Pedro Miguel Vilaça dos Santos

Licenciado em Ciências de
Engenharia Electrotécnica e de Computadores

Methodology for WSN communication technologies automated field tests

Dissertação para obtenção do Grau de
Mestre em Engenharia Electrotécnica e de Computadores

Orientador:	Doutor Pedro Miguel Negrão Maló, Prof. Auxiliar, Universidade Nova de Lisboa
Co-orientador:	Mestre Bruno Miguel Pereira de Almeida, CEO, Unparallel Innovation

Júri:

Presidente:	Doutor Luís Filipe dos Santos Gomes
Arguente:	Doutor Paulo Miguel de Araújo Borges Montezuma de Carvalho
Vogal:	Doutor Pedro Miguel Negrão Maló

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2015

Methodology for WSN communication technologies automated field tests

"The only source of knowledge is experience."
Albert Einstein

Acknowledgements

Quero começar por agradecer ao meu orientador, o Professor Pedro Maló, dedicação, disponibilidade e paciência demonstrada durante todos os momentos desta dissertação. Agradeço tambem ao meu coorientador Bruno Almeida, pela disponibilidade prestada e pelo rigor requerido, contribuindo para uma melhoria da qualidade do trabalho.
À empresa Unparallel Innovation um muito obrigado pelo equipamento disponibilizado, sem ele esta dissertação não seria possível.

Aos meus companheiros de tese, Fernando Rosado e Miguel Costa, pelo companheirismo e pelas sugestões que contribuiram para o desenvolvimento desta dissertação.

A todos os colegas que fizeram parte do meu percurso universitário, nomeadamente, Bruno Nascimento, Emanuel Sequeira, Filipe Viegas, Joana Martelo, João Filipe, João Silva, Nuno Vasconcelos, Pedro Almeida, Ricardo Lampreia, Sergio Silva, levo a vossa amizade por muitos e bons anos.
Quero agradecer também aos meus amigos fora do curso, Bruno Cristino, Fernando Bastos, Hugo Borges, Pedro Monteiro, Sónia Moura e Vanessa Marques pelo constante apoio e momentos de lazer proporcionados.

Um agradecimento muito especial a toda a minha família, e em especial aos meus pais e irmã pela compreensão, carinho e muita paciência durante todo o percurso universitário, bem como pela força e motivação que me foram dados. Deixo também uma palavra de agradecimento à Aida, por toda a amizade, amor, dedicação, compreenção e por ter estado sempre presente nos momentos mais difíceis.

A todos os que me acompanharam durante o meu percurso académico e que não mencionei, o meu mais sincero obrigado!

Abstract

Wireless Sensor Networks(WSN) are networks of devices used to sense and act that applies wireless radios to communicate. To achieve a successful implementation of a wireless device it is necessary to take in consideration the existence of a wide variety of radios available, a large number of communication parameters (payload, duty cycle, etc.) and environmental conditions that may affect the device's behaviour. However, to evaluate a specific radio towards a unique application it might be necessary to conduct trial experiments, with such a vast amount of devices, communication parameters and environmental conditions to take into consideration the number of trial cases generated can be surprisingly high. Thus, making trial experiments to achieve manual validation of wireless communication technologies becomes unsuitable due to the existence of a high number of trial cases on the field. To overcome this technological issue an automated test methodology was introduced, presenting the possibility to acquire data regarding the device's behaviour when testing several technologies and parameters that care for a specific analysis. Therefore, this method advances the validation and analysis process of the wireless radios and allows the validation to be done without the need of specific and in depth knowledge about wireless devices.

Keywords: Automation, Testing, Wireless Devices, Wireless Sensor Networks

RESUMO

As Wireless Sensor Networks (WSN) são redes de dispositivos usadas para sentir e atuar que usam rádios sem fios para comunicar. Para alcançar uma implementação bem-sucedida de um dispositivo wireless é necessário ter em consideração a existência de uma ampla variedade de rádios disponíveis, um grande numero de parâmetros de comunicação (payload, duty cycle, etc) e de condições ambientais que podem ter efeito no comportamento do dispositivo. Contudo, para avaliar um rádio específico para uma aplicação singular, pode ser necessário realizar experiências de teste, uma vez que com uma vasta quantidade de dispositivos, parâmetros de comunicação e condições ambientais a ter em consideração o número de experiências de teste pode ser surpreendentemente grande. Assim, fazer experiências de teste para alcançar uma validação manual das tecnologias de comunicação wireless torna-se improprio devido ao elevado número de casos experimentais no terreno. Para ultrapassar esta questão tecnológica uma metodologia automática de teste foi apresentada, apresentando a possibilidade de adquirir dados relativamente ao comportamento do dispositivo ao testar diversas tecnologias e parâmetros que carecem de uma analise especifica. Por conseguinte, este método introduz avanços no processo de validação e analise de rádios sem fios e permite que a validação seja feita sem existir a necessidade de possuir conhecimentos específicos e aprofundados sobre dispositivos wireless.

Palavras-chave: Automação, Testes, Dispositivos sem fios, Redes de Sensores sem Fios

Contents

List of Figures

LIST OF TABLES

LISTINGS

GLOSSARY

BLE Bluetooth Low Energy.

FCT Faculdade de Ciências e Tecnologia.

GUI Graphical User Interface.

I/O Input/Output.

IC Integrated Circuit.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IoT Internet of Things.

ISM Industrial,Scientific and Medical.

LoS Line of Sight.

NLoS Non Line of Sight.

QoS Quality of Service.

RF Radio Frequency.

SoC System-on-Chip.

TTCN-2 Tree and Tabular Combined Notation - Version 2.

UNL Universidade Nova de Lisboa.

WLAN Wireless Local Area Network.

WSN Wireless Sensors Networks.

CHAPTER 1

INTRODUCTION

1.1 Motivation Scenario: Smart Campus

The broad idea behind the Internet of Things (IoT) is that the whole constellation of objects "things" can be connected and interconnected thus allowing them to be monitored, controlled and connected to the Internet. This interconnection of objects can be obtained through Wireless Sensors Networks (WSN), Wireless Local Area Network (WLAN), or other means of connection, and it can lead to advanced awareness and control in some environments through detailed sensing and actuating.

The objects connected to the IoT can be our everyday objects or others that aren't currently connected to the information grid. These objects can be electrical/electronic devices or not devices at all, just common objects such as house equipment, food, clothes, parking lots, city landmarks, etc, like represented in the figure 1.1.

Since the main objective of IoT is the communication, the devices can have wireless communication modules, modules that are usually based on standard communication protocols, such as 802.11, 802.15.1, 802.15.4, etc. This independence of the physical networks can lead us to place IoT devices almost everywhere and a wide deployment of WSN can raise various concerns, concerns like:

Accessibility: A WSN node can be deployed in places with bad access like ceilings, rooftops, the woods, etc.

Power Consumption: Wireless communication modules can be very power hungry because different technologies apply different protocols and therefore different modulations and those modulations have different power needs.

Power Supply: When deploying a WSN node in an inaccessible location it can be difficult to provide a steady and constant power supply requiring, for example, the use of a power-bank instead. This deployment could be compromised if the device drains too much energy.

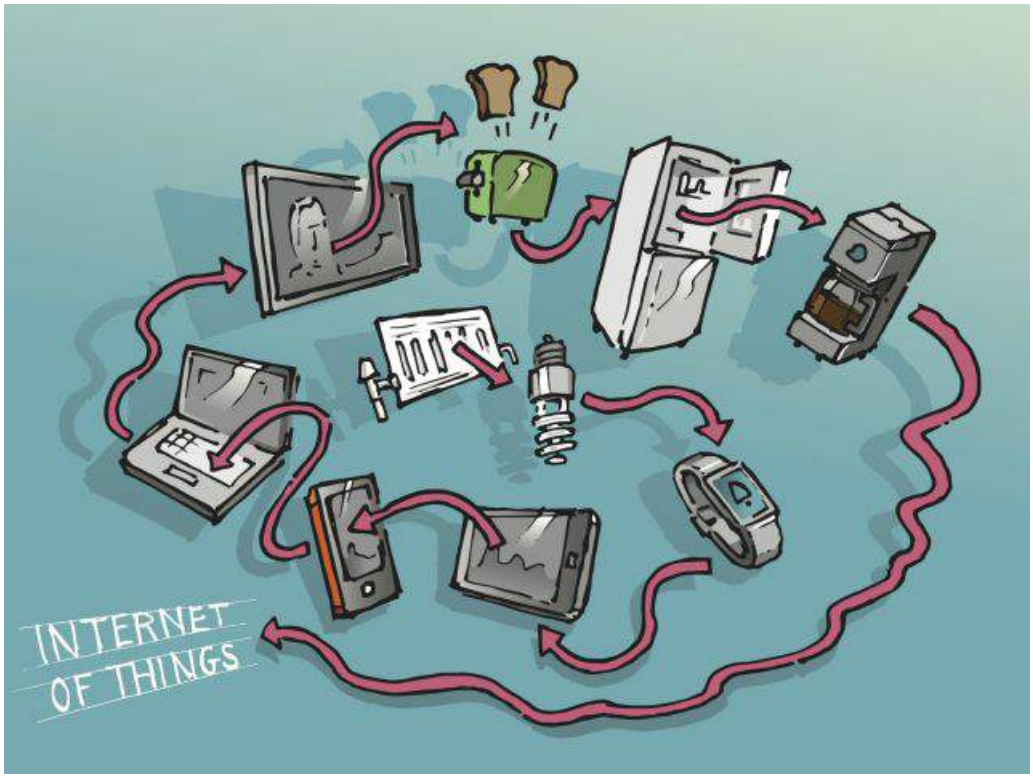


Figure 1.1: IoT network diversity;

For instance, lets consider the outdoors an university campus as our main scenario, the Faculdade de Ciências e Tecnologia (FCT) - Universidade Nova de Lisboa (UNL) campus. The deployment of the IoT on the campus outdoors could be achieved through a WSN and it could have an impact in the institution way of life by improving maintenance operations, gaining energy efficiency, reducing water wastes and increase students and employers satisfaction, by providing at least three major benefits that could include:

Cost Savings: A constant monitoring of luminosity, temperature and humidity of several campus areas could lead to a system with intelligence awareness that could activate and regulate the street illumination instead of the traditional timer clock, and the activation and duration of the gardens sprinklers systems instead of the traditional timer clock or manual activation. This particular situations could lead to a reduce of the electricity and water consumption therefore a cost saving;

Control: A student/employer could know in real time which parking lots were available even before arrive at the campus. The institution administration could have access to more data such as the production of electricity by photovoltaic panels.

Communication: The use of a WSN attached to the power grid could warn the maintenance services of faulty equipments such as photovoltaic panels, street lamps, and other; and in the water grid troubles like plumbing leaks or malfunctions in the gardens sprinklers systems;

1.1.1 FCT- UNL Case Scenario

To achieve a successful deployment of any wireless network it's necessary to choose the technology and know their capacities and limitations, but knowing their properties it is not enough, it is necessary to take into account the properties of the radio signal and the deployment environment. Failing to take under advisement this concerns can lead to a wireless network with bad coverage areas and the existence of dead-zones(zones without radio signal). To minimize those planning faults a planed was designed following the following specifications:

Signal Range: The existing technological solutions have a variety of signal ranges, to this particular project it was decided to use a technology that can provide 100 m of signal range. For projection security purposes the range of the technology was reduced from 100 meters to 75 meters, with this limitation its introduced a safety margin of 25 meters. With this safety margin it is possible to counter the effects from mobile objects that can temporarily jam the signal affecting the range(i.e. vehicles).

Environmental Effect: One of the factors that affects the signals propagation in urban environments is the buildings construction itself. In the FCT campus has a well know example, the Electrical Engineering building (Ed.10). This building isolates a variety of radio signals from and to the building and this is due to the building construction materials and its construction process. Taking this effect into account it's difficult to know with precision if the signal from one node could reach another node when having this building as an obstacle, even if its under the signal range. The plan for the deployment of the nodes through out the campus will take under consideration the previous scenario and the nodes signal range will not be defined by the maximum range but by the node field of view.

Summarizing the previous plan, the signal range of a WSN base station will be 75 meters limited to the node field of view. With these parameters the nodes will be projected to be deployed in the buildings exterior walls and only one node per wall. To achieve maximum coverage of the campus base stations can be deployed in security booths or in the illumination street lamps of the campus, in this case the base stations will have a field of view of 360 degrees instead of the 180 degrees when deployed in walls.
With the defined project specifications it's possible to obtain the full coverage of the university campus by deploying 58 base stations as shown in the Figure 1.2. This deployment create a wide WSN through the campus without creating any wireless dead-zones. In this

case the deployment locations were randomly selected by the author to accomplish the required specifications. In a real deployment scenario mathematical models and simulations should be run to achieve maximum performance and deployment optimization.

Figure 1.2: Nodes distribution through the campus

For a better visualization and comprehension, the campus map was divided in three horizontal sections. In each section is shown the signal coverage of each one of the base stations deployed through the campus.
A base station is one of the WSN player, it's a device that can be connected directly to the information grid through Ethernet, GPRS, WIFI, or other. This devices allow the various nodes deployed within it's range to send the information gathered and store it outside fo

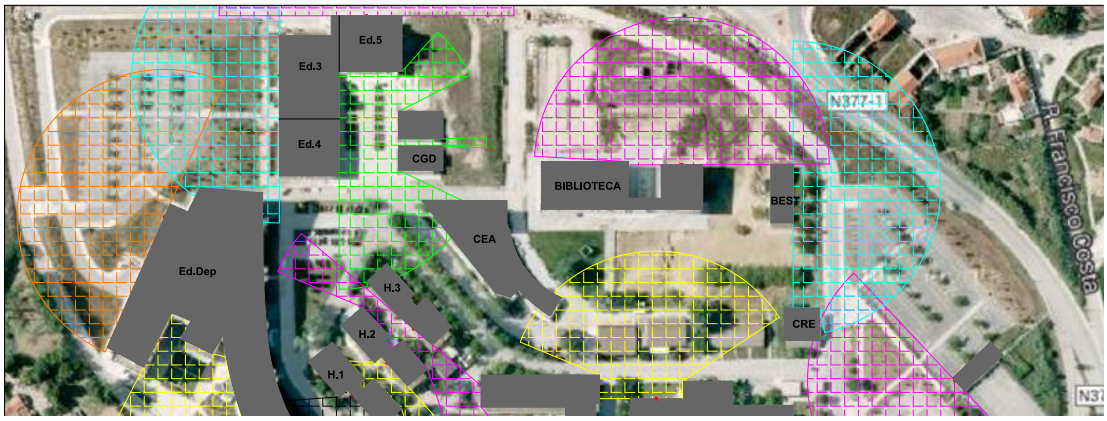


Figure 1.3: North Sector of the Campus

the WSN, this storage could be a simple text file, a database or a cloud.
Figures 1.3 to 1.5 show the signal range of the base stations deployed in the north division of the campus, this is the division with the highest number of base stations. This happens only because the of the irregular design of the building on this division, like the small alleys in the Building H.1 to H.3 and like the curved walls of the Ed.Dep and CEA.
Yet in this division is possible to see the use of independent base stations in the Figure 1.4 used in a security booth (red circle) and in a street lamp (dark blue circle).

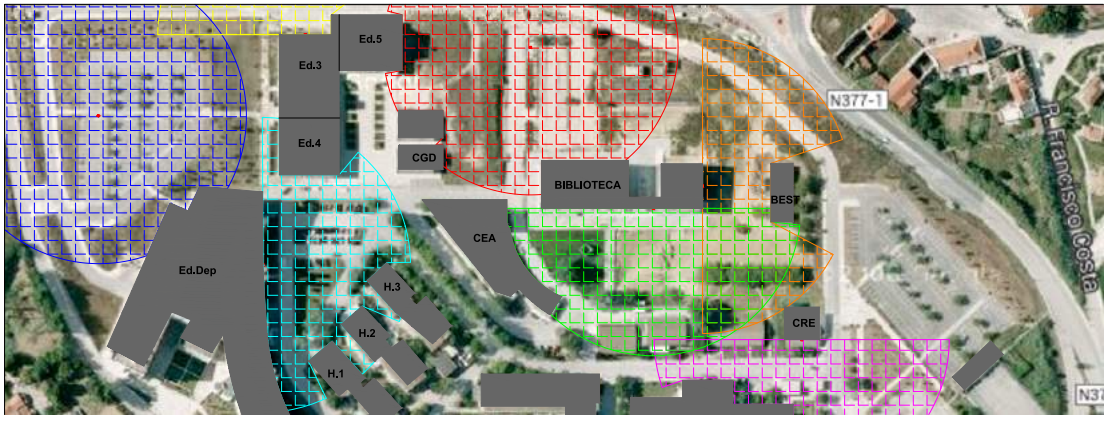


Figure 1.4: North division of the Campus

One of the applications of the WSN in the campus could be monitor the parking lots occupation. The campus has 6 external parking lots and some the parking lots inside the campus, and the information provided from nodes deployed in these it could possible to prevent overcrowded parking lots and illegal parking aside from the possibility to provide in real time information regarding vacant slots in each parking lot.
As it's possible to see in the Figures 1.3 to 1.5, there are 2 major parking lots in the north sector, to reach the parking lot on the right side of the image it's required to pass through the park on the left side of the image and drive all the way to the parking lot on the right. Assuming that a driver wants to park his car on the parking lot on the right and this one is

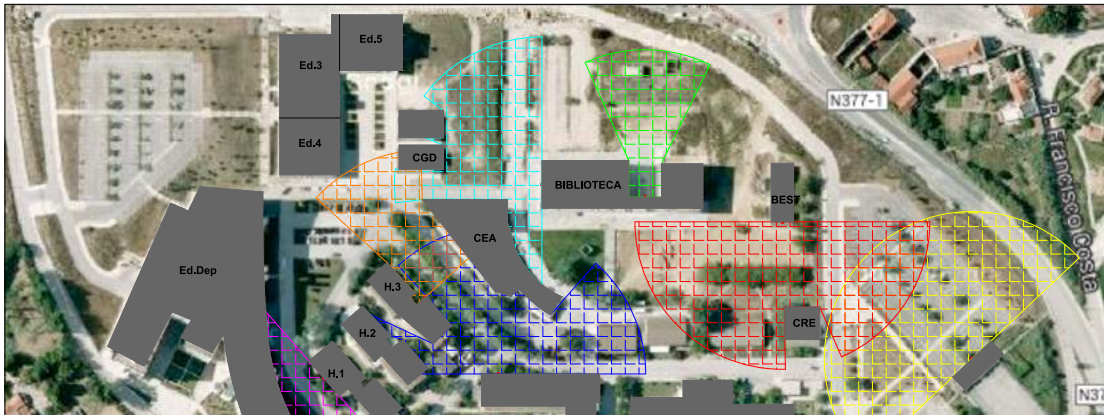


Figure 1.5: North division of the Campus

full, the driver makes a unnecessary drive and wastes time looking for a slot that doesn't exist. This was an easily preventable situation if the information off the parking lot was available for consultation.

In the Figures 1.6 to 1.8 is possible to see the central division of the campus and see that it holds some parking lots but its possible to see that it holds a higher number of gardens. In almost every garden it's possible to see the sprinkler system working in rainy days and this happens because some of this systems are operated by a timer clock and others are operated manually. This represents a waste of water and therefore a waste of money, and a WSN could help prevent this situation.

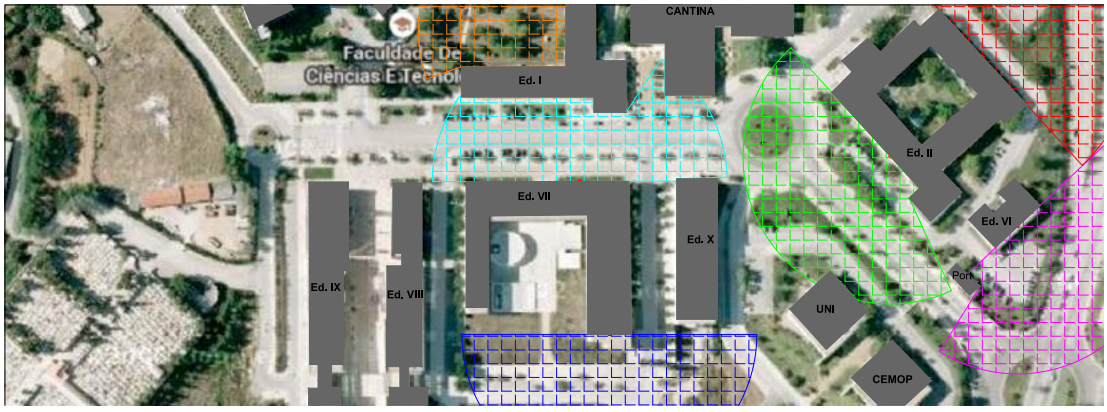


Figure 1.6: Central division of the Campus

A simple weather station in the campus area could acquire enough data about the rainfall providing it to a external storage, and soil moisture sensor spread across the campus gardens could provide enough information to have an autonomous system taking care of the campus gardens. This could be possible by having an external agents to interpret merge the data acquired by the previous sensors and calculating the right amount of water needed to the sprinklers systems in each garden and the time needed to complete the task.

