FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Evaluation of IEEE 802.11a/g/p Transceiver for SDR

José Pedro Pereira dos Santos

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Advisor: Prof. Manuel Alberto Pereira Ricardo (PhD)

Co-Advisor: Eng. Filipe Borges Teixeira (MSc)

July 29, 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Evaluation of IEEE 802.11a/g/p Transceiver for SDR"

foi aprovada em provas realizadas em 16-07-2015

o júri

Presidente Professor Doutor Mário Jorge Moreira Leitão
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Adriano Jorge Cardoso Moreira
Professor Associado do Departamento de Sistemas de Informação da Escola de Engenharia da Universidade do Minho

Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - José Pedro Pereira dos Santos

Faculdade de Engenharia da Universidade do Porto

Resumo

Nos últimos anos, os *software defined radios* (SDRs) mudaram drasticamente a forma como a investigação e os trabalhos experimentais são conduzidos na área das comunicações sem fios. A ideia principal do SDR é implementar a maioria das funcionalidades de rádio e processamento de sinal, tais como a modulação, a desmodulação, a codificação e a filtragem em software, tornando as soluções baseadas em SDR muito mais flexíveis e facilmente modificadas do que as soluções baseadas em rádios tradicionais, onde tudo é implementado em hardware. Assim, através da tecnologia SDR as alterações na camadas física (PHY) ou na de controlo de acesso ao meio (MAC) são mais fáceis de reconfigurar.

Esta dissertação tem como objetivo principal realizar uma avaliação exaustiva da performance de uma implementação de uma rede IEEE 802.11 para SDR, bem como a sua adaptação para funcionar noutros ambientes que requerem operação a frequências mais baixas, como em ambientes subaquáticos, marítimos e subterrâneos.

Testes de interoperabilidade foram realizados com equipamento comercial, a fim de avaliar o desempenho da implementação SDR, assim como comunicações entre dois dispositivos SDR. Os resultados experimentais obtidos em diferentes frequências operação com comunicações através do ar, mostram que a utilização de redes IEEE 802.11 através de SDR permite atingir débitos UDP até 1,8 Mbit/s com atrasos inferiores a 1 ms. A adaptação da implementação SDR para operar em frequências mais baixas foi testada num testbed de água doce, onde foi possível alcançar débitos máximos na ordem dos 500 kbit/s a uma distância de 3 m, para uma frequência de 200 MHz e uma largura do canal de 5 MHz. O alcance máximo conseguido para a utilização de redes IEEE 802.11 em água doce foi de 5 m, a 200 MHz, o que excede em mais de 200% o alcance obtido nos trabalhos anteriores. Por outro lado, nós alcançamos um alcance máximo de 1.8 metros para um testbed em água salgada. Além disso, o Round Trip Time (RTT) médio entre dispositivos SDR foi menor do que 50 ms para todas as frequências avaliadas. Estes resultados comprovam que as implementações baseadas em SDR podem ser usadas para investigação e desenvolvimento na área das comunicações sem fios, como a exploração de comunicações de rádio frequência em ambientes subaquáticos.

Abstract

In recent years, software-defined radios (SDRs) have drastically changed the way that research and experimental works are made in the wireless area. The main idea of SDR is to implement most of the radio functionality and signal processing, such as modulation, demodulation, coding and filtering with software, making the SDR solutions much more flexible and easily modified than traditional radios, which everything is implemented in hardware. Thus, with SDR technology modifications at the physical (PHY) or medium access control (MAC) layer are much easier to reconfigure.

The main objective of this dissertation is to perform a detailed evaluation of the performance of one IEEE 802.11 SDR transceiver implementation as well as their adaptation to operate in other environments that require operation at lower frequencies, such as underwater, maritime and underground environments.

Multiple interoperability tests were conducted with commercial equipment in order to evaluate the performance of the SDR implementation, as well as communications between two SDR devices. Experimental results obtained with over-the-air communications at different operating frequencies show that the communication through SDR achieves UDP throughputs up to 1.8 Mbit/s with an average delay lower than 1 ms. The adaptation of the SDR implementation to operate at low frequencies was tested in a freshwater testbed, where it was possible to achieve maximum UDP throughputs of 500 kbit/s at a distance of 3 meters, for a frequency of 200 MHz and a signal bandwidth of 5 MHz. The maximum range achieved for the use of 802.11 networks in freshwater environments was 5 meters, for a frequency of 200 MHz, which exceeds in more than 200% the range achieved in previous works. On the other hand, we achieved a maximum range of 1.8 meters in a seawater testbed. Besides, the average Round Trip Time (RTT) between SDRs was lower than 50 ms for all the frequencies. These results show that SDR-based implementations can be used for research and development in the wireless area, such as the exploration of RF underwater communications.

Acknowledgments

First of all, I would like to thank my advisor Prof Dr. Manuel Ricardo for giving me the opportunity to develop my MSc dissertation inside of an extraordinary team and structure at INESC Porto.

A very special thanks to my co-advisor Eng. Filipe Borges Teixeira for all the guidance and patience during the research and evaluation period, finding always an alternative to solve the problems and to motivate me. Another big contribution was giving from Bruno Ferreira, Mário Lopes, Filipe Ribeiro, Mário Pereira and Luís Pessoa, for sharing their scientific advices and knowledge. I would also like to thank INESC Crob (LSA and Oceansys groups), for providing the facilities required to perform the dissertation underwater experiments.

I would like to express my gratitude to all my colleagues and friends that helped me during my dissertation. Finally, I would, obviously, like to mention my family, specially, my parents Helena and José for everything I am today and also for making it possible for me to accomplish my degree and for never allowing me to give up. Last but not the least, I want to leave a special thanks to my favorite future clinical analyst Catarina for all the patience and support during my dissertation, which would not been possible without her. Thank you.

José Santos

"The true sign of intelligence is not knowledge but imagination."

Albert Einstein

Contents

List of Figures

List of Tables

Abbreviations

ACK	Acknowledgement
ADC	Analog to Digital Converter
API	Application Programming Interface
ARP	Address Resolution Protocol
AUV	Autonomous Underwater Vehicle
AWGN	Additive White Gaussian Noise
BER	Bit Error Ratio
BPSK	Binary Phase Shift Keying
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
BTS	Base Transceiver Station
CCK	Complementary Code Keying
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CTS	Clear To Send
DC	Direct current
DCF	Distributed Coordination Function
DIFS	DCF Interframe Space
DSRC	Dedicated Short-Range Communications
DSP	Digital Signal Processor
FANET	Flying Ad-Hoc Network
FER	Frame Error Ratio
FFT	Fast Fourier Transformation
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
GRC	GNU Radio Companion
GPP	General-Purpose Processor
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate Frequency
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
ITS	Intelligent Transportation Systems
MAC	Medium Access Control
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input and Multiple-Output
NI	National Instruments
OFDM	Orthogonal Frequency-Division Multiplexing
PC	Personal Computer
PHY	Physical Layer
PLCP	Physical Layer Convergence Protocol

PMD	Physical Medium Dependent
PDR	Packet Delivery Ratio
PSDU	Physical layer Service Data Unit
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RTT	Round Trip Time
RTS	Request To Send
SDR	Software Defined Radio
SIFS	Short Inter-Frame Space
SIP	Session Initiation Protocol
SNR	Signal-to-Noise Ratio
UDP	User Datagram Protocol
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
VANET	Vehicular Ad Hoc Network
VNA	Vectorial Network Analyzer
VOLK	Vectorized Library of Kernels
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Context

In recent years, software-defined radios (SDRs) have drastically changed the way that research and experimental works are made in the wireless arena. Their advantage in relation to traditional hardware-based radios is that significant aspects of the communication stack, are implemented in software programs [16], while in hardware radios the lower layers are implemented in hardware, making these radios extremely limited when changes on PHY or MAC layers are required. Thus, with SDRs, most of the radio functionality and signal processing, especially modulation, demodulation, coding, encoding and filtering are implemented in software that runs on a computer and the only task of the hardware is taking care of functions, such as transmissions and reception of the signal [17]. Due to their innovative characteristics, SDRs have become an extremely useful tool for research and development in the wireless area, for the design and validation of network protocols [18].

However, there are other important factors that were crucial to ensure that the SDR technology could obtain this tremendous success, in particular, their available processing power, reduced cost, and software-only upgrades [8]. Besides, traditional radios are not flexible, specially regarding hardware updates and reconfigurations. SDRs can be easily modified by software, which is not possible in conventional radios.

In accordance with the evolution of SDR, GNURadio SDK platform [19] is currently the open-source reference tool for wireless research and academic work [18, 20] allowing the researcher to implement the signal processing blocks without needing additional hardware. Furthermore, over the past decade, a large amount of topics were addressed in many GNURadio-based open source projects, such as Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards, digital TVs, Bluetooth, Zigbee, radio astronomy and global system for mobile communications (GSM) [8].

1.2 Motivation

Wireless communications have become even more popular in the last few years, becoming one of the most active areas of technology development. In 1985, with the assignment of unlicensed spectrum in three different regions for use in Industry, Science, and Medication (ISM) application – 902-928 MHz, 2400-2483.5 MHz and 5725-5850 MHz, there was a tremendous change in wireless communications, causing the appearance of Wireless local area networks (802.11 WLAN), currently referred as Wi-Fi [21]. Then, in 1997, the first original IEEE standard was released [22] and many researches have been conducted to study this standard and the next amendments that were introduced.

However, all the studies in the past used devices that were hardware-based, which is a considerable limitation. The lack of flexibility was very problematic, specially regarding hardware reconfigurations, because all signal operations were implemented in hardware, which made the reconfiguration of the devices limited to the Application Programming Interface (API) of the driver provided by the manufacturer. Furthermore, these devices were limited to their hardware capacities, and cannot be configured to perform tasks that go beyond their capabilities.

So, SDR emerged as a solution to the hardware problem. With SDR platforms, multiple stages of a radio system will no longer be implemented in hardware solutions, but instead some stages will be implemented in software, making the radios hardware flexible, easily configured and modified, and developed for multiple systems [23]. Besides that, GNURadio can provide all the necessary tools to implement wireless testbeds that are fully programmable in software at the MAC and PHY level. Thus, our goal is to use SDR and GNURadio to explore the limitations of the IEEE 802.11 standards, in order to change and adapt them to new wireless environments, such as underwater, maritime and underground communications.

RF communications suffer high attenuation underwater, especially for high frequencies, which limits the range of underwater standard IEEE 802.11 networks to a few centimeters. Therefore, the flexibility given by SDR, together with GNURadio platform and a SDR implementation of IEEE 802.11 standard may increase the communications range by using sub-GHz frequencies, where the attenuation of RF waves is reduced progressively.

1.3 Objectives

The goal of this dissertation is to validate the gr-ieee802.11 transceiver implementation on a SDR platform, by performing not only over-the-air measurements but also underwater experiments in freshwater and seawater environments. Therefore, a battery of interoperability tests with off-the-shelf equipment and between two SDR devices were performed at 2.4 GHz and 5 GHz, and also the gr-ieee802.11 implementation was optimized to operate in frequencies ranging from 70 MHz to 773 MHz, in order to increase the RF underwater range.

To reach this goal the following objectives were considered:

- Understand the concepts of SDR and learn how to use GNURadio as a software platform to implement the IEEE 802.11 standard;
- Study the existing IEEE 802.11 implementations for SDR in GNURadio, in order to understand their limitations and change the operating frequency;
- Study the RF propagation models for air and water;
- Implement a testbed for over-the-air, freshwater and seawater environments at different frequencies;
- Analyze and compare the obtained results against theoretical models and obtain conclusions about the performance of the implemented solution;

1.4 Results

The main contributions of this dissertation are summarized as following:

- **Validation of an IEEE 802.11a/g/p SDR transceiver implementation** - Multiple interoperability tests with commercial equipment and two USRP B210 boards were performed and the SDR implementation was validated.
- **Adaptation of the SDR implementation to operate at different frequencies** - The SDR implementation was adapted to operate at different frequencies, especially low frequencies, such as 700 MHz, 450 MHz and 100 MHz, in order to use the SDR implementation to explore new wireless environments, such as underwater and underground communications.
- **Experimental analysis of IEEE 802.11 networks at multiple frequencies** - Therefore, an underwater testbed was performed to validate the adaptation. Throughput, jitter and RTT were measured for the different operating frequencies and the obtained results with over-the-air experiences and underwater communications in a freshwater tank and a seawater environment were analyzed.

In our interoperability tests between a SDR and Wi-Fi cards, we achieved UDP throughputs up to 1.8 Mbit/s with an average delay lower than 1 ms. Jitter and throughput were higher when the SDR was the transmitter and the Wi-Fi was the receiver. SDR allows to successfully receive data rates in the order of 500 kbit/s in over-the-air communications between two SDR boards. Besides, RTT is higher than in traditional radios. It is quite acceptable considering that all signal processing is made locally on each computer. We achieved values lower than 40 ms for over-the-air transmissions.

Finally, experimental results in a large freshwater tank showed that was possible to achieve maximum UDP throughputs of 500 kbit/s at a distance of 3 meters, for a frequency of 200 MHz and a channel bandwidth of 5 MHz. The maximum range achieved for the use of 802.11 networks in freshwater environments was 5 meters with data rates in the order of 309 kbit/s, for a frequency

of 200 MHz, which exceeds in more than 200% the range achieved in [24], where a maximum range of 2.15 meters was achieved for 768 MHz. Besides, we achieved a maximum range of 1.8 meters in a seawater environment for a frequency of 70 MHz.

1.5 Structure

This document is organized in 6 Chapters. In Chapter 2 we describe the state of the art and all related work in this field. In Chapter 3 we describe the gr-ieee802.11a/g/p SDR Transceiver implementation that was evaluated in this dissertation and in Chapter 4 we detail the experimental planning and the testbed design.

Finally, Chapter 5 presents the obtained results in the multiple testbeds and Chapter 6 draws the main conclusions and points out the future work.

Chapter 2

State of the Art

In this chapter, we overview the SDR radio communication system. The multiple SDR platforms that are being used for research and development are presented, among with practical applications where such SDRs can be used. Since this thesis is focused on an SDR application for the IEEE 802.11 technology, the IEEE 802.11a/g/p standards are presented and some wireless communication projects based on SDR solutions are also addressed. Besides, some wireless communications projects based on SDR solutions are also addressed. The applicability and limitations of each project are studied, in order to compare them and to explain why the gr-ieee802.11 implementation by B. Bloessl et al.[12, 10] was chosen to be the subject of evaluation by this dissertation.

2.1 Software defined radio

The term "Software Defined Radio (SDR) was introduced in 1991 by Joseph Mitola III and in 1992 he published his first paper on this issue [25]. The IEEE have defined SDR as a "radio in which some or all of the physical layer functions are defined in software" [26].

The main idea of SDR is to perform all signal processing in software by using technologies, such as Field Programmable Gate Array (FPGA), General-Purpose Processor (GPP) and Digital Signal Processor (DSP) to implement all the software radio elements [4]. This characteristic of SDRs allows great flexibility and reconfigurable capacity, which make them suitable for research and development in multiple areas, in particular, wireless communications, because they allow that network protocols can be more easily tested and verified. In the past, there were radios with dedicated hardware such as Application Specific Integrated Circuits (ASIC) that didn't allow this flexibility. Besides, these traditional radios have increased maintenance costs, because normally they require new pieces of hardware while an SDR only requires some sort of software reconfiguration [27].

Figure 2.1 presents the fundamental architecture of the SDR. The digitalization work in SDR is performed by the Analog to Digital Converter (ADC) after the Radio Frequency (RF) front end circuit. The RF front end module converts the signal to the lower frequency called an Intermediate Frequency (IF). Then, ADC digitizes the signal and forwards it to the baseband processor, where

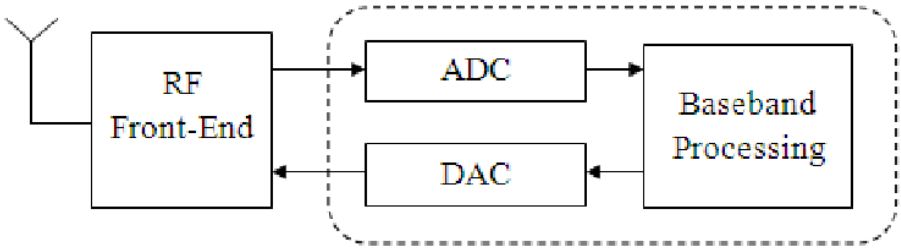


Figure 2.1: Software Defined Radio Block Diagram [1].

other signal operations are performed, such as demodulation and channel coding. In traditional radios, all this operations are performed in hardware [1].

In conclusion, SDR-based solutions are easier to implement, more flexible and reconfigurable, more efficient and cost effective than conventional radios.

2.1.1 SDR platforms

In this section, we overview some of the SDR platforms currently available for research and development. Then, we compare their characteristics in order to select the SDR device that is going to be used in this dissertation. Finally, we present the open-source reference tool for programming this devices, the GNURadio platform.

2.1.1.1 BladeRF

BladeRF is a low-cost SDR platform, designed to explore and experiment the RF communications. In [2], there is a lot of information and documentation about this SDR platform, being even provided some source code and easy tutorials in order to make the familiarization with the device more quickly. There are two versions of BladeRF, the X40 model that costs about 420$ and the X115 model, about 650$, which both can operate between 300MHz to 3.8GHz. More features of this SDR platform will be presented further in this section. In Figure 2.2 it is shown the BladeRF X40 Hardware kit, which is very similar to X115 model.

Figure 2.2: Blade RFX40 Hardware Kit [2].

2.1.1.2 HackRF One

Hack RF One is an open source SDR platform from Great Scott Gadgets [3] that is designed for research and development of modern radio technologies. In [3], there is many documentation about the device and also some source code and hardware designs files. HackRF One can operate between 30MHz to 6GHz and it only costs 328.00$, which make it an excellent SDR low cost platform. More HackRF One characteristics will be addressed further in this section. In Figure 2.3 it is shown the HackRF One SDR platform.

Figure 2.3: HackRF One SDR platform [3].

2.1.1.3 Ettus Universal Software Radio Peripheral

Universal Software Radio Peripheral (USRP) is a flexible low-cost open source SDR platform designed by Matt Ettus [6]. Typically, a USRP system consists of two main boards: the daughter board and the motherboard [1]. The daughterboard is functioning as the RF front-end of the SDR, while the motherboard consists of four ADC and DAC converters, and also has an FPGA that is responsible for some critical data processing operations. Furthermore, a USRP is connected to the host Personal Computer (PC) by using a Universal Serial Bus (USB) interface or a Gigabit Ethernet interface. In Figure 2.4 it is shown the USRP block diagram. USRP models and their characteristics will be presented further in this section.

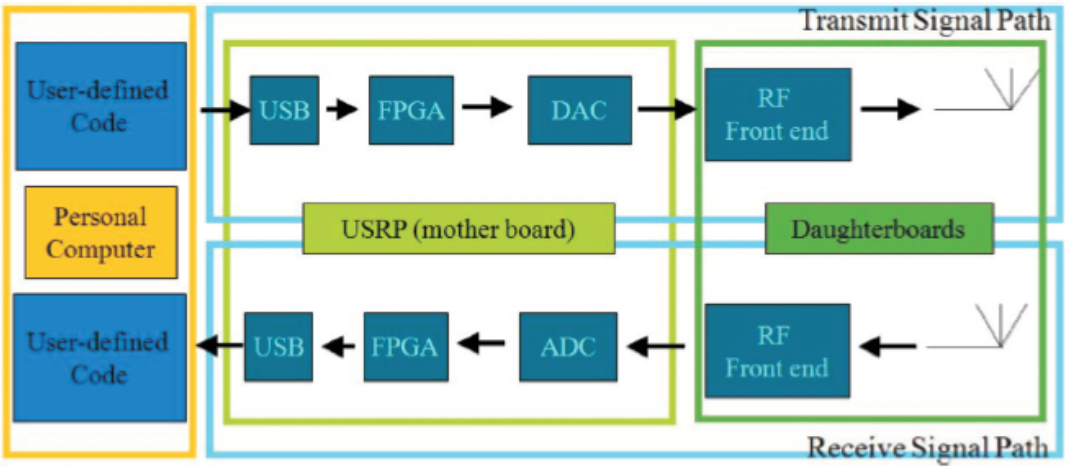


Figure 2.4: USRP Block Diagram [4].

2.1.1.4 National Instruments (NI) USRP

The USRP models by NI [5] are a flexible SDR device that makes the pairing of the USRP platform with revolutionary LabVIEW Communication software possible. This software offers a versatile design environment integrated with the USRP for rapidly prototyping communications systems. Besides, this combination provides an excellent solution for researchers who are getting started with communication systems design. The USRP transceivers allow the prototyping of a wide range of single-channel and Multiple-Input and Multiple-Output (MIMO) wireless communication systems [5]. The USRP models by NI will be more detailed further in this section. In Figure 2.5 it is shown the NI USRP platform combined with the LabVIEW Communication System Design Software.

Figure 2.5: USRP NI platform combined with the LabVIEW Communications System Design Software [5].

2.1.1.5 Comparison between SDR platforms

Table 2.1 presents some features of the USRP SDR devices designed by Ettus Research that currently are available for research and experimental work. On the other hand, Table 2.2 summarizes the characteristics of the HackRF One and BladeRF models. Besides, the Table 2.3 and the Table 2.4 show some features of the USRP SDR platforms designed by National Instruments.

First of all, there are many different USRP models developed by Ettus Research, which is the oldest SDR producer. However, their recent B200/B210 model is drastically different from all the previous models, specially because it is a SDR single board, instead of a daughter/motherboard solution. Besides, B210 model is the first board that can operate with 2x2 MIMO and supports USB 3.0 interface.

Table 2.1: Comparison between USRP SDR platforms designed by Ettus Research

SDR platform	USRP					
Model	X310	X300	N210	N200	B210	B200
Radio Spectrum	DC to 6GHz		DC to 6GHz		50Mhz-6GHz	
Duplex	Full		Full		2x2 MIMO	Full
Sample Size (ADC/DAC)	14/16 bit		14/16 bit		12 bit	
Sample Rate (ADC/DAC)	200/800 MS/s		100/400 MS/s		61.44 MS/s	
Host Sample Rate (16 bit)	200 MS/s		25 MS/s		61.44 MS/s	
Interface (Speed)	Dual 10 Gigabit Ethernet		Gigabit Ethernet		USB 3.0 (5 Gigabit)	
FPGA	Kintex 7-410T	Kintex 7-325T	Spartan 3A-DSP 3400	Spartan 3A-DSP 1800	Spartan 6 XC6SLX150	Spartan 6 XC6SLX75
Cost	4590.00€	3730.00€	1640.00€	1450.00€	1050.00€	645.00€

Secondly, most of the USRP models that support Gigabit Ethernet interfaces can operate between an enormous amount of the radio spectrum, which combined with their higher sample rates allows a big advantage in terms of data processing. However, their host sample rate is much inferior than their ADC/DAC rates, because the Gigabit Ethernet interface does not allow high sample rates when the data is transferred between the FPGA to a host PC. Besides, when comparing with other models, especially the ones from National Instruments, their high cost is a serious disadvantage.

Relatively, to the HackRF One and BladeRF models they are a much more economic solution than USRP platforms, but their features are much inferior when compared to this devices. Besides,

Table 2.2: Comparison between HackRF One and BladeRF SDR platforms

SDR platform	HackRF One	BladeRF	
Model		X40	X115
Radio Spectrum	30MHz-6GHz	300MHz-3.8GHz	
Duplex	Half	Full	
Sample Size (ADC/DAC)	8bit	12 bit	
Sample Rate (ADC/DAC)	20MS/s	40MS/s	
Host Sample Rate (16 bit)	8-20 MS/s	40 MS/s	
Interface (Speed)	USB 2.0	USB 3.0 (5Gigabit)	
FPGA	No FPGA	Altera Cyclone IV	
Cost	328.00$	420.00$	650.00$

Table 2.3: Comparison between USRP SDR platforms designed by National Instruments

SDR platform	USRP				
Model	2920	2921	2922	2930	2932
Radio Spectrum	50MHz–2.2GHz	2.4 – 2.5(GHz) 4.9 – 5.9(GHz)	400MHz–4.4GHz	50MHz – 2.2GHz	400MHz–4.4GHz
Duplex	Full	Half	Full		
Sample Size (ADC/DAC)	14/16 bit				
Sample Rate (ADC/DAC)	100/400 MS/s				
Host Sample Rate (16 bit)	50 MS/s				
Interface (Speed)	Gigabit Ethernet				
FPGA	Kintex 7				
Cost	2680.00€			3580.00€	

it should be noted that the HackRF One model does not support full duplex communication unlike all the other boards.

On the other hand, another important characteristic is the Sample Size of the ADC/DAC, which establish how much precise is the sample. Adding one bit to the Sample Size is actually doubling the precision of the sample. Another question about the ADCs and DACs is how fast they can process the samples, specially because the larger the number of samples, much more bandwidth they could process. Although some older technologies such as Frequency Modulation (FM) radio and GSM channels can use extremely slow ADCs and DACs, some new technologies require faster ADCs and DACs, especially IEEE 802.11a/g digital signals.

Another important issue to take into account about SDR is the transport of the data between the SDR platform to a PC, which is considered one of the most negative aspects of SDR technology, because of the latency that occurs in the communication between the SDR and the PC.

Table 2.4: Comparison between USRP RIO SDR platforms designed by National Instruments

SDR platform	USRP RIO					
Model	2940R	2942R	2943R	2950R	2952R	2953R
Radio Spectrum	50MHz–2.2GHz	400MHz–4.4GHz	1.2 - 4.4 (GHz)	50MHz–2.2GHz	400MHz–4.4GHz	1.2 - 6 (GHz)
Duplex	2x2 MIMO					
Sample Size (ADC/DAC)	14/16 bit					
Sample Rate (ADC/DAC)	120/400 MS/s					
Host Sample Rate (16 bit)	120 MS/s					
Interface (Speed)	High-speed,low-latency PCI Express x4					
FPGA	Kintex 7					
Cost	6020.00€			7260.00€		

However, the USB 3.0 interface might reduce this problem, because allows that more samples can be transmitted. The B210/B200 USRP board that is possible to observe in the Figure 2.6 supports this interface. However, there is a new interface that can mitigate this problem as well, which is the PCI Express interface. The USRP RIO SDR models, one of which being shown in Figure 2.7, support this interface.

Figure 2.6: USRP B210 board [6].

Figure 2.7: USRP-2940R model [5].

Regarding the FPGA, the BladeRF, the USRP B210/B200 models and the USRP SDR platforms from National Instruments are the ones with more computational power. The most significant advantage of an FPGA on SDR platforms is that all processing operations can be done in parallel, rather than serial.

In conclusion, this are the best currently available SDR platforms that can be used for experimental research. BladeRF has a very good FPGA and an USB 3.0 connection, which makes this board suitable for multiple applications. HackRF is a good platform, but in comparison with the other platforms has a lack of capacity, specially because does not have an FPGA and only supports the USB 2.0 interface, reducing the host sampling rate. On the other hand, USRP platforms are actually the most powerful SDR solutions, which allow USB 3.0 connections, support higher

sample rates and sizes, and operate in a larger amount of the radio spectrum. Finally, the NI USRP models are a very good SDR platform, but their discontinuities in the supported frequencies and their high cost, due to the software bundle added to the product, make this platform inadequate for this dissertation. Besides, the software bundle will not be used during this thesis.

2.1.2 SDR SDK - GNURadio

GNURadio [19] is a free open-source software development toolkit that was founded by Eric Blossom with the intention of providing all necessary signal processing blocks, in order to build and simulate SDR systems. These blocks can be classified as sources, operators, synchronizers, modulators, demodulators and filters, which can be connected with each other in order to implement full communication systems. Usually, communication blocks are written in C++, while the systems that define these connections are primarily written using Python [8].

Anoter important aspect of GNURadio platform is its graphical user interface (GUI), named GNU Radio Companion (GRC), which is a fast design tool where we can build functional SDR systems by connecting the signal processing blocks.

In GNURadio-based SDR platforms, RF front-end is responsible for implementing some signal operations at the transmitter, such as the digital-to-analog conversion, amplification and carrier modulation. However, RF front-end is also responsible for implementing signal operations at the receiver, especially carrier demodulation, amplification and analog-to-digital conversion [8].

In conclusion, GNURadio provides all the necessary tools to implement SDR-based solutions that are fully programmable in software at the PHY and MAC level.

2.1.3 SDR applications

In recent years, SDR platforms have been widely used in scientific areas. Therefore, many projects based on SDR technologies have been conducted. In this section, we present some of these applications, showing the power of SDR platforms in real world scenarios.

The OpenBTS project

The OpenBTS project [28] is an application that uses SDR to implement a GSM base transceiver station (BTS). OpenBTS was created for GNURadio platform, which allows simulating the behavior of a GSM BTS.

Although, it is a project mainly developed in software, certain hardware modifications are necessary on the SDR motherboard, in order to allow users to send text messages and make calls. This changes include the set up of a Session Initiation Protocol (SIP) server, the configuration of this server in order to provide connectivity between mobile devices, and the Voice over Internet Protocol (VoIP) backhaul [29].

With the OpenBTS project, a new type of cellular network can be developed and operated at considerable low cost than other existing technologies, due to the combination of the GSM air interface with low-cost VoIP backhaul forms [28].

The SDR receiver for Dedicated Short-Range Communications (DSRC)

The SDR receiver for DSRC was developed by P. F. Rito in his MSc dissertation at the University of Aveiro [30]. DSRC are short-range wireless communication channels, specially designed for vehicular networks. These type of communication is being widely used in Intelligent Transportation Systems (ITS).

The main goal of this dissertation was an implementation of a DSRC receiver in a SDR platform to be used in electronic charging tolls, in order to make the system more effective and with lower costs.

The IEEE 802.15.4 transceiver

The IEEE 802.15.4 Zigbee transceiver for GNU Radio v3.7 was designed and developed by B. Bloessl, C. Leitner, F. Dressler, and C. Sommer [31, 32]. The source code of their implementation is available at [33] so that other researchers can use this implementation on their experimental work.

This implementation, among other features, includes interoperability with TelosB sensor motes and Contiki, GNURadio blocks that implement all physical modulation and live packet tracer via Wireshark and Rime dissector.

The SDR for a Power Line Communication

In [34], a power line communication based on SDR between an electrical motor and a frequency converter was implemented and tested. Their testbed included a 90-meter-long motor power cable, a frequency converter and an electrical motor (2.2 kW). The results obtained showed that SDR and the GNURadio platform can be used to develop and experiment power line communication applications.

SDR for a total power radiometer implementation

This implementation is based on the M. E. Nelson dissertation, which used SDR and the GNURadio platform to implement a fully functional radiometer in software. Radiometers are highly sensitive receivers designed to measure the power of electromagnetic radiation [35]. The results obtained by this dissertation showed that using a SDR as a radiometer allows some considerable benefits, especially in terms of flexibility and cost.

2.2 IEEE 802.11

IEEE 802.11 standards are a set of MAC and PHY specifications that uses several modulation and coding schemes for implementing WLANs, commonly referred to as "Wi-Fi". The MAC layer implements listen-before-send techniques, such as Carrier Sense Multiple Access Collision Avoidance (CSMA/CA) in order to avoid collisions by waiting for a backoff interval before transmitting every frame [36].

