

MASTER OF TELECOMMUNICATION SYSTEM ENGINEERING
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



End-of-Master's Project

Analysis of LoRa and Bluetooth Low Energy to create networks for IoT.

AUTHOR: RAFAEL PÉREZ CAMPOS
DIRECTOR: DAVID A. SÁNCHEZ HERNÁNDEZ
CO-DIRECTOR: MIGUEL ÁNGEL GARCÍA FERNÁNDEZ

Cartagena, September 2019

Acknowledgements

This work has been developed thanks to the help provided by my directors, that is, David Sánchez and Miguel A. García. They have been my guides in this unknown way, at least, in the beginning. Furthermore, they have been always willing to solve all my doubts, whatever it is.

I also want to give thanks to the University of Cartagena, as WPAN and IoT interested me after being accepted as an intern. I will be always grateful for having enjoyed it, as I would have never thought that my first work experience would be offered by the University. I would say this experience has been one of the best experience as an engineer that I have ever had.

And I am going to give thanks to EMITE, the whole team, for being part in this project. Some of the work I have developed depended on resources and knowledge, which I had not. However, I could count on their help whenever I needed it. What is more, this investigation has given me the opportunity to carry out an investigation in practice, and so I have learnt how companies really work.

Finally, I must not forget to give thanks to the person who has been on my side in every singular fight that I have had, and whose name is Óscar Ortega. Both he and me wanted to add a victory every day.

Index

1.	INTRODUCTION.....	14
1.1.	INTERNET OF THINGS	14
1.1.1.	<i>Personal Area Networks (WPAN)</i>	15
1.1.2.	<i>Different technologies for IoT</i>	16
1.2.	UTILIZED HARDWARE	20
1.2.1.	<i>To create a LoRa network</i>	20
1.2.2.	<i>To create a BLE WPAN</i>	23
1.3.	UTILIZED SOFTWARE	26
1.4.	DESCRIPTION OF THIS PROJECT	28
1.4.1.	<i>Purpose</i>	28
1.4.2.	<i>Status of the issue</i>	28
1.4.3.	<i>Objectives</i>	29
1.4.4.	<i>Structure of this document</i>	29
2.	LORA.....	31
2.1.	LORA PARAMETERS.....	31
2.1.1.	<i>Spreading Factor</i>	31
2.1.2.	<i>Data Rate</i>	31
2.1.3.	<i>Time on Air</i>	32
2.1.4.	<i>Adaptive Data Rate (ADR)</i>	32
2.2.	THE THINGS NETWORK.....	33
2.2.1.	<i>Registering a gateway</i>	33
2.2.2.	<i>Creating an Application</i>	34
2.2.3.	<i>Registering a Device</i>	35
2.2.4.	<i>Payload format</i>	36
2.3.	LORAWAN.....	37
2.3.1.	<i>OTAA Method</i>	39
2.3.2.	<i>ABP Method</i>	40
2.4.	KERLINK LORA IOT STATION AS A PART OF A LORA NETWORK	40
2.4.1.	<i>Updating Firmware</i>	41
2.4.2.	<i>Packet Forwarder</i>	42
2.5.	RN2483 MOTE AS A PART OF A LORA NETWORK	43
2.6.	TESTING WITH E500 REVERBERATION CHAMBER.....	45
2.6.1.	<i>DUT inside of the chamber</i>	45
2.6.2.	<i>LoRa gateway inside of the chamber</i>	47
3.	BLE.....	49
3.1.	DEFINITION	49

3.1.1.	<i>BLE vs classic bluetooth</i>	49
3.1.2.	<i>Main features</i>	49
3.2.	6LoWPAN.....	50
3.3.	ASUS RT-N16 AS A PART OF A BLE NETWORK.....	52
3.3.1.	<i>Installing OpenWRT</i>	52
3.3.2.	<i>Installing all required packages for both USB Bluetooth support and 6LoWPAN devices support on OpenWRT</i>	53
3.3.3.	<i>Connecting RT-N16 router with a BLE device (with 6LoWPAN)</i>	53
3.4.	RASPBERRY PI AS A PART OF A BLE NETWORK	55
3.4.1.	<i>Spice up Raspbian for the IoT</i>	55
3.4.2.	<i>Raspberry Pi as an end-device</i>	58
3.4.3.	<i>Setup a native 6LoWPAN router</i>	60
3.4.4.	<i>Raspberry as a router</i>	62
4.	RESULTS	65
4.1.	LORA	65
4.1.1.	<i>First test to clarify</i>	65
4.1.2.	<i>Testing in a RC</i>	66
4.2.	IPV6 OVER BLE.....	72
4.2.1.	<i>Raspberry as an end-device</i>	72
4.2.2.	<i>Raspberry as a router</i>	75
5.	CONCLUSIONS.....	79
5.1.	LORA TECHNOLOGY	79
5.2.	IPV6 OVER BLE.....	79
6.	FUTURE LINES OF INVESTIGATION	81
6.1.	LORA	81
6.2.	BLE WITH IP.....	81
7.	BIBLIOGRAPHY	83
ANNEX A: DIFFERENT ENVIRONMENTS TO TEST A DEVICE		85
A.1	<i>Anechoic Chamber (AC)</i>	85
A.2	<i>Reverberation Chamber (RC)</i>	89
A.3	<i>Advantages and disadvantages of anechoic and reverberation chamber</i>	91
ANNEX B: UTILIZED REVERBERATION CHAMBER		96
ANNEX C: HOW TO CALCULATE OBTAINED THROUGHPUT		98
ANNEX D: CONNECTING RASPBERRY PI WITH A WIRELESS NETWORK.....		99

Figure Index

Figure 1.1 Summarizing IoT	15
Figure 1.2 Example of a WPAN.....	16
Figure 1.3 Example of a LoRa application.....	17
Figure 1.4 SigFox	18
Figure 1.5 ZigBee.....	19
Figure 1.6 Kerlink Wirnet LoRa IoT Station	20
Figure 1.7 LoRa RN2483	21
Figure 1.8 Wirgrid Debug Board.....	23
Figure 1.9 Raspberry Pi 2B	23
Figure 1.10 Laird DVK-BT850	24
Figure 1.11 Router RT-N16.....	24
Figure 1.12 CSR 4.0 Bluetooth Dongle	25
Figure 1.13 Running packet forwarder	27
Figure 1.14 EMITE software.....	27
Figure 2.1 Distances covered by LoRa depending on DR and SF	32
Figure 2.2 LoRa gateway in TTN.....	34
Figure 2.3 Application parameters in TTN	35
Figure 2.4 Device parameters in TTN	36
Figure 2.5 Luminosity and temperature given by RN2483	37
Figure 2.6 LoraWAN from LoRa	38
Figure 2.7 LoRaWAN network.....	38
Figure 2.8 LoRaWAN Over-The-Air Activation	39
Figure 2.9 LoRaWAN Activation-By-Personalization.....	40
Figure 2.10 Updated firmware version	41
Figure 2.11 Gateway ID from loraboard_conf.json	42
Figure 2.12 Global_conf.json configuration	43
Figure 2.13 OTAA parameters in the server.....	44
Figure 2.14 ABP parameters in the server	45
Figure 2.15 Example of configuration to use Remote Utilities	46
Figure 3.1 Channels and frequencies used by BLE	50
Figure 3.2 6LoWPAN architecture.....	51
Figure 3.3 List with all connected BLE devices	54
Figure 3.4 Channel and PAN ID	56
Figure 3.5 Constrained Application Protocol	57
Figure 3.6 Topology to implement a BLE connection (I).....	58
Figure 3.7 IPv6 over BLE connection process (I)	59
Figure 3.8 Fast Lexical Analyser Generator	60

Figure 3.9 Topology to implement a BLE connection (II).....	62
Figure 3.10 BLE modules controlled by raspberry.....	63
Figure 3.11 IPv6 over BLE connection process (II)	64
Figure 4. 1 Map for LoRa measurements at ELDI	65
Figure 4. 2 Inside of a RC	68
Figure 4.3 Attenuation depending on distance	70
Figure 4.4 Obtained throughput with the DUT inside of the RC.....	71
Figure 4.5 Obtained throughput with the LoRa gateway inside of the RC	72
Figure 4.6 bt0 interface and ping6.....	73
Figure 4.7 RT-N16 console when using Iperf3 (I).....	74
Figure 4. 8 Raspberry console when using Iperf3 (I).....	74
Figure 4.9 bt0 interface in raspberry.....	75
Figure 4.10 bt0 interface in RT-N16	75
Figure 4.11 ping6 from the Laird to Dongle CSR 4.0 Dongle	76
Figure 4.12 RT-N16 console when using Iperf3 (II).....	77
Figure 4.13 Raspberry console when using Iperf3 (II).....	77
Figure A.1 View of an Anechoic Chamber with pyramidal absorber	86
Figure A.2 Locally plane wave front	87
Figure A.3 Phase difference between a spherical wave and a plane one	88
Figure A.4 Inside of a reverberation chamber for measurements.....	90
Figure A.5 Measuring TRP with an AC based on EIRP measurements	93
Figure A.6 Measuring TIS with an AC based on EIS measurements	95
Figure B.1 E500 Reverberation Chamber	96
Figure B.2 Efficiency(%) depending on frequency for AC and RC	97
Figure C.1 Example of obtained throughput.....	98
Figure D.1 Creating a wireless network in windows	99

Tables index

Table 1-1 SNR depending on SF	21
Table 1-2 EU863-870 TX power	22
Table 1-3 EU863-870 TX Data Rate	22
Table 3-1 Comparison of BR, EDR and BLE.....	49
Table 4-1 LoRa measurements in ELDI	66
Table 4-2 Modes for testing in a RC.....	67
Table 4-3 Obtained data for mode 6 (DUT inside of RC)	68

Abbreviations

3GPP	3 rd Generation Partnership Project
ABP	Activation-By-Personalization
AC	Anechoic Chamber
API	Application Programming Interface
AUT	Antenna Under Test
BLE	Bluetooth Low Energy
BR	Basic Rate
BW	Bandwidth
CATR	Compact Antenna Test Range
CTIA	Cellular Telecommunication Industry Association
DFF	Direct Far Field
DR	Data Rate
DUT	Device Under Test
EDR	Enhanced Data Rate
EIRP	Effective Isotropic Radiated Power
EIS	Effective Isotropic Sensitivity
EUI	End-device Unique Identifier
FLEX	Fast Lexical Analyzer Generator
IFF	Indirect Far Field
IoT	Internet of Things
LoRa	Long Range
LoWPAN	Low-Power WPAN
M2M	Machine-to-Machine
MIMO	Multiple-Input Multiple Output
OTA	Over-The-Air
OTAA	Over-The-Air Activation
PDP	Power Delay Profile
PL	Path Loss
QZ	Quiet Zone
RADVD	Router Advertisement Daemon
RC	Reverberation Chamber
RF	Radio Frequency
RFU	Reserved for Future Use
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SNR	Signal-to-Noise Ratio
TIS	Total Isotropic Sensitivity

TPUT	MIMO Throughput
TRP	Total Radiated Power
TTN	The Things Network
WPAN	Wireless Personal Area Networks

1.Introduction

1.1.Internet of Things

The Internet of Things (IoT) shall be able to incorporate transparently and seamlessly a large number of different and heterogeneous end systems, while providing open access to selected subsets of data for the development of a plethora of digital services. Building a general architecture for the IoT is hence a very complex task, mainly because of the extremely large variety of devices, link layer technologies, and services that may be involved in such a system [7].

Fuelled by the recent adaptation of a variety of enabling wireless technologies such as RFID tags and embedded sensor and actuator nodes, the IoT has stepped out of its infancy and is the next revolutionary technology in transforming the Internet into a fully integrated Future Internet. As one can easily imagine, any serious contribution to the advance of the Internet of Things must necessarily be the result of synergetic activities conducted in different fields of knowledge, such as telecommunications, informatics, electronics and social science.

It is not difficult to see that there are more devices connected each other over time, as a bigger amount of information is being transmitted continuously. And we can see how this fact is coming true in many sectors, such as healthcare or transportation. What IoT offers is a way to “think” and perform jobs as it let us to share as much as information as we want. To do that, many underlying technologies have been needed e.g. sensor networks, embedded devices and Internet protocol, among other things.

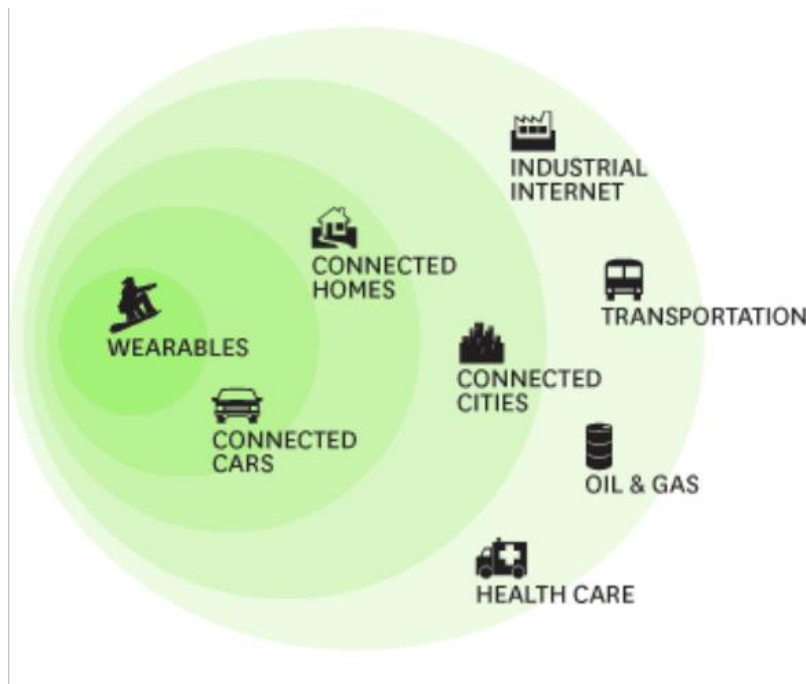


Figure 1.1 Summarizing IoT

Therefore, IoT is creating new business applications and so improving both our quality life and world's economy. Then it is necessary to generate new protocols for communication compatibility between every device that are part of IoT. What is more, actual Internet architecture will have to be redefined in order to solve the IoT challenges, such as the incredibly large amount of devices that will be connected. Due to that, using a large addressing space such as IPv6 seems one of the possible solutions.

