UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING

Development of Low Cost Spectrum Qualifier for new IoT Technologies.

Leonardo Ribeiro Gonçalves

Itajubá, August 10, 2021

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING

Leonardo Ribeiro Gonçalves

Development of Low Cost Spectrum Qualifier for new IoT Technologies.

Thesis presented to the Graduate Program in Electrical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

Concentration Area: Microelectronics

Supervisor: Prof. Dr. Tales Cleber Pimenta
Co-supervisor: Prof. Dr. Rômulo Mota Volpato

August 10, 2021
Itajubá

Leonardo Ribeiro Gonçalves

Development of Low Cost Spectrum Qualifier for new IoT Technologies

Thesis presented to the Graduate Program in Electrical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

Approved Work. Itajubá, 16 of July de 2021:

Itajubá
August 10, 2021

Leonardo Ribeiro Gonçalves
Development of Low Cost Spectrum Qualifier for new IoT Technologies/ Leonardo Ribeiro Gonçalves. – Itajubá, August 10, 2021-
65 p. : il. (algumas color.) ; 30 cm.

Supervisor: Prof. Dr. Tales Cleber Pimenta

Thesis (Master)
Universidade Federal de Itajubá - UNIFEI
Graduate Program in Electrical Engineering , August 10, 2021.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 07:181:009.3

"We don't need myths, we need examples to be followed. Examples of courage, determination, hope. We need to believe it's possible to win. And it is our duty to pursue it."

(Ayrton Senna)

Acknowledgements

I would like to express my deepest gratitude to my advisor, Tales Pimenta, for his support, mentoring, and encouragement throughout my study journey. I am very grateful to Rômulo Volpato for his endless support during this research. He was enormously helpful in providing feedback during my research and through many revisions of this thesis.

I cannot express enough gratitude for Tiago Magalhães for his support in collaborations and valuable insights. I thank to Marco Casaroli for his insightful conversations about Software Defined Radio and providing me the LimeSDR radio for my experiments. I thank Carlos Nazareth, director of my undergraduate university, who met his goal of making me a good telecommunication engineer. I would like to thank Daniel Mazzer, who gave me feedback and attention on my Thesis.

I would like to extend special thanks to people from LABCAL at Inatel, for allowing me to make my measurements at their anechoic chamber.

Lastly, I must thank my wife Tatiana, who literally worked alongside me at home for months during a global pandemic as I have written this thesis. I also would like to offer special thanks to my lovely family and friends, at UNIFEI, Inatel and elsewhere, for their endless support.

Abstract

With the evolution of radio technologies during the last decades, the companies are replacing equipment with individual components with particular functions by systems where most signals are treated digitally through digital signal processing techniques. With this, engineers can develop many applications at the software level, enabling new analyses to exists.

With the growth of the internet of things, companies are deploying new solutions around the world. Most of these solutions are wireless applications that use the unlicensed frequency spectrum band, which increases the density of wireless devices in a location. In environments with numerous wireless equipment transmitting simultaneously, it is essential to detect how busy the frequency spectrum is before deploying a new solution. A spectrum analyzer can analyze this, but the cost of acquiring suitable equipment can be very high. Besides that, these devices only allow users to analyze the frequency spectrum, not the technology itself, packet loss, data rate, and coverage, without purchasing expensive software licenses.

This thesis presents the development of a software-defined radio application capable of evaluating the performance of radio technologies, such as Bluetooth or Wi-SUN, and shows their behavior in a noisy environment.

Key-words: Sofware-Define Radio, Bluetooth, Wi-SUN, Spectrum Sensing, GNU Radio, LimeSDR.

List of Figures

List of Tables

List of abbreviations and acronyms

2nd Generation Mobile Networks	2G
3rd Generation Mobile Networks	3G
4th Generation Mobile Networks	4G
5th Generation of Mobile Networks	5G
Analog-to-Digital Converter	ADC
Bit Error Rate	BER
Brazilian Telecommunication Regulatory Agency	Anatel
Digital Signal Processing	DSP
Digital-to-Analog Converter	DAC
Direct Sequence Spread Spectrum	DSSS
Field Area Network	FAN
Field Programmable Gate Array	FPGA
Forward Error Correction	FEC
Frequency-Shift Keying	FSK
Gaussian Frequency-Shift Keying	GFSK
Industrial Scientific and Medical	ISM
Institute of Electrical and Electronics Engineers	IEEE
Internet Control Message Protocol	ICMP
Internet of Things	IoT
Long Range	LoRa
On-Off Keying	OOK
Packet Error Rate	PER
Phase Shift Keying	PSK
Printed Circuit Board	PCB
Quadrature and Amplitude Modulation	QAM
Radio Frequency	RF
Radio-Frequency IDentification	RFID
Software Defined Radio	SDR
Universal Serial Bus	USB
Wireless Fidelity	Wi-Fi
Wireless Local Area Network	WLAN
Wireless Regional Area Network	WRAN

Wireless Smart Ubiquitous Networks	Wi-SUN

Summary

1 Introduction

The telecommunication market evolves every year, always aiming for more excellent connectivity and data throughput. Recently, research and development companies are developing devices with the ability to exchange data. Electronic devices and home appliances that previously had no Internet connection, such as televisions, refrigerators, air conditioners, and even a simple power outlet, are now capable of generating or receiving some information [1]. With the advent of *5th Generation of Mobile Networks* (5G) and other new wireless technologies, the *Internet of Things* (IoT) has expanded in mobile communications jointly with mobile operators, given the relevance and the possibility of new business. In this context of IoT, researchers discuss several types of research and proposals in areas such as ultra-dense connections, devices with a very low transmission rate, and the durability of the batteries[2].

According to Juniper Research [3], challenging topics in this IoT context are the interoperability, fragmentation, and standardization, as several technologies can be employed, such as RFID, Wi-Fi, LoRa, Zigbee, Bluetooth, *4th Generation Mobile Networks* (4G) and Wi-SUN and each one uses it is own communication protocol. Also according to Juniper [3], the IoT market will grow around 130% over the next four years, going from 35 billion connections to 83 billion, thus becoming the key driver of this change in industrial application. With this number of connections, the revenue for 2020 reached $66 billion, which corresponds to an increase of 20% over the previous year.

With this massive number of connected devices, the occupancy of the electromagnetic spectrum increases. Technologies such as Wi-SUN, Bluetooth, LoRa, Sigfox, and Zigbee use unlicensed frequencies. A device that uses those technologies does not need to pay any fee to any governmental agency(*Brazilian Telecommunication Regulatory Agency* (Anatel), in Brazil) to communicate in these frequencies. The most commonly unlicensed frequencies used worldwide, at sub 6 GHz, are 902 to 928 MHz, 2400 MHz to 2483.5 MHz, and 5725 MHz to 5850 MHz [4]. Hence, these wireless technologies may saturate some spectrum bands in certain central places, and an IoT may not work as expected due to the amount of electromagnetic interference.

In today's market, companies can choose a lot of wireless IoT networks in smart cities solutions. This work tests two prominent of them, Wi-SUN and Bluetooth.

Wireless Smart Ubiquitous Networks (Wi-SUN) is a technology-based on IEEE 802.15.4g standard [5]. Wi-SUN Alliance [6] supports it, and many companies are developing their own certified Wi-SUN stack conformant to the standard proposed by the alliance to ensure interoperability with other certified equipment[7, 8, 9]. This technology supports start and mesh topology, but Wi-SUN networks are usually mesh networks, where neighboring IoT devices can serve as an access point to the network coordinator, called Border Router. Wi-SUN provides different data rates modes that enable the users to choose between bandwidth and latency. This work tests the 100kbps profile.

Bluetooth technology has emerged as the global wireless standard powering the IoT evolution. Bluetooth SIG Core Working Group regularly updates new specifications to satisfy evolving technology and market needs[10]. Bluetooth supports start, mesh, and point-to-point topology. To meet projects requiring throughput and long-range, it provides different data rates as 2Mbps, 1Mbps, 500kbps, and 125kbps. This work tests the 1Mbps profile.

A prior survey of the frequency spectrum is essential to monitor the interference in a determined region to improve the IoT network design. Such a situation typically requires a spectrum analyzer, but this approach's main barrier is the price, which can fluctuate from a few hundred up to some thousands of dollars, depending on its configuration. Moreover, many spectrum analyzers do not record the spectrum for further processing but only collect real-time data. Besides, if the user wants to check various bandwidth and central frequency combinations, he needs to make many local configurations on the spectrum analyzer. These changes in the analyzer's settings may demand much time.

This work aims to develop a low-cost solution using *Software Defined Radio* (SDR) to monitor the spectrum on various unlicensed frequencies and collect data for further analysis. As a result, it can provide information regarding the spectrum in the region where the equipment is and measure the channel occupation in a specific technology. With this data, the user can anticipate problems and choose the best technology to meet the region's IoT project expectations. The LimeSDR [11] hardware was chosen to realize the measures. It is low-cost software-defined radio with a *Universal Serial Bus* (USB) input that can be easily connected to a computer and programmed using a graphical interface, like GNU Radio software.[12]

The organization of this work is as follows: Chapter 2 introduces the main aspects of an SDR, the historical marks, the LimeSDR board technical specification, and the initial programming steps. Chapter 3 describes the process of setting up the environment required to measure the scenario of each technology. Chapter 4 presents two test proposals for channel interference and demonstrates the results obtained using LimeSDR. Finally, Chapter 5 presents the concluding results and the prospects of future works.

2 Software-Defined Radio

This chapter introduces the history and evolution of *Software Defined Radio* (SDR). It describes the definition of software-defined radio, its architecture, and how it works. This chapter also provides the LimeSDR hardware specifications.

2.1 Radio technologies evolution

For many decades, the radio consisted of many internal components, such as oscillators, filters, mixers, circulators. Since the development of the superheterodyne radio in 1917 [13, Appendix A.3], the radio manufacturers primarily adopted its design in almost all modern radio receivers due to some benefits: very low spurious emissions and channel bandwidth, and selectivity due to intermediate frequency use, and a trade-off between noise figure and linearity [13, Sec. 1.4]. Figure 1 shows an example of superheterodyne transceiver radio.

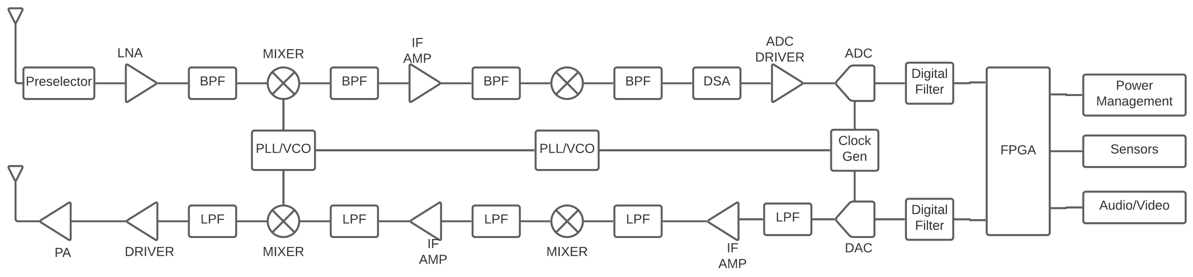


Figure 1 – Superheterodyne transmitter and receiver radio [13, Sec. 1.4].

As a result of this extensive superheterodyne radio employment, there have been significant improvements across the entire signal chain. Higher quality and lower losses components introduce less distortion on signals and offer better linear behavior, but they are more expensive than standard components.Although there are great difficulties and high costs for constructing superheterodyne receivers, their implementations represented the possibility of constructing frequency agnostic software-defined radios. The FPGA software structure allows implementing the most different and comprehensive waveforms, using the inverse and direct Fast Fourier Transforms (IFFT and FFT). However, only the superheterodyne circuit allows the cognitive radios to operate in large frequency bands.