The fundamental MAC access technique of the IEEE 802.11 is the distributed coordination function (DCF), which is a random access scheme based on the CSMA/CA. So, a station before

transmitting a frame, must sense the medium to determine if no other station is already transmitting. If the medium is continuously idle for Distributed Inter-Frame Space (DIFS) duration, then the station is allowed to transmit before that interval elapses. However, if the medium is found busy during the DIFS interval, the station must defer its transmission in order to avoid collisions. After deferral or before a station can attempt to transmit again, it should select a random backoff time interval and wait that time while the medium is sensed idle [37]. When the destination station receives a frame, a acknowledgement (ACK) is transmitted in order to signal the successful frame transmission. So, the ACK is transmitted following the received frame, after a Short Inter-Frame Space (SIFS). However, if the ACK is not received in a specified ACK timeout by the transmitting station or if a transmission of a different frame on the channel is detected, the transmitting station reschedules the frame transmission according to the backoff rules previously presented. The efficiency of CSMA/CA depends strongly on the hidden terminals problem, which compromises seriously the communications performance [38].

In order to mitigate collisions and the hidden node problem, the Request-to-send / Clear-to-send (RTS/CTS) mechanism can be implemented. The idea is that a sender station before transmitting a frame, begins by transmitting a RTS message and waits until it receives a CTS message. If the CTS message is received, it means that the medium is free and then the station can transmit the frame. The RTS/CTS mechanism is illustrated in Figure 2.8.

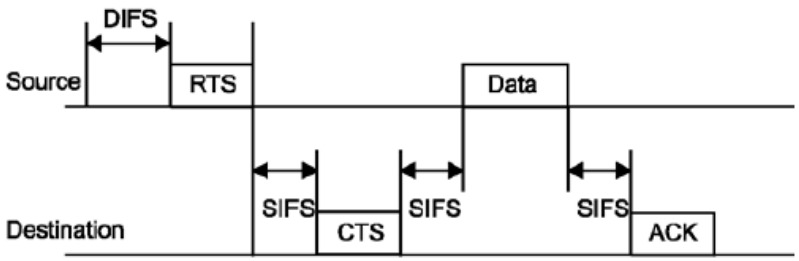


Figure 2.8: RTS/CTS mechanism [7].

In recent years, some amendments have been done to the original 802.11 standard that was released in 1997. These amendments are shown in Table 2.5 where some of their characteristics are presented.

2.2.1 IEEE 802.11a/g/p standards

In this section, we discuss in particular the three standards of the IEEE 802.11 that will be explored in this dissertation, namely 802.11a, 802.11g and 802.11p.

Table 2.5: IEEE 802.11 Standards

Standards	802.11	802.11b	802.11a	802.11g	802.11n	802.11p	802.11ad
Release Date	1997	1999	1999	2003	2009	2010	2012
RF Band (GHz)	2.4	2.4	5	2.4	2.4 or 5	5.470 - 5.925	60
Bandwith (MHz)	20	20	20	20	20 or 40	10	2160
Modulation	DSSS, FHSS	DSSS	OFDM	DSSS, OFDM	OFDM	OFDM	SC, OFDM
Maximum data rate (bit/s)	2 M	17 M	54 M	54 M	660 M	27 M (half-duplex)	6,76 G

2.2.1.1 IEEE 802.11a

IEEE 802.11a was the first amendment to the IEEE 802.11 standard that defined requirements for an orthogonal frequency-division multiplexing (OFDM) communication system. The OFDM scheme allows rates up to 54 Mbit/s operating within the 5GHz ISM band [39]. The use of the 5 GHz band for 802.11a is a significant advantage, because it allows high levels of performance, since, as the 2.4 GHz band is widely used it does not allow the achievement of this high levels of performance. Besides, the increased number of OFDM channels allows significant aggregate bandwidth and reliability advantages to the IEEE 802.11a, in comparison with other IEEE amendments, such as the IEEE 802.11b and the IEEE 802.11g. However, this made 802.11a Wi-Fi cards more expensive, which is a considerable disadvantage of this standard.

2.2.1.2 IEEE 802.11g

IEEE 802.11g is an amendment to the IEEE 802.11 standard for WLANs that allows transmissions over relatively short distances with rates up to 54 Mbit/s using the same 2.4 Ghz band as 802.11b. This extension uses the same OFDM modulation scheme used in 802.11a, which allows higher data speed [40]. Therefore, the 802.11g extension provided a number of improvements over its predecessor, the 802.11b standard.

2.2.1.3 IEEE 802.11p

IEEE 802.11p is an approved amendment to the IEEE 802.11 standard to provide wireless access in vehicular environments. In this environments, vehicular applications cannot tolerate long connection establishment delays before being allowed to communicate with other vehicles or roadside infrastructures [41]. The communication link between this applications might exist only for a short period of time, which made the previous IEEE 802.11 standards limited for this type of networks. So, the 802.11p defines enhancements to the 802.11 in order to support this applications.

The 802.11p standard operates in the licensed ITS band of 5.9GHz (5.85-5.925 GHz), divided into seven channels of 10 MHz each [36].

2.2.2 IEEE 802.11 implementations for SDR

In this section, some IEEE 802.11 implementations for SDR are presented. The design and the results obtained by this implementations were analyzed as well as their main problems and limitations.

2.2.2.1 BBN project

The BBN project is a partial implementation of IEEE 802.11b for SDR. This project was developed by BBN technologies, now a subsidiary of Raytheon, as a part of the ADROIT project [42]. The main goal of ADROIT's project is to build a cognitive wireless network, which can modify and reconfigure the radios in order to support network needs and channel conditions [42].

The goal of the BBN project was to build a radio for an USRP platform that could have been cognitively controlled. However, this implementation has some limitations. In [8], the use of GNURadio and SDR as a research tool was evaluated by experimenting the performance of the BBN 802.11b implementation. After a series of experiments at the bit error level, it was concluded that the BBN 802.11 implementation has some flaws, in particular the high bit error rate even under high signal-to-noise ratio (SNR) conditions. In Figure 2.9 it is shown the differences between the simulated and measured packet error rate.

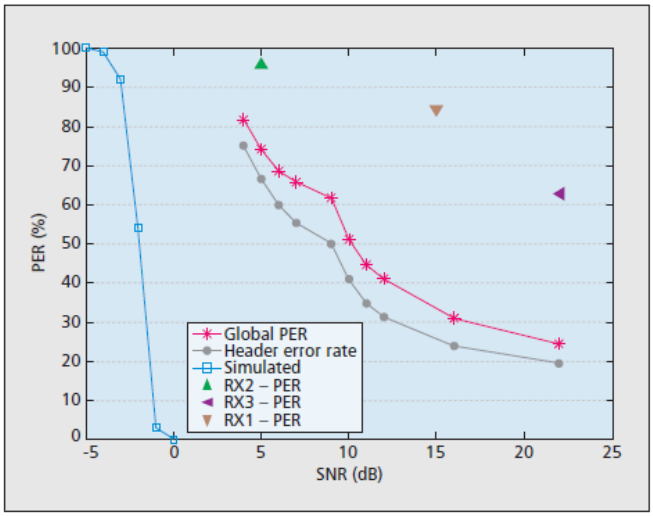


Figure 2.9: Packet error rate: simulated vs. measured [8].

Besides, it was observed that the synchronization is not robust, leaving a drift that increases packet loss probability [8].

In order to quantify this synchronization issues, it was measured the location of byte errors in the packet payload for different SNR values. The results obtained are shown in Figure 2.10, where the positive slope defines an increasing error probability with the byte location [8]. This drift increases packet loss probability, when the transmitted packets are longer.

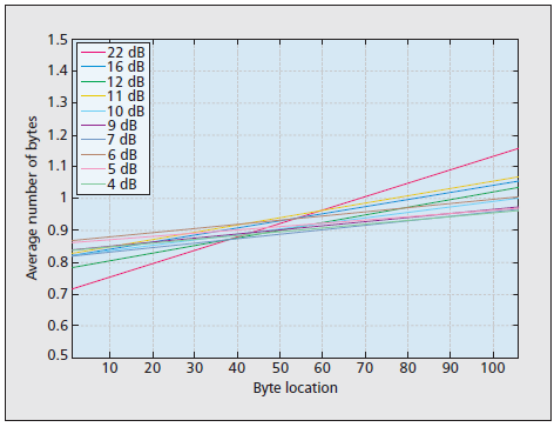


Figure 2.10: Average location of byte errors in packet payload for different received SNR values [8].

On the other hand, in Figure 2.11 it is shown the behavior of Bit Error Ratio (BER) between the decoded packets with a simulated BER for IEEE 802.11b. These results represent the performance of the SDR at the bit level, which are quite distinct. The BER specifications of the SDR are far from those obtained with commercial wireless cards and the simulated scenario.

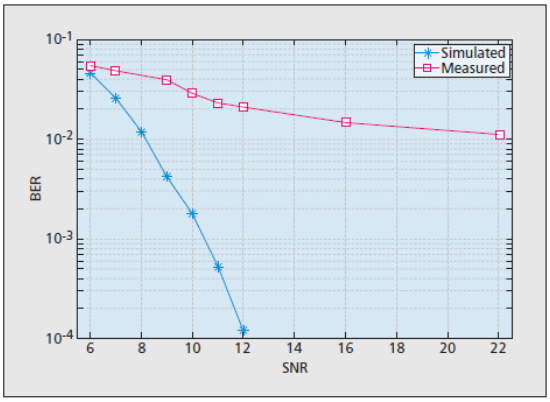


Figure 2.11: Bit error rate: simulated vs. measured [8].

In conclusion, these restrictions make the BBN 802.11b implementation more limited by reducing its capacity to overcome adverse conditions.

2.2.2.2 FTW 802.11p Encoder and Transmitter

The FTW 802.11p Encoder and Transmitter [9, 43] is an implementation of an IEEE 802.11a/g/p transmitter for SDR. This SDR application was designed by researchers at Forschungszentrum Telekommunikation Wien (Telecommunications Research Center of Vienna) and the University of Salento. Furthermore, this implementation is compatible with GNURadio platform and USRP SDR platforms.

They validated their implementation by using over-the-air measurements. The PC and the USRP were connected with a Gigabit Ethernet interface, and the USRP was equipped with a XCVR2450 daughter board, which allows operating frequencies ranging from 2.4 - 2.5 GHz band, and 4.9 - 5.9 GHz band.

The results shown in the Figures 2.12 and 2.13, demonstrate the differences in the transmit power-spectrum obtained between their implementation and the Atheros-based prototype chipset. With this results, it is shown that the GNURadio Spectrum contains peaks at 6MHz, which are not present in the Atheros spectrum. Furthermore, at the main band (4MHz) are attenuated. They concluded that this imperfections are due to the "roll-off characteristics of the interpolation filter in the up-conversion processing of the USRP2" [9]. However, this problem could be solved by doing part of the interpolation in the GNURadio encoder and by lowering the interpolation factor on the USRP2.

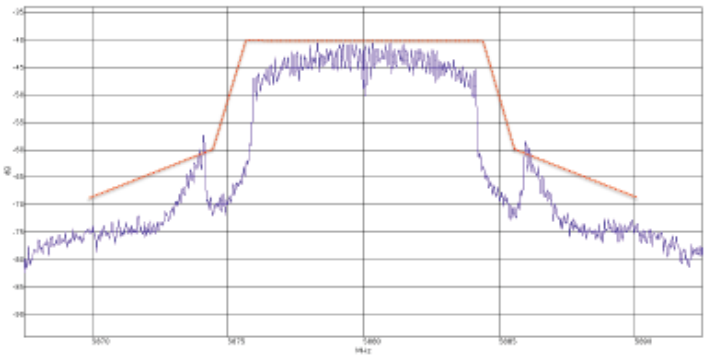


Figure 2.12: Power Spectrum of the GNURadio implementation, recorded with a second USRP2.The red line corresponds to the class A spectrum mask that is defined in the IEEE802.11p Draft 9.0 standard document [9].

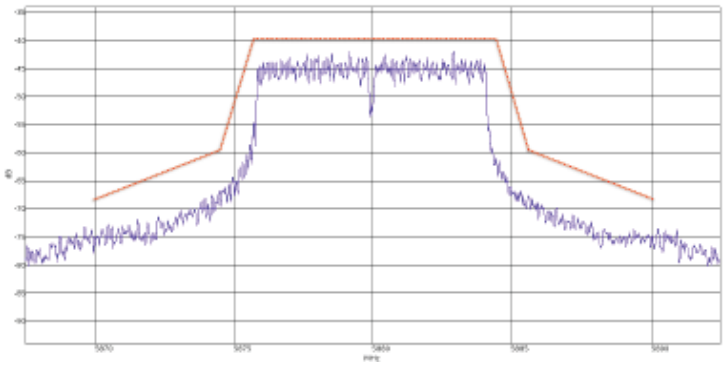


Figure 2.13: Power Spectrum of the Atheros-based prototype, recorded with the USRP2 [9].

Another comparison of the two transmitters is presented in the Figure 2.14, which represents the frame error ratio (FER) observed in the two implementations. As we can observe, the SDR

implementation performance in terms of FER vs. SNR is at max 0.5 - 1dB worse than the Atheros-based transmitter in the low-to-medium SNR region [9].

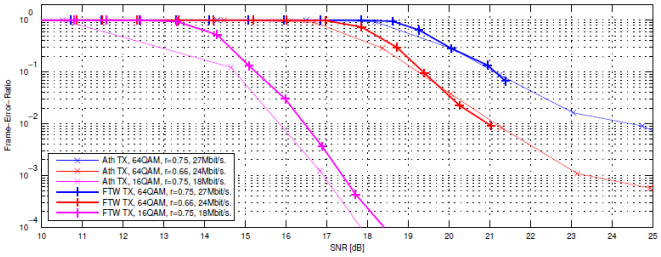


Figure 2.14: FER received using the two implementations [9].

In conclusion, this results indicate that this GNURadio SDR-based transmitter can generate frames compliant with the IEEE 802.11a, 802.11g and 802.11p standards.

2.2.2.3 gr-ieee802.11 IEEE 802.11a/g/p transceiver

A - Introduction

The IEEE 802.11a/g/p transceiver [44] was designed and developed by B. Bloessl et al., based on GNURadio v3.7. It is a SDR-based OFDM transceiver, specially for IEEE 802.11a/g/p networks. It should be highlighted that this implementation supports all modulations and coding schemes and runs completely in software on a PC without any reconfigurations on the FPGA. They tested this implementation by conducting a series of interoperability tests with the USRP N210 SDR platform that it is shown in the Figure 2.15.

Figure 2.15: USRP N210 SDR platform [6].

In relation to the GNURadio, the transceiver implementation is shown in the Figure 2.16, where the transmitter structure is in the top half of the figure and the receiver part in the bottom half. Their transmitter implementation has an important feature that is supporting variable packet sizes and multiple encodings. This part of the implementation includes mainly encoding, Fast Fourier transformation and addition of the cyclic prefix [10]. On the other hand, the receiver structure begins by calculating the autocorrelation coefficient in order to "detect the cyclic pattern

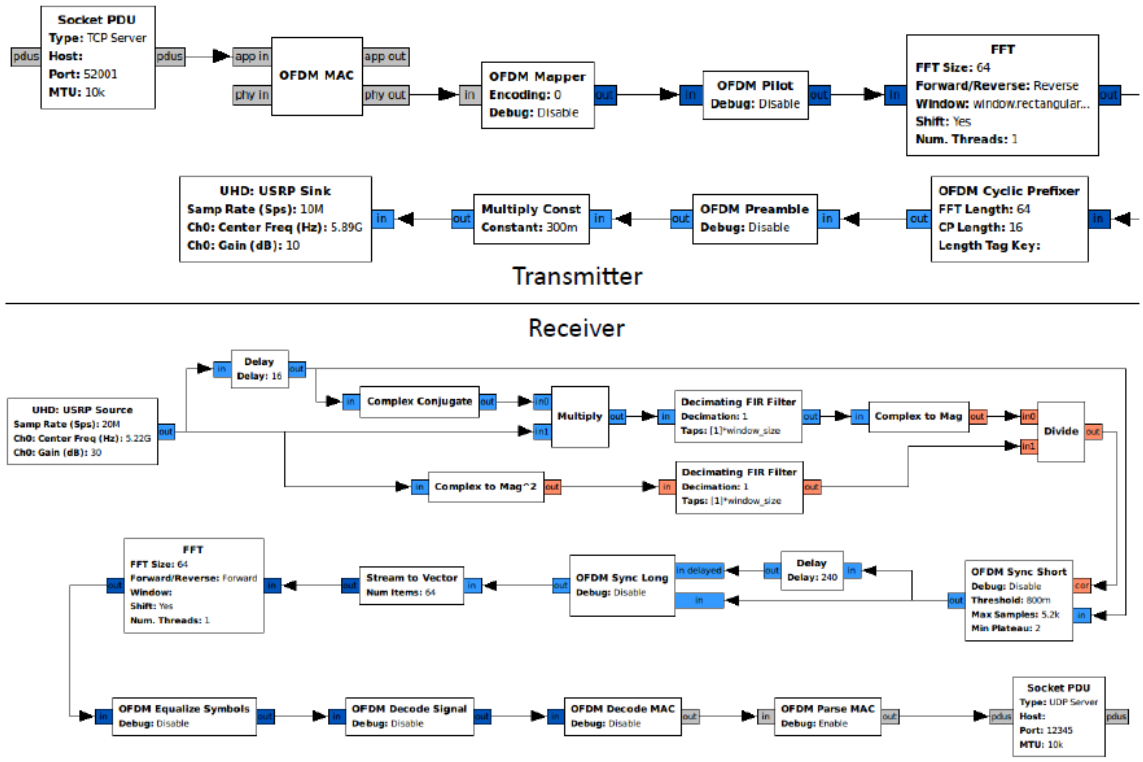


Figure 2.16: Overview of the transceiver structure in GNURadio Companion [10].

of the short preamble of OFDM frames, which is used for frame detection by the OFDM Sync Short block" [10]. Then, the following blocks are responsible for multiple tasks, such as frame alignment, channel estimation and frequency offset correction [10].

In [11], B. Bloessl et al., demonstrated the capabilities of the OFDM receiver by plotting the raw complex base band signal acquired from the USRP N210 in time domain, which is shown on the top right position in the Figure 2.17. Besides, in this Figure it is illustrated the constellation plot of Quadrature Phase Shift Keying-modulated (QPSK-modulated) symbols on the bottom right position and also the live packet visualization with Wireshark on the left position.

B - Implementation limitations

The transceiver implementation has some limitations, among them, in relation to the transmitter, the latency that occurs between the PC and the USRP N210, make it impossible to implement the carrier sensing logic in software, since "this introduces a large blind spot between the time the medium is sensed and when it is finally accessed" [10]. The only way to solve this problem is to implement the CSMA/CA on the FPGA, because the frames can be stored in a memory flash that most of the SDR platforms have. Thus, the channel access can be handled in hardware.

It should also be noted that the receiver experiences latency in the communication between the PC and the USRP N210, making it impossible the use of unicast transmissions due to the tough

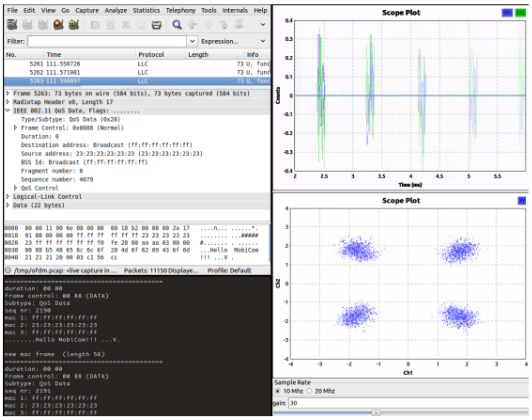


Figure 2.17: Screenshot of the live visualizations while the receiver is running: Packets in Wireshark (left), time domain signal (right), and constellation plot of (here) QPSK-modulated symbols (bottom right) [11].

timing constraints of RTS/CTS and ACK frames that cannot be fulfilled. However, there is a way to investigate unicast transmissions, which is by "disabling retries due to the missing ACKs and by setting the RTS/CTS threshold to infinity" [10]. Due to this limitation, in their testbeds they used broadcast transmissions to disseminate information.

C - Results obtained

In [12], the authors presented some results obtained with the receiver by representing them in Packet Delivery Ratio (PDR) curves. They investigated the performance of the receiver for different modulation and coding schemes. In the Figure 2.18 it is shown the results obtained for IEEE 802.11a packets and in the Figure 2.19 the results for IEEE 802.11p packets.

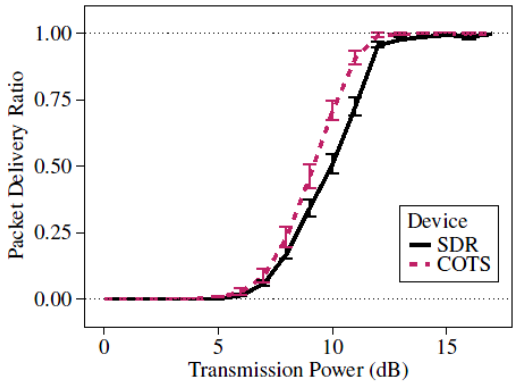


Figure 2.18: PDR of IEEE 802.11a packets, sent from a Unex device. The packet size is 95Byte, all packets are BPSK modulated with coding rate R=1/2 [12].

For the IEEE 802.11a measurements we can conclude that the performance of the receiver is comparable to consumer grade devices. In the IEEE 802.11p measurements, they used Binary

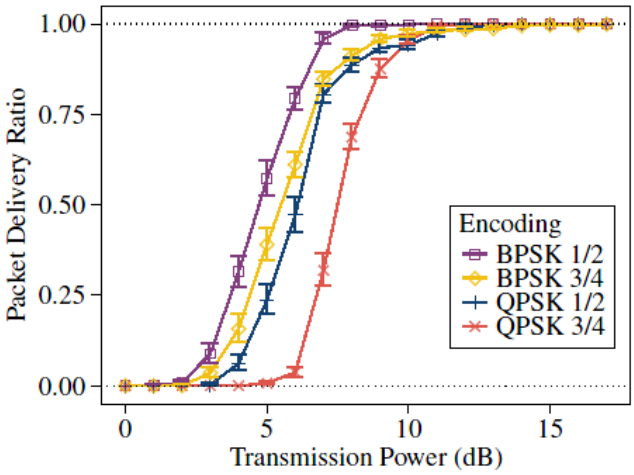


Figure 2.19: PDR of IEEE 802.11p packets, sent from a MK2 from Cohda Wireless. The packet size is 95Byte [12].

Phase Shift Keying (BPSK) and QPSK modulations, each with coding rates 1/2 and 3/4. As we can see, the four modulations are supported and the results obtained were quite reasonable, since higher bitrates suffer from higher packet loss as expected.

On the other hand, in [10] they presented the measurement results for the transmitter. The measurements were performed with IEEE 802.11p mode on the 5.86 GHz band, which is reserved for ITS applications. In the Figure 2.20 it is shown the error curves obtained by sending frames with the SDR and receive them with commercial Wi-Fi cards, while in the Figure 2.21 it is illustrated the error curves obtained by sending frames with commercial Wi-Fi cards and receive them with the SDR.

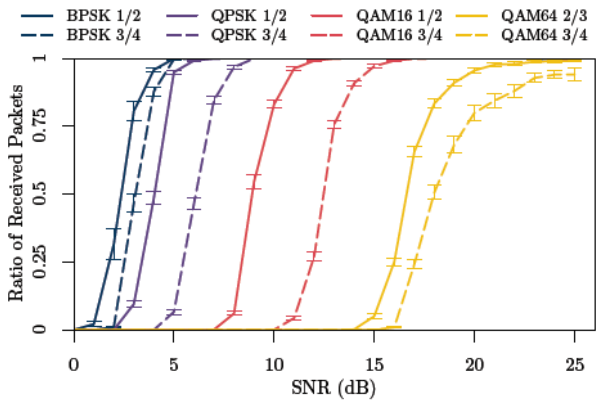


Figure 2.20: PDR of frames sent from the SDR and received with a commercial device. The devices are connected via cable and the packet size is 133 Byte [10].

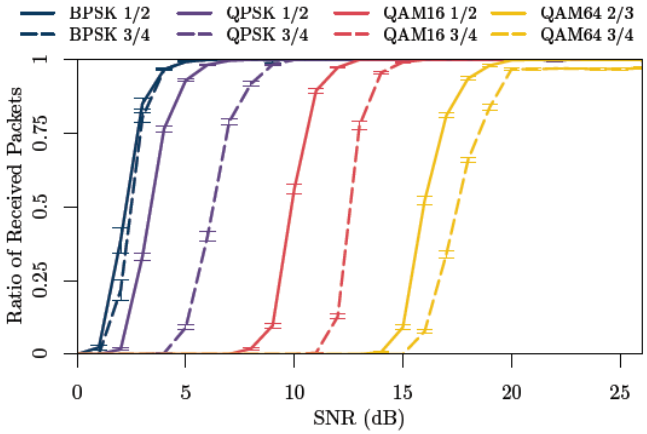


Figure 2.21: PDR for two commercial grade IEEE 802.11p devices. The devices are connected via cable and the packet size is 133 Byte [10].

Besides, they conducted simulations over an Additive White Gaussian Noise (AWGN) channel, in order to show that they achieved reasonable performance with their SDR implementation. With this simulation they determined packet delivery ratios that are shown in the Figure 2.22. The results obtained with the SDR and the commercial cards matched very closely, except for the 64 Quadrature Amplitude Modulation (QAM-64) 3/4 encoding, where they experienced worse performance with the SDR [10]. Thus, the results show that the IEEE 802.11a/g/p implementation for SDR has a performance that matches commercial cards, despite their considerable limitations that were previously presented.

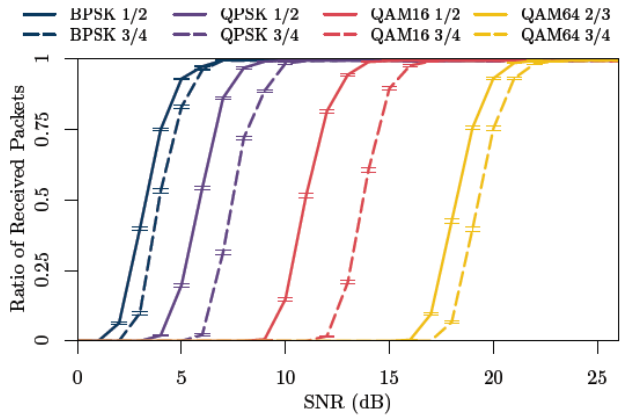


Figure 2.22: Simulative determined packet delivery ratio of 133 Byte sized packets over an AWGN channel [10].

In order to mitigate these limitations, they implemented the carrier sensing and the CSMA logic on the FPGA which can be verified in [45]. Thus, they programmed a CSMA state machine on the FPGA in order to control frame transmission. Only this two time critical functions where

implemented in hardware and all the other remaining processing operations where implemented in software. Therefore, the implementation follows the split functionality approach, where the main idea is to implement time critical functionality in hardware in order to achieve deterministic timing, while the non time critical functions are implemented in software to achieve higher flexibility.

Finally, it should be noted that the authors have made the source code of their implementation available at [46] so that others can test and experiment it.

2.3 Propagation Models

Our goal is to use the SDR and the GNURadio to overcome the restrictions of the IEEE 802.11 standards, in order to change and adapt them to new wireless environments. Thus, as the RF waves propagate differently in different environments, we present in this section the RF propagation models for the air and for the underwater scenario.

In the air, the loss in signal strength of an electromagnetic wave that would result from a line-of-sight path through free space, without obstacles nearby to cause reflection or diffraction is defined as the Free Space Path Loss. The Free Space Path Loss is described by the Equation 2.1.

$$FSPL(dB) = 10log_{10}\left(\frac{4\pi}{c}df\right)^2 \ \ (dB) \tag{2.1}$$

Regarding an underwater environment, the RF waves propagate slower in the water, since it contains dissolved salts and other matter, which makes it a partial conductor. The attenuation of radio signals is greater when the water's conductivity is higher [47]. Sea water has an average conductivity of 4 Siemens/meter (S/m), because it has high salt content. On the other hand, for freshwater the value of the water conductivity is 0.01 S/m [48, 47]. The propagation constant of RF waves is given by Equation 2.2 [49],

$$\gamma = \sqrt{j\omega\mu(\sigma + j\omega\varepsilon)} = \alpha + j\beta \ \ (m^{-1}) \tag{2.2}$$

where $\mu = \mu_r.\mu_0$ is the permeability of the medium in N/A^2, σ the conductivity of water in S/m and $\varepsilon = \varepsilon_r.\varepsilon_0$ the permittivity of the medium in F/m. Besides, ε_0 is the dielectric permittivity of free space in F/m and the propagation constant γ is a complex quantity composed by α, the attenuation factor and β, the phase factor. The two factors are described by Equation 2.3 and Equation 2.4, respectively [50].

$$\alpha = \omega\sqrt{\mu\varepsilon)}\left[\frac{1}{2}\left(\sqrt{1+(\frac{\sigma}{\omega\varepsilon})^2}-1\right)\right]^{\frac{1}{2}} \ \ (Np/m) \tag{2.3}$$

$$\beta = \omega\sqrt{\mu\varepsilon}\left[\frac{1}{2}\left(\sqrt{1+(\frac{\sigma}{\omega\varepsilon})^2}+1\right)\right]^{\frac{1}{2}} \quad (rad/m) \tag{2.4}$$

The real part of the permittivity ε_r is dependent of the complex frequency, and is commonly described with the Debye model, according to Equation 2.5 [50]:

$$\varepsilon_r = \varepsilon_\infty + \left[\frac{\varepsilon_s - \varepsilon_\infty}{1+\left(j\frac{f}{f_{ref}}\right)}\right] \tag{2.5}$$

where ε_s and ε_∞ are the real relative permittivity at low and high frequencies in F/m, respectively, and f_{ref} is the relaxation frequency in Hz [24].

Regarding the propagation speed of the RF waves, it is directly proportional to the frequency [51, 52] and can be calculated using Equation 2.6.

$$v_p = \frac{\omega}{\beta} \quad (m/s) \tag{2.6}$$

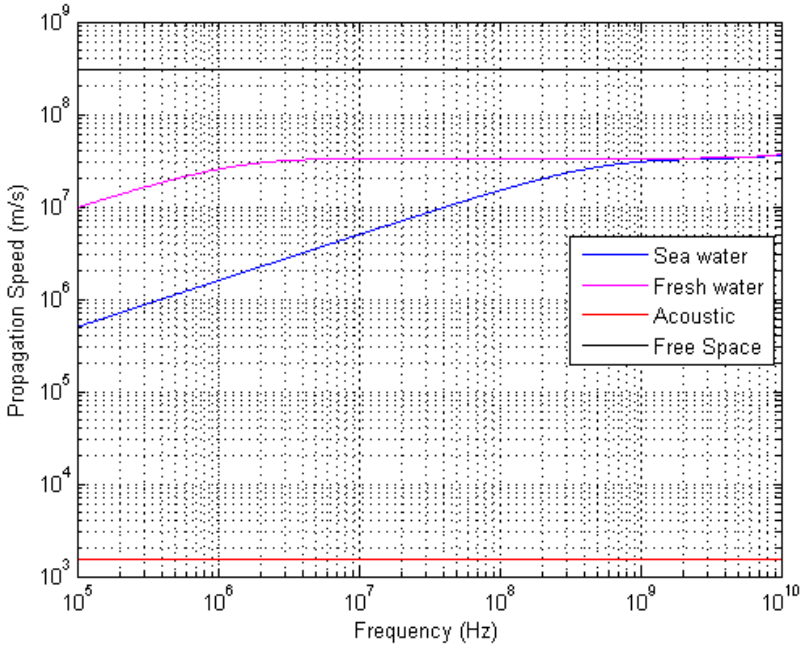


Figure 2.23: Propagation speed for RF waves and Acoustic waves.

Figure 2.23 shows a comparison between the propagation of RF waves underwater for both fresh and sea water, the propagation speed of RF waves in air, and the propagation of acoustic waves underwater. With this figure, we can conclude that although RF waves propagate slower

in water environments, as mentioned before, they still propagate much faster than acoustic waves, even when considering frequencies below 1 MHz [51]. This characteristic is important for low delay applications.

