well designed so we could calculate the limits of our system and present those limits to the ResponDrone project.

However, a hardware defect was not detected and in flying trials the UAV suffered engine damage that could not be repaired in time. Since there was no time left in the development of the dissertation, we will detail the conclusions we achieved with the best solution, when considering the ground tests that were previously analyzed.

4.2.4.2 Best Solution Discussion

After selecting the tests that were carried out using the TL-WN722N configured for Channel 13, we plotted the distribution of the RSSI data and matched it with two predictions based on two different models: Friis and the Log Distance using a gamma of 3.5 and a reference distance of 10 meters, that seemed reasonable for the environment where the tests were occurred. The obtained Downlink results are presented in Figure 4.28. This tests only considered the RSSI values captured in the client side because its the side that received the data. Figure 4.29 shows the Uplink test results, whose RSSI values were captured in the router side for the previously mentioned reason.

Analysing the results plotted in Figure 4.28 we observe that the Friis predictions for those distances are always distant from the majority of the distribution of RSSI obtained with the tests that were done. The Log distance model on the other hand leads to theoretical values relatively close to the average obtained experimentally. The reason behind the inaccuracy observed in the small distances may be due to the fact that the reference distance (calculated with Friis) chosen was of 10 meters, which is still very close to the first distance where tests were performed, at location A (17.7 m). When observing the Uplink results at Figure 4.29 we can also conclude that the Friis prediction was over optimistic and that even the Log Distance values were far from the Max value registered across all the tests carried in location A. However, as we increase the distance, following the same logic explained for the Downlink tests, we can observe that the Log Distance model using a gamma of 3.5 is very close to the majority of results obtained for those locations. Since the path loss is a logarithmic expression, we can conclude that these values would remain close within each

other for longer distances, making the Long Distance model viable to predict the distance limits using this theoretical path loss model. Regarding the sensitivity of the hardware used throughout this work, we can observe that the minimum values were never below the -100 dBm mark initially predicted, which makes the distance levels shorter than it was predicted in Section 3.5.

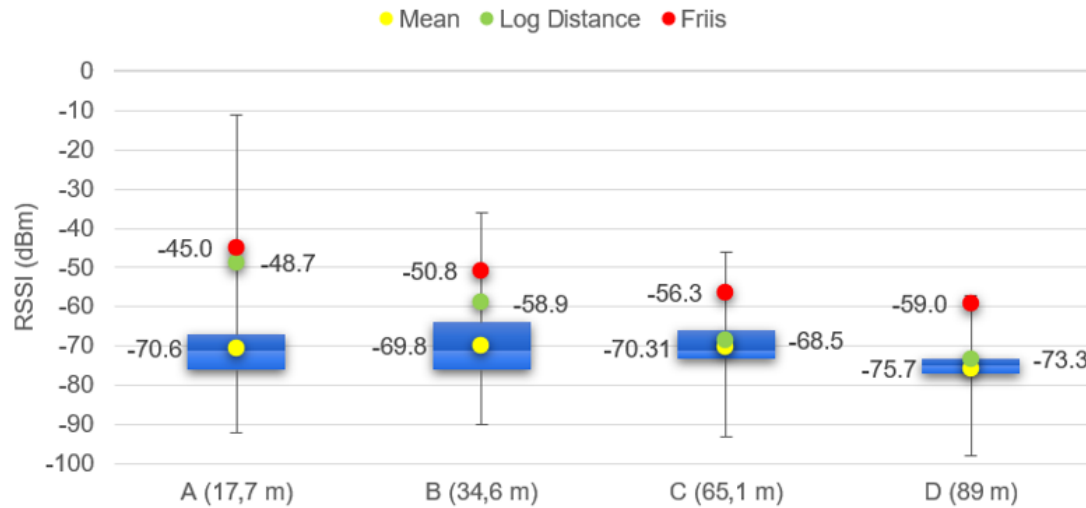


Figure 4.28: Downlink RSSI Results using Best Solution.

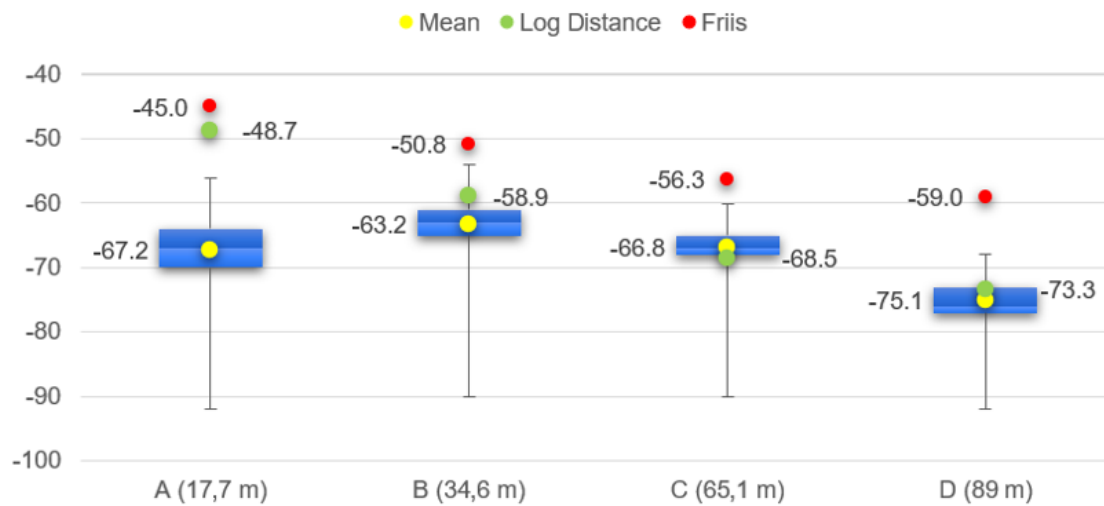


Figure 4.29: Uplink RSSI Results using Best Solution.

Chapter 5

Conclusion and Future Work

With the work that was done we could realise that it is possible to deploy a low cost system capable of providing LTE cellular network coverage. When running the software we noticed that the range does not match what we needed for the project and the hardware used had some problems. Due to the recent situation of the pandemic caused by Covid-19, we were late to find where the sources of the problem were and we did not have enough time to come up with a different solution that could provide us different results. This led to focusing the last part of the work on working with the Wi-Fi solution initially proposed.

This research regarding the first proposed solution had a considerable amount of obstacles that came in the way. Some of them were quick to solve. Others took longer time. The remaining obstacles remain unsolved and are left for future work. A lot of incomplete documentation, led us to use Mailing lists where we met the srsLTE community with whom we discussed results and troubleshooting. Some problems that arose back then are now properly documented so that new users know how to solve them. The feedback from some developers of srsLTE was also important to reach some conclusions regarding the problems that occurred throughout the development of this work and what they managed to achieve with similar hardware to the one we used.

The Wi-Fi solution proved to be a viable option to deploy the system we needed since we could actually test it at considerable ranges while maintaining a certain level of quality in all metrics that were discussed. The system as a whole is cheaper, smaller and easier to power than the LTE solution and still provides enough throughput for what it is designed for.

We did not expect that Adapter P would have worse overall performance than Adapter B since it had 2 antennas and was transmitting at double the power of the latter. The decision of the channel was more complex since the average values had little to no difference between the three that were used. It is important to mention that this choice will always be dependent of the physical location where the network operates, thus the drawn conclusions were based on the experimental results that were obtained. Comparing the RSSI values with the predictions previously calculated allowed us to understand that the accuracy of the path loss model increases with the distance. In the future we should use different hardware with higher TX power to see how the system would

improve and at what cost, in order to reach an even better solution than the one we used in this work.

In the future we should test different hardware and try to compare different metrics of the SDR low cost solutions with the commercial solutions designed to deploy LTE network. With the hardware we had we noticed instability in the tests even with a strong link. We should also experiment with different antennas, designed specifically for the scenario. This work contributed to decide that the Wi-Fi solution was better at the time for the scope of the European ResponDrone project but there is still much to learn in this promising field.

Appendix A

Images

Here are shown images of the setups that were used and other images that were taken during the experiments.

A.1 Example of a Wi-Fi Implementation Setup

Figure [A.1](#) shows one of the setup that was used during the implementation.

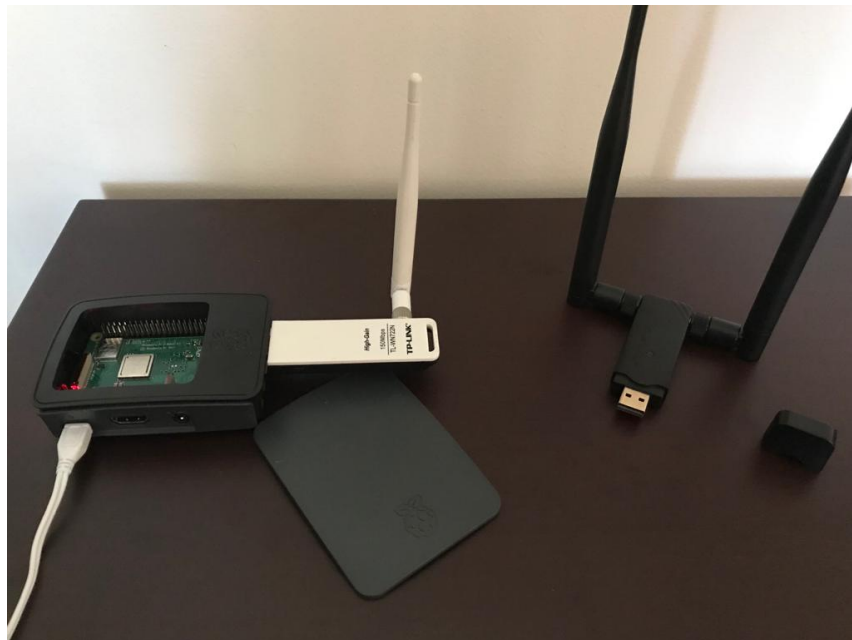


Figure A.1: Example of a Wi-Fi implementation setup.

A.2 Example of a LTE Implementation Setup

Figure A.2 shows one possible setup that was used in the LTE implementation, with both NUCs, the UE and the RF Front End with the respective amplifiers.