1.1.1.1. Personal Area Networks (WPAN)

One of the most important things when speaking about IoT is WPAN. We could define it as a network made by different IoT devices in order to accomplish certain requirements i.e. Imagine you want to know the humidity level from five different locations. You can install a humidity sensor on each location and so a gateway, which will receive the information sent by the sensors and forwarder it to a server. Therefore, you will be able to know the humidity level if and when your mobile has Internet access. Then, the sensors and the gateway are creating a WPAN.

Depending on the features of the WPAN, that is to say, the distances between gateway and sensors, amount of data to transmit, required security level to implement the network, etc. one or another technology will be used. It is important to keep in mind that a WPAN may be either extremely large (i.e. to be made by hundreds and hundreds of devices) or very little (e.g. one sensor and one gateway).

In other words, what defines a network as a WPAN is not its size but the type of devices that are creating the network. As an example, next picture below is shown in order to get it across.

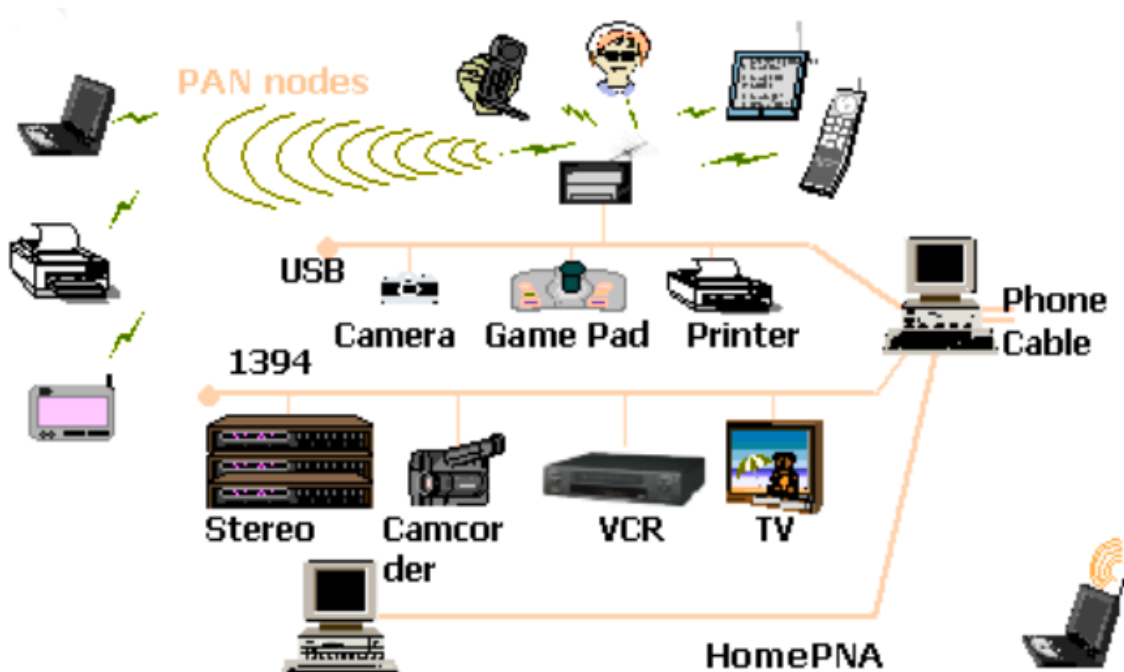


Figure 1.2 Example of a WPAN

As can be seen, many devices are connected each other so that most user requirements are satisfied. For instance, nowadays we can control our TV with our mobile thanks to Wi-Fi technology. Other example would be we can know how many steps we walk every day looking at our mobile (or our smartwatch), and BLE is what make it possible.

1.1.2. Different technologies for IoT

As it was said before, depending on the desired features, one or another technology will be used. Due to that, this subsection is made to get it across, as different IoT technologies will be explained. Then, advantages and limitations of each technology will be known just as their main features (frequencies, use cases, etc.).

🚦 LoRa (Long Range)

This technology uses an expanded spectrum modulation. It is a new wireless protocol, whose objective is long-range and low-power networks. It is quite useful for M2M and IoT as low-power is its most important feature.

One of the some advantages that LoRa offers is its modulation, that is, a radio frequency modulation, which lets various users to be connected with the same network, even though they are using different applications.

Its long-range and low-power features make LoRa is used for many applications possible. For instance, LoRa may be used to intelligent detection in civil infrastructures, intelligent measurements and so on. Furthermore, LoRa has an integrated AES-128 coded mechanism from end-device to end-device.

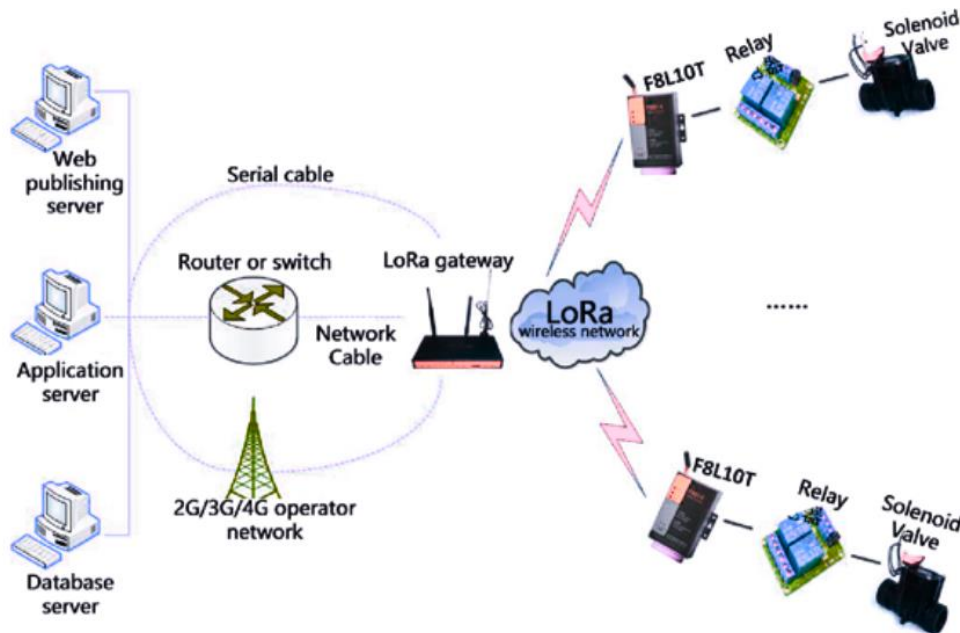


Figure 1.3 Example of a LoRa application

🚀 SigFox

In the IoT, the challenge lies in connecting multiple devices across distributed locations in a cost- and energy-efficient manner. Devices are expected to run on a single coin-cell for years without manual intervention.

This type of technology has achieved that the volume of data transmitted in M2M is much less than in either a voice or video call. This is where communication networks like Sigfox stands out; the transceiver sleeps most of the time and works when transmitting data. Thus, it saves a large amount of power.”

SigFox can be used in a multitude of applications. Globally, Smart Cities are readily adopting this technology. Hence, SigFox is being greatly used in smart waste management i.e. fire-hydrant control, streetlight control, air-quality monitoring and real-time water-level updates [9].

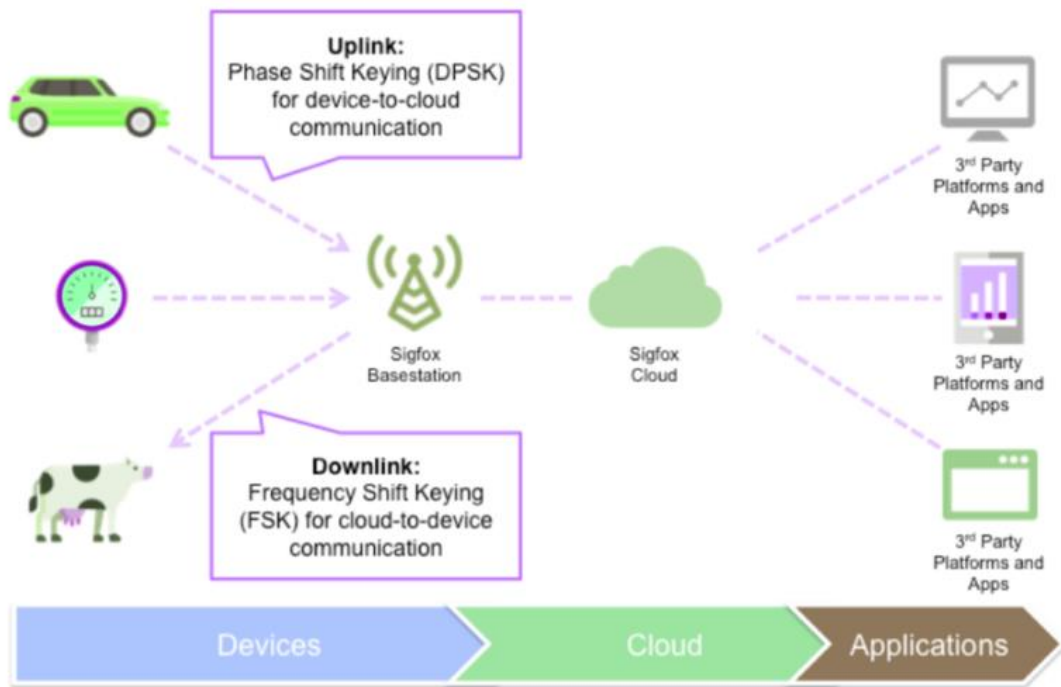


Figure 1.4 SigFox

Bluetooth Low Energy

Assigned 802.15.1 standard by IEEE, Bluetooth is used in various personal and commercial devices. Bluetooth Low-Energy (BLE), which launched in 2009, sets the stage for future applications in IoT as the field was taking off. BLE is a specification, whose objective is for small-scale IoT applications that require devices to send small amounts of data using the minimum possible amount of energy.

Increasing the modulation index and throttling load-intensive (and thus power-intensive) data types has enabled Bluetooth to lower power consumption by 95 to 99 percent (depending on the use case). Upgrading message encryption to 128-bit AES-CCM (government grade) was the final prerequisite to pivoting the BLE topology away from salesmen on their Bluetooth Classic earbuds and toward swarms of low-power nodes relaying simple messages across warehouses and smart homes alike 83[8].

ZigBee

ZigBee is a flexible RF-based communication technology. Mostly used for industrial applications, it has been assigned standard 802.15.4 by IEEE. ZigBee works in the same bandwidth as Wi-Fi, which is 2.4GHz, but consumes less power and offers a high level of security through 128-bit encryption. However, it enables limited data exchange.

ZigBee-based sensor networks are widely used in multilevel parking monitoring systems, warehouse measurement and control systems, and various

environmental monitoring systems. Sensors are spread over the area to transmit information, and the products required to communicate are tagged in the range for flawless data travel [9]. Figure below shows it to get it across.

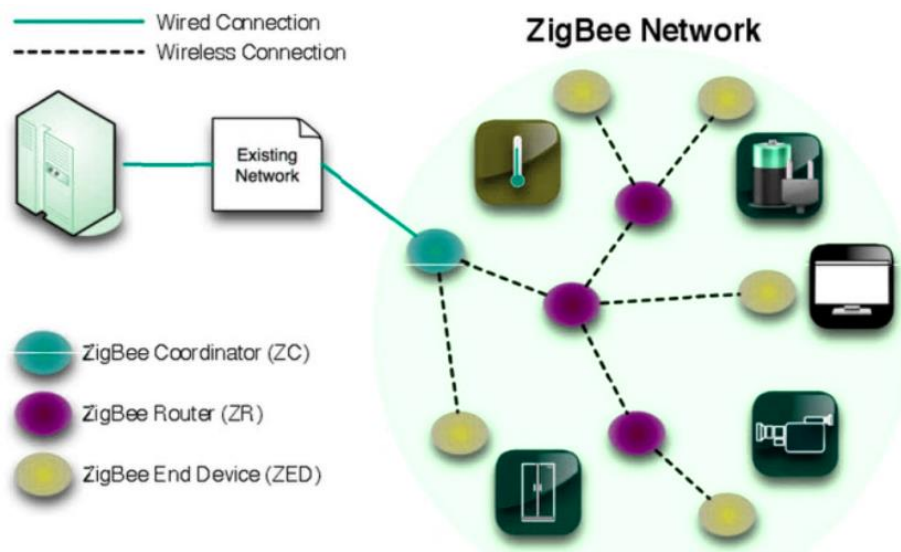


Figure 1.5 ZigBee

WiFi

This is one of the most commonly used communication standard, assigned as 802.11 by the Institute of Electrical and Electronics Engineers (IEEE). It operates in the 2.4-5GHz band. With the rollout of 5G network, Wi-Fi will play an even bigger role in wireless transmission.

Wi-Fi technology is promoted by the not-for-profit Wi-Fi Alliance. The alliance certifies products with Wi-Fi interoperability. Wi-Fi Alliance also owns the Wi-Fi trademark. However, it should not be the criteria to choose a product. Many products support Wi-Fi technology but are not certified due to the cost involved. Certification only increases the chances of interoperability [9].