A critical point in the wireless transmission is the frequency since higher frequencies require better radio design. For example, the track width in a *Printed Circuit Board* (PCB) can act as a radiant element since the wavelength is proportional to the size of the elements. Also, the internal components can suffer from non-linearities and others hardware impairments [14].

Over the years, Zero-IF (Zero intermediate frequency) architecture rises as an alternative to superheterodyne radio [13, Sec. 1.4]. This type of radio uses fewer components than superheterodyne, and it is based on a direct down-conversion from passband to baseband, with only one local oscillator. Hence, this oscillator considerably influences the project and suffers many hardware impairments, mainly when used in higher frequencies. The oscillator must ensure a perfect 90° between components in-phase (I) and quadrature (Q), or it will result in performance degradation [14]. Figure 2 shows an example of a Zero-IF transceiver.

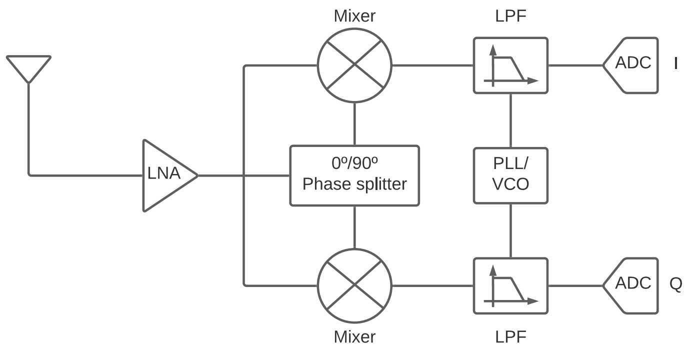


Figure 2 – Zero-IF receiver radio [13, Sec. 1.4].

Unfortunately, if the project requires changes, such as bandwidth or central frequency, it could not be effortless since it needs to be physically modified. With technological advancement, all types of radio architecture can be built in a monolithic transceiver. Nowadays, filters, oscillators, amplifiers, and others can be placed inside a single silicon chip, thus minimizing many hardware imperfections and variations compared to the dedicated radio-frequency components. This technology can reduce the time to perform hardware changes since the chip vendor allows reconfiguration in some chips using a proprietary protocol [15].

In a digital environment using modulations like *Phase Shift Keying* (PSK) or *Quadrature and Amplitude Modulation* (QAM), a radio project need to implement other functionalities to deal with wireless transmission imperfections like errors on oscillators such as phase noise, errors on modulation/demodulation, synchronization extraction failure, and others [16]. Software processing is applied to the raw data at the modulation/demodulation step to overcome some of these obstacles. In this case, the responsible engineer should address all those considerations individually.

Suppose the transmission power suffers due to radio mobility. In that case, some changes on the modulation scheme or channel coding have to be applied to aim throughput or reliability to reduce errors due to propagation and fading. High order modulations like 256-QAM require more power than 64-QAM to maintain the same bit error rate if the objective is throughput. If the signal-to-noise ratio worsens, radio should reduce modulation order. In some cases, this reduction is insufficient, and channel coding should be changed to achieve higher reliability. Those real-time modifications need to be software implemented [17].

2.2 SDR Definition

The first time the term *Software Defined Radio* (SDR) appeared was in the 1970s by Joseph Mitola [18]. Nevertheless, the critical milestone of Mitola's idea occurred in 1990 with the first public-funded initiative, the SpeakEasy I/II, by the U.S military [19]. More than ten military communication standards adopted this project, ranging from 2 MHz up to 2 GHz. Nonetheless, due to technical limitations at that time, the physical size of the SpeakEasy prototype was so large that it required a truck [20, Sec. 1.1].

Software-defined radio is a class of reconfigurable/reprogrammable transceivers. Most of the physical layer characteristics can be significantly modified or implemented by software, i.e., it is general-purpose hardware whose configuration is under software control [20, Sec. 1.2.1], [21, Sec. 2.1]. Thus, all radio functionalities such as modulation, source coding, equalizer, filters, and other functional blocks are software-based. The code can be stored in memory and then called upon by an automated process.

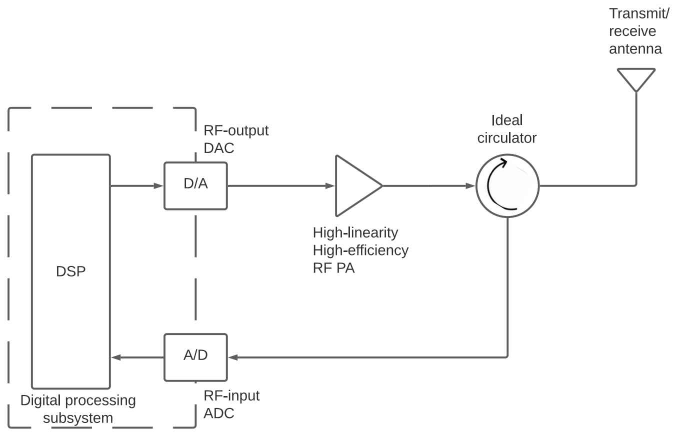


Figure 3 – Ideal SDR architecture.

Figure 3 represents a *Software Defined Radio* (SDR) architecture. A *Digital Signal Processing* (DSP) is responsible for all mathematical computations and can be relatively easily applied to process digital communication modules. The *Analog-to-Digital Converter* (ADC) and *Digital-to-Analog Converter* (DAC) are the key components to transform the digital data to a suitable waveform based on the channel used. Important to highlight that those ADC and DAC converters are assumed to have built-in anti-alias and reconstruction filters, respectively. The DAC power output is not high enough to propagate the signal for a long distance, independent of the channel. A *Radio Frequency* (RF) power amplifier is necessary to ensure a sufficient high-power signal suitable for transmission in such a situation. Most of the radios are capable of transmitting and receiving signals but, to do so, it is essentially a type filter to separate both paths. Figure 3 used a circulator instead of a conventional diplexer because this second one is highly frequency-dependent, introducing frequency restriction on a multi-band radio. Finally, to propagate the signals through a wireless channel, an antenna is needed. Figure 3 assumes an ideal match between circulator, antenna, and power amplifier, but, in practice, that does not occur. Additional elements are needed to reduce the mismatch losses that grow according to frequency. So, one can conclude that most modern SDR devices share a similar structure shown previously, aiming always to convert analog baseband domain into IQ samples.

2.3 Lime SDR

The software-defined radio selected to collect all data in this work is the Lime SDR Mini [11]. It is a highly cost-optimized hardware platform focused on high-performance digital and RF designs. Figure 4 shows a board based on the LMS7002M radio transceiver and Altera MAX 10 *Field Programmable Gate Array* (FPGA).

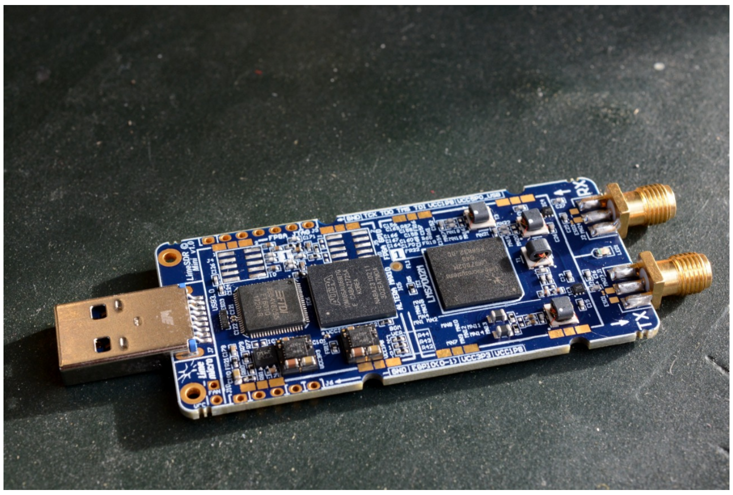


Figure 4 – Lime SDR Mini.

2.3.1 Hardware

According to the vendor, it is possible to implement many RF designs as digital audio broadcasting, cellular technologies such as 2G, 3G and 4G, and additional standards, including WiFi [22]. The main characteristics of this hardware are:

- 10 MHz - 3.5 GHz frequency range,
- FPGA with 169-pin package,

- 45 $\times$ 18 $\times$ 18-bit multipliers,
- 130 general-purpose input-output,
- 2 $\times$ 128 kB for RF transceiver firmware and data,
- 4 MB flash memory for data,
- USB 3.0 Type-A,
- 2 coaxial SMA connectors,
- 12 bit ADC DAC,
- Supports time and frequency division duplexing.

2.3.2 Cost

In order to justify the goal of this work, which is a development of a low-cost device that analyzes wireless channels, Table 1 shows a cost comparison between other SDR and Spectrum Analyzers to the LimeSDR.

Table 2 – Cost comparison table

Device	**Frequency Range**	**RF Bandwidth**	**Price**
LimeSDR-Mini	10 MHz - 3.5 GHz	30.72 MHz	$159.00
HackRF One	1 MHz - 6 GHz	20 MHz	$399.99
Ettus B210	70 MHz - 6 GHz	61.44 MHz	$1,695.00
Nuand BladeRF x40	300 MHz - 3.8 GHz	40 MHz	$480.00
RTL-SDR	500 kHz - 1.766 GHz	2.4 MHz	$29.95
Tektronix MDO34	9 kHz - 3 GHz	3 GHz	$4,850.00
Rigol RSA5032	9 kHz - 3.2 GHz	3.2 GHz	$6,999.00
Rohde and Schwarz FPH-COM1	5 kHz - 4 GHz	4 GHz	$7,140.00

Analyzing Table 2, only RTL-SDR has a lower cost than LimeSDR; however, this SDR cannot analyze the 2.4GHz band. The last three devices in table 2 are spectrum analyzers which are more complex products with the manufacturer's operating system installed and many other measurement features, which explains the much higher cost.

It is possible to verify that the most expensive SDR, the Ettus B210, is much cheaper than the Spectrum Analyzer with the lowest cost, the Tektronix MDO34. With this, *Software Defined Radio* (SDR) are critical enablers for spectrum analysis, providing a low-cost and flexible radio architecture through the use of software that offers suitable measures compared to expensive spectrum analyzers, which rely entirely on hardware[23]. The choice for LimeSDR is its cost-effectiveness and its ability to analyze the 900MHz and 2.4GHz bands.

2.4 Summary

This chapter presented the radio evolution up to the *Software Defined Radio* (SDR), hardware characteristics, and cost of the LimeSDR, the equipment used in the experiments.

3 Environment for RF measurements

An RF spectrum sniffer application that executes channel power measurements was developed using the LimeSDR platform and GNU Radio software [22] [24].

This chapter describes the process of setting up the environment to collect the data for each technology used. The LimeSDR collected the measurements on the 915MHz band with Wi-SUN signals and the 2.4GHz band with Bluetooth signals. Table 3 shows all the necessary equipment to construct the solution presented in this work. All the specific datasheets are available in the references [25, 26, 27, 28].

Table 3 – Devices for setting up the test environment.

Device	**Manufacturer**	**Model**
LimeSDR-Mini	Lime Microsystems Ltd	v1.1
3x Wireless Starter Kit	Silicon Labs	BRD4001A
3x EFR32 Radio Board	Silicon Labs	BRD4164A
3x 915 MHz dipole antenna	Linx	ANT-916-CW-HWR-SMA
3x 2.4 GHz dipole antenna	Linx	ANT-2.4-CW-HWR-SMA
Spectrum Analyzer	Tektronix	MDO34
Personal Computer	-	-

3.1 Measurement Setup

The setup shown in Figure 5 is composed of a LimeSDR Mini connected by USB to a computer responsible for running the sniffer application built for radio channel power scan. Two nodes, TX and RX, are used to test the scenario technology and check *Bit Error Rate* (BER) with and without interference. The interferer node transmits an unwanted signal to the environment with the same characteristics of the tested technology. All the signals are scanned and measured by the LimeSDR. This setup follows Anatel 946, item 10.3.3, which uses the following diagram to check the BER, shown in Figure 6 [29].