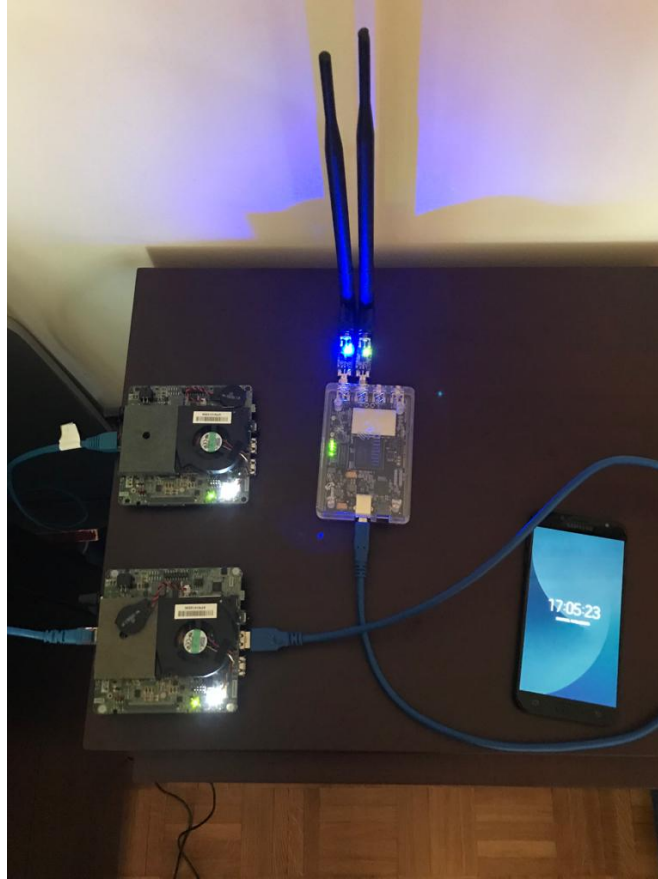


Figure A.2: Example of a LTE implementation setup.

A.3 Experiments Locations

Here are presented some pictures that were taken during the Experiments days. Figure A.3 shows two different perspectives of the Router Node location in one of the INESC TEC's window. Figure A.4, Figure A.5, Figure A.6 and Figure A.5 show the locations and respective perspective to the Router Node position from the location where the experiments were performed.



Figure A.3: Router Node window positioning.



Figure A.4: Location A Perspective.



Figure A.5: Location B Perspective.



Figure A.6: Location C Perspective.



Figure A.7: Location D Perspective.

Appendix B

Scripts Logs and Codes

Appendix B addresses scripts codes and logs that were used for the development of this work.

B.1 OAI Script

Listing B.1 presents the script used to run the OAI eNB software.

Listing B.1: runOAIScript.sh

```
#!/bin/bash

GUI=$1
WORKING_DIR=/home/bladerf1/OAI_BLADERF/openairinterface5g
/enb-band7-5mhz.conf
cd /home/bladerf1/OAI_BLADERF/openairinterface5g
source oaienv
vmstat -wtn 1 200 > sysENB.log &
mpstat -P ALL 1 > cpuENB.log &
if [ $1 == 'gui' ]
then
echo 'GUI VERSION'
sudo -E ~/OAI_BLADERF/openairinterface5g/cmake_targets
/lte_build_oai/build/lte -softmodem -O $WORKING_DIR
--T_stdout 0 | ts | tee logENB.log
else
sudo -E ~/OAI_BLADERF/openairinterface5g/cmake_targets
/lte_build_oai/build/lte -softmodem -O $WORKING_DIR
| ts | tee logENB.log
fi
```

B.2 Data Processing Script

Listing B.2 presents the script used to analyze the captured raw CVS files for process.

Listing B.2: Processing of raw csv files script.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Aug  5 11:49:06 2020

@author: rubenq
"""

#MASTER PYTHON
import pandas as pd
import numpy as np
from collections import Counter

ant_type = ['B', 'P']
data_ori = ['DL', 'UL']
channel = ['1', '7', '13']
location = ['A', 'B', 'C', 'D']

placeholder=[]
fileID=[]

HEADERrssi=['ID', 'who', 'RSSIcount', 'RSSImean', 'RSSIstd', 'RSSImin',
'RSSI25', 'RSSI50', 'RSSI75', 'RSSImax']
HEADERrssiCLIENT=['ID', 'RSSIclientCOUNT', 'RSSIclientMEAN',
'RSSIclientSTD', 'RSSIclientMIN', 'RSSIclient25', 'RSSIclient50',
'RSSIclient75', 'RSSIclientMAX']
HEADERrssiPI=['ID', 'RSSIpiCOUNT', 'RSSIpiMEAN', 'RSSIpiSTD',
'RSSIpiMIN', 'RSSIpi25', 'RSSIpi50', 'RSSIpi75', 'RSSIpiMAX']
HEADERthroughput=['ID', 'countALL', 'meanALL', 'count1', 'mean1',
'count2', 'mean2', 'count3', 'mean3', 'count4', 'mean4', 'count5', 'mean5']
HEADERhist=['ID', 'HISTsgi', 'HISTlgi', 'HISToth']
HEADERbigHIST=['ID', 'HISTlist', 'HISTbin']
HEADER_HIST=['ID', '1', '2', '5.5', '6', '6.5', '7.2222', '9', '11', '12',
'13', '14.4444', '18', '19.5', '21.6667', '24', '26', '28.8889', '36', '39',
'43.3333', '48', '52', '54', '57.7778', '58.5', '65', '72.2222']
HEADERsecthroughput=['ID', 'thCOUNT', 'thMEAN', 'thSTD', 'thMIN', 'th25',
'th50', 'th75', 'thMAX']

BIGbin=[1,2,5.5,6,6.5,7.2222,9,11,12,13,14.4444,18,19.5,21.6667,
24,26,28.8889,36,39,43.3333,48,52,54,57.7778,58.5,65,72.2222]
bin802_11nSGI=[7.2222,14.4444,21.6667,28.8889,43.3333,57.7778,
65,72.2222]
bin802_11nLGI=[6.5,13,19.5,26,39,52,58.5,65]

```

```

bin802_11bg=[1,2,5.5,6,9,11,12,18,24,36,48,54]

finalRSSI = pd.DataFrame(placeholder , columns=HEADERrssi)
finalTHROUGHPUT = pd.DataFrame(placeholder ,
columns=HEADERthroughput)
finalHIST = pd.DataFrame(placeholder , columns=HEADER_HIST)
finalSECTHROUGHPUT=pd.DataFrame(placeholder ,
columns=HEADERsecthroughput)

for at in ant_type :
    for do in data_ori :
        if do == 'DL':
            fileRSSI="RSSI_pi.csv"
            fileRSSIandTX="RSSIandTX_client.csv"
            fileTH="_client.csv"
            WHOrssiandtx="Client"
            WHOrssi="Router"
        elif do == 'UL':
            fileRSSI="RSSI_client.csv"
            fileRSSIandTX="RSSIandTX_pi.csv"
            fileTH="_pi.csv"
            WHOrssiandtx="Router"
            WHOrssi="Client"
        for ch in channel :
            for lo in location :
                fileID = at + do + ch + lo + 'F'
                print('Processing' + fileID)
#—— Processing RSSI file ——#
                fdata = pd.read_csv('./id_'+fileID+'/' +fileRSSI ,
                delimiter='\t' , names=['RSSI'])
                fdata = fdata.describe()
                fdata = fdata.rename(index={'count': 'RSSIcount' ,
                'mean': 'RSSImean' , 'std': 'RSSIstd' , 'min': 'RSSImin' ,
                '25%': 'RSSI25' , '50%': 'RSSI50' , '75%': 'RSSI75' ,
                'max': 'RSSImax' })
                fdata = fdata.transpose()
                fdata.insert(0,"ID",fileID)
                fdata.insert(1,"who",WHOrssi)
                finalRSSI = finalRSSI.append(fdata ,
                ignore_index=True)
#—— End of Processing RSSI file ——#
#—— Processing RSSIandTX file ——#
                df= pd.read_csv('./id_'+fileID+'/' +fileRSSIandTX ,
                delimiter='\t' , names=['TX_RATE' , 'SHORTGI' , 'RSSI' ])
#-RSSI-#

```

```

dfRSSI=df[ 'RSSI' ]
dfRSSI=dfRSSI.to_frame()
dfRSSI=dfRSSI.describe()
dfRSSI=dfRSSI.rename(index={ 'count': 'RSSIcount',
'mean': 'RSSImean', 'std': 'RSSIstd', 'min': 'RSSImin',
'25%': 'RSSI25', '50%': 'RSSI50', '75%': 'RSSI75',
'max': 'RSSImax' })
dfRSSI=dfRSSI.transpose()
dfRSSI.insert(0,"ID",fileID)
dfRSSI.insert(0,"who",WHOrssiandtx)
finalRSSI = finalRSSI.append(dfRSSI,
ignore_index=True)

#-END_OF_RSSI-#
#-TX-#

dfTX= df[ [ 'TX_RATE', 'SHORTGI' ] ]
dfTX[ 'TX_RATE' ]=dfTX[ 'TX_RATE' ].str.replace(",",".")
dfTX[ 'TX_RATE' ]= dfTX[ 'TX_RATE' ].astype( float )
dfTX=dfTX[ 'TX_RATE' ]
lst_BIG=dfTX.tolist()
results=Counter(lst_BIG)
finalHIST=finalHIST.append({ 'ID': fileID ,
'1': results[1], '2': results[2], '5.5': results[5.5],
'6': results[6], '6.5': results[6.5],
'7.2222': results[7.2222], '9': results[9],
'11': results[11], '12': results[12],
'13': results[13], '14.4444': results[14.4444],
'18': results[18], '19.5': results[19.5],
'21.6667': results[21.6667], '24': results[24],
'26': results[26], '28.8889': results[28.8889],
'36': results[36], '39': results[39],
'43.3333': results[43.3333], '48': results[48],
'52': results[52], '54': results[54],
'57.7778': results[57.7778], '58.5': results[58.5],
'65': results[65], '72.2222': results[72.2222] },
ignore_index=True)

#-END_OF_TX-#
#—— End of Processing RSSIandTX file ——#
#—— Processing THROUGHPUT file ——#

countV= np.zeros(5)
throughputV=np.zeros(5)
totalCOUNT=0
totalTHROUGHPUT=0
DATAperSEC=[]
for x in range(5):
    nfile=str(x+1)

```