1.2. Utilized Hardware

1.2.1. To create a LoRa network

Kerlink 868 IoT LoRa Station



Figure 1.6 Kerlink Wirnet LoRa IoT Station

The Wirnet Station is a robust, performant and highly reliable outdoor LoRaWAN gateway for smart IoT network. Introduced in 2014, the Wirnet Station was the first LoRaWAN gateway commercially available on the market worldwide.

Based on LoRa technology, this gateway is since a leading reference for the deployment of resilient public and private operated networks. It is available in most global unlicensed frequency bands (like ISM ones) and is proposed through ruggedized, compact and light casing, making it easy to install and deploy [4].

The main features of kerlink 868 IoT Station are:

- Sensitivity: up to -141 dBm
- Transmitted conducted power: from 0 dBm to +27 dBm
- 49 LoRa demodulators over 9 channels
- More than 15 Km range in sub-urban situation
- More than 2 Km range in urban situation

Despite the maximum value of the Station transmitted conducted power, the power value must be adjust depending on the country. The maximum Tx power allowed for EU868 (Europe, frequency: 868 MHz) is 14 dBm.

In addition, the maximum antenna gain is 2.5 dBi (or a little bit more if you take some loss in account)

If we choose a spreading factor of 7, a bandwidth of 125 kHz and a 868 MHz frequency, the sensitivity value will be -129 dBm (with 10% of PER and 20 bytes payload). However, the station is able to receive from -20 dBm to -141 dBm (it depends on the SF and BW). The station will assign to each received frame a RSSI and SNR value.

RSSI varies from -20 dBm to -120 dBm. -120 dBm is the measured noise floor with a 200 kHz BW. On the other hand, SNR is between 10 to 15 dB for strong

signals, and it is close to 0 dB when the signal strength approaches – 120 dBm. In addition, it can decrease to -7 or -20 dB depending on the SF:

Table 1-1 SNR depending on SF

Spreading Factor	LoRa Demodulator SNR (dB)
SF7	-7.5
SF8	-10
SF9	-12.5
SF10	-15
SF11	-17.5
SF12	-20

RN2483 Mote



Figure 1.7 LoRa RN2483

The RN2483 is a fully-certified 433/868 MHz module based on wireless LoRa technology. The RN2483 utilizes a unique spread spectrum modulation within the Sub-GHz band to enable long range, low-power, and high network capacity.

The module's embedded LoRaWAN Class A protocol enables seamless connectivity to any LoRaWAN compliant network infrastructure, whether public or privately deployed. The module is specifically designed for ease of use, which shortens development time and speeds time to market. LoRa technology is ideal for battery-operated sensors and low-power applications such as IoT, M2M, Smart City, Sensor networks, Industrial automation, among others [5].

This device measures both temperature and humidity and so these two parameters will be sent to the kerlink IoT Station when testing.

If we want to know some features of RN2483 such as transmission power, it is necessary to have a look to the *LoRaWAN Specification reference*, concretely, *LoRa Alliance Regional Parameters*. There, we will found the next relationship between the $\langle pwrIndex \rangle$ (that is a number we introduce to indicate transmission power when sending a packet) and the real value.

Table 1-2 EU863-870 TX power

Tx Power	Configuration (EIRP)
0	Max EIRP
1	Max EIRP – 2dB
2	Max EIRP – 4dB
3	Max EIRP – 6dB
4	Max EIRP – 8dB
5	Max EIRP – 10dB
6	Max EIRP – 12dB
7	Max EIRP – 14dB
8...14	RFU
15	Defined in LoRaWAN

NOTE: By default, Max EIRP is considered to be +16 dBm.

The other two most important parameters to configure when sending a packet are Data Rate (DR) and Spreading Factor (SF). To know the connection between the DR value and the spreading factor, we have to take a look at *LoRaWAN Specification reference*, concretely, *LoRa Alliance Regional Parameters*:

Table 1-3 EU863-870 TX Data Rate

Data Rate	Configuration	Indicative physical bit rate (bit/s)
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8...14	RFU	
15	Defined in LoRaWAN	

Wirgrid Debug Board

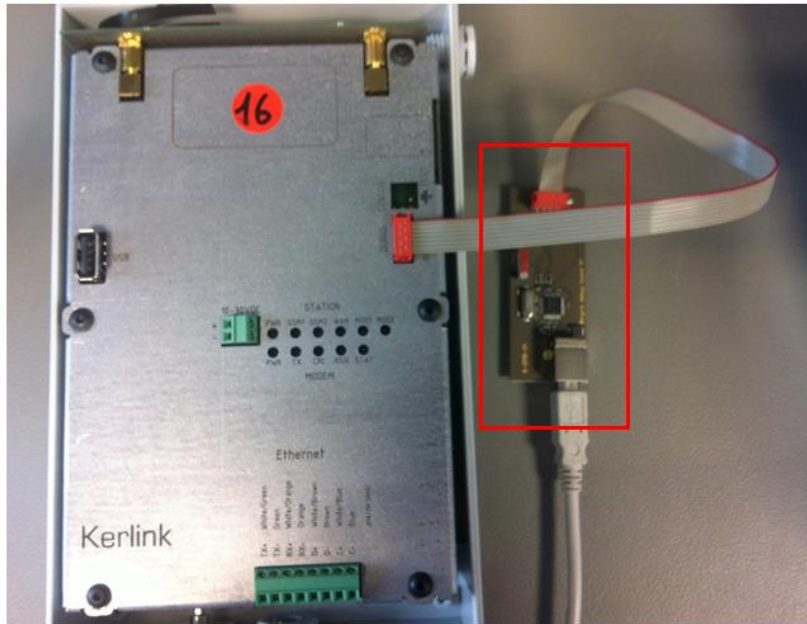


Figure 1.8 Wirgrid Debug Board

This hardware is necessary as we need to send commands to the LoRa IoT Station. There is an Ethernet cable that is connecting the control PC and the station, but this cable is what the station needs to access internet and so to control the station through this Ethernet connection is not possible. Hence, we need another connection between the station and the control PC, and wirgrid debug board provides that: a USB connection to control the station from a control PC.

1.2.2. To create a BLE WPAN

Raspberry Pi 2B



Figure 1.9 Raspberry Pi 2B

The Raspberry Pi Model B+ incorporates a number of enhancements and new features. Improved power consumption, increased connectivity and greater IO are among the improvements to this powerful, small and lightweight ARM based computer.

Laird DVK-BT850

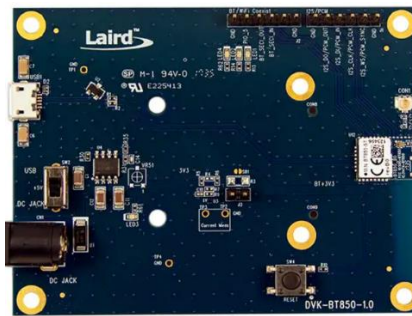


Figure 1.10 Laird DVK-BT850

This device has a DVK-BT850-ST as a Bluetooth module. The BT85x series of USB HCI modules and Adapter leverage the Cypress CYW20704 A2 chipset to provide exceptionally low-power consumption with outstanding range for OEMs needing both Classic Bluetooth and Bluetooth Low Energy support. The Bluetooth v4.2 core specification shortens your development time and provides enhanced throughput, security and privacy.

The BT850 modules are ideal when designers need both performance and minimum size. For maximum flexibility in integration, they support a host USB interface, I 2S and PCM audio interfaces, GPIO, and Cypress'GCI coexistence (2-Wire). The modules provide excellent RF performance and identical footprint options for integrated antenna or an external antenna via a trace pin.

Asus RT-N16



Figure 1.11 Router RT-N16

ASUS RT-N16 Wi-Fi Router-N 300 provides a strong Gigabit Ethernet Online connectivity, and a Wi-Fi network. In fact, it has a few enhanced options including a built-in bit torrent client and a print server but we will focus on the VPN (Virtual Private Network) credentials of the router.

Before installing the ASUS Wireless Router, ensure that your system/network meets the following requirements:

- An Ethernet RJ-45 port
- A least one IEEE 802.11b/g/n device with wireless capability
- An installed TCP/IP and Internet browser

CSR 4.0 Dongle



Figure 1.12 CSR 4.0 Bluetooth Dongle

This adapter creates cable-free connections between PCs and other Bluetooth devices such as Bluetooth enabled headsets, speakers, cell phones, keyboard or mouse and more (with Microsoft built-in drivers) at speeds of up to 3 Mbps.

Its Bluetooth version is 4.0. Backward compatible with legacy Bluetooth equipment Bluetooth 3.0, 2.1, 2.0 and 1.0 Equipment. Improves Bluetooth support for Windows 10, 8.1/8, 7, Vista, XP, 2003, 2000, Me, 32 bit and 64 bit.

Supports Bluetooth stereo headsets and more with built-in Windows 10 and 8 drivers or using the included drivers for Windows 7 and earlier. Specifically works with Microsoft's new Bluetooth 4.0 Low Energy support in Windows 8 and later. However, it does not add Bluetooth Low Energy.

1.3. Utilized Software

Linux

Nowadays, Linux is one of the most used (or the most, directly) operative system. When controlling a device is desired, it is the best solution to do it and perhaps it is also the only one. For this project, this software was quite necessary, as we worked with many devices whose operative system was Linux (or a distribution thereof). In fact, various Linux distribution were used.

- *OpenWrt*

OpenWrt is a firmware based on a Linux distribution, and it is very used in personal routers. Therefore, it is a distribution with certain limitations such as a limited amount of packets. In this project, we had to use it when configuring the RT-N16 router, as its operative system is OpenWrt.

- Raspbian

It is one of the most famous Linux distribution based on Debian, that is to say, is free for Raspberry Pi. We needed it as we worked with a raspberry Pi 2b and so we found out the large amount of advantages that raspbian offers.

Putty

This software is well-known for all of us. Putty is a SSH client with a free license. Due to that it was perfect for us as we had to access as a client when controlling a device was desired. Concretely, we used it to control every device but Mote RN2483, although using Putty was not impossible.

Tera Term

It is an open source and terminal emulator program. It was very useful as it supports SSH and serial port connections, which we needed to control mote RN2483. We chose it instead of Putty as controlling that mote was easier with Tera Term due to its more personalized configuration.

Packet forwarder

A LoRa packet forwarder is a program running on the host of a LoRa gateway that forwards RF packets receive by the concentrator to a server through a IP/UDP link, and emits RF packets that are sent by the server.

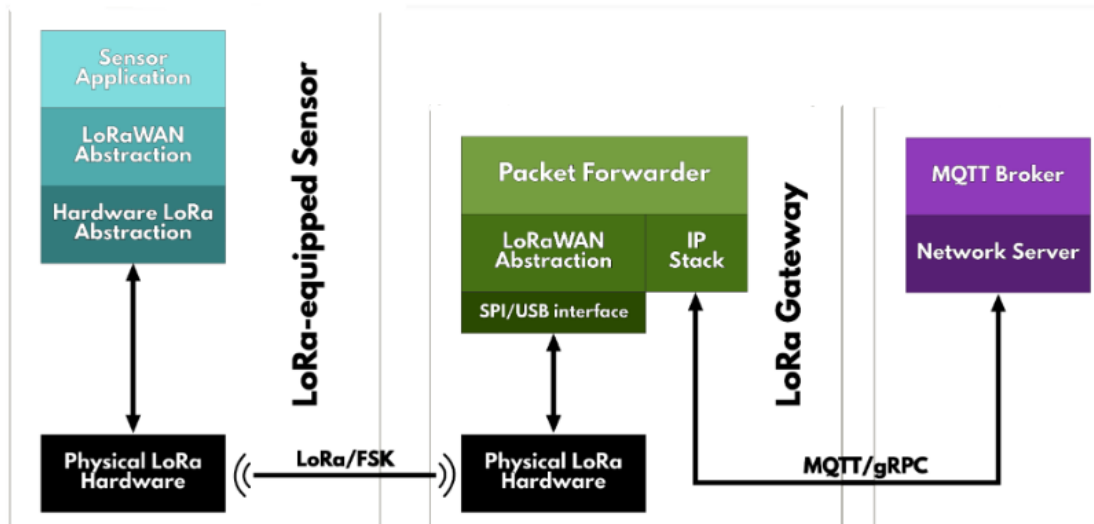


Figure 1.13 Running packet forwarder

🚦 MIMO Analyser GUI

The graphic user interface reference guide briefly explains the function of all input and output elements in the software interface, and it is based on C++ high level programming language. One of the most important factors for the implementation of calibration and final use of a mode stirred source-stirred reverberation chamber (MSRC) for MIMO measurements is the control software [6].