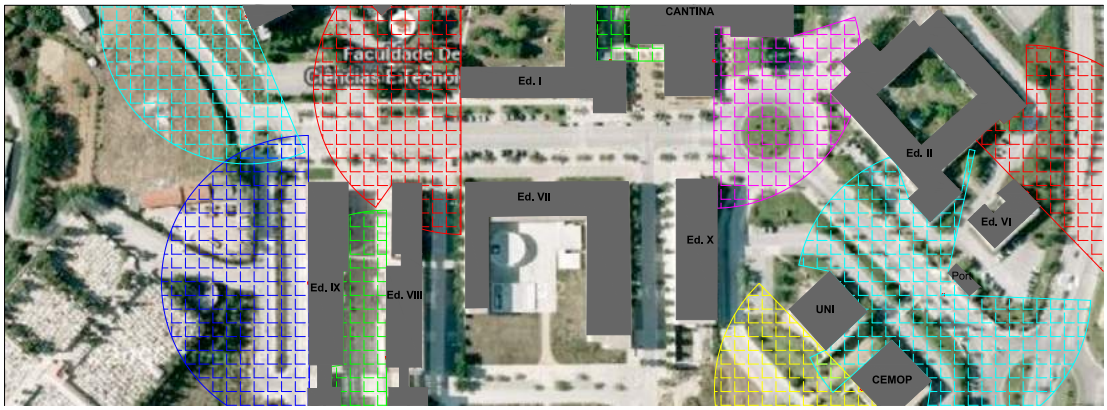


Figure 1.7: Central division of the Campus

In the Figure 1.7 its possible to see another independent base station (light blue circle) deployed in a security booth allowing the deployment of nodes through the surrounding gardens. But gardens aren't the only situation that could benefit from a WSN, a little bit of imagination and a lot of situations could emerge.

Every morning it's possible to see university employers waiting inline inside their cars to get inside the campus all because the people in the car in front of them don't know where they have their NFC card. Replacing a NFC card to a RFID chip could reduce this waiting time significantly, since the RFID is in the car and don't need contact to authenticate the employer.

Figure 1.8: Central division of the Campus

In the Figures 1.9 and 1.10 is represented the south division of the campus, and its possible to see another independent base station Figures 1.9 attached to a street lamp (yellow circle). Another situation that could be a great improvement to the campus people was a better regulation of street lamps, correcting this flaw in the system it could be much more helpful to everyone, the administration that could save money, the campus users that could have balanced street light and the environment that could have a reduce of

electricity consumption. Similar to the gardens waste of water there's waste of energy in the street lamps, because the Street illumination is timer activated, having nodes deployed through the campus to acquire information relative to the luminosity could bring some improvement to the street illumination system.

Figure 1.9: South division of the Campus

Some of the base stations deployed in this division could be used to monitor sportsman's health like the dark blue circle in the Figure 1.9 and the green circle in the Figure 1.10. If sportsman's use wearable technology like Blood Pressure Monitors the information acquired by this devices could be a great step to prevent and allow a quick intervention in potential risk situations to the sportsman.

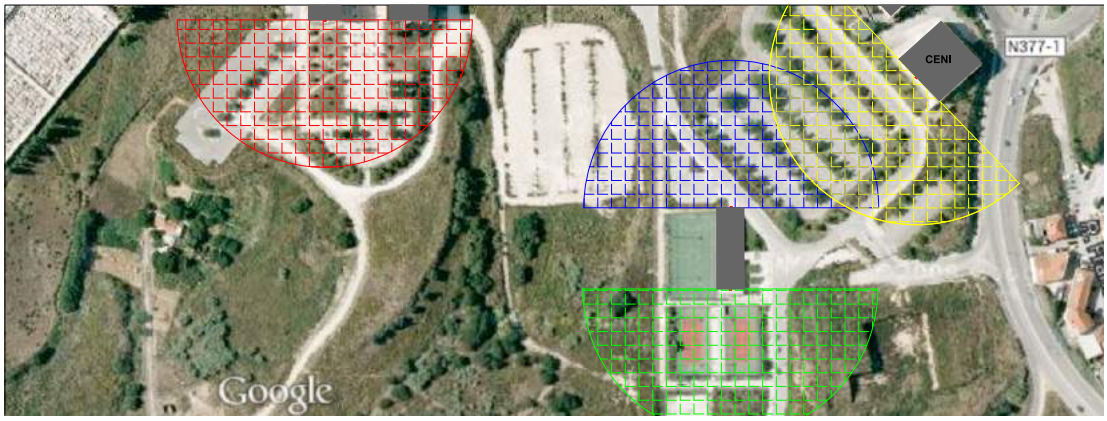


Figure 1.10: South division of the Campus

1.2 Generic Application: Low Power WSN

In the previous points some of the WSN applications were described, applications like sportsman care, parking occupation, gardens irrigation and street lighting but these aren't the only applications of the WSN. The number of applications is huge and it can vary from Traffic Congestion, Waste Management, Structural Health, Forest Fire Detection, Fleet Tracking, Environmental Parameters (urban noise, electromagnetic field, radiation, air pollution, ultraviolet radiation,...).

Taking the previous University Campus as a model, it's possible to generalize the WSN deployments. The WSN can be deployed virtually almost everywhere, whether it's an urban or rural environment.

In an WSN environment there are some network topologies that can be applied, the most common topologies are Star, Tree, Mesh as shown in Figure 1.11.

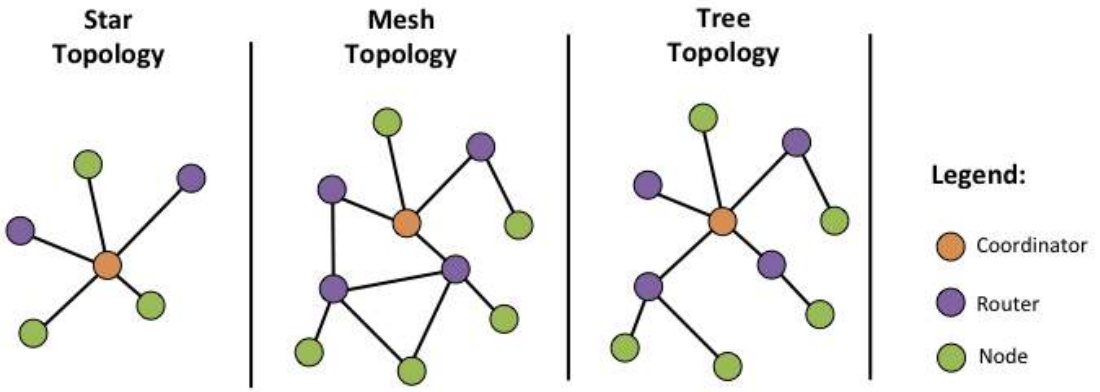


Figure 1.11: Network Topology

The Tree and Mesh topology rely on multi-op transmission, this means that one device transmits a packet to another device and the device is responsible to transmits the packet to another device or to the gateway device. This scenario can increase the network deployment range and reliability. In the other hand this could mean an overall increase of network power consumption, since one device could be responsible for the transmission of its own packets and other devices packets.

In the last topology, Star topology, each device is only responsible for it's own packet transmission to a gateway device. This topology doesn't provide the same network reliability and range increase as the others topologies but it can provide a smaller power consumption footprint.

1.3 Research Question

It is possible to relate all the previous issues just by adding a little extrapolation, a device with an higher signal range can have a higher power consumption. In the same line of thinking if the device have a higher signal range this signal could be more susceptible to signal interference and therefore it's more likely to have a higher power consumption. With a high power consumption it could be expect a lower battery duration and this could be ineffective if a device it's deployed in a low accessibility area. Having this much issues to start a WSN this can lead to a research question, witch supports this master thesis work:

How to choose the appropriate Wireless Technology for a WSN application?

1.4 Work Methodology

The work methodology used in this thesis is based on the principles of Scientific Method (Carey 2011) and (Schafersman 1997). This methodology approach is illustrated in the figure 1.12 is composed by the following steps:

1. Problem Characterization;
2. Background Research;
3. Formulate Hypothesis;
4. Setup an Experiment;
5. Test Hypothesis;
6. Hypothesis Validation;
7. Publish Results.

1. Problem Characterization: The main objective in this step is the identification and characterization of a significant problems. This study towards the problem can lead to a formulation of a research question that will be the basis of this research work. The problem identified in this thesis is how to choose the appropriate Wireless Technology for a WSN application.

2. Background Research: This step demonstrates the study of similar work on the subject that could bring insight on the problem characterization and the picking of scientific information about the existing work about this subject. Information about methodologies to choose an appropriate Wireless technology regarding the application.

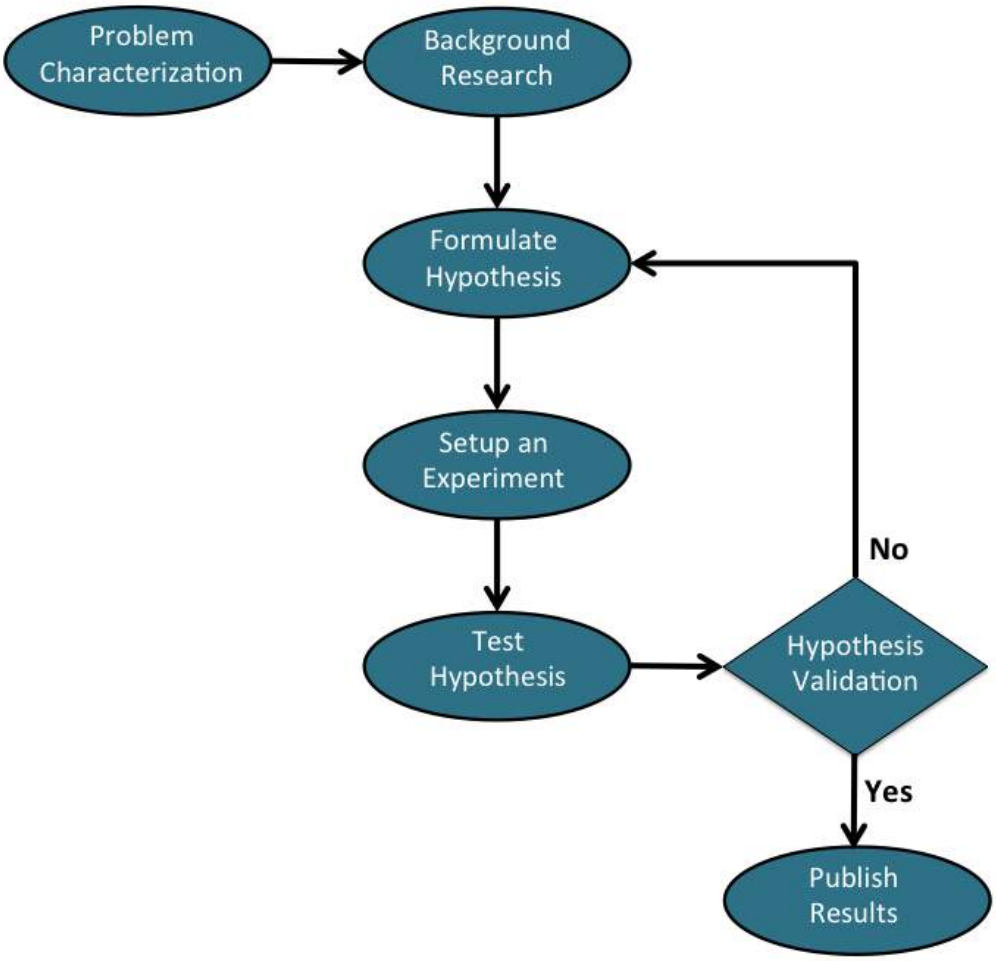


Figure 1.12: Work methodology used in this thesis

3. Formulate Hypothesis: Based on the background research it's possible to establish conceptual achievement, the hypothesis, in order to allow the elaboration of an experiment. It is required that the hypothesis clarify, explain and specify a solution that answers the problem characteristics. In this work the hypothesis consists....

4. Setup an Experiment: Since the previous hypothesis needs to be validated this step consists in the technological realisation of it through the definition of the experiment specifications. This experiment is a proof of concept and its meant to be replicated by others in a feasible way.

5. Test Hypothesis: The tests which the implementation of the hypothesis will be submitted to are defined in this topic. For each test, the data should be collected for further qualitative and quantitative data analysis. All the tests must be executed in a controlled environment in order to control all the results of the experiment and ensures that these testing can be reproduced.

6. **Hypothesis Validation:** With this step is possible to observe the results of the testing hypothesis and assess the quality of the proposed solution. In some cases the hypothesis results can lead to a weakening of the hypothesis strength, or even put in jeopardy all of the assumptions made in the very beginning of the research. This should not be interpreted as a failure, but as a way to improve the original approach and try another one with new expertise of the subject. If the hypothesis didn't pass in the hypothesis validation the researcher should retake the third step leading to a new iteration of the scientific method.

7. **Publish Results:** After the positive results, is possible to consider the future and define the recommendations for further research. Discussion regarding literature, research objectives and questions should be taken into account, and draw conclusions out of it. The outcome of the research should result in a contribution to the scientific community assuming the form of a final report and/or scientific papers. In this particular case the the work will be published as a final report, this particular document.

1.5 Dissertation Outline

This dissertation is composed by five, being the first chapter the present one. In this chapter was presented the motivation scenario,followed by the problem presentation and characterization witch lead to the research question "How to choose the appropriate Wireless Technology to a WSN". Afterwards, its presented the approach used to do this work, which was based on the scientific method. It is the author intention to produce all the chapters in a stand-alone mode to expedite this thesis analysis. The following chapters will be:

State-of-the-Art: This chapter presents a background research related to the studied work elements. This research focuses in the identification of important key-point that could provide useful information regarding the WSN deployment and informations about parameters that could influence the behaviour of wireless technologies. At the end of the chapter in the advancement section is presented a brief explanation of the exploited characteristics.

Methodology for wireless technologies automated tests: This third chapter proposes a methodology to test wireless technologies in a automated way. At first, it is presented the concept behind the proposed methodology, followed by the presentation and definition of the used interaction diagrams, behavioural diagrams and a detailed architecture of the proposed methodology.

Testing and Validation: The fourth chapter presents the tests used to validate the formulated hypothesis. In the beginning of the chapter is described the methodology adopted to test the hypothesis, followed by the description of the implemented proof

of concept and the tests definition and execution is made. The results of the tests are then presented and through its analysis is verified if the initial objectives were achieved.

Conclusions and Future Work: The fifth and final chapter presents a summary of this dissertation, highlighting the most important aspects of the developed work. It also takes in concern the results obtained with the realization of the testing process, and is suggested a direction for future research.

CHAPTER 2

LITERATURE REVIEW

In the previous chapter the motivation scenario was presented, the deployment of a wsn in the FCT-UNL campus. In the planning stage of the WSN a problem was unveiled and this problem lead to a research question "How to choose the appropriate Wireless Technology for a WSN application?". The previous chapter also explain the scientific method used and the composition of this thesis one way to help us solve our problems is through observation, in this particular case to see how others solved problems related with ours, being this the main objective of this chapter. This research was done in order to help the development of this thesis by discovering similar systems, technologies and approaches that can handle the problem of the appropriate wireless technology for each scenario. The articles chosen articles handle topics about deployment strategies, applications, wireless technologies and are presented now in alphabetic order:

- **"Choosing a Wireless Implementation Strategy and Applications (Gupta 2007):** This white paper addresses the deployment of wireless solutions in an industrial environment and the key aspects that need to be taking in consideration when planning and deploying a wireless strategies.
- **"The Fundamentals of ShortRange Wireless Technology (Frenzel 2012)":** Although it is a web-page it is a quite extensive and comprehensive one. It contains information related to the major wireless technologies and explain in detail some of the critical factors that can have impact in the design phase. This document also contains a detailed check-list of aspects that need to be addressed when selecting a wireless solution.
- **"Using Wireless Technologies for Healthcare Monitoring at Home: a Survey (Zatout 2012)":** The main subject addressed in this paper is the application of Wireless Technologies in the Healthcare environment the requirements that needs to be met and a

wide range of wireless technologies that can be chosen from.

- **"Wireless connectivity for the Internet of Things (Reiter 2014)":** This white-paper covers some of the main characteristics of the wireless technologies mainly when applied to the IoT, such as frequencies used and their regulations, the communications protocols, the network range, the network topology and size and the standard/proprietary protocols used.

2.1 Individual Review

2.1.1 Choosing a Wireless Implementation Strategy and Applications

This white paper written by Gupta 2007 describes a strategy to deploy wireless solutions in an industrial environment and some key considerations when doing so. An wide industrial wireless deployment can be somewhat expensive, with so, it is important to avoid money waste. To comply with that it is important to analyse business needs and applications such as required data throughput, range, QoS, etc.

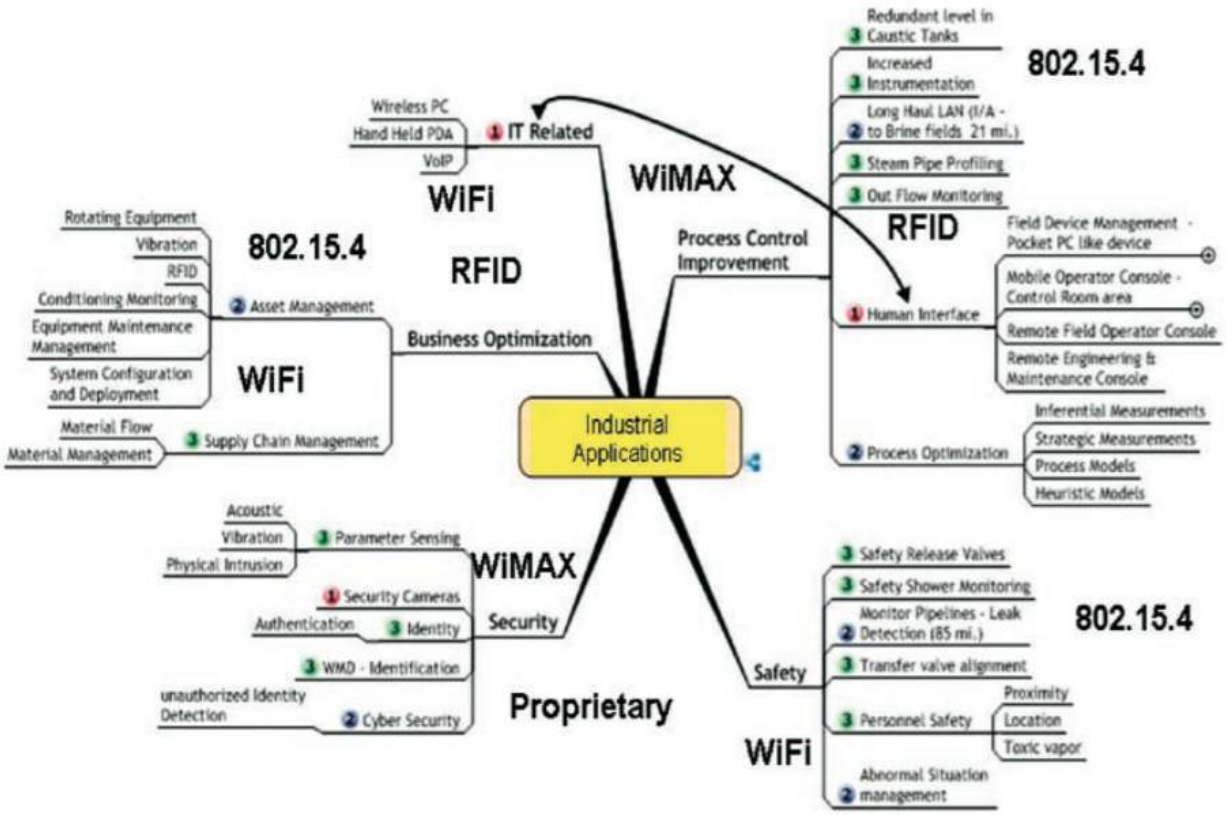


Figure 2.1: Unified Wireless Comunications, Gupta 2007

It is also necessary to plan future needs of the wireless solutions so the wireless infrastructure does not become obsolete if more devices need to be added or if higher data throughputs are required. In an industrial environment it is possible to encounter multiple sources information that can/will be transmitted over wireless networks. Different sources have different need and characteristics, with this diversity of requirements multiple communication protocols must be used since the current radio's can not meet all the requirements. Having multiple standards operating in the same environment creates a large ecosystem and it is important that all wireless communications become centralized and then distribute the acquired data to the appropriated persons/departments as described in the figure 2.1. Industrial environments can be more likely to have more security needs, and wireless networks need to be enforced as well.