```

df= pd.read_csv('./id_'+ fileID +
'/THROUGHPUT'+ nfile + fileTH ,
delimiter='\t' , names=[ 'TIME' , 'LENGTH' ])
firstTIME=df.head(1)
lastTIME=df.tail(1)
totalTIME=lastTIME.iloc[0].loc[ 'TIME' ]
-firstTIME.iloc[0].loc[ 'TIME' ]
count=lastTIME.index[0]
totalDATA=0
secDATA=0
currentTIME=firstTIME.iloc[0].loc[ 'TIME' ]
for datarow in df.iterrows():
    totalDATA=totalDATA + datarow[1].LENGTH
    secDATA=secDATA+datarow[1].LENGTH
    if (datarow[1].TIME-currentTIME >= 1):
        throughputsec=secDATA
        /( datarow[1].TIME-currentTIME)*8
        currentTIME=datarow[1].TIME
        DATAperSEC.append( throughputsec )
        secDATA=0
#THROUGHPUT in BITS/SEC
    throughput=(totalDATA/totalTIME)*8
    countV[x]=count
    throughputV[x]=throughput
for x in range(5):
    totalCOUNT = totalCOUNT + countV[x]
    totalTHROUGHPUT = totalTHROUGHPUT +
    throughputV[x]
finalSECFRAME=pd.DataFrame([DATAperSEC])
finalSTATISTIC=finalSECFRAME.transpose()
finalSTATISTIC=finalSTATISTIC.describe()
finalSTATISTIC=finalSTATISTIC.rename(
index={ 'count': 'thCOUNT' , 'mean': 'thMEAN' ,
'std': 'thSTD' , 'min': 'thMIN' , '25%': 'th25' ,
'50%': 'th50' , '75%': 'th75' , 'max': 'thMAX' })
finalSTATISTIC=finalSTATISTIC.transpose()
finalSTATISTIC.insert(0,"ID",fileID)
totalTHROUGHPUT=totalTHROUGHPUT/5
finalFRAME=[[ fileID ], [totalCOUNT],
[totalTHROUGHPUT],[countV[0]], [throughputV[0]],
[countV[1]], [throughputV[1]],
[countV[2]], [throughputV[2]],
[countV[3]], [throughputV[3]],
[countV[4]], [throughputV[4]]]
finalFRAME=pd.DataFrame(finalFRAME)

```

```

finalFRAME=finalFRAME.rename(index={0:'ID',
    1:'countALL',2:'meanALL',3:'count1',4:'mean1',
5:'count2',6:'mean2',7:'count3',8:'mean3',
9:'count4',10:'mean4',11:'count5',12:'mean5'})
finalFRAME=finalFRAME.transpose()
finalTHROUGHPUT = finalTHROUGHPUT.append(finalFRAME,
ignore_index=True)
finalSECTHROUGHPUT= finalSECTHROUGHPUT.append(
finalSTATISTIC, ignore_index=True)
#—— End of Processing THROUGHPUT file ——#
finalRSSI.to_csv('./RSSI_everything.csv')
finalHIST.to_csv('./HIST_everything.csv')
finalSECTHROUGHPUT.to_csv('./secTHROUGHPUT_everything.csv')
finalTHROUGHPUT.to_csv('./THROUGHPUT_everything.csv')

```

B.3 OAI-RAN Configuration Script

Listing B.3 presents the file used to configure the OAI-RAN (eNB) software.

Listing B.3: OAI-RAN Config File.

```

Active_eNBs = ( "eNB-Eurecom-LTEBox" );
# Asn1_verbosity, choice in: none, info, annoying
Asn1_verbosity = "none";
eNBs =
(
{
# real_time choice in {hard, rt-preempt, no}
real_time      = "no";
////////// Identification parameters:
eNB_ID         = 0xe00;
cell_type      = "CELL_MACRO_ENB";
eNB_name       = "eNB-Eurecom-LTEBox";
// Tracking area code, 0x0000 and 0xfffe are reserved values
tracking_area_code = 1;
plmn_list = (
{ mcc = 208; mnc = 92; mnc_length = 2; }
);
tr_s_preference = "local_mac"

////////// Physical parameters:

component_carriers = (
{
node_function = "eNodeB_3GPP";
node_timing   = "synch_to_ext_device"; node_synch_ref
= 0;
nb_antenna_ports = 1;

```

```

        ue_TransmissionMode          = 1;
            frame_type                  = "FDD";
        tdd_config                     = 3;
        tdd_config_s                   = 0;
            prefix_type                 = "NORMAL";
            // eutra_band                = 7 ;
        // eutra_band                   = 3;
            eutra_band                  = 20;
            // downlink_frequency         = 2655000000L;
        // downlink_frequency           = 1842500000L;
        downlink_frequency              = 806000000L;
            // uplink_frequency_offset    = -120000000;
            // uplink_frequency_offset    = -95000000;
        uplink_frequency_offset         = 41000000; Nid_cell
    = 0;
    N_RB_DL                             = 25;
    Nid_cell_mbsfn                      = 0;          nb_antennas_tx
    = 1;
    nb_antennas_rx                     = 1;
    prach_root                         = 0;
    tx_gain                            = 90;
    rx_gain                            = 115;
    pbch_repetition                    = "FALSE";
    prach_config_index                 = 0;
    prach_high_speed                   = "DISABLE";
        prach_zero_correlation          = 1;
        prach_freq_offset               = 2; pucch_delta_shift
    = 1;
    pucch_nRB_CQI                     = 0;
    pucch_nCS_AN                      = 0;
    pucch_n1_AN                       = 0;
    pdsch_referenceSignalPower         = -29;
        pdsch_p_b                      = 0;
        pusch_n_SB                     = 1;
    pusch_enable64QAM                 = "DISABLE";
        pusch_hoppingMode               = "interSubFrame";
    pusch_hoppingOffset               = 0;
    pusch_groupHoppingEnabled          = "ENABLE";
    pusch_groupAssignment              = 0;
    pusch_sequenceHoppingEnabled       = "DISABLE";
    pusch_nDMRS1                      = 1;
    phich_duration                    = "NORMAL";
    phich_resource                     = "ONESIXTH";
    srs_enable                         = "DISABLE";
/*   srs_BandwidthConfig              =;
        srs_SubframeConfig             =;
        srs_ackNackST                  =;
        srs_MaxUpPts                   =;*/

```

```

pusch_p0_Nominal                = -96;
pusch_alpha                     = "AL1";
pucch_p0_Nominal                = -108;
msg3_delta_Preamble            = 6;
pucch_deltaF_Format1           = "deltaF2";
pucch_deltaF_Format1b          = "deltaF3";
pucch_deltaF_Format2           = "deltaF0";
pucch_deltaF_Format2a          = "deltaF0";
pucch_deltaF_Format2b          = "deltaF0";

rach_numberOfRA_Preambles      = 64;
rach_preamblesGroupAConfig     = "DISABLE";

/*
                                rach_sizeOfRA_PreamblesGroupA      = ;
                                rach_messageSizeGroupA              = ;
                                rach_messagePowerOffsetGroupB       = ;

*/

                                rach_powerRampingStep              = 4;
                                rach_preambleInitialReceivedTargetPower = -108;
                                rach_preambleTransMax               = 10;
                                rach_raResponseWindowSize           = 10;
                                rach_macContentionResolutionTimer    = 48;
                                rach_maxHARQ_Msg3Tx                 = 4;

                                pcch_default_PagingCycle            = 128;
                                pcch_nB                             = "oneT";
                                bcch_modificationPeriodCoeff        = 2;
                                ue_TimersAndConstants_t300           = 1000;
                                ue_TimersAndConstants_t301           = 1000;
                                ue_TimersAndConstants_t310           = 1000;
                                ue_TimersAndConstants_t311           = 10000;
                                ue_TimersAndConstants_n310           = 20;
                                ue_TimersAndConstants_n311           = 1;

// Parameters for SIB18
rxPool_sc_CP_Len                = "normal";
rxPool_sc_Period                 = "sf40";
rxPool_data_CP_Len              = "normal";
rxPool_ResourceConfig_prb_Num    = 20;
rxPool_ResourceConfig_prb_Start  = 5;
rxPool_ResourceConfig_prb_End    = 44;
rxPool_ResourceConfig_offsetIndicator_present = "prSmall";
rxPool_ResourceConfig_offsetIndicator_choice = 0;
rxPool_ResourceConfig_subframeBitmap_present = "prBs40";
rxPool_ResourceConfig_subframeBitmap_choice_bs_buf = "00000000000000000000";
rxPool_ResourceConfig_subframeBitmap_choice_bs_size = 5;
rxPool_ResourceConfig_subframeBitmap_choice_bs_bits_unused = 0;
/*
rxPool_dataHoppingConfig_hoppingParameter = 0;
rxPool_dataHoppingConfig_numSubbands      = "ns1";

```

```

    rxPool_dataHoppingConfig_rbOffset = 0;
    rxPool_commTxResourceUC-ReqAllowed = "TRUE";
*/
// Parameters for SIB19
discRxPool_cp_Len = "normal"
discRxPool_discPeriod = "rf32"
discRxPool_numRetx = 1;
discRxPool_numRepetition = 2;
discRxPool_ResourceConfig_prb_Num = 5;
discRxPool_ResourceConfig_prb_Start = 3;
discRxPool_ResourceConfig_prb_End = 21;
discRxPool_ResourceConfig_offsetIndicator_present = "prSmall";
discRxPool_ResourceConfig_offsetIndicator_choice = 0;
discRxPool_ResourceConfig_subframeBitmap_present = "prBs40";
discRxPool_ResourceConfig_subframeBitmap_choice_bs_buf = "f0ffffffff";
discRxPool_ResourceConfig_subframeBitmap_choice_bs_size = 5;
discRxPool_ResourceConfig_subframeBitmap_choice_bs_bits_unused = 0;

    }
);

srb1_parameters :
{
# timer_poll_retransmit = (ms) [5, 10, 15, 20,... 250, 300, 350, ... 500]
    timer_poll_retransmit = 80;

# timer_reordering = (ms) [0,5, ... 100, 110, 120, ... ,200]
    timer_reordering = 35;

# timer_reordering = (ms) [0,5, ... 250, 300, 350, ... ,500]
    timer_status_prohibit = 0;

# poll_pdu = [4, 8, 16, 32 , 64, 128, 256, infinity(>10000)]
    poll_pdu = 4;

# poll_byte = (kB) [25,50,75,100,125,250,375,500,750,1000,1250,1500,2000
    ,3000,infinity(>10000)]
    poll_byte = 99999;

# max_retx_threshold = [1, 2, 3, 4 , 6, 8, 16, 32]
    max_retx_threshold = 4;
}

# ----- SCTP definitions
SCTP :
{
# Number of streams to use in input/output
SCTP_INSTREAMS = 2;
SCTP_OUTSTREAMS = 2;

```

```

};

////////// MME parameters :
mme_ip_address      = ( { ipv4      = "192.168.1.91";
                          ipv6      = "192:168:30::17";
                          active    = "yes";
                          preference = "ipv4";
                        }
                      );

enable_measurement_reports = "no";

///X2
enable_x2 = "no";
t_reloc_prep      = 1000;      /* unit: millisecond */
tx2_reloc_overall = 2000;      /* unit: millisecond */

NETWORK_INTERFACES :
{
    ENB_INTERFACE_NAME_FOR_S1_MME          = "eth0";
    ENB_IPV4_ADDRESS_FOR_S1_MME            = "192.168.1.87";

    ENB_INTERFACE_NAME_FOR_S1U              = "eth0";
    ENB_IPV4_ADDRESS_FOR_S1U                = "192.168.1.87";
    ENB_PORT_FOR_S1U                        = 2152; # Spec 2152