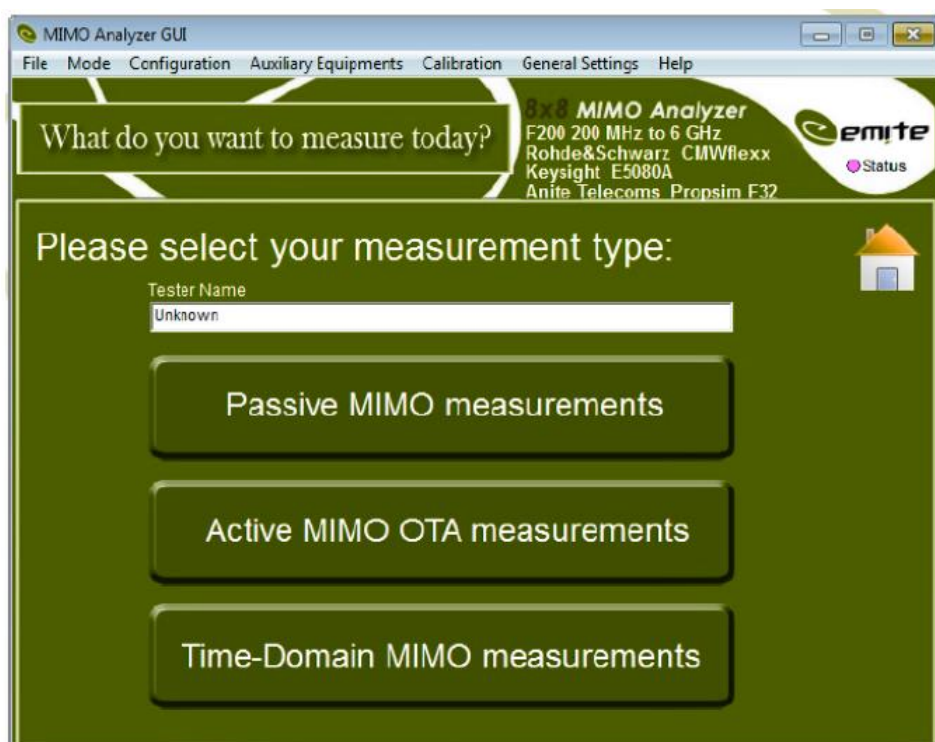


Figure 1.14 EMITE software

1.4. Description of this project

1.4.1. Purpose

The main purpose of this project is to demonstrate that low-power networks may be made using not expensive devices. Once that the different networks are already created, it is possible to measure their throughput, even though here the main objective is to create a network. Hence, we chose two technologies that their boom has come true due to IoT, and these technologies are LoRa and BLE. Both LoRa and BLE stands out because of they are extremely low-power.

On one hand, we created a LoRa network, in other words, a LoRa WAN, which offers a very easy way to connect various devices with a gateway, even a large distance, and what is more, let us know what information is being sent by these devices if and when we and the created networks have internet access.

On the other hand, we researched on how to create a network based on BLE devices, meaning that these devices must be a IP address. In our case, we used and IPv6 address. What this part gives a way to communicate two BLE devices each other without any necessity they to be connected by BLE each other, in other words, they have only to be part of the same network.

1.4.2. Status of the issue

Nowadays, all of us know about IoT and its importance. Many companies are actually investing much money and time in the technologies that will be part of IoT, such as LoRa and BLE. The main purpose of these technologies is to provide solutions to transmit all type of information, independent of circumstances. In other words, these technologies can either communicate devices that are far away each other or transmit information using a large bandwidth.

Therefore, when mixing these technologies with Internet we can take advantages of both these technologies and Internet i.e. mixing LoRa and Internet we can communicate LoRa devices (e.g. gateway LoRa IoT Station and RN2483 mote) even a large distances. We can also know the obtained temperature and humidity (as they are the parameters sent by RN2483) by consulting servers on Internet that gateway forwarders the information. These advantages will happen if and when we and the gateway have access to Internet.

This fact let us know why mixing different technologies we can obtain the best part of them. Therefore, it is also necessary to be able to test these technologies, meaning to measure their throughput. As we was working with both LoRa and BLE, we must clarify that two different methods were required for testing.

To measure LoRa devices throughput, we used a RC, among other things. Concretely, E500 was the chosen RC due to its large amount of frequencies for testing. We also needed to create some bash codes, as the generated

information by the kerlink IoT Station was saved in a txt file. Those bash codes made to work out the throughput possible.

The situation was quite different to measure BLE throughput. In this case, installing a considerable amount of software on the devices just as configuring a huge amount of parameters was required. In fact, a software was required to measure throughput, whose name is *iperf3*.

1.4.3. Objectives

The main objective of this project is to analyse some of the most important technologies that will be necessary to create IoT, which are LoRa and BLE. Therefore, the first thing we have to keep in mind is the need for understanding how these technologies works: frequency and protocols, among other things, as we want to know their advantages and limitations.

Then, we will try to create one network with every technology. Therefore we must work with some devices in order to each device does its function as a part of a network. Having said that we could say that other objective would be to 'spice up' all the used devices for creating that network.

Finally, the final step is to measure their throughput. Different methodologies have to be applied depending on the tested technology. Therefore, another objective is to know how to test a device in a RC and so, using a specified software and chamber. Despite of the created network works properly, we must verify that every device also works correctly. The best way to achieve that is testing them and so we need a RC for testing adequately.

1.4.4. Structure of this document

This document has been divided as follows:

- First section is the introduction, which explains the main concepts we are going to work with just as shows the used tools. It also takes a look at the very recent history to get the importance of this project across.
- Second section explains all we need to know about one of the studied technologies: LoRa. The main objective of this section is to clarify how LoRa works, from security protocols to work frequencies, limitations and so on. This section also explains all the done work step by step.
- Third section is about BLE, the other studied technology. As second section, the purpose is to understand better what BLE is for a better knowledge about its advantages and limitations. In addition, all the done work will be also explained in this section step by step.
- Fourth section will show the obtained results once that all devices to be well-configured. In other words, in this section we can see the obtained throughput with both Lora and BLE.

- Fifth section is made to explain the meaning of the results. For instance, we can predict the distance that could be covered by a technology when knowing throughput, transmitted power, sensibility and evaluation distance. This section will also show why these technologies are used for different applications.
- Sixth section enumerates the amount of things that are not yet studied about these technologies. It is important to realize that these technologies are having their boom thanks to IoT and so it is necessary to study them as much as possible, in order to be able to take advantage of them in the best possible way.
- Seventh section enumerates all the information sources that were required to carry out this project. Many of them are books and user's guides. However, others sources are links, as an important amount of information was reached by Internet, the most useful tool now.
- Finally, some annexes have been written to clarify some concepts about testing, the used RC, etc. It is recommendable to have a look at these annexes if you do not become familiar with some concepts that are being used in this project.

2.LoRa

LoRa (Long Range) is a spread spectrum modulation technique derived from chirp spread spectrum technology. This technology is patented by Semtech.

LoRa technology is a long range, low-power wireless platform that has a great importance in Internet of Things (IoT) networks worldwide in smart cities. The main advantages of LoRa technology are high interferences resistance, high sensitivity (-168 dB), based on Chirp modulation, low-power consumption (until 10 years with a battery), long-range (from 10 to 20 km), low data transfer (until 255 bytes), point-to-point connection

Its supported ISM Bands are:

- 915 MHz in America
- 868 MHz in Europe
- 433MHz in Asia

2.1.LoRa parameters

2.1.1. Spreading Factor

This parameter is the relation between chip rate and symbol rate. The basic principle of spread spectrum is that each bit of information is encoded as multiple chirps. For LoRa modulation may differ between spreading factor (SF) 7 to 12. The end-device may transmit on any channel available at any time, using any available data rate, depending on the following rules:

- The end-device changes channel in a pseudo-random fashion for every transmission. The resulting frequency diversity makes the system more robust to interferences.
- The end-device respects the maximum transmit duty cycle relative to the sub-band used and local regulations.
- The end-device respects the maximum transmit duration (or dwell time) relative to the sub-band used and local regulations.

Higher SF means lower data rate, higher sensitivity and longer range. Lower SF means higher data rate, lower sensitivity and shorter range.

2.1.2. Data Rate

Communication between end-devices and gateways is spread out over different frequency channels and data rates. The selection of the data rate is a trade-off between communication range and message duration. Within the selected

channel the LoRa protocol, which is a chirp spread spectrum modulation technique, determines how many bits are required to code the data (coding rate) which results in a maximum data rate.

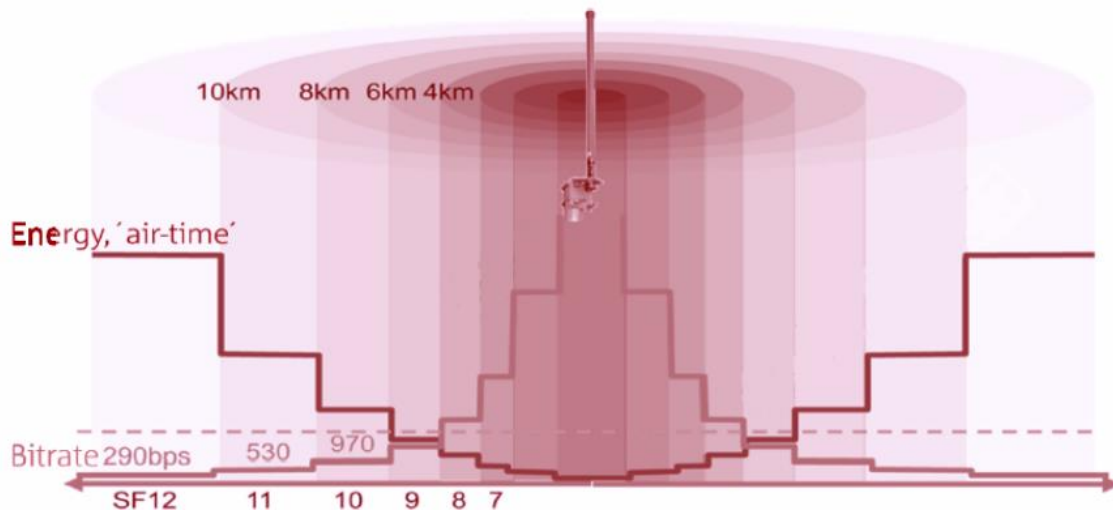


Figure 2.1 Distances covered by LoRa depending on DR and SF

2.1.3. Time on Air

An important consequence of using a higher spreading factor for LoRa is a longer Time-on-Air (ToA). The LoRa Radio module needs more time to send the same amount of data. This means that power consumption increases with increasing Spreading Factor.

2.1.4. Adaptive Data Rate (ADR)

LoRa data rates range from 0.3 kbps to 50 kbps. Depending on the environmental conditions between the communication device and the gateway, in our case Kerlink IoT Station, the network will determine the best spreading factor (SF) to work on. It is recommended using a SF with value of 7 and a Data Rate of 5 if a high speed is required.

To maximize both battery life, range and overall network capacity, the LoRa network infrastructure can manage the data rate and output power used for the communication for each end-device individually by means of an adaptive data rate (ADR) scheme. Meaning the better the coverage the lower the SF will be (see the figure below). Using an ADR functionality is requested by the device, not by the network. Switching ADR OFF is only advised for continually moving objects.

2.2. The Things Network

The Things Network (TTN) is a global LoRaWAN public network kernel based on crowd-source infrastructure. Nowadays, hundreds of users, who work with LoRaWAN networks, use this platform due to it is quite easy to use. In addition, thanks to it we can know some properties of the data sent, such as RSSI, SNR, data rate, airtime, frequency, spreading factor and the payload, for example.

2.2.1. Registering a gateway

If we want to see each and every one of the packets that our IoT Station is receiving, we have to create an account in TTN (<https://www.thethingsnetwork.org/>), as **¡Error! No se encuentra el origen de la referencia.** explains.

The first step is to create a Gateway in the Console. To do that, following steps must be done:

1. Click on the account name and then, click on Console.
2. Click on Gateway and register gateway. It will appear a menu like that:
3. Fill in the gaps with a suitable information. The Gateway ID is, perhaps, the most important parameter we use. The ID, which has been obtained in the *loraboard_conf.json* file, must be the same that is written in *global_conf.json*. Then, "I'm using the legacy packet forwarder" box has to be selected (because our gateway is a Kerlink IoT Station and this type of gateway needs this function).
4. Rest of parameters are as follows:
 - description is optional
 - frequency plan depends on the region where our gateway is. For instance, in our case, we choose "Europe 868 MHz"
 - router: it will appear after the frequency plan selection
 - location: the place where the gateway should be already marked in the map
 - indoor or outdoor (kerlink gateways must be outdoor)
 - once you have filled all the gaps, click on Register Gateway

Now, the Station is registered in The Things Network Map and therefore we can see it in that map.

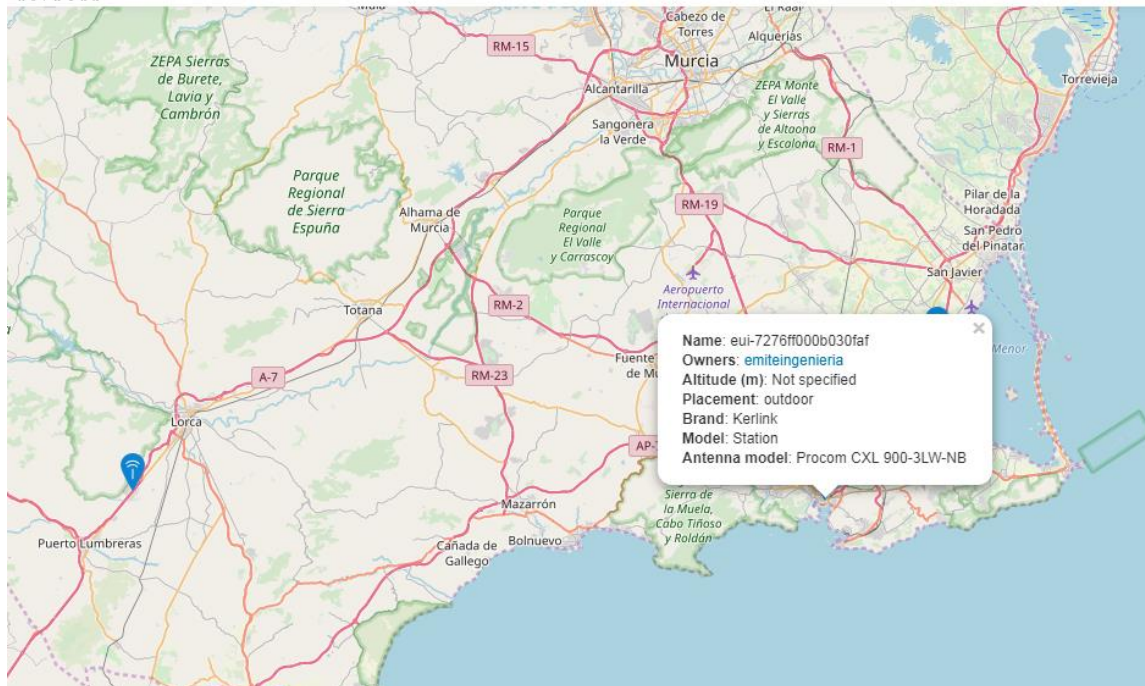


Figure 2.2 LoRa gateway in TTN

2.2.2. Creating an Application

The second necessary item is an Application, which defines the purpose that we want to reach. In our case, to measure the RSSI power and throughput of a DUT. Then, the used DUT will have to be register inside the application.