Another important aspect is the wavelength (λ) of the RF wave, which is given by Equation 2.7:

$$\lambda = \frac{2\pi}{\beta} \ (m) \tag{2.7}$$

In the figure 2.24, it is shown the wavelength vs frequency in sea water and fresh water. As the propagation speed is lower underwater, the wavelength is also lower. This effect leads to differences in the development and use of antennas for different environments, such as underwater and terrestrial communications [47].

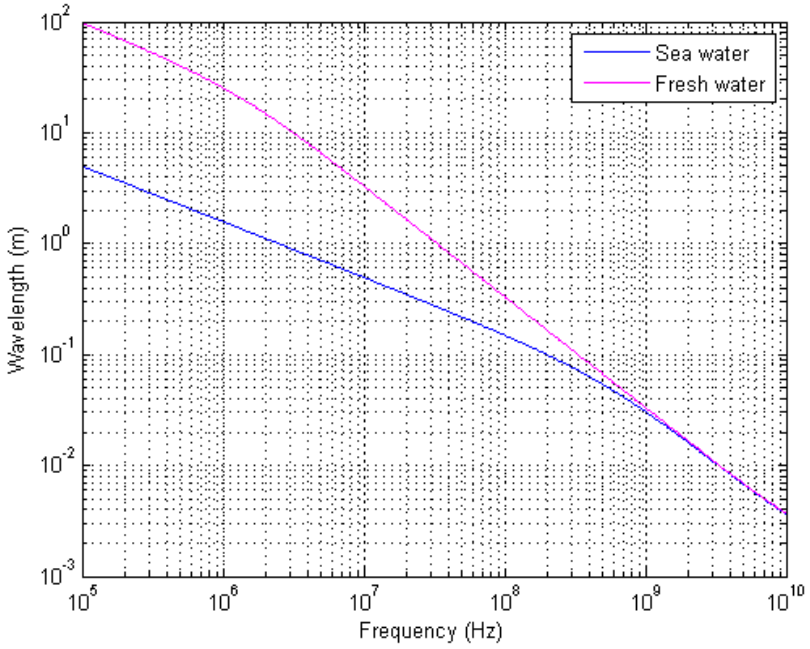


Figure 2.24: RF Wavelength vs. Frequency in Sea Water and Fresh Water.

Another important aspect is the signal power attenuation, because there are differences between over-the-air and underwater communications. In Figure 2.25 it is shown the attenuation of RF waves underwater for sea water and fresh water. This figure shows that the attenuation of RF waves in underwater is extremely high, especially for high frequencies. Therefore, we can conclude that there is a reduced range for underwater RF communications. The signal power attenuation is the α factor of the propagation constant γ, both described above, respectively, in Equation 2.3 and 2.2.

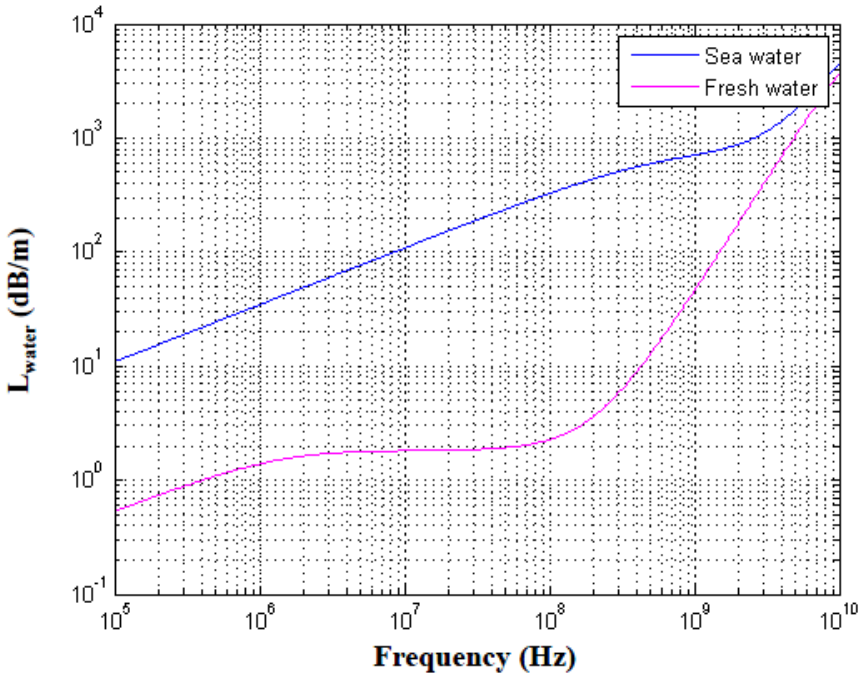


Figure 2.25: Attenuation of RF waves underwater.

Received power is another important parameter, which can be described as a function of transmitted signal, path loss and antenna gain at the receiver end by the Equation 2.8,

$$P_{rec}(dBm) = P_t + G_t + G_r - L_{pathloss} - L_{water} \tag{2.8}$$

where P_t is the transmitted power in dBm, G_t and G_r are the gains of the transmitter and the receiver, respectively, in dBi, $L_{pathloss}$ is the free space path loss, and L_{water} is the attenuation of the RF waves in the water, which depends on the operating frequency and the conductivity of the water [24].

Finally, the Table 2.6 presents the values of the attenuation and the propagation speed for the 768 MHz, 2.462 GHz and 5.240GHz frequencies. Besides, the SDR will allow the use of even lower frequencies, reducing the attenuation values and increasing communication range.

Table 2.6: Propagation Speed and Attenuation for 768 MHz, 2.462 GHz and 5.240 GHz frequencies in Fresh Water (σ = 0.01 *S/m*)

Frequency	768 MHz	2.462GHz	5.240GHz
Propagation Speed (m/s)	3.33×10^7	3.35×10^7	3.43×10^7
Attenuation (db/m)	28	269	1161

2.4 Summary

In this chapter, the state of the art of the SDR technology was presented. In recent years, this technology have drastically changed the way that the research community has been working in their experimental research, especially in the wireless communications area.

SDR platforms currently available for research and development were addressed in the Section 2.1.1. For this dissertation, in particular, it was decided to use the USRP as a SDR platform, especially the USRP B210 model, due to their superior features in comparison with the other USRP models and the other SDR platforms, namely the higher host sampling bandwidth and wider operating frequency range. The GNURadio SDK platform was also introduced, and will be used in order to implement all the necessary signal operations in software. In Section 2.1.3 some SDR applications were presented to demonstrate that SDR and GNURadio SDK platform can build powerful solutions that are capable of competing with commercial hardware based solutions.

In this dissertation, we intend to evaluate the performance of a SDR implementation and adapt it to new application scenarios. To do that, the IEEE 802.11 standard was explained in Section 2.2 together with their respective amendments. Besides, the RF propagation models for the air and for the underwater scenario were addressed in the Section 2.3.

Finally, some IEEE 802.11 SDR implementations were presented. For this thesis,in particular, it was decided to evaluate the gr-ieee802.11 IEEE 802.11a/g/p transceiver implementation introduced in the Section 2.2.2.3, specially because the results obtained were quite promising, despite its considerable limitations.

Chapter 3

Overview of the gr-ieee802.11 Structure

In this chapter, we present a description of the structure of the gr-ieee802.11 transceiver implementation. First of all, we detail the structure of the OFDM receiver and then the structure of the OFDM transmitter, both constituent parts of the transceiver implementation developed by B. Bloessl et al.

3.1 GNURadio OFDM Receiver

The structure of the OFDM receiver is illustrated in the Figure 3.1. The receiver is divided into three parts: the part on the top is responsible for frame detection, the middle part is responsible for frame decoding and the bottom part is the user application part. In order to understand how these steps are performed it is important to know the composition of a IEEE 802.11 frame and in particular of the OFDM physical layer convergence protocol (PLCP) preamble. In the following, we discuss some specific aspects about the IEEE 802.11 PHY, before explaining the signal processing blocks of the OFDM receiver in detail.

3.1.1 IEEE 802.11 PHY layer

In October 1997 the IEEE 802 Executive Committee approved two projects for higher rate PHY extensions to IEEE 802.11. The first extension, IEEE 802.11a, defines requirements for a PHY operating in the 5.0 GHz U-NII frequency and data rates ranging from 6 Mbps to 54 Mbps. The second one, IEEE 802.11b, defines a set of PHY specifications operating in the 2.4 GHz ISM frequency band up to 11 Mbps. Both PHY are defined to operate with the existing MAC layer [13].

The PHY defines the means of transmitting raw bits rather than logical data packets over a physical link connecting network nodes. It is the interface between the MAC layer and wireless media. There is three levels of functionality in the PHY: it provides a frame exchange between the MAC and PHY under the control of the PLCP sublayer; it uses signal carrier and spread spectrum modulation to transmit data frames over the media under the control of the physical

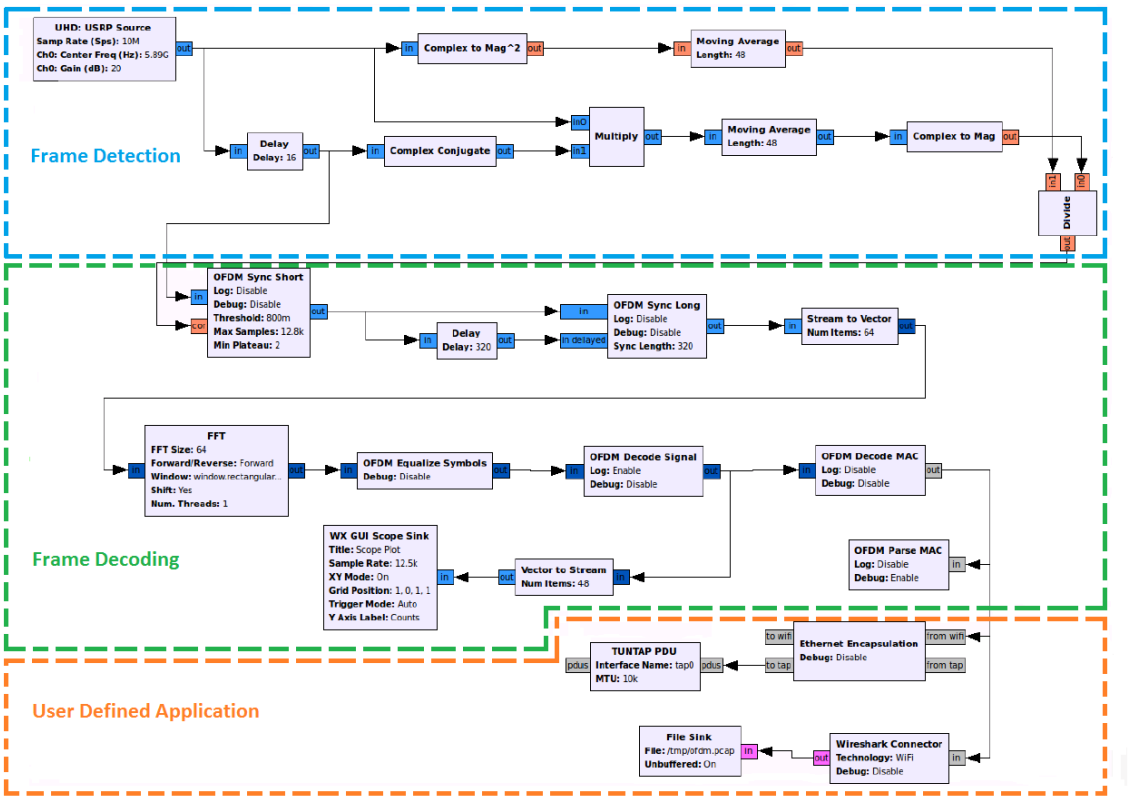


Figure 3.1: Overview of the OFDM receiver flow graph in GNURadio Companion.

medium dependent (PMD) sublayer and the PHY provides a carrier sense indication back to the MAC layer to verify activity on the media [53].

IEEE 802.11a - The OFDM PHY

The gr-ieee802.11 transceiver implementation was developed based on the first extension, the IEEE 802.11a OFDM PHY. The IEEE 802.11a PHY adopts OFDM PHY, which provides the capability to transmit physical layer service data unit (PSDU) frames at multiple data rates up to 54 Mbps for WLAN networks, where transmission of multimedia content is a consideration [13]. The PPDU is unique to the OFDM PHY, namely to the OFDM PLCP Sublayer illustrated in the Figure 3.2. The PPDU frame consists of a PLCP preamble and signal and data fields as shown in the Figure 3.3.

The PLCP preamble is used to acquire the incoming signal and train, and synchronize the receiver. It consists of 12 symbols, ten of which are short symbols, and two long symbols. Twelve subcarriers are used for the short symbols and 53 for the long. The training of an OFDM is accomplished in 16 seconds. The PLCP preamble is BPSK-OFDM modulated at 6 Mbps.

The signal is a 24-bit field, which contains information about the rate and length of the PSDU. The Signal field is convolutional encoded rate $\frac{1}{2}$, BPSK-OFDM modulated. Four bits (R1 - R4) are used to encode the rate, eleven bits are defined for the length, one reserved bit, a parity bit, and

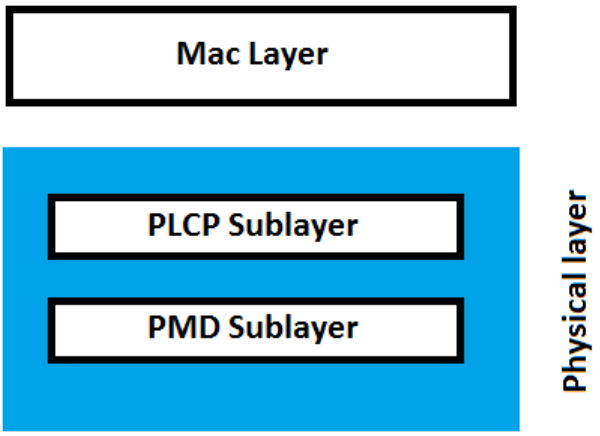


Figure 3.2: The sublayers of the PHY.

six 0 tail bits. The length field is an unsigned 12-bit integer that indicates the number of octets in the PSDU.

On the other hand, the data field contains the service field, PSDU, tails bits, and pad bits. A total of six tail bits containing 0s are appended to the PPDU to ensure that the convolutional encoder is brought back to zero state.

3.1.2 Frame Detection

The first task of the receiver is to detect the start of an OFDM frame based on the frame detection algorithm that has been introduced in [54]. This algorithm is based on the autocorrelation of the

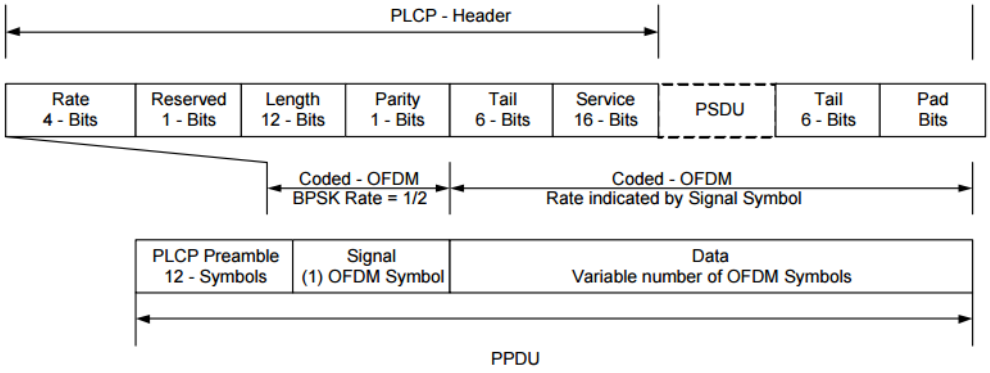


Figure 3.3: OFDM PLCP Preamble, Header, and PSDU [13].

short training sequence. Each IEEE 802.11a/g/p frame starts with a short preamble sequence, which consists of a pattern that spans 16 samples and repeats 10 times. The receiver exploits this cyclic pattern and calculates the autocorrelation value a of the incoming sample stream s with lag 16 by summing up the autocorrelation coefficients over an adjustable window N_{win} [12]:

$$a[n] = \sum_{k=0}^{N_{win}-1} s[n+k]\bar{s}[n+k+16] \tag{3.1}$$

The authors experimented different values for the N_{win} and decided that 48 was a good value. The autocorrelation is high at the start of an IEEE 802.11a/g/p frame due to this cyclic property of the short training sequence. Besides, the authors decided to normalize the autocorrelation with the average power p, so that the receiver was independent of the absolute level of incoming samples. The autocorrelation coefficient c is calculated as follows:

$$p[n] = \sum_{k=0}^{N_{win}-1} s[n+k]\bar{s}[n+k] \tag{3.2}$$

$$c[n] = \frac{|a[n]|}{p[n]} \tag{3.3}$$

In the receiver they considered that there is a plateau if three consecutive samples are over a configurable threshold. In the Figure 3.4, it is shown the plateau of high autocorrelation coefficients during the short training sequence [12]. If a frame is detected, a fixed number of samples is then piped to the subsequent blocks.

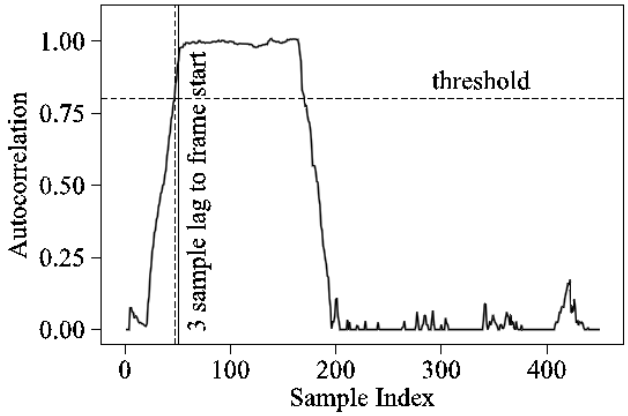


Figure 3.4: Characteristic behavior of the autocorrelation function during frame reception [12].

All the eight blocks involved in the frame detection make use of the Vectorized Library of kernels (VOLK) Library, which allow the implementation to support sample rates of 20 Msps for IEEE 802.11a/g and 10 Msps for IEEE 802.11p. Thus, the increased processing speed with the use of this library is crucial for the receiver, since all the blocks involved have to process the sample stream from the USRP at full speed [12].

After the calculation of the autocorrelation coefficient, the next block is the *OFDM sync short*, whose inputs are the samples from the USRP and the normalized autocorrelation coefficient. If a plateau is detected, the block pipes a fixed number of samples into the subsequent blocks of the flow graph. However, if the plateau isn't detected, the frames are dropped. Nevertheless, this approach has some limitations, such as the size of the frames that can be decoded, because the size is limited to the number of OFDM symbols. Besides, if a frame is received shortly after other, this frame will not be detected. In order to surpass the size limitation, the authors decided to set the maximum number of samples that are streamed in to the next blocks according to the maximum number of OFDM symbols per frame [12].

3.1.3 Frequency Offset Correction

The next block in the receiver implementation is the *OFDM Sync Long*, which is responsible for frequency offset correction and symbol alignment. Frequency offset correction is needed, since the local oscillators of sender and receiver might work on slightly different frequencies. In order to compensate that, the authors used the algorithm introduced in [55], which estimates the frequency offset correction based on the short training sequence. Ideally, during the short sequence a sample $s[n]$ should correspond to the sample $s[n+16]$ due to its cyclic property [12]. However, if noise and a frequency offset are introduced, this is no longer the case, and $s[n]\bar{s}[n+16]$ is not a real number, as in the idealized case [12]. The final value for the frequency offset $\mathrm{d}f$ is then calculated by the Equation 3.4:

$$df = \frac{1}{16} arg\left(\sum_{n=0}^{N_{short}-1-16} s[n]\bar{s}[n+16]\right), \tag{3.4}$$

where N_{short} is the length of the short training sequence. The frequency offset is then applied to each sample as

$$s[n] \leftarrow s[n]e^{i(ndf)}. \tag{3.5}$$

3.1.4 Symbol Alignment

The *OFDM Sync Long* block is also responsible for symbol alignment. Each OFDM symbol spans 80 samples, consisting of 16 samples of cyclic prefix and 64 data samples. The process of symbol alignment consists in the calculation of the start of a symbol, the extraction of the data symbols,

and feeding them to an algorithm doing a Fast Fourier Transformation (FFT) [12]. Besides, the alignment is done with the help of the long training sequence, which is composed of a 64 sample long pattern that repeats 2.5 times. In the Figure 3.5, it is shown the correlation of the input stream with the known sequence.

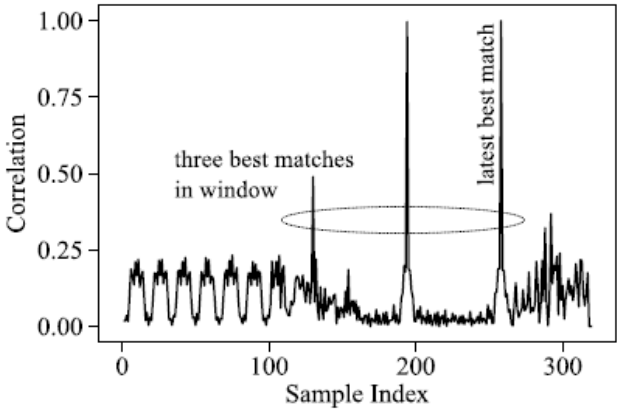


Figure 3.5: Characteristic behavior of the correlation of the input stream with the known sequence calculated in the *OFDM Sync Long* block [12].

The indices of the highest three peaks are calculated as

$$N_{\mathcal{P}} = \underset{n \in \{0,\ldots,N_{\text{preamble}}\}}{\arg\max_3} \sum_{k=0}^{63} \mathrm{s}[n+k]\,\overline{\mathrm{LT}}[k], \tag{3.6}$$

where argmax_3 returns the top 3 indices maximizing the expression, $N_{preamble}$ corresponds to the added length of the short and long preambles and LT is the repeating pattern of the long training sequence spanning 64 samples [12].

The first data symbol starts at sample index

$$n_{\mathcal{P}} = \max\left(N_{\mathcal{P}}\right) + 64, \tag{3.7}$$

as the latest peak of the matched filter output is 64 samples before the end of the long training sequence [12]. Finally, by knowing the start of the data symbols, we can remove the cyclic prefix by subsetting the data stream and grouping the samples that correspond to individual data symbols as

$$\mathrm{s} \leftarrow \Big(\underbrace{\mathrm{s}[n_{\mathcal{P}}+16],\ldots,\mathrm{s}[n_{\mathcal{P}}+79]}_{\text{first symbol}},\underbrace{\mathrm{s}[n_{\mathcal{P}}+80+16],\ldots}_{\text{second symbol}}\Big). \tag{3.8}$$

3.1.5 Phase Offset Correction

The next block in the receiver chain is the FFT block, which is responsible for the transition from time to frequency domain. The first block in the frequency domain is the *OFDM Equalize Symbols*, which applies phase offset correction and channel estimation [12]. As the symbol alignment is not perfect, a phase offset is introduced, which is linear with frequency and can be corrected with the help of pilot subcarriers. IEEE 802.11 mandates four pilot subcarriers that encode a predefined BPSK constellation, which is the same for each frame, but different from symbol to symbol. Thus, it is required to know the symbol index of the frame, which will be signed by a tag in the sample stream that is added by the *OFDM Equalize Symbols* block [12]. The phase offset is then estimated by a linear regression and compensated based on the four pilot subcarriers.

3.1.6 Channel Estimation

As it was mentioned before, the *OFDM Equalize Symbols* block is also responsible for channel estimation, since the magnitude of the carriers has to be corrected. This estimation is especially important if higher encodings are used, such as QAM-16 and QAM-64, where the magnitude carries information [12].

The *OFDM Equalize Symbols* block also removes Direct Current (DC), guard and pilot subcarriers and thus subsets the 64 symbol input vector into 48 symbols [12].

3.1.7 Signal Field Decoding

The next block in the receiver chain is the *OFDM Decode Sygnal*. In each frame, after the short and long training sequences, there is the signal field, which is a BPSK modulated OFDM symbol encoded with a rate of $\frac{1}{2}$ that carries information about the encoding and length of the next symbols [12]. If the signal field is decoded successfully, i.e., if the rate field contains a valid value and if the parity bit is correct, this block annotates the sample stream with a tag, carrying a tuple of encoding and length of the frame [12]. This tag is then used by the following block to decode the payload.

3.1.8 Frame Decoding

The final task in the receiver chain is the decoding of the actual payload. This task is performed by multiple blocks. The *OFDM Decode MAC* block receives vectors of 48 constellation points in the complex plane, corresponding to the 48 data subcarriers per OFDM symbol [12]. Then, depending on the Modulation and Coding Scheme (MCS), the bits of a symbol can be permuted.

The final step in the decoding process is descrambling. Then, the payload is packed into a GNURadio message and passed to the following blocks in the flow graph [12].

3.1.9 User Defined App

The next blocks are Ethernet Encapsulation, TUNTAP PDU, and the *Wireshark Connector* block, which was created by the authors. All these blocks can be used to monitor the frames in wireshark in real time, from which we can see the detailed information of each frame.

3.2 GNURadio OFDM Transmitter

The structure of the OFDM transmitter is illustrated in the Figure 3.6. Before the authors implemented their transmitter, there has already been presented another IEEE 802.11p transmitter for the GNURadio [9, 43], which is described in more detail in the Section 2.2.2.2. They decided to implement their transmitter because this one was implemented in an older version of GNURadio and partly implemented in Python. Besides, the transmitter developed by B. Bloessl et al. supports variable packet sizes and allows to specify the encoding on a per packet basis [10].

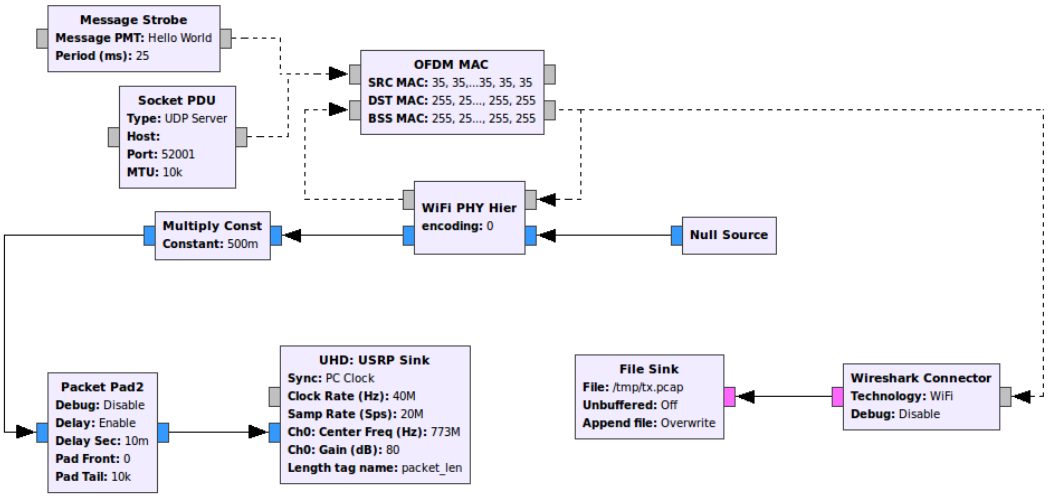


Figure 3.6: Overview of the OFDM transmitter flow graph in GNURadio Companion.

In order to understand the structure of the transmitter, it is also important to overview an essential part of it, the *WiFi PHY Hier* block, which represents the PHY. The PHY is encapsulated in a hierarchical block allowing for a clearer structure of the transmitter and the transceiver in GNU Radio Companion. The structure of the *WiFi PHY Hier* block is implemented in another flow graph in GNURadio Companion, which is illustrated in the Figure 3.7. The *OFDM Mapper* is the first block in the *WiFi PHY Hier* block, which receives the MCS as input and is responsible for multiple operations, such as the generation of the data field, in which it is included the tail and pad bits. Besides, it is also responsible for the scrambling and interleaving of the bits.

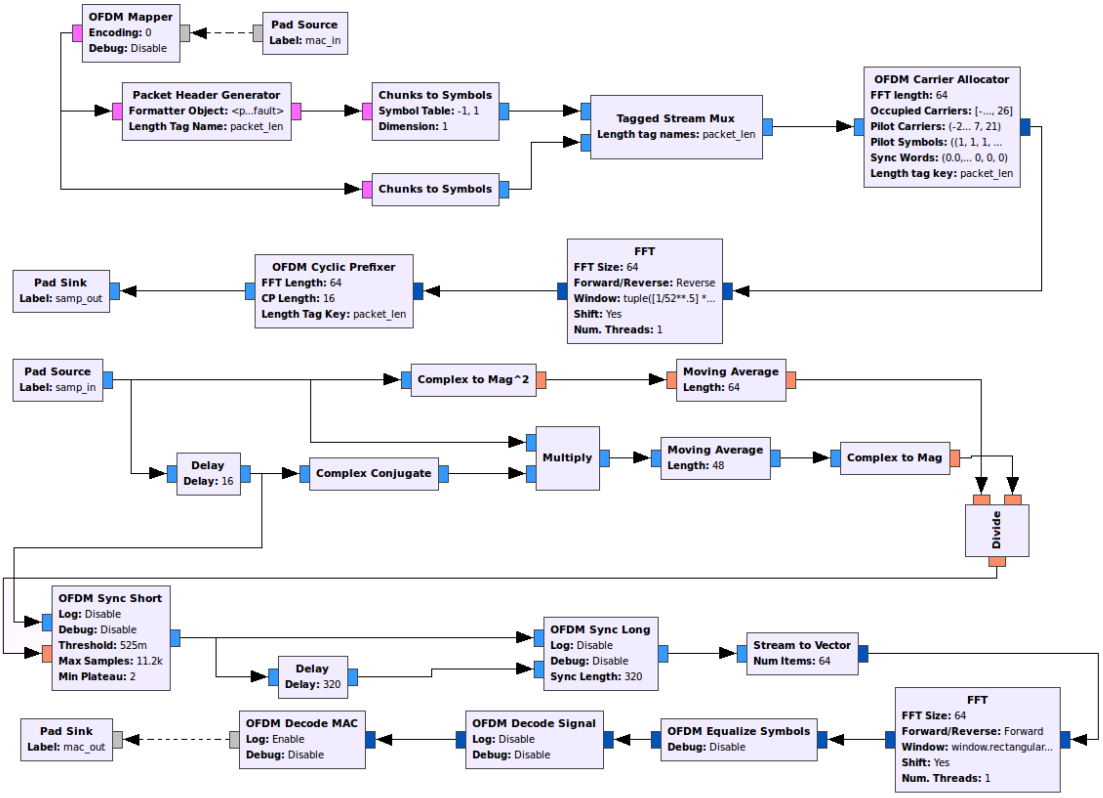


Figure 3.7: Overview of the *WiFi PHY Hier* flow graph in GNURadio Companion.

The next block in the Wifi PHY Hier chain is the *Packet Header Generator*, which generates the header of the frame, including the signal and service fields. The header is BPSK modulated by the top *Chunks to Symbols* block and the remaining frame is modulated by the bottom *Chunks to Symbols* block, according to the chosen modulation. Then, the header is finally joined to the remaining of the frame.

The next block is the *OFDM Carrier Allocator* that is responsible for the aggregation of the pilot subcarriers and the *FFT* block is responsible for the inverse FFT, i.e., for the transition from frequency to time domain. The authors decided to use 1 as the transition width of the window function, because it asserts that the output signal honors the spectral mask defined in the standard. Besides, this contributes to the faster decay of the signal in frequency domain, thus, limiting adjacent channel interference [10]. Finally, the *OFDM Cyclic Prefixer* block aggregates the guard intervals to each symbol of the frame.