    ENB_IPV4_ADDRESS_FOR_X2C                = "127.0.0.1";
    ENB_PORT_FOR_X2C                        = 36422; # Spec 36422
};

log_config :
{
    global_log_level          ="info";
    global_log_verbosity      ="high";
    hw_log_level              ="info";
    hw_log_verbosity          ="medium";
    phy_log_level             ="info";
    phy_log_verbosity         ="medium";
    mac_log_level             ="info";
    mac_log_verbosity         ="high";
    rlc_log_level             ="debug";
    rlc_log_verbosity         ="high";
    pdcp_log_level            ="info";
    pdcp_log_verbosity        ="high";
    rrc_log_level             ="info";
    rrc_log_verbosity         ="medium";
};
}

```

```

);

MACRLCs = (
    {
        num_cc = 1;
        tr_s_preference = "local_L1";
        tr_n_preference = "local_RRC";
        phy_test_mode = 0;
        puSch10xSnr      = 210;
        puCch10xSnr      = 210;
    }
);

THREAD_STRUCT = (
    {
        parallel_config = "PARALLEL_RU_L1_TRX_SPLITaaaaa";
        worker_config = "ENABLE";
    }
);

L1s = (
    {
        num_cc = 1;
        tr_n_preference = "local_mac";
    }
);

RUs = (
    {
        local_rf      = "yes"
        nb_tx          = 1
        nb_rx          = 1
        att_tx         = 73
        att_rx         = 0;
        bands          = [7];
        max_pdschReferenceSignalPower = -28;
        max_rxgain      = 96;
        eNB_instances  = [0];
    }
);

log_config :
{
    global_log_level      = "info";
    global_log_verbosity  = "high";
    hw_log_level          = "info";
    hw_log_verbosity      = "medium";
    phy_log_level         = "info";

```

```

phy_log_verbosity          ="medium";
mac_log_level              ="info";
mac_log_verbosity          ="high";
rlc_log_level              ="info";
rlc_log_verbosity          ="high";
pdcp_log_level             ="info";
pdcp_log_verbosity         ="high";
rrc_log_level              ="info";
rrc_log_verbosity          ="medium";
};

```

B.4 Log of Network Cell Info Lite

Figure B.1 presents one example Log captured by the UE.

Network Cell Info, support@m2catalyst.com

Logs follow the below format (Lat, Lon are the Google Maps coordinates at the Time instant of the logged measurement):

For network types: GSM, GPRS, EDGE:

Network Type | MCC+MNC | LAC | CID | RSSI [dBm] | Lat, Lon | Time

For network types: UMTS, HSDPA, HSUPA, HSPA, HSPA+:

Network Type | MCC+MNC | LAC | UCID | PSC | RSSI [dBm] | Lat, Lon | Time

For network types: LTE:

Network Type | MCC+MNC | TAC | ECI | PCI | RSRP [dBm] | Lat, Lon | Time

For network types: CDMA, EVDO:

Network Type | MCC+MNC | NID | BID | SID | RSSI [dBm] | Lat, Lon | Time

```

LTE | 20892 | 7 | 105217 | 1 | -79 | 41.3118359, -7.7311254 | 12:48:26
LTE | 20892 | 7 | 105217 | 1 | -75 | 41.3118341, -7.7311253 | 12:48:52
LTE | 20892 | 7 | 105217 | 1 | -72 | 41.3118414, -7.7311289 | 12:49:08
LTE | 20892 | 7 | 105217 | 1 | -76 | 41.3118359, -7.7311254 | 12:49:28
LTE | 20892 | 7 | 105217 | 1 | -76 | 41.3118770, -7.7311752 | 12:49:48
LTE | 20892 | 7 | 105217 | 1 | -72 | 41.3118423, -7.7311231 | 12:50:08
LTE | 20892 | 7 | 105217 | 1 | -77 | 41.3118601, -7.7311509 | 12:50:28
LTE | 20892 | 7 | 105217 | 1 | -74 | 41.3118775, -7.7311784 | 12:50:48
LTE | 20892 | 7 | 105217 | 1 | -74 | 41.3118344, -7.7311214 | 12:51:10
LTE | 20892 | 7 | 105217 | 1 | -83 | 41.3118404, -7.7311309 | 12:51:30

```

Figure B.1: Example Log of Network Cell Info Lite.

B.5 srsEPC Report of UE Connection

Figure B.2 presents the report by the srsEPC when a new UE connects to the network.

```

Sending SI Setup Response
Initial UE message: LIBLTE MME MSG TYPE ATTACH_REQUEST
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0x439c2997
Attach request -- eNB-UE S1AP Id: 1
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 11110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 1
PDN Connectivity Request -- ESM Information Transfer requested: false
UL NAS: Received Identity Response
ID Response -- IMSI: 208920123456780
Downlink NAS: Sent Authentication Request
UL NAS: Received Authentication Response
Authentication Response -- IMSI 208920123456780
UE Authentication Accepted.
Generating KeNB with UL NAS COUNT: 0
Downlink NAS: Sending NAS Security Mode Command.
UL NAS: Received Security Mode Complete
Security Mode Command Complete -- IMSI: 208920123456780
Getting subscription information -- QCI 7
Sending Create Session Request.
Creating Session Response -- IMSI: 208920123456780
Creating Session Response -- MME control TEID: 1
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_CREATE_SESSION_REQUEST
SPGW: Allocated Ctrl TEID 1
SPGW: Allocated User TEID 1
SPGW: Allocate UE IP 172.16.0.2
Received Create Session Response
Create Session Response -- SPGW control TEID 1
Create Session Response -- SPGW S1-U Address: 192.168.1.85
SPGW Allocated IP 172.16.0.2 to IMSI 208920123456780
Adding attach accept to Initial Context Setup Request
Initial Context Setup Request -- eNB UE S1AP Id 1, MME UE S1AP Id 1
Initial Context Setup Request -- E-RAB id 5
Initial Context Setup Request -- S1-U TEID 0x1. IP 192.168.1.85
Initial Context Setup Request -- S1-U TEID 0x1. IP 192.168.1.85
Initial Context Setup Request -- QCI 7
Received Initial Context Setup Response
E-RAB Context Setup. E-RAB id 5
E-RAB Context -- eNB TEID 0x460003; eNB GTP-U Address 192.168.1.87
UL NAS: Received Attach Complete
Unpacked Attached Complete Message. IMSI 208920123456780
Unpacked Activate Default EPS Bearer message. EPS Bearer id 5
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_MODIFY_BEARER_REQUEST
Sending EMM Information
Received UE Context Release Request. MME-UE S1AP Id 1
There are active E-RABs, send release access bearers request
Received GTP-C PDU. Message type: GTPC_MSG_TYPE_RELEASE_ACCESS_BEARERS_REQUEST
UE is ECM IDLE.
Received UE Context Release Complete. MME-UE S1AP Id 1
UE is not ECM connected. No need to release S1-U. MME UE S1AP Id 1
UE Context Release Completed.

```

Figure B.2: srsEPC report when new UE connects.

B.6 Pullscript

Figure B.3 shows the script that was used to collect and centralize the data of the Ground Experiments.

```
#!/bin/bash

#SCRIPT DESIGNED TO GATHER THE CAPTURE RESULTS AND CONFIGURATIONS
#FROM NUCs IN THE NETWORK

IPNUC1=192.168.1.87
HOSTNAME1=bladerf1
IPNUC2=192.168.1.85
HOSTNAME2=bladerf2
IPVM3=192.168.1.91
HOSTNAME3=oaiepc

echo "Capture ID_?"
read id
echo " oai or srs ?"
read software
cd ~/captures && mkdir id_$id && cd id_$id
pwd=$(pwd)
echo "ATUALIZAR FICHEIRO DE CONFIGURACAO COM CONFIG ATUAL E DOWNLOAD LOG UE" > config

if [ $software == 'srs' ]
then
scp $HOSTNAME1@$IPNUC1:~/sysENB.log $cwd
scp $HOSTNAME1@$IPNUC1:~/logENB.log $cwd
scp $HOSTNAME1@$IPNUC1:~/cpuENB.log $cwd
scp $HOSTNAME1@$IPNUC1:/etc/srslte/enb.conf $cwd
scp $HOSTNAME2@$IPNUC2:~/sysEPC.log $cwd
scp $HOSTNAME2@$IPNUC2:~/cpuEPC.log $cwd
scp $HOSTNAME2@$IPNUC2:~/logEPC.log $cwd
scp $HOSTNAME2@$IPNUC2:~/iperf3.log $cwd
scp $HOSTNAME2@$IPNUC2:/etc/srslte/epc.conf $cwd

elif [ $software == 'oai' ]
then
scp $HOSTNAME1@$IPNUC1:~/sysENB.log $cwd
scp $HOSTNAME1@$IPNUC1:~/logENB.log $cwd
scp $HOSTNAME1@$IPNUC1:~/cpuENB.log $cwd
scp $HOSTNAME1@$IPNUC1:~/OAI_BLADERF/openairinterface5g/enb-band7-5mhz.conf $cwd
scp $HOSTNAME3@$IPVM3:~/openair-cn/scripts/sysOAIIEPC.log $cwd
scp $HOSTNAME3@$IPVM3:~/openair-cn/scripts/cpuOAIIEPC.log $cwd
scp $HOSTNAME3@$IPVM3:~/openair-cn/scripts/logHSS.log $cwd
scp $HOSTNAME3@$IPVM3:~/openair-cn/scripts/logMME.log $cwd
scp $HOSTNAME3@$IPVM3:~/openair-cn/scripts/logSPGW.log $cwd
scp $HOSTNAME3@$IPVM3:~/iperf3.log $cwd