In this configuration, shown in Figure 6 there is a random sequence generator connected to the transmission system modulator. The transmitter block receives the signal from the modulator block and adapts its levels and frequency. The receiver block receives the signal and adapts its level and frequency compatible with the demodulator block. The demodulator will detect the generated random sequence and is capable of detecting errors. Thus, if an error occurs in the received sequence, the error detector will count. In this specific test, the acceptable *Bit Error Rate* (BER) is 10E-3. Note that the signal at the receiver input is in a confined environment (via coaxial cable) in this system. Thus the

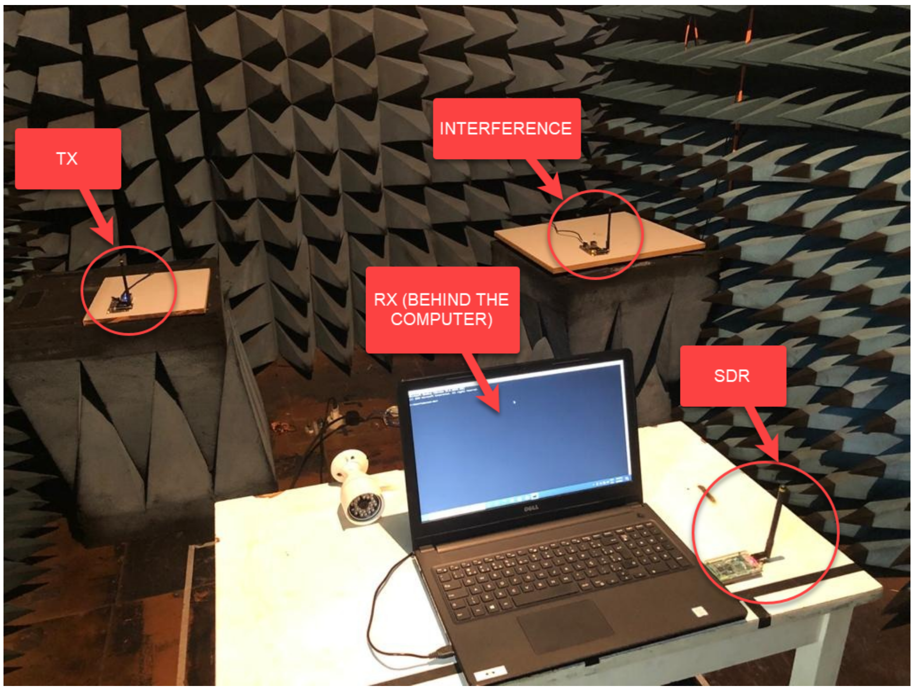


Figure 5 – Laboratory setup environment.

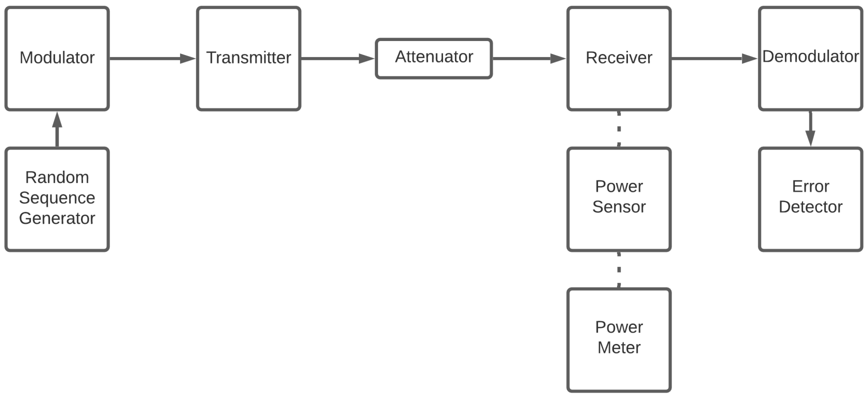


Figure 6 – Setup for *Bit Error Rate* (BER) tests.

variable attenuator can control the reception level at the input. This procedure cannot be used for wireless technologies because the connection between the transmitter and receiver is via free space; thus, unwanted signals may contaminate the received signal and change the receiver's reception level. Therefore, the transmission system is set in a controlled environment, like an anechoic chamber, that reduces the reception of other strange and undesirable signals.

All signals required to evaluate this work are generated by the FLEX-Gecko EFR32 Silicon Labs SoC [26]. The node is composed of a low-power microcontroller with an integrated radio supporting sub-GHz and 2.4GHz frequencies. It supports modulation of the schemes *On-Off Keying* (OOK), shaped *Frequency-Shift Keying* (FSK), shaped OQPSK with *Direct Sequence Spread Spectrum* (DSSS) and *Forward Error Correction*

(FEC). Figure 7 shows the radio board used for tests.

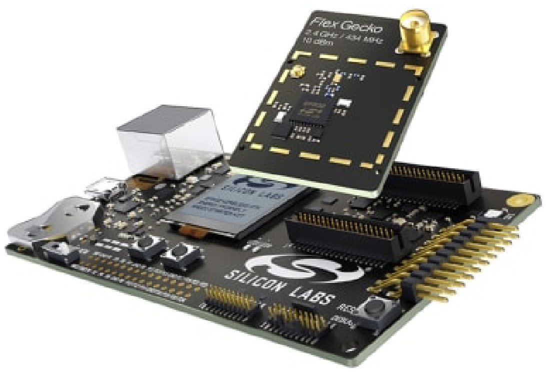


Figure 7 – EFR32FG12 Main Board.

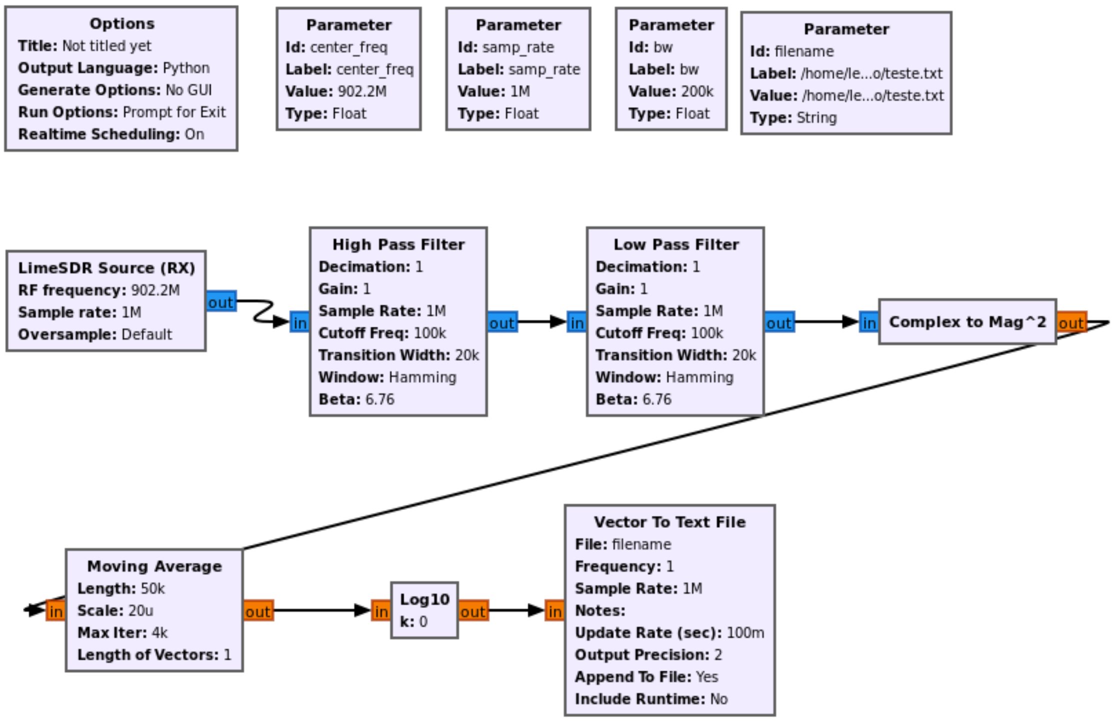
Interfering and interfered signals are transmitted through the air in an anechoic chamber to ensure that no other signal is present in the measurements. A distance of approximately 1.5 meters separates them. LimeSDR scans all channels for 14 seconds because the media should only be occupied up to 0.4 seconds in 14 seconds, as recommended by Anatel to check the channel occupancy [30, Sec. 10.2.5.2].

3.2 Application Development

The application designed for LimeSDR was developed using GNU Radio Companion software. This software has a graphical interface that allows users to quickly create and execute signal processing applications by drag-and-drop blocks, as shown in Figure 8. With this type of digital radio, any new digital signal processing feature can be validated, compiled, embedded, and debugged in the real world. The output file generated by GNU software is available in Annex A.

These blocks process the signal and collect the average channel power in the band selected by the filters. The explanation of how each block works is as follows:

- Parameter(center_freq): Center frequency used for measures.
- Parameter(samp_rate): Signal sample rate frequency.
- Parameter(bw): Channel bandwidth.
- Parameter(filename): File used to save the signal power measures.
- LimeSDR Source: Responsible for making the interface between the computer and the external RF hardware. The signal in a given center frequency of the desired bandwidth and converted to a baseband signal to be processed.


Options
Title: Not titled yet
Output Language: Python
Generate Options: No GUI
Run Options: Prompt for Exit
Realtime Scheduling: On

Parameter
Id: center_freq
Label: center_freq
Value: 902.2M
Type: Float

Parameter
Id: samp_rate
Label: samp_rate
Value: 1M
Type: Float

Parameter
Id: bw
Label: bw
Value: 200k
Type: Float

Parameter
Id: filename
Label: /home/le...o/teste.txt
Value: /home/le...o/teste.txt
Type: String

LimeSDR Source (RX)
RF frequency: 902.2M
Sample rate: 1M
Oversample: Default

High Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Cutoff Freq: 100k
Transition Width: 20k
Window: Hamming
Beta: 6.76

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Cutoff Freq: 100k
Transition Width: 20k
Window: Hamming
Beta: 6.76

Complex to Mag^2

Moving Average
Length: 50k
Scale: 20u
Max Iter: 4k
Length of Vectors: 1

Log10
k: 0

Vector To Text File
File: filename
Frequency: 1
Sample Rate: 1M
Notes:
Update Rate (sec): 100m
Output Precision: 2
Append To File: Yes
Include Runtime: No

Figure 8 – Signal processing blocks.

- High Pass Filter: Eliminates low-frequency noise.
- Low Pass Filter: Eliminates high-frequency noise.
- Complex to Mag^2: Measure the signal power in Watts. It is responsible for converting the input voltage into Watts.
- Moving Average: This block calculates the moving average over the samples.
- Log10: Converts the power to decimal scale (dBm).
- Vector to Text File: Saves the results in a specified file.

When executing the code generated by GNU Radio, written in Python, some configurable parameters, such as the channel, the bandwidth, and the file's name that save the power measurements, table 4 describes these parameters.

After completing the collection process for each channel of the desired technology, the application developed in C language treats the data to get the mean of all measurements, check the channels occupancy, and save it into a file. The python application shown in annexes C and E reads the file with occupancy and mean values and plot the graphs to show the results to the users. The Python language provides an extensive library of tools to treat and display the information. Annexes A, B, C, D, and E present the codes used to treat the data, as explained by the application flow shown in Figure 9.