To create an Application is as follows:

1. In the account panel, click on *applications* and then, click on "add application". It will appear a menu like next one:

ADD APPLICATION

Application ID
The unique identifier of your application on the network

Description
A human readable description of your new app

Application EUI
An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page.

EUI issued by The Things Network

Handler registration
Select the handler you want to register this application to

Figure 2.3 Application parameters in TTN

2. In *Application ID* and *Description*, write an ID for the application (in our case, I could be “EMITE App”).
3. *Application EUI* and *Handler* registration are given by TTN.

2.2.3. Registering a Device

Once the application is created, it is necessary to register a device. As we already know, a gateway has been registered and application created. Hence, we have what forwarder packets to the server (gateway), and a way to see that information (application). However, it is required a device to send packets in order for gateway can forwarder these packets to the servers.

Therefore, to register a device: click on application (in TTN) → *Devices* → *Register Device*, and follow next steps to fill in box shown in next figure:

Figure 2.4 Device parameters in TTN

1. Write the Device ID (it usually is the device's name).
2. Click on the symbol inside of the red circle to auto generate a Device EUI.
3. The App Key and the App EUI are default values that give you TTN. Finally, click on Register.

Once the Device is created, the Activation Method can be "OTAA" or "ABP". The differences between OTAA and ABP are described in 2.3. Depending on the Method, several keys will be generated by TTN. These keys are extremely important to have connection between the server and the physical device, as that connection is not reachable if those keys are unknown.

2.2.4. Payload format

Another important advantage of using TTN is that it let us to create Payload Formats (decoder, encoder, validator and converter). The functions to make will change depending on the used device. For example, RN2483 is able to measure temperature and luminosity, so we implemented next code (a decoder) to know these values in a correct way from the payload:

```

function Decoder(bytes, port){

var luminosity = (bytes[0]-48)*100 + (bytes[1]-48) * 10
+ (bytes[2]-48);
var temperature = (bytes[4]-48)*100 + (bytes[5]-48) *
10 + (bytes[6]-48);

    return {
        luminosity: luminosity,
        temperature: temperature
    }
}

```

This decoder makes to know temperature and humidity levels sent by RN2483 possible. This device sends those level using ASCII codes (in hexadecimal) and so it is necessary to decode it. Thanks to before code, we can know temperature and humidity correctly.

dev id: rn2483	payload: 32 30 36 20 30 32 35 00	luminosity: 206	temperature: 25
dev id: rn2483	payload: 32 30 35 20 30 32 34 00	luminosity: 205	temperature: 24
dev id: rn2483	payload: 32 30 36 20 30 32 35 00	luminosity: 206	temperature: 25
dev id: rn2483	payload: 32 30 30 20 30 32 35 00	luminosity: 200	temperature: 25
dev id: rn2483	payload: 31 39 38 20 30 32 37 00	luminosity: 198	temperature: 27

Figure 2.5 Luminosity and temperature given by RN2483

2.3.LoRaWAN

LoRaWAN is a network protocol that uses LoRa technology to communicate LoRa devices. It is based on two elements: gateways and nodes. Gateways are in charge of receiving and sending information to the nodes; nodes are the end - devices that send/receive information to/from gateway.

In other words, LoRa is the physical layer, meaning defines parameters such as transmission frequency. However, LoRaWAN is a network protocol and so does a series of transmission techniques in order to reach a greater security communication and to improve the user's experience.

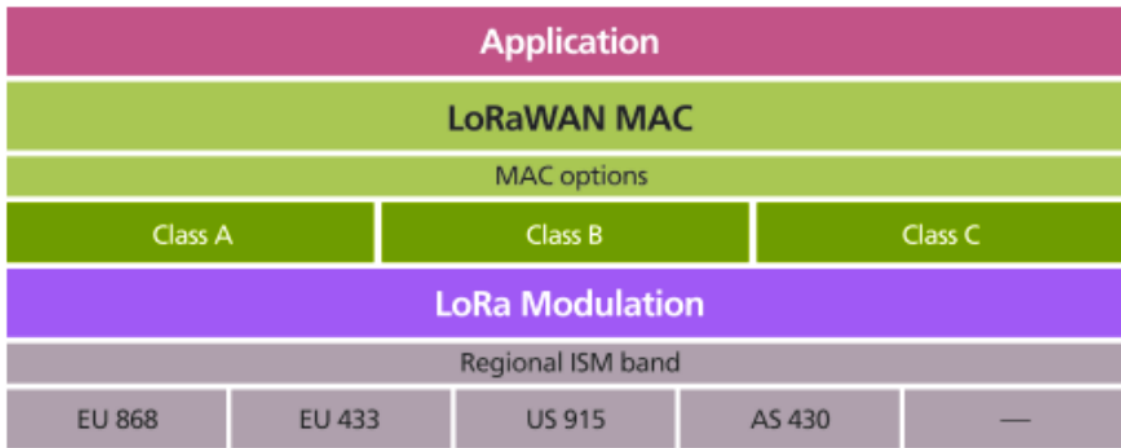


Figure 2.6 LoraWAN from LoRa

The main features of LoRaWAN are as follows:

- start topology
- low-power consumption and long-range (from 10 to 15 km line-of-sight)
- support 3 nodes types (A, B and C)
- private and public networks
- devices management,
- low-data-transfer (to 242 bytes)
- AES 128 encryption

A classic LoRaWAN network would be based on the next figure, where some end - devices are connected with gateways (concentrators) that send all the information to a server, which sends data to a final user application using an API.

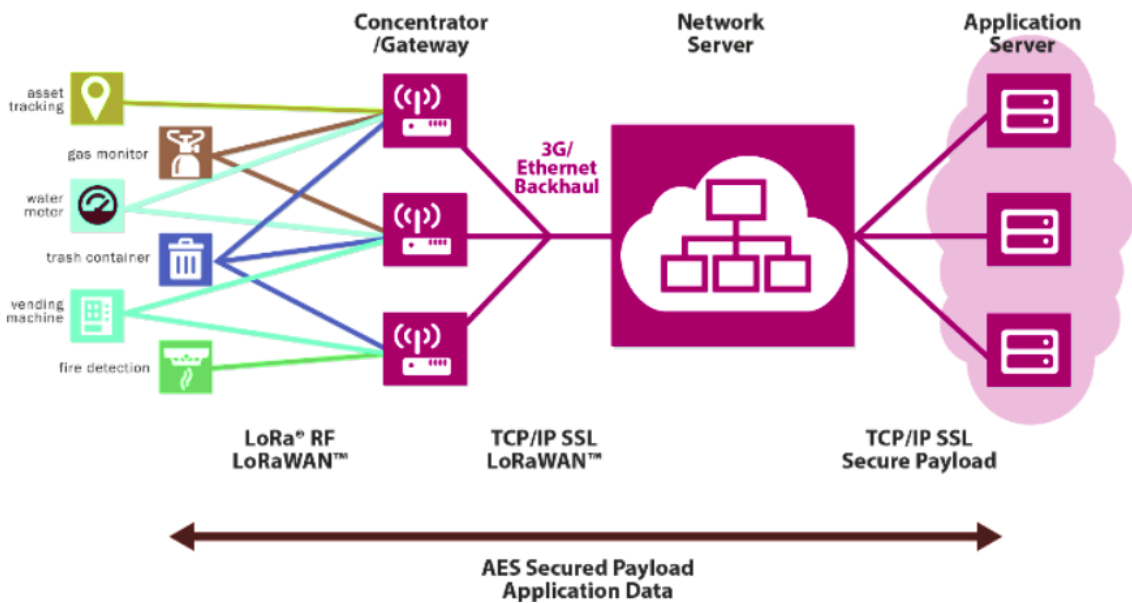


Figure 2.7 LoRaWAN network

2.3.1. OTAA Method

Over-the-Air Activation (OTAA) is the preferred and most secure way to connect with TTN. Devices perform a join-procedure with the network, and a dynamic *DevAddr* is assigned and security keys are negotiated with the device. The configuration parameters are:

- *DevEUI*: it is a factory identifier and it makes each device as unique. It is possible to adjust this configuration on some devices.
- *AppEUI*: unique application identifier used to group objects. This 64-bit address is used to classify devices by application. This configuration can be adjusted.
- *AppKey*: A 128-bit AES key shared between the peripheral device and the network. It is used to determine the session keys. This configuration can be adjusted.

With this data in a simple way, the connection is made as follows:

1. The node requests a join to the network with the configuration data and opens the reception window.
2. The Gateway receives the request and sends it to the server.
3. The server verifies that the node is registered and the encryption key is correct.
4. If it is correct, assign a temporary session and send it through the gateway to the node, if the data is incorrect, reject the join.
5. The node receives the temporary session and can send data to the network.

The main advantage of the OTAA connection is the security since the session says: "it is created in the air" and it is renewed every time the device loses connection, it is turned off or restarted, this makes it difficult for someone to steal the session and clone the device.

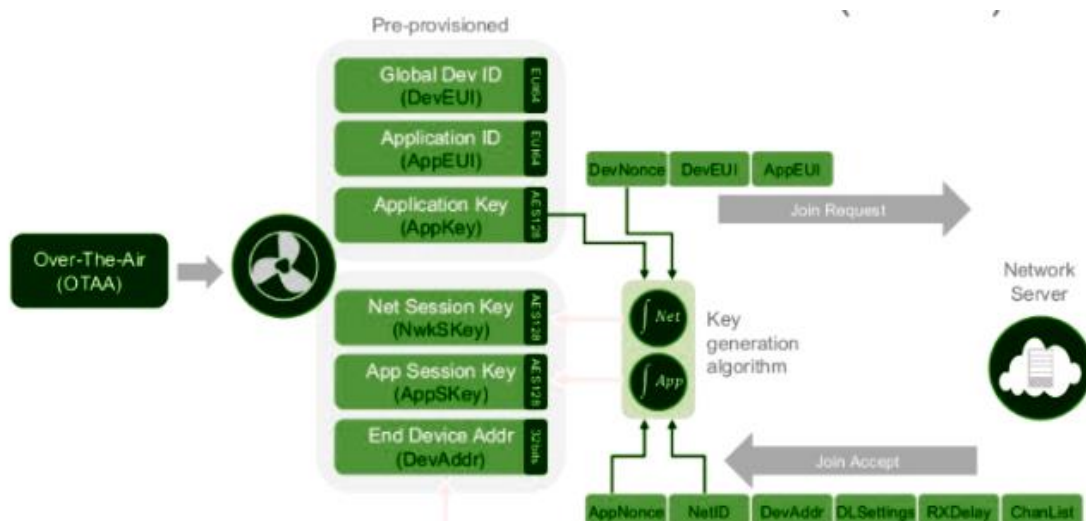


Figure 2.8 LoRaWAN Over-The-Air Activation

2.3.2. ABP Method

Activation by Personalization (ABP). In some cases, it could be necessary to hardcode the *DevAddr* as well as the security keys in the device. This mode is the simplest mode of connection. The connection parameters are:

- *DevAddress*: logical address (equivalent to IP address) that will be used for all subsequent communication with the network.
- *NetworkSessionKey*: encryption key between the device and the operator used for transmissions and used to validate the integrity of the messages.
- *ApplicacionSessionKey*: encryption key between the device and the operator (through the application) used for transmissions and used to validate the integrity of the messages.

With these parameters, the connection is made as follows:

1. The device sends data to the Gateway
2. The Gateway valid that the data corresponds to the session
3. If the session is correct, the data is processed, if not rejected

The main advantage of this type of connection is that it is not required to join the network to send data. Server confirmation is not necessary since the session is already manually assigned. For devices that are in motion or do not have an excellent reception, this type of connection is the best. The disadvantage is that when the encryption key is found in the device it can be extracted and cloned by an attacker.