Key-Points
The approach taken in this article provided strong foundations to apply in the planning stage of a wide wireless deployment, these foundations were presented in the document as key-points being the following:

- "Identify user requirements";
- "Select and purchase hardware and software that is proven, scalable and capable of understanding diverse protocols";
- "Plan in detail for network migration or expansion, since networks are now the lifeline of the company";
- "Utilize the services of consultants so that you get quality advice from people who have gone through this experience";
- "Conduct an RF site survey, prior to implementation, to identify wireless signal paths and sources of potential interference";
- "Build ongoing maintenance, support and optimization services into the plan";
- "Strive to integrate wireless security and mobility products with existing enterprise identity management solutions";
- "Determine which classes of devices are allowed to touch the network and the policies that are involved";
- "Identify a wireless platform that can grow with changing networks and devices";
- "Storyboard the wireless application, verify user requirements and only then start development";

Analysis

This article can provide an overall insight over one approach to deploy a WSN. Despite the usefulness of some provided key-points they do not explicitly indicate the parameters that need to be addressed, requiring some text interpretation to identify them. There are also some key-points that only have purpose in a enterprise environment such as the use of consultants, the creation of maintenance and support services, the network migration and the identification of devices that are allowed to use the network.

Key-points such as the identification of user requirements and the selection of hardware and software can be two relevant points to the WSN deployment. Although this two key-points can provide a simple way to narrow the list of equipments that can be chosen to the network, this key-points cannot be used in this document because they cannot fit in any of the chosen parameters.
The RF site survey is the only key-point that can be somewhat linked to one of the existing parameters, the device environment. The existence of potential sources of interference can have different effects over the wireless technology itself.

2.1.2 The Fundamentals of ShortRange Wireless Technology

The approaches found in the article written by Frenzel 2012 explain some of the popular existing wireless technologies, their characteristics and some of their typical applications. It is also explained the factors that affect the performance of a wireless network, being these mainly physical factors as described in the figure 2.2. Those critical factors can be range, transmit power, antenna gains, frequency used, and receiver sensitivity.

When planning a wireless deployment its important to take under advisement that with the same conditions devices with lower frequencies will have an higher range that devices with higher frequencies. It is also important to choose the right type of antenna since there are some different kinds of antennas and they can influence the range and the transmission/reception power. As stated before data rates can influence the both range and reliability of the signal, this can be accomplished by using lower data rates.

The surrounding environment must be considered they can introduce interference that can affect the wireless signal. Some of the sources that can affect the signal can be obstacles, walls, trees, electrical and electronic devices. It is also important to understand the application where the wireless devices will be uses, since there are countless deployment applications and all of them have different requirements and will use a wireless network in different ways.

POPULAR SHORT-RANGE WIRELESS TECHNOLOGIES				
Technology or standard	Frequency	Range	Features	Common applications
ANT+	2.4 GHz	< 10 m	Low power	Health, Sports monitoring
Bluetooth	2.4 GHz	< 10 m, up to 100 m with higher power	Low-power version Available	Wireless headsets, audio apps
Cellular	Common cellular band	Several km	Long range	M2M
IEEE 802.15.4	2.4 GHz	< 10 m	Multiple protocols available	Wireless networks
IEEE 802.22	470 to768MHz	Many miles	Designed for white Spaces, cognitive radio	Broadband, backhaul, not yet used
ISA 100a	2.4 GHz	< 10 m	Extra security and reliability	Industrial monitoring and control
Infrared (IrDA)	800 to 1000 µm	< 1 m	Security, high speed	Remote control, data transfer
ISM band	Part 15 frequencies	< 10 m	Low cost, simplicity	Monitoring and control
NFC	13.56 MHz	< 30 cm	Security	Payment, inventory
RFID	125 kHz 13.56 MHz 902 to 928 MHz	< 1 m	Low cost	Tracking, inventory, access
6LoWPAN	2.4 GHz	< 10 m	Internet access	Monitor and control Via Internet
UWB	3.1 to 10.6 GHz	< 10 m	Low power, High-speed data	Video transfer
Wi-Fi	2.4 GHz and 5GHz	< 100 m	High-speed, ubiquity	Local networks, Internet access, broadband
Wireless HART	2.4 GHz	< 10 m	HART protocol	Industrial monitoring and control
WirelessHD	60 GHz	< 10 m	Very high speed	Video transfer
WirelessUSB	2.4 GHz	< 10 m	Proprietary protocol	HID
ZigBee	2.4 GHz	< 10 m	Mesh networks	Home, industry monitoring and control
Z-Wave	908.42 MHz	< 30 m	Simple protocol	Home monitoring and control

Figure 2.2: Popular Short-Range Wireless Technologies, Frenzel 2012

Key-Points

This article has the majority of the key-points explicitly identified, other than that some of the key-points presented in the article were not directly copied to this list because they lack in some specificity. The summarized key-points will be:

- "Range";
- "Simplex/Duplex";
- "Number of nodes";
- "Data rate";
- "Potential interference";
- "Environment";

- "Power source";
- "Size and space";
- "Regulatory issues";
- "Licensing fees";
- "User type and experience";
- "Security";

Analysis

With the information existing in this article it is possible to extract some knowledge regarding the wireless technologies, since the majority of the listed key-points represent important parameters to take in account in a WSN context. Despite of the usefulness of the listed key-points some of this will not be pursued, such as "Number of nodes", "Power source","Size and space","Regulatory issues","Licensing fees","User type and experience" and "Security" since they can not be related with this work.

The key-points "Potential interference", "Environment" and "Range" key-points can be bundled into one of the previously identified parameters. That parameter is the device environment since this parameter will be used to define all the conditions that can affect and be affected by environmental conditions. Other key-points can be related in parameters useful for this thesis, key-points such as the "Simplex/Duplex" and "data rate" can be respectively linked to the Communications Type parameter and to the Payload parameter.

2.1.3 Using Wireless Technologies for Healthcare Monitoring at Home: a Survey

The article written by Zatout 2012 is mainly focused in the healthcare applications and their usual requirements. Although this study focuses an indoor environment such as home and/or healthcare facilities, it presents an analysis on the choosing method for the appropriate wireless technology for this particular application taking in advisement the presented requirements.

This study presents a summarized table, figure 2.3, explaining with some degree of detail some of the existing wireless technologies their typical applications and their characteristics, presenting some advantages and drawbacks for each technology. This study also includes information and explanation regarding critical factors that may affect the performance of a wireless network, some generic foundations that can be used in the

	WPAN Technologies			WLAN				Proprietary
	Zigbee	Bluetooth	UWB	Wi-Fi				standards
Standard	IEEE 802.15.4	IEEE 802.15.1	IEEE 802.15.3a	IEEE 802.11b	IEEE 802.11g	IEEE 802.11a	IEEE 802.11n	Proprietary
Industrial organizations	Zigbee Alliance	Bluetooth SIG	UWB Forum and WiMedia™ Alliance	Wi-Fi Alliance				N/A
RF Frequency	2.4 Ghz, 868/ 915 MHz	2.4 Ghz (79 channels - 1,6 MHZ)	3.1-10.6 Ghz	2.4 Ghz-5.8 Ghz				433/868/900 MHz, 2.4 GHz
Speed	250 Kbits/s at 2,4 GHz (World) 40 Kbits/s at 915 Mhz (USA) 20 Kbits/s at 868 Mhz (Europe)	1 Mbps - v1.2 3 Mbps - v2.0+EDR	110 Mbps- 480 Mbps (see up to 1.6 Gbps in some applications)	11 Mbps	54 Mbps	54 Mbps	200 Mbps (540 in some applications	10–250 Kbps
Maximum range	10-100 m (see up to 300 m)	10-100 m	3-10 m	10-100m				10–70m
Energy, PE: mw/Mbps	30 mw (1000 mw/Mbps)	100 mw (100mw/Mbps)	400 mw for 200 Mbps (2mw/Mbps)	750 mw (68mw/Mbps)	1000 mw (19mw/Mbps)	1500 mw (27mw/Mbps)	2000 mw (10mw/Mbps)	Very low-low
Battery (TTGB)	3.1 days	2.2 hours	40 sec	1.2 min	2.5 min	2.5 min	40 sec	
Cost	2 $	3 $	7 $	5 $	9 $	12 $	20 $	
Nodes (per Network)	65000 (20 Kbps), 255 (250 Kbps)	8 (7 slave+1 master), 10 piconets	128	32				100-1000
Modulation/ Spread Spectrum	DSSS, BPSK(868/915 MHz), O-QPSK (2,4 GHz)	FHSS, GFSK (It defines three power classes)	OFDM or DS-UWB	Four ≠ Physical Layers: FHSS, IR, OFDM, DSSS				
Main applications	Wireless Sensor Networks, Remote control, home automation Medical assistance	Audio applications, Replaces wires on desktops	Multimedia Applications, industry Health applications	Home Automation Networks Replaces the Ethernet cables				
Advantages	Very low power consumption MAC/PHY layers strong and effective Reliability (CSMA, security ...) Reduced cost Low latrency regarding to Bluetooth Can support several nodes	Widespread Average Consumption It costs 3 times cheaper and 5 times less in power than Wi-Fi	Low cost, low power consumption High data rate Low radiated energy Precise localisation Support the Qos in WSNs Less interference Signals can cross obtacles (doors, etc.) Low delay regarding to Bluetooth Resistance to multi-path networks	High data rate Widespread Wide-ranging radio Security, guarantee of QoS (IEEE 802,11e) Products widely known Ease of deployment, and low cost Keep the same infrastructure				
Drawbacks	Specification that still moving Minimized range at 2,4 GHz (10m)	It is not suitable for WSN Speed limited, important delay regarding to Zigbee Interference with Wi-Fi Protocol rather complex The effect of radiation on health	Technology not yet ratified Lack of UWB products in the market. Limited range (10 m) regarding to Wi-Fi High synchronization constraint	High energy consumption Latency potentially important Multiple variants Limited use for WSN				

Figure 2.3: Comparison Sheet of Wireless Technologies, Zatout 2012

planning stage of a WSN.

Key-Points

After a brief analysis of this article it was possible to identify some key-points that can be used in the planning phase of a WSN either it is an indoor or an outdoor environment. The key-points that can provide useful informations to this document are the following:

- "Quality of service";
- "Energy consumption";
- "Scalability";
- "Mobility";
- "Environment constraints";
- "Radio bandwidth";
- "Device power source";
- "Specific protocols";
- "Heterogeneity";

Analysis
Resembling other article, in the listed key-points can be found useful informations that can support the deployment of wireless technologies. From the previously listed key-points the following will not be explored for obvious reasons, energy consumption", quality of service, scalability", mobility", radio bandwidth, device power source and Heterogeneity.

The remaining key-points environment constraints and specific protocols can be related to the previously defined parameters, parameters such as device environment and communications protocol respectively.
Despite of the fact that the power level was not identified in the previous list of key-point, the author mentioned the power level during the explanation of a wireless protocol. With so, this will be the only reference of such parameter in the analysed article.

2.1.4 Wireless connectivity for the Internet of Things

This document written by Gil Reiter in Reiter 2014 focus in the a simple statement "One size does not fit all". With this statement the author explains that there are not wireless technologies that can incorporate the great majority of properties. Some of the most prevailing wireless technologies are based on interoperability and on international standards, those technologies will have their characteristics and foundations explained in the article.

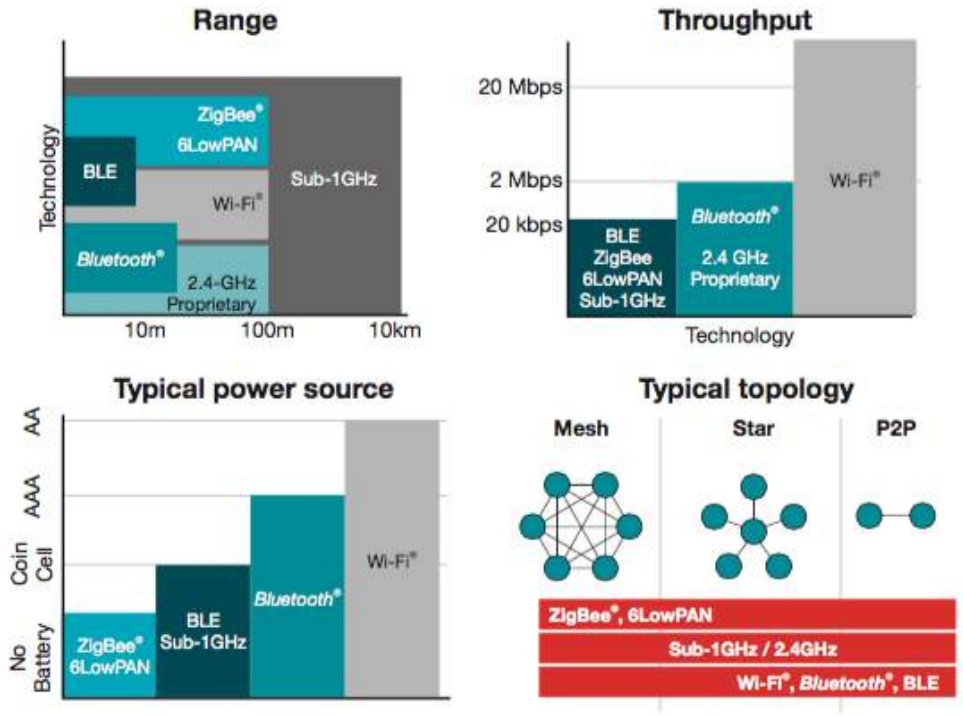


Figure 2.4: Summary of wireless technology parameters Reiter 2014

Characteristics such as frequencies are regulated by national and international authorities

and depending on the frequency used can be licensing cost associated. Different frequencies also have different properties and generally they represent a trade-off between range and data throughput.

The document also makes a brief introduction to communication protocols and the advantages of their use. One of those advantages is the usage of the IP stack and therefore the usage of IP addresses that allows devices to directly communicate with the internet instead of the non IP devices that only can achieve the internet using a internet gateway. These communication networks can use a variety of network topologies all of them with different characteristics being explained two of these topologies. A summary overview of wireless technologies parameters is also provided in this document in a figure form, figure 2.4, and can provide a helpful insight over the specifications of some technology standards, such as range, data throughput, typical power source, typical topology.

Key-Points
This article can provide useful information regarding the topics that need to be addressed when planning a wireless deployment and valuable insight over some of the most prevailing existing wireless standards and their characteristics. In order to acquire the essential parameters from this key-points it will be required some text interpretation. The key-points presented in the article are the following:

- "Frequency bands and worldwide regulations";
- "Communication protocols";
- "To IP or not to IP";
- "Network Range";
- "Network topology and size";
- "Standards and interoperability";
- "Radio transceivers and proprietary protocols";
- "Not only wireless";
- "Power source";
- "Data throughput";

Analysis
From the previous list of key-points obtained with the document analysis, there are some key-points that will not be pursued due to the fact that they do not relate to the previous established parameters. The key-points in question will be Frequency bands and worldwide regulations, Network Range, Network topology and size, Not only wireless and

Power source.

With the remaining key-points is possible to establish a link between the "communication protocols", "to IP or not to IP", "standards and interoperability" and "radio transceivers and proprietary protocols" key-points to the pre-defined Communication Protocol parameter. In addition to this key-points it is also possible to establish a relation between the "data throughput" key-point and the Payload parameter

2.2 Synthesis

Having already presented and described each of the state-of-the-art element, it is now important to summarize the knowledge from each document and relate it with the problem parameters presented before. The Table 2.1, presents the authors point of view in how each state-of-the-art element relates to the previously defined characteristics.

The Communication Protocol was referenced as an important key-point in almost every studied element, besides Gupta 2007. The existing information regarding communications protocols included the number of nodes allowed per protocol, the range between devices, important features, common applications.

The study element of Frenzel 2012 was the only study element that addressed the Communication Type as an important key-point for a WSN deployment. The author considered this because he believes that applications such monitoring or remote control only need one-way communication, in different directions, and the same does not apply to an application that would require both monitoring and remote control, a two-way communication.

One can notice that all of the studied elements have identified the Device Environment as one of the fundamental parameters to take into advisement in the planning stage of a wireless network. The majority of the elements provided factors that can affect the behaviour of the wireless signal, factors such as tree walls, electrical circuits and communication lines can affect the performance of a wireless signal

In none of the studied elements was made any reference to the Payload parameter, despite of this fact several references were made to the data-throughput and to data-rate by the Frenzel 2012 and Reiter 2014.

With respect to the radio Power Level only the elements written by Frenzel 2012 and Reiter 2014 gave some small emphasis to this parameter. Despite of this fact, in both elements the authors were referring to particular wireless cases giving a small amount of information regarding the power level in a Wi-Fi network and the maximum,normal and minimum

power level in others wireless protocols.

	Gupta, S. (2007)	Frenzel, L. (2012)	Zatout, Y. (2012)	Reiter, G. (2014)
Communication Protocol	-	Detailed information regarding some of the most popular	Brief explanation about the parameter, with some explanation about the importance of it	Brief explanation about the parameter, with some explanation about the importance of it
Communication Type	-	Give an brief explanation about it's use and importance	-	-
Device Environment	Suggests RF survey	Gives examples of interference sources and suggestion to minimize interference	Explains the parameter and its importance.	Mentioned during Protocols exposure
Payload	-	Mention Data-Rate	-	Mention Data-Throughput
Power Level	-	Mentioned during Protocols exposure	Mentioned during one Protocol exposure	Mentioned during Protocols exposure

Table 2.1: Literature Review synthesis table

2.3 Advancement

Taking the scientific method into account, is possible to understand that the creation of a solution for the research question should use the knowledge from past experiences and experiments as support (Schafersman, 1997). The usage of this approach allows a scientific and a technological evolution through the transmission of the previously acquired knowledge. Therefore, the used documents and their analysis can be used to bring some insight knowledge in the form of experiences and ideas that will be reflected in the proposed solution for this work.
In order to determine the documents contribution to the solution, the analysis of each document against the characteristics of the problem, summarised in Table 2.1, is taken into consideration resulting in the following conclusions.

After studying all State of the Art elements previously presented, it is possible to concluded that none of the presented elements can individually present a full methodology that could be followed when planning a wireless network deployment. Despite of that fact the sum of key-points presented in the state of the art elements provide a wider perspective of the task at hand.

The existing information regarding the Communication Protocol turned out to be a useful

source of information, with interesting technical data. The concept behind the Communication Protocol, namely the relation between the communication protocol itself and its influence in the overall behaviour of the wireless radio. The associated hypothesis will focus in the measurements of the power consumption of various Communications Protocols.

In relation to the Communication Type parameter lack of support in the studied elements, the concept behind this parameter cares for an intensive analysis regarding his effects over the power consumption. Therefore in the hypothesis will be measured the radio's power consumption in different communication situations without changing any other parameter.

Although the Device Environment parameter has some backing information in the studied elements, the existing practical information is still dim. Therefore the concept associated to this parameter lies in the analysis of the environment interference over the other existing parameters. To assess this parameter the hypothesis will focus the measurements of the radio power consumption in different environmental conditions and with different ranges between the radio devices.

The concept behind the Payload parameter consists in the analysis of the behaviour Regarding the Payload parameter the hypothesis will measure the amount of data exchange between radios and the power consumption associated to the communication without looking the the maximum allowed payload per message. To achieve this objective, the hypothesis will consists in the analysis of the amount of transmitted data against the received data.

With respect to the radio Power Level parameter the information in the studied elements is almost non existent, nevertheless the study of this parameters seems to be relevant. The concept behind this parameter consists in the study of the trade-off between the Power level and the range between devices thereafter the effect over the power consumption.

Despite of the fact that the Duty Cycle parameter was not addressed in any of the state of the art elements, despite of this fact, it is important to address this topic in a WSN context. In this case, the concept behind this parameter consists in the analysis of the effects of different duty cycle over other parameters. Therefore the hypothesis will be consists in analysis of the radio's behaviour and the power consumption of the wireless radio when sending wireless messages with different duty cycles.

CHAPTER 3

METHODOLOGY FOR WSN COMMUNICATION TECHNOLOGIES AUTOMATED FIELD TESTS

3.1 Problem

To achieve an approach that could promote an optimization of low-power IoT devices it is necessary to analyse, test and observe the effects of some parameters. In the state of the art chapter, documents were studied and analysed in order to retrieve the important features to get a good WSN performance. The parameters obtained in the analysis were the communication protocol, the communication type mode, the duty cycle, the radio power level, message payload, and device location.

Parameters	Explanation
Communication Protocol	The choice of this parameter was due to the existence of a great number of wireless protocols and devices ;
Communication Type	This parameter was included in order to analyse the impact of the type of communication in any other parameters;
Device Environment	Since the wireless signal can be degraded in a great number of ways it is important to analyse the impact of this parameter over the other important parameters, being also possible to analyse the device behaviour when testing it in different deployment environments;
Duty Cycle	The analysis of the different duty cycle effects over other parameters can provide information regarding the wireless device behaviour;
Payload	The existence of this parameter is due to the fact that an behaviour analysis of the wireless device can represent an asset when analysing the device performance when transmitting messages with a specific payload;
Power Level	To achieve an optimal WSN deployment it can be necessary the analysis of this parameter against the power consumption and the signal range;

Table 3.1: Summary of Parameters

The necessity to analyse and test the parameters previously enumerated brought one concern about the test itself, the concern about the difficulty to model some of the parameters. The device environment parameter is one of the parameters that makes very difficult to generate an accurate testing model. This happens because the WSN surrounding environment can unintentionally introduce a wide variety of interferences to the system and therefore, this traduces an increase difficulty to generate an accurate simulation model.