Table 4 – Software defined radio main configuration parameters.

Parameters	**Description**
Center Frequency	Specifies the channel center frequency
Sample Rate	Specifies the sample rate of the received signal. In this project, this parameter determines the signal bandwidth.
File	The filename that saves the power measurements.

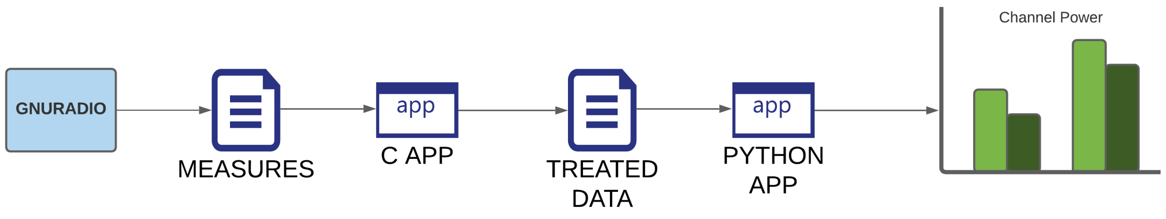


Figure 9 – Application flowgram.

3.2.1 Practical fundamentals

This subsection introduces the terminology and concepts to clarify the experiment fundamentals. Regardless of the wireless technology analyzed, the terms presented here are the same in any telecommunication system.

Transmitter Output Power – Output power signal of the transmitter radio. The transmitter output power will depend on the technology used and the country's regulation. In Brazil, the maximum output power allowed in the *Industrial Scientific and Medical* (ISM) band is 1 Watt[30].

Antenna Gain – Capacity of the antenna to concentrate the transmitted or the received signal in a specified direction. In other words, when transmitting, the gain describes how well the antenna converts the power into radio waves. When receiving, the gain describes how well the antenna converts radio waves into electrical power. Typically is expressed in dBi, which is the gain referred to as a perfect isotropic radiator.

Received Signal Strength – Received signal strength at the receiver radio.

Receiver Sensitivity – The received signal strength required by the receiver radio to detect the RF signal and decode it at an acceptable bit error rate.

3.3 Measurements procedures

3.3.1 Procedures to assure power level measurement

The collected data need to be compared to a reference value to assure the LimeSDR measures the correct power level. Therefore, the reference spectrum analyzer Tektronix MDO34 and the LimeSDR are placed at the anechoic chamber. No signal is present in this test, and both devices measure the noise floor power level. Besides that, to minimize possible differences at the signal reception, the same antenna was used for both types of equipment. The comparison of all measurements generates an error offset was for each frequency band, as shown in Table 5. The offset values and the measures captured by LimeSDR are combined to guarantee more accurate results. The bash script used to collect this data is available in Annex D.

In order to get the data from LimeSDR, execute the following procedures:

- Connect the LimeSDR on PC USB 3.0 port.
- Run the python code generated by the GNU Radio, passing the central frequency and the channel bandwidth arguments.
- Save the power measures in a file.

The following procedures are performed to collect the data from the spectrum analyzer:

- Turn on the spectrum analyzer.
- Insert the antenna used for measurements on the RF input.
- Configure the same central frequency, as used on the SDR.
- Configure the power measurement functionality with the same bandwidth used for measuring channel power on LimeSDR.
- Collect channel power on the specified channel.
- Save the power measurement value.

After completing the previous procedures, take the difference between the values and get the offset between both types of equipment. Since the spectrum analyzer is a trusted device, it is the reference. Thus, using the International Metrology Vocabulary, in its definition of measurement error in item 2.16, is a measured value minus a reference value [31].

$$Error = P_{SDR} - P_{SA} \tag{3.1}$$

However, the necessary measure correction, or offset, is the inverted error. Furthermore, the expression is:

$$Offset = P_{SA} - P_{SDR} \tag{3.2}$$

Where P_{SA} is the power measured by the spectrum analyzer, and P_{SDR} is the power received by SDR. As shown in Table 5, the offset calculation results are negative numbers, which means that the reference equipment can detect lower signal levels due to a higher receiver sensibility.

Table 5 – Error offset between spectrum analyzer and LimeSDR at each frequency.

Frequency	**LimeSDR**	**MDO34**	**Offset**
902MHz	-98.25	-98.47	-0.22dB
904MHz	-97.39	-97.59	-0.20dB
906MHz	-97.34	-97.55	-0.21dB
908MHz	-97.04	-97.24	-0.20dB
910MHz	-97.32	-97.55	-0.23dB
912MHz	-97.28	-97.5	-0.22dB
914MHz	-97.09	-97.29	-0.20dB
916MHz	-98.27	-98.48	-0.21dB
918MHz	-98.45	-98.66	-0.21dB
920MHz	-97.44	-97.64	-0.20dB
922MHz	-97.24	-97.47	-0.23dB
924MHz	-97.44	-97.63	-0.19dB
926MHz	-97.24	-97.45	-0.21dB
928MHz	-98.44	-98.66	-0.22dB
2.400GHz	-78.48	-78.59	-0.11dB
2.410GHz	-77.49	-77.61	-0.12dB
2.420GHz	-77.47	-77.58	-0.11dB
2.430GHz	-77.31	-77.41	-0.10dB
2.440GHz	-78.09	-78.22	-0.13dB
2.450GHz	-77.23	-77.35	-0.12dB
2.460GHz	-77.02	-77.12	-0.10dB
2.470GHz	-77.02	-77.14	-0.12dB
2.480GHz	-77.77	-77.88	-0.11dB

The errors in measurements across both frequency bands are slight, highlighting the LimeSDR measurement accuracy compared to equipment with a much higher cost.

3.3.2 Procedures for checking interference on the same channel

Co-channel interference is the interference between two or more transmitting radios using the same radio frequency channel. Perform the following steps to check the influence of this interference in the communication link:

- Configure LimeSDR to scan on entire bandwidth, running the code on all channels.
- Configure the interferer node.
  - Channel: 0
  - Power: 20dBm
- Configure the receiver node.
  - Channel: 0
  - Sensitivity: -105.1dBm - 0.1% BER (915MHz)
  - Sensitivity: -94.8dBm - 0.1% BER (2.4GHz)
  - Packet length expected: 64 bytes
- Configure the transmitter node.
  - Channel: 0
  - Power: 20dBm
  - Packet length: 64 bytes
- Run LimeSDR script.
- Start interferer node.
- Start receiver node.
- Start transmitter node.
- Check BER on receiver node.
- Run applications to collect signal statistics and plot graphs.
- Analyze channel map interference.

3.3.3 Procedures for checking interference on the adjacent channel

Adjacent-channel interference is the interference caused by a radio operating on an adjacent radio frequency channel. Perform the following steps to check the influence of this interference in the communication link:

- Configure LimeSDR to scan on entire bandwidth, running the code on all channels.
- Configure the interferer node.
  - Channel: 1
  - Power: 20dBm
- Configure the receiver node.
  - Channel: 0
  - Sensitivity: -105.1dBm - 0.1% BER (915MHz)
  - Sensitivity: -94.8dBm - 0.1% BER (2.4GHz)
  - Packet length expected: 64 bytes
- Configure the transmitter node.
  - Channel: 0
  - Power: 20dBm
  - Packet length: 64 bytes
- Run LimeSDR script.
- Start interferer node.
- Start receiver node.
- Start transmitter node.
- Check BER on receiver node.
- Run applications to collect signal statistics and plot graphs.
- Analyze channel map interference.

3.4 Summary

This chapter described the setup and the fundamentals for enabling a *Software Defined Radio* (SDR) application development to capture and analyze the power measures from frequency spectrum to check channel occupancy and channel mean power for Wi-SUN and Bluetooth technologies.

It is more feasible to assemble the setup and make the measurements with the solution proposed by the thesis if compared to the condition imposed by spectrum analyzers. Furthermore, the difference between the measurements is minimal, which highlights the viability of the proposal.

The next chapter shows the proposed test procedures.

4 Tests proposals and Measurement Results

This chapter presents the proposal and results of two experiment procedures for channel interference on systems based on *Gaussian Frequency-Shift Keying* (GFSK) physical layer modulation techniques, operating in 900 MHz and 2400MHz frequency bands. The term interference should be interpreted as unwanted radio frequency signals that degrade, obstructs, or interrupts a radio communication service. All the codes used to transform the collected data in graph format can be found at Annex E and F.

4.1 Proposed Scenarios

The following experiments simulate a hypothetical *Wireless Regional Area Network* (WRAN) scenario, where the system operates in the standard ISM bands, with interference present on the same, and on adjacent channels. The main objective is to demonstrate the advantages of scanning the media before deploying a new generation of technologies for IoT in smart cities applications.

Even though the IEEE 802.15.4g standard has defined the use of GFSK as the physical layer modulation. The procedures demonstrates the advantages of using the frequency-hopping implementation techniques [17, Sec 7.1.6]. The interfered and interfering signals, simulate static conditions in a controlled environment. The multi-path fading is disregarded in this analysis.

In scenario A, the Wi-SUN technology is tested. The interference signal generated in this scenario have the same characteristics of the Wi-SUN signal. In scenario B, the Bluetooth technology is tested. The interference signal generated in this scenario have the same characteristics of the Bluetooth signal.

In last scenario, to verify the solution's behavior in a real environment, data from the frequency spectrum of a street lighting solution was collected. This solution uses Wi-SUN technology, which is a IPv6 wireless sensor network with mesh topology. This network consists of router nodes, which have the capability of expand the network coverage, and a gateway called the border router, which provides access to the internet as shown in Figure 10.

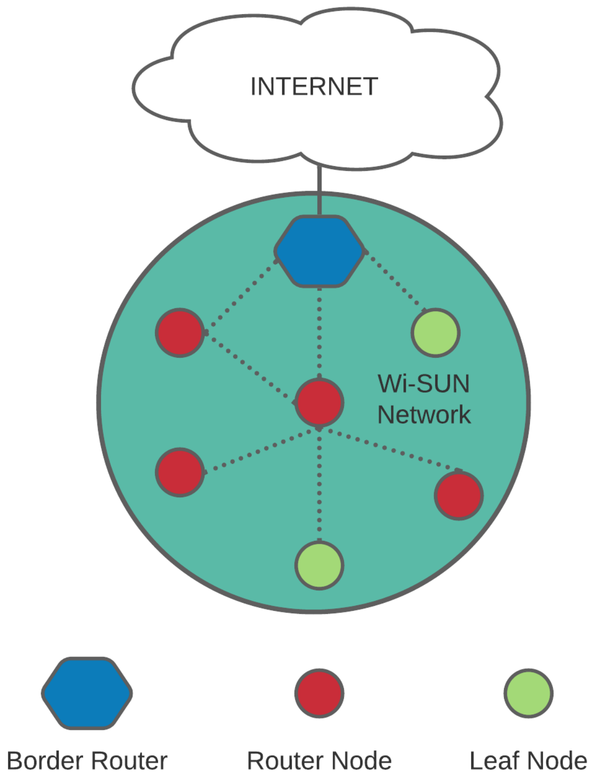


Figure 10 – Wi-SUN Topology.

4.2 Tests Output

The measurements creates a channel mask in which the higher interference channels are signalized to be avoided. The Wi-SUN technology has 90 channels for the Brazilian 900MHz ISM band operating at 100 kbps rate, as illustrated in Figure 11. Bluetooth has 40 channels across the 2.4GHz ISM spectrum channel arrangement as demonstrated in Figure 12. Both technologies use the frequency-hopping technique to avoid interference from other devices operating in the same spectrum, such as *Wireless Local Area Network* (WLAN).