ABP pre-provisions keys and device address

Join procedure is bypassed

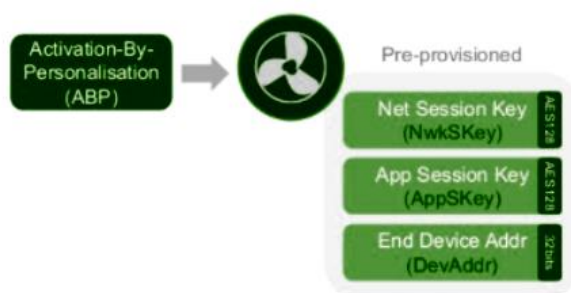


Figure 2.9 LoRaWAN Activation-By-Personalization

2.4. Kerlink LoRa IoT Station as a part of a LoRa network

To connect with the Station, it is required to open a *Putty* terminal and configure the following parameters:

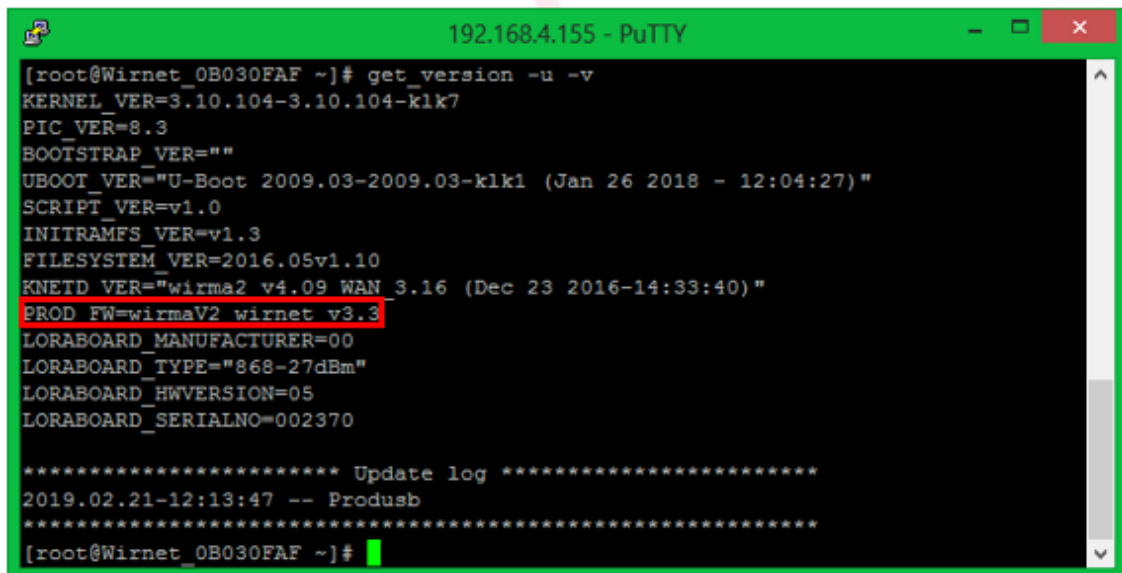
Speed (baud): 115200

Data bits: 8
Stop bits: 1
Parity: None
Flow control: None

2.4.1. Updating Firmware

At the beginning, the firmware version was v2.0. It is recommended to update the firmware version to the last one. In our case, the last version was v3.3 and so we had to update it.

First of all, a USB is necessary in order to update all the required files. All these files may be downloaded from the *Wiki* page of *Kerlink*. It is necessary to copy the recent downloaded firmware in a USB, and plug it in the kerlink IoT Station. Once that firmware is properly updated, a new password is generated. The password is made by connecting “pdmk-0” + “<MAC address of LoRa Station>”. Finally, we can check the new software version.



```
[root@Wirnet_0B030FAF ~]# get_version -u -v
KERNEL_VER=3.10.104-3.10.104-klk7
PIC_VER=8.3
BOOTSTRAP_VER=""
UBOOT_VER="U-Boot 2009.03-2009.03-klk1 (Jan 26 2018 - 12:04:27)"
SCRIPT_VER=v1.0
INITRAMFS_VER=v1.3
FILESYSTEM_VER=2016.05v1.10
KNETD_VER="wirma2 v4.09 WAN 3.16 (Dec 23 2016-14:33:40)"
PROD_FW=wirmaV2 wirnet v3.3
LORABOARD_MANUFACTURER=00
LORABOARD_TYPE="868-27dBm"
LORABOARD_HWVERSION=05
LORABOARD_SERIALNO=002370

***** Update log *****
2019.02.21-12:13:47 -- Produzb
*****
[root@Wirnet_0B030FAF ~]#
```

Figure 2.10 Updated firmware version

An important file to look over is "*loraboard_conf.json*". Thanks to this file, gateway ID may be known, which will be really necessary later.

```
COM12 - PuTTY
[root@Wirnet_OB030FAF fsuser-1]# cat loraboard_conf.json
{
  "SX1301_conf": {
    "radio_0": { "rssi_offset": -170.0 },
    "radio_1": { "rssi_offset": -169.8 },
    "tx_lut_0": {"dig_gain": 3, "pa_gain": 1, "mix_gain": 8, "rf_power": -3 },
    "tx_lut_1": {"dig_gain": 3, "pa_gain": 1, "mix_gain": 9, "rf_power": -1 },
    "tx_lut_2": {"dig_gain": 3, "pa_gain": 1, "mix_gain": 10, "rf_power": 1 },
    "tx_lut_3": {"dig_gain": 3, "pa_gain": 1, "mix_gain": 11, "rf_power": 3 },
    "tx_lut_4": {"dig_gain": 3, "pa_gain": 1, "mix_gain": 13, "rf_power": 6 },
    "tx_lut_5": {"dig_gain": 3, "pa_gain": 2, "mix_gain": 9, "rf_power": 8 },
    "tx_lut_6": {"dig_gain": 3, "pa_gain": 2, "mix_gain": 10, "rf_power": 10 },
    "tx_lut_7": {"dig_gain": 3, "pa_gain": 2, "mix_gain": 11, "rf_power": 12 },
    "tx_lut_8": {"dig_gain": 3, "pa_gain": 2, "mix_gain": 13, "rf_power": 15 },
    "tx_lut_9": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 9, "rf_power": 17 },
    "tx_lut_10": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 10, "rf_power": 19 },
    "tx_lut_11": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 11, "rf_power": 21 },
    "tx_lut_12": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 12, "rf_power": 23 },
    "tx_lut_13": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 13, "rf_power": 25 },
    "tx_lut_14": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 14, "rf_power": 26 },
    "tx_lut_15": {"dig_gain": 3, "pa_gain": 3, "mix_gain": 15, "rf_power": 27 }
  },
  "gateway_conf": {
    "gateway_ID": "7276FF000B030FAF"
  }
}
```

Figure 2.11 Gateway ID from loraboard_conf.json

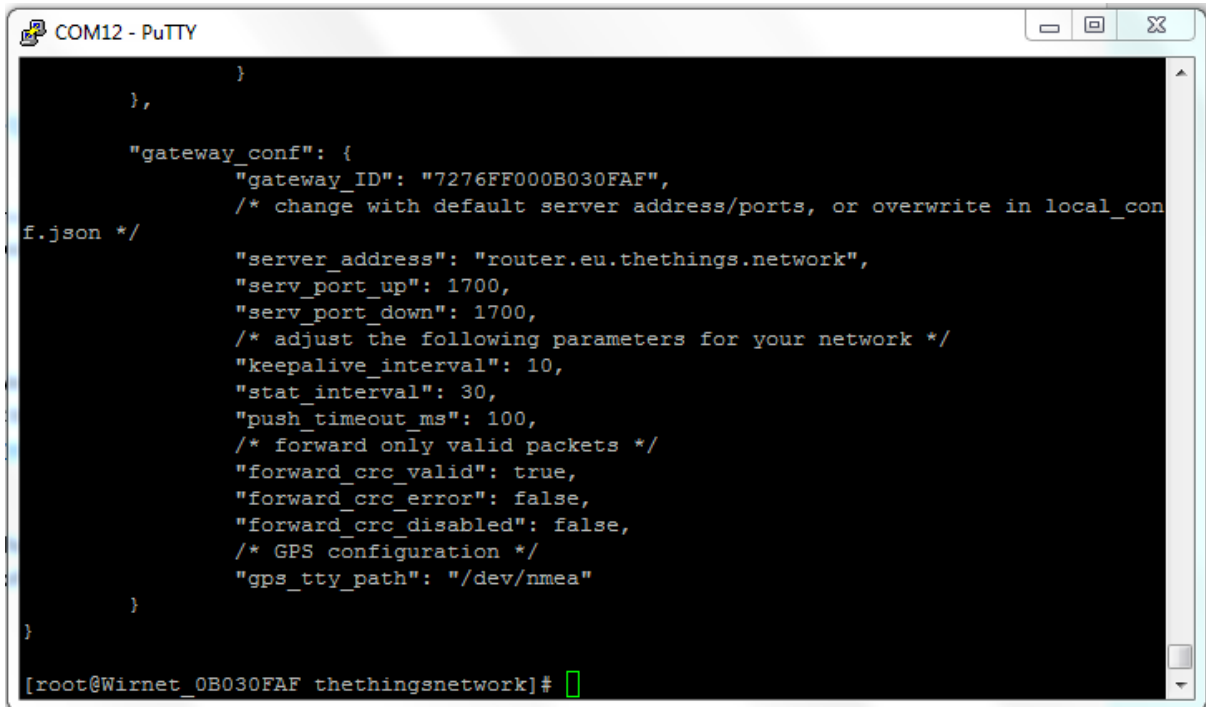
2.4.2. Packet Forwarder

To be able to see what LoRa gateway is receiving, it is necessary that this gateway forwarder received packets to a server. In order for our gateway works as we desire a specific software must be installed, whose name is *Packet Forwarder*. The Packet Forwarder is a program running on the host of a Lora Gateway that forwards RF packets received by the gateway to a server through an IP/UDP link, and emits RF packets that are sent by the server. The chosen server is a public server, whose name is The Things Network, which was briefly described in subsection 2.2.

One way to get Internet access is by means of connecting the IoT Station with a PC, using a RJ45 cable. Then, it is necessary to create a bridge connection between LAN network (network created by a control PC and the IoT Station) and Wi-Fi network (network used by the PC to access to Internet). Thus, our LoRa Station can access to Internet. Finally, a reboot will be necessary and LoRa Station IP direction will change.

A specific packet forwarder must be downloaded and installed in order to communicate the Gateway IoT Kerlink Wirnet Station with the TTN public server. As when updating the firmware, the files have to be put into an empty USB, then, plug it on the Station, and wait some minutes. Then, a "prodbus.log" file will be created in the USB.

Once that packet forwarder is correctly installed in our gateway, the first step is to configure the file "global_conf.json" with the Gateway parameters. In our case, gateway was configured as shown in next picture:



```
COM12 - PuTTY
}
},
"gateway_conf": {
  "gateway_ID": "7276FF000B030FAF",
  /* change with default server address/ports, or overwrite in local_conf
f.json */
  "server_address": "router.eu.thethings.network",
  "serv_port_up": 1700,
  "serv_port_down": 1700,
  /* adjust the following parameters for your network */
  "keepalive_interval": 10,
  "stat_interval": 30,
  "push_timeout_ms": 100,
  /* forward only valid packets */
  "forward_crc_valid": true,
  "forward_crc_error": false,
  "forward_crc_disabled": false,
  /* GPS configuration */
  "gps_tty_path": "/dev/nmea"
}
}
[root@Wirnet_0B030FAF thethingsnetwork]#
```

Figure 2.12 Global_conf.json configuration

2.5.RN2483 mote as a part of a LoRa network

By means of writing commands in the DUT's terminal it is possible to send data from the device to the gateway. Therefore, we used *Putty* with next configuration:

Speed (baud): 57600
Data bits: 8
Stop bits: 1
Parity: None
Flow control: None

A peculiarity of this communication is how to issue each command, as *enter* is not the button to do that properly but *enter* and *ctrl+J*.

We have to keep in mind that the type of connection (ABP or OTAA) must be indicated in our DUT as well as in the app (TTN). The value of the parameters introduced in the DUT must be the same that the server parameters. Every time

the type of connection is changed in the settings site of the server, these values are automatically updated, so their values must also be updated.

OTAA Activation Method

First, we must know what values of parameters related to OTAA the server has.

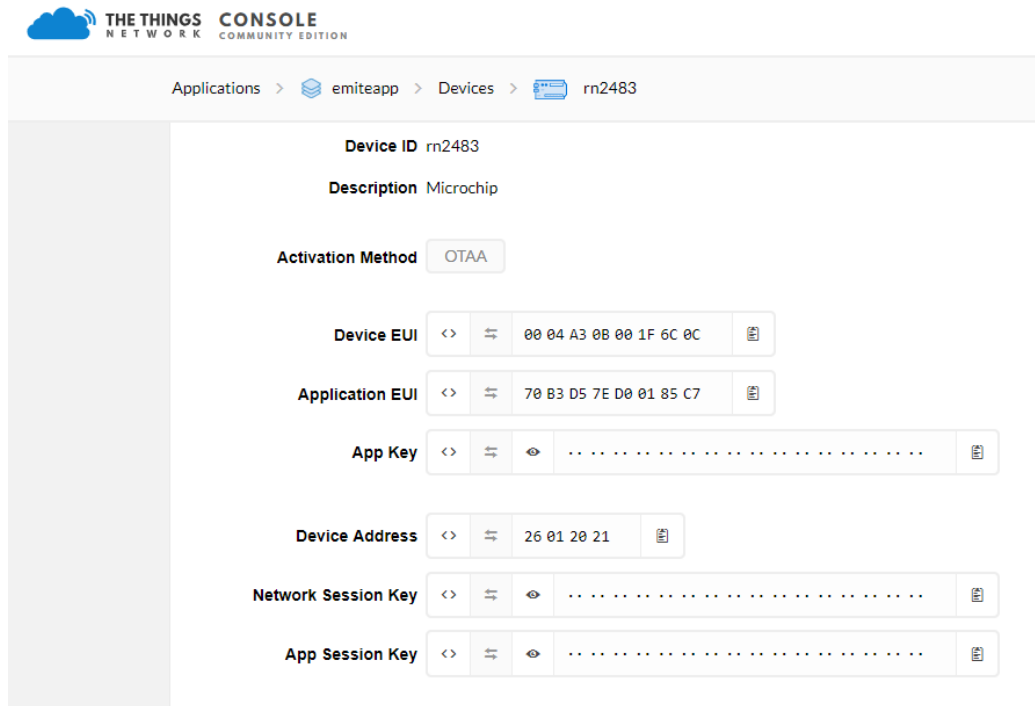


Figure 2.13 OTAA parameters in the server

Then, RN2483 mote must have the same values for these parameters:

- Device EUI
- Application EUI
- Application Key (App key)

Once that these values of the OTAA parameters are known, it is necessary to have an adequate spreading factor to reach an OTAA connection. It is also required to have a correct data rate, that is, data rate must support the value of the spreading factor. To know which combinations may be used, have a look at Table 1-3 EU863-870 TX Data Rate.