Regarding the structure of the transmitter again, there is a *Message Strobe* block responsible for defining the text message that will be sent and the time period between messages. Besides, as the receiver, the transmitter has the User Defined App blocks. Therefore, the *Wireshark Connector* block allows the user to see the detailed information of each frame that is built by the implementation and sent by the SDR.

Finally, the *USRP Sink* block defines the parameters on the USRP SDR board, such as the transmitter antenna, the SDR clock and the sample rate.

3.3 GNURadio OFDM Transceiver

As it was mentioned before, both the structure of the transmitter and the structure of the receiver are constituent parts of the transceiver implementation that is illustrated in the Figure 2.16.

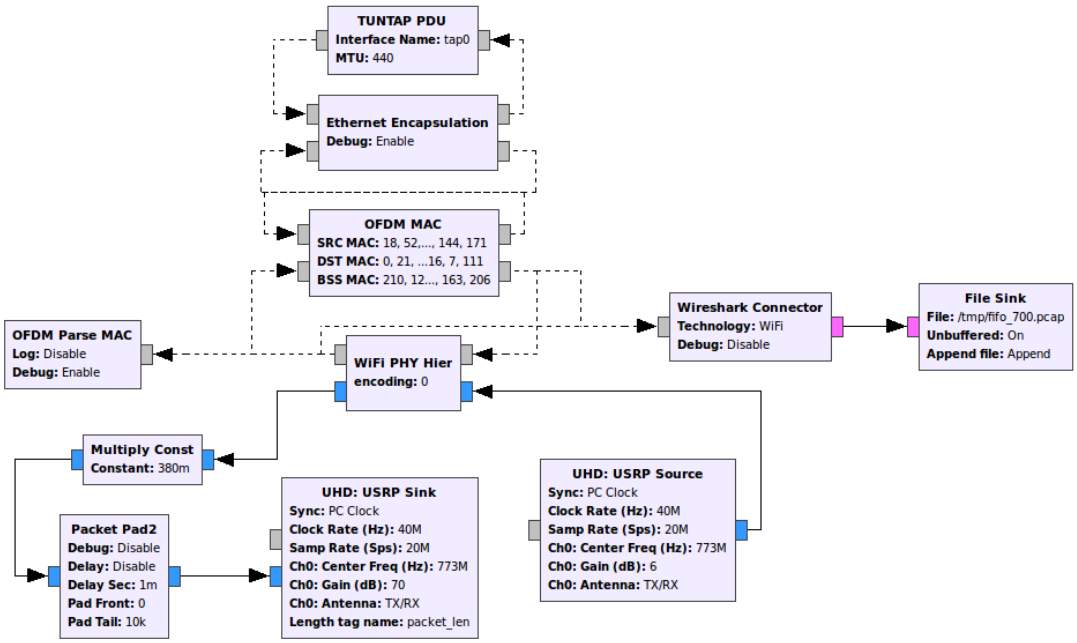


Figure 3.8: Overview of the Transceiver flow graph in GNURadio Companion.

The *Tuntap PDU* block is responsible for sending the messages received by the GNURadio from the SDR to the virtual interface which it is assigned to. Besides, assigning an Internet Protocol (IP) address to the virtual interface will allow the user to send packets from the transmitter part, because the packets will be forwarded to the GNURadio, encapsulated with the parameters that have been established and then sent to the SDR and transmitted over-the-air.

Finally, the *USRP Sink* block and the *USRP Source* block define the parameters on the USRP board, such as the transmitter and receiver antennas of the SDR, respectively.

Chapter 4

Experimental Planning and Testbed Design

In this chapter, we describe the hardware and the software specifications that were used in the testbeds and we also overview the measurement scenarios, where we conducted the experiments in order to obtain the experimental results. Then, we further detail the metrics that were used to evaluate the IEEE 802.11a/g/p SDR Transceiver implementation and the planning of the underwater testbed.

4.1 Hardware Specifications

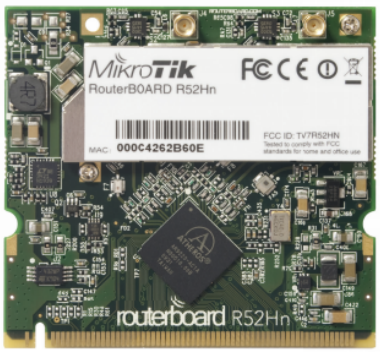
This section refers to all the hardware that was used during the experimental measurements during the dissertation, including the SDR board, the commercial wireless cards, the adapted antennas and the acrylic cylinders used during the underwater testbed.

4.1.1 SDR Board and Wireless Cards

As derived from Section 2.4, we decided to use the B210 USRP model as a SDR platform, due to their superior features in comparison with the other USRP models and the other SDR platforms. The B210 USRP model is a SDR single board, instead of a daughter/motherboard solution like the other USRP models. This device is also the first board that can operate with 2x2 MIMO and it supports USB 3.0 interface. This device is shown in the Figure 2.6 (Section 2.1.1), in which some SDR platforms currently available for research and development are presented and compared.

On the other hand, in the series of interoperability tests performed in order to evaluate the implementation, it was necessary to use commercial wireless cards. It was decided to use the RouterBOARD R52n-M [56] for the IEEE 802.11 radio for the 2.4 and 5GHz band, which is a miniPCI network adapter that supports dual band IEEE 802.11a/b/g/n standards and has low power consumption. It also has a high power transmitter, that allows even more range and supports up to 300Mbps physical data rates and up to 200Mbps of actual user throughput on the uplink and downlink connections, thus allowing a higher performance.

Besides, in the testbeds, it was used the 700MHz band. To achieve this band, it was used the Ubiquiti XTREMERange7 [57], which is a compact radio module that supports 32 bits mini-PCI type IIIA standard and features high output power (600mW) [58]. Ubiquiti XTREMERange7 can be set to operate in four different channels within 760-780 MHz, based on the IEEE 802.11g standard (OFDM). The two devices are presented in Figure 4.1a and Figure 4.1b, respectively.



(a) RouterBOARD R52n-M [56].

(b) UBIQUITI XR7 [57].

Figure 4.1: Commercial Wireless Cards.

Finally, these commercial wireless cards were assembled in the system board Alix3d3 manufactured by PC Engines [14], which is a miniPC board optimized for wireless routing and network security applications that has low power consumption and a 500 MHz AMD Geode processor (x86 architecture) and 256 MB DDR RAM. This device, shown in the Figure 4.2, also allows great flexibility because it enables the installation of multiple applications and operating systems.

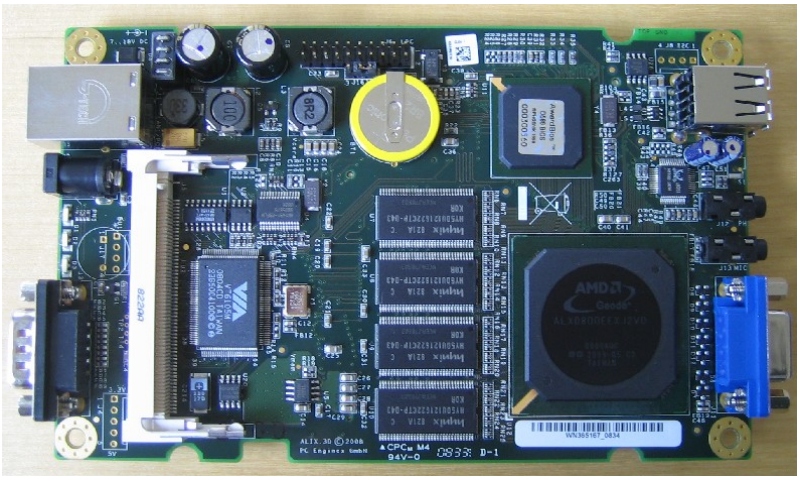


Figure 4.2: Alix3d3 system board [14].

4.1.2 700 MHz Band Antenna

The antenna used for the 700 MHz testbed was a loop antenna. This antenna was created previously for other dissertations regarding maritime and underwater communications that used the same wireless cards [59, 15]. The loop antenna is easy to build, only requiring a copper cable with an acceptable width. Besides, it is necessary to cut and solder to the connector, if little adjustments are required. It should also be noted that this antenna can be used for other lower frequencies, with significant gain losses. Therefore, we evaluated the performance of the SDR implementation at low frequencies by using this antenna in air scenarios, and we used a water-adapted antenna for the underwater environments. The Figure 4.3 shows the size difference between the air and water adapted antennas.

Figure 4.3: Air Adapted and Underwater Adapted (smaller) 768 MHz antennas [15].

Figure 4.4 shows the S21 parameter measurements from the Vectorial Network Analyzer (VNA) by using two 700MHz water-adapted loop antennas in an underwater scenario. S21 represents the power received at antenna 2 relative to the power input to antenna 1. These measurements were carried in the optoelectronics laboratory at INESC.

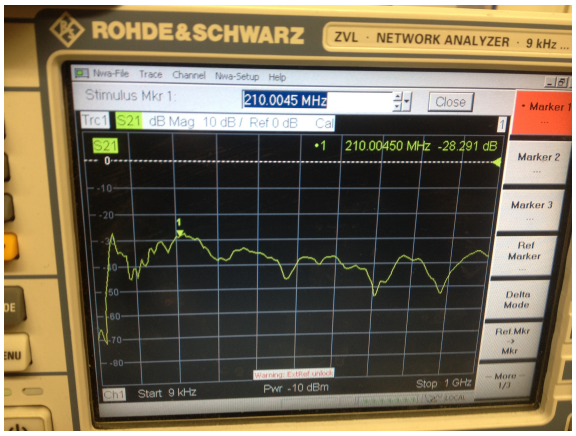


Figure 4.4: S21 values for the water-adapted 700 MHz loop antenna.

As we can see, the antennas' S21 parameter had a maximum return loss point around the 210 MHz frequency. Therefore, we conclude that this antenna may perform well for underwater communications in the 200-220 MHz band frequency. In our testbed, we will test multiple frequencies, such as 100, 200, 300 MHz, though we expect a higher performance with a frequency of 200 MHz than in the other frequencies.

4.1.3 Acrylic Cylinder and End Caps

In this dissertation, in order to submerge the two SDR boards, we used two airtight acrylic cylinders developed by OceanSys Group at FEUP. In the Figure 4.5, it is shown a view of one of the cylinders. The end caps were machined in the FEUP's laboratories and the system allows depths of at least 10 meters.

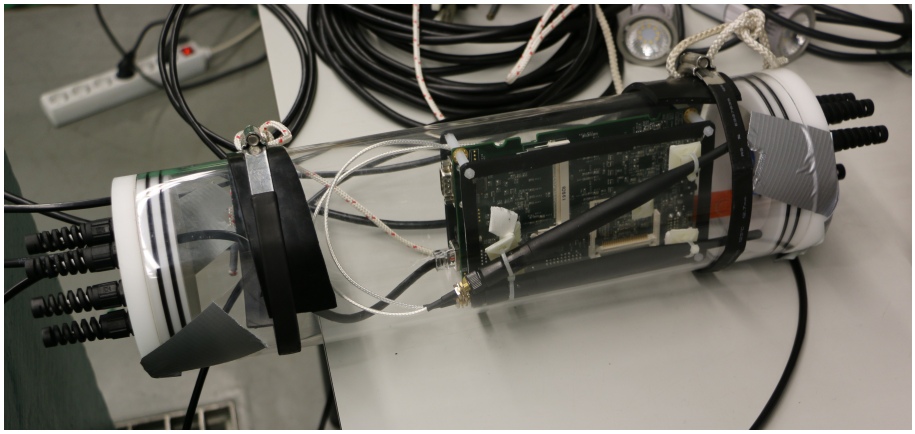


Figure 4.5: View of the acrylic cylinder.

4.2 Software Specifications

This section refers to all the software that was used during the experimental measurements during the dissertation, including operating systems, wireless drivers and additional tools that were used in order to extract reliable data from the testbeds.

Operating System and Wireless Drivers

In this dissertation, we decided to use the OpenWrt Linux based operating system [60]. OpenWrt is a Linux distributed version for embedded devices, that is supported by the Ubiquiti hardware. This operating system can be easily modified because it supports a fully writable file system with package management. Besides, it allows the full customization of the wireless card via its GUI. In our testbeds, we used the kernel version r44532 (also known as Chaos Calmer (Bleeding Edge)) together with the ath9k and ath5k drivers, in order to work with the Routerboard r52n-M wireless card for the 2.4 and 5 GHz bands and the UBIQUITI XR7 wireless card for the 700MHz

band, respectively. On the other hand, we used the GNURadio SDK Platform in order to conduct testbeds with the SDR board. The GNURadio SDK Platform is a free open-source software development toolkit that provides all necessary signal processing blocks, in order to build and simulate SDR systems. The GNURadio SDK platform is described in more detail in the Section 2.1.2.

Additional Software

Additional packages were required in order to proceed with the measurements in the testbeds. These software packages were installed directly over the OpenWRT operative system:

- **Iperf** [61] is a traffic generator software used to measure network performance. It permits to create UDP/TCP data streams in order to measure throughput, delay jitter and packet loss.
- **Iwinfo** is a tool included in the **Wireless Utilities** package that can be used to monitor a set of parameters of the wireless card, such as the Output Power, the Signal and the Link Synchronized Throughput.
- **Tcpdump** [62] is a powerful command-line packet analyzer; and libpcap, a portable C/C++ library for network traffic capture.
- **Horst** [63] is a small, lightweight IEEE 802.11 wireless LAN analyzer with a text interface.

4.3 Measurement Scenarios

In this section, we overview the measurement scenarios that were used to obtain the experimental results.

4.3.1 Laboratory Environment

Initially, simple tests with the SDR and GNURadio SDK platform were performed, such as the receiving of FM signals. Then, interoperability tests with the gr-ieee802.11 IEEE 802.11a/g/p transceiver implementation were conducted using off-the-shelf equipment. All this initial tests occurred in laboratory, in particular, in FEUP and INESC. In this way, by initially testing in a controlled scenario, it was possible to detect any failure or error and, therefore, fix it previously. All the obtained results are more detailed in the Chapter 5, in the Section 5.1.

4.3.2 Air Environment

After conducting the initial tests in a laboratory environment, multiple tests were performed in an exterior scenario, in particular, with over-the-air communications. This testbed aimed to establish a connection between two B210 USRP models, in which one is the receiver and the other is the transmitter. The two SDR platforms were connected to a PC via an USB 3.0 interface, because it allows high sample rates when the data is transferred between the FPGA from the SDR to a host PC. Besides, interoperability tests with commercial equipment were conducted, in order to

test the connectivity between the SDR board and commercial Wi-Fi cards. It should be noted that interoperability tests were also performed between two SDR boards. Therefore, there were tests in which the SDR board was the transmitter and the Wi-Fi card was the receiver, but there were also some performed the opposite way. All the obtained results for this environment are more detailed in the Chapter 5, in the Section 5.2. A simple scheme of the proposed scenarios are illustrated in the Figures 4.6 and 4.7.

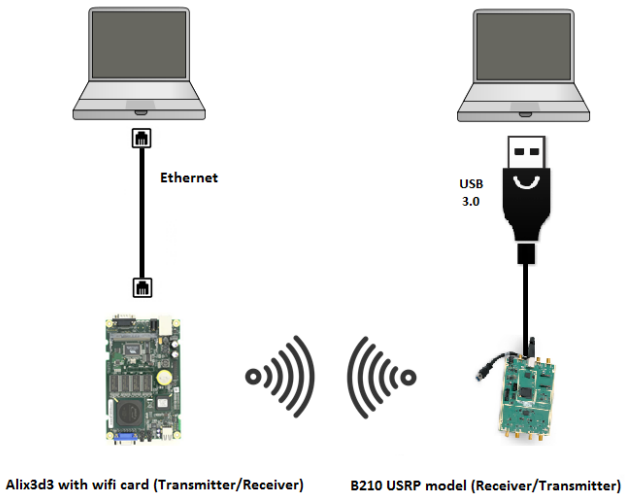


Figure 4.6: Scheme of one of the tests performed in an air environment.

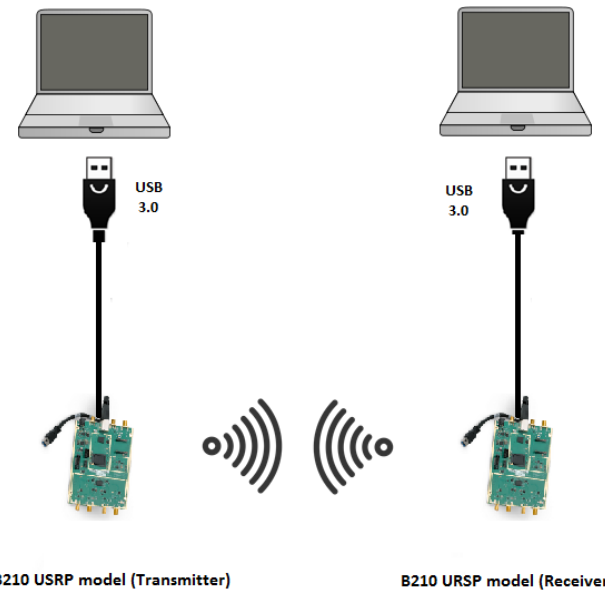


Figure 4.7: Scheme of one of the tests performed in an air environment.

4.3.3 Underwater Environment

The last scenario is an underwater testbed, where several experiments were conducted. These experiments were performed in two places: in a freshwater tank with 6 meters long, 5 meters wide and 2 meters depth placed in FEUP OceanSys Group facilities [63] and in a freshwater tank with 10 meters long, 6 meters wide and 5.5 meters depth placed in INESC TEC facilities at ISEP [64]. A view of the OceanSys Group freshwater tank is shown in the Figure 4.8 and a view of the INESC TEC Robotics freshwater tank is shown in the Figure 4.9. Besides, we performed tests in a seawater environment in the Alfeite Lisbon Naval base, where the water conductivity was 4.8 *S/m* at a temperature of 25° C.

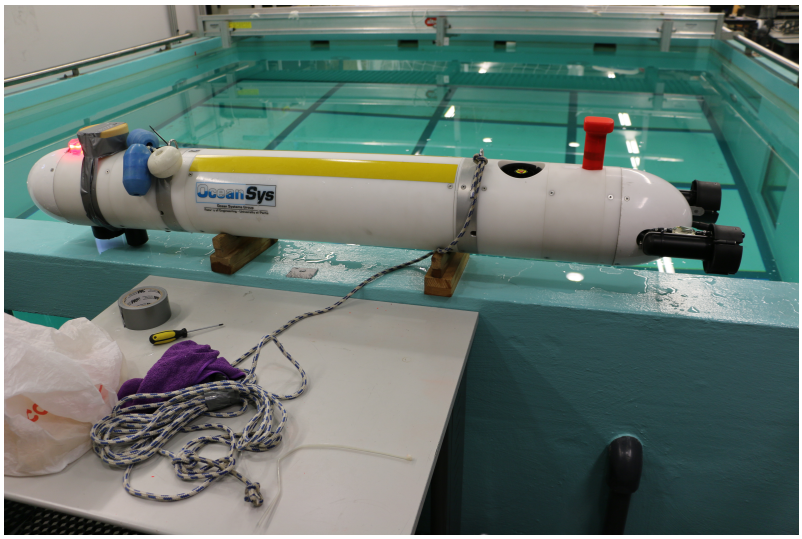


Figure 4.8: OceanSys Group Fresh Water Tank.

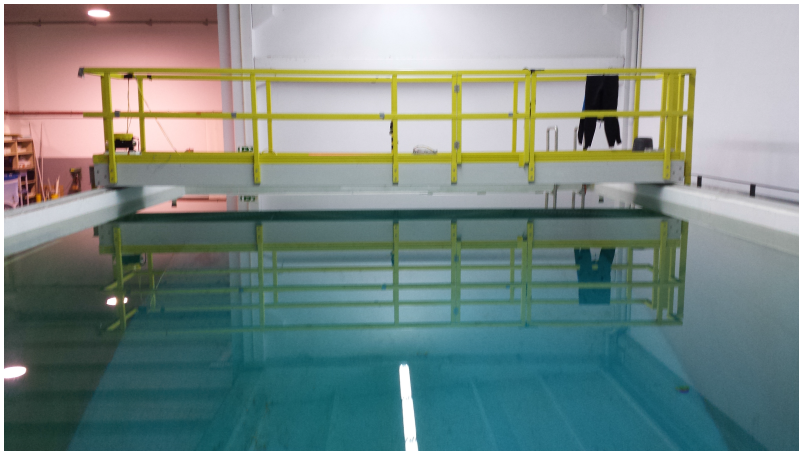


Figure 4.9: INESC TEC Robotics Fresh Water Tank [15].

Our main goal was to evaluate the performance of the gr-ieee802.11 SDR transceiver implementation by transmitting Wi-Fi packets between two submerged B210 USRP SDR platforms. A simple scheme of the proposed scenario is illustrated in the Figure 4.10.

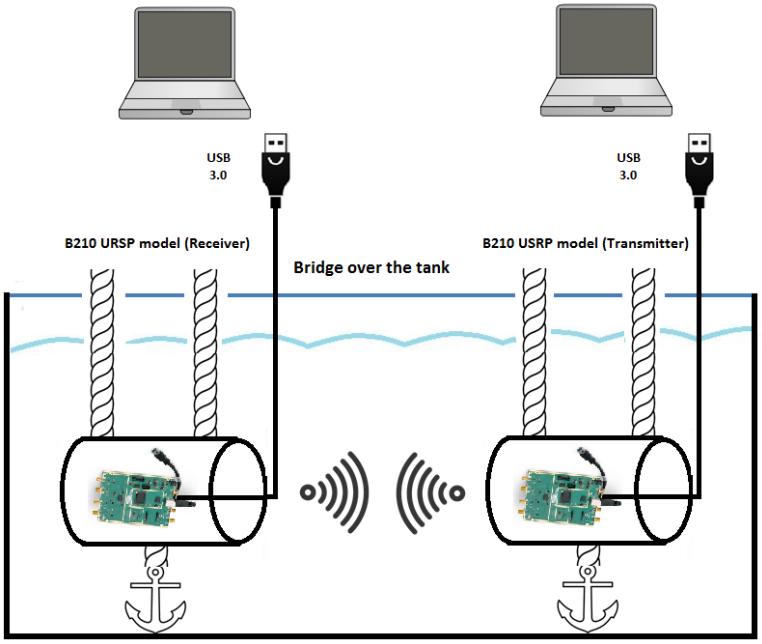


Figure 4.10: Scheme of the water environment testbed.

The SDR platforms were suspended through a cord along with an USB 3.0 communication wire. Besides, these devices were inserted in an airtight acrylic cylinder with two end caps and an anchor was used in order to sink the cylinder to a certain depth. It should also be noted that in order to pass the USB 3.0 cable through the end cap, we needed to cut it, because the header adapter couldn't pass the end cap. Therefore, after passing the cable through the cylinder, a new header adapter was soldered to the head of the USB cable and multiple tests were conducted to see if the cable was working well, in air and water environments. The figure 4.11 shows a view of one of the SDR boards inside the cylinder, already including the USB 3.0 cable and the anchor.

On the other hand, for this testbed we needed two computers so we could connect both SDR boards via USB 3.0 interface and start to obtain the experimental results.

These environment is a way of simulating a real case scenario, as two devices transmitting underwater at a certain depth. In this sense, the depth and the distance to the tank walls of the underwater devices were such that the air propagation of the RF waves and the wall reflections does not interfere with the results obtained during the testbed. The cylinders were positioned at 2.5 meters from the surface, at 3 meters from the tank bottom, and were at a distance of at least 2 meters from each wall. Besides, the electromagnetic insulation of the USB cables was such that it didn't affect the results obtained, since the RF signals could be propagated through the cables and crosstalk could be experienced. However, we took this possibility into account in our tests, but it was not experienced. It should also be noted that the depth of the INESC TEC Robotics

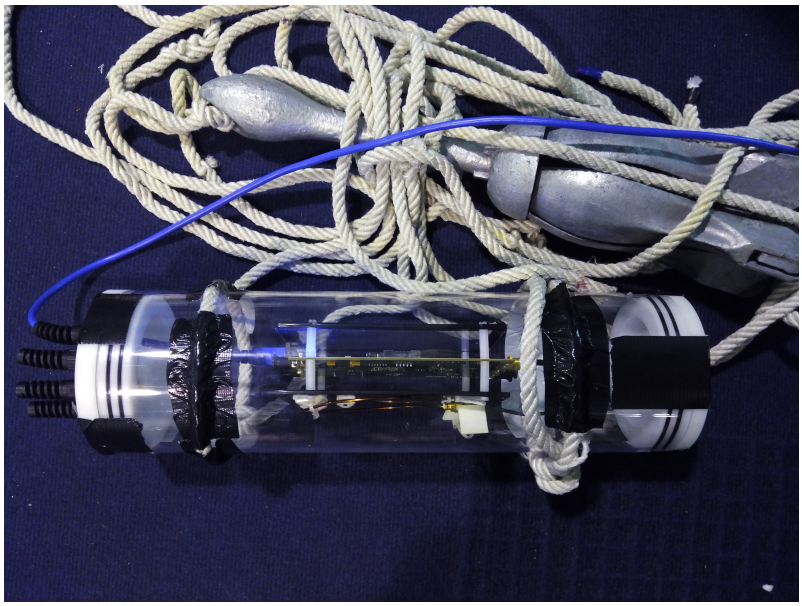


Figure 4.11: View of the SDR board inside the airtight cylinder.

freshwater tank was enough to ensure that no communication is made over-the-air when higher distances were tested, because in the OceanSys Group freshwater tank the maximum distance that can be tested is 2.25 meters in order to ensure that the signal is only propagated through the water and not through the walls instead.

According to the measurements performed by P. Freitas [15], the water conductivity of the INESC TEC Robotics freshwater tank is 0.0487 S/m, which can be considered the worst case scenario for freshwater, because its streams should have a conductivity between 0.015 to 0.05 S/m. This scenario provides an evaluation of the gr-ieee802.11 IEEE 802.11a/g/p transceiver implementation in a completely new wireless environment, where the behavior of different frequency bands was tested.

All the obtained results for this environment are further detailed in Chapter 5, in Section 5.4.

Metrics to be considered in the evaluation

After each scenario change, several parameters were tested and evaluated in order to analyze the performance of the implementation. These different metrics were also useful to compare our results with simulations and previous works. The metrics are:

- TCP and UDP Throughput: Measure the maximum achievable throughput from the transmitter to the receiver;
- Packet loss: Represents the amount of packets that fail to reach their destination during transmission;
- Delay: Represents the time that a packet takes to reach the destination, after being transmitted by the source.

- Jitter: Variation of time between arriving packets;
- Transmission power: Variable which provides current power in the Transmitter (dBm);
- Distance: Represents the current distance between the transmitter and the receiver;
- SNR: Measure that represents the comparison between the level of a desired signal to the level of background noise. It is usually expressed in decibels;

Chapter 5

Performance Results

In this chapter, we overview the performance results obtained in the different scenarios, the difficulties we faced in each scenario and the steps we went through to overcome the problems in order to achieve our objectives.

First of all, the familiarization with all the GNURadio SDK Platform was done through the completion of several simple tests. Secondly, some initial tests with the existing gr-ieee802.11 SDR transceiver implementation were carried out, and then interoperability tests with commercial wireless cards were performed, in order to evaluate the performance of the SDR implementation. We measured PDR, UDP throughput, jitter and RTT through the use of multiple frequency bands.

Finally, we present and analyze the results obtained from the different measurement scenarios: the laboratory environment, the over-the-air scenario and the underwater environment.

5.1 Laboratory Environment

Initially, several simple tests with the SDR and the GNURadio SDK platform were performed in a controlled scenario - in particular, in FEUP and INESC - where it is possible to detect any failure or error, and, therefore, fix it previously. Thus, some tutorials with the GNURadio SDK and the USRP B210 SDR platform were conducted in order to begin the familiarization with this platform, namely the creation of a receiver of FM signals.

The next step was to perform some tests with SDR applications produced by the GNU Radio community, such as the gr-rds [65] and gr-air-modes [66] applications. The gr-rds is an implementation of broadcast FM radio RDS reception developed by B. Bloessl et al, while gr-air-modes implements a SDR receiver for Mode S transponder signals, including ADS-B reports from equipped aircraft. Upon the execution of some tests with both implementations, it was concluded that the USRP B210 SDR platform was working properly, since the SDR board did not show any problem in the multiple tests performed.

Then, we decided to start testing the IEEE 802.11a/g/p transceiver implementation for SDR by performing a battery of interoperability tests with off-the-shelf equipment and with two SDR boards. A view of our testbed can be seen in the figure 5.1.

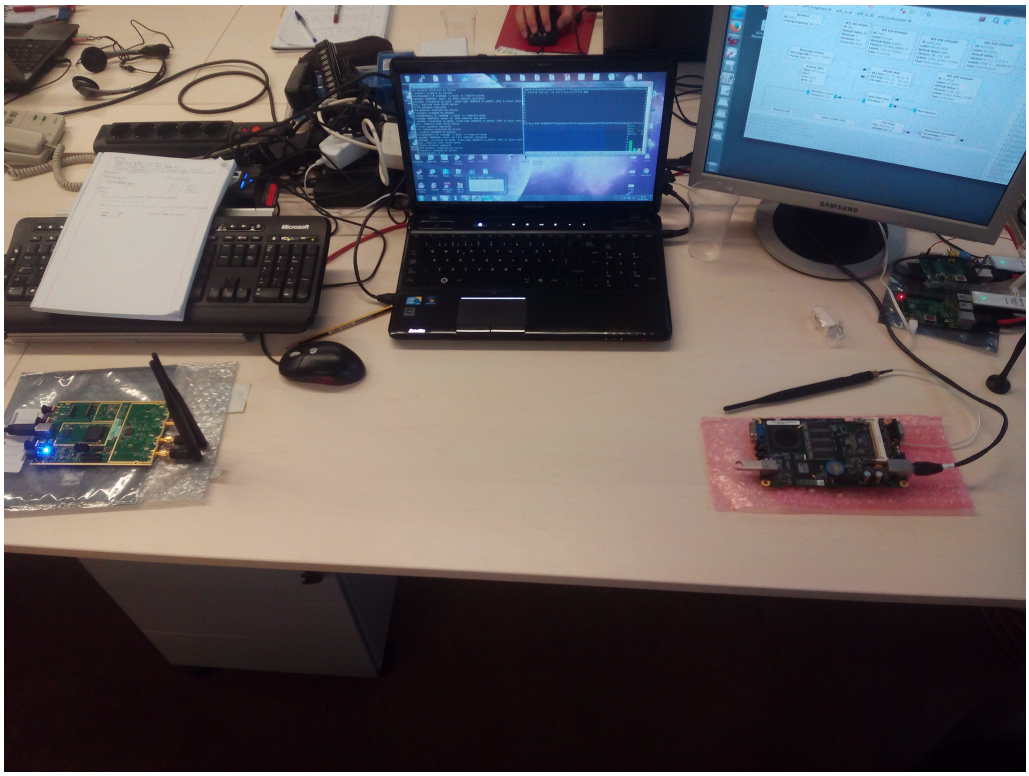


Figure 5.1: View of the laboratory environment testbed.