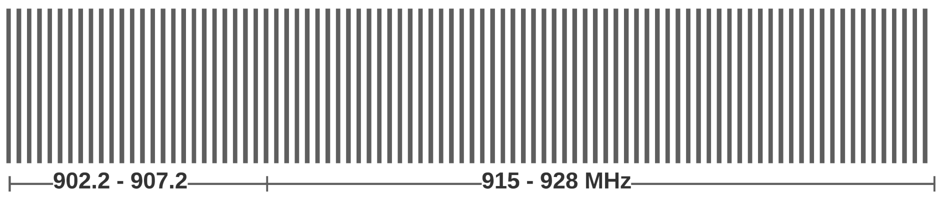


Figure 11 – Wi-SUN channel mapping for Brazil.

This work performs the spectrum evaluation by measuring the channel level by searching interfering signals while observing the nodes communicating. The transmitter node signal level is kept constant at a predetermined value, while the interferer node

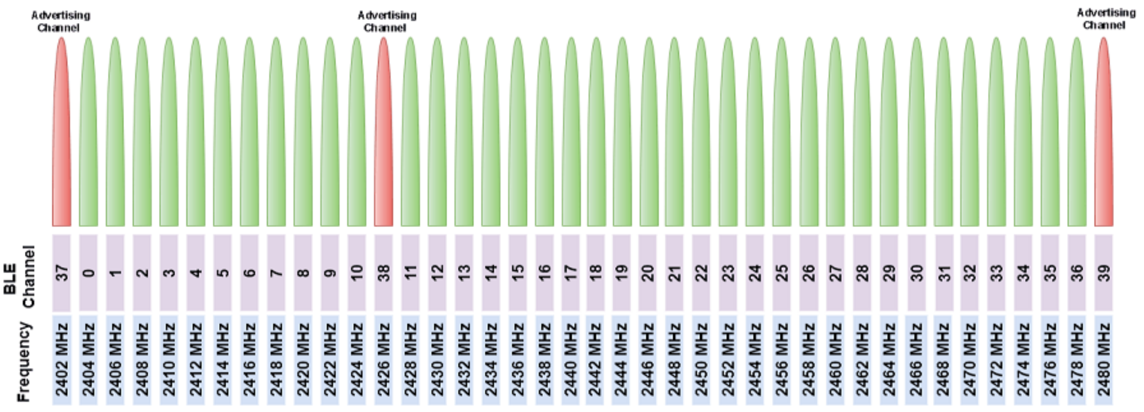

Advertising Channel
Advertising Channel
Advertising Channel
BLE Channel
Frequency
37 0 1 2 3 4 5 6 7 8 9 10 38 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 39
2402 MHz 2404 MHz 2406 MHz 2408 MHz 2410 MHz 2412 MHz 2414 MHz 2416 MHz 2418 MHz 2420 MHz 2422 MHz 2424 MHz 2426 MHz 2428 MHz 2430 MHz 2432 MHz 2434 MHz 2436 MHz 2438 MHz 2440 MHz 2442 MHz 2444 MHz 2446 MHz 2448 MHz 2450 MHz 2452 MHz 2454 MHz 2456 MHz 2458 MHz 2460 MHz 2462 MHz 2464 MHz 2466 MHz 2468 MHz 2470 MHz 2472 MHz 2474 MHz 2476 MHz 2478 MHz 2480 MHz

Figure 12 – Bluetooth worldwide channel mapping [32]

transmits the interfering signal in the same channel and adjacent ones. As recommended by Anatel, the limit corresponds to the condition of no packet errors during the first 14 seconds of observation[30].

4.3 Laboratory Environment

The signals' characterization parameters are presented in Table 6. This work adopted two scenarios on an anechoic chamber, utilizing Wi-SUN at 900MHz and Bluetooth at 2.4GHz, and generating interference on the same and adjacent channels, as previously mentioned. At the receiver the The *Packet Error Rate* (PER), that is the rate of erroneous packets received, indicates how much the interference degrades the communication link.

Table 6 – Characterization of tested signals.

Scenario	A	B
Technology	Wi-SUN	Bluetooth
Power	20dBm	
Modulation	2-GFSK	
Frequency	915MHz	2.4GHz
Bit Rate	100kbps	1Mbps
Channel Bandwidth	200kHz	2MHz

4.3.1 Receiver Operational Level Offset

Initially, the operational levels of the evaluated SDR receiver were obtained and compared to the level received at the reference spectrum analyzer, according to the procedure described in Section 3.3.1. Table 7 is an example of obtained channel offset for Wi-SUN and Bluetooth technology on their channel 0 and 37, 902.2MHz and 2402MHz respectively. The transmitted Bluetooth waveform is shown in Figure 13.

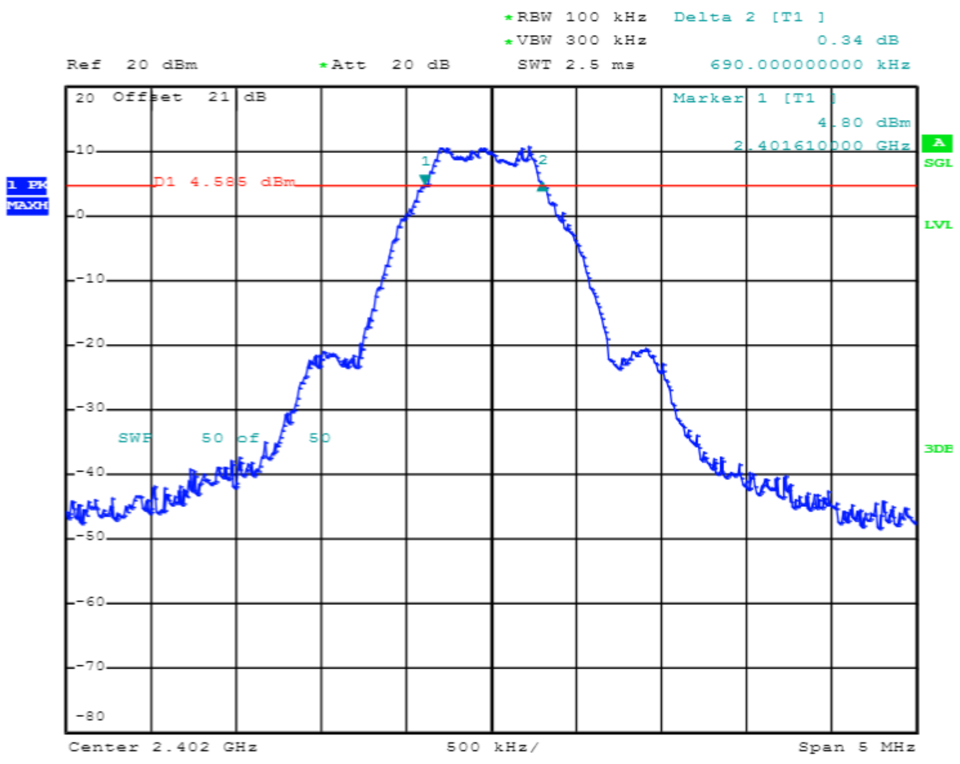


Figure 13 – Bluetooth signal on channel 0.

Table 7 – Channel receiver level difference between LimeSDR and spectrum analyzer.

Technology	**LimeSDR**	**Spectrum Analyzer**	**Offset**
Wi-SUN	-98.25	-97.47	-0.22dB
Bluetooth	-78.48	-78.59	-0.11dB

4.3.2 Interference Evaluation

The interference behavior in the communication link was analyzed by observing co-channel and adjacent channel interference. At the receiver node, the PER reflects the influence of the interference in the communication link. Figure 14 shows an example of a co-channel and adjacent channel interference waveform. The reference spectrum analyzer collected the emissions.

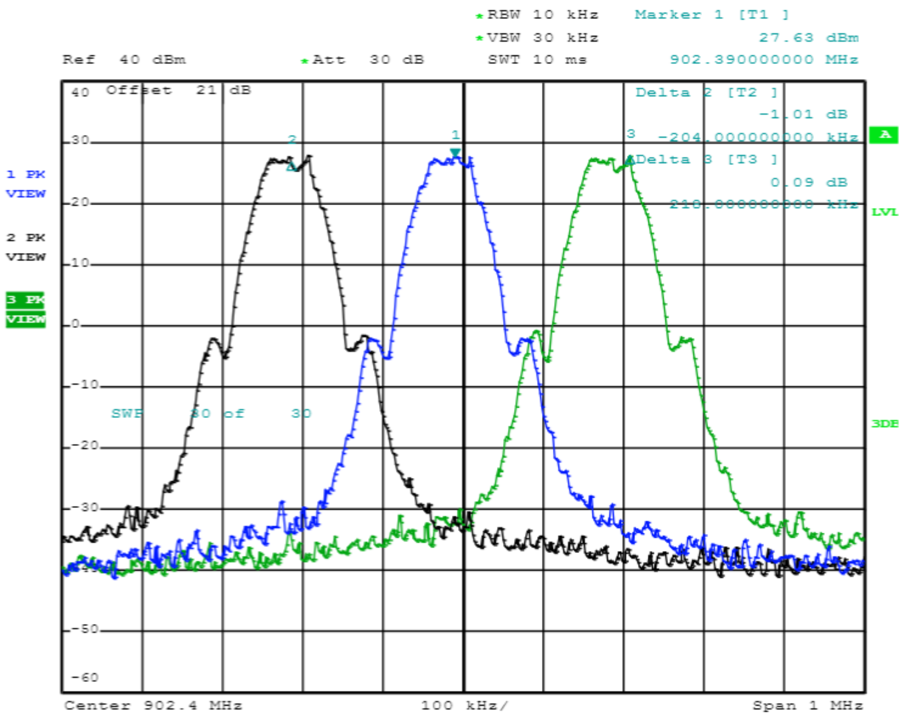


Figure 14 – Wi-SUN signals on channels 0, 1 and 2.

4.3.3 Co-Channel Interference Test Results

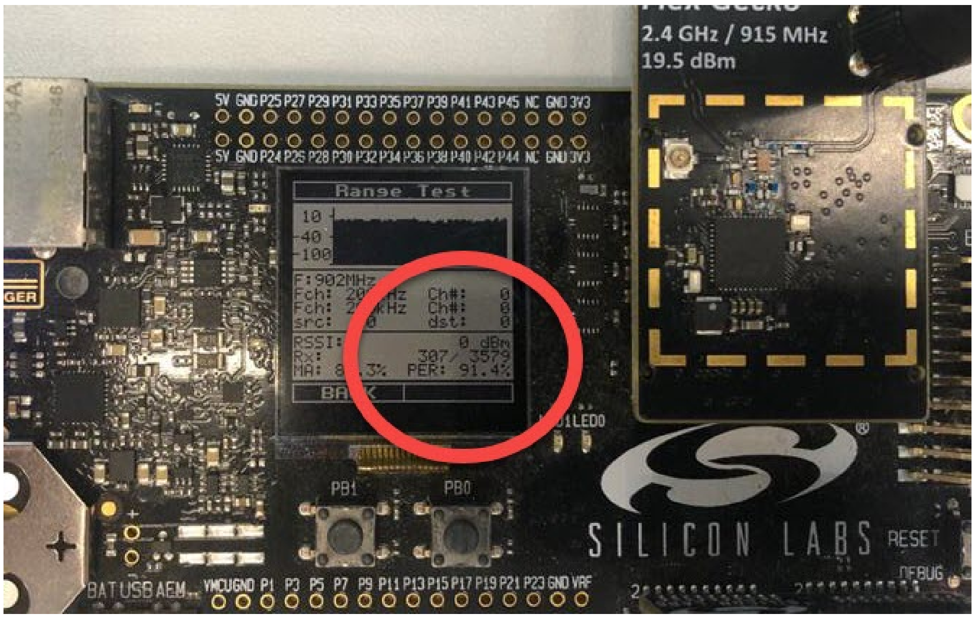


Figure 15 – Wi-SUN PER on channel 0 in co-channel interference condition.