In this particular case, the real-world tests can be more accurate than simulations because the generated results take into account all of those interferences, since this type of interferences cannot be modelled and therefore reproduced in a simulation environment. Hence this fact in this thesis will be used in exclusive real-world tests to produce results and conclusions. In order to completely understand the influence of any parameter in the WSN overview it will require the variation of every parameter. This parameters variation can show the complexity involved in a real-world test, by showing the number of iterations required to test every scenario as it can be seen in the table 3.2.

Duty Cycle	Payload	Communication Type	Power Level	Device Environment
0	1	Tx-Rx	P1	Indoor
100	10	Rx-Tx	P2	Outdoor
1000	25	Request-Reply	P3	LoS
10000	50	-	P4	NLoS
30000	100	-	P5	-
60000	-	-	-	-

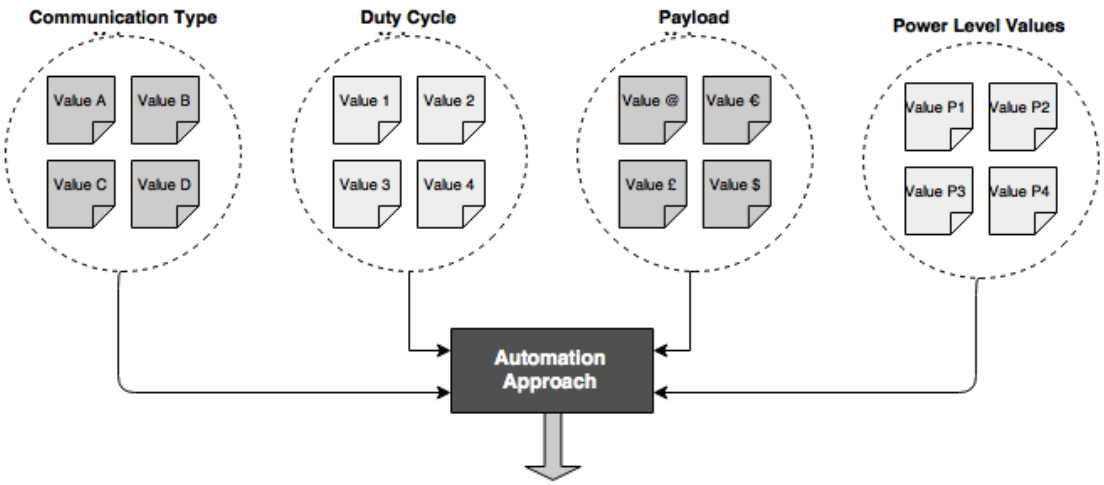
Table 3.2: Summary of Testing Characteristics

To produce the previously showed test for a single radio module covering all the presented parameters with some degree of accuracy, each test iteration would have to transmit/receive 10 messages. This would represent 450 iterations for a single location, making a total of 1800 iterations. It's important to state that the Power Level items (P1,P2,...) are generic indicators of the power levels existing in the wireless devices.

The results produced in a single test just by changing the iterations manually could represent an endless and boring task. Taking all this factors into account the required time to run the previous scenario for a single radio device it would be required approximately 60 hours without taking into account the required device changes between iterations, such as device reprogramming, radio module configuration and devices relocation. In this presented case scenario it is preferable to automate the test iterations instead of testing all this parameters manually since it is not practical nor time efficient.

3.2 Concept

The employment of automation notions in real world tests can offer a faster and efficient way to test the predefined testing scenarios. Applying this notions to the WSN test environment can provide a substantial help to the user that want to perform such deployment, allowing the discover and analysis of one or more wireless technology towards a specific wireless scenario. An example of this type of scenario is depicted in the figure 3.1.
In this scenario the automated tests are used to analyse the parameters of wireless technologies and its behaviours on the device overall power consumption.



Testing Pool

#	Communication Type	Duty Cycle	Payload	Power Level
1	Value A	Value 1	Value @	Value P1
2	Value B	Value 1	Value @	Value P1
...	...	...	...	...
12	Value A	Value 1	Value @	Value P1
13	Value A	Value 1	Value @	Value P2
...	...	...	...	...
21	Value A	Value 1	Value @	Value P1
22	Value A	Value 2	Value @	Value P1
...	...	...	...	...

Figure 3.1: Concept of Automation Approach

The hypothesis proposed in this work is the usage of this automated test to measure the variation effects of the available parameters on the wireless device overall power consumption. The acquired information can be used to analyse and compare the wireless technologies against the parameters and, for instance, to infer which wireless technology has the appropriated characteristics and can be used in a specific environment and conditions.

3.2.1 Architecture

The Automation Methodology proposed in this work is illustrated in Figure 3.2. In order to accomplish this automation methodology is required three devices, a Coordinator device, a End device, and an Auxiliary device. In the following paragraphs each one of intervening devices of this automation methodology will be explained, as well as the relevant characteristics.

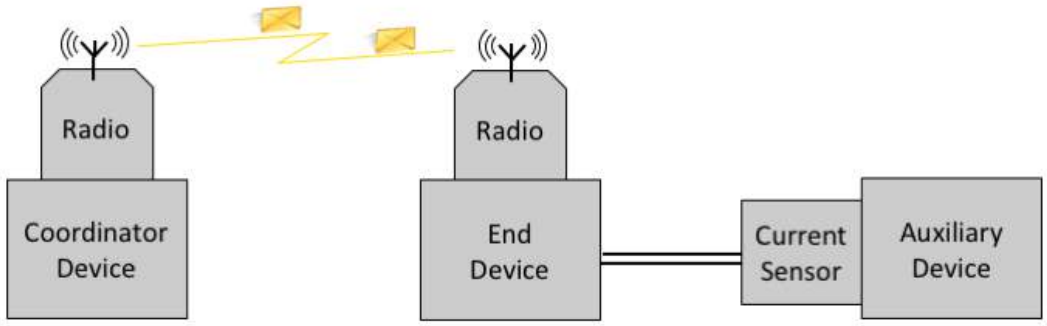


Figure 3.2: Achitecture Diagram

Coordinator Device
In this automation model, this device is of a paramount importance since it is responsible for establishing and maintaining the synchronism between devices. This device is also responsible for generation of configuration messages the distribution of Configuration Messages to the End Device.

End Device
The End Device represents the most important unit of this automation methodology representing the device that will be tested and analysed. This device is responsible for the reception of configuration messages and following interpretation. The configuration message interpretation as well as the extraction of the embedded parameters inside the message, being this also responsible for the adoption of the extracted parameters and for carrying out the test in the conditions defined by the configuration message.

Auxiliary Device
To complete this automation model, it is required one more device, the Auxiliary device. This device will be monitoring and analyse the End device power consumption during the trial field tests.

3.3 Activity Interaction

This section will provide the specifications for the various types of communication that will occur between this automation methodology. As stated earlier in this document the existence three communication types appear from the necessity to analyse the power consumption of the wireless radio in each of this situations. In the following sections it is going to be presented the interaction diagrams for each one of the Communication Type. The presented sequence diagrams are defined according to the UML format Bell 2004, in which, each architecture intervener is represented by a lifeline block being the communications between blocks represented by a closed end arrow.

3.3.1 Transmitter Diagram

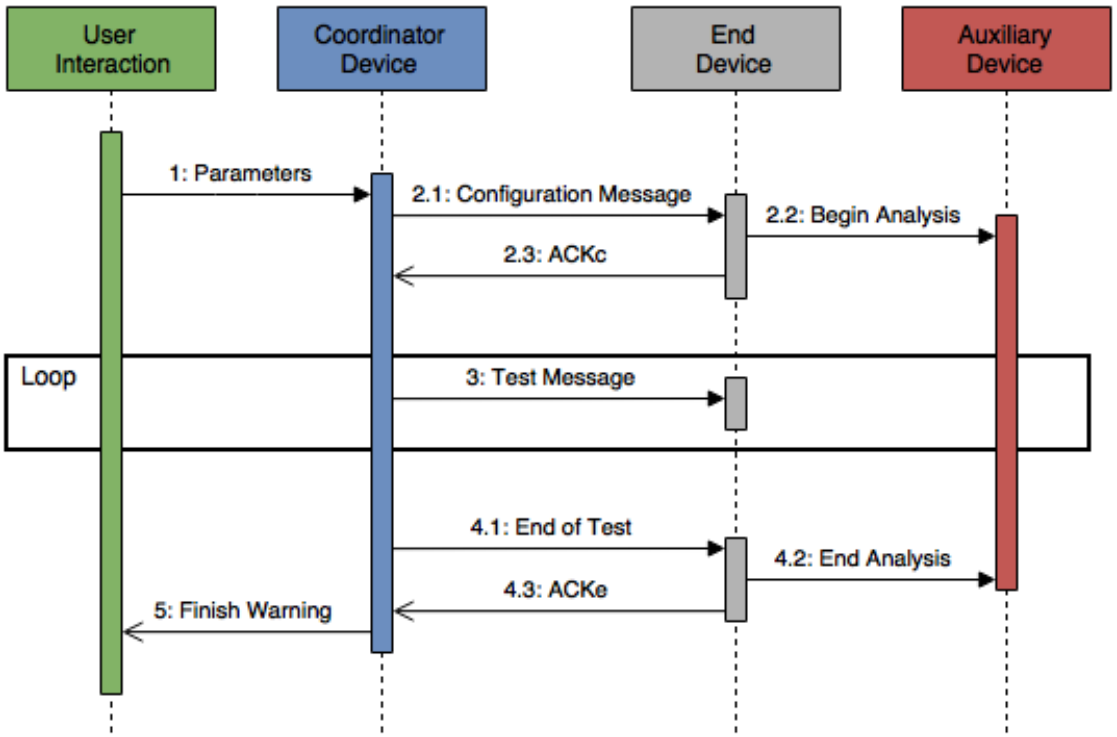


Figure 3.3: Transmitting sequence diagram

The diagram presented in this section, Figure 3.3, provides a detailed description of the existing communication between each one of the intervener. The testing process begins with the Configuration Message transmission from the Coordinator to the End device. With the reception of this message the End device is tasked to reply to the Coordinator accusing the Configuration message correct reception being also tasked with the transmission of an activation command to the Auxiliary device.

Achieving this point, the devices will start the real test assuming the Coordinator device the Transmitter Role (Tx) and the End device the Receiver Role(Rx). This test execution implies the exchange of a pre-defined number of test messages, being this number defined in the received Configuration Message. With the end of the testing loop, the Coordinator will transmit to the End device an message indicating the End of Test, being the End device oblige to reply to the a message to the Coordinator device finishing the one test process.

3.3.2 Request-Reply Diagrams

In the following diagram, Figure 3.4, one can notice the resemblance from this diagram with the previous one. This interaction diagram quits the resemblance only in the testing loop, being both devices obliged to perform both roles, the Transmitter and Receiver role.

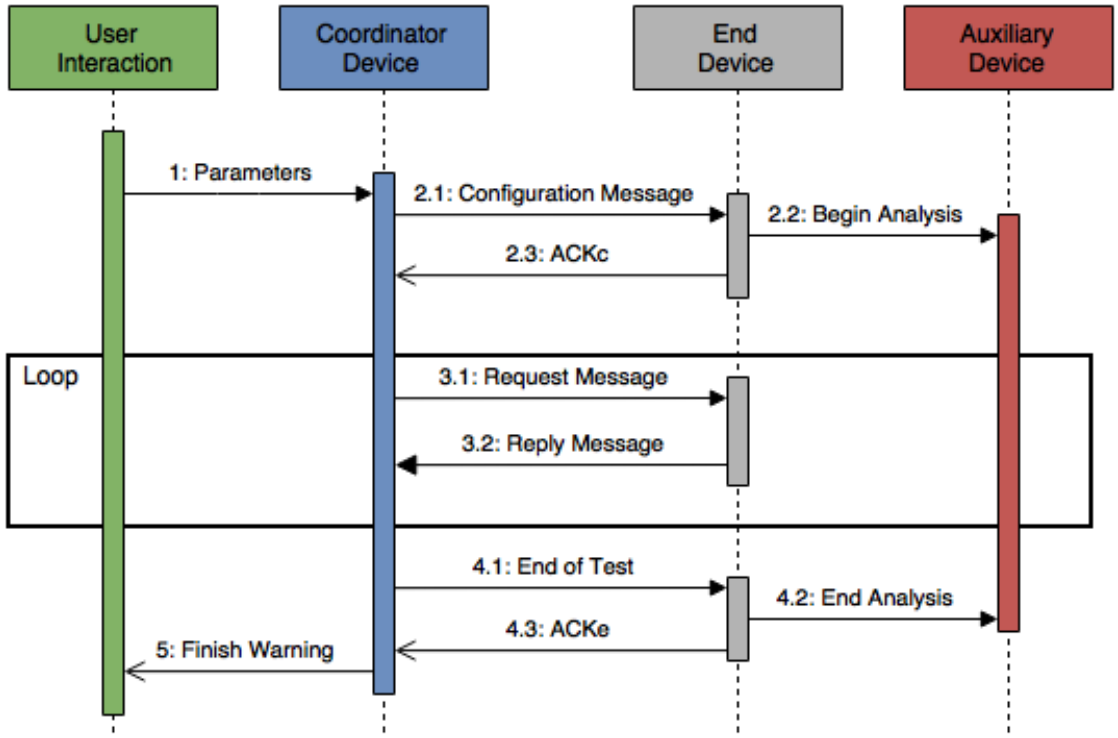


Figure 3.4: Request-Reply sequence diagram

To prevent the appearance of unwanted scenarios in this automation methodology such as synchronized transmission from both devices, the Coordinator device will begin the test transmissions and for the same reasons it will be also the Coordinator device the responsible for transmitting the End of Test message to the End device.
Following a test transmission from the Coordinator can exist a transmission from the End device, in this particular case the Coordinator exchange temporarily its role with the End device, being the coordinator tasked with the reception of test messages and the End device responsible for its transmission. Finishing this transmission from the End device, both devices restore their primarily role.

3.3.3 Receiving Diagrams

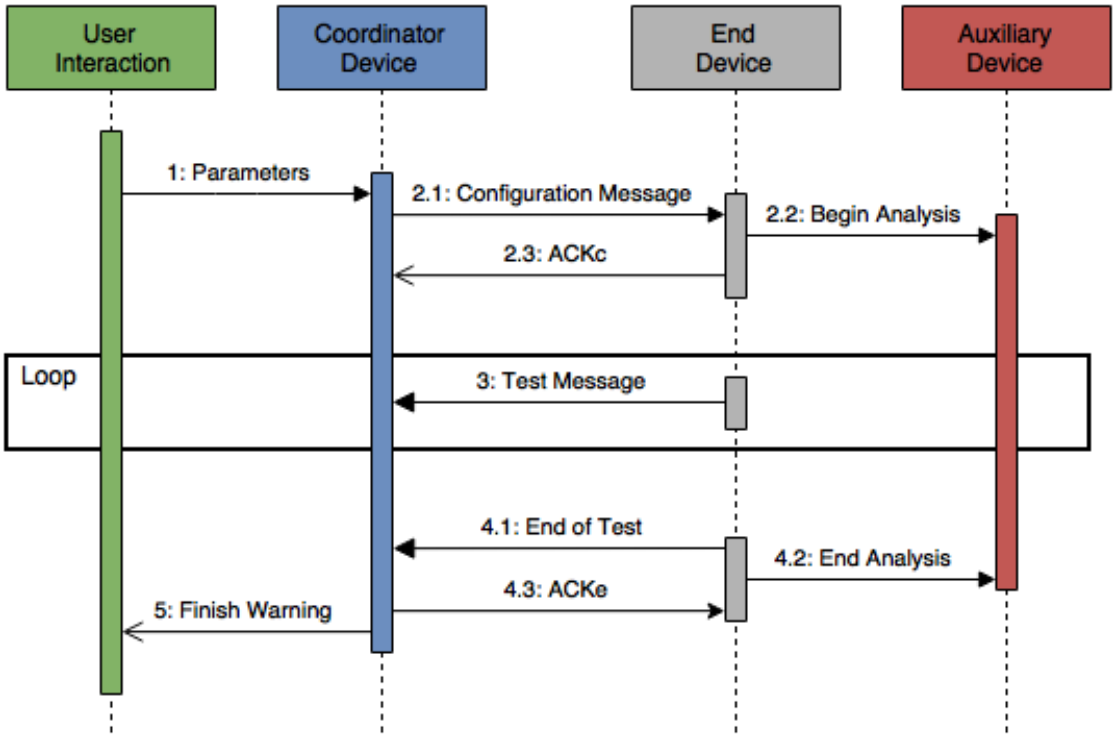


Figure 3.5: Receiving sequence diagram

In this last interaction diagram, Figure 3.5, it is presented the detailed information regarding the communications and interactions between the devices. As it can be seen the initial communications between the devices remains the same, the configuration message transmission and the associated confirmation message.
During the test execution significant changes are presented such as the roles assumed by each device in the testing loop, the coordinator will assume the Receiver role (Rx) being the End device tasked with the Transmitter role(Tx).

As previously explained the device that begins the testing loop transmission must end it with the transmission of the End of Test message, this case is no different being this task done by the End device. Followed to this device communication, the device in question is responsible for the command transmission that disengages the Auxiliary device and also for the End of Test message transmission to the coordinator device.

3.4 Configuration Messages Definition

To implement this conceptual scenario in every test iteration will be transmitted a configuration message from the coordinator device to the end device. This transmission will be the first communication between the devices and will mark the begin of the iteration, making the end device aware of the testing parameters and condition to the actual iteration. In order to realize the automated test it is necessary to generate the configuration message by assembling together the necessary parameters in the configuration message template, as it can be seen in the figure 3.6.

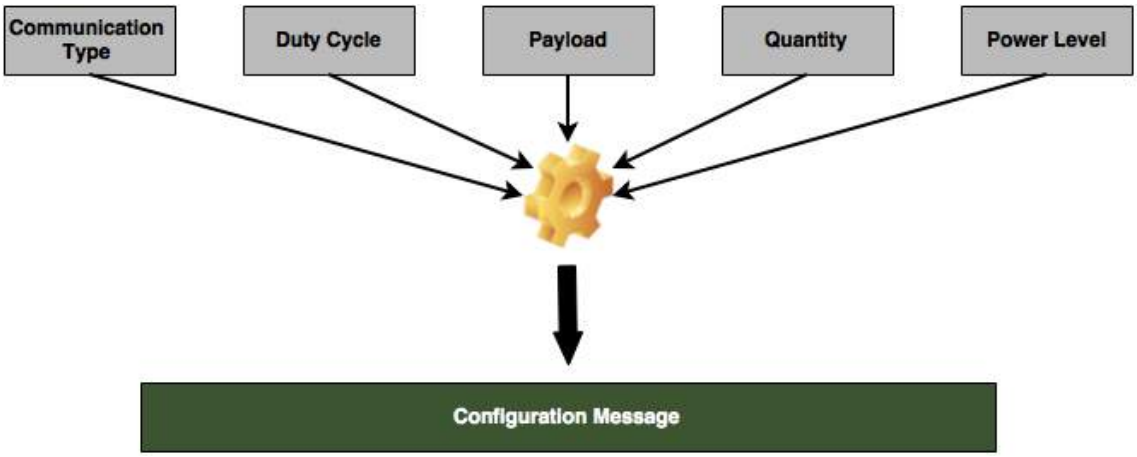


Figure 3.6: Configuration Message Assemble

The Configuration Message template contain the 5 parameters that will define the test iteration. Parameters such as the Communication Type (CT), the transmission Duty Cycle (DC), the Quantity of messages (QT) that will be sent, the Payload (PA) of each message, and the radio Power Level (PL).

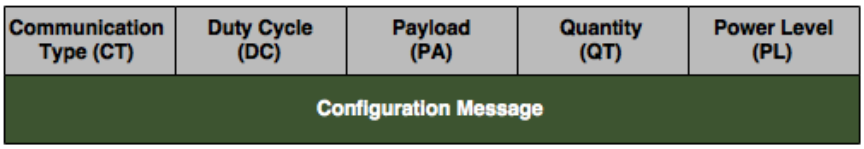


Figure 3.7: Configuration Message Template

Communication Type

The Communication Type message parameter will be used to represent the three cases of communication previously defined. The content of this parameter is a number, being the number 1 the command to set the coordinator as a transmission and the end device as a receiver, the number 2 the command to set the coordinator as a receiver and the end device as a transmitter, the number 3 command is responsible to set a request-reply communication between the coordinator and the end device.

Duty Cycle
To implement the time between communications, exists the Duty Cycle message parameter. Although the duty cycle is defined in percentage, in this parameter the duty cycle will be used as period, defining the time between two consecutive transmissions.

Payload
The Payload message parameter is defined by the number of Bytes that will be sent in one transmission.

Quantity
To give the user the opportunity to realize more than one test exists the Quantity parameter. The idea behind this parameter lies in usefulness of acquisition of more tests allowing the possibility to create a bigger sample.