ABP Activation Method

First, we must know what values of parameters related to ABP the server has.

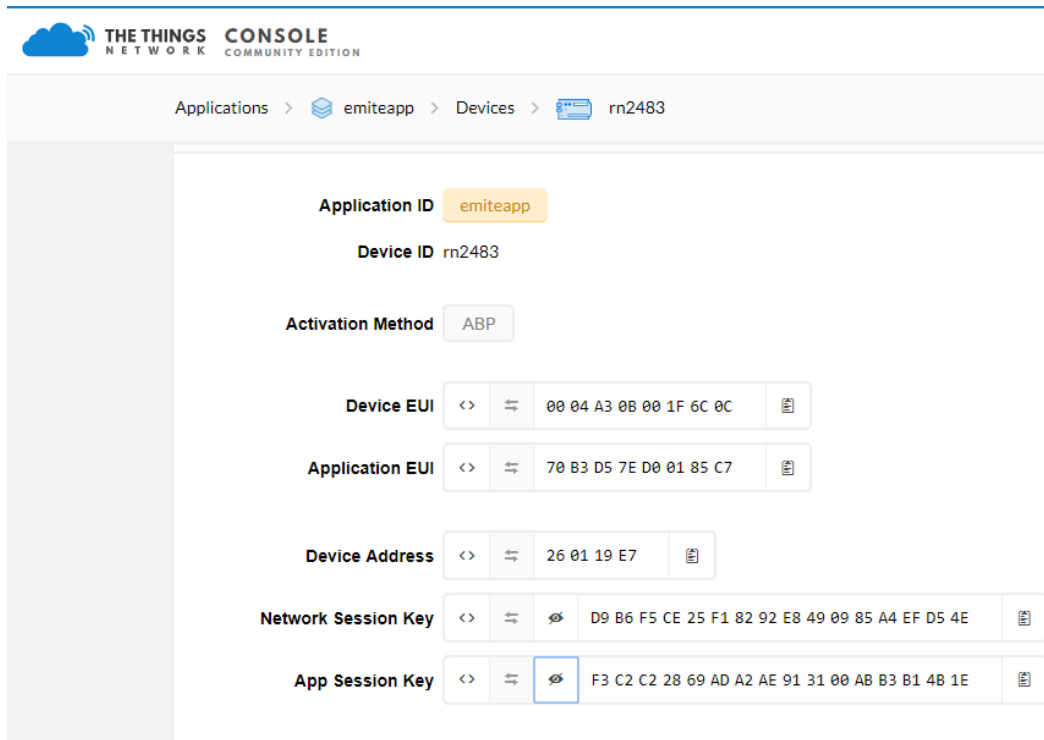


Figure 2.14 ABP parameters in the server

Then, RN2483 mote must have the same values for these parameters.

- Device address
- Network Session Key
- Application Session Key (App Session key)

Once that these values of the ABP parameters are known, it is necessary to have an adequate spreading factor to reach an ABP connection, as before. It is also required to have a correct data rate, that is to say, data rate must support the value of the spreading factor. Hence, to know which combinations may be used, take a look at Table 1-3 EU863-870 TX Data Rate.

2.6. Testing with E500 Reverberation Chamber

2.6.1. DUT inside of the chamber

This subsection explains how to test using a reverberation chamber. Concretely, the used DUT, which is the RN2483 model, will be inside of the chamber and the IoT LoRa Station (this station is a LoRa gateway, but it is called 'station' as it is connected with a DUT, as a base station) will be connected with the chamber through a coaxial cable. That is to say, when the DUT transmits, the signal will be received by the antennas of the reverberation chamber, which are connected with the LoRa gateway through the named coaxial cable.

Remote utilities

In order to control the DUT once that it is inside of the chamber, it is necessary to connect a PC with the RN2483 DUT. We have to keep in mind that this PC is unreachable by us, as it is inside of the chamber and we cannot use it until the test is finished. Therefore, we need to control this PC in a remote way. Another problem is this PC have not any Wi-Fi connection so we must connect it with the Ethernet internal port of the chamber. The external Ethernet port will be connected with another PC. Both internal and external PCs have to be in the same network, meaning they must ping each other properly. To do that, we access to: *Control panel* → *Network* → *Sharing center* → *Ethernet* → *Properties* → *Internet protocol version 4*. Next picture shows an example of configuration.

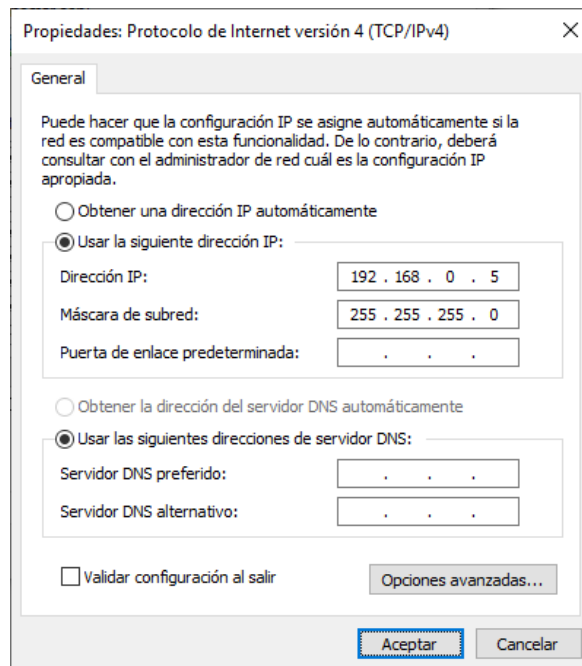


Figure 2.15 Example of configuration to use Remote Utilities

This configuration must be applied on one PC (for example, on the PC inside of the chamber). Therefore, the other PC must have an IP direction that is in the same network e.g. *192.168.0.6*.

The PC inside of the chamber will be controlled, so will be the *host*. Then we must install in it *remote utilities* to be a host. However, with the other PC suffers the opposite situation: it will be a client so *remote utilities* to be a client will have to be installed in it. In the host, we will choose a password, which will be required when accessing from the external PC (client).

LoRa Mote

Once that we are able to control the DUT from a PC that is out of the chamber, we will sent packet from the DUT to the LoRa gateway. First, it is required to be

connected with the LoRa gateway (either ABP or OTAA). In this case, we did that by OTAA.

Afterwards, we will be able to send as many packets as we desire with the corresponding command.

LoRa Gateway

LoRa gateway must be connected with Internet and controlled by another PC. The first thing we must do after turning on the LoRa gateway is to check its internet connection. If it does not work, we have to habilitate a bridge between the WiFi connection of our control PC and its Ethernet connection. Then, reboot the gateway.

When the IoT LoRa Station to be restarted, its direction IP will be changed, meaning its IP direction will not be *192.168.4.155*, as the gateway LoRa will have an IP direction that lets to access to Internet e.g. *192.168.2.44*.

Finally, we must execute packet forwarder and save the generated information in a file, as we will need it to work out the obtained throughput after testing is finished.

When a new state of testing is finished, we have to finish packet forwarder as we want one file for each state. In other words, each state must have one file as each state has its own throughput for a certain circumstances.

2.6.2. LoRa gateway inside of the chamber

This subsection explains how to test using a reverberation chamber. Concretely, the used LoRa, which is a Kerlink LoRa IoT Station, will be inside of the chamber and the RN2483 will be connected with the chamber through a coaxial cable. That is to say, when the mote transmits, the signal will be conducted through the coaxial cable and transmitted by the antennas of the reverberation chamber. Then, that signal will be received by the LoRa gateway.

LoRa Mote

A PC will control this device. Here there is not any problem as both RN2483 and PC are out of the chamber. Therefore, we will be able to send packets from the DUT to the LoRa gateway, which is the inside of the RC. First, it is required to be connected with the LoRa gateway (either ABP or OTAA). In this case, we did that by OTAA.

Afterwards, we will be able to send as many packets as we desire with the corresponding command.

LoRa Gateway

The first thing we must do after turning on the LoRa gateway is to check its internet connection. If it does not work, we have to habilitate a bridge between the WiFi connection of our control PC and the Ethernet connection. Then, reboot the gateway in order to change its IP direction and so it has internet connection.

Finally, we must execute packet forwarder and save the generated information in a file, as we will need it to work out the obtained throughput after testing is finished.

When a new state of testing is finished, we have to finish packet forwarder as we want one file for each state. In other words, each state must have one file as each state has its own throughput for a certain circumstances. For doing that, one PC (which controls the LoRa gateway) will be also inside of the chamber, as it is required to execute and stop packet forwarder.

3. BLE

3.1. Definition

Bluetooth Low Energy is a WPAN technology that includes some applications in security and healthcare, among others things. One of the most noted features of this technology is its reduced power consumption (as we will see later) even though it has a similar communication range than classic Bluetooth. In this section, this technology will be compared with other types of Bluetooth. In addition, the most important features of BLE will be described in order to get why we use this technology across.

3.1.1. BLE vs classic bluetooth

Bluetooth Low Energy has a large amount of features very similar to classic Bluetooth, such as work frequency. Concretely, for BLE these frequencies are from 2402 MHz to 2480 MHz. Both classic Bluetooth and BLE use a GFSK modulation for 1 Mbps, although they do not use the same modulation index.

Enhanced Data Rate (EDR) may be also compared to others two types of Bluetooth. Its modulation is not a GFSK modulation, however its has the same number of channels than classic Bluetooth, but BLE. In addition, each type of Bluetooth has a different channel separation and a different maximum throughput. Due to that, a BLE device cannot be communicated with a device that uses classic Bluetooth, and vice versa. Despite of that, there are some devices, which are called as Dual Mode, that can support both technologies BLE and classic Bluetooth.

To sum up, next table collects all this information and shows it in order to make its understanding easier.

Table 3-1 Comparison of BR, EDR and BLE

	BR	EDR	BLE
Modulation	GFSK 0.28 to 0.35	DQPSK / 8DPSK	GFSK 0.45 to 0.55
Nº of channels	79	79	40
Separation	1 MHz	1 MHz	2 MHz
Max. Throughput	1 Mbit/s	2 and 3 Mbit/s	1 Mbit/s

3.1.2. Main features

Therefore, BLE stands out due to its already shown main features. Another important feature is advertising, which is a process with a BLE device makes itself known so that other BLE device can make a connection with it. To do that, three

channels are exclusively dedicated in BLE (so, thirty seven channels are for data transmission). The channels 37, 38 and 39 are in charge of doing advertising.

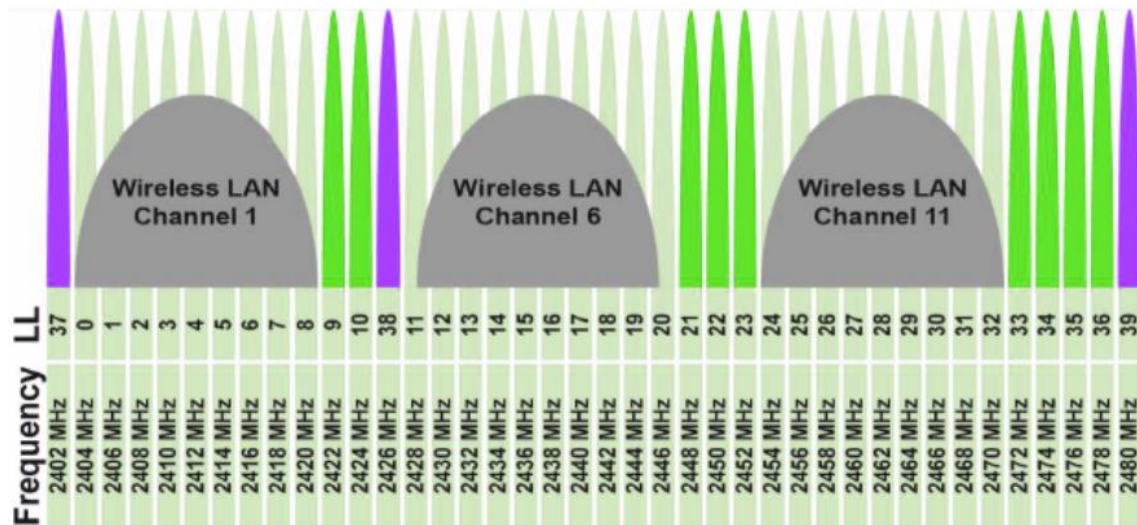


Figure 3.1 Channels and frequencies used by BLE

These three channels (the *advertising* channels) are separated each other, as shown in Figure 3.1, in order to avoid interferences, that is, there are other technologies e.g. ZigBee which use the same frequency. What is more, BLE use Frequency Hopping Spread Spectrum when transmitting to reduce interferences.

To sum up, the main advantages offered by BLE are next ones:

- Low-power consumption: this advantage turns BLE into one of the most popular technology for IoT.
- Easy to install and use: it can be used by small devices such as smartwatches. What is more, directive antennas can be integrated to increase the maximum connection distance.
- Supported by smartphones: as we already know, everybody is constantly using their smartphones and therefore the future technologies must be adapted to that.
- Long battery-life: due to its lower consumption. Therefore, BLE is becoming a quite economic technology.

3.2.6 LowPAN

6LoWPAN is a somewhat contorted acronym that combines the latest version of the Internet Protocol (IPv6) and Low-power Wireless Personal Area Networks (LowPAN). The 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices, and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.