As illustrated in Figure 5, two nodes are communicating in a determined channel and a third node is generating the interference signal in the same channel. In this experiment, three different channels were used for each technology, to check the behavior in the entire operating band. The results were collected, and on the receiving node, the PER was observed. Figure 15 show the results collected on channel 0 using the Wi-SUN physical layer. Tables 8 and 9 shows that an unwanted signal in the same channel interfere the link at the highest level, causing a high packet error rate. Another point that could be observed is that Bluetooth suffers slightly less than Wi-SUN.

Table 8 – Wi-SUN co-channel interference behavior.

Frequency	Channel	PER
902.2 MHz	0	91.4
916 MHz	30	97.3
927.8 MHz	89	93.7

Table 9 – Bluetooth co-channel interference behavior.

Frequency	Channel	PER
2402 MHz	37	90.4
2442 MHz	18	95.3
2478 MHz	36	92.7

4.3.4 Adjacent Channel Interference Test Results

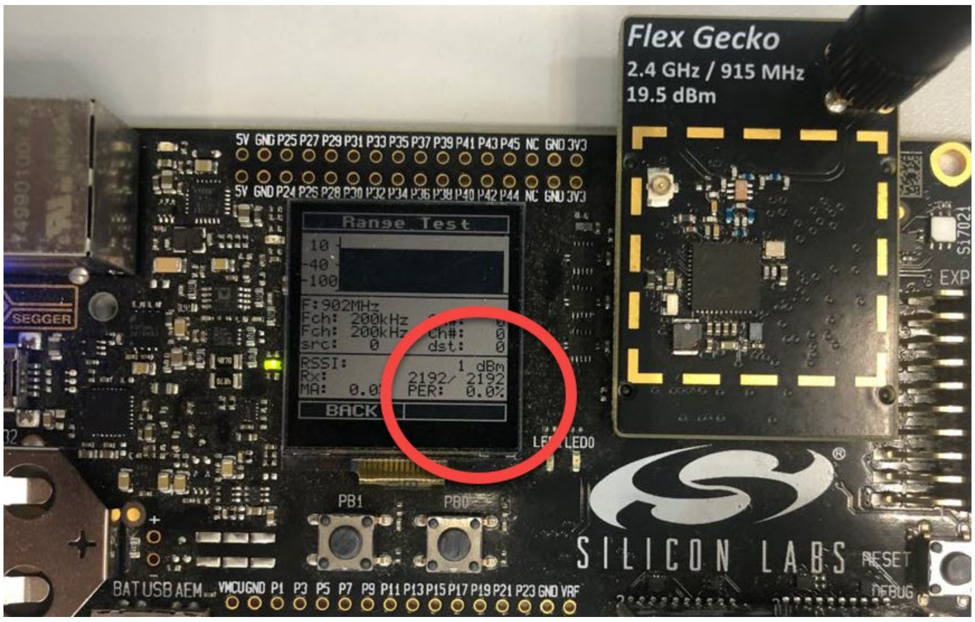

```
Range Test
10
-40
-100
F:902MHz
Fch: 200kHz
Fch: 200kHz     Ch#:     0
src:     0      dst:     0
RSSI:              1 dBm
Rx:          2192/ 2192
MA:    0.0   PER:   0.0%
BACK
```

Figure 16 – Wi-SUN PER on channel 0 in adjacent interference condition.

As mentioned in Chapter 3, two nodes are communicating in a determined channel and a third node generates the unwanted signal in the adjacent channel. For the first channel, it was adopted $n + 1$ channel, and for the last channel, it was adopted $n - 1$ channelization. As in the previous experiment, three different channels were used for each technology, to check the behavior in the entire operating band. The data was collected, and on the receiving node, the PER was observed. Figure 16 shows the error ratio collected on channel 0 using the Wi-SUN physical layer and the results were summarized in Tables 10 and 11. As it can observed, in both technologies, adjacent channel interference could not cause an influence on packet error ratio, which means, an excellent filtering process.

Table 10 – Wi-SUN adjacent interference behavior.

Frequency	**Channel**	**PER**
902.2 MHz	0	0
916 MHz	30	0
927.8 MHz	89	0

Table 11 – Bluetooth adjacent interference behavior.

Frequency	**Channel**	**PER**
2402 MHz	37	0
2442 MHz	18	0
2478 MHz	36	0

4.3.5 Graphical Results

Analyzing both scenarios measurements, shown in Tables 8, 9, 10 and 11, it can be observed that co-channel interference must be avoided, because the PER is too high. As the receiver distance from transmitting and interference node are the same, the received signal strengths are the same, which makes it impossible to distinguish between them. On other hand, the adjacent channels can be used because the interference causes no error on the link.

Figure 17 presents the spectrum on scenario A using LimeSDR to measure. It was possible to observe the filter sharp attenuation of 30dB from the crest between two adjacent channels. Thus the receiving node demodulates the transmitting packets correctly. The n+1 channel is approximately 40dB below, in comparison to the co-channel interference and the n+2 channel is around 60dB below. The noise floor of the software-defined radio is around -97dBm, which means that any signal at this level will be considered as white noise.

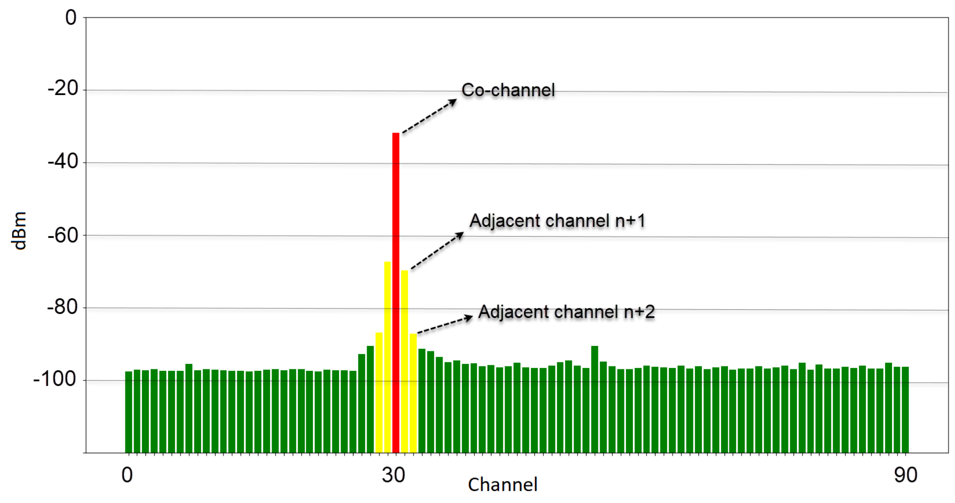


Figure 17 – Spectrum analyzes of filter attenuation in adjacent channels.

To calculate the channel occupation, the higher measured value of a link channel is taken as reference. A level 10.3 dB below this value, it should not be considered as a harmful interference, since the radio can demodulate the signal at a BER of 0.01% [33]. For this scenario, the maximum received value was -32.2dBm at channel 30 (916MHz) and all values under -42.5dBm were considered as susceptible interference. This way, the occupancy graph is generated considering this threshold of harmful interference.

Figure 18 shows the behavior of measured levels captured by the software-defined radio for the full range of possible channels at the anechoic chamber. The interference is locked on channel 30 (916MHz) and all the other channels are free. In this case, it confirms a 100% channel occupancy.

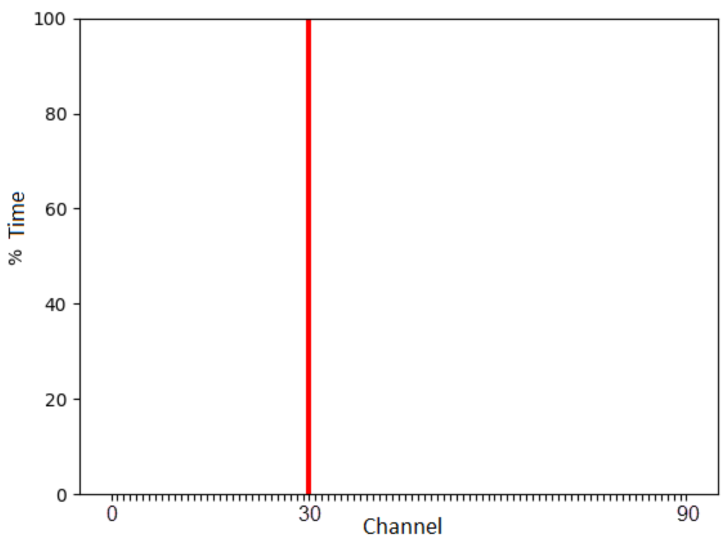
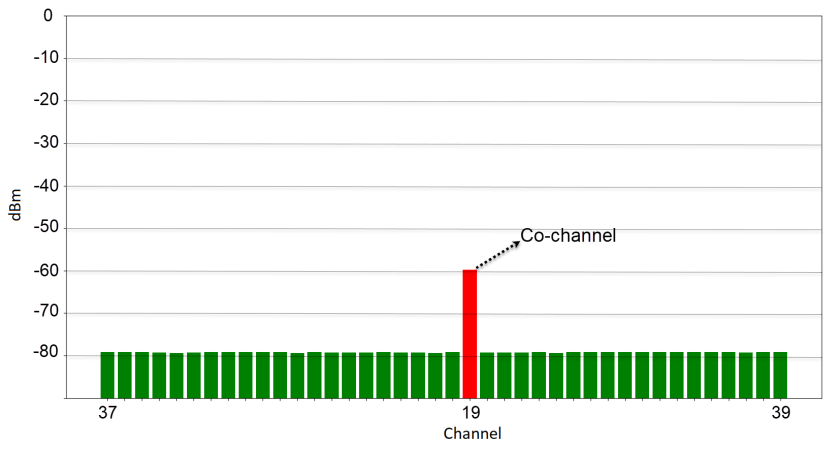
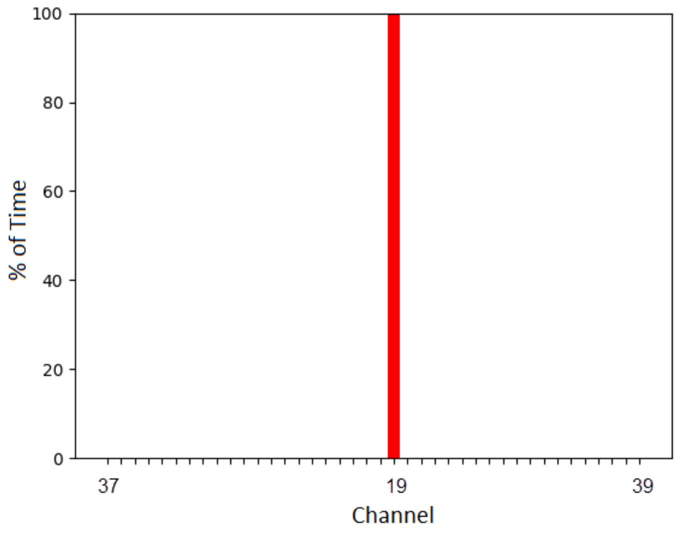


Figure 18 – Spectrum analyzes in 90 Wi-SUN channels with channel 30 occupied.

On scenario B, the Bluetooth technology behavior is analyzed. On Figure 19a, it can be observed that the noise floor is higher than the observed on scenario A, because the Bluetooth signal bandwidth is larger, resulting in a higher channel power. Figure 19b shows the interference detected on channel 19 of Bluetooth technology and the channel map of this analysis.