Power Level
In order to allow the user to test the effects of different power levels of the wireless radio exists the Power Level message parameter. Some of the most common short-range radio devices have five different levels of power, with so, it was defined that to set the highest power level it is required to set the Power Level parameter at the value 4, the same way, to stablish the lowest power level it is required to set the Power Level parameter at the value 0.

3.5 Behaviour Diagram

Despite the fact that the Activity Interaction Diagrams can provide a useful information regarding the interaction between the architecture intervener's, for implementation purposes a behavioural diagram can provide an easier understanding of the automation methodology, as well as a more extensive information regarding the internal actions of each device.
To accomplish this fact, the Behavioural Diagrams are implemented in a state machine form, allowing the possible to stablish the synchronism across the devices more quickly, as it can be seen in the following figures 3.8 to 3.9.The majority of the existing states in the diagrams are common to both devices, coordinator and end device, even so there are states specific to each of the existing devices. In the following behavioural diagrams the transitions between states are represented with a black closed line. To prevent deadlock case scenarios it was added to the architecture some fail-safes, as it can be seen in the diagrams as a red open line.

3.5.1 Auxiliary Device

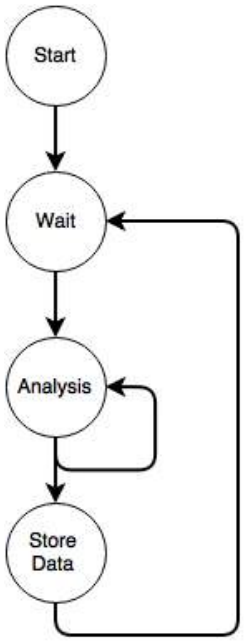


Figure 3.8: Auxiliar Device state machine

Start: Although it is an obvious state, this state cares for a small explanation. The state represents the initial state of the device, the manual action of turning on the device.

Analysis: The analysis state is unique to the Auxiliary device, figure 3.8. This state is continuous loop responsible for the data acquisition power consumption values during the entire test execution.

Store Data: This state is responsible for the storage of the data acquired in the analysis state, due to the nature of the state it only can be found in the Auxiliary device, figure 3.8.

Wait: The current state will maintain a loop until any input exists, in the Coordinator case a manual input generated by the user, a Configuration message in the case of the End device or in the auxiliary device case a start command generated by the end device.

Completed Test: Although this state it is not used in the Auxiliary device this state will be used in the other devices that composes this automation methodology. The state in question do not execute any function, being included only add some visual perception to the behaviour and provide a more easy deployment.

3.5.2 Coordinator Device

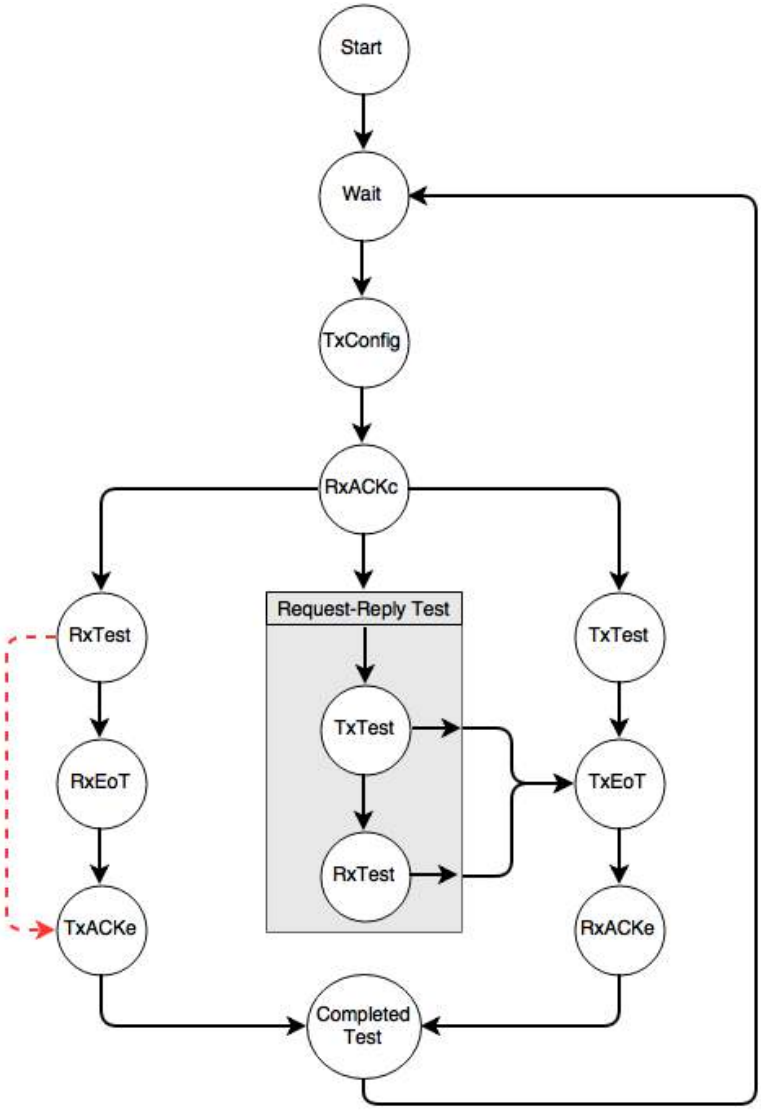


Figure 3.9: Coordinator Device state machine

TxConfig: This state is used exclusively by the Coordinator device, figure 3.9, this state is responsible for the acceptance of the testing parameters from the user and consequently its validations. After the parameters validations, the state generates a configuration message with validates testing parameters and transmits it to the end device.

RxACKc: As well as the previous state, this state is appears exclusively in the the Coordinator device behavioural diagram. This state is responsible for the reception of a confirmation from the end device stating that the configuration message was received successfully. After such confirmation, this state is responsible for the next state correct according to the initial parameters received from the user.

TxTest: This state is used in both,coordinator and end, devices, working as a finite loop state. The operations done in this loop are the generation of testing messages with the defined parameters and conditions set by the user or by the configuration message.

RxTest: As it can be expected this state is the exact opposite of the previous state. It is a loop state with the reception objective, receiving testing messages sent by the other device. The state keeps track the number of received messages comparing it to the number of expected messages established in the configuration message. In case of an exception event occur, the device can detect the reception of a message that does not correspond to an test message. If the message received in this exception situation is a End-of-Test message the device will automatic generate an exception forcing the test termination and jump to the wait state.

Request-Reply Test: In this state, both devices will be tasked with the transmission and reception of test messages. In this state the first testing message is always sent by the coordinator device and received by the end device. After a successful reception in the end device, this device is responsible for the transmission of the second testing message, being the coordinator device obliged with the reception, and so on until the end of messages. Similarly to the RxTest, this state also counts the number of test messages comparing it to the number of expected messages.
The deadlock control was also added to this state being only applied in the end device, since the coordinator is always the first to initialize the test communication. The behaviour of this exception is absolutely the same as the RxTest exception.

TxEoT: This state is used in both devices,Coordinator and End device, being the state responsible for the transmission the information that the test was ended. In order to maintain the architecture free of deadlocks, the transmission this synchronism message is the solely responsibility of the device that start the transmission of the testing messages.

RxEoT: As it can be expected this state is responsible for the reception of a message representing the End-of-Test, being used in both devices.

TxACKe: In order to maintain the synchronism between the Coordinator and End device, this state is responsible for the transmission of one acknowledge confirming the end of test.

RxACKe: As it can be expected, this state is in the architecture to allow the reception of the confirmation of the acknowledge generated by the previous state.

3.5.3 End Device

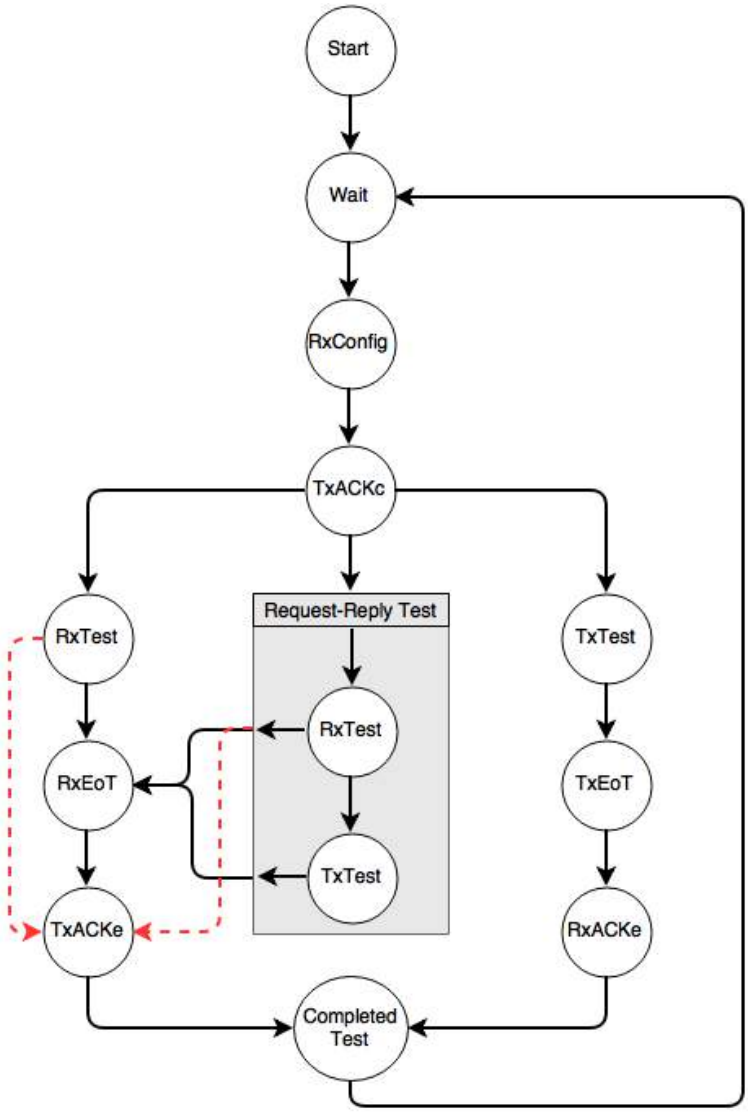


Figure 3.10: End Device state machine

RxConfig: The interaction with the end device, figure 3.10, begins with this state. This state is responsible for the reception of the configuration message from the coordinator device and consequently its validations.

TxACKc: The existence of this state is exclusive to the end device to maintain the synchronism level. In case of a successfully configuration message reception this state is responsible for the transmission, to the coordinator device, of one acknowledge indicating a successful reception. After send this confirmation the state is responsible to make the choice of the adequate next state to the architecture based on the received configuration message.

CHAPTER 4

TESTING AND VALIDATION

4.1 Testing Methodology

The process of finding errors in a system implementations through experimentation is also know as testing process. The testing process usually is carried out in a controlled environment where the normal and exceptional use of the system can be simulated. It is important to understand that testing process cannot ensure complete correctness of an implementation, it only show the presence of errors, not their absence (Tretmans, 2001).

Regarding to testing methodologies, there are several methodologies to evaluate the suitability of solutions to meet their requirements, each with its specific field of application (Onofre, 2007). The testing methodology chosen in this work is the ISO-9646: "Open Systems Interconnection (OSI) Conformance Testing Methodology and Framework". This standard aims to the definition of a testing methodology, a framework for specifying test suites, and procedures to be followed during testing. It is due to the fact that this standard provides a generic testing methodology that it was chosen.

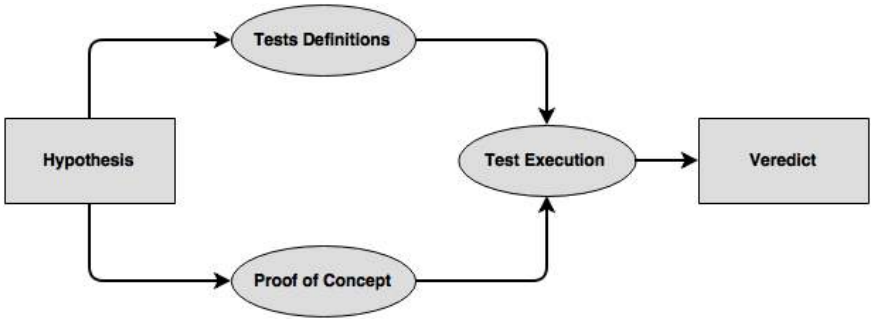


Figure 4.1: Conformance Testing Process, based on (Tretmans, J. 2001)

A simplified version of this standard is used presented in the Figure 4.1. In order to test the hypothesis specifications two situations must occur, a set of tests must be defined and the hypothesis must be implemented as proof-of-concept. The implementation of the proof-of-concept does not required a fully-functional implementation, it only requires that the implementation comply with the defined architecture proposes. The test execution results are then observed, leading to a verdict on the compliance of the system under test with the initial requirements defined (Tretmans 2001).

To ensure that the test can be reproduced, they are defined according to the standard ISO 9646 recommendation, a notation independent of any implementation with an wide acceptance, the Tree and Tabular Combined Notation - Version 2 (TTCN-2). With this notation it is possible to specify the behaviour of tests by the sequence of events that need to occur during the test Tretmans 1992.

The Tree and Tabular Combined Notation - Version 2 (TTCN-2) notation table it is composed by a header, where is defined the test name, its purpose, the inputs needed during the test execution and the resultant outputs. This table is defined in a tabular form, using incremental indentation to specify a successive chain of events and the same indentation to define alternative events. It is important to highlight that the specified events preceded by the usage of a exclamation mark indicates actions to be executed, being the events preceded by the question mark conditional event. Each sequence ends with the specification of the verdict that is assigned when the execution of the sequence terminates that can be a: “Success”, “Fail” or “Inconclusive”, as despicted in the Table 4.1.

Multiplication by 2 Test Case		
Test name:	Test the Service multiplication by 2	
Purpose:	Check if the Service is available and provide the result of the operation	
Inputs:	[I1]: Number to be multiplied, [I2]:Expected result	
Outputs	[O1]: Result of the multiplication of the input by two	
Line Number	**Behaviour**	**Verdict**
1	! Invoke Service with parameters (I1)	
2	? Returned data as Service Result (O1) being the Service Available	
3	? Result (O1) is equal to Expected (I2)	SUCCESS
4	? Result (O1) is different from the Expected (I2)	FAIL
5	? No Service Result	INCONCLUSIVE

Table 4.1: Example of a TTCN based test table

The previously presented table exhibits a service invocation test, a multiplication by two test case followed by the comparison with an expected outcome. The verdict can output three different results: "Success", "Fail", or "Inconclusive". "Success" indicates that the test was executed successfully, in this particular case it indicates that the first provided number when multiplied by two has the same result as the provided expected number. The "Fail" result indicates that the implementation does not conform to the specification,

in this particular case it can indicate that the multiplication of the first provided number by two has not the expected result. "Inconclusive" results indicates that no evidence of non-conformance was found, but that the test purpose was not achieved.

Test	Input		Output	Result	
	I1: Number	I2: Expected Result	O1: Result	Expected	Actual
1	10	20	20	(3)	(3)
2	80	170	160	(3)	(4)
3	0	0	No data	(3)	(5)

Table 4.2: Test Case Example

The obtained results from the tests execution then need to be presented, in this particular case to provide a more analytical view it was presented as a table, Table 4.2. This illustrate the previous example test, in which, each row represents a specific test case and the verdict number resultant from the TTCN table, for the expected and actual results.

4.2 Proof of Concept Implementation

The implementation of the Proof of Concept is in fact the implementation of the methodology defined in the previous chapter, the *Methodology for WSN communication technologies automated field tests*.

In order to implement the communication between the Coordinator and the End device several wireless radio devices were used, such as HM-11 radio device that implements Bluetooth Low Energy (BLE) protocol, XBee-S1 and XBeePro-S1 that implements 802.15.4 protocol. Once the communication is established the devices will exchange messages according to the configuration message template previously defined. To accomplish this, after each configuration message component will appear the equal sign representing the beginning of each parameter value followed by a semicolon representing the end of the value, as it can be seen in the Figure 4.2.

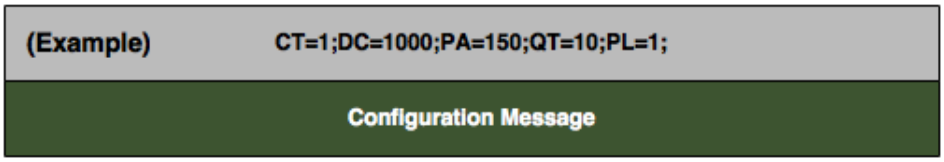


Figure 4.2: Example of configuration message

Coordinator device

The existence of several testing parameters required a solution that could provide the maximum user interaction when choosing the testing parameters and creating the testing cases. To accomplish this, the coordinator device was defined using software, a computer to run it and a USB-XBee adapter to establish the communication between the device and the network.

In order to develop the software used in the creation of the proof of concept the programming language used was the C#, due to the existence of native Input/Output (I/O) libraries that allow a faster software development.

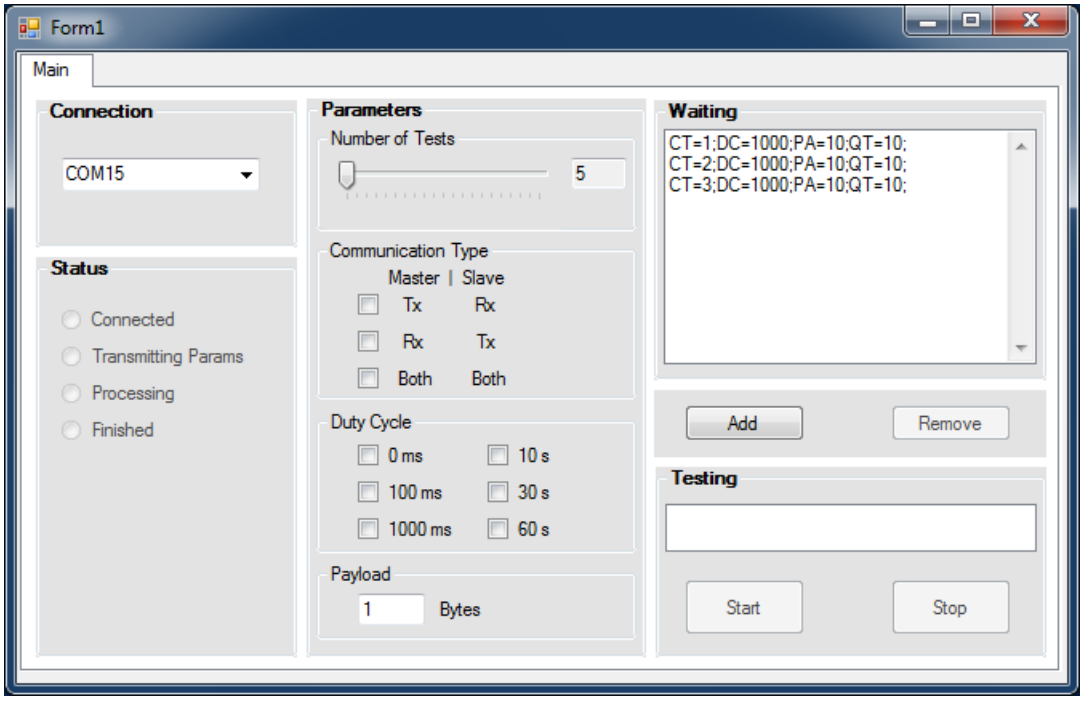


Figure 4.3: Proof of concept Coordinator GUI

End Device

To implement the end device was used a Arduino UNO and the target wireless radio device, since the tested radios had the XBee format it was also used a XBee Shield to connect the Arduino to the radio. Despite of the End device receive a complete Configuration Message, the Power Level field was not used, due to some technical difficulties on the hardware side. This difficulties implementing an automatic Power Level change on the wireless radio were caused by the wireless radio, since it unintentionally forces the host to reboot, causing a synchronism lost with the coordinator device. The request-reply scenario is implemented using both the transmit and the receive scenario's, with such the end device will only respond to a request in the next duty cycle period.

Auxiliary Device

The device implementation used was meant to provide a simplistic way to process the test methodology. To accomplish this it is used a Arduino UNO with the Current/Power Sensor INA219, is connected to a computer to provide the data acquisition.

4.3 Test Definition and Execution

4.3.1 Test Definition

This test is performed to grant that the methodology can actually analyse different wireless radio with different parameters, which is a main characteristic of the methodology. To perform the test there is the loading the testing parameters to the execution engine so that it can perform the model transformation execution.