In the first initial tests with the implementation, we tried to receive frames in multiple frequencies in the 2.4 and 5 GHz band through the use of some commercial wireless cards to generate the frames. Although the frames were generated correctly, the ones received by the SDR weren't decoded properly, because their reception experienced errors, especially parity and Cyclic Redundancy Check (CRC) errors. Besides, we also experienced some problems with the transmitter part of the implementation, because we were unable to receive frames with wireless cards in monitor mode in the 2.4 and 5 GHz band when we tried to send them from the SDR board. Therefore, it was decided to use the *uhd fft* to check if the frames were received properly by the SDR and if the error was in the implementation. The *uhd fft* is a very simple spectrum analyzer tool, which uses a connected SDR device to display the spectrum at a given frequency [19]. With this tool, it was possible to see that the spectrum of the Wi-Fi signals sent by the wireless cards and received by the USRP B210 SDR board had the correct shape. Thus, it was decided to perform more tests with the implementation in order to detect and correct the error that was causing the issues in the decoding and transmitting of the frames. Before conducting these tests, the implementation was changed to receive smaller frames, because the authors had checks on the implementation code that discarded frames with data fields lower than 30 bytes, so this was changed to a lower value, so that we could receive smaller packets. Besides, it should also be noted that we changed the implementation to operate at a sample rate of 5 MHz and we introduced the *Wireshark Connector* block in the transmitter part, so that the packets sent by the SDR could be monitored in real time

with the Wireshark tool, from which we can see the detailed information of each frame and see if they were created by the GNURadio SDK without errors.

After this modifications, we tested the reception of frames in multiple channels at a sample rate of 5, 10 and 20 MHz; the transmission of frames in different frequency bands at a sample rate of 5, 10 and 20 MHz; the reception of beacon frames from wireless cards in ad hoc mode at several sample rates and using also two wireless cards in ad hoc mode in the same frequency band, in order to receive the frames of their association in the SDR. However, the problem persisted, with CRC errors in the received frames. We also used a Rohde & Schwarz SMJ100A Vector Signal Generator [67] located in the optoelectronics laboratory at INESC, in order to confirm the reception of Wi-Fi signals and their correct shape.

Figure 5.2 presents a view of the vector generator signal used in the tests to generate Wi-Fi packets. With this generator, we were able to test the reception of Wi-Fi signals in multiple frequency bands, including the 70 MHz band.

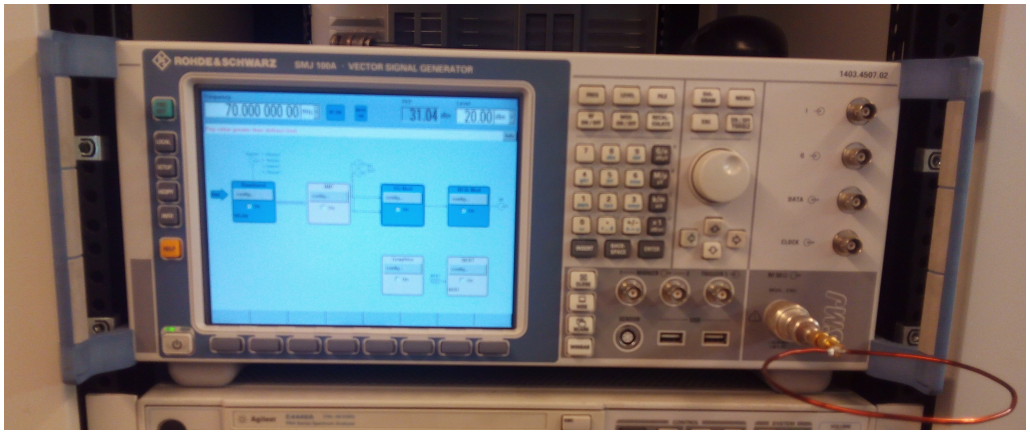
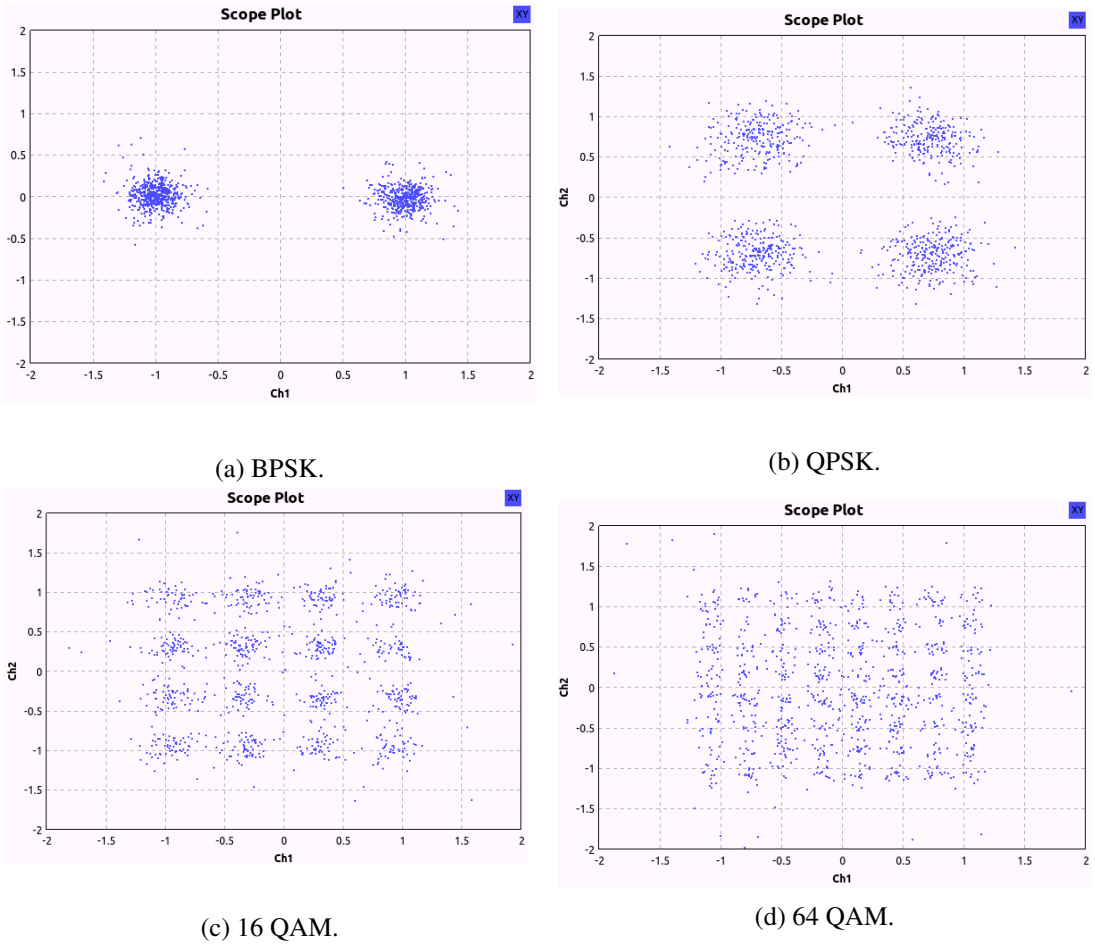


Figure 5.2: Vector generator signal.

After analyzing the results of the tests and the SDR parameters in the GNURadio blocks, it was noticed that the default clock of the USRP B210, unlike the USRP N210, is set to 32 MHz, which could be causing the errors obtained with the SDR implementation. Therefore, we decided to change the default clock of the SDR to 40 MHz to see if the problem persisted, because the SDR changes the sample rate to 16 MHz, when we set it at 20 MHz, with the default clock set at 32 MHz, but with a default clock of 40 MHz, the SDR can change the sample rate to 20 MHz, which was exactly what we needed. When we changed the default clock of the SDR to 40 MHz, the SDR implementation started to work properly: the frames were then decoded correctly and the Wi-Fi cards received the frames sent by the SDR at multiple data rates and in different frequency bands. Then, we performed more tests to see if there were no more errors in the SDR implementation. For instance, one of the tests consisted in the reception of Wi-Fi signals sent by commercial Wi-Fi cards at multiple data rates, where the Figure 5.3 presents the constellations plots of the multiple modulated symbols received in this test by the SDR and that are supported by the receiver part of the implementation.



(a) BPSK.

(b) QPSK.

(c) 16 QAM.

(d) 64 QAM.

Figure 5.3: Constellation plot of modulated symbols.

It should also be noted that for each frequency tested, we had to manually adjust the receiver gain of the SDR, so we could optimize the reception of the OFDM signal. There is no automatic process to adjust the receiver gain of the SDR, although the authors are developing an automatic gain control algorithm that can be implemented in the FPGA of the SDR. They introduce this algorithm in [68]. On the other hand, we updated the gr-ieee802.11 SDR implementation to the most recent version during the testbed, because the authors introduced some new features to the implementation, including the creation of the packet length variable, which allows the modification of this value in real time, i.e., when the SDR is transmitting packets. Besides, the maximum packet size allowed in the implementation was increased to 1500 bytes, while in the previous version tested the maximum allowed size was 281 bytes, which was a considerable limitation.

Finally, we adapted the SDR implementation to operate at lower frequencies, such as 700, 300 or 200 MHz. In order to test the reception of Wi-Fi signals with the SDR board at these low frequencies, we used the vector generator signal to generate the packets and the *uhd fft* and the gr-ieee802.11 SDR implementation to confirm the reception of the Wi-Fi signal at these low frequencies.

Figure 5.4 presents a view of the test environment. In this test, we used air adapted loop

antennas to test these low frequencies and we were even able to receive Wi-Fi signals at the 70 MHz band, which is the lowest allowed frequency by the USRP B210 board. The Figure 5.5 presents the results obtained with the *uhd fft* tool for the 70 MHz band.

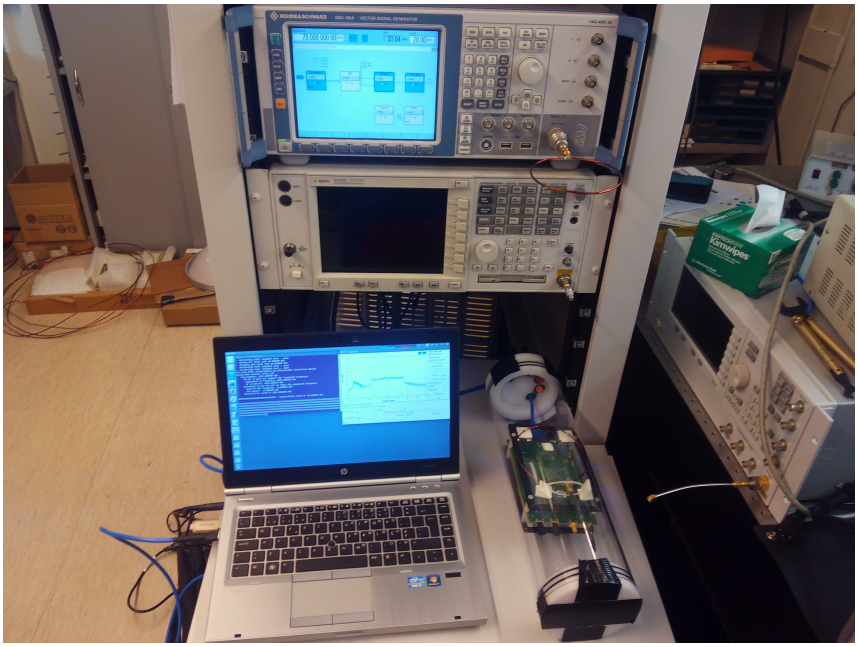
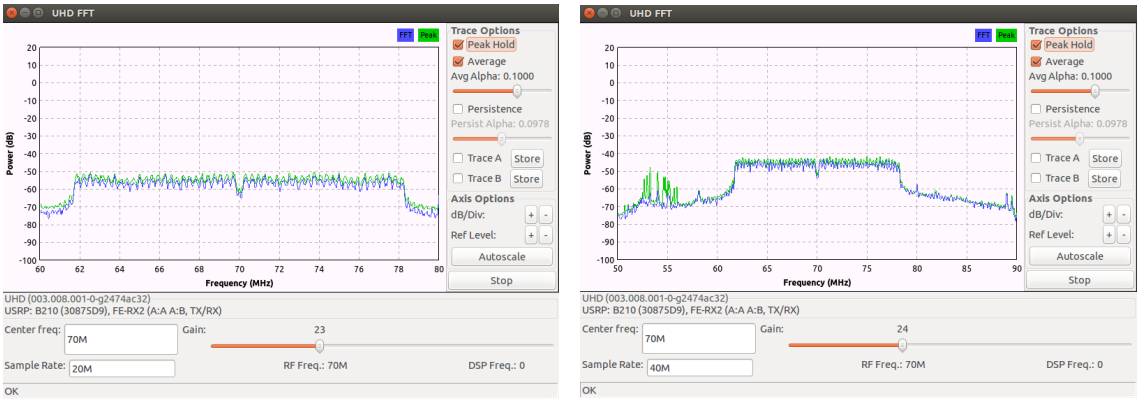


Figure 5.4: View of the tests performed with the vector generator signal.



(a) Sample rate 20 MHz. (b) Sample rate 40 MHz.

Figure 5.5: Spectrum of the Wi-Fi signals received by the *uhd fft* tool.

After solving the initial problems with the SDR implementation, we decided to conduct a testbed in an exterior environment with over-the-air communications. The experimental results obtained from this testbed are further described in the following section.

5.2 Air Environment @ INESC TEC

In the air environment testbed, the evaluation of the implementation was based on the metrics that are described in the Section 4.3. In these experiments, we tested the behavior of different frequency bands. Therefore, in this section, we describe all the tests that were conducted with the multiple devices through over-the-air communications. We give details on these tests and then we present the results obtained.

Finally, our results were analyzed and compared with theoretical and simulation models.

5.2.1 Replication of some tests performed by B. Bloessl et al.

First of all, we decided to replicate some tests performed by B. Bloessl et al., in order to see if the implementation was working properly and if we could achieve the same results as the authors obtained. Thus, in this section, we present the results obtained in our tests and then we compare them with the ones obtained by the authors.

5.2.1.1 PDR of frames sent from the SDR and received with a Wi-Fi card

In this test, we sent frames from the SDR and received them in the RouterBOARD R52n-M Wi-Fi card. All measurements were performed on channel 165 with a frequency of 5.825 GHz, a packet size of 133 bytes, a rate of 30 packets per second - which results in a data rate of 31.92 kbit/s - and the distance between the devices was approximately 1.2 meters, meeting the experimental conditions of the authors. Besides, the channel bandwidth used was 10 MHz.

The results obtained by the authors are depicted in the Figure 5.6 and ours are shown in the Figure 5.7.

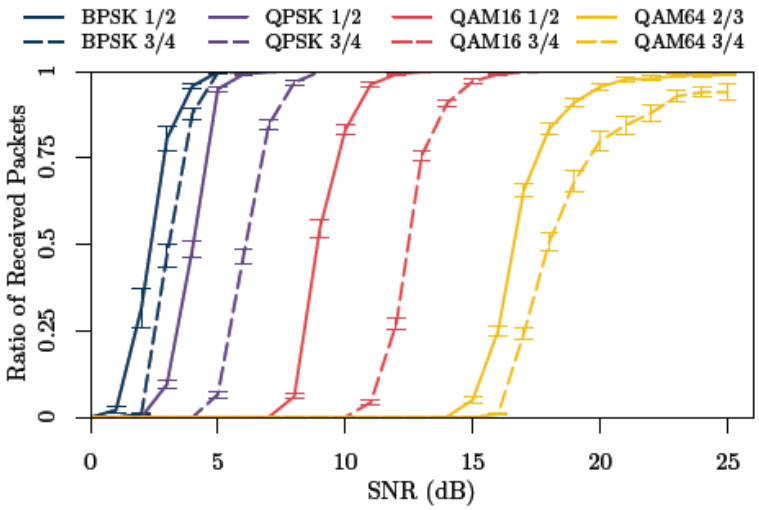


Figure 5.6: PDR of frames sent from the SDR and received with a commercial device. The devices are connected via cable and the packet size is 133 Byte [10].

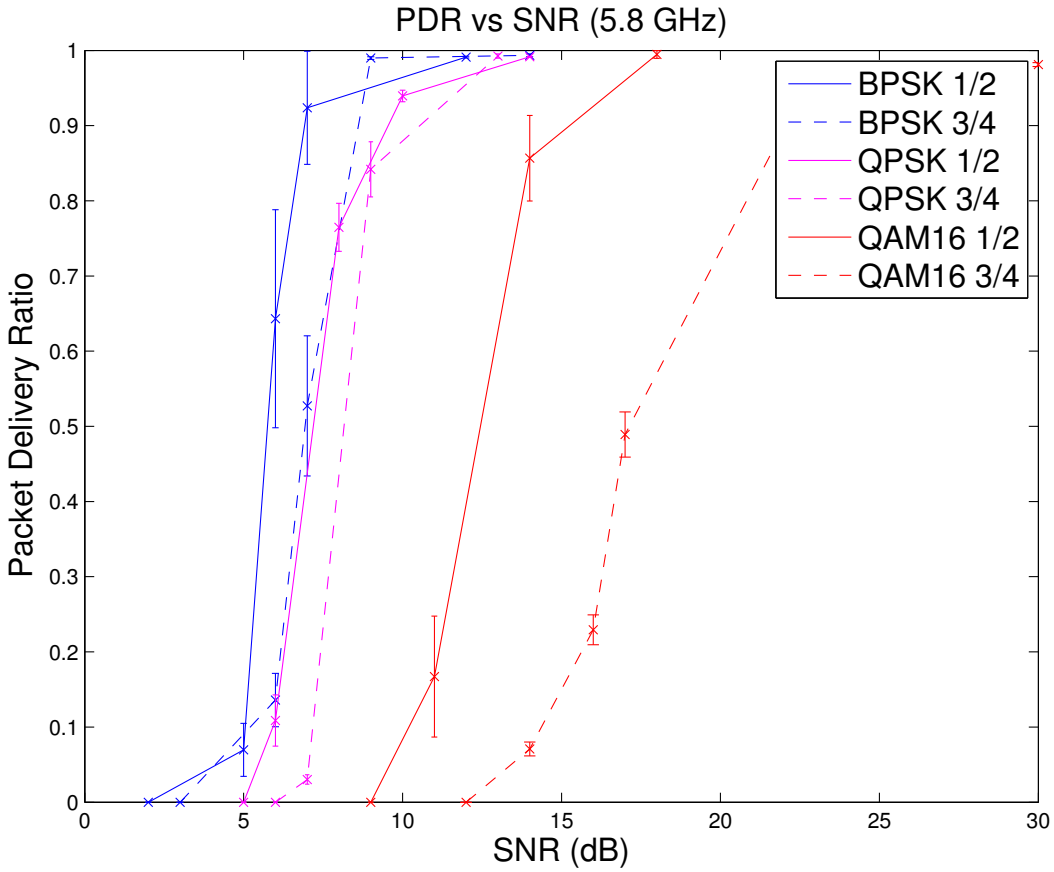


Figure 5.7: PDR of frames sent from the SDR and received with the RouterBOARD R52n-M Wi-Fi card. The packet size is 133 bytes.

The SNR was measured by the Wi-Fi card, since the Wi-Fi cards annotate each received frame with metadata including signal and noise levels when set to monitor mode. The noise level of the Wi-Fi card was -95 dBm.

Comparing our results with the ones obtained by B. Bloessl et al., we can conclude that for the same PDR we need a slightly higher SNR for each modulation. Besides, in the higher modulations, i.e., QAM64 3/4 and QAM64 4/3, we were unable to receive packets from the SDR, but the authors achieved good PDR values with a higher transmitter power in this modulations. Thus, we tested them with sample rates of 5 and 20 MHz to see if we could in fact receive frames in this modulations from the SDR, and we were finally able to do so at 20 MHz. It should also be noted that our USRP board is different than the one used by the authors: they used a N210 board and we used a B210 board, which could have influenced the obtained results.

5.2.1.2 PDR of frames sent from a Wi-Fi card and received by the SDR

In this test, we sent frames from the RouterBOARD R52n-M Wi-Fi card with the fping command and received them in the SDR board. All measurements were performed on channel 165 with a

frequency of 5.825 GHz, a packet size of 95 bytes and a rate of 5 packets per second - which results in a data rate of 3.8 kbit/s - and the distance between the devices was approximately 6 meters. Besides, the channel bandwidth used was 10 MHz.

The results are depicted in the Figure 5.8 and the results obtained by the authors are depicted in the Figure 5.9.

As we can see in the figure, we only tested two modulations, because our version of OpenWrt didn't allow us to change to a data rate of 9 and 18 Mbit/s. We even tested two more versions of OpenWrt, the r45847 and r42625, but the problem remained. Regarding the results, we achieved the same PDR for a higher transmitter power, when compared with the results from the authors. Besides, we used the fping command to generate the packets from the Wi-Fi card and the authors in their test used a different operating system, a Linux environment, and other program to generate the packets, a version of the packetspammer - a program developed in C that can inject packets in the medium through the use of a monitor interface - which could be the reason why our results are slightly different than the ones obtained by them.

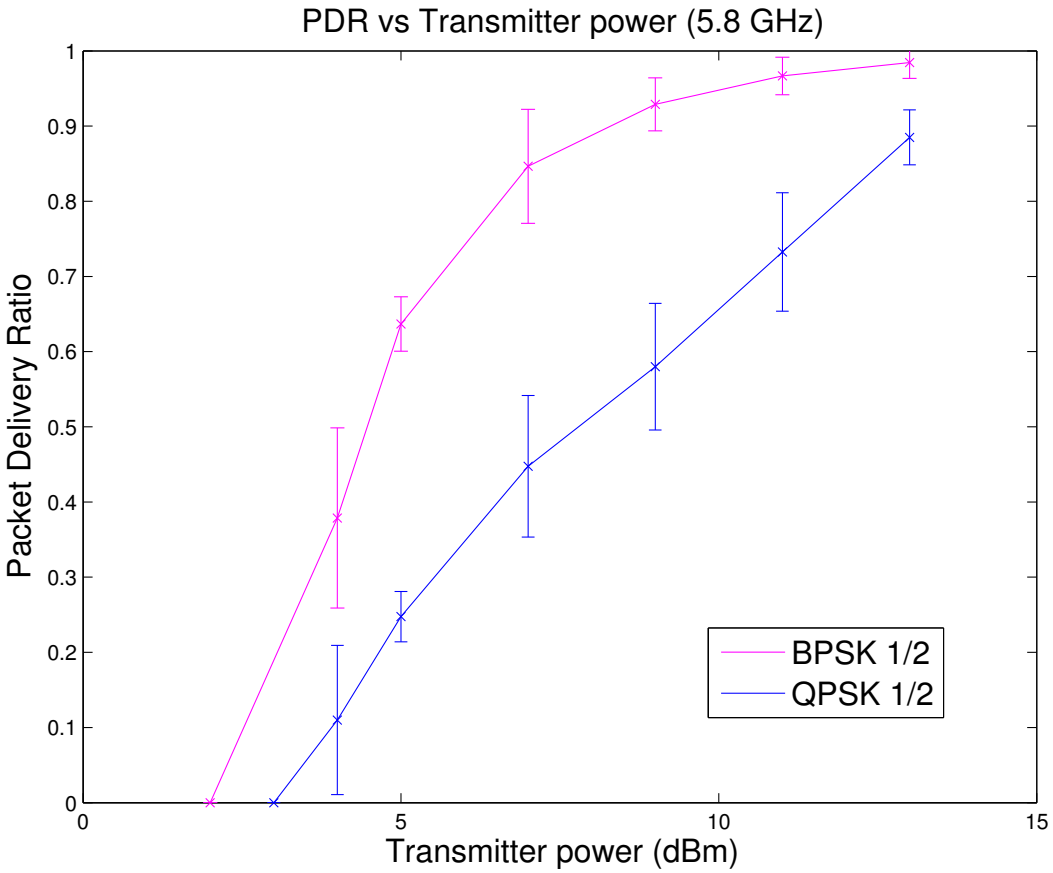


Figure 5.8: PDR of frames sent from the RouterBOARD R52n-M Wi-Fi card and received with the SDR. The packet size is 95 bytes.

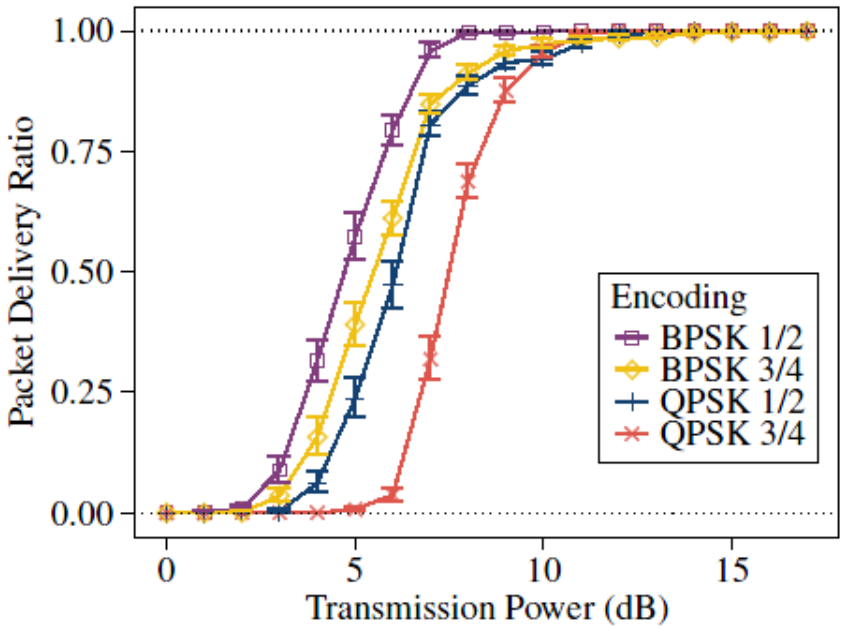


Figure 5.9: PDR of IEEE 802.11p packets, sent from a MK2 from Cohda Wireless. The packet size is 95Bytes [12].

5.2.2 Tests with broadcast communications

In this section, we describe the tests that were performed with a SDR and Wi-Fi cards through broadcast communications. Each test performed will be further described in this section.

5.2.2.1 PDR between SDR and Ubiquity XR7 Wi-Fi card

In this test, we sent frames from the SDR and received them with the UBIQUITI XR7 Wi-Fi card. The channel bandwidth used was 20 MHz, because we wanted the highest possible throughput. XR7 card can only operate at 20 MHz in 2 channels (channel 5 and 6). We decided to use channel 6, with a carrier frequency of 773 MHz. Besides, all measurements were performed with a packet size of 504 bytes and a rate of 200 packets per second - which results in a data rate of 806.4 kbit/s - and the distance between the devices was approximately 1.4 meters.

The Figures 5.10 and 5.11 present the results obtained. As we can see from the figures, we can conclude that the PDR curve approaches one for higher transmission powers and that higher bitrates suffer from higher packet loss, as expected.

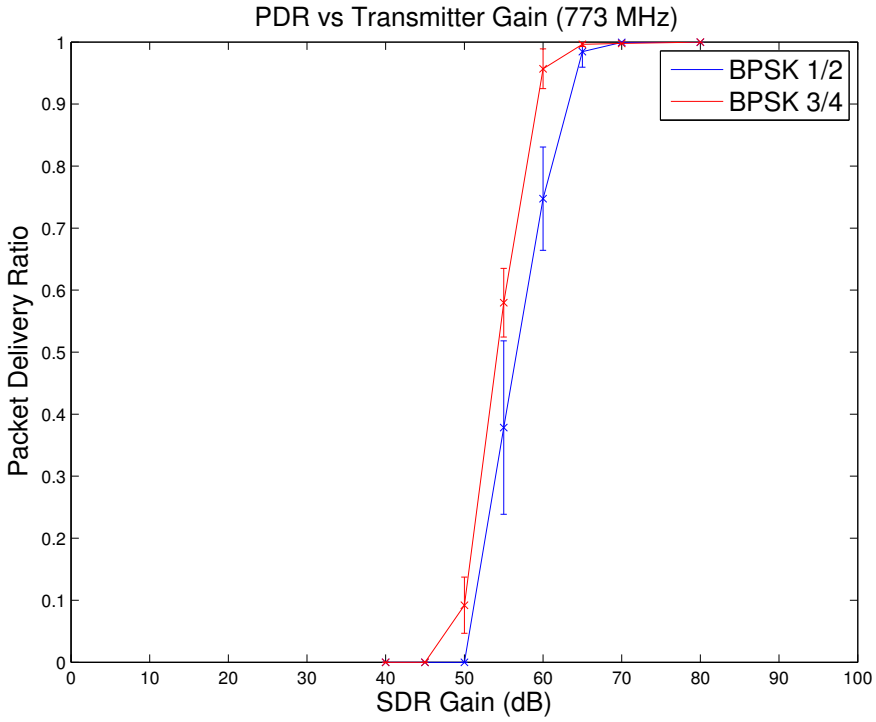


Figure 5.10: PDR of IEEE 802.11g frames between SDR and Ubiquity XR7 Wi-Fi card @ 773 MHz - BPSK.

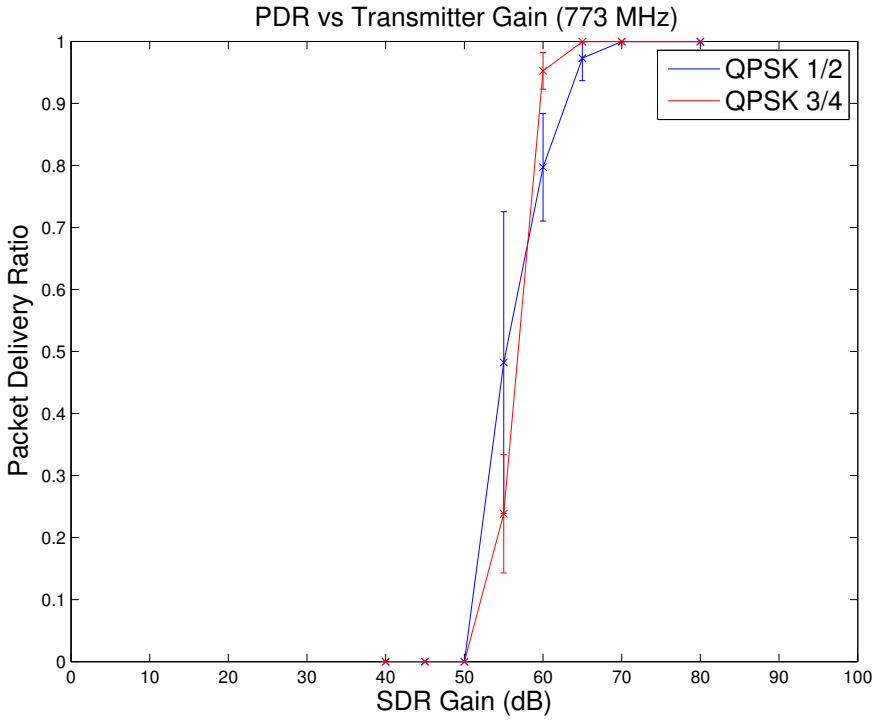


Figure 5.11: PDR of IEEE 802.11g frames between SDR and Ubiquity XR7 Wi-Fi card @ 773 MHz - QPSK.

5.2.2.2 PDR between SDR and RouterBOARD R52n-M Wi-Fi card

In this test, we sent frames from the SDR and received them with the RouterBOARD R52n-M Wi-Fi card with a frequency of 5.240 (channel 48) and 5.825 (channel 165) GHz. The measurements performed on channel 48 had a packet size of 504 bytes and a rate of 200 packets per second - which results in a data rate of 100.8 kbit/s - and the distance between the devices was approximately 1.4 meters. The measurements performed on channel 165 had a packet size of 381 bytes and a rate of 200 packets per second - which results in a data rate of 76.2 kbit/s - and the distance between the devices was approximately 0.8 meters. Besides, the channel bandwidth used was 20 MHz for each frequency.