(a)



(b)

Figure 19 – Power level from measured channels: (a) Bluetooth power (b) and occupancy per channel.

4.4 Field Environment

In last experiment, it was utilized the LimeSDR in a field environment to collect and evaluate how the channels are occupied in a real deployed architecture. This solution uses Wi-SUN technology and it is composed by 1 gateway and 26 nodes, responsible to transmit the data from street lights. For this test, all nodes are responding to *Internet Control Message Protocol* (ICMP) ping packets during the measurement time, to generate traffic and occupy more channels to check LimeSDR channel detection capacity in real world, as it can be seen on Figure 20. In this experiment, it was possible to check that some channels are more occupied than other, this occurs because others communications and signals can be present in real world. Analyzing this Scenario, the field engineer can classify and select the best channels according to the project requirements.

This experiment takes 1260 seconds to be realized. As mentioned before in this work, each channel was measured by 14 seconds and the acceptable occupation is 400ms per channel, resulting in a maximum percentage of approximately 2.86%. Channels with percentage of occupation below 2.86% was classified as good, represented as green colored bars. Channels with occupation between 2.86% up to 5.72% can be classified as moderate channels, represented as yellow bars. And, channels above the threshold of 5.72% was classified by bad channels, represented as red bars on the graph.

Figure 21 shows the street light controllers installed on LED street lights in São Paulo city.

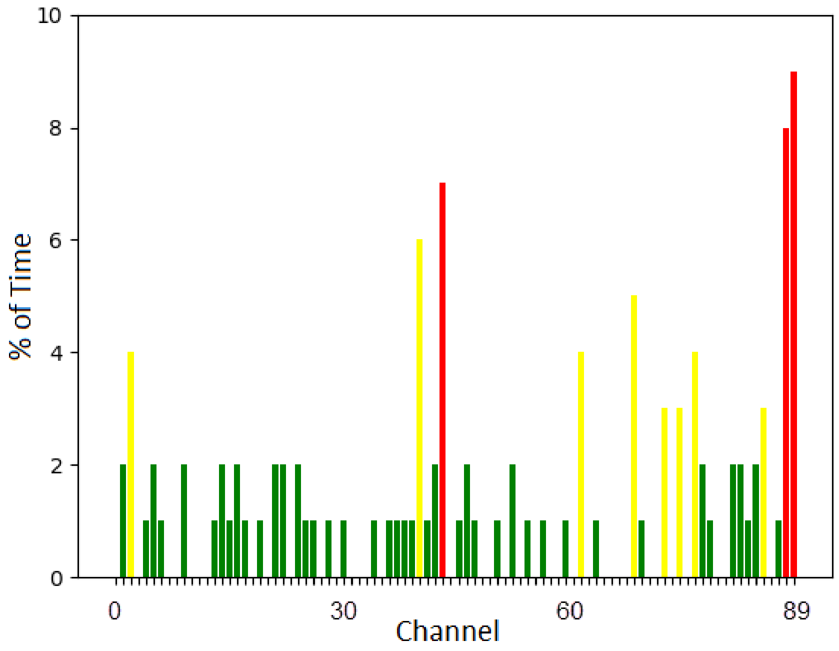


Figure 20 – Channel occupancy measured in a Wi-SUN solution.

Figure 21 – Street Light Solution.

4.5 Summary

This chapter shows the scenarios for collect measures results for two technologies, Wi-SUN and Bluetooth, in different ISM bands, 915MHz, and 2.4GHz. It analyzes the influence of interference on the same and adjacent channels on channel occupation calculation.

Also, it shows the graphical results of channels average power and occupation and presents a reference methodology to qualify them. It presents a real Scenario that tests the device developed in this work in the field environment.

5 Conclusions

This work presented a *Software Defined Radio* (SDR) implementation using the LimeSDR board. This low-cost equipment made it possible to develop a spectrum qualifier for ISM bands at 900MHz and 2.4GHz. It tests two imminent wireless technologies for IoT, Wi-SUN and Bluetooth. It had good power level measurements accuracy compared to a reliable spectrum analyzer since the measurement errors are meager.. As a result of the analysis generates a channel map indicating the channels' average power and occupancy, allowing the field engineer to select the best channels that satisfy future wireless IoT solutions requirements. Therefore, the main contributions are:

- Development of an application that runs on a a portable device that can help field engineers to measure the frequency spectrum before deploying new solutions;
- Detailed measures of the physical layer of the Wi-SUN FAN and Bluetooth technologies;
- Methodology for acquiring data through practical experiments;
- Evaluation of the Wi-SUN and Bluetooth channels under RF interference;
- Data analysis of the power measurements collected from the observed channels;
- Contribution to a possible product launch for IoT project analysis.

A lot of captured data from media was acquired in this experiment to analyze the channels for Bluetooth and Wi-SUN technologies. With these data and their analysis, it is possible to confirm that is important to choose the right channels to deploy wireless solutions for Smart Cities application. The wireless spectrum of the cities, mainly in central places, is very busy. It can be observed that Wi-SUN and Bluetooth suffers with co-channel interferences, and are robust against co-channel interferences. Using the channel masks generated by the results of the tests, it is possible to select the best channels to communicate and improve the performance of the technology.

As a contribution to the scientific community, all codes used in the proposed device development are available in annexes A, B, C, D, E, and F. With this on hand, researchers can perform the same tests to make other types of analysis and work.

5.1 Proposals for future work

This study collaborates with frequency spectrum qualifiers for wireless technologies deployments. As the spectrum power measurements are available, using this data is possible to make other analyses, for example, the traffic evaluation, the probability of congesting, the interference model.

As a suggestion for future works, it is recommended:

- Detailed analysis of the data measures to detect the kind of interferences;
- Machine learn techniques implementation to detect data patterns and make a correlation to discover particular events on the frequency spectrum, for example, detection of illegal radios signals on the air;
- A Real-time machine learning model for pattern recognition application of frequency spectrum behavior. As an example detects when stores close and open based on spectrum occupancy caused by information traffic;
- Measurement on large cities central places or cities that have a smart meter and smart street light solutions implemented to evaluate frequency spectrum occupation;
- Development of real-time spectrum qualifier that renews channel mask on radios;
- Development of a traffic analyzer for wireless IoT communication;
- Analysis of multi technologies scenario to evaluate the interferences between them, observing the frequency spectrum.

Appendix

ANNEX A – GNU Radio Companion flowgraph file

This is the GNU Radio generated file based on Figure 8.

filename: meas_channel_power_final.grc

************* Code Below *************

```
options:
parameters:
  author: ''
  category: '[GRC Hier Blocks]'
  cmake_opt: ''
  comment: ''
  copyright: ''
  description: ''
  gen_cmake: 'On'
  gen_linking: dynamic
  generate_options: no_gui
  hier_block_src_path: '.:'
  id: power_final
  max_nouts: '0'
  output_language: python
  placement: (0,0)
  qt_qss_theme: ''
  realtime_scheduling: '1'
  run: 'True'
  run_command: '{python} -u {filename}'
  run_options: prompt
  sizing_mode: fixed
  thread_safe_setters: ''
  title: Not titled yet
  window_size: ''
states:
  bus_sink: false
  bus_source: false
  bus_structure: null
```

```
    coordinate: [8, 8]
    rotation: 0
    state: enabled

blocks:
- name: blocks_complex_to_mag_squared_0
  id: blocks_complex_to_mag_squared
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    maxoutbuf: '0'
    minoutbuf: '0'
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [687, 249]
    rotation: 0
    state: true
- name: blocks_moving_average_xx_0
  id: blocks_moving_average_xx
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    length: '50000'
    max_iter: '4000'
    maxoutbuf: '0'
    minoutbuf: '0'
    scale: 20e-6
    type: float
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [891, 188]
    rotation: 0
    state: true
- name: blocks_nlog10_ff_0
  id: blocks_nlog10_ff
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    k: '0'
    maxoutbuf: '0'
    minoutbuf: '0'
    n: '10'
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [1114, 242]
    rotation: 0
    state: true
- name: bw
  id: parameter
  parameters:
    alias: ''
    comment: ''
    hide: none
    label: bw
    short_id: ''
    type: eng_float
    value: '200000'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [535, 14]
    rotation: 0
    state: enabled
- name: center_freq
  id: parameter
  parameters:
    alias: ''
    comment: ''
    hide: none
    label: center_freq
    short_id: ''
    type: eng_float
    value: '902200000'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [311, 14]
    rotation: 0
    state: enabled
- name: filename
  id: parameter
  parameters:
    alias: ''
    comment: ''
    hide: none
    label: filename
    short_id: ''
    type: str
    value: /home/leo/mestrado/teste.txt
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [626, 14]
    rotation: 0
    state: enabled
- name: filerepeater_VectorToTxtFile_0
  id: filerepeater_VectorToTxtFile
  parameters:
    WriteTimeHeader: 'False'
    affinity: ''
    alias: ''
    append: 'True'
    comment: ''
    filename: filename
    frequency: '1'
    notes: ''
    precision: '2'
    sampleRate: samp_rate
    updateRateSec: '0.1'
    vectorsize: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [1270, 175]
    rotation: 0
    state: true
- name: high_pass_filter_0
  id: high_pass_filter
  parameters:
    affinity: ''
    alias: ''
    beta: '6.76'
    comment: ''
    cutoff_freq: bw/2
    decim: '1'
    gain: '1'
    interp: '1'
    maxoutbuf: '0'
    minoutbuf: '0'
    samp_rate: samp_rate
    type: fir_filter_ccf
    width: bw/10
    win: firdes.WIN_HAMMING
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [253, 170]
    rotation: 0
    state: enabled
```

```
- name: limesdr_source_0
  id: limesdr_source
  parameters:
    affinity: ''
    alias: ''
    allow_tcxo_dac: '0'
    analog_bandw_ch0: 1.5e6
    analog_bandw_ch1: 1.5e6
    calibr_bandw_ch0: 2.5e6
    calibr_bandw_ch1: 2.5e6
    channel_mode: '0'
    comment: ''
    dacVal: '180'
    digital_bandw_ch0: samp_rate
    digital_bandw_ch1: samp_rate
    filename: ''
    gain_dB_ch0: '40'
    gain_dB_ch1: '1'
    lna_path_ch0: '255'
    lna_path_ch1: '2'
    maxoutbuf: '0'
    minoutbuf: '0'
    nco_freq_ch0: '0'
    nco_freq_ch1: '0'
    oversample: '0'
    rf_freq: center_freq
    samp_rate: samp_rate
    serial: ''
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [9, 185]
    rotation: 0
    state: true
- name: low_pass_filter_0
  id: low_pass_filter
  parameters:
    affinity: ''
    alias: ''
    beta: '6.76'
    comment: ''
    cutoff_freq: bw/2
    decim: '1'
    gain: '1'
    interp: '1'
    maxoutbuf: '0'
    minoutbuf: '0'
    samp_rate: samp_rate
    type: fir_filter_ccf
    width: bw/10
    win: firdes.WIN_HAMMING
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [466, 185]
    rotation: 0
    state: enabled
- name: samp_rate
  id: parameter
  parameters:
    alias: ''
    comment: ''
    hide: none
    label: samp_rate
    short_id: ''
    type: eng_float
    value: '1000000'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [422, 14]
    rotation: 0
    state: enabled

connections:
```

```
- [blocks_complex_to_mag_squared_0, '0', blocks_moving_average_xx_0, '0']
- [blocks_moving_average_xx_0, '0', blocks_nlog10_ff_0, '0']
- [blocks_nlog10_ff_0, '0', filerepeater_VectorToTxtFile_0, '0']
- [high_pass_filter_0, '0', low_pass_filter_0, '0']
- [limesdr_source_0, '0', high_pass_filter_0, '0']
- [low_pass_filter_0, '0', blocks_complex_to_mag_squared_0, '0']

metadata:
  file_format: 1
```