Wireless Radio Test Case		
Test name:	Test Wireless Radio	
Purpose:	Acquire data regarding the Wireless radio power consumption	
Inputs:	[I1]: Communication Type, [I2]:Duty Cycle, [I3]: Communication Type, [I4]:Payload, [I5]: Power Level, [I6]: Device environment	
Outputs:	[O1]: Acquired Data	
Line Number	**Behaviour**	**Verdict**
1	! Choice of the wireless radio to be tested	
2	! Deployment of the wireless radio into the Coordinator and End Device	
3	! Setup Testing Cases on Coordinator device using the Inputs [I1], [I2], [I3, [I4], [I5]	
4	! Place devices in test environment conditions defined by Input [I6]	
5	! Connect together the End and Auxiliary Devices	
6	! Activate the Coordinator Device	
7	? Activate the End Device and the Auxiliary	
8	? Execute device analysis successfully	
9	? Successfully retrieve [O1] from Auxiliary device	SUCESS
10	? Failed to retrieve [O1] from Auxiliary device	FAIL
11	? Failed to execute device analysis	FAIL
12	? Failed to Activate the End Device and the Auxiliary	FAIL

Table 4.3: Wireless Radio Test Definition

4.3.2 Test Execution

More than one test is executed using different inputs from the previous definitions to verify the consistency of results. To perform all the tests in the Table XX, the

Test 1: In this case the methodology is applied to two wireless devices in order to acquire data regarding the overall power consumption behaviour of the device when facing a range of payloads, being necessary to perform this test several complementary tests, presented in the Table 4.4.

Test	Input						Output	Result	
	I1: Communication Protocol (Device Used)	I2: Duty Cycle	I3: Communication Type	I4: Payload	I5: Power Level	I6: Device Environment	O1: Results (mW)	Expected	Actual
2.1	XBee S1	1000	End Device Transmit	10	4	LoS	444.5	(9)	(9)
2.2	XBee S1	1000	End Device Transmit	100	4	LoS	444.4	(9)	(9)
2.3	XBee S1	1000	End Device Transmit	200	4	LoS	443.9	(9)	(9)
2.4	XBee S1	1000	End Device Transmit	300	4	LoS	443.5	(9)	(9)
2.5	XBee S1	1000	End Device Receive	10	4	LoS	433.4	(9)	(9)
2.6	XBee S1	1000	End Device Receive	100	4	LoS	433.6	(9)	(9)
2.7	XBee S1	1000	End Device Receive	200	4	LoS	433.5	(9)	(9)
2.8	XBee S1	1000	End Device Receive	300	4	LoS	433.6	(9)	(9)
2.9	XBee S1	1000	Request-Reply	10	4	LoS	434,9	(9)	(9)
2.10	XBee S1	1000	Request-Reply	100	4	LoS	434,2	(9)	(9)
2.11	XBee S1	1000	Request-Reply	200	4	LoS	433,5	(9)	(9)
2.12	XBee S1	1000	Request-Reply	300	4	LoS	433,1	(9)	(9)
2.13	XBee-PRO S1	1000	End Device Transmit	10	4	LoS	451.8	(9)	(9)
2.14	XBee-PRO S1	1000	End Device Transmit	100	4	LoS	451.5	(9)	(9)
2.15	XBee-PRO S1	1000	End Device Transmit	200	4	LoS	451.5	(9)	(9)
2.16	XBee-PRO S1	1000	End Device Transmit	300	4	LoS	452.1	(9)	(9)
2.17	XBee-PRO S1	1000	End Device Receive	10	4	LoS	448.6	(9)	(9)
2.18	XBee-PRO S1	1000	End Device Receive	100	4	LoS	448.7	(9)	(9)
2.19	XBee-PRO S1	1000	End Device Receive	200	4	LoS	449.1	(9)	(9)
2.20	XBee-PRO S1	1000	End Device Receive	300	4	LoS	450.8	(9)	(9)
2.21	XBee-PRO S1	1000	Request-Reply	10	4	LoS	466.8	(9)	(9)
2.22	XBee-PRO S1	1000	Request-Reply	100	4	LoS	467.1	(9)	(9)
2.23	XBee-PRO S1	1000	Request-Reply	200	4	LoS	466.9	(9)	(9)
2.24	XBee-PRO S1	1000	Request-Reply	300	4	LoS	466.8	(9)	(9)
2.25	BLE	1000	End Device Transmit	10	4	LoS	218.4	(9)	(9)
2.26	BLE	1000	End Device Transmit	100	4	LoS	218.2	(9)	(9)
2.27	BLE	1000	End Device Transmit	200	4	LoS	218.1	(9)	(9)
2.28	BLE	1000	End Device Transmit	300	4	LoS	218.0	(9)	(9)
2.29	BLE	1000	End Device Receive	10	4	LoS	213.4	(9)	(9)
2.30	BLE	1000	End Device Receive	100	4	LoS	212.4	(9)	(9)
2.31	BLE	1000	End Device Receive	200	4	LoS	211.4	(9)	(9)
2.32	BLE	1000	End Device Receive	300	4	LoS	210.5	(9)	(9)
2.33	BLE	1000	Request-Reply	10	4	LoS	219.5	(9)	(9)
2.34	BLE	1000	Request-Reply	100	4	LoS	219.1	(9)	(9)
2.35	BLE	1000	Request-Reply	200	4	LoS	218.7	(9)	(9)
2.36	BLE	1000	Request-Reply	300	4	LoS	218.4	(9)	(9)

Table 4.4: Payload Test Execution

It is expected that the output from this several test the return of data, and after the execution this data is processed obtaining the results presented in the Figure 4.4.

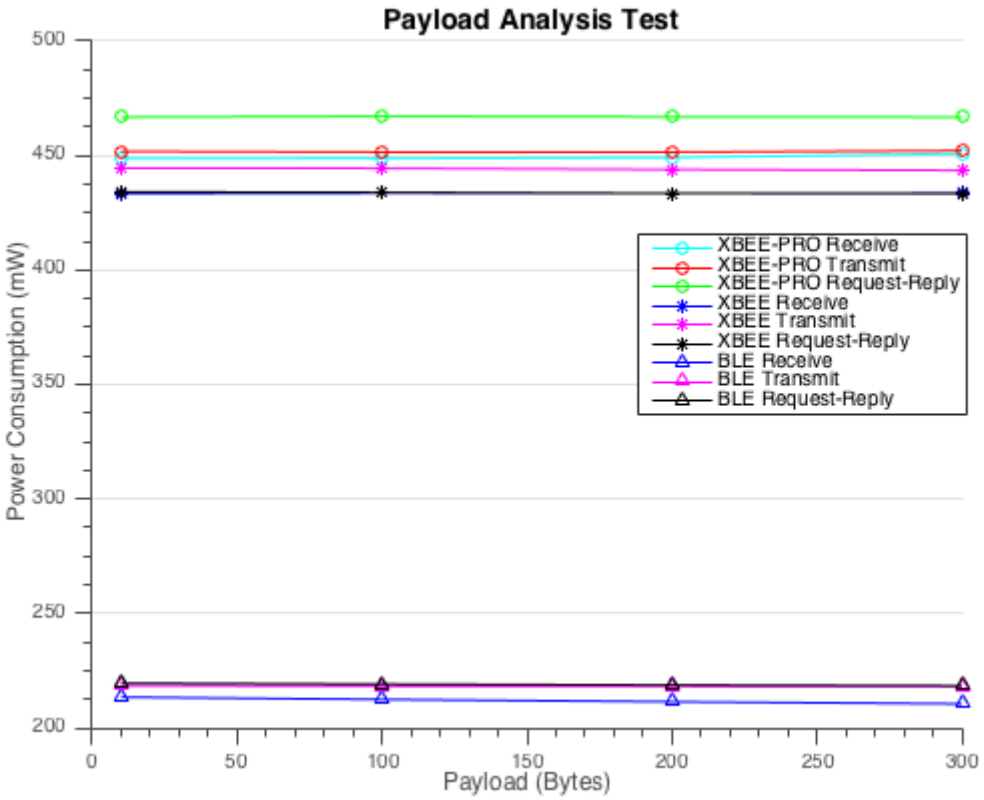


Figure 4.4: Payload analysis test

With the analysis of the previous image, Figure 4.4 it's possible to conclude that with the presented tests it was possible to characterise the power consumption of the wireless radios when facing communications with messages with different payloads.

Test 2: In this case the methodology is applied to two wireless devices in order to acquire data regarding the overall power consumption behaviour of the device when facing a range of duty cycle, being necessary to perform this test several complementary tests,presented in Table 4.5.

Test	Input						Output	Result	
	I1: Communication Protocol (Device Used)	I2: Duty Cycle	I3: Communication Type	I4: Payload	I5: Power Level	I6: Device Environment	O1: Results (mW)	Expected	Actual
1.1	XBee S1	0	End Device Transmit	10	4	LoS	445,1	(9)	(9)
1.2	XBee S1	100	End Device Transmit	10	4	LoS	443,2	(9)	(9)
1.3	XBee S1	1000	End Device Transmit	10	4	LoS	442,2	(9)	(9)
1.4	XBee S1	10000	End Device Transmit	10	4	LoS	440,1	(9)	(9)
1.5	XBee S1	30000	End Device Transmit	10	4	LoS	439,4	(9)	(9)
1.6	XBee S1	60000	End Device Transmit	10	4	LoS	438,9	(9)	(9)
1.7	XBee S1	0	End Device Receive	10	4	LoS	443,0	(9)	(9)
1.8	XBee S1	100	End Device Receive	10	4	LoS	442,5	(9)	(9)
1.9	XBee S1	1000	End Device Receive	10	4	LoS	441,0	(9)	(9)
1.10	XBee S1	10000	End Device Receive	10	4	LoS	440,5	(9)	(9)
1.11	XBee S1	30000	End Device Receive	10	4	LoS	440,0	(9)	(9)
1.12	XBee S1	60000	End Device Receive	10	4	LoS	439,0	(9)	(9)
1.13	XBee S1	0	Request-Reply	10	4	LoS	444,2	(9)	(9)
1.14	XBee S1	100	Request-Reply	10	4	LoS	443,8	(9)	(9)
1.15	XBee S1	1000	Request-Reply	10	4	LoS	443,7	(9)	(9)
1.16	XBee S1	10000	Request-Reply	10	4	LoS	441,2	(9)	(9)
1.17	XBee S1	30000	Request-Reply	10	4	LoS	440,8	(9)	(9)
1.18	XBee S1	60000	Request-Reply	10	4	LoS	440,2	(9)	(9)
1.19	XBee-PRO S1	0	End Device Transmit	10	4	LoS	462,1	(9)	(9)
1.20	XBee-PRO S1	100	End Device Transmit	10	4	LoS	461,6	(9)	(9)
1.21	XBee-PRO S1	1000	End Device Transmit	10	4	LoS	457,7	(9)	(9)
1.22	XBee-PRO S1	10000	End Device Transmit	10	4	LoS	457,2	(9)	(9)
1.23	XBee-PRO S1	30000	End Device Transmit	10	4	LoS	458,2	(9)	(9)
1.24	XBee-PRO S1	60000	End Device Transmit	10	4	LoS	458,7	(9)	(9)
1.25	XBee-PRO S1	0	End Device Receive	10	4	LoS	452,1	(9)	(9)
1.26	XBee-PRO S1	100	End Device Receive	10	4	LoS	451,5	(9)	(9)
1.27	XBee-PRO S1	1000	End Device Receive	10	4	LoS	451,5	(9)	(9)
1.28	XBee-PRO S1	10000	End Device Receive	10	4	LoS	443,6	(9)	(9)
1.29	XBee-PRO S1	30000	End Device Receive	10	4	LoS	444,2	(9)	(9)
1.30	XBee-PRO S1	60000	End Device Receive	10	4	LoS	445,3	(9)	(9)
1.31	XBee-PRO S1	0	Request-Reply	10	4	LoS	446,4	(9)	(9)
1.32	XBee-PRO S1	100	Request-Reply	10	4	LoS	446,1	(9)	(9)
1.33	XBee-PRO S1	1000	Request-Reply	10	4	LoS	445,8	(9)	(9)
1.34	XBee-PRO S1	10000	Request-Reply	10	4	LoS	442,9	(9)	(9)
1.35	XBee-PRO S1	30000	Request-Reply	10	4	LoS	442,0	(9)	(9)
1.36	XBee-PRO S1	60000	Request-Reply	10	4	LoS	441,8	(9)	(9)
1.37	BLE	0	End Device Transmit	10	4	LoS	219,4	(9)	(9)
1.38	BLE	100	End Device Transmit	10	4	LoS	219,9	(9)	(9)
1.39	BLE	1000	End Device Transmit	10	4	LoS	220,8	(9)	(9)
1.40	BLE	10000	End Device Transmit	10	4	LoS	221,0	(9)	(9)
1.41	BLE	30000	End Device Transmit	10	4	LoS	221,1	(9)	(9)
1.42	BLE	60000	End Device Transmit	10	4	LoS	221,1	(9)	(9)
1.43	BLE	0	End Device Receive	10	4	LoS	217,9	(9)	(9)
1.44	BLE	100	End Device Receive	10	4	LoS	218,0	(9)	(9)
1.45	BLE	1000	End Device Receive	10	4	LoS	218,3	(9)	(9)
1.46	BLE	10000	End Device Receive	10	4	LoS	218,9	(9)	(9)
1.47	BLE	30000	End Device Receive	10	4	LoS	218,9	(9)	(9)
1.48	BLE	60000	End Device Receive	10	4	LoS	219,1	(9)	(9)
1.49	BLE	0	Request-Reply	10	4	LoS	219,3	(9)	(9)
1.50	BLE	100	Request-Reply	10	4	LoS	219,3	(9)	(9)
1.51	BLE	1000	Request-Reply	10	4	LoS	219,2	(9)	(9)
1.52	BLE	10000	Request-Reply	10	4	LoS	219,1	(9)	(9)
1.53	BLE	30000	Request-Reply	10	4	LoS	219,0	(9)	(9)
1.54	BLE	60000	Request-Reply	10	4	LoS	218,9	(9)	(9)

Table 4.5: Duty cycle Test Execution

It is expected that the output from this several test the return of data, and after the execution this data is processed obtaining the results presented in the Figure 4.5.

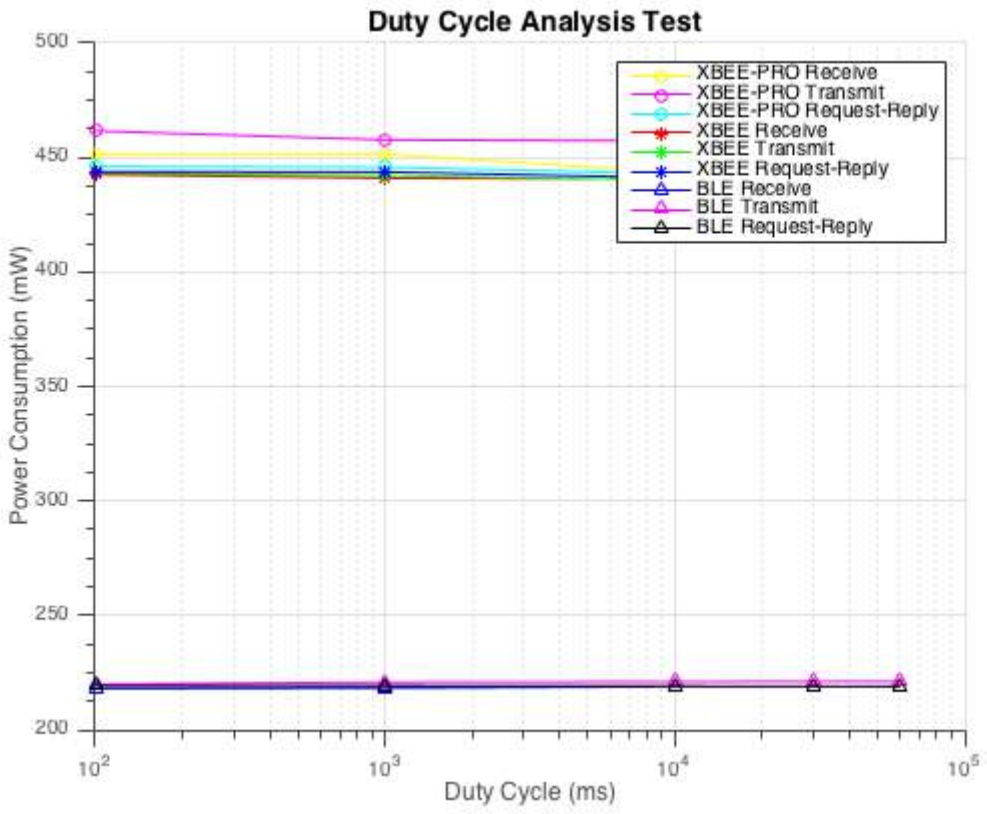


Figure 4.5: Duty cycle analysis test

With the analysis of the previous image, Figure 4.5 it's possible to conclude that with the presented tests it was possible to characterise the power consumption of the wireless radios when facing communications with different duty cycle messages.

Test 3: In this case the methodology is applied to two wireless devices in order to acquire data regarding the overall power consumption behaviour of the device when facing a range of duty cycle, being necessary to perform this test several complementary tests,presented in the Table 4.6.

Test	Input						Output	Result	
	I1: Communication Protocol (Device Used)	I2: Duty Cycle	I3: Communication Type	I4: Payload	I5: Power Level	I6: Device Environment	O1: Results (mW)	Expected	Actual
1.1	XBee S1	1000	End Device Transmit	10	0(-10 dBm)	LoS	431.2	(9)	(9)
1.2	XBee S1	1000	End Device Transmit	10	1 (-6 dBm)	LoS	431.9	(9)	(9)
1.3	XBee S1	1000	End Device Transmit	10	2 (-4 dBm)	LoS	433.3	(9)	(9)
1.4	XBee S1	1000	End Device Transmit	10	3 (-2 dBm)	LoS	433.9	(9)	(9)
1.5	XBee S1	1000	End Device Transmit	10	4 (0 dBm)	LoS	445.2	(9)	(9)
1.6	XBee S1	1000	End Device Receive	10	0(-10 dBm)	LoS	426.9	(9)	(9)
1.7	XBee S1	1000	End Device Receive	10	1 (-6 dBm)	LoS	426.8	(9)	(9)
1.8	XBee S1	1000	End Device Receive	10	2 (-4 dBm)	LoS	427.3	(9)	(9)
1.9	XBee S1	1000	End Device Receive	10	3 (-2 dBm)	LoS	428.8	(9)	(9)
1.10	XBee S1	1000	End Device Receive	10	4 (0 dBm)	LoS	440.4	(9)	(9)
1.11	XBee S1	1000	Request-Reply	10	0(-10 dBm)	LoS	430,1	(9)	(9)
1.12	XBee S1	1000	Request-Reply	10	1 (-6 dBm)	LoS	432,0	(9)	(9)
1.13	XBee S1	1000	Request-Reply	10	2 (-4 dBm)	LoS	432,5	(9)	(9)
1.14	XBee S1	1000	Request-Reply	10	3 (-2 dBm)	LoS	439.0	(9)	(9)
1.15	XBee S1	1000	Request-Reply	10	4 (0 dBm)	LoS	442,1	(9)	(9)
1.16	XBee-PRO S1	1000	End Device Transmit	10	0 (-3 dBm)	LoS	464.2	(9)	(9)
1.17	XBee-PRO S1	1000	End Device Transmit	10	1 (0 dBm)	LoS	463.9	(9)	(9)
1.18	XBee-PRO S1	1000	End Device Transmit	10	2 (2 dBm)	LoS	464.2	(9)	(9)
1.19	XBee-PRO S1	1000	End Device Transmit	10	3 (8 dBm)	LoS	464.9	(9)	(9)
1.20	XBee-PRO S1	1000	End Device Transmit	10	4 (10 dBm)	LoS	467.2	(9)	(9)
1.21	XBee-PRO S1	1000	End Device Receive	10	0 (-3 dBm)	LoS	458.7	(9)	(9)
1.22	XBee-PRO S1	1000	End Device Receive	10	1 (0 dBm)	LoS	458.9	(9)	(9)
1.23	XBee-PRO S1	1000	End Device Receive	10	2 (2 dBm)	LoS	459.1	(9)	(9)
1.24	XBee-PRO S1	1000	End Device Receive	10	3 (8 dBm)	LoS	459.5	(9)	(9)
1.25	XBee-PRO S1	1000	End Device Receive	10	4 (10 dBm)	LoS	462.0	(9)	(9)
1.26	XBee-PRO S1	1000	Request-Reply	10	0 (-3 dBm)	LoS	462.0	(9)	(9)
1.27	XBee-PRO S1	1000	Request-Reply	10	1 (0 dBm)	LoS	462.3	(9)	(9)
1.28	XBee-PRO S1	1000	Request-Reply	10	2 (2 dBm)	LoS	462,6	(9)	(9)
1.29	XBee-PRO S1	1000	Request-Reply	10	3 (8 dBm)	LoS	464,5	(9)	(9)
1.30	XBee-PRO S1	1000	Request-Reply	10	4 (10 dBm)	LoS	466,5	(9)	(9)
1.31	BLE	1000	End Device Transmit	10	0(-23 dBm)	LoS	-	(9)	(11)
1.32	BLE	1000	End Device Transmit	10	1(-12 dBm)	LoS	-	(9)	(11)
1.33	BLE	1000	End Device Transmit	10	2 (-6 dBm)	LoS	-	(9)	(11)
1.34	BLE	1000	End Device Transmit	10	3 (0 dBm)	LoS	-	(9)	(11)
1.35	BLE	1000	End Device Transmit	10	4 (6 dBm)	LoS	220,8	(9)	(9)
1.36	BLE	1000	End Device Receive	10	0(-23 dBm)	LoS	-	(9)	(11)
1.37	BLE	1000	End Device Receive	10	1(-12 dBm)	LoS	-	(9)	(11)
1.38	BLE	1000	End Device Receive	10	2 (-6 dBm)	LoS	-	(9)	(11)
1.39	BLE	1000	End Device Receive	10	3 (0 dBm)	LoS	-	(9)	(11)
1.40	BLE	1000	End Device Receive	10	4 (6 dBm)	LoS	218,3	(9)	(9)
1.41	BLE	1000	Request-Reply	10	0(-23 dBm)	LoS	-	(9)	(11)
1.42	BLE	1000	Request-Reply	10	1(-12 dBm)	LoS	-	(9)	(11)
1.43	BLE	1000	Request-Reply	10	2 (-6 dBm)	LoS	-	(9)	(11)
1.44	BLE	1000	Request-Reply	10	3 (0 dBm)	LoS	-	(9)	(11)
1.45	BLE	1000	Request-Reply	10	4 (6 dBm)	LoS	219,9	(9)	(9)

Table 4.6: Power Level Test Execution

It is expected that the output from this several test the return of data, and after the execution this data is processed obtaining the results presented in the Figure 4.6.