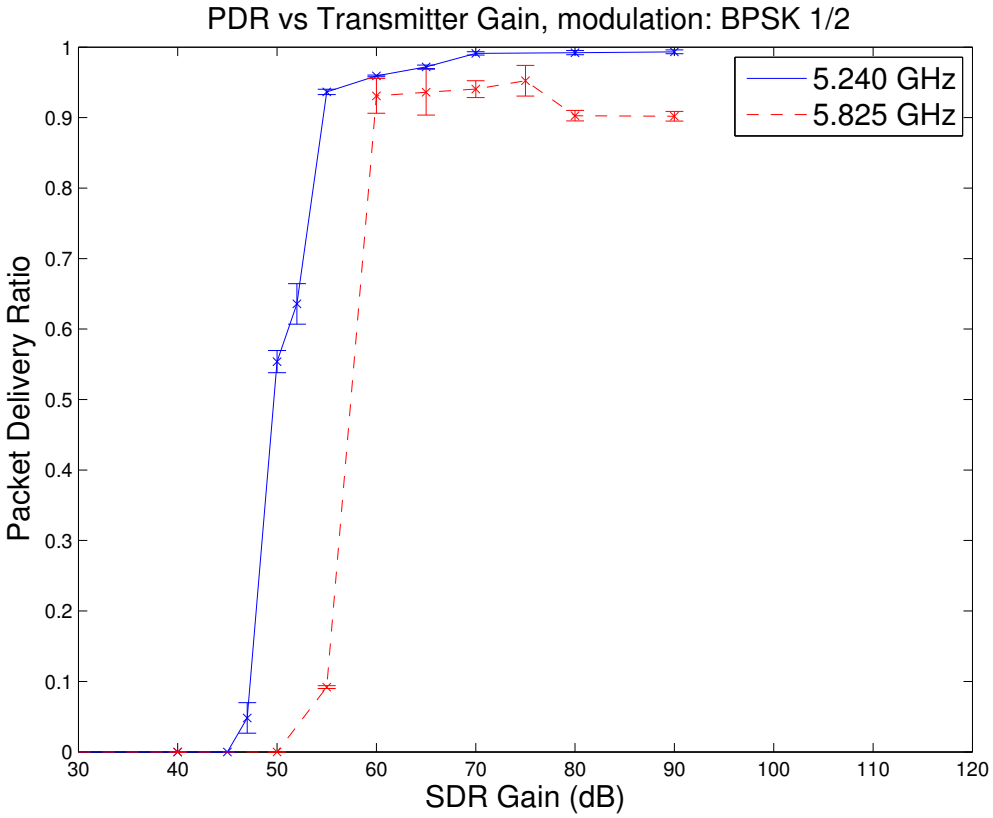


Figure 5.12: PDR of IEEE 802.11a frames sent from the SDR and received by the the RouterBOARD R52n-M Wi-Fi card.

In the Figure 5.12 it is shown the results obtained. It should also be noted that for a frequency of 5.240 GHz, we had a higher distance between the devices, which is reflected in the graph, because we needed a higher gain to receive packets in this channel. Besides, in this channel, there were other devices in the medium, which could affect the performance of the SDR, especially because of the lack of CSMA/CA described before. Despite that, this test was conducted in order to conclude if the performance of the SDR is in fact limited when other devices are present in the medium. We achieved PDR values around 90% for higher transmission powers, which is quite

acceptable, despite the presence of beacons and small traffic loads.

On the other hand, on channel 165, there weren't other devices in the medium. Therefore, we can conclude that the PDR achieved is quite acceptable, since the results obtained showed that the PDR curve approaches one for higher transmission powers.

5.2.3 Tests with unidirectional communications

After conducting successful tests with broadcast communications in multiple frequency bands, most on the 2.4 and 5 GHz band, it was decided to test unidirectional communications between SDRs and between a SDR and Wi-Fi cards. In the Figures 4.6 and 4.7, in the Section 5.2.1 are illustrated the schemes of the proposed tests with the SDRs and Wi-Fi cards. In order to establish an unidirectional communication between the Wi-Fi card and the SDR, we needed to introduce static routing and Address Resolution Protocol (ARP) entries in both devices; disable ACKs retries from the packets sent from the Wi-Fi card, because the transceiver implementation can not respond to ACKs in time and disable the RTS/CTS mechanism for the same reason, by setting the RTS/CTS threshold in the Wi-Fi card to infinity, i.e., 2437 which is the highest value acceptable.

On the other hand, in the transceiver implementation, the packets transmitted by the SDR had the Wi-Fi card's MAC address as the destination address and the Basic Service Set Identifier (BSSID) of the ad hoc network as the Basic Service Set (BSS) MAC address. It should be noted that, in order to establish an unidirectional communication between the UBIQUITI XR7 Wi-Fi card and the SDR, we also needed to change the modulation of the packets sent by the Wi-Fi card to OFDM, because the packets were CCK modulated and the transceiver implementation does not support this modulation - only IEEE 802.11 OFDM frames are supported. Thus, after changing the modulation to OFDM, we were able to establish the unidirectional communication between the SDR and the UBIQUITI XR7 Wi-Fi card.

In this section, we evaluate the performance of unidirectional communications. Measurements for 5.825 and 5.240 GHz, and 773 and 700 MHz were performed with different SDR gains. Table 5.1 presents the SDR receiver and transmitter gains for each tested frequency. Besides, the channel bandwidth used was 20 MHz for all the tested frequencies.

Table 5.1: SDR Receiver and Transmitter Gains for over-the-air communications

Frequency (Hz)	5.825 G	5.240 G	773 M	700 M
SDR Transmitter Gain (dB)	70			80
SDR Receiver Gain (dB)	20		6	

5.2.3.1 RTT & Packet Loss

In this test, we used the ping command to send frames from the SDR to measure the RTT and the Packet Loss between a SDR and the RouterBOARD R52n-M Wi-Fi card. However, we measured the RTT between two SDR boards for a frequency of 700 MHz. Figure 5.13 represents the measure of the RTT and Figure 5.14 shows the correspondent values of the packet loss.

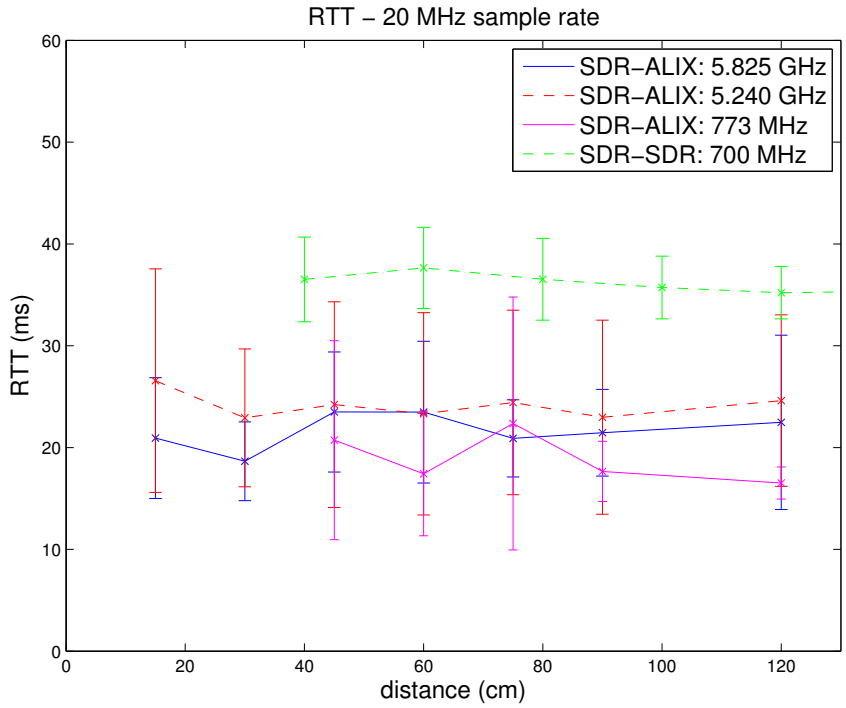


Figure 5.13: RTT.

For a frequency of 5.825 GHz, 5.240 GHz and 773 MHz, we experienced some oscillations throughout all the distances, which may be due to the fact that the packets are sent by a virtual interface created in the Linux environment that it is connected to the GNURadio companion, which forwards the packets to the SDR so that they are transmitted over-the-air. Then, the Wi-Fi card responds with the ping reply and, when these packets are received by the SDR, they perform the same path detailed before in the opposite direction. Represented by the confidence interval, we can see some variations in each distance, but the RTT remains stable throughout all the distances, being between 18 and 30 ms. These values of RTT are higher when compared to traditional radios, but are lower when compared with a typical Internet connection. Besides, for a frequency of 5.825 GHz, the packet loss we experienced was very low: about 5%, increasing to 16% as the distance increased to 60 cm. This packet loss can be eventually decreased with an automatic gain control, which optimizes the SNR at the receiver.

For a frequency of 5.240 GHz, we can see some variations in each distance, which may be due to the fact that the SDR was not the only device in the medium in this channel frequency, as it was mentioned before. On the other hand, for a frequency of 773 MHz, the packet loss for

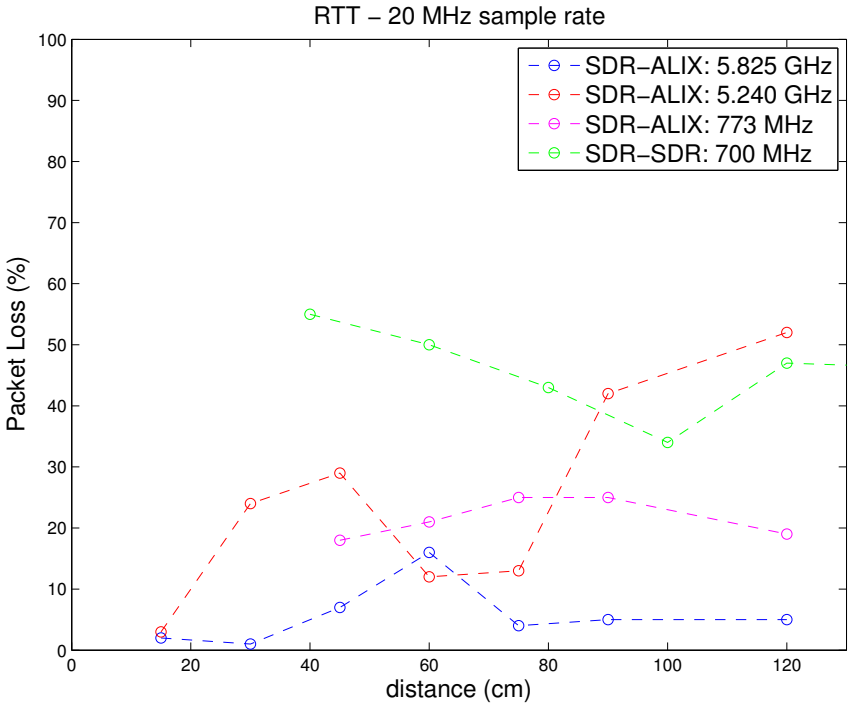


Figure 5.14: Packet Loss.

each distance was between 19% and 25%. Besides, for a frequency of 700 MHz, we achieved an average RTT around 37ms and the packet loss remained stable and around 50%. These values of RTT are higher, since the packets are exchanged by two SDR boards, which generate a higher latency on the communication, because the signal processing on both devices is implemented in software. Therefore, it is expected a higher RTT between two SDR boards, as it was experienced.

In conclusion, the values of RTT for a frequency of 773 MHz were considerably lower than the ones obtained at a frequency of 5.240 GHz, where the SDR was not the only device in the medium.

5.2.3.2 RTT & Packet Loss for higher distances

In this test, we measured the RTT at much higher distances than in the previous RTT test. All measurements were performed on channel 165 (5.825 GHz) with a SDR transmitter gain of 80 dB, a SDR receiver gain of 30 dB and the transmitter gain of the RouterBOARD R52n-M Wi-Fi card was 20 dBm. Besides, the channel bandwidth used was 20 MHz.

Figure 5.15 represents the measures of RTT. We experienced some oscillations while increasing the distance in the RTT and in the packet loss. The lowest average value obtained for the RTT was about 18 ms, for a distance of 4 meters, and we also obtained the lowest value o the packet loss for the same distance, being around 1%.

On the other hand, the highest average value for the RTT was about 25.6 ms, for a distance of 8 meters. For this distance, a packet loss of 40% was experienced. We also measured a RTT of

22.2 ms for a distance of 12 meters, and the packet Loss was 95%, and the packet loss was 100% for a distance of 15 meters. If there was an automatic gain control mechanism, we wouldn't have experienced these higher values of packet loss.

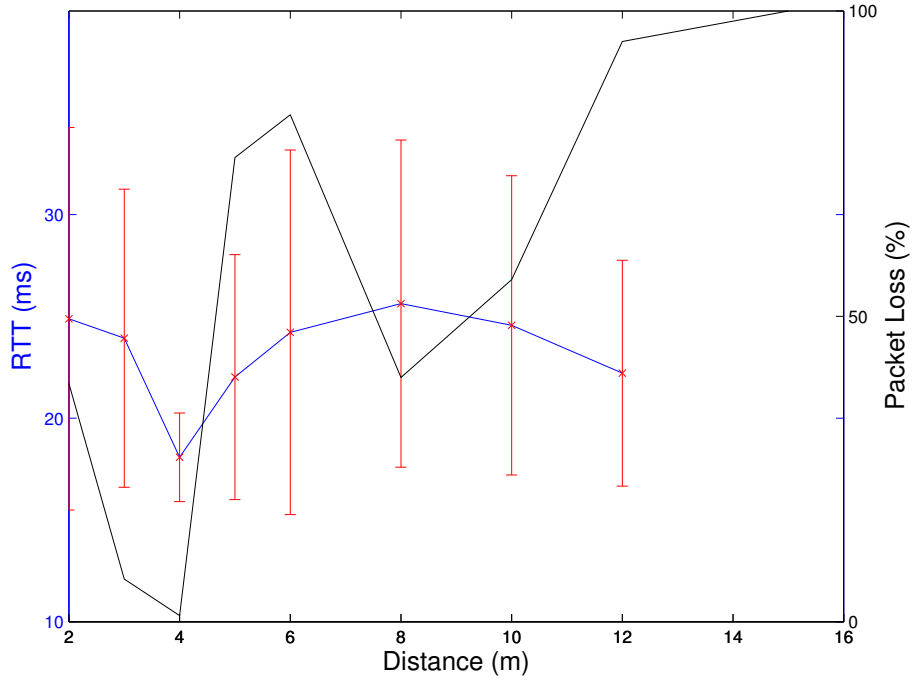


Figure 5.15: RTT & Packet Loss @ 5.825 GHz.

5.2.3.3 Jitter

Jitter between a SDR board and a Wi-Fi card

In this test, we used the Iperf tool to measure the jitter between the SDR and the RouterBOARD R52n-M Wi-Fi card and between the SDR and the UBIQUITI XR7 Wi-Fi card, in both directions. The distance between the devices was approximately 1.4 meters. It should also be noted that a bandwidth of 0.5 Mbit/s and a packet size of 281 bytes were chosen to send the packets through the SDR, but we chose a bandwidth of 0.2 Mbit/s and a packet size of 179 bytes instead when the Wi-Fi card was the transmitter, since the SDR experiences a higher packet loss when it is receiving packets. Even with these changes, for a frequency of 5.825 GHz, the packet loss experienced by the SDR when it is receiving was about 45%, instead of about 0.68% when it is transmitting, even in higher bandwidths. On the other hand, for a frequency of 773 MHz, the packet loss experienced by the SDR when it is receiving was about 50%, instead of about 0.25%, when the it is transmitting. Figure 5.16 and Figure 5.17 refer to the jitter using Iperf.

For a frequency of 5.825 GHz, we can see that jitter remains stable and lower than 0.5 ms throughout all the distances measured when the SDR is transmitting to the Wi-Fi card. However, when the SDR is receiving the packets, the jitter remains stable but its value is higher that in

the opposite direction. In this situation, values around 3.5 ms were experienced through all the distances.

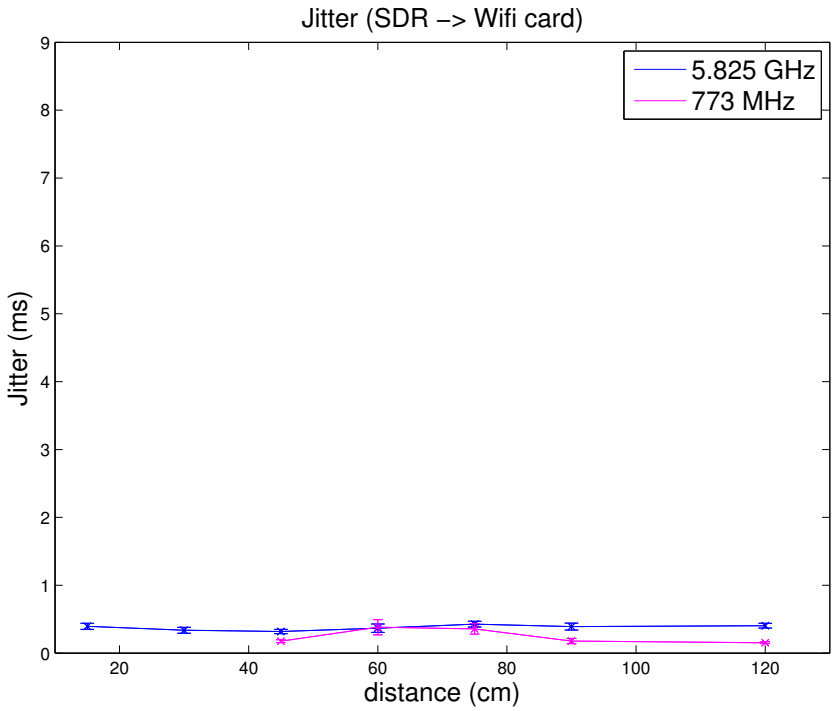


Figure 5.16: Jitter: SDR -> Wi-Fi card

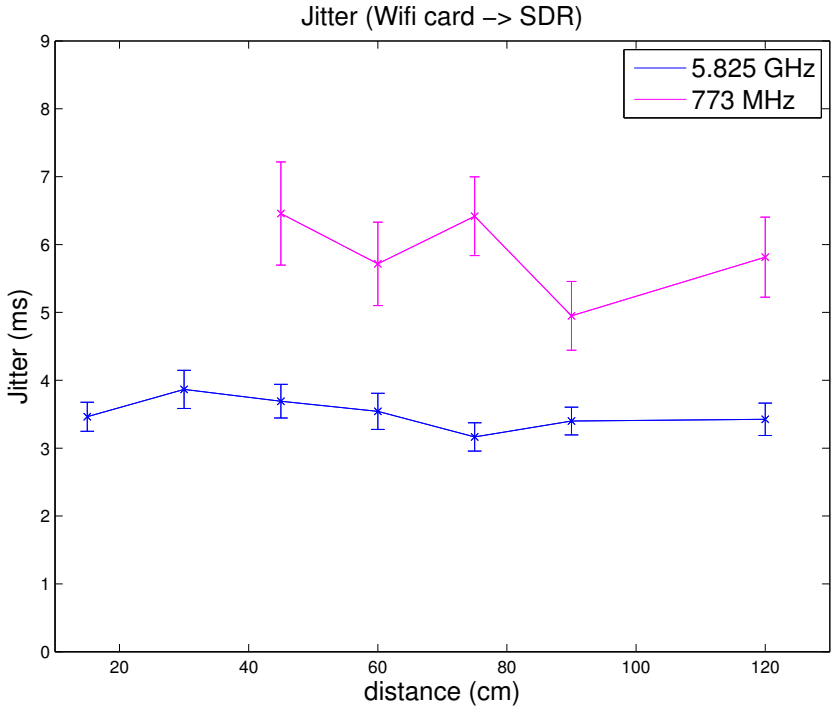


Figure 5.17: Jitter: Wi-Fi card -> SDR.

For a frequency of 773 MHz, we can see that jitter experiences some oscillations but it is always lower than 0.5 ms throughout all the distances measured when the SDR is transmitting to the Wi-Fi card. Besides, by increasing the distance, jitter becomes lower and values around 0.2 ms were experienced until the maximum distance. However, when the SDR is receiving the packets, the jitter is much higher that in the opposite direction. In this situation, some oscillations were experienced but the jitter is always lower than 8 ms throughout all the distances.

Jitter between two SDR boards

Figure 5.18 refers to the jitter using Iperf between the two SDRs. All measurements were performed with the SDR transmitter sending packets, with a packet size of 384 bytes.

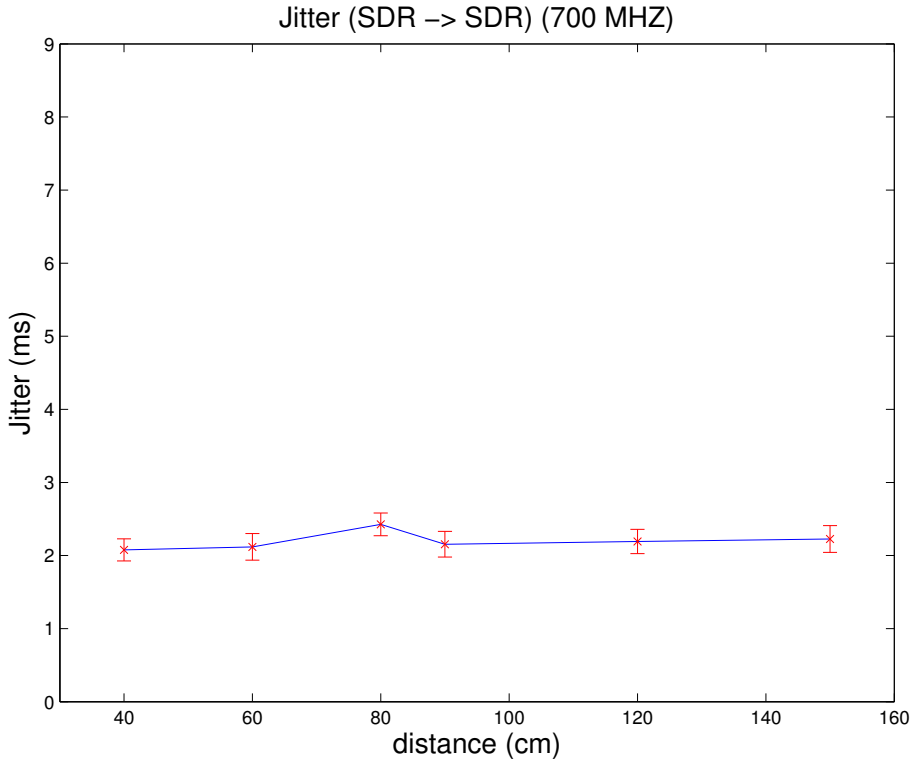


Figure 5.18: Jitter: SDR -> SDR.

As we can see, jitter remains stable about 2.2 ms throughout all the tested distances. Besides, with these results, we can conclude that the jitter between two SDR boards is quite low and acceptable and also that the jitter between a SDR and a commercial wireless card is higher than with two SDR boards.

5.2.3.4 UDP Throughput

UDP Throughput between a SDR board and a Wi-Fi card

In this test, we used the Iperf tool to measure the maximum UDP throughput that can be achieved between a SDR and the RouterBOARD R52n-M Wi-Fi card and between a SDR and the UBIQUITI XR7 Wi-Fi card, in both directions. The distance between the devices was approximately 1.4 meters and the packet size of 281 bytes was chosen when the SDR was the receiver, and a packet size of 384 bytes was chosen instead when the SDR was transmitting. Figure 5.19 and Figure 5.20 refer to the UDP throughput using Iperf.

For a frequency of 5.825 GHz, the SDR can transmit packets with a maximum bandwidth of 1.82 Mbit/s, but can only receive packets with a maximum bandwidth of 84 kbit/s for a packet size of 281 bytes. The SDR remains much more stable when it is transmitting, since the UDP connection between the SDR and the Wi-Fi card sometimes fails when the SDR is receiving the packets, which contributes to a lower bandwidth in this situation. It should also be noted that the receiver part of the implementation is more complex and needs more processing speed than the transmitter part in order to achieve a satisfactory performance. Therefore, these reasons contribute to lower values when the SDR is the receiver.

For a frequency of 773 MHz, we can see from the figures that the SDR can transmit packets with a maximum bandwidth of 1.83 Mbit/s, but can only receive packets with a maximum bandwidth of 84.4 kbit/s. As we can see, the SDR remains much more stable when it is transmitting. The same reasons explained before with a frequency of 5.825 GHz also affected the performance of the SDR in this test, contributing to more unstable values when the SDR is the receiver.

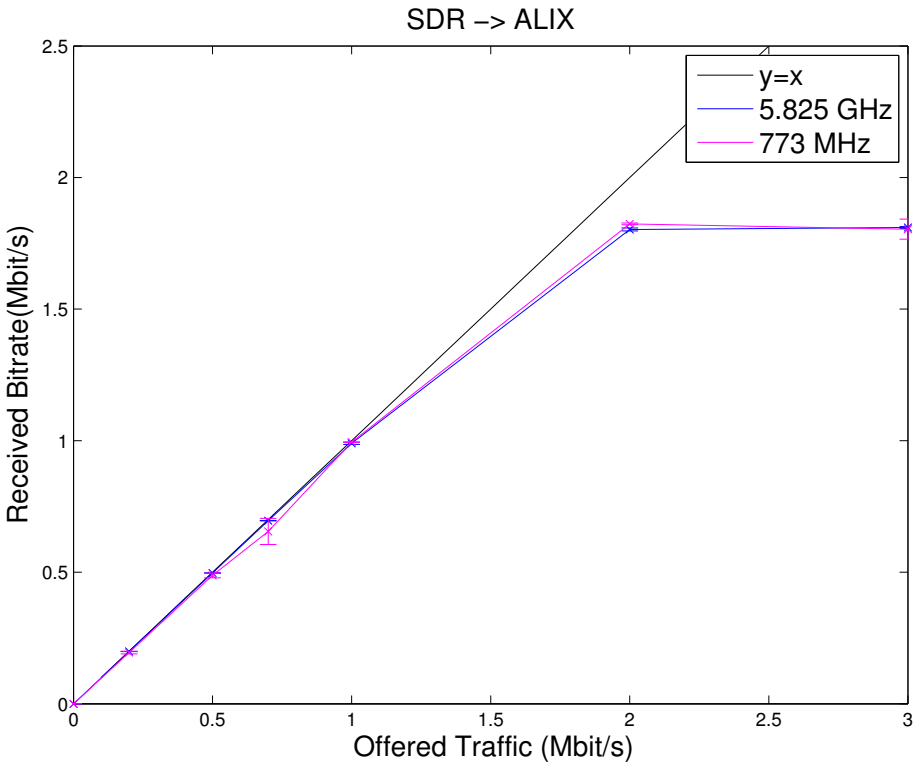


Figure 5.19: UDP Throughput: SDR -> Wi-Fi card

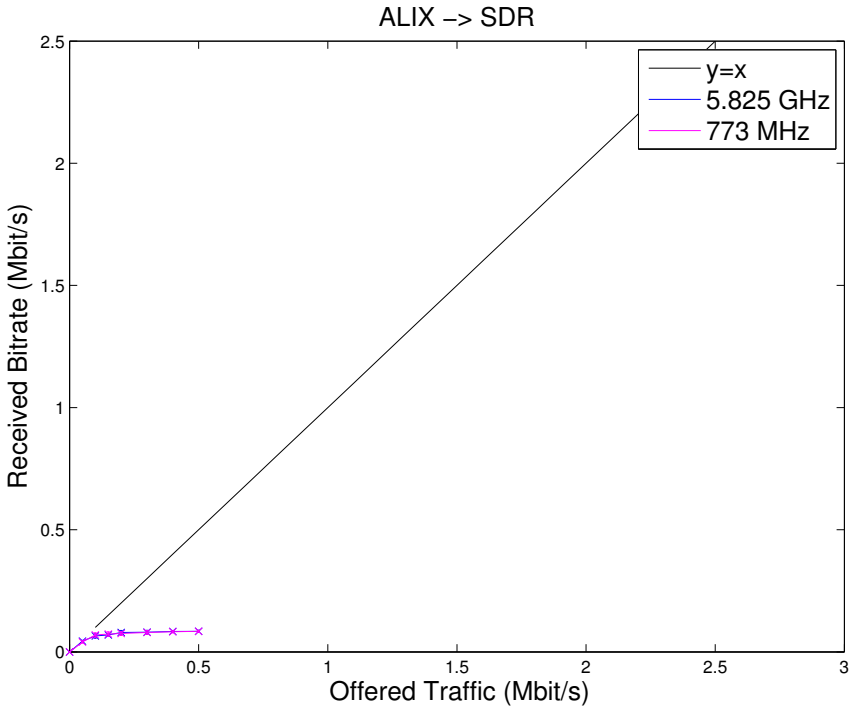


Figure 5.20: UDP Throughput: Wi-Fi card -> SDR.

UDP Throughput between two SDR boards

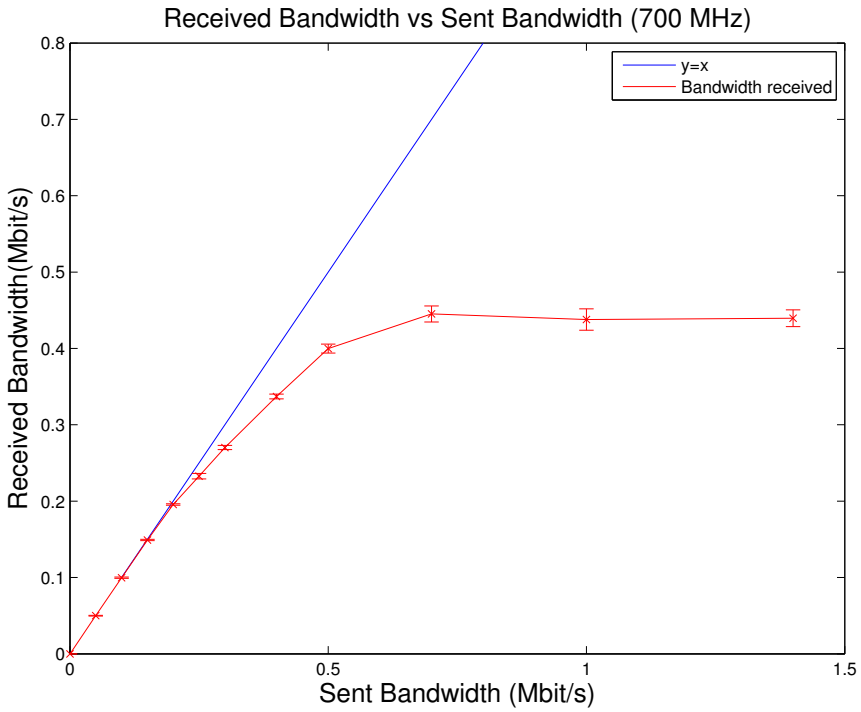


Figure 5.21: UDP Throughput: SDR -> SDR

In this test, we used the Iperf tool to measure the maximum UDP bandwidth that can be achieved between two SDR boards. All packets sent from Iperf had a packet size of 384 bytes, the distance between the devices was about 0.85 meters, and the modulation used was BPSK $\frac{1}{2}$.

Figure 5.21 refers to the UDP bandwidth using Iperf. As we can see, the SDR can receive packets with a maximum bandwidth of 440 kbit/s. Besides, for bandwidths below 300 kbit/s, the packet loss is very low and around 10%. However, when a larger bandwidth is requested, the packet loss increases. For example, for a bandwidth of 700 kbit/s, a packet loss of 36% was experienced, because it overcomes the maximum bandwidth that we achieved (440kbit/s).