fi

echo "Finished!"
```

Figure B.3: Script designed to centralize the logs in one computer.

Appendix C

CSV Files

Appendix C presents the complete CSV files from where the plots and conclusions were drawn. There are four different CSV files:

C.1 TX_Rate Occurrences.

Table C.1 presents the complete CSV File of each TX_Rate occurrence of the data that was captured during the Ground Experiments.

Table C.1: TX_Rate Occurrences.

ID	1	2	5.5	6	6.5	7.2222	9	11	12	13	14.4444	18	19.5	21.6667	24	26	28.8889	36	39	43.3333	48	52	54	57.7778	58.5	65	72.2222
0	BDL1AF	30	73	401	0	0	0	1663	0	1621	0	0	2162	0	0	6360	0	0	3711	9050	0	3738	0	4985	3084	10199	12659
1	BDL1BF	202	394	701	0	0	0	2652	0	6152	0	0	9788	0	0	9557	0	0	3071	9028	0	3491	0	7046	2027	5759	3484
2	BDL1CF	39	136	341	0	0	0	2283	0	3834	0	0	4916	0	0	7193	0	0	3597	18029	0	6111	0	9771	3443	3597	359
3	BDL1DF	598	1236	4298	0	0	0	6621	0	7946	0	0	14170	0	0	11790	0	0	1695	1724	0	183	0	228	7	42	17
4	BDL7AF	73	79	294	0	0	0	1305	0	2602	0	0	5093	0	0	9227	0	0	5751	14491	0	5033	0	6122	3691	7335	3258
5	BDL7BF	12	65	104	0	0	0	3016	0	3786	0	0	6234	0	0	10952	0	0	6984	13315	0	5793	0	5048	2774	3821	851
6	BDL7CF	34	128	1221	0	0	0	1868	0	3351	0	0	5688	0	0	4506	0	0	3880	13984	0	5226	0	10212	4653	9396	3385
7	BDL7DF	1144	1929	3853	0	0	0	4308	0	7567	0	0	30251	0	0	6653	0	0	939	808	0	5	0	0	0	0	0
8	BDL13AF	324	432	1006	0	0	0	1557	0	3965	0	0	5276	0	0	23037	0	0	9360	20515	0	6439	0	10090	3699	7432	5420
9	BDL13BF	17	21	33	0	0	0	604	0	801	0	0	1371	0	0	2798	0	0	2764	13773	0	5058	0	8846	3380	8759	15653
10	BDL13CF	4	9	23	0	0	0	317	0	1049	0	0	1985	0	0	9413	0	0	4454	35564	0	4574	0	5084	3601	8347	2144
11	BDL13DF	519	982	3952	0	0	0	5774	0	15017	0	0	20696	0	0	9013	0	0	1841	2709	0	144	0	24	0	0	0
12	BUL1AF	0	3	0	0	0	0	8	0	31	85	0	379	752	0	1869	4067	0	9099	18670	0	19364	0	37240	25337	53542	21358
13	BUL1BF	0	0	0	0	0	0	0	0	391	341	0	889	1461	0	2882	5705	0	15178	22796	0	18494	0	32930	15650	36136	18571
14	BUL1CF	0	20	13	0	0	0	46	0	865	1554	0	3533	7274	0	7137	14141	0	23416	43053	0	17355	0	20710	1294	1981	86
15	BUL1DF	0	20	41	0	0	0	327	0	6516	13965	0	8351	13426	0	9494	9512	0	95	795	0	52	0	40	2	23	0
16	BUL7AF	0	0	0	0	0	0	0	0	48	62	0	300	683	0	2659	9033	0	7198	18245	0	15819	0	27586	18369	59377	32777
17	BUL7BF	0	0	0	0	0	0	0	0	1254	1017	0	2022	2188	0	3743	7624	0	11747	21370	0	20132	0	27974	18760	41849	15026
18	BUL7CF	0	0	10	0	0	0	16	0	1617	3632	0	2097	3583	0	3724	7739	0	14554	31355	0	19642	0	27346	10960	17845	4830
19	BUL7DF	0	3	1749	0	0	0	2512	0	3739	5033	0	8076	11619	0	9668	22218	0	9226	6670	0	7	0	7	4	1	0
20	BUL13AF	0	0	0	0	0	0	0	0	192	800	0	1015	3056	0	4390	11622	0	16553	27659	0	24449	0	32686	21828	27873	6264
21	BUL13BF	0	0	1	0	0	0	0	0	7	27	0	174	342	0	693	1400	0	3459	10675	0	13007	0	35154	27974	117021	58934
22	BUL13CF	0	0	0	0	0	0	0	0	0	0	0	153	265	0	738	2723	0	10960	35638	0	27834	0	61644	20007	43278	4480
23	BUL13DF	0	11	121	0	0	0	1028	0	8204	16137	0	11134	20819	0	16106	19165	0	4405	8745	0	215	0	48	15	8	0
24	PDL1AF	0	0	0	0	0	0	0	0	1939	2652	0	3530	4851	0	7798	10650	0	6931	8672	0	2377	0	2829	306	458	91
25	PDL1BF	0	0	0	0	0	0	0	0	3578	3889	0	3552	4141	0	4477	5099	0	112	90	0	29	0	33	2	2	0
26	PDL1CF	0	0	0	0	0	0	0	0	2468	3052	0	3198	3766	0	5172	6237	0	136	123	0	2	0	6	0	0	0
27	PDL1DF	0	0	0	0	0	0	0	0	1833	2200	0	859	1254	0	182	197	0	0	0	0	0	0	0	0	0	0
28	PDL7AF	0	0	0	0	0	0	0	0	1418	1926	0	4187	4744	0	9521	10961	0	8367	10830	0	3879	0	4771	467	855	166
29	PDL7BF	0	0	0	0	0	0	0	0	3159	3424	0	3423	3800	0	3696	5164	0	1918	1994	0	114	0	46	1	2	0
30	PDL7CF	0	0	0	0	0	0	0	0	1959	1884	0	5531	5652	0	5638	5836	0	2367	2675	0	2	0	13	0	0	0
31	PDL7DF	0	0	0	0	0	0	0	0	1650	2101	0	2487	2576	0	960	1036	0	1	0	0	0	0	0	0	0	0
32	PDL13AF	0	0	0	0	0	0	0	0	895	1417	0	3202	4655	0	9380	18321	0	9790	16058	0	3490	0	4825	527	883	155
33	PDL13BF	0	0	0	0	0	0	0	0	3129	3985	0	6345	9476	0	7539	11779	0	1387	1644	0	138	0	86	1	2	0
34	PDL13CF	0	0	0	0	0	0	0	0	1661	2315	0	4952	4742	0	12559	13723	0	739	807	0	0	0	0	0	0	0
35	PDL13DF	0	0	0	0	0	0	0	0	3569	3726	0	3753	4468	0	3409	4243	0	3	10	0	0	0	0	0	0	0
36	PUL1AF	0	30	0	0	0	0	36	0	456	884	0	5839	11251	0	16074	25686	0	43082	73570	0	10873	0	27412	4994	14874	12323
37	PUL1BF	0	20	20	0	0	0	69	0	3192	4713	0	13645	22282	0	19782	23548	0	15581	31044	0	4061	0	11439	1689	4455	3113
38	PUL1CF	0	0	46	0	0	0	85	0	4893	5152	0	15042	25234	0	17640	22500	0	16046	28638	0	2541	0	10788	1494	3924	2885
39	PUL1DF	0	8	147	0	0	0	790	0	8545	9557	0	7345	6933	0	3702	3448	0	44	20	0	0	0	0	0	0	0
40	PUL7AF	0	0	0	0	0	0	0	0	447	536	0	4447	6069	0	13008	28979	0	66821	143663	0	16414	0	30913	5512	29204	14167
41	PUL7BF	0	0	4	0	0	0	31	0	5636	7778	0	26181	43159	0	20575	35082	0	19129	34226	0	3103	0	12053	1758	4278	3006
42	PUL7CF	0	0	59	0	0	0	279	0	13522	22538	0	13522	22538	0	17240	28358	0	19177	27638	0	958	0	1827	221	265	103
43	PUL7DF	0	12	124	0	0	0	661	0	11436	17107	0	18588	17658	0	7333	10432	0	440	708	0	14	0	22	5	2	0
44	PUL13AF	0	0	0	0	0	0	0	0	379	484	0	4234	9293	0	18339	36458	0	69279	125806	0	13829	0	34288	10431	29428	24729
45	PUL13BF	0	0	3	0	0	0	0	0	5730	8601	0	19933	29147	0	29878	51114	0	27426	48269	0	5451	0	15007	2057	5535	3507
46	PUL13CF	0	3	0	0	0	0	36	0	2379	4913	0	19352	29668	0	22360	44874	0	24047	61765	0	359	0	433	196	212	1
47	PUL13DF	0	6	61	0	0	0	517	0	5631	6550	0	18409	28958	0	24722	36377	0	9777	11895	0	9	0	16	2	1	0

C.2 RSSI Statistical Data.

Table [C.3](#) presents the complete CSV File of the RSSI statistical data that was captured during the Ground Experiments.

Table C.2: RSSI Statistical Data.

	ID	RSSI25	RSSI50	RSSI75	RSSICount	RSSImax	RSSImean	RSSImin	RSSIstd	who
0	BDL1AF	-67.0	-64.0	-62.0	41688.0	-55.0	-64.50645269621954	-90.0	4.009441038359149	Router
1	BDL1AF	-72.0	-67.0	-63.0	59736.0	-32.0	-67.31265903307889	-95.0	7.914269726836209	Client
2	BDL1BF	-69.0	-65.0	-63.0	34214.0	-57.0	-66.43303910679839	-88.0	4.292522254618238	Router
3	BDL1BF	-73.0	-68.0	-63.0	63352.0	-35.0	-68.69224333880541	-95.0	7.777138683403123	Client
4	BDL1CF	-71.0	-70.0	-68.0	33241.0	-63.0	-69.44598537950122	-89.0	2.866280093584326	Router
5	BDL1CF	-73.0	-67.0	-65.0	63649.0	-47.0	-69.19899762761395	-94.0	6.933137786992154	Client
6	BDL1DF	-80.0	-78.0	-77.0	22106.0	-66.0	-78.33665068307246	-89.0	2.294754160605269	Router
7	BDL1DF	-77.0	-74.0	-72.0	50555.0	-59.0	-75.17709425378301	-98.0	5.663591130399989	Client
8	BDL7AF	-67.0	-65.0	-62.0	56257.0	-53.0	-64.88968483922001	-91.0	3.575274817277515	Router
9	BDL7AF	-76.0	-70.0	-64.0	64354.0	-44.0	-69.92556795226405	-93.0	7.2586378166256775	Client
10	BDL7BF	-68.0	-66.0	-64.0	49401.0	-59.0	-65.9191514341815	-91.0	3.023320595911724	Router
11	BDL7BF	-74.0	-66.0	-63.0	62755.0	-44.0	-68.61442116166043	-92.0	7.502314129021261	Client
12	BDL7CF	-70.0	-68.0	-66.0	51397.0	-62.0	-68.27919917504913	-90.0	3.258857945078604	Router
13	BDL7CF	-73.0	-67.0	-66.0	67532.0	-46.0	-69.92594621808921	-96.0	7.060596181380001	Client
14	BDL7DF	-78.0	-75.0	-74.0	30621.0	-69.0	-75.80879135233399	-89.0	2.9408576651925813	Router
15	BDL7DF	-76.0	-74.0	-72.0	57457.0	-52.0	-74.18697460709748	-98.0	5.764579200635772	Client
16	BDL13AF	-68.0	-66.0	-64.0	47891.0	-56.0	-66.16710864254244	-91.0	3.6361667587882787	Router