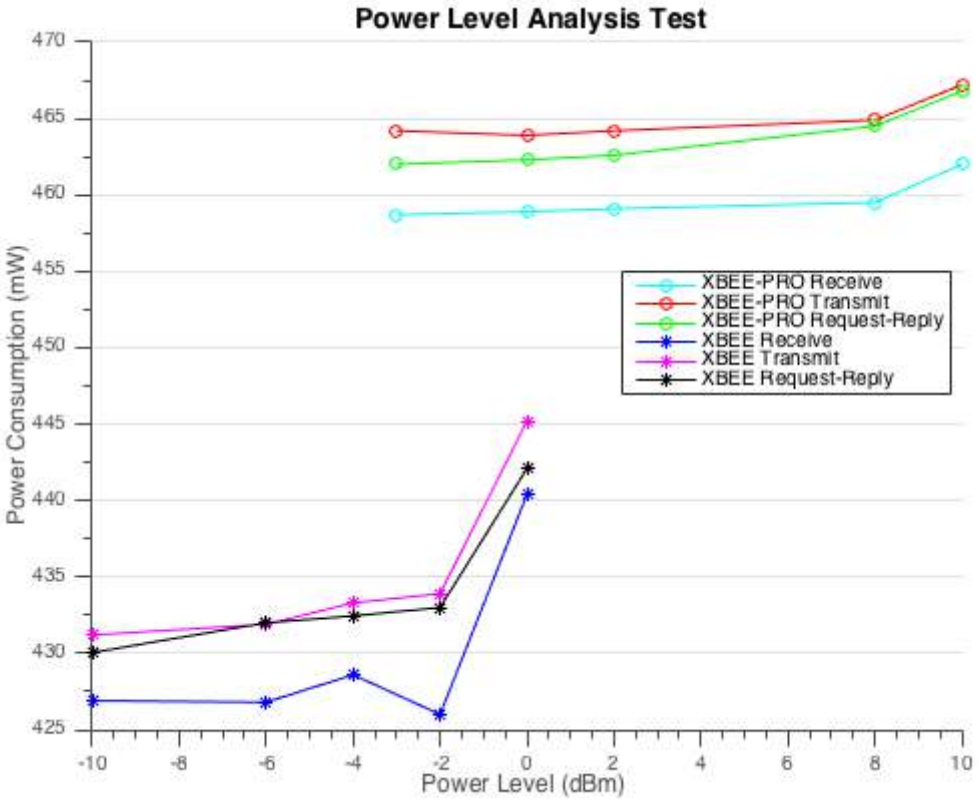


Figure 4.6: Power Level analysis test

With the analysis of the previous image, Figure 4.6 it's possible to conclude that with the presented tests it was possible to characterise the power consumption of the wireless radios when using different radio power levels.

4.4 Verdict

From the executed tests, several conclusions can be drawn. The first conclusion is that the implemented proof of concept successfully passed all the tests. It successfully acquires data from the wireless devices power consumptions and allow the analysis of the acquired results.

The three sets of test done were designed taking into consideration the characteristics of the problem presented in earlier chapters: achieve an approach that could allow the characterization of the radio device's power consumption. It is also important to stress that this methodology and the executed tests are not intended to provide a comparison between the devices, been the true purpose of it the analysis of the consumption of the wireless radio's when facing different characteristics and parameters.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Internet of Things (IoT) is no longer a prediction from the future, it started to be built around Predictive Maintenances, Smart Grids, Smart Homes, Intelligent Buildings, etc; providing us useful resources in our daily life. One form of IoT deployment is through a Wireless Sensor Network (WSN), a network of devices that use wireless devices to communicate amongst themselves or a central device, mainly used to retrieve sensing data and actuate over other devices.

In order to completely understand the existing difficulties when deploying such networks a case scenario was presented, the deployment of such network across a campus environment providing a scenario within the IoT itself. The study of this scenario led to the problem definition and presented the research question: *How to choose the appropriate Wireless Technology for a WSN application?*

A state-of-the-art research was performed as a way to gather information regarding the characterized problem and to identify similar existing work related with the presented problem. Each of the study elements has been analysed, in order to identify the different approaches that could be taken into account in order to solve the characterized problem. None of the studied elements provided a complete automated methodology that could be used to satisfy the encountered problem, even so, each of the individual elements provided some important key-points with valuable information regarding the problem characteristics.

- Communication Protocol:The choice of this parameter was due to the existence of a great number of wireless protocols and devices;
- Communication Type: This parameter was included in order to analyse the impact of the type of communication in any other parameters;

- Device Environment: Since the wireless signal can be degraded in a great number of ways it is important to analyse the impact of this parameter over the other important parameters, being also possible to analyse the device behaviour when testing it in different deployment environments;

- Payload: The existence of this parameter is due to the fact that an behaviour analysis of the wireless device can represent an asset when analysing the device performance when transmitting messages with a specific payload;

- Power Level: To achieve an optimal WSN deployment it can be necessary the analysis of this parameter against the power consumption and the signal range;

- Duty Cycle: The analysis of the different duty cycle effects over other parameters can provide information regarding the wireless device behaviour;

When facing the identified problem characteristics, it was clear that a manual testing, validation and analysis of each wireless radio and the associated characteristics would not be a good method. To improve this situation, a methodology for wireless technologies automated field tests was introduces as an alternative to the existing validation methods. The presented methodology allows the acquisition of multiple types of data regarding the device's behaviour when facing a vast number of trial field tests, providing a simplified validation method of wireless technologies for non-technical users.

In order to gather enough information to perform the automated validation method, three devices were defined with the following roles:

- Coordinator device representing the device where the tests will be defined;

- End device representing the targeted hardware where the wireless radio will be set to be tested;

- Auxiliary device representing the device that will acquire the results from the trial field testing;

To provide detailed information regarding the interactions performed by each device in this methodology several activity interaction diagrams were presented, one for each communication types. With this information it is possible to have a better understanding of the existing steps in each communication type.
For this methodology to work, it was necessary to create a configuration message template. This template would be filled by the coordinator device with the configured parameters used in the following trial field test. This message would be then transmitted to the end device where it would be opened, it's content extracted and then applied to the end device, so it could have the appropriate behaviour to the trial test.

The internal architecture of each device was defined by behavioural diagrams as a possibility to provide a more tangible insight over this methodology, thus allowing a faster deployment in a real device.
This methodology was implemented as a proof-of concept being used in several case scenarios providing data and analysis regarding the power consumption of the wireless devices in several conditions. Thus, allowing to conclude that all the tests passed with a successful verdict.
At last it can be concluded that the work developed in the elapse of this thesis resulted in a very satisfactory result. The initial problem with all its characteristics was solved successfully, and it was proved that the proposed hypothesis is correct.

5.1 Future Work

One of the possible enhancements that could be applied to the developed methodology is the possibility to modify the architecture to support custom configuration messages. With this modification, more flexibility can be given to the user in order to prepare and run different types of field trial tests.

Another interesting feature that could be implemented is the possibility to expand the existing methodology to accept different communication paradigm, in order to other communication scenarios, such as multiple requests to one reply and vice versa.

The integration of an automated methodology to perform the results comparison would minimize the effort required to analyse each test individually. This integration would represent a significant improvement since the amount of data generated in a field test can be surprisingly high.

At last, combining the proposed enhancements with the presented methodology, a reference tool could be developed in order to provide an easiest way to test, analyse and validate the wireless radios, allowing a more easy deployment of future WSN.

BIBLIOGRAPHY

Bell, D. (2004). *UML basics: The sequence diagram*. URL: http://www.ibm.com/developerworks/rational/library/3101.html.

Carry, S. S. (2011). *A Beginner's Guide to Scientific Method, Fourth Edition*. Wadsworth Publishing.

Frenzel, L. (2012). *The Fundamentals Of ShortRange Wireless Technology*. URL: http://electronicdesign.com/communications/fundamentals-short-range-wireless-technology.

Gupta, S. (2007). *Choosing a Wireless Implementation Strategy and Applications*. Tech. rep. Invensys.

ISO/IEC 9646-1:1994 (1994). Tech. rep.

Onofre, S. (2007). "Plataforma para testes de conformidade de sistemas baseados em módulos conceptuais step". MA thesis. Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia - Departamento de Engenharia Electrotécnica.

Reiter, G. (2014). *Wireless connectivity for the Internet of Things*. Tech. rep. Texas Instruments.

Schafersman, S. D. (1997). *An introduction to science. Scientific Thinking and the Scientific Method.* URL: http://isites.harvard.edu/fs/docs/icb.topic265890.files/Critical_Thinking_File/05_ScientificMethod_and_Critical_Thinking.pdf.

Tretmans, J. (1992). *A Formal Approach to Conformance Testing*. Tech. rep. University of Twente.

— (2001). *An overview of osi conformance testing*. Tech. rep. University of Twente.

Zatout, Y. (2012). "Using Wireless Technologies for Healthcare Monitoring at Home: a Survey". In: *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*.

APPENDIX A

COORDINATOR DEVICE CODE

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Windows.Forms;
9  using System.IO.Ports;
10
11 namespace AuxiliarSaver{
12     public partial class Form1 : Form{
13         public int auxConnect = 0;
14         public int GlobalRSSI = 0;
15         public int GlobalTxC = 0;
16         public int GlobalRxE = 0;
17         public int GlobalTxE = 0;
18         public int GlobalRxC = 0;
19
20         public Form1(){
21             InitializeComponent();
22         }
23
24         private void comboSerial_Click(object sender, EventArgs e){
25             this.comboSerial.Items.Clear();
26             string[] ports = SerialPort.GetPortNames();
27             foreach (string port in ports){
28                 this.comboSerial.Items.Add(port);
29             }
30         }
31
```

```
32          public void SetAuxConnect() {
33              auxConnect = 1;
34          }
35          private void comboSerial_SelectedIndexChanged(object sender, EventArgs e){
36              this.serialPort1.PortName = this.comboSerial.SelectedItem.ToString();
37              this.buttonStart.Enabled = true;
38          }
39
40          private void wait_queue_SelectedIndexChanged(object sender, EventArgs e){
41              this.buttonRemove.Enabled = true;
42          }
43
44          private void trackNumber_ValueChanged(object sender, EventArgs e){
45              this.textNumber.Text = this.trackNumber.Value.ToString();
46          }
47
48          private void textPayload_TextChanged(object sender, EventArgs e){
49              int aux = 0;
50              if (int.TryParse(this.textPayload.Text.ToString(), out aux) == true) {
51                  if (aux > 100){
52                      ShowMessage("The␣choosen␣number␣is␣greater␣than␣100.
53 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Choose␣a␣smaller␣number!");
54                      this.textPayload.Text = "100";
55                  }
56                  if (aux <= 0){
57                      ShowMessage("The␣choosen␣number␣must␣be␣greater␣than␣0!");
58                      this.textPayload.Text = "1";
59                  }
60              }
61              else{
62                  ShowMessage("Must␣choose␣a␣number!");
63                  this.textPayload.Text = "1";
64              }
65          }
66
67          private void ShowMessage(String msg){
68              MessageBox.Show(msg, "Warning␣Message", MessageBoxButtons.OK,
69                  MessageBoxIcon.Error);
70          }
71
72          private void CreateValues(String DCValue){
73              String aux1, aux2, aux3;
74              String Comp1 = "DC=" + DCValue + ";";
75              String Comp2 = "PA=" + this.textPayload.Text + ";";
76              String Comp3 = "QT=" + this.textNumber.Text + ";";
77              if (this.checkTxRx.Checked){
78                  aux1 = "CT=1;" + Comp1 + Comp2 + Comp3;
79                  this.wait_queue.Items.Add(aux1);
80              }
81              if (this.checkRxTx.Checked){
```

```
82              aux2 = "CT=2;" + Comp1 + Comp2 + Comp3;
83              this.wait_queue.Items.Add(aux2);
84          }
85          if (this.checkBoth.Checked){
86              aux3 = "CT=3;" + Comp1 + Comp2 + Comp3;
87              this.wait_queue.Items.Add(aux3);
88          }
89      }
90
91      void InsertStatistics(String inParam){
92          int auxIntCT = Selector("CT",inParam);
93          String auxCT = "";
94          String auxDC = Selector("DC",inParam).ToString();
95          String auxPA = Selector("PA", inParam).ToString();
96          String auxQT = Selector("QT", inParam).ToString();
97          if (auxIntCT == 1) {
98              auxCT = "TxRx";
99          }
100         if (auxIntCT == 2){
101             auxCT = "RxTx";
102         }
103         if (auxIntCT == 3){
104             auxCT = "BiDirectional";
105         }
106         dataGrid.Rows.Add(auxCT,auxDC,auxPA,auxQT,
107             GlobalRSSI.ToString(),GlobalTxC.ToString(),
108             GlobalRxE.ToString(), GlobalRxC.ToString(),
109             GlobalTxE.ToString());
110     }
111
112     private void buttonStart_Click(object sender, EventArgs e){
113         if (this.wait_queue.Items.Count != 0){
114             SerialConnect();
115             this.buttonStart.Enabled = false;
116             this.buttonStop.Enabled = true;
117             this.buttonAdd.Enabled = false;
118             this.buttonRemove.Enabled = false;
119             int qtdtestes = this.wait_queue.Items.Count;
120             int idtestes = 0;
121             String Params;
122             while (qtdtestes != idtestes){
123                 idtestes++;
124                 Console.WriteLine("\n\n-------TEST␣" + idtestes +
125                     "␣of␣"+qtdtestes+"-------");
126                 Params = "";
127                 Get_the_first();
128                 Params = this.testing_queue.Items[0].ToString();
129
130                 System.Threading.Thread.Sleep(1000);
131                 RunMachine(Params);
```

```
132                     InsertStatistics(Params);
133                     Clear_testing_Queue();
134                     System.Threading.Thread.Sleep(5000);
135                 }
136                 this.buttonStart.Enabled = true;
137                 this.buttonStop.Enabled = false;
138                 this.buttonAdd.Enabled = true;
139                 this.buttonRemove.Enabled = false;
140             }
141             else {
142                 ShowMessage("Please␣add␣some␣Parameters␣to␣the␣Waiting␣Queue!");
143             }
144         }
145 
146         private void buttonStop_Click(object sender, EventArgs e){
147             this.buttonStop.Enabled = false;
148             this.buttonStart.Enabled = true;
149         }
150 
151         private void buttonAdd_Click(object sender, EventArgs e){
152             if (((checkDC0.Checked) || (checkDC1.Checked) ||
153                 (checkDC2.Checked) || (checkDC3.Checked) || (checkDC4.Checked)
154                 || (checkDC5.Checked)) && ((checkTxRx.Checked) ||
155                 (checkRxTx.Checked) || (checkBoth.Checked))){
156                 if (checkDC0.Checked) { CreateValues("0"); }
157                 if (checkDC1.Checked) { CreateValues("100"); }
158                 if (checkDC2.Checked) { CreateValues("1000"); }
159                 if (checkDC3.Checked) { CreateValues("10000"); }
160                 if (checkDC4.Checked) { CreateValues("30000"); }
161                 if (checkDC5.Checked) { CreateValues("60000"); }
162             }
163             else{
164                 if ((!checkTxRx.Checked) && (!checkRxTx.Checked) &&
165                     (!checkBoth.Checked)){
166                         ShowMessage("Check␣at␣least␣one␣Comunication␣Type");
167                 }
168                 if ((!checkDC0.Checked) && (!checkDC1.Checked) &&
169                     (!checkDC2.Checked) && (!checkDC3.Checked) &&
170                     (!checkDC4.Checked) && (!checkDC5.Checked)) {
171                         ShowMessage("Check␣at␣least␣one␣Duty␣Cycle");
172                 }
173             }
174         }
175         private void buttonRemove_Click(object sender, EventArgs e){
176             this.wait_queue.Items.Remove(this.wait_queue.SelectedItem);
177             this.buttonRemove.Enabled = false;
178         }
179 
180         private void Clear_testing_Queue() {
181             if (this.testing_queue.Items.Count != 0){
```

```
182                 this.testing_queue.Items.RemoveAt(0);
183             }
184         }
185 
186         private void Get_the_first() {
187             if(this.wait_queue.Items.Count !=0){
188                 Clear_testing_Queue();
189                 String first = this.wait_queue.Items[0].ToString();
190                 this.testing_queue.Items.Add(first);
191                 this.wait_queue.Items.Remove(first);
192                 }
193         }
194 
195         private String GeneratePayload(int val) {
196             String payload = "";
197             for (int i = 0; i < val; i++) {
198                 payload += "X";
199             }
200             return payload;
201         }
202 
203         ////////////////////////////////////////////////////////////////////
204         ////////////            STATE MACHINE FUNCTIONS          ////////////
205         ////////////////////////////////////////////////////////////////////
206         enum States { TxConfig, RxACKc, TxTest, RxTest, BothTest,
207             TxEoT, RxEoT, RxACKe, TxACKe, Wait }
208 
209         private States ActualState = States.TxConfig;
210 
211         private void TxConfigCoding(String msg){
212             Console.WriteLine("STATE--->" + ActualState.ToString());
213             SerialWrite(msg);
214             Console.WriteLine("Sending:"+msg);
215             ActualState = States.RxACKc;
216             this.mark_transmit.Checked = true;
217         }
218 
219         private void RxACKcCoding(String msg){
220             Console.WriteLine("STATE--->" + ActualState.ToString());
221             int CTValue = Selector("CT", msg);
222             String aux ="␣␣␣␣";
223             while ((aux[0] !='A')&&(aux[1] !='C')&&(aux[2] !='K')&&
224                 (aux[3] !='c')) {
225                 aux = "␣␣␣␣";
226                 aux = SerialRead();
227                 Console.WriteLine("Received:"+aux);
228             }
229             GlobalRSSI = Selector(aux[5].ToString(), aux);
230 
231             if (CTValue == 1){
```

```
232                 ActualState = States.TxTest;
233             }
234             if (CTValue == 2){
235                 ActualState = States.RxTest;
236             }
237 
238             if (CTValue == 3){
239                 ActualState = States.BothTest;
240             }
241 
242             this.mark_processing.Checked= true;
243             System.Threading.Thread.Sleep(1000);
244         }
245 
246         private void TxTestCoding(String msg){
247             Console.WriteLine("STATE--->" + ActualState.ToString());
248             int PAValue = Selector("PA", msg);
249             int QTValue = Selector("QT", msg);
250             int DCValue = Selector("DC", msg);
251             int count = 0;
252             String payload = GeneratePayload(PAValue);
253             int testeshow;
254             while (count != QTValue){
255                 testeshow = count + 1;
256                 SerialWrite(payload);
257                 count++;
258                 Console.WriteLine("Test:" + count + "/" + QTValue);
259                 System.Threading.Thread.Sleep(DCValue);
260             }
261             GlobalTxC = count;
262             ActualState = States.TxEoT;
263         }
264 
265         private void RxTestCoding(String msg){
266             Console.WriteLine("STATE--->" + ActualState.ToString());
267             int PAValue = Selector("PA", msg);
268             int QTValue = Selector("QT", msg);
269             int DCValue = Selector("DC", msg);
270             int count = 0;
271             String aux = "␣␣␣";
272             while (((aux[0] != 'E') || (aux[1] != 'o') || (aux[2] != 'T'))
273                 && (count != QTValue)){
274                 aux = "";
275                 aux = SerialRead();
276                 Console.WriteLine("Receiving:" + aux);
277                 System.Threading.Thread.Sleep(DCValue);
278                 count++;
279             }
280             if (aux == "EoT"){
281                 Console.WriteLine("Jumping␣to␣STATE␣---->␣TxACKe"
```