ANNEX B – C code for calculate mean power and channel occupation

This is a code in C language responsible for parsing a file with measures, calculate mean and occupancy and write it to a file, for a future plotting.

filename: line_read.c

************* Code Below *************

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define LSIZ 128
#define RSIZ 10

int main(int argc, char *argv[])
{
    FILE *fp;
    FILE *meas_file;
    char *line = NULL;
    size_t len = 0;
    ssize_t read;
    float mean = 0;
    uint32_t counter = 0;
    char str[80];
    /*occupancy*/
    uint32_t occ_counter = 0;
    uint8_t occupancy = 0;

    meas_file = fopen("../mestrado_file_plot/meas.txt", "w+");
    //meas_file = fopen("meas.txt", "w+");

    for (int i = 1; i < argc; i++)
```

```
{
    counter = 0;
    mean = 0;
    occ_counter = 0;

    printf("argv %d: %s\n", i, argv[i]);

    /* To calculate signal mean power from file */
    fp = fopen(argv[i], "r");
    if (fp == NULL)
        exit(EXIT_FAILURE);

    while ((read = getline(&line, &len, fp)) != -1)
    {
        //printf("Retrieved line of length %zu:\n", read);
        //printf("%s", line);
        if (isdigit(line[1]))
        {
            float j = atof(line);
            //printf("%02f\n", i);
            if(j > (-150)) {
                counter++;
                mean += j;
            }
            if(j > (-65)) {
                occ_counter++;
            }
            //printf("mean %02f\n", mean);
        }
    }
    mean /= counter;
    occupancy = (occ_counter/counter) * 100;
    //printf("occupancy: %02d \n", occupancy);
    //printf("final mean %02f\n", mean);
    sprintf(str, "%02d,%f,%03d\n", (i - 1), mean, occupancy);
    printf("%s", str);
    if(i < 100) {
        fwrite(str , 1 , strlen(str) , meas_file );
    }
```

```
        memset(str, 0x00, 80);
        fclose(fp);
    }
    fclose(meas_file);
    if (line)
        free(line);
    exit(EXIT_SUCCESS);
}
```

ANNEX C – Python code for plotting channel mean power

This is a code in python language responsible for parsing the file with power measures and plot in bar graph format.

filename: plot_channels.py

************* Code Below *************

```
import matplotlib.pyplot as plt
import csv

x = []
y = []
freq_label = []
color_array = []

with open('meas.txt','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(int(row[0]))
        y.append((120 - abs(float(row[1]))))
        if (float(row[1]) < -90) :
                color_array.append('green')
        elif (float(row[1]) > -90 and float(row[1]) < -60) :
                color_array.append('yellow')
        else :
                color_array.append('red')

print(color_array)

freq_label = x
plt.bar(x, y, bottom=-120, tick_label=freq_label, width=0.8, color=color_array)
plt.ylim(top=0)
plt.xlabel('Channel')
plt.ylabel('dBm')
plt.title('Power Level from measured channels\nWi-SUN')
```

```
#plt.legend()
plt.show()
```

ANNEX D – Script for media scan

This is a bash script for scan all channels. Note, that xx, yy and zz should be changed respectively from 22 up to 74, 152 up to 278 and 402 up to 480.

filename: scan_media.sh

************* Code Below *************

```
LimeQuickTest
python power_final.py --center-freq=90xx00000 --filename=9022.txt
LimeQuickTest
python power_final.py --center-freq=90xx00000 --filename=9074.txt

LimeQuickTest
python power_final.py --center-freq=9yyy00000 --filename=9152.txt
LimeQuickTest
python power_final.py --center-freq=9yyy00000 --filename=9278.txt
LimeQuickTest

python power_final.py --center-freq=2zzz000000 --filename=2402.txt
LimeQuickTest
LimeQuickTest
python power_final.py --center-freq=2zzz000000 --filename=2480.txt
```

ANNEX E – Python code for plotting channel occupancy

This is a code in python language responsible for parsing the file with channel occupancy and plot in bar graph format.

filename: plot_channels_occupancy.py

************* Code Below *************

```
import matplotlib.pyplot as plt
import csv

x = []
y = []
freq_label = []
color_array = []

with open('meas.txt','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(int(row[0]))
        y.append(int(row[2]))
        if (int(row[2]) < 5) :
                color_array.append('green')
        elif (int(row[2]) > 5 and int(row[2]) < 40) :
                color_array.append('yellow')
        else :
                color_array.append('red')

print(color_array)

freq_label = x
plt.bar(x, y, bottom=0, tick_label=freq_label, width=0.8, color=color_array)
plt.ylim(top=100)
plt.xlabel('Channel')
plt.ylabel('%')
plt.title('Occupancy from measured channels\nWi-SUN')
```

```
#plt.legend()
plt.show()
```

ANNEX F – Python code for plotting channel power histogram

This is a code in python language responsible for parsing the file with power measures and plot in histogram graph format.

filename: plot_channel_hist.py

************* Code Below *************

```
import matplotlib.pyplot as plt
import csv
import numpy as np
import scipy.stats

x = []

with open('9022.txt','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append((float(row[0])))

mean = np.mean(x)
result = scipy.stats.describe(x, ddof=1, bias=False)
print(result)

plt.hist(x, bins=np.linspace(result.minmax[0] - 3, result.minmax[0] + 3, num=300))#
plt.xlabel('Level (dBm)')
plt.ylabel('Occurrence')
plt.title('Histogram from channel\n')
#plt.legend()
plt.show()
```

Bibliography

[1] Jodi Thurtell. *What is IoT? Check what is IoT and how it works.* 2019. URL: https://www.iot-now.com/2019/07/07/97056-what-is-iot.

[2] Wi-SUN Alliance. *Comparing IoT Networks at a Glance: How Wi-SUN FAN stacks up against LoRaWAN and NB-IoT.* 2019. URL: https://www.wi-sun.org/wp-content/uploads/WiSUN-Alliance-Comparing-IoT-Networks-2019-Nov-A4.pdf.

[3] Markus Rothmuller and Sam Barker. *IOT - The Internet of Transformation 2020.* Apr. 2020. URL: http://www.juniperresearch.com.

[4] International Telecommunication Union. *Rec. ITU-R SM.1056-1: Limitation of Radiation from Industrial, Scientific and Medical (ISM) Equipment.* 2007.

[5] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks". In: *IEEE Std 802.15.4g-2012 (Amendment to IEEE Std 802.15.4-2011)* (2012), pp. 1–252. DOI: 10.1109/IEEESTD.2012.6190698.

[6] Wi-SUN. *Wi-SUN Alliance.* July 2020. URL: https://www.wi-sun.org/ (visited on 08/23/2020).

[7] Silicon Labs. *Wi-SUN Silicon Labs.* Jan. 2021. URL: https://www.silabs.com/wireless/wi-sun.

[8] Texas Instruments. *Wi-SUN | Overview | Wireless Connectivity.* Jan. 2021. URL: https://www.ti.com/wireless-connectivity/wi-sun/overview.html.

[9] Renesas. *Sub-GHz/Wi-SUN Protocol Stack.* Jan. 2021. URL: https://www.renesas.com/us/en/software-tool/sub-ghzwi-sun-protocol-stack.

[10] Bluetooth SIG. *Bluetooth Core Specification 5.3.* 2021. URL: https://www.bluetooth.com/specifications/specs/core-specification/.

[11] Lime Microsystems. *LimeSDR Mini.* Apr. 2020. URL: https://limemicro.com/products/boards/limesdr-mini/.

[12] GNU Radio. *About GNU Radio.* Jan. 2021. URL: https://www.gnuradio.org/about/.

[13] Alexander M Wyglinski, Robin Getz, Travis Collins, and Di Pu. *Software-Defined Radio for Engineers.* Artech House, 2018.

[14] Dayan Adionel Guimarães and Tiago Magalhães Reis. “Impairment Models for Performance Evaluation of Digital Modulations in the Millimeter Wave Range”. In: *Journal of Communication and Information Systems* 33 (2018).

[15] Nordic Semiconductor. *NRF24L01 Datasheet*. 2008. URL: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf.

[16] Aniruddha Chandra, Ananya Patra, and Chayanika Bose. “Effect of Imperfect Phase Synchronization on the Error Rate Performance of m-PSK in Rayleigh, Rician and Nakagami Fading Channels”. In: *2010 Annual IEEE India Conference (INDICON)*. IEEE. 2010, pp. 1–4.

[17] Dayan Adionel Guimaraes. *Digital Transmission: A Simulation-aided Introduction with VisSim/Comm*. Springer Science & Business Media, 2010.

[18] Joseph Mitola. “Software Radios: Survey, Critical Evaluation and Future Directions”. In: *IEEE Aerospace and Electronic Systems Magazine* 8.4 (1993), pp. 25–36.

[19] RI Lackey and Donald W Upmal. “Speakeasy: The military Software Radio”. In: *IEEE Communications Magazine* 33.5 (1995), pp. 56–61.

[20] Alexander M Wyglinski and Di Pu. *Digital Communication Systems Engineering with Software-Defined Radio*. Artech House, 2013.

[21] Peter B Kenington. *RF and Baseband Techniques for Software-Defined Radio*. Artech House on Demand, 2005.

[22] Lime Microsystems. *LMS7002M Datasheet*. Oct. 2020. URL: https://limemicro.com/technology/lms7002m/.

[23] A. Fanan, N. Riley, M. Mehdawi, M. Ammar, and M. Zolfaghari. “Comparison of spectrum occupancy measurements using software defined radio RTL-SDR with a conventional spectrum analyzer approach”. In: *2015 23rd Telecommunications Forum Telfor (TELFOR)*. 2015, pp. 200–203. DOI: 10.1109/TELFOR.2015.7377447.

[24] Q. Zhang, B. Nikfal, and C. Caloz. “High-resolution real-time spectrum sniffer for wireless communication”. In: *2013 International Symposium on Electromagnetic Theory*. 2013, pp. 64–66.

[25] Silicon Labs. *BGM121/BGM123 Blue Gecko Bluetooth Data Sheet*. 2019. URL: https://br.mouser.com/datasheet/2/368/bgm12x-datasheet-1368486.pdf.

[26] Silicon Labs. *EFR32MG12 2.4 GHz/915 MHz Dual Band Data Sheet*. 2017. URL: https://br.mouser.com/datasheet/2/368/brd4164a-rm-1100229.pdf.

[27] Linx. *ANT-916-CW-HWR Data Sheet*. 2017. URL: https://linxtechnologies.com/wp/wp-content/uploads/ant-916-cw-hwr.pdf.

[28] Linx. *ANT-2.4-CW-HWR Data Sheet*. 2017. URL: https://linxtechnologies.com/wp/wp-content/uploads/ant-2.4-cw-hwr.pdf.

[29] Anatel. *Ato nº 946*. 2018. URL: https://informacoes.anatel.gov.br/legislacao/atos-de-certificacao-de-produtos/2018/1169-ato-946.

[30] Anatel. *Ato nº 14448*. 2017. URL: https://informacoes.anatel.gov.br/legislacao/component/content/article/96-atos-de-certificacao-de-produtos/2017/1139-ato-14448.

[31] Joint Committee for Guides in Metrology. *International Vocabulary of Metrology: Basic and general concepts and associated terms*. 2012.

[32] MathWorks. *BLE Channel Selection Algorithms*. Feb. 2021. URL: https://la.mathworks.com/help/comm/ug/ble-channel-selection-algorithms.html.

[33] Silicon Labs. *EFR32MG12 datasheet*. Apr. 2020. URL: https://www.silabs.com/documents/public/data-sheets/efr32mg12-datasheet.pdf.