17	BDL13AF	-76.0	-71.0	-67.0	98552.0	-11.0	-70.64234109911519	-92.0	7.4127759293909605	Client
18	BDL13BF	-66.0	-64.0	-61.0	53454.0	-54.0	-63.85033860889737	-90.0	3.2170722683205706	Router
19	BDL13BF	-76.0	-71.0	-64.0	63878.0	-36.0	-69.80598954256551	-90.0	7.902170664453974	Client
20	BDL13CF	-68.0	-68.0	-67.0	55545.0	-64.0	-67.67680259249258	-88.0	1.4727221108862931	Router
21	BDL13CF	-73.0	-71.0	-66.0	76568.0	-46.0	-70.3194415421586	-93.0	6.495366939351436	Client
22	BDL13DF	-79.0	-77.0	-74.0	29281.0	-69.0	-76.80605170588436	-90.0	2.9710095593643877	Router

Table C.2: RSSI Statistical Data.

ID	RSSI25	RSSI50	RSSI75	RSSICount	RSSIImax	RSSIImean	RSSIImin	RSSIstd	who
23	-77.0	-75.0	-73.0	60671.0	-57.0	-75.70224654282936	-98.0	5.250224800411697	Client
24	-75.0	-69.0	-65.0	104681.0	-33.0	-69.83169820693344	-90.0	7.342188159396892	Client
25	-67.0	-64.0	-62.0	191804.0	-52.0	-64.24735667660737	-91.0	4.040196518177861	Router
26	-78.0	-70.0	-64.0	89343.0	-45.0	-70.90505131907368	-93.0	7.702161512235942	Client
27	-68.0	-65.0	-62.0	171424.0	-55.0	-65.562173324622	-90.0	4.410133408674006	Router
28	-76.0	-70.0	-66.0	70750.0	-48.0	-72.15598586572438	-95.0	7.067669376539081	Client
29	-72.0	-69.0	-67.0	142651.0	-64.0	-69.7548141968861	-89.0	2.839529170834655	Router
30	-79.0	-76.0	-74.0	28106.0	-56.0	-77.12399487653882	-100.0	5.831558646248017	Client
31	-79.0	-78.0	-77.0	65078.0	-70.0	-78.03850763698946	-92.0	2.0713397201126824	Router
32	-77.0	-75.0	-68.0	97572.0	-41.0	-72.32722502357234	-93.0	7.072056507662104	Client
33	-65.0	-63.0	-61.0	192238.0	-52.0	-63.04410678429863	-90.0	3.785180381286137	Router
34	-79.0	-68.0	-63.0	69794.0	-46.0	-70.48340831590109	-91.0	8.372387729248821	Client
35	-67.0	-65.0	-63.0	174820.0	-57.0	-65.22940167028943	-90.0	3.3126237115837083	Router
36	-76.0	-73.0	-68.0	80039.0	-54.0	-72.47779207636277	-96.0	5.706214771751654	Client
37	-69.0	-68.0	-67.0	153474.0	-63.0	-68.21515044893597	-90.0	2.576757808748099	Router
38	-77.0	-72.0	-71.0	35131.0	-44.0	-75.16165210213202	-100.0	6.564170048759807	Client
39	-75.0	-74.0	-73.0	86803.0	-69.0	-74.80513346312915	-94.0	2.5748016194042	Router
40	-77.0	-69.0	-66.0	97804.0	-29.0	-71.50781154145025	-97.0	7.1393163295281905	Client
41	-70.0	-67.0	-64.0	178418.0	-56.0	-67.17659092692442	-92.0	3.967876546419392	Router
42	-78.0	-73.0	-65.0	137799.0	-39.0	-71.94531164957655	-91.0	7.290227550060731	Client
43	-65.0	-63.0	-61.0	268901.0	-54.0	-63.15636237871931	-90.0	2.744376302919553	Router
44	-81.0	-68.0	-65.0	111992.0	-54.0	-71.66302057289806	-90.0	7.718122145134044	Client
45	-68.0	-67.0	-65.0	207724.0	-60.0	-66.83823246230575	-90.0	2.185406226029179	Router

Table C.2: RSSI Statistical Data.

	ID	RSSI25	RSSI50	RSSI75	RSSICount	RSSImin	RSSImax	RSSImean	RSSImin	RSSIstd	who
46	BUL13DF	-77.0	-75.0	-74.0	46491.0	-57.0	-75.6670753479168	-75.6670753479168	-97.0	4.869695916006395	Client
47	BUL13DF	-77.0	-76.0	-73.0	109405.0	-68.0	-75.10911749920022	-75.10911749920022	-92.0	2.779880780039734	Router
48	PDL1AF	-70.0	-68.0	-66.0	161395.0	-58.0	-67.19194522754732	-67.19194522754732	-82.0	3.8982059108912277	Router
49	PDL1AF	-82.0	-72.0	-67.0	56448.0	-33.0	-73.30187074829932	-73.30187074829932	-93.0	8.686644601604582	Client
50	PDL1BF	-74.0	-70.0	-68.0	102472.0	-60.0	-70.49080724490592	-70.49080724490592	-92.0	4.0942141871218345	Router
51	PDL1BF	-74.0	-71.0	-70.0	31362.0	-51.0	-73.36123333971048	-73.36123333971048	-93.0	6.221926427323101	Client
52	PDL1CF	-78.0	-72.0	-68.0	104814.0	-60.0	-72.04331482435552	-72.04331482435552	-88.0	5.333011138326258	Router
53	PDL1CF	-76.0	-75.0	-74.0	29196.0	-70.0	-75.23948486093985	-75.23948486093985	-94.0	2.899288713121637	Client
54	PDL1DF	-84.0	-84.0	-82.0	21835.0	-76.0	-83.3189832837188	-83.3189832837188	-92.0	2.0776940045438796	Router
55	PDL1DF	-83.0	-81.0	-80.0	13343.0	-69.0	-81.2882410252567	-81.2882410252567	-89.0	2.0878154955505415	Client
56	PDL7AF	-70.0	-68.0	-66.0	180783.0	-58.0	-67.57416349988661	-67.57416349988661	-92.0	3.4236588912356964	Router
57	PDL7AF	-82.0	-71.0	-67.0	64298.0	-38.0	-73.52843012224331	-73.52843012224331	-92.0	8.22857901282776	Client
58	PDL7BF	-72.0	-70.0	-68.0	109666.0	-60.0	-70.38730326628125	-70.38730326628125	-84.0	4.005023154594919	Router
59	PDL7BF	-75.0	-73.0	-71.0	32733.0	-57.0	-74.29868939602237	-74.29868939602237	-94.0	5.62065266110375	Client
60	PDL7CF	-78.0	-76.0	-74.0	62789.0	-68.0	-76.05360811607129	-76.05360811607129	-88.0	3.138252126710367	Router
61	PDL7CF	-77.0	-75.0	-73.0	35127.0	-59.0	-75.32883536880462	-75.32883536880462	-94.0	3.446852158707772	Client
62	PDL7DF	-82.0	-80.0	-78.0	29651.0	-74.0	-80.2226569087046	-80.2226569087046	-94.0	2.2503514365874873	Router
63	PDL7DF	-80.0	-79.0	-77.0	17746.0	-67.0	-78.71379465795108	-78.71379465795108	-91.0	2.279502586313629	Client
64	PDL13AF	-70.0	-68.0	-66.0	163486.0	-58.0	-67.38981931174534	-67.38981931174534	-88.0	3.0582990947053332	Router
65	PDL13AF	-82.0	-73.0	-67.0	74802.0	-22.0	-74.47125745300927	-74.47125745300927	-91.0	8.210454803132189	Client
66	PDL13BF	-74.0	-72.0	-68.0	147960.0	-60.0	-71.62738577994052	-71.62738577994052	-82.0	4.210758407937724	Router
67	PDL13BF	-78.0	-74.0	-70.0	50751.0	-50.0	-74.2069909525133	-74.2069909525133	-93.0	6.705081610307962	Client
68	PDL13CF	-78.0	-78.0	-76.0	85928.0	-72.0	-77.47402476491946	-77.47402476491946	-86.0	1.724802631769166	Router

Table C.2: RSSI Statistical Data.

	ID	RSSI25	RSSI50	RSSI75	RSSICount	RSSImax	RSSImean	RSSImin	RSSIstd	who
69	PDL13CF	-76.0	-75.0	-74.0	44269.0	-62.0	-75.42151392622377	-93.0	2.342608375219942	Client
70	PDL13DF	-80.0	-78.0	-76.0	55062.0	-74.0	-78.55704478587774	-92.0	2.884160193340484	Router
71	PDL13DF	-80.0	-77.0	-76.0	29874.0	-62.0	-77.77324764008837	-93.0	3.8432967079983493	Client
72	PUL1AF	-82.0	-77.0	-69.0	78864.0	-32.0	-75.78209322377764	-94.0	7.260594525254663	Client
73	PUL1AF	-70.0	-68.0	-64.0	247567.0	-58.0	-66.85989651286319	-84.0	3.768330707609565	Router
74	PUL1BF	-77.0	-72.0	-70.0	41427.0	-53.0	-73.53518236898641	-92.0	5.451992150237334	Client
75	PUL1BF	-74.0	-72.0	-70.0	159470.0	-60.0	-71.3702765410422	-82.0	3.6756802540866307	Router
76	PUL1CF	-79.0	-77.0	-75.0	35221.0	-61.0	-77.16413503307686	-93.0	3.5534538044273387	Client
77	PUL1CF	-78.0	-74.0	-70.0	158800.0	-60.0	-73.58479848866499	-92.0	5.3721415945037805	Router
78	PUL1DF	-83.0	-81.0	-79.0	15139.0	-70.0	-81.02562916969417	-90.0	2.371775173031094	Client
79	PUL1DF	-84.0	-84.0	-82.0	49007.0	-74.0	-83.25443304017793	-94.0	2.463571570693916	Router
80	PUL7AF	-81.0	-73.0	-69.0	112623.0	-44.0	-74.46401711906094	-94.0	7.063041692456178	Client
81	PUL7AF	-68.0	-66.0	-62.0	360203.0	-56.0	-65.65044155656672	-82.0	3.533833229517087	Router
82	PUL7BF	-77.0	-73.0	-71.0	55848.0	-52.0	-74.64014825956167	-93.0	4.9260781910136835	Client
83	PUL7BF	-72.0	-72.0	-70.0	217319.0	-60.0	-71.02244166409749	-84.0	3.0150519180681847	Router
84	PUL7CF	-78.0	-76.0	-74.0	52196.0	-66.0	-76.17899839068127	-92.0	3.388197956437368	Client
85	PUL7CF	-80.0	-76.0	-74.0	150139.0	-68.0	-76.26920387107947	-94.0	3.3254225334558507	Router