```
282                 + ActualState.ToString());
283             ActualState = States.TxACKe;
284         }
285         else{
286             ActualState = States.RxEoT;
287         }
288         GlobalRxC = count;
289     }
290 
291     private void BothTestCoding(String msg){
292         Console.WriteLine("STATE--->" + ActualState.ToString());
293         int PAValue = Selector("PA", msg);
294         int QTValue = Selector("QT", msg);
295         int DCValue = Selector("DC", msg);
296         int count = 0;
297         String aux="␣␣␣";
298         String payload = GeneratePayload(PAValue);
299         while ((count != QTValue) && ((aux[0] != 'E') && (aux[1] != 'o')
300             && (aux[2] != 'T'))){
301             System.Threading.Thread.Sleep(DCValue);
302             SerialWrite(payload);
303             count++;
304             GlobalTxC++;
305             Console.WriteLine("TX␣GC:"+count);
306             System.Threading.Thread.Sleep(DCValue);
307             if (count != QTValue) {
308                 aux = "␣␣␣";
309                 while ((aux != payload) && ((aux[0]!='E') &&
310                     (aux[1]!='o')&&(aux[2]!='T'))) {
311                     aux = "";
312                     aux = SerialRead();
313                     Console.WriteLine("Receiving:" + aux);
314                     if (aux.Length < 3){
315                         aux = "␣␣␣";
316                     }
317                 }
318                 count++;
319                 GlobalRxE++;
320                 Console.WriteLine("RX␣GC:" + count);
321             }
322         }
323         ActualState = States.TxEoT;
324     }
325 
326     private void TxEoTCoding(){
327         Console.WriteLine("STATE--->" + ActualState.ToString());
328         String aux = "EoT";
329         System.Threading.Thread.Sleep(1000);
330         SerialWrite(aux);
331         Console.WriteLine("Sending:" + aux);
```

```
332             System.Threading.Thread.Sleep(1000);
333             ActualState = States.RxACKe;
334         }
335 
336         private void RxEoTCoding(){
337             Console.WriteLine("STATE--->" + ActualState.ToString());
338             String aux = "␣␣␣";
339             while ((aux[0] != 'E') && (aux[1] != 'o') && (aux[2] != 'T')){
340                 aux = SerialRead();
341                 Console.WriteLine("Received:" + aux);
342             }
343             GlobalTxE = Selector("T", aux);
344             ActualState = States.TxACKe;
345         }
346 
347         private void RxACKeCoding(String msg){
348             Console.WriteLine("STATE--->" + ActualState.ToString());
349             int CTValue=Selector("CT",msg);
350             String aux = "␣␣␣␣";
351             while ((aux[0] != 'A') && (aux[1] != 'C') && (aux[2] != 'K')
352                 && (aux[3] != 'e')){
353                     aux = SerialRead();
354                     Console.WriteLine("Received:" + aux);
355             }
356             if (CTValue == 1){
357                 GlobalRxE = Selector("R", aux);
358             }
359             if (CTValue == 3){
360                 CTValue = Selector("B", aux);
361             }
362             ActualState = States.Wait;
363         }
364 
365         private void TxACKeCoding(){
366             Console.WriteLine("STATE--->" + ActualState.ToString());
367             String aux = "ACKe";
368             System.Threading.Thread.Sleep(1000);
369             SerialWrite(aux);
370             Console.WriteLine("Sending:" + aux);
371             ActualState = States.Wait;
372         }
373 
374         private void WaitCoding(){
375             Console.WriteLine("STATE--->" + ActualState.ToString());
376             ActualState = States.TxConfig;
377         }
378 
379         public void RunMachine(String msg){
380             while (ActualState != States.Wait){
381                 switch (ActualState){
```

```
382                     case States.TxConfig:
383                         TxConfigCoding(msg);
384                         break;
385                     case States.RxACKc:
386                         RxACKcCoding(msg);
387                         break;
388                     case States.TxTest:
389                         TxTestCoding(msg);
390                         break;
391                     case States.RxTest:
392                         RxTestCoding(msg);
393                         break;
394                     case States.BothTest:
395                         BothTestCoding(msg);
396                         break;
397                     case States.TxEoT:
398                         TxEoTCoding();
399                         break;
400                     case States.RxEoT:
401                         RxEoTCoding();
402                         break;
403                     case States.RxACKe:
404                         RxACKeCoding(msg);
405                         break;
406                     case States.TxACKe:
407                         TxACKeCoding();
408                         break;
409                 }
410             }
411             WaitCoding();
412         }
413
414         private int Selector(String param, String configparam){
415             String[] mult = configparam.Split(';');
416             String sub = "0";
417             if (param == "CT"){
418                 sub = mult[0].Substring(3);
419             }
420             if (param == "DC"){
421                 sub = mult[1].Substring(3);
422             }
423             if (param == "PA"){
424                 sub = mult[2].Substring(3);
425             }
426             if (param == "QT"){
427                 sub = mult[3].Substring(3);
428             }
429             if (param == "T"){
430                 sub = mult[1].Substring(1);
431             }
```

```
432             if (param == "R"){
433                 sub = mult[1].Substring(1);
434             }
435             if (param == "I"){
436                 sub = mult[1].Substring(1);
437             }
438             if (param == "B"){
439                 String beta = mult[1].Substring(1);
440                 beta.Replace("B","");
441                 String[] val = beta.Split('/');
442                 GlobalRxE = Int32.Parse(val[0].Substring(0));
443                 GlobalTxE = Int32.Parse(val[1].Substring(0));
444                 sub = "0";
445             }
446             return Int32.Parse(sub);
447         }
448         ////////////////////////////////////////////////////////////////////
449         ////////////             SERIAL FUNCTIONS             ////////////
450         ////////////////////////////////////////////////////////////////////
451         private void SerialConnect() {
452             try {
453                 this.serialPort1.Open();
454             }catch (Exception ex) { ShowMessage(ex.Message.ToString()); }
455         }
456 
457         private void SerialDisconnect(){
458             try{
459                 this.serialPort1.Close();
460             }catch (Exception ex) { ShowMessage(ex.Message.ToString()); }
461         }
462 
463         private void SerialWrite(String msg) {
464             try{
465                 this.serialPort1.WriteLine(msg);
466             }catch (Exception ex) { ShowMessage(ex.Message.ToString()); }
467         }
468 
469         private String SerialRead() {
470             String aux = "";
471             try{
472                 aux = this.serialPort1.ReadLine();
473             }catch (Exception ex) { ShowMessage(ex.Message.ToString()); }
474             aux = aux.Replace("\r", "");
475             return aux;
476         }
477     }
478 }
```

APPENDIX B

END DEVICE CODE

```
#include <SoftwareSerial.h>

SoftwareSerial BeeSerial(2, 3);

////////////////////////////////////////////////////////
///                    DATA DEFINITION                ///
////////////////////////////////////////////////////////
//Definition of State Machine
enum ExistingStates{RxConfig,TxACKc,TxTest,RxTest,
                    BothTest,TxEoT,RxEoT,TxACKe,RxACKe,Wait};
ExistingStates State=RxConfig;

String ParametersMessage = "";
int CTValue=0;
long int DCValue=0;
int PLValue=0;
int QTValue=0;

int led = 13;
int globalcount =0;
String LocalData ="";
String RemoteData="";
String Payload ="";
int CountTX=0;
int CountRX=0;

////////////////////////////////////////////////////////
///               FUNCTIONS DEFINITION                ///
////////////////////////////////////////////////////////
void setup() {
  pinMode(led, OUTPUT);
```

```
32    Serial.begin(9600);
33    BeeSerial.begin(9600);
34  }
35
36  void loop() {
37    switch(State){
38      case RxConfig:
39            RxConfigCoding();
40            break;
41      case TxACKc:
42            TxACKcCoding();
43            break;
44      case TxTest:
45            TxTestCoding();
46            break;
47      case RxTest:
48            RxTestCoding();
49            break;
50      case BothTest:
51            BothTestCoding();
52            break;
53      case TxEoT:
54            TxEoTCoding();
55            break;
56      case RxEoT:
57            RxEoTCoding();
58            break;
59      case TxACKe:
60            TxACKeCoding();
61            break;
62      case RxACKe:
63            RxACKeCoding();
64            break;
65      case Wait:
66            WaitCoding();
67            break;
68    }
69  }
70
71  void RxConfigCoding(){ //Receive PARAMS from Coord Device
72    CTValue = -1;
73    DCValue = -1;
74    PLValue = -1;
75    QTValue = -1;
76    String aux="";
77    while ((aux[0]!='C')&&(aux[1]!='T')){
78      aux="";
79      aux= ReadCommunication();
80      Serial.print("XBEE_RX:");Serial.println(aux);
81    }
```

```
82    msgSorter(aux);
83    State=TxACKc;
84    Serial.println("RxConfigCoding");
85  }
86
87  void TxACKcCoding(){ //Receive PARAMS from Coord Device
88    delay(1500);
89    String aux="ACKc;I";
90    aux=aux+"0";
91    WriteCommunication(aux);
92      if(CTValue == 1){
93        State = RxTest;
94      }
95      if(CTValue == 2){
96        State = TxTest;
97      }
98      if(CTValue == 3){
99        State = BothTest;
100     }
101     BlinkN(1000);
102     Serial.println("TxACKcCoding");
103
104 }
105
106 void RxTestCoding(){//Receives TEST PACKETS from COORD Device
107   String aux="";
108   globalcount=0;
109   String Payload = generatePayload(PLValue);
110   while ((globalcount!=QTValue) &&
111       ((aux[0]!='E')&&(aux[1]!='o')&&(aux[2]!='T'))){
112     aux="";
113     aux= ReadCommunication();
114     if(Payload==aux){
115         globalcount++;
116     }
117   }
118   if(aux=="EoT"){
119     CountRX= globalcount;
120     globalcount =0;
121     State = TxACKe;
122     Serial.println("Jumping␣to␣TxACKe");
123   }
124   else{
125     CountRX= globalcount;
126     State = RxEoT;
127   }
128   Serial.println("RxTestCoding");
129 }
130
131
```

```
132 void TxTestCoding(){//Transmits TEST PACKETS to END Device
133   globalcount = 0;
134   Payload =generatePayload(PLValue);
135   while(globalcount!=QTValue){
136     WriteCommunication(Payload);
137     delay(DCValue);
138     globalcount++;
139   }
140   delay(1000);
141   State = TxEoT;
142   Serial.println("TxTestCoding");
143 }
144
145
146
147 void BothTestCoding(){
148   String aux="";
149   globalcount=0;
150   String Payload = generatePayload(PLValue);
151   while ((globalcount!=QTValue)&&(aux!="EoT")){
152     aux="";
153     delay(DCValue);
154     while((aux!=Payload)&&(aux!="EoT")){
155       aux= ReadCommunication();
156     }
157     delay(DCValue);
158     globalcount++;
159     if((globalcount!=QTValue)&&(aux!="EoT")){
160       WriteCommunication(Payload);
161       globalcount++;
162     }
163     Serial.println("");Serial.println("SAAI");
164   }
165   if(aux=="EoT"){
166     State = TxACKe;
167     Serial.println("Jumping␣to␣TxACKe");
168   }
169   else{
170     State = RxEoT;
171   }
172   Serial.println("BothTestCoding");
173 }
174
175
176
177
178 void TxEoTCoding(){
179   delay(1000);
180   String aux ="EoT;T";
181   aux= aux + String(CountTX);
```

```
182   WriteCommunication(aux);
183   State = RxACKe;
184   Serial.println("TxEoTCoding");
185 }
186
187 void RxEoTCoding(){ //Receive ACK from END Device
188   String aux="";
189   while (aux!="EoT"){
190     aux="";
191       aux= ReadCommunication();
192     }
193     State = TxACKe;
194     Serial.println("RxEoTCoding");
195 }
196
197 void TxACKeCoding(){
198   delay(1000);
199   String aux;
200   if(CTValue==1){
201      aux ="ACKe;R";
202      aux= aux + String(CountRX);
203    }
204    if(CTValue ==3){
205      aux ="ACKe;B";
206      aux= aux + String(CountRX)+"/"+String(CountTX);
207    }
208
209
210   WriteCommunication(aux);
211   State = Wait;
212   Serial.println("TxACKeCoding");
213 }
214
215 void RxACKeCoding(){ //Receive ACK from END Device
216   String aux="";
217   while (aux!="ACKe"){
218     aux="";
219     aux= ReadCommunication();
220   }
221   State = Wait;
222   Serial.println("RxACKeCoding");
223 }
224 void WaitCoding(){
225   BlinkN(3000);
226   delay(1000);
227   State = RxConfig;
228   Serial.println("WaitCoding");
229 }
230
231 ////////////////////////////////////////////////////////
```

```
232 /// GENERAL FUNCTIONS ///
233 ////////////////////////////////////////////////////////////
234 String generatePayload(int value){
235   String aux="";
236   for(int i=0;i<value;i++){
237     aux +="X";
238   }
239   return aux;
240 }
241
242 void BlinkN(int duration){
243     digitalWrite(led, HIGH);
244     delay(duration);
245     digitalWrite(led, LOW);
246 }
247
248 void String2UInt(String input,int SizeI, uint8_t output[]){
249   char charB[SizeI+1];
250   input.toCharArray(charB,SizeI+1);
251   for(int i=0; i<SizeI;i++){
252     output[i]=charB[i];
253   }
254 }
255
256 //Update Each values of the data received
257 void msgSorter(String data){
258   String trimmed, aux;
259   int dim=0;
260   for(int i=0;i<=3;i++){
261     trimmed="";
262     trimmed=MsgTrim(data,';',i);
263     if((trimmed[0]=='C')&&(trimmed[1]=='T')){
264       aux=MsgValue(trimmed);
265       CTValue=aux.toInt();
266     }
267     if((trimmed[0]=='D')&&(trimmed[1]=='C')){
268       aux=MsgValue(trimmed);
269       DCValue=aux.toInt();
270     }
271     if((trimmed[0]=='P')&&(trimmed[1]=='L')){
272       aux=MsgValue(trimmed);
273       PLValue=aux.toInt();
274     }
275     if((trimmed[0]=='Q')&&(trimmed[1]=='T')){
276       aux=MsgValue(trimmed);
277       QTValue=aux.toInt();
278     }
279   }
280 }
281
```

```
282 //Get the value existing in each string.
283 String MsgValue(String data){
284   String aux="";
285   for (int i=3; i<=data.length();i++){
286     aux+=data[i];
287   }
288   return aux;
289 }
290
291 //Breaks the Received String in Smaller Strings
292 String MsgTrim(String data, char separator, int index){
293   int maxIndex = data.length()-1;
294   int j=0;
295   String chunkVal = "";
296   for(int i=0; i<=maxIndex && j<=index; i++){
297     chunkVal.concat(data[i]);
298     if(data[i]==separator){
299       j++;
300       if(j>index){
301         chunkVal.trim();
302         return chunkVal;
303       }
304       chunkVal = "";
305     }
306   }
307 }
308
309 void WriteCommunication(String msg){
310   BeeSerial.println(msg);
311   Serial.print("TX:");
312   Serial.println(msg);
313 }
314
315 String ReadCommunication(){
316   String aux ="";
317   aux=BeeSerial.readStringUntil('\n');
318   Serial.print("RX:");
319   Serial.println(aux);
320     return aux;
321 }
```

APPENDIX C

AUXILIARY DEVICE CODE - ARDUINO

```
#include <Wire.h>
#include <Adafruit_INA219.h>

Adafruit_INA219 ina219;

int i = 0;

void setup(void){
  uint32_t currentFrequency;

  Serial.begin(115200);
  ina219.begin();
}

void loop(void){
  float busvoltage = 0;
  float current_mA = 0;

  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();

  Serial.println("A");
  Serial.println(busvoltage);
  Serial.println(current_mA);
  Serial.println("B");

}
```

APPENDIX D

AUXILIARY DEVICE CODE - COMPUTER

```
1  import processing.serial.*;
2
3
4  Serial mySerial;// Define serial port
5  int lf = 10;    // Linefeed in ASCII
6
7  Table table;
8
9  String filename = null;
10 String Date = null;
11 String Time = null;
12 String BusVoltage = null;
13 String Current = null;
14
15 int flag = 0;
16
17 float shuntv = 0;
18 float busv = 0;
19 float curr = 0;
20 float avg_curr = 0;
21 float min_curr = 1000;
22 float max_curr = 0;
23
24 float curr_sum = 0;
25 float sample_num = 0;
26
27 long savedTimeMillis = 0;
28 int savedTimeSeconds = 0;
29
30 long savedTime;
31 int totalTime = 300000; //time interval between file saves in ms.
```

```
32
33 void setup() {
34   size(400, 400);
35
36   // List all the available serial ports
37   println(Serial.list());
38
39   // Open the port you are using at the rate you want:
40   mySerial = new Serial(this, Serial.list()[2], 115200);
41   mySerial.clear();
42
43   int dd = day();    // Values from 1 - 31
44   int mm = month();  // Values from 1 - 12
45   int yyyy = year();   // 2003, 2004, 2005, etc.
46
47   String[] date = new String[3];
48   date[0] = String.valueOf(dd);
49   date[1] = String.valueOf(mm);
50   date[2] = String.valueOf(yyyy);
51   Date = join(date, ":");
52
53   int s = second();  // Values from 0 - 59
54   int m = minute();  // Values from 0 - 59
55   int h = hour();    // Values from 0 - 23
56
57   String[] time = new String[3];
58   time[0] = String.valueOf(h);
59   time[1] = String.valueOf(m);
60   time[2] = String.valueOf(s);
61   Time = join(time, ".");
62
63   filename = "Data/Data␣" + Date + "␣" + Time+".csv";
64   table = new Table();
65
66   table.addColumn("Date");
67   table.addColumn("Time");
68   table.addColumn("Millis");
69   table.addColumn("Bus␣Voltage␣(V)");
70   table.addColumn("Current␣(mA)");
71
72   savedTime = millis();
73 }
74
75 void refreshDateTime() {
76   int s = second();  // Values from 0 - 59
77   int m = minute();  // Values from 0 - 59
78   int h = hour();    // Values from 0 - 23
79   int dd = day();    // Values from 1 - 31
80   int mm = month();  // Values from 1 - 12
81   int yyyy = year();   // 2003, 2004, 2005, etc.
```

```
82
83    if ( savedTimeSeconds != s)
84    {
85      savedTimeSeconds = s;
86      savedTimeMillis = millis();
87    }
88
89    String[] date = new String[3];
90    date[0] = String.valueOf(dd);
91    date[1] = String.valueOf(mm);
92    date[2] = String.valueOf(yyyy);
93    Date = join(date, "/");
94
95    int sss = (int) (millis() - savedTimeMillis);
96
97    String[] time = new String[3];
98    time[0] = String.valueOf(h);
99    time[1] = String.valueOf(m);
100   time[2] = String.valueOf(s);
101   Time = join(time, ":");
102
103 }
104
105
106 void draw() {
107   while (mySerial.available () > 0) {
108     String myString = mySerial.readStringUntil(lf);
109     if (myString != null) {
110
111       if ( flag == 2 ) {
112         Current = myString;
113         flag = 3;
114       }
115
116       if ( flag == 1 ) {
117         BusVoltage = myString;
118         flag = 2;
119       }
120
121       String str = "A";
122       if (myString.indexOf(str) != -1)
123         flag = 1;
124
125       String str2 = "B";
126       if (myString.indexOf(str2) != -1 && flag == 3) {
127         flag = 0;
128
129         busv = float(BusVoltage);
130         curr = float(Current);
131         curr_sum += curr;
```

```
132         sample_num++;
133 
134         if (curr > max_curr)
135           max_curr = curr;
136         if (curr < min_curr)
137           min_curr = curr;
138         avg_curr = curr_sum / sample_num;
139 
140         refreshDateTime();
141         TableRow newRow = table.addRow();
142         newRow.setString("Date", Date);
143         newRow.setString("Time", Time);
144         newRow.setInt("Millis", millis());
145         newRow.setFloat("Bus␣Voltage␣(V)", busv);
146         newRow.setFloat("Current␣(mA)", curr);
147 
148         background(204);
149         textSize(30);
150         fill(50);
151         textAlign(CENTER);
152         text("Do␣not␣stop!", 50, 10, 300, 350);
153         text("Press␣'S'␣to␣save␣and␣close.", 50, 75, 300, 300);
154         textSize(15);
155         fill(50);
156         text("Bus␣Voltage␣(V):␣"+BusVoltage, 50, 230, 300, 350);
157         text("Current␣(mA):␣"+Current, 50, 260, 300, 350);
158         text("Average␣Current␣(mA):␣"+
159           String.format("%.2f", avg_curr), 50, 290, 300, 350);
160         text("Maximum␣Current␣(mA):␣"+
161           String.format("%.2f", max_curr), 50, 320, 300, 350);
162         text("Minimum␣Current␣(mA):␣"+
163           String.format("%.2f", min_curr), 50, 350, 300, 350);
164       }
165     }
166   }
167   // Calculate how much time has passed
168   long passedTime = millis() - savedTime;
169   if (passedTime > totalTime) {
170     saveTable(table, filename, "csv");
171     savedTime = millis(); // Save the current time to restart the timer!
172   }
173 }
174 
175 void keyPressed() {
176   if (key == 's' || key == 'S') {
177     saveTable(table, filename, "csv");
178     exit();  // Stops the program
179   }
180 }
```