5.2.3.5 TCP Throughput

In this test, we used the Iperf tool to measure the maximum TCP bandwidth that can be achieved between two SDR boards. All measurements were performed with a frequency of 700 MHz. Figure 5.22 refers to the TCP bandwidth using Iperf. As we can see, the average TCP traffic between the SDRs was about 50 kbit/s. For a distance of 0.85 meters, the maximum TCP bandwidth of 56 kbit/s was achieved. These low throughput results can be due to the fact that a TCP connection had to be established between both devices. Therefore, ACKs were exchanged between the SDR boards, which caused some interruptions and a large period of time between transmissions of the frames. These reasons led to the low throughputs experienced.

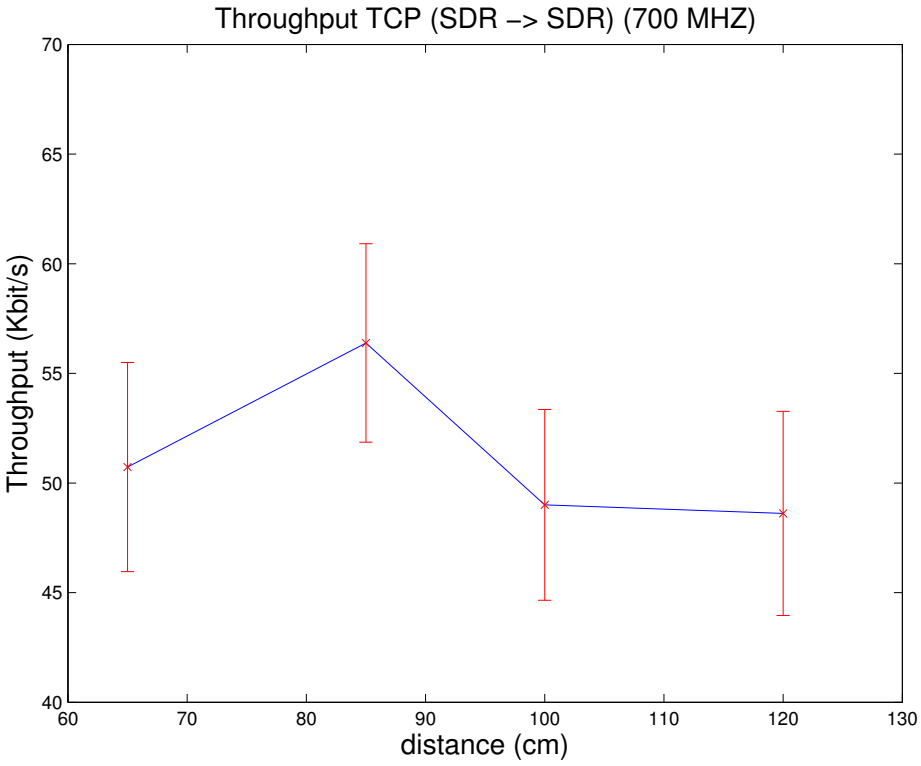


Figure 5.22: TCP throughput: SDR -> SDR.

5.3 Air Environment @ Alfeite Lisbon Naval Base

In this air environment, the evaluation of the implementation was based on the metrics that are described in the Section 4.3. We tested the behavior of multiple frequency bands with different antennas. Throughput, RTT, packet loss and jitter were analyzed.

5.3.1 RTT & Packet Loss

In this test, we used the dipole antenna for a frequency of 382.5 MHz and the loop antenna for a frequency of 900 MHz. Table 5.2 show the obtained results for each antenna. As we can see, we achieved an average RTT around 45 ms for each antenna, although the packet loss was higher with the dipole antenna around 56%.

Table 5.2: RTT - Channel Bandwidth: 5 MHz

Antenna	Loop	Dipole
Frequency (MHz)	900	382.5
Modulation	BPSK 1/2	
Distance (m)	41	
Average RTT (ms)	46.302	45.135
Packet Loss (%)	16	56

5.3.1.1 UDP Throughput & Jitter

In this test, we used the dipole and the loop antennas to measure the UDP Throughput and the jitter that can be achieved in all tested modulations. Table 5.3 show the obtained results. As we can see, we achieved similar results with both antennas, but it should be highlighted that for a 16 QAM $\frac{1}{2}$ modulation and with loop antennas, we obtained an average throughput of 511.92 kbit/s.

Table 5.3: UDP Throughput & Jitter - Channel Bandwidth: 5 MHz

Antenna	Loop			Dipole		
Frequency (MHz)	900			382.5		
Distance (m)	41					
Modulation	BPSK 1/2	QPSK 1/2	16 QAM 1/2	BPSK 1/2	QPSK 1/2	16 QAM 1/2
Average Throughput (kbit/s)	390.41	477.65	511.92	391.12	455.82	259.17
Average Jitter (ms)	2.334	1.951	1.706	2.406	2.181	2.765

5.4 Underwater Environment

5.4.1 Freshwater Environment

After the optimization of the implementation to new wireless scenarios, a new testbed was implemented in a freshwater environment. The evaluation was based on the parameters described above in Section 4.3 and we tested the performance of the implementation at low frequencies. Thus, we decided to conduct tests with a frequency of 700, 450, 300, 200 and 100 MHz, at different channel bandwidths, such as 5, 10 and 20 MHz. It should also be noted that we needed to manually adjust the SDR transmitter and receiver gains for each frequency and for each channel bandwidth. Table 5.4 presents the SDR receiver and transmitter gains for each frequency tested. However, when we tested higher distances, such as 4 and 5 meters, we increased the SDR transmitter gain to 100 dB. We describe and present the results obtained through underwater communications between the two SDR boards further in this section.

Table 5.4: SDR Receiver and Transmitter Gains for underwater communications

Frequency (MHz)	700			450			300			200			100		
Sample rate (MHz)	20	10	5	20	10	5	20	10	5	20	10	5	20	10	5
SDR Receiver Gain (dB)	6			30						60		30	60		30
SDR Transmitter Gain (dB)	95			90		95	90								

Finally, the measurement results were analyzed and compared with theoretical and simulation models. The freshwater testbed is shown in Figure 5.23.

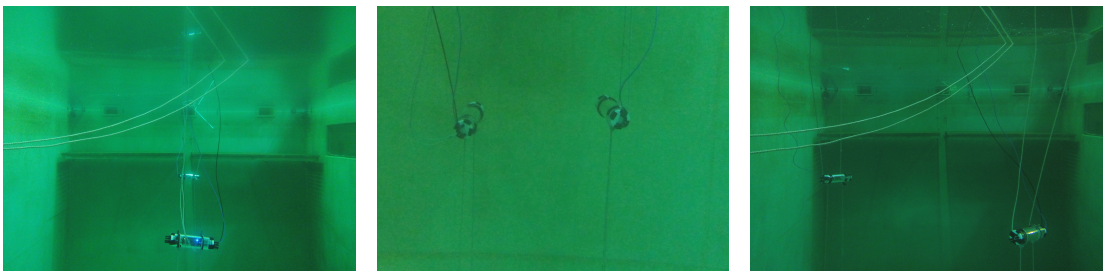


Figure 5.23: Views of the underwater testbed.

5.4.1.1 RTT & Packet Loss

We used the ping command to measure the RTT that can be achieved between the two submerged SDR boards for multiple operating frequencies. Figures 5.24, 5.25 and 5.26 show the obtained values of RTT for the three channel bandwidths tested and Figures 5.27, 5.28 and 5.29 show the corresponding packet loss.

As we can see, for a channel bandwidth of 20 MHz, for a frequency of 200 MHz, the RTT remains stable and around 35 ms throughout all the tested distances and, for a frequency of 100 MHz, the RTT increases with the increase of distance, always being lower than 50 ms, since we were unable to receive the ping replies from the SDR for a distance of 1.6 meters.

Regarding the results obtained for a channel bandwidth of 10 MHz, we achieved a 100% packet loss for a frequency of 700 MHz for the distances tested. As we can see, for frequencies of 450 MHz, 300 MHz and 100 MHz, the RTT remains stable and around 38 ms. For a frequency of 200 MHz, the RTT remains stable and around 37 ms until a distance of 2 meters, but then begins to increase. For instance, for a distance of 4 meters, the RTT reaches a maximum value of 40.8 ms. On the other hand, we experienced a high packet loss for frequencies of 450, 300 and 100 MHz, for distances higher than 1.6 meters, reaching 100%, but for 200 MHz, the packet loss was considerably higher for low distances, when compared to other frequencies, though it only reaches 100% for a distance of 5 meters.

On the other hand, for a channel bandwidth of 5 MHz, we can see that, for all frequencies tested, we experienced RTT values lower than 42 ms for distances lower than 1.6 meters. Besides, we experienced a low packet loss for all the tested frequencies. The packet loss is lower than 10% for distances lower than 1.6 meters.

Finally, we observed that the SDR implementation achieves lower RTT values and low packet losses for a channel bandwidth of 5 MHz, when compared to the results obtained with a channel bandwidth of 10 and 20 MHz.

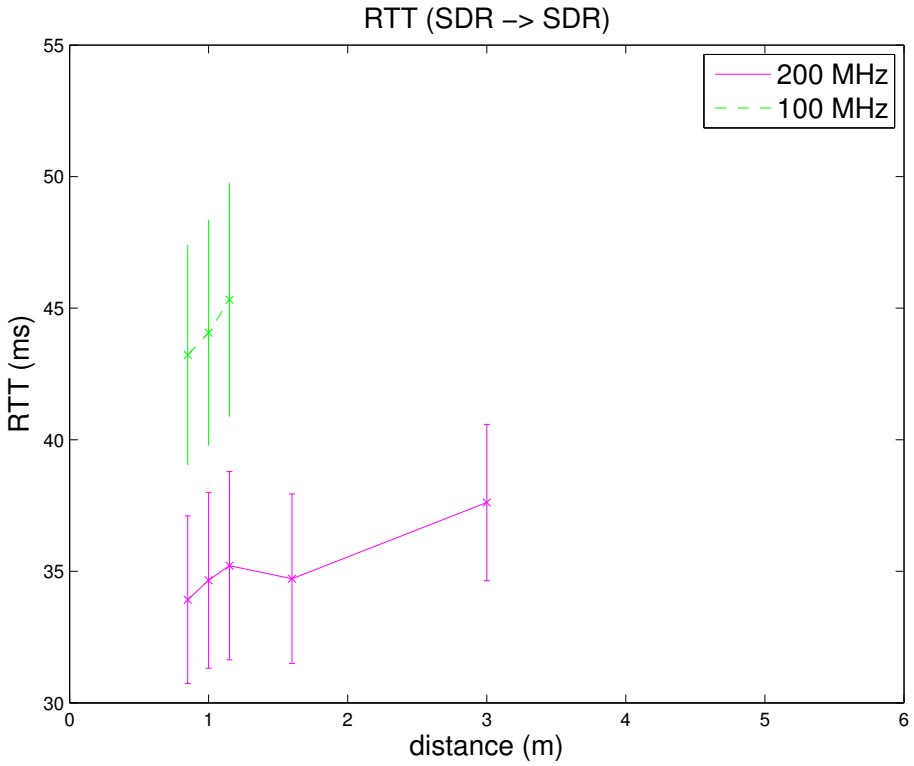


Figure 5.24: RTT - Channel Bandwidth: 20 MHz

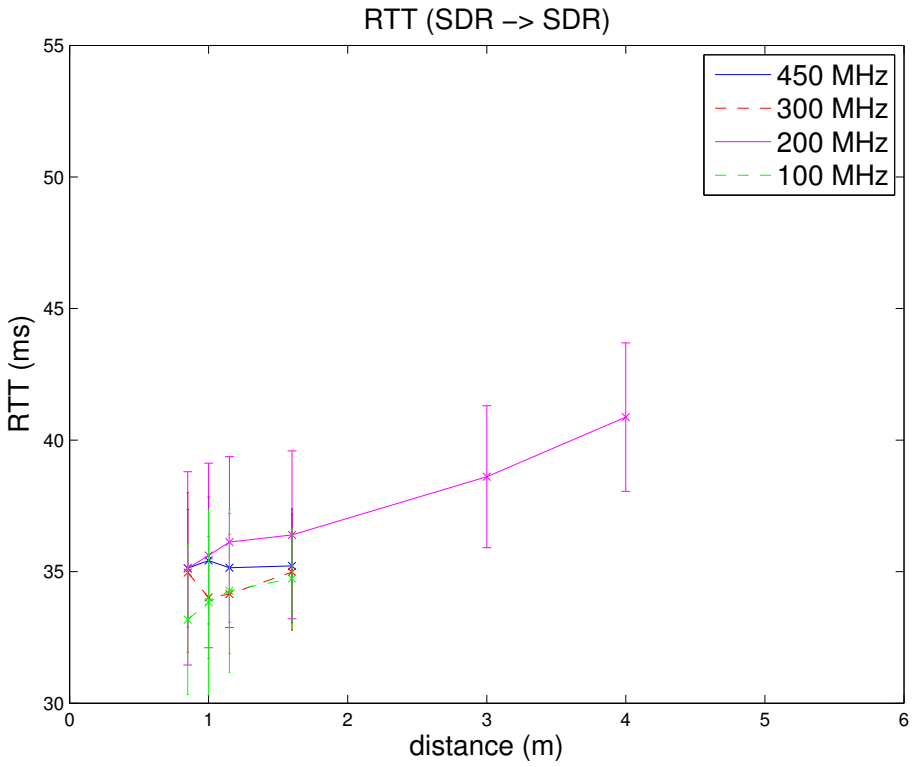


Figure 5.25: RTT - Channel Bandwidth: 10 MHz

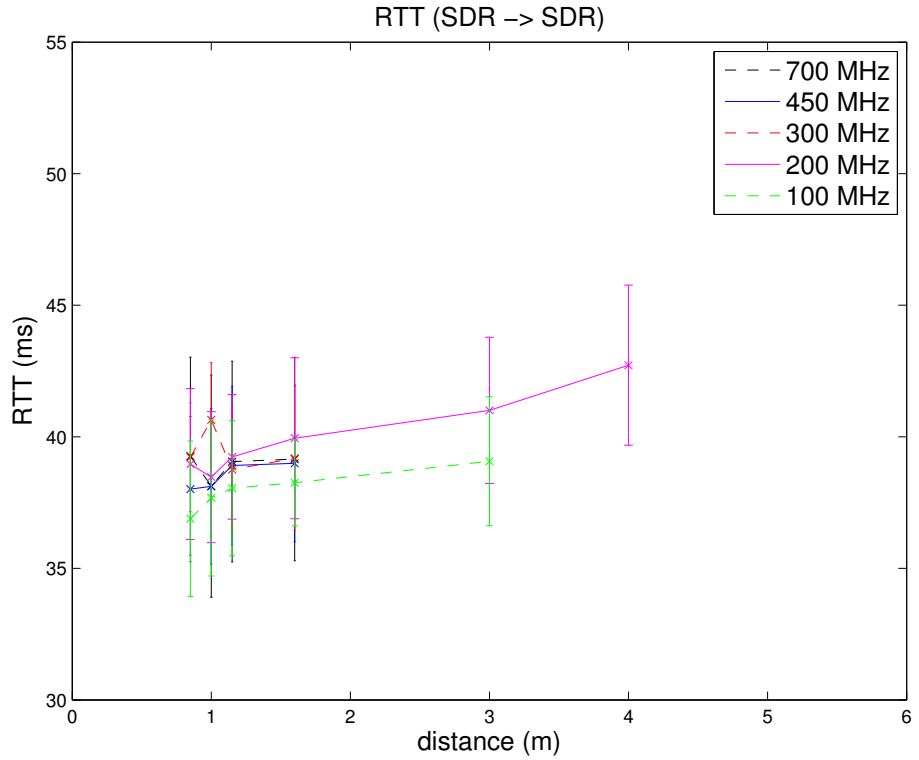


Figure 5.26: RTT - Channel Bandwidth: 5 MHz

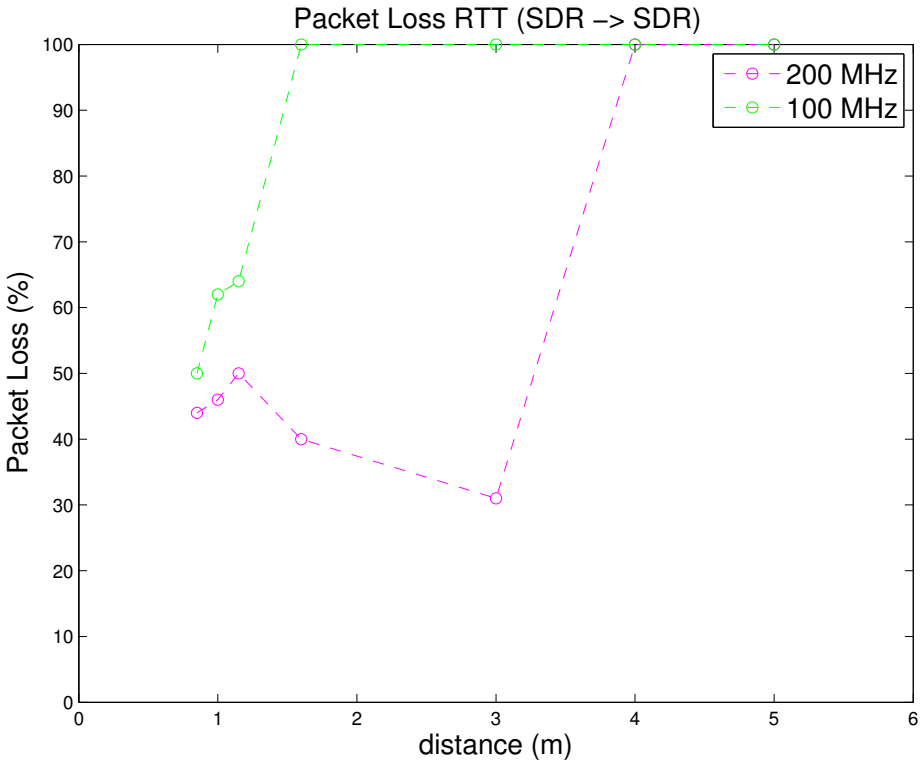


Figure 5.27: RTT Packet Loss - Channel Bandwidth: 20 MHz

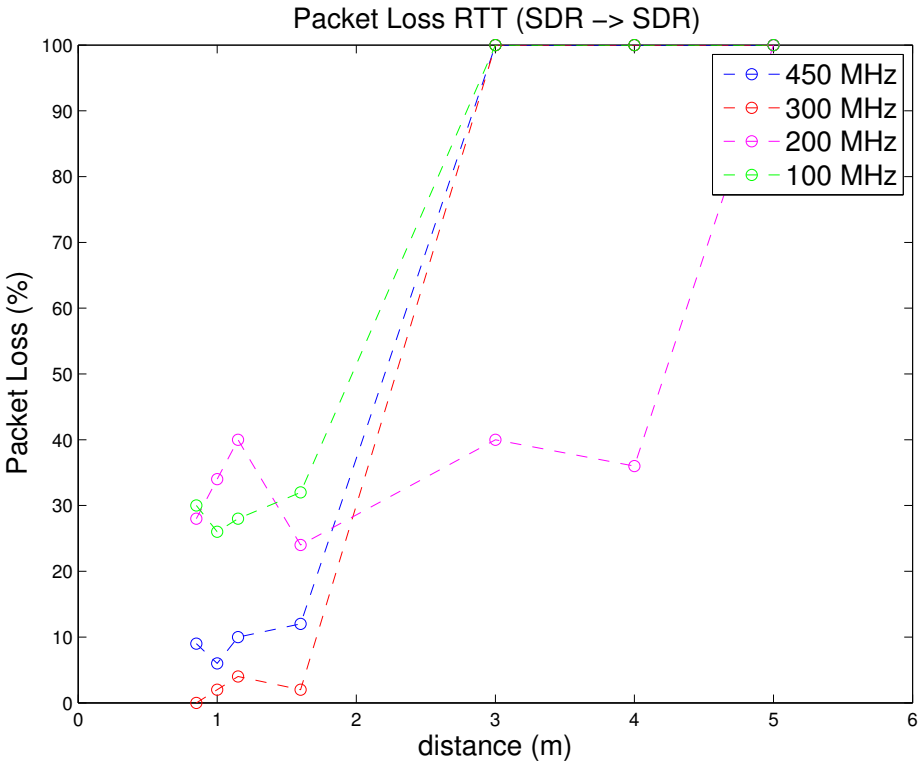


Figure 5.28: RTT Packet Loss - Channel Bandwidth: 10 MHz

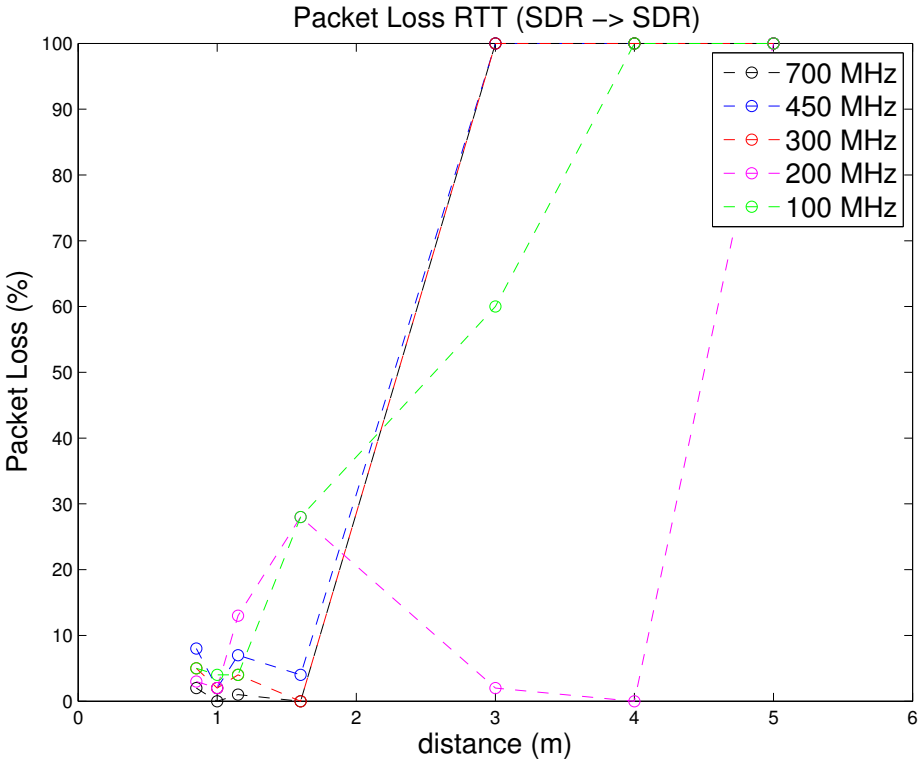


Figure 5.29: RTT Packet Loss - Channel Bandwidth: 5 MHz

5.4.1.2 Jitter

In this test, we used the Iperf tool to measure the jitter that can be achieved between the two submerged SDR boards. Figures 5.30, 5.31 and 5.32 show the obtained jitter for the three channel bandwidths tested.

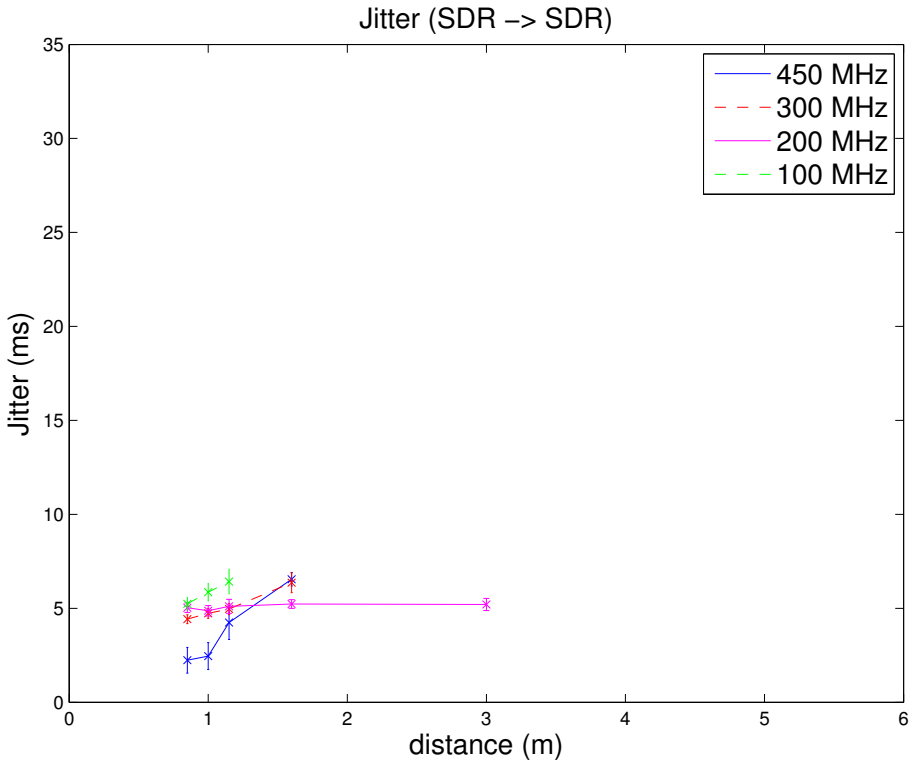


Figure 5.30: Jitter - Channel Bandwidth: 20 MHz

For a channel bandwidth of 20 MHz, we can see that jitter remains lower than 8 ms for all frequencies. It should also be noted that, for a distance of 3 meters, we achieved a jitter around 5 ms, but we only obtained measurements for a frequency of 200 MHz.

Regarding the results obtained for a channel bandwidth of 10 MHz, we can see that jitter remains lower than 8 ms and stable for the multiple frequencies tested. On the other hand, for a channel bandwidth of 5 MHz, we can see that jitter remains lower than 3 ms and stable for all the distances tested, even at 4 and 5 meters.

In conclusion, we achieved lower jitter values when a channel bandwidth of 5 MHz is used, similarly to the results obtained with the RTT.

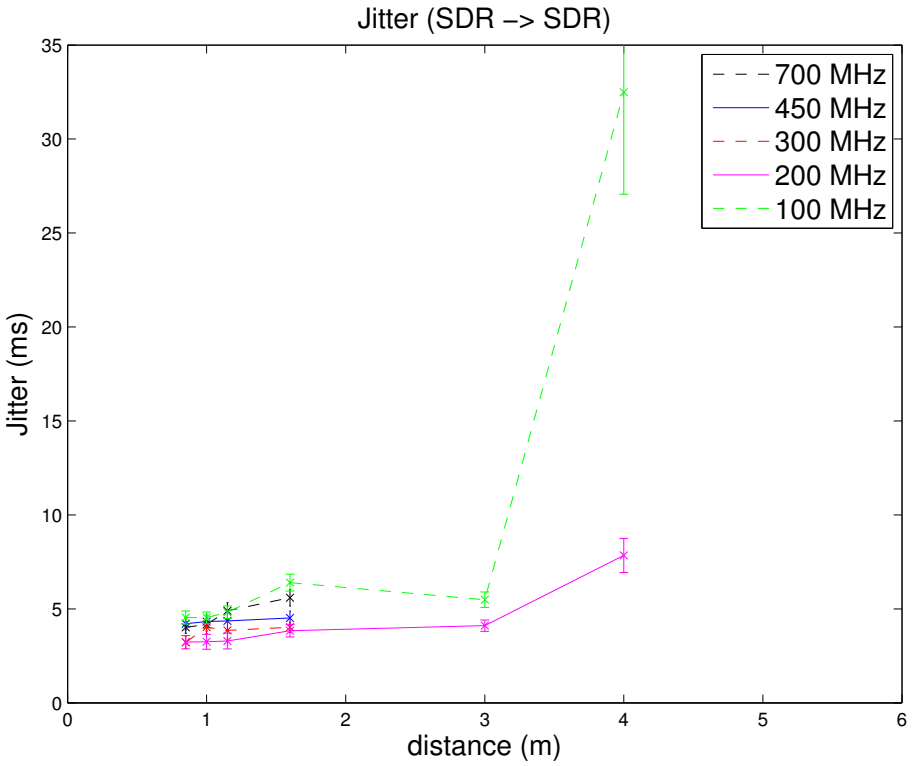


Figure 5.31: Jitter - Channel Bandwidth: 10 MHz

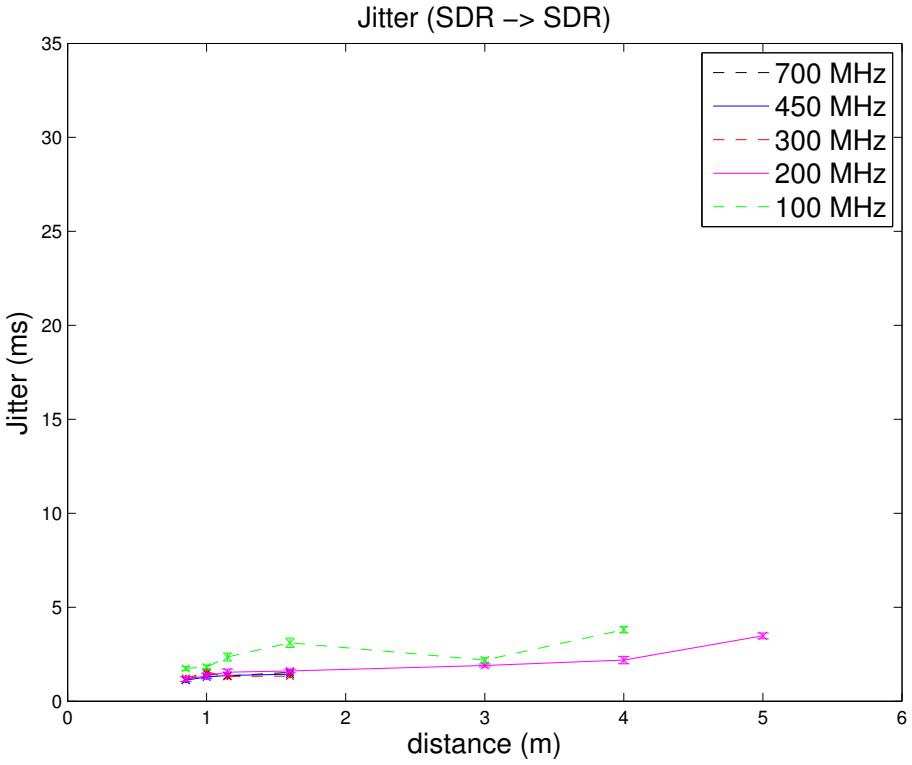


Figure 5.32: Jitter - Channel Bandwidth: 5 MHz

5.4.1.3 UDP Throughput

In this test, we used the Iperf tool to measure the UDP throughput that can be achieved between the two submerged SDR boards. Figures 5.33, 5.34 and 5.35 show the obtained throughput values for the three channel bandwidths tested.