86	PUL7DF	-81.0	-79.0	-78.0	32366.0	-69.0	-79.35098560217513	-89.0	2.1864490289217886	Client
87	PUL7DF	-82.0	-80.0	-80.0	89713.0	-72.0	-80.559874265714	-94.0	2.169408674096641	Router
88	PUL13AF	-81.0	-73.0	-69.0	115339.0	-26.0	-74.52513026816601	-92.0	6.88308231260459	Client
89	PUL13AF	-70.0	-68.0	-66.0	376985.0	-58.0	-67.23420295237212	-84.0	3.323915336040036	Router
90	PUL13BF	-79.0	-75.0	-73.0	73684.0	-54.0	-75.49511427175506	-92.0	4.469970528526425	Client
91	PUL13BF	-76.0	-72.0	-68.0	251982.0	-60.0	-71.88246779531872	-84.0	3.882450114756819	Router

Table C.2: RSSI Statistical Data.

	ID	RSSI25	RSSI50	RSSI75	RSSIcount	RSSIImax	RSSIImean	RSSIImin	RSSIstd	who
92	PUL13CF	-77.0	-75.0	-74.0	66138.0	-58.0	-75.66545707460159	-94.0	2.7344699491458564	Client
93	PUL13CF	-78.0	-76.0	-76.0	227049.0	-70.0	-76.47921814233932	-86.0	1.8193953844231912	Router
94	PUL13DF	-78.0	-77.0	-76.0	44347.0	-65.0	-77.29036913432701	-95.0	2.7484960880254343	Client
95	PUL13DF	-78.0	-78.0	-76.0	147723.0	-72.0	-78.15925752929469	-92.0	2.0414486706573594	Router

C.3 Throughput Statistical Data.

Table [C.2](#) presents the complete CSV File of the Throughput statistical data that was captured during the Ground Experiments.

C.4 Throughput Packet Count and Mean per Iteration.

Table [C.4](#) presents the complete CSV File of the Throughput Packet Count and Mean Per Iteration of test that was captured during the Ground Experiments.

Table C.3: Throughput Statistical Data.

	ID	thCOUNT	thMEAN	thSTD	thMIN	th25	th50	th75	thMAX
0	BDL1AF	287	1.86E+07	1.08E+07	7.80E+05	1.00E+07	1.70E+07	2.67E+07	4.57E+07
1	BDL1BF	286	1.18E+07	7.70E+06	3.10E+04	4.68E+06	1.15E+07	1.75E+07	3.33E+07
2	BDL1CF	286	1.22E+07	5.06E+06	1.79E+06	8.00E+06	1.27E+07	1.61E+07	2.44E+07
3	BDL1DF	275	3.98E+06	2.61E+06	7.09E+03	2.19E+06	3.69E+06	5.24E+06	1.51E+07
4	BDL7AF	289	1.78E+07	7.87E+06	7.97E+05	1.24E+07	1.71E+07	2.28E+07	4.10E+07
5	BDL7BF	289	1.29E+07	5.30E+06	7.25E+04	8.72E+06	1.31E+07	1.62E+07	2.78E+07
6	BDL7CF	287	1.77E+07	8.21E+06	1.73E+06	1.24E+07	1.78E+07	2.32E+07	4.55E+07
7	BDL7DF	251	3.46E+06	2.94E+06	7.42E+03	1.00E+06	2.83E+06	5.01E+06	1.43E+07
8	BDL13AF	289	1.38E+07	8.32E+06	2.13E+05	7.77E+06	1.29E+07	1.81E+07	4.20E+07
9	BDL13BF	290	2.52E+07	1.13E+07	1.24E+06	1.74E+07	2.31E+07	3.22E+07	5.00E+07
10	BDL13CF	290	2.01E+07	6.15E+06	2.65E+06	1.58E+07	2.02E+07	2.39E+07	4.10E+07
11	BDL13DF	259	4.56E+06	3.55E+06	8.71E+03	2.24E+06	4.09E+06	5.80E+06	2.29E+07
12	BUL1AF	291	1.14E+07	3.19E+06	3.81E+06	8.97E+06	1.09E+07	1.33E+07	2.00E+07
13	BUL1BF	290	9.82E+06	3.69E+06	3.51E+06	7.02E+06	9.10E+06	1.20E+07	2.03E+07
14	BUL1CF	293	7.40E+06	3.19E+06	1.04E+06	4.45E+06	7.63E+06	9.78E+06	1.55E+07
15	BUL1DF	282	2.85E+06	1.74E+06	5.82E+04	1.34E+06	2.55E+06	4.05E+06	7.74E+06
16	BUL7AF	290	1.64E+07	6.20E+06	2.64E+06	1.20E+07	1.56E+07	2.09E+07	3.26E+07
17	BUL7BF	290	1.07E+07	3.35E+06	2.53E+06	8.45E+06	1.05E+07	1.26E+07	2.05E+07
18	BUL7CF	291	8.75E+06	4.04E+06	1.28E+06	5.64E+06	9.30E+06	1.16E+07	1.82E+07
19	BUL7DF	268	4.04E+06	2.57E+06	3.21E+03	1.63E+06	4.04E+06	6.43E+06	9.08E+06
20	BUL13AF	292	1.08E+07	4.35E+06	2.05E+06	7.53E+06	1.02E+07	1.40E+07	2.34E+07
21	BUL13BF	293	1.71E+07	4.68E+06	4.97E+06	1.48E+07	1.77E+07	2.02E+07	2.86E+07
22	BUL13CF	291	1.51E+07	3.81E+06	5.40E+06	1.24E+07	1.49E+07	1.80E+07	2.29E+07
23	BUL13DF	274	5.54E+06	2.85E+06	5.05E+03	3.80E+06	5.31E+06	6.83E+06	1.66E+07
24	PDL1AF	268	9.74E+06	5.57E+06	3.38E+05	4.65E+06	9.64E+06	1.41E+07	2.40E+07
25	PDL1BF	261	2.56E+06	2.00E+06	3.22E+04	9.56E+05	2.19E+06	3.62E+06	9.11E+06
26	PDL1CF	241	2.55E+06	2.02E+06	2.03E+04	1.13E+06	1.99E+06	3.48E+06	9.62E+06
27	PDL1DF	181	1.06E+06	1.06E+06	4.98E+03	2.45E+05	7.97E+05	1.55E+06	6.10E+06
28	PDL7AF	281	1.04E+07	5.29E+06	1.85E+04	6.89E+06	1.02E+07	1.35E+07	2.69E+07
29	PDL7BF	245	3.13E+06	2.56E+06	9.54E+03	1.32E+06	2.60E+06	4.21E+06	1.45E+07
30	PDL7CF	225	4.46E+06	3.55E+06	3.66E+03	1.76E+06	3.66E+06	6.37E+06	1.52E+07
31	PDL7DF	198	1.60E+06	1.70E+06	3.64E+03	3.96E+05	1.06E+06	2.29E+06	8.80E+06
32	PDL13AF	288	1.38E+07	5.16E+06	7.79E+05	1.02E+07	1.42E+07	1.78E+07	2.73E+07
33	PDL13BF	279	7.34E+06	4.61E+06	2.32E+04	3.76E+06	6.29E+06	1.05E+07	2.17E+07
34	PDL13CF	246	5.49E+06	3.43E+06	1.01E+04	2.97E+06	5.19E+06	7.52E+06	1.73E+07
35	PDL13DF	234	3.27E+06	2.72E+06	3.64E+03	1.22E+06	2.69E+06	4.60E+06	1.25E+07
36	PUL1AF	292	1.01E+07	3.56E+06	2.40E+06	7.43E+06	9.81E+06	1.26E+07	2.10E+07
37	PUL1BF	288	6.43E+06	3.83E+06	2.78E+05	4.14E+06	5.61E+06	7.34E+06	2.10E+07
38	PUL1CF	287	6.44E+06	3.97E+06	3.34E+04	3.66E+06	5.45E+06	8.46E+06	2.10E+07
39	PUL1DF	239	2.08E+06	1.44E+06	3.70E+03	1.01E+06	1.83E+06	3.00E+06	7.28E+06
40	PUL7AF	293	1.48E+07	4.36E+06	2.92E+06	1.17E+07	1.47E+07	1.82E+07	2.47E+07
41	PUL7BF	286	8.89E+06	3.38E+06	6.59E+05	6.55E+06	8.67E+06	1.10E+07	2.10E+07
42	PUL7CF	284	6.00E+06	3.13E+06	6.38E+05	3.53E+06	5.55E+06	8.03E+06	1.54E+07
43	PUL7DF	277	3.60E+06	1.83E+06	7.64E+03	2.19E+06	3.53E+06	4.73E+06	9.17E+06
44	PUL13AF	293	1.55E+07	4.59E+06	2.67E+06	1.23E+07	1.54E+07	1.94E+07	2.43E+07
45	PUL13BF	289	1.03E+07	3.95E+06	1.02E+04	7.33E+06	9.80E+06	1.30E+07	2.10E+07
46	PUL13CF	283	9.20E+06	3.64E+06	3.18E+04	6.75E+06	8.98E+06	1.18E+07	2.07E+07
47	PUL13DF	253	6.48E+06	3.24E+06	8.18E+03	4.10E+06	6.31E+06	8.91E+06	1.46E+07

Table C.4: Throughput Packet Count and Mean per Iteration.

	ID	countALL	meanALL	count1	mean1	count2	mean2	count3	mean3	count4	mean4	count5	mean5
0	BDL1AF	428817	1.85E+07	109925	2.45E+07	135030	2.91E+07	46761	9.84E+06	42673	9.06E+06	94428	2.00E+07
1	BDL1BF	273755	1.17E+07	89447	1.90E+07	60533	1.28E+07	21641	4.60E+06	39523	8.45E+06	62611	1.34E+07
2	BDL1CF	285075	1.22E+07	61019	1.31E+07	69526	1.50E+07	67589	1.43E+07	25386	5.42E+06	61555	1.32E+07
3	BDL1DF	90324	3.87E+06	17663	3.80E+06	15738	3.39E+06	12075	2.64E+06	22693	4.84E+06	22155	4.67E+06
4	BDL7AF	416755	1.78E+07	94577	2.04E+07	91327	1.94E+07	68159	1.46E+07	89603	1.90E+07	73089	1.55E+07
5	BDL7BF	303276	1.29E+07	50010	1.06E+07	80892	1.71E+07	56496	1.21E+07	52507	1.12E+07	63371	1.34E+07
6	BDL7CF	413761	1.77E+07	28489	6.10E+06	89029	1.90E+07	73958	1.57E+07	130791	2.80E+07	91494	1.95E+07
7	BDL7DF	72125	3.07E+06	34525	7.35E+06	5587	1.18E+06	13094	2.78E+06	13846	2.94E+06	5073	1.11E+06
8	BDL13AF	321754	1.38E+07	73422	1.61E+07	62171	1.34E+07	46963	9.92E+06	112546	2.37E+07	26652	5.79E+06
9	BDL13BF	586933	2.53E+07	106009	2.26E+07	194263	4.31E+07	67903	1.45E+07	95504	2.02E+07	123254	2.61E+07
10	BDL13CF	469091	2.00E+07	77222	1.67E+07	77250	1.64E+07	116042	2.47E+07	97605	2.07E+07	100972	2.16E+07
11	BDL13DF	98123	4.23E+06	21892	4.66E+06	19038	4.04E+06	17974	3.94E+06	34556	7.52E+06	4663	1.00E+06
12	BUL1AF	267066	1.14E+07	55868	1.21E+07	53930	1.17E+07	54215	1.15E+07	47087	9.88E+06	55966	1.18E+07
13	BUL1BF	230505	9.79E+06	50872	1.08E+07	51092	1.09E+07	34631	7.34E+06	34376	7.28E+06	59534	1.26E+07
14	BUL1CF	175115	7.42E+06	16349	3.42E+06	44687	9.42E+06	48632	1.04E+07	23927	5.01E+06	41520	8.86E+06
15	BUL1DF	65388	2.78E+06	6586	1.40E+06	18397	3.92E+06	5583	1.21E+06	20805	4.35E+06	14017	3.04E+06
16	BUL7AF	384392	1.64E+07	59329	1.29E+07	85154	1.80E+07	80331	1.74E+07	85681	1.80E+07	73897	1.56E+07
17	BUL7BF	252719	1.07E+07	39833	8.42E+06	56412	1.18E+07	55236	1.19E+07	53097	1.13E+07	48141	1.03E+07
18	BUL7CF	205125	8.68E+06	12200	2.64E+06	56023	1.17E+07	37108	7.98E+06	50908	1.07E+07	48886	1.04E+07
19	BUL7DF	87726	3.80E+06	29899	6.26E+06	29899	6.26E+06	7364	1.38E+06	15057	3.91E+06	5507	1.18E+06
20	BUL13AF	254525	1.08E+07	31627	6.76E+06	65059	1.39E+07	63396	1.34E+07	52353	1.11E+07	42090	8.91E+06
21	BUL13BF	403777	1.71E+07	81078	1.72E+07	94197	2.00E+07	54785	1.15E+07	94838	2.02E+07	78879	1.66E+07
22	BUL13CF	354569	1.51E+07	50167	1.08E+07	66434	1.41E+07	81852	1.74E+07	76246	1.60E+07	79870	1.70E+07
23	BUL13DF	123495	5.21E+06	39145	8.19E+06	25546	5.44E+06	22413	4.76E+06	25112	5.25E+06	11279	2.41E+06
24	PDL1AF	210015	9.24E+06	15814	4.82E+06	16495	3.55E+06	49285	1.04E+07	62950	1.35E+07	65471	1.39E+07
25	PDL1BF	56194	2.44E+06	13247	2.85E+06	14427	3.14E+06	5070	1.07E+06	7227	1.55E+06	16223	3.59E+06
26	PDL1CF	54215	2.34E+06	6545	1.40E+06	12335	2.65E+06	10005	2.23E+06	12296	2.66E+06	13034	2.79E+06
27	PDL1DF	18681	8.10E+05	2075	4.58E+05	2766	6.12E+05	2943	6.41E+05	3172	6.80E+05	7725	1.66E+06
28	PDL7AF	236312	1.01E+07	45355	9.65E+06	42412	9.30E+06	30621	6.58E+06	60850	1.30E+07	57074	1.22E+07
29	PDL7BF	65921	2.83E+06	15025	3.21E+06	19537	4.24E+06	11450	2.48E+06	14035	2.97E+06	5874	1.25E+06
30	PDL7CF	89203	3.85E+06	8563	1.87E+06	22788	4.90E+06	8578	1.87E+06	16940	3.64E+06	32334	6.94E+06
31	PDL7DF	29412	1.28E+06	3446	7.52E+05	4372	9.27E+05	5355	1.14E+06	9985	2.24E+06	6254	1.33E+06
32	PDL13AF	321440	1.37E+07	43751	9.40E+06	62355	1.32E+07	61771	1.31E+07	80270	1.71E+07	73293	1.58E+07
33	PDL13BF	166833	7.13E+06	19147	4.10E+06	65561	1.40E+07	29471	6.29E+06	34830	7.43E+06	17824	3.84E+06
34	PDL13CF	119981	5.16E+06	26191	5.60E+06	28390	6.13E+06	22043	4.75E+06	18287	3.95E+06	25070	5.37E+06
35	PDL13DF	66817	2.86E+06	5310	1.16E+06	12312	2.62E+06	15368	3.28E+06	20496	4.38E+06	13331	2.87E+06
36	PUL1AF	240389	1.01E+07	60025	1.27E+07	37477	7.82E+06	56713	1.19E+07	43917	9.17E+06	42257	8.85E+06
37	PUL1BF	152880	6.39E+06	60025	1.27E+07	27782	5.75E+06	20297	4.26E+06	16931	3.51E+06	27845	5.77E+06
38	PUL1CF	152414	6.37E+06	60025	1.27E+07	15436	3.13E+06	20894	4.41E+06	21167	4.45E+06	34892	7.20E+06
39	PUL1DF	42264	1.79E+06	8046	1.73E+06	9205	1.93E+06	4830	9.97E+05	5788	1.22E+06	14395	3.07E+06
40	PUL7AF	352423	1.48E+07	60025	1.27E+07	63182	1.32E+07	80390	1.68E+07	76103	1.60E+07	72723	1.54E+07
41	PUL7BF	209819	8.86E+06	60025	1.27E+07	37357	7.90E+06	29628	6.12E+06	39122	8.46E+06	43687	9.16E+06
42	PUL7CF	142355	5.93E+06	17299	3.64E+06	35515	7.43E+06	17805	3.69E+06	26057	5.49E+06	45679	9.43E+06
43	PUL7DF	83938	3.50E+06	16315	3.43E+06	10671	2.24E+06	21772	4.58E+06	14008	2.89E+06	21172	4.37E+06
44	PUL13AF	370340	1.55E+07	60025	1.27E+07	65291	1.36E+07	90786	1.90E+07	76212	1.59E+07	78026	1.65E+07
45	PUL13BF	243959	1.02E+07	60025	1.27E+07	69640	1.44E+07	41935	8.67E+06	44195	9.22E+06	28164	6.01E+06
46	PUL13CF	219148	9.09E+06	45802	9.59E+06	47180	9.75E+06	45713	9.51E+06	35167	7.31E+06	45286	9.31E+06
47	PUL13DF	137556	5.80E+06	4368	9.42E+05	32157	6.63E+06	28267	6.00E+06	36769	7.75E+06	35995	7.68E+06

References

- [1] Situational Awareness System for First Responders, Feb 2020. [Online; accessed 21. Sep. 2020]. URL: <https://respondroneproject.com>.
- [2] File:2-Ray Ground Reflection.png - Wikimedia Commons, Jan 2020. [Online; accessed 21. Sep. 2020]. URL: https://commons.wikimedia.org/wiki/File:2-Ray_Ground_Reflection.png.
- [3] Tri-Band Antenna - Nuand, Sep 2020. [Online; accessed 19. Sep. 2020]. URL: <https://www.nuand.com/product/tri-band-antenna>.
- [4] WLE200NX | Compex Systems: Leader in OEM/ODM Wireless Integration, Oct 2018. [Online; accessed 21. Sep. 2020]. URL: <https://compex.com.sg/shop/wifi-module/802-11n/wle200nx>.
- [5] 802.11n primer, August 2008. [Online; accessed 21. Sep. 2020]. URL: <https://web.archive.org/web/20130217085339/http://www.airmagnet.com/assets/whitepaper/WP-802.11nPrimer.pdf>.
- [6] M. Srinivasan, S.; Haenggi. Path loss exponent estimation in large wireless networks. *Information Theory and Applications Workshop*, pages 124–129, Feb 2009.
- [7] Rádio Renascença. Fogo de Pedrógão Grande é o 11.º mais mortal do mundo desde 1900 - Renascença. *Rádio Renascença*, Jun 2017. [Online; accessed 21. Set. 2020]. URL: <https://rr.sapo.pt/2017/06/21/pais/fogo-de-pedrogao-grande-e-o-11-mais-mortal-do-mundo-desde-1900/especial/86656>.
- [8] IEEE P802.11 - TASK GROUP AX, Dec 2019. [Online; 21. Sep. 2020]. URL: http://www.ieee802.org/11/Reports/tgax_update.htm.
- [9] C. W. Bostian and A. R. Young. The application of cognitive radio to coordinated unmanned aerial vehicle (uav) missions. Technical report, Virginia Polytechnic Inst. and State Univ. Blacksburg, 2011.
- [10] M. Jakubiak. Cellular network coverage analysis using uav and sdr. Technical report, Master’s thesis, Tampere University of Technology, 2014.
- [11] D. J. Rogers. Gsm network employment on a man-portable uas. Technical report, Master’s thesis, Naval Postgraduate School, 2012.
- [12] Oleksandr Andryeyev, Alina Rubina, Oleg Golokolenko, Oleksandr Artemenko, and Andreas Mitschele-Thiel. SkySAIL: A Flexible Software-Defined Radio Enabled Micro Aerial Vehicle. *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6, Aug 2016. doi:10.1109/ICCCN.2016.7568575.

- [13] bladeRF x40 - Nuand, Sep 2020. [Online; accessed 13. Sep. 2020]. URL: <https://www.nuand.com/product/bladerf-x40>.
- [14] bladeRF 2.0 micro xA9 - Nuand, Sep 2020. [Online; accessed 13. Sep. 2020]. URL: <https://www.nuand.com/product/bladerf-xa9>.
- [15] A. National Instruments Brand. USRP B210 USB Software Defined Radio (SDR) - Ettus Research, Sep 2020. [Online; accessed 13. Sep. 2020]. URL: <https://www.ettus.com/all-products/ub210-kit>.
- [16] LimeNET Mini - Lime Microsystems, Jul 2019. [Online; accessed 19. Sep. 2020]. URL: <https://limemicro.com/products/systems/limenet-mini>.
- [17] LTE frequency band, Sep 2020. [Online; accessed 20. Sep. 2020]. URL: https://www.sqimway.com/lte_band.php.
- [18] All in one OpenAirInterface – 4G and 5G reference software, Sep 2020. [Online; accessed 16. Sep. 2020]. URL: <https://open-cells.com/index.php/2019/09/22/all-in-one-openairinterface>.
- [19] oai / openairinterface5G, Sep 2020. [Online; accessed 18. Sep. 2020]. URL: <https://gitlab.eurecom.fr/oai/openairinterface5g>.
- [20] Index of /releases/19.07.4/targets/brcm2708/bcm2710/, Sep 2020. [Online; accessed 17. Sep. 2020]. URL: <https://downloads.openwrt.org/releases/19.07.4/targets/brcm2708/bcm2710>.
- [21] ShareTechnote, Apr 2015. [Online; accessed 21. Sep. 2020]. URL: https://www.sharetechnote.com/html/Handbook_LTE_RI.html.
- [22] ShareTechnote, May 2019. [Online; accessed 19. Sep. 2020]. URL: https://www.sharetechnote.com/html/Handbook_LTE_PHR.html.
- [23] BT-100 Bias-tee Power Amplifier - Nuand, Sep 2020. [Online; accessed 18. Sep. 2020]. URL: <https://www.nuand.com/product/bt-100>.
- [24] BT-200 Bias-tee Low Noise Amplifier - Nuand, Sep 2020. [Online; accessed 18. Sep. 2020]. URL: <https://www.nuand.com/product/bt-200>.