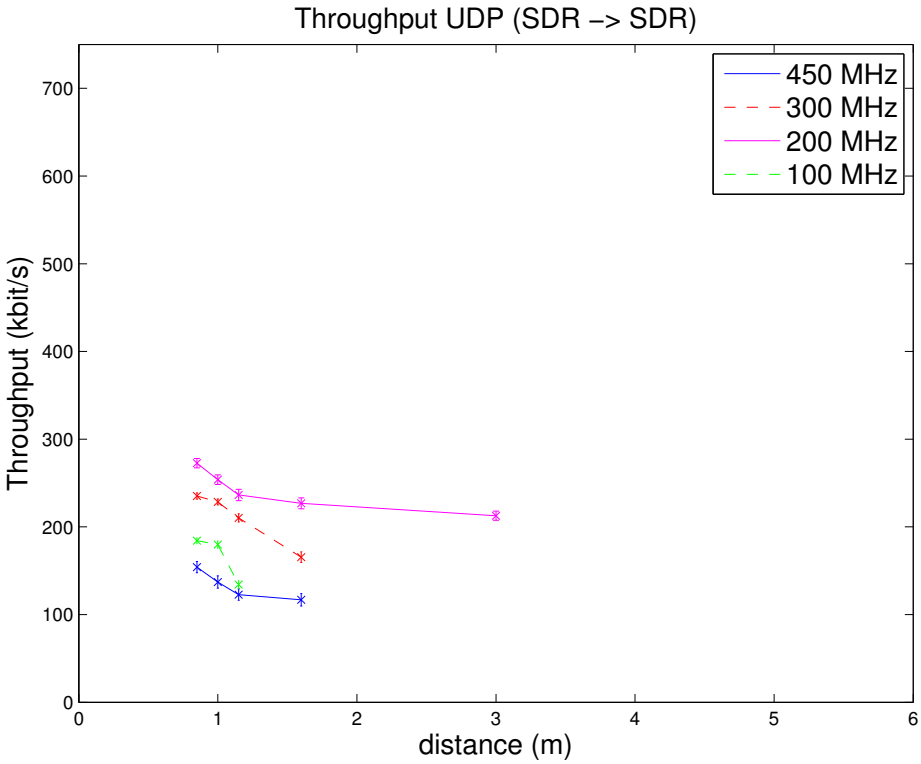


Figure 5.33: UDP Throughput - Channel Bandwidth: 20 MHz

As we can see, we achieved a maximum throughput of around 272 kbit/s for a channel bandwidth of 20MHz and a frequency of 200 MHz, for a distance of 0.85 meters. These results are lower when compared to the throughputs obtained by P. Freitas in [15], but we achieved a throughput of 212.7 kbit/s for a distance of 3 meters and he only achieve throughputs until a distance of 2.15 meters.

It should also be noted that, for a channel bandwidth of 10 MHz, we increased the communication distance between the SDR boards, since we were able to communicate at 4 meters, achieving a throughput of 70.62 kbit/s with a frequency of 200 MHz. Besides, we achieved an average throughput between 300 and 500 kbit/s for all the frequencies tested, until a distance of 2 meters.

Finally, for a channel bandwidth of 5 MHz, we were able to communicate at 5 meters, which is a considerable improvement in RF underwater communications. Thus, we achieved a throughput of 309.6 kbit/s, for a frequency of 200 MHz at 5 meters. We achieved average throughputs between 400 and 550 kbit/s for all the frequencies tested, until a distance of 1.6 meters.

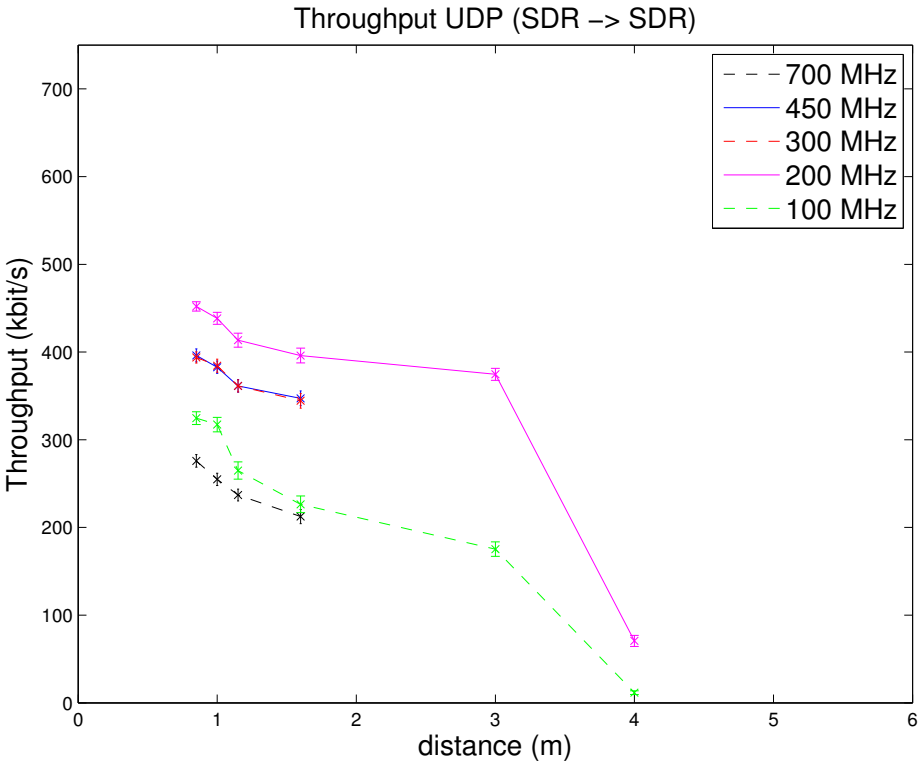


Figure 5.34: UDP Throughput - Channel Bandwidth: 10 MHz

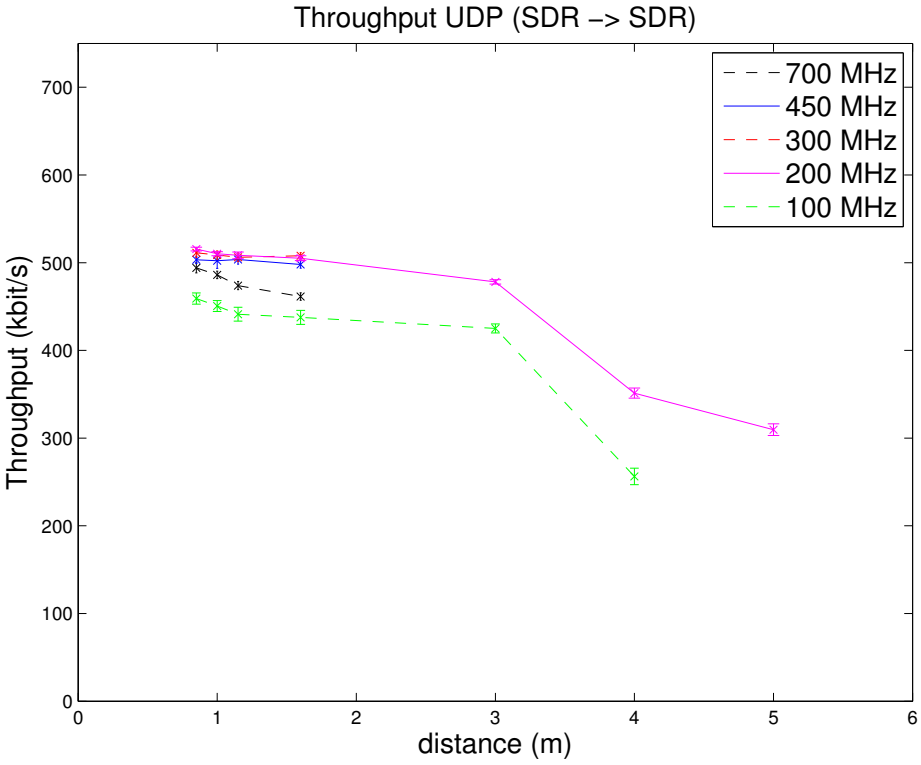


Figure 5.35: UDP Throughput - Channel Bandwidth: 5 MHz

5.4.1.4 Maximum theoretical distance values for low frequencies in freshwater communications

According to the RF propagation models for the underwater scenario that were addressed in Chapter 2, Section 2.3, we calculated the maximum range of distances for low frequencies that could be achieved in underwater communications. The SDR board has a sensitivity of -93 dBm and the SDR transmitted power was 18 dBm. The gains of the receiver and the transmitter were 1 dBi each. The attenuation values for the frequencies tested in our underwater testbed are shown in Table 5.5. These attenuation values were calculated for a water conductivity of 0.0487 S/m. Using Equation 2.8, we calculated the maximum range that could be achieved in each tested frequency. Those theoretical values are also presented in Table 5.5.

Table 5.5: Attenuation and maximum range values for 450, 300, 200, 100 MHz at INESC TEC Robotics freshwater tank (σ = 0.0487 *S/m*)

Frequency (MHz)	450	300	200	100	70
Attenuation (db/m)	17.90	12.92	10.70	9.29	9.05
Maximum range (m)	4.14	5.79	7.14	8.69	9.21

5.4.2 Seawater Environment

In this environment, we used the dipole antenna for a frequency of 70 MHz. The SDR receiver gain, in Table 5.6, needed to be adjusted since there is a different optimum value for each distance.

Figure 5.36 show the obtained results. As we can see, values until a distance of 1.8 meters were achieved with average throughputs around 300 kbit/s. Besides, jitter was around 3 ms until 1.2 meters, reaching an average value of 19.62 ms for 1.8 meters. The seawater testbed is shown in Figure 5.37.

Table 5.6: SDR Receiver and Transmitter Gains for seawater communications

Frequency (MHz)	70										
Modulation	BPSK 1/2										
Distance (m)	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.8
SDR Receiver Gain (dB)	30			35	40			45		50	30
SDR Transmitter Gain (dB)	90										

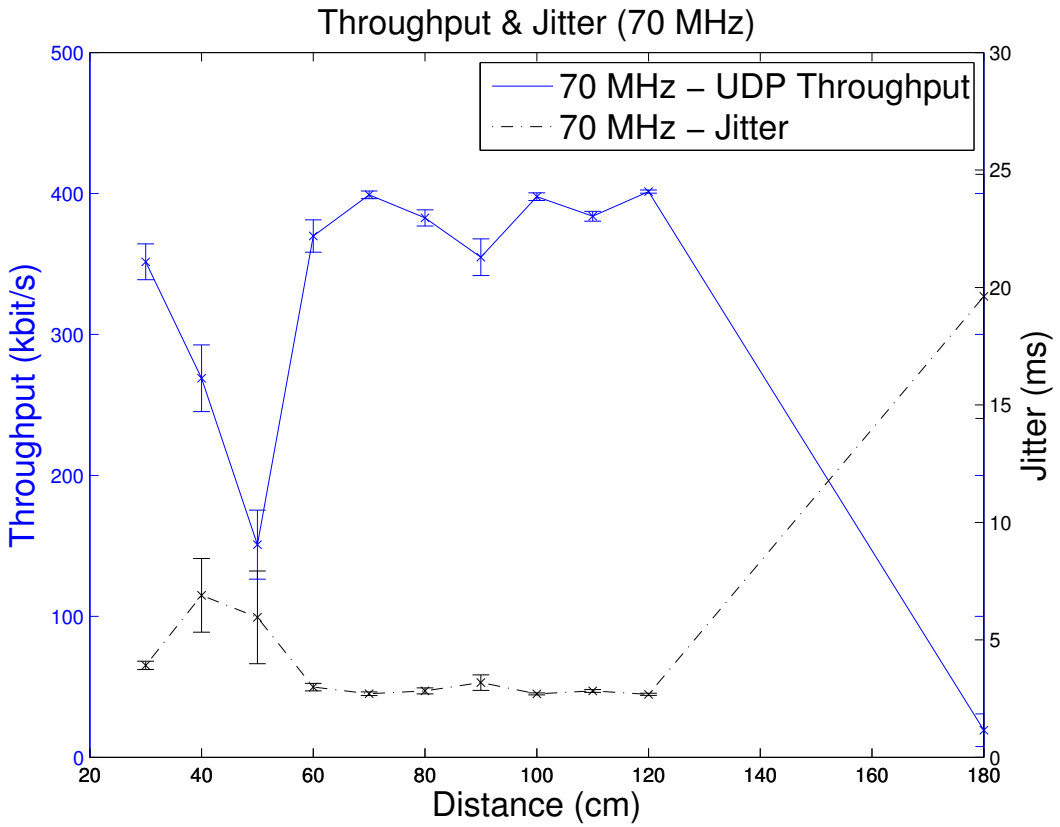


Figure 5.36: UDP Throughput & Jitter - Channel Bandwidth: 5 MHz

Figure 5.37: Views of the seawater testbed.

5.5 Discussion

In this section, we discuss the performance results achieved in the different proposed measurement scenarios. First of all, we were able to optimize and adapt the SDR implementation to receive smaller frames and operate at low frequencies, such as 100 MHz. This optimization and adaptation was performed in a laboratory environment.

Regarding the exterior environment, we started to replicate some tests that the authors conducted and published about the SDR implementation. Our results validated the SDR implementation, though we can conclude that they obtained better results than we did. We needed a higher transmission power to achieve the same value of PDR as them. Besides, in this environment, we performed tests that allowed us to experience interoperability between the SDR board and some commercial wireless cards and also between two SDR boards. On the other hand, we were able to obtain the values of RTT, throughput and jitter between the SDR board and the Wi-Fi card through unidirectional communications. Jitter and RTT values were higher than the ones that are experienced with traditional radios, although they are quite acceptable and meet the requirements of any existing or future application, including the demands of real-time communications.

Experimental results showed a maximum throughput value of 1.82 Mbit/s for communications between a SDR and a Wi-Fi card, when the Wi-Fi card was the receiver. When the SDR was the receiver in these communications, low throughputs were experienced, around 84 kbit/s. These values could have been increased if the SDR implementation supported higher packet sizes, such as 1500 bytes. However, the results were quite better in communications between two SDR boards.

In unidirectional communications between two SDRs, we achieved average data rates of 500 kbit/s and the RTT was higher in this type of communications - but still quite acceptable - and the average value was lower than 40 ms.

Regarding the results obtained in the underwater environments, they validated the use of this SDR implementation at low frequencies, which is adequate for new wireless scenarios, such as underground and underwater communications. Besides, we increased the communication range of RF underwater communications in freshwater to 5 meters, which is a considerable accomplishment, since we achieved more than twice than the previous work at a frequency of 700 MHz. Experimental results showed a maximum throughput of 309.6 kbit/s for a frequency of 200 MHz, at a distance of 5 meters. According to the RF propagation models for the underwater scenario that were addressed in Chapter 2, Section 2.3, it was expected a maximum range of 8.22 meters for a frequency of 200 MHz and a range of 9.87 meters for 100 MHz in a freshwater scenario. We were only able to achieve a maximum range of 4 meters for 100 MHz, since the S21 values of our adapted antennas have low return points near the 100 MHz band, so poorer results were expected to experience in this frequency band. Besides, we achieved better results for a frequency of 200 MHz, which was expected, since our adapted antenna had a maximum return loss for a frequency of 210 MHz, which was mentioned before in Section 4.1.2. Besides, we experienced differences when we compare our obtained results with theoretical values that may be due to the fact that there is an air-water interface, the loop antenna is not in contact with the water, which introduced losses.

On the other hand, decoding the signals from the SDR requires a slightly higher SNR than an ordinary equipment. It should also be noted that we achieved a maximum range of 1.8 meters in a seawater environment and quite acceptable results for jitter and throughput, although the values obtained with a channel bandwidth of 5 MHz were considerable better than the ones obtained with 10 and 20 MHz.

Finally, we can conclude that the experimental results obtained are quite interesting and proved that a SDR system can be used in a real environment, particularly in a real underwater application.

Chapter 6

Conclusions and Future Work

In the last few years, SDRs have become an extremely useful tool for research and development in multiple areas, especially in the wireless domain. Traditional radios are limited, because everything is implemented in hardware. Therefore, SDRs can be a solution to improve the radio flexibility, since they implement most of the signal operations in software, making these radios easily configured.

This dissertation aims at evaluating the performance of an IEEE 802.11 implementation for Software Defined Radio and to adapt and optimize it for new wireless scenarios, namely underwater and underground communications. Therefore, our goal is to enable the usage of IEEE 802.11 networks at low frequencies, which are adequate for these environments. The use of RF waves underwater has not been much explored, especially due to the high attenuation at the 2.4 and 5 GHz band, though we can operate at a very low frequencies with the adaptation of the SDR implementation in order to overcome the underwater attenuation and increase the communication range.

First of all, we validated the IEEE 802.11a/g/p transceiver implementation on a SDR platform with over-the-air communications between the USRP B210 and commercial off-the-shelf equipment and also between two USRP B210. The over-the-air measurements revealed that the SDR could be a solution to the lack of radio flexibility that traditional radios suffer. We achieved a maximum throughput of 1.82 Mbit/s and we experienced an average RTT around 30 ms. In our testbeds, we were able to adapt the SDR implementation to operate at lower frequencies, such as 100 MHz, which is crucial for testing in underwater scenarios, since it is possible to achieve higher distances with lower frequencies. It should also be noted that, in our experiments, we encountered one considerable limitation on the SDR implementation, which is the need to manually adjust the SDR receiver gain, since there is a different optimum value for each frequency. B. Bloessl et al. are already developing an algorithm to introduce an Automatic Gain Control mechanism in a SDR system. Their main objective is to implement this mechanism in a FPGA. In [68], they introduce the algorithm and show some of the results that they already obtained.

Regarding the underwater testbed, we were able to conduct experiences at multiple operating frequencies. Underwater experimental results showed that we were able to increase the maximum

range in RF underwater communications to 5 meters in freshwater, with an average throughput of 309.6 kbit/s for a frequency of 200 MHz and a channel bandwidth of 5 MHz, which represents a 200% increase over the previous work, where the authors achieved a maximum range of 2.15 meters at 768 MHz [24]. Besides, we achieved a maximum range of 1.8 meters in seawater. These results showed that the use of IEEE 802.11 networks in underwater at frequencies lower than 1 GHz may allow wireless communication between an autonomous underwater vehicle (AUV) and an underwater Access Point, or a docking station, proving the feasibility of IEEE 802.11 networks for short range broadband underwater communications.

On the other hand, it is also worth noting that a poster has been presented at the First Doctoral Congress in Engineering, held at FEUP from 11th to 12th of June, 2015, where the main objectives were presented and the methodology was discussed.

Finally, we conclude that the underwater environment presents a lot of challenges, as its particular properties affect the behavior of RF waves in ways that are still not precisely known, as the research, design and optimization are not yet fully developed. Therefore, SDR solutions can have a positive impact on these developments.

6.1 Future Work

The development of this MSc dissertation brought new important results. Nevertheless, we know that there are many research problems and that improvements can be made based on IEEE 802.11 SDR implementations, especially regarding underwater scenarios:

- **Implementation of the CSMA/CA mechanism on a FPGA and multiple access evaluation on the SDR implementation -** One of the most important limitations of the IEEE 802.11 SDR implementation is the latency that occurs between the PC and the SDR board. Therefore, it is impossible to implement the carrier sensing logic in software, since there is a large period of time between the moment the medium is sensed and when it is finally accessed. The only way to implement the CSMA/CA mechanism in a SDR system is on a FPGA, because the frames can be stored in a memory flash that most of the SDR platforms have. Thus, we think that the CSMA/CA implementation will allow the possibility of evaluating the multiple access mechanism on the IEEE 802.11 SDR implementation. This is a supplement to CSMA/CA that aims to reduce frame collisions that occur, because of the hidden node problem. Thus, performance tests using this mechanism should be carried out in the future and compared to results obtained in this dissertation.

- **Evaluate Wi-Fi performance by using antennas in contact with water -** All the underwater tests conducted in the dissertation were achieved with the adapted antenna inside the node. When the packets were transmitted, the electromagnetic waves had to go through different types of mediums: air, case material and water. These medium changes may affect the quality of the signal that arrives at the receiver. Therefore, underwater antennas for the 100

- 400 MHz frequencies should be designed, in order to optimize the antenna performance and increase the maximum range and throughput.

- **New methods to test underwater antennas -** New methodologies and devices should be used in order to get a better characterization on submerged antennas. The use of underwater anechoic chambers could provide a greater depth of knowledge in this area.

References

[1] A. Marwanto, M. Sarijari, N. Fisal, S. K. S. Yusof, and R. A. Rashid. Experimental study of OFDM implementation utilizing GNU Radio and USRP-SDR. In *proceedings of the IEEE 9th Malasya International Conference (MICC)*, pages 132–135. IEEE, 2009.

[2] bladeRF Software Defined Radio. Nuand. Accessed: 2014-11-20. URL: http://www.nuand.com/.

[3] HackRF Software Defined Radio. One Great Scott Gadgets. Accessed: 2014-11-20. URL: http://greatscottgadgets.com/hackrf/.

[4] C. Chen, F. Tseng, K. Chang, H. Chao, and J. Chen. Reconfigurable software defined radio and its applications. *Tamkang Journal of Science and Engineering*, 13(1):29–38, 2010.

[5] USRP Software Defined Radio. National Instruments. Accessed: 2015-01-29. URL: http://www.ni.com/sdr/usrp/pt/.

[6] Ettus Research. Accessed: 2014-11-22. URL: http://www.ettus.com/.

[7] LAN/MAN standards Committee et al. Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE-SA Standards Board*, 2003.

[8] T. Vilches and D. Dujovne. GNUradio and 802.11: Performance Evaluation and Limitations. *IEEE Network*, 28(5):27–31, September 2014. doi:10.1109/MNET.2014.6915436.

[9] P. Fuxjäger, A. Costantini, D. Valerio, P. Castiglione, G. Zacheo, T. Zemen, and F. Ricciato. IEEE 802.11 p transmission using GNURadio. In *Proceedings of the 6th Karlsruhe Workshop on Software Radios (WSR)*, pages 1–4. Citeseer, 2010.

[10] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNURadio. In *Proceedings of the 5th IEEE Vehicular Networking Conference (VNC 2013)*, pages 143–149, Boston, MA, December 2013. IEEE. doi:10.1109/VNC.2013.6737601.

[11] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. Decoding IEEE 802.11 a/g/p OFDM in software using GNU radio. In *Proceedings of the 19th ACM annual international conference on Mobile computing & networking*, pages 159–162, 2013.

[12] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. An IEEE 802.11a/g/p OFDM Receiver for GNU Radio. In *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*, pages 9–16, Hong Kong, China, August 2013. ACM. doi:10.1145/2491246.2491248.

[13] M. Ergen. IEEE 802.11 Tutorial. *University of California Berkeley*, 2002.

[14] PC Engines alix3d3 System Board. Accessed: 2015-01-25. URL: http://pcengines.ch/alix3d3.htm.

[15] P. Freitas. Evaluation of Wi-Fi Underwater Networks in Freshwater. Master's thesis, Faculty of Engineering of Porto (FEUP), 2014.

[16] H. A. Haldren III. Studies in Software-Defined Radio System Implementation. 2014.

[17] A. Reis, A. Selva, K. Lenzi, S. Barbin, and L. Meloni. Software defined radio on digital communications: A new teaching tool. In *Proceedings of Wireless and Microwave Technology Conference (WAMICON), 2012 IEEE 13th Annual*, pages 1–8. IEEE, 2012.

[18] A. Palomo, R. Villing, and R. Farrell. Software defined radio architectures evaluation. *SDR Technical Forum, Washington DC*, October 2008.

[19] GNU Radio. Accessed: 2014-11-19. URL: http://gnuradio.org/redmine/projects/gnuradio/wiki.

[20] T. Ulversoy. Software defined radio: Challenges and opportunities. *IEEE Commun. Surveys & Tutorials*, 12(4):531–550, 2010.

[21] M. L. Dickens. *Surfer: Any-Core Software Defined Radio*. PhD thesis, University of Notre Dame, 2012.

[22] IEEE 802.11 Working Group et al. IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std*, 802:11p, 2010.

[23] A. C. Tribble. The software defined radio: Fact and fiction. In *Radio and Wireless Symposium, 2008 IEEE*, pages 5–8, Jan 2008. doi:10.1109/RWS.2008.4463414.

[24] F. Teixeira, P. Freitas, L. Pessoa, R. Campos, and M. Ricardo. Evaluation of IEEE 802.11 Underwater Networks Operating at 700 MHz, 2.4 GHz and 5 GHz. In *Proceedings of the ACM International Conference on Underwater Networks & Systems*, page 11, 2014.

[25] J. Mitola III. Software radios: Survey, critical evaluation and future directions. *IEEE Aerospace and Electronic Systems Magazine*, 8(4):25–36, 1993.

[26] Wireless Innovation Forum. Accessed: 2014-11-20. URL: http://www.wirelessinnovation.org/.

[27] B. Gu, J. Heo, S. Oh, N. Park, G. Jeon, and Y. Cho. An SDR-based wireless communication gateway for vehicle networks. In *Proceedings of the Asia-Pacific Services Computing Conference, (APSCC'08)*, pages 1617–1622. IEEE, 2008.

[28] Open source cellular infrastructure - OpenBTS. Accessed: 2014-11-21. URL: http://www.ettus.com/.

[29] E. Natalizio, V. Loscri, G. Aloi, N. Paoli, and N. Barbaro. The practical experience of implementing a GSM BTS through Open Software/Hardware. In *Proceedings of the 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, pages 1–5. IEEE, 2010.

[30] P. F. Rito. Recetor SDR para comunicações DSRC. Master's thesis, Universidade de Aveiro, 2011.

[31] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer. A GNU Radio-based IEEE 802.15.4 Testbed. In *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, pages 37–40, Cottbus, Germany, September 2013.

[32] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. Distributed embedded systems - software, an IEEE 802.15.4 transceiver for gnu radio v3.7. Accessed: 2014-11-25. URL: http://www.ccs-labs.org/software/gr-ieee802-11/.

[33] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. Github repository for an IEEE 802.15.4 zigbee transceiver. Accessed: 2014-11-25. URL: https://github.com/bastibl/gr-ieee802-15-4.

[34] A Pinomaa, H Baumgartner, J Ahola, and A Kosonen. Utilization of software-defined radio in power line communication between motor and frequency converter. In *Proceedings of the IEEE International Symposium on Power Line Communications and Its Applications (ISPLC), 2010*, pages 172–177. IEEE, 2010.

[35] M. E. Nelson. Implementation of a Total Power Radiometer in Software Defined Radios. Master's thesis, Iowa State University, 2014.

[36] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke. The IEEE 802.11 universe. *Communications Magazine, IEEE*, 48(1):62–70, 2010.

[37] V. Jain, S. D. Choubey, R. Singh, and S. Gupta. Performance Improvement of 802.11 MAC by enhancements in DCF.

[38] J. H. Schiller. *Mobile communications*. Pearson Education, 2003.

[39] A. Doufexi, S. Armour, M. Butler, A. Nix, D. Bull, J. McGeehan, and P. Karlsson. A comparison of the HIPERLAN/2 and IEEE 802.11 a wireless LAN standards. *Communications Magazine, IEEE*, 40(5):172–180, 2002.

[40] D. Vassis, G. Kormentzas, A. Rouskas, and I. Maglogiannis. The IEEE 802.11 g standard for high data rate WLANs. *IEEE Network*, 19(3):21–26, 2005.

[41] D. Jiang and L. Delgrossi. IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, (VTC'08)*, pages 2036–2040. IEEE, 2008.

[42] R. Dobbins. *Software Defined Radio Localization Using 802.11-style Communications*. PhD thesis, Worcester Polytechnic Institute.

[43] G. Zacheo, D. Djukic, A. Dorni, F. Babich, and F. Ricciato. A Software-Defined Radio implementation of an 802.11 OFDM Physical Layer transceiver. In *Proceedings of the IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA), 2012*, pages 1–4. IEEE, 2012.

[44] An IEEE 802.11a/g/p Transceiver for GNU Radio v3.7 Distributed Embedded Systems Software. Accessed: 2014-11-25. URL: http://www.ccs-labs.org/software/gr-ieee802-11/.

[45] B. Bloessl, C. Puschmann, A.and Sommer, and F. Dressler. Timings Matter: Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture. In *20th ACM International Conference on Mobile Computing and Networking (MobiCom 2014), 9th ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH 2014)*, pages 57–63, Maui, HI, September 2014. ACM. doi:10.1145/2643230.2643240.

[46] GitHub Repository for An IEEE 802.11a/g/p Transceiver for GNURadio. Accessed: 2014-11-25. URL: https://github.com/bastibl/gr-ieee802-11.

[47] B. Benson, Y. Li, R. Kastner, B. Faunce, K. Domond, D. Kimball, and C. Schurgers. *Design of a low-cost, underwater acoustic modem for short-range sensor networks*. IEEE, September 2010.

[48] R. Moore. Radio communication in the sea. *IEEE Spectrum*, 4(11):42–51, 1967.

[49] C. Uribe and W. Grote. Radio communication model for underwater WSN. In *Proceedings of the 3rd International Conference on IEEE New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2009.

[50] S. Jiang and S. Georgakopoulos. Electromagnetic wave propagation into fresh water. *Journal of Electromagnetic Analysis and Applications*, 3(07):261, 2011.

[51] X. Che, I. Wells, G. Dickers, P. Kear, and X. Gong. Re-evaluation of RF electromagnetic communication in underwater sensor networks. *IEEE Communications Magazine*, 48(12):143–151, 2010.

[52] A. Elrashidi, A. Elleithy, M. Albogame, and K. Elleithy. Underwater wireless sensor network communication using electromagnetic waves at resonance frequency 2.4 GHz. In *Proceedings of the 15th Communications and Networking Simulation Symposium*, page 13, 2012.

[53] C. de Morais Cordeiro and D. P. Agrawal. *Ad hoc and sensor networks: theory and applications*. World scientific, 2011.

[54] C. Liu. On the design of OFDM signal detection algorithms for hardware implementation. In *Proceedings of the IEEE Global Telecommunications Conference, (GLOBECOM'03)*, volume 2, pages 596–599. IEEE, 2003.

[55] E. Sourour, H. El-Ghoroury, and D. McNeill. Frequency Offset Estimation and Correction in the IEEE 802.11 a WLAN. In *Proceedings of the IEEE 60th Vehicular Technology Conference, (VTC'04-Fall)*, volume 7, pages 4923–4927. IEEE, 2004.

[56] RouterBOARD R52Hn 802.11a/b/g/n dual band miniPCI card. MikroTik. Accessed: 2015-01-25. URL: http://i.mt.lv/routerboard/files/R52Hn.pdf.

[57] XTREMERange7 700MHz WiFi radio module. UBIQUITI. Accessed: 2015-01-25. URL: http://dl.ubnt.com/xr7_datasheet.pdf.

[58] Ubiquiti XtremeRange7 MiniPCI WiFi Card. Accessed: 2015-01-25. URL: http://www.data-alliance.net/ubiquiti-xtremerange7-minipci-wifi-card/.

[59] L. S. Santos. Wi-Fi Maritime Communications using TV White Spaces. Master's thesis, Faculty of Engineering of Porto (FEUP), 2013.

[60] OpenWrt Linux distribution. Accessed: 2015-01-29. URL: https://openwrt.org/.

[61] Iperf - TCP/UDP bandwidth measurement tool. Accessed: 2015-02-2. URL: https://iperf.fr/.

[62] Tcpdump - powerful command-line packet analyzer. Accessed: 2015-02-2. URL: http://www.tcpdump.org/.

[63] Horst - lightweight IEEE802.11 wireless LAN analyzer. Accessed: 2015-02-2. URL: http://br1.einfach.org/tech/horst/.

[64] INESC TEC Robotics Facilities at ISEP. Accessed: 2015-01-29. URL: http://www.lsa.isep.ipp.pt/.

[65] GitHub Repository for a GNURadio FM RDS/TMC Transceiver implementation. Accessed: 2015-03-8. URL: https://github.com/bastibl/gr-rds.

[66] GitHub Repository for a GNURadio Mode-S/ADS-B radio implementation. Accessed: 2015-03-8. URL: https://github.com/bistromath/gr-air-modes.

[67] Rohde & Schwarz, SMJ100A Vector Signal Generator. Accessed: 2015-02-20. URL: http://www.rohde-schwarz.com/en/product/smj100a-productstartpage_63493-7560.html.

[68] B. Bloessl, C. Sommer, and F. Dressler. Power Matters: Automatic Gain Control for a Software Defined Radio IEEE 802.11a/g/p Receiver. In *34th IEEE Conference on Computer Communications (INFOCOM 2015), Demo Session*, pages 25–26, Hong Kong, April 2015. IEEE.