Using IPv6 allows that every device to have its own IP address and so these devices to have internet access. This removes the reliance on smartphones, and would allow billions of devices to connect and exchange information each other in a standardized way over the Internet. As BLE does not natively communicate with IP, the best way to achieve this is using 6LoWPAN. 6LoWPAN (IPv6 over Low-power Wireless Personal Area Networks) is a standard that makes to use IPv6 over networks based on IEEE 802.15.4 standard possible.

The 6LoWPAN system is used for a variety of applications including wireless sensor networks. This form of wireless sensor network sends data as packets and using IPv6 - providing the basis for the name - IPv6 over Low-power Wireless Personal Area Networks.

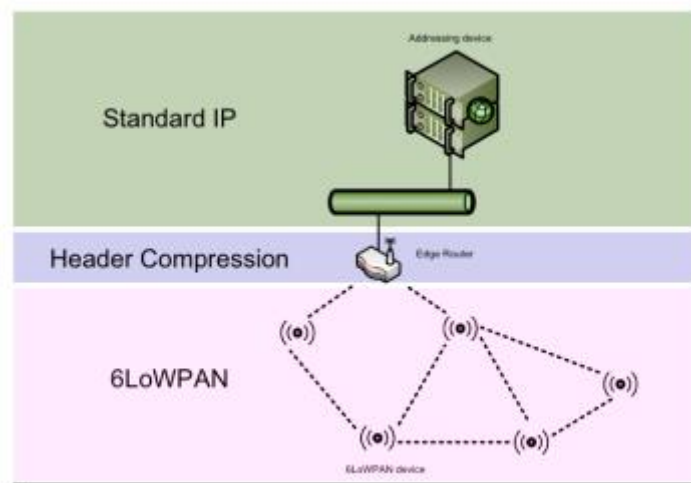


Figure 3.2 6LoWPAN architecture

This standard (IPv6 over Low-Power Wireless Personal Area Network) offers header compression and so decreases the sent overhead, just as meets the IPv6 Maximum Transmission Unit (MTU) requirement, and the most important thing, forwarders to link-layer. Thus, multi-hop delivery is supported.

This simplifies the IP headers, compresses data and encapsulates the IP packets to allow them to be sent via Bluetooth, conserving bandwidth and power. The combination of Bluetooth Low Energy and IPv6 brings much closer to the goal of having small and low-power devices that can communicate each other and with Internet, without having to have different hubs sitting in the middle.

To summarize the enormous amount of reasons to use this standard, next list about 6LowPAN is made:

- Uses IPv6 and so make to connect with other IP networks possible.
- It can be applied in most IoT devices as 6Lowpan is a standard.
- The same issue with operative system; 6LowPAN can be used independently of that.

- It offers a huge amount of directions as it uses IPv6. Therefore, it is perfect for large networks, meaning networks with a large amount of sensors.
- This standard hardly requires a bit of memory as it only affects IP level.

To sum up, we used this standard as we had created a BLE network and this network was not connected with anything, as it was impossible. To use 6LoWPAN gave the opportunity to communicate our devices with all the IoT devices. Furthermore, this standard made to measure throughput using *iperf3* possible.

3.3.ASUS RT-N16 as a part of a BLE network

In this section, how to turn a RT-N16 router into a part of a BLE network is explained. First, to install a distribution of Linux on the router will be necessary. After that, other packages will have to be install in order to support both BLE and 6LoWPAN. In addition, this section is also going to show how to connect a BLE device controlled by this router with another BLE device.

To control the router, it is necessary to access via Ethernet. Obviously, an user name and a password will be required. Next step is to update firmware, which is easily downloadable from Internet, in order for the router supports *OpenWRT*.

3.3.1. Installing OpenWRT

For doing that, the router must be powered on. After that, it is necessary to press the *restore* button on the router until the PWR led blinks. Thus, the router will complete a full reset. Once the router is rebooted, it is advisable to disconnect the control PC from the Internet and plug an Ethernet cable from the computer into one of the router LAN ports. In this way, default web browser may automatically open and go to the router welcome page (if it exists)

Otherwise, we can open a web browser and go to the router welcome URL (or the URL corresponding to the router gateway, when no router welcome page exists). Therefore, if the router welcome page exists, first click on *Go*, and secondly go to the URL <http://192.168.1.1/index.asp> (IP direction may be different depending on the router). Otherwise, login with factory default user name and password.

Then, we have to go to the firmware upgrade tab. Click on *Browse* to have a look at the previously downloaded OpenWrt factory firmware image file. Then, this file must be loaded. It is necessary to wait until *OpenWrt* factory firmware image file is loaded (3 minutes, approximately). Finally, re-start the router. Once that these steps are done, *OpenWRT* can be installed.

First of all, Linux kernel version shall be 3.17 or higher in order to support 6LoWPAN for Bluetooth Low Energy (BLE) through the kernel module *bluetooth_6lowpan*. We can issue the following command to verify the linux kernel version:

```
uname -a
```

In our case, the obtained answer was next one:

```
Linux EMITE_PAN 4.14.95 #0 Mon Jan 28 08:54:32 2019 mips GNU/Linux
```

In other words, the linux kernel version of the router is 4.14.

3.3.2. Installing all required packages for both USB Bluetooth support and 6LoWPAN devices support on OpenWRT

Next step is to install all packages required for USB Bluetooth support. This step had to be manually done and so, next commands must be introduced:

```
opkg update
```

```
opkg install kmod-bluetooth bluez-libs bluez-utils kmod-usb-core kmod-usb-uhci kmod-usb2 usbutils bluez-daemon
```

Note that *opkg* command is not really common. This command is used when introducing commands in the RT-N16 router as it has OpenWRT as a operative system, not a Ubuntu, which is one of the most common used distribution of Linux at the moment. In other words, *opkg* is like *apt-get* but for OpenWRT.

Besides that, some dependencies as *dbus* or *libexpat* packages must also be installed. Therefore, the following commands were used in order to install all packages required for 6LoWPAN and so IPv6 over Bluetooth Low Energy (BLE).

```
opkg update
```

```
opkg install kmod-6lowpan kmod-bluetooth kmod-bluetooth_6lowpan kmod-usb-core kmod-usb-ohci kmod-usb2 bluez-libs bluez-utils
```

Finally, it is necessary to enable automatic Bluetooth device support. For doing that, next command were introduced:

```
/etc/init.d/dbus enable
```

```
/etc/init.d/dbus start
```

```
/etc/init.d/bluetoothd enable
```

```
/etc/init.d/bluetoothd start
```

3.3.3. Connecting RT-N16 router with a BLE device (with 6LoWPAN)

First step is to connect a Bluetooth Low Energy (BLE) dongle (either Laird DVK-BT850 or CSR 4.0 Dongle) to one of the router USB sockets. Then, it is advisable to verify that the modules required for 6LoWPAN over Bluetooth Low Energy (BLE) devices support are already loaded. This verification may be done by issuing the following command:

```
lsmod | grep bluetooth_6lowpan
```

A noted fact is that the following commands to load the 6LoWPAN modules have to be introduced every time the router is started (even if they are already loaded):

```
modprobe 6lowpan
```

```
modprobe bluetooth
```

```
modprobe bluetooth_6lowpan
```

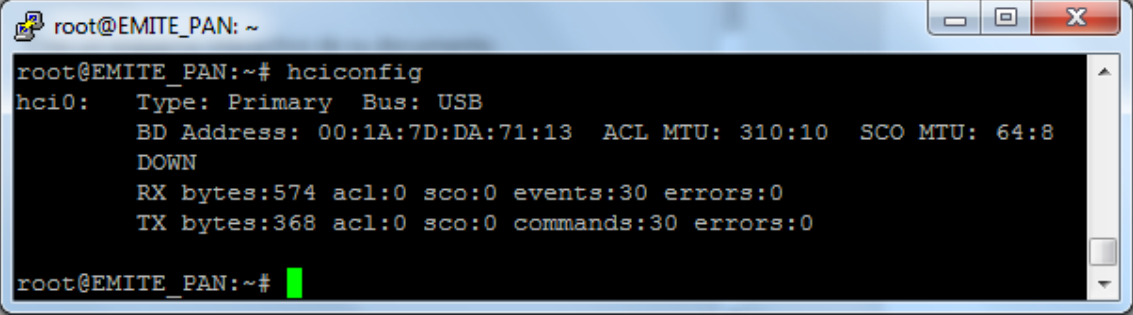
Final step is to enable the Bluetooth 6lowpan module. For doing that, next command must be issued:

```
echo 1 > /sys/kernel/debug/bluetooth/6lowpan_enable
```

Now it would be necessary to look for available HCI devices. If the HCI device to use is down, then it is required to reset the HCI device. After that, the HCI device must be *up* and *running*. It is possible to check that the Bluetooth device installed in the HCI device has Low Energy (LE) support by issuing the following command:

```
hciconfig hci0 lestates
```

The list with all connected Bluetooth Low Energy (BLE) devices may be known, as shown in next picture.

A terminal window titled 'root@EMITE_PAN: ~' showing the output of the 'hciconfig' command. The output displays details for the hci0 interface, including its type (Primary), bus (USB), BD Address (00:1A:7D:DA:71:13), ACL and SCO MTU values, and statistics for RX and TX bytes, events, and errors. The device is currently in a 'DOWN' state.

```
root@EMITE_PAN:~# hciconfig
hci0:  Type: Primary  Bus: USB
       BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10  SCO MTU: 64:8
       DOWN
       RX bytes:574  acl:0 sco:0 events:30 errors:0
       TX bytes:368  acl:0 sco:0 commands:30 errors:0
root@EMITE_PAN:~#
```

Figure 3.3 List with all connected BLE devices

Once that BLE devices that can be controlled through RT-N16 router are known, it is possible to make a connection between this device and another BLE device. The connection process is as follows:

- One device (device 1) starts to do advertising
- Other device (device 2) starts to scan. A list of BLE devices, which are doing advertising, will appear and so their corresponding MAC addresses can be read.
- Device 2 can connect with one of these listed devices as a master. The chosen device will be a slave.

To connect the controlled device through RT-N16 router with other BLE device, next command must be issued:

```
echo "connect <MAC address> 1" >
/sys/kernel/debug/bluetooth/6lowpan_control
```

It is important to understand what this command does. It does not only order to which device to connect with through BLE, but it also activates 6LoWPAN. Respect to the used number (which is "1"), is due to next reason: this number indicates the BLE address type, which are:

- 1; for public address
- 2; for random address

In our case, the address was not a random one but a public one.

It is important to clarify that there are other ways to make a BLE connection, such as by using next command:

```
hcitool -i hci0 lecc <MAC address>
```

However, this command is not able to create an IPv6 over BLE connection, as it does not give any order to 6LoWPAN, which is what we want.

Having said that, we can give way to explain final step. When a BLE connection is made using 6LoWPAN, a new interface appears in each connected device, whose name is *bt0*. This interface is created by 6LoWPAN, and it gives an IPv6 direction for each device. Therefore, to connect, at least, two BLE devices using IPv6 over BLE is already possible.

To verify all the done work, we must do a ping using the IPv6 directions from *bt0* interfaces. For doing that, the used command is next one:

```
ping6 -c 4 -I bt0 <IPv6 direction>
```

“ping6” indicates that the direction to use is an IPv6 direction; “-c 4” is due to four packets will be sent, and “-I *bt0*” indicates that the used interface to send packets will be *bt0*.

Hence, when this BLE connection is finished, *bt0* interface will disappear in every device. All the obtained results are explained in detail later, when results are shown. This section is to clarify how the work has been done, in order for the reader can understand better the way to go until reaching the proposed objectives.

3.4. Raspberry Pi as a part of a BLE network

Before a RT-N16 router was required to be able to control a Bluetooth LE dongle. In this section, the same philosophy is going to be applied. In other words, raspberry pi will be used as a router in order to control a Bluetooth LE dongle. Despite of raspberry pi has its own BLE module, the objective is to create a BLE network, that is to say, not only one router is required.

Furthermore, using a raspberry as a router let us to connect some BLE dongles with the raspberry, so that a larger amount of devices can be part of that network. Obviously, raspberry pi can also be an end-device, but that was not the proposed objective.

3.4.1. Spice up Raspbian for the IoT

First of all, raspbian must be updated. Therefore, an internet connection is required just as enough memory space in the raspberry.

WPAN and 6LoWPAN

Next step was to install WPAN tools and configure our 6LoWPAN device. For doing that, wpan-tools were previously downloaded. Besides that, some dependencies and packages, whose name is *libnl*, were installed before building the wpan-tools. Having done that, we can continue with wpan-tools, as we have to:

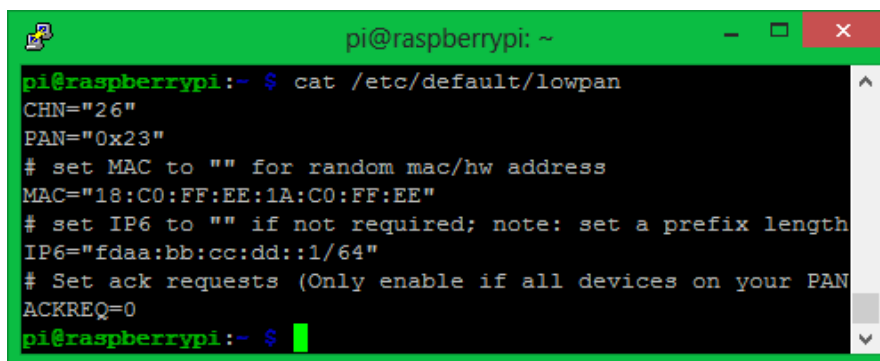
- Configure it
- Build it
- Install it

Then, it is necessary to run a test command to verify all this software are correctly configured and installed.

Other software are also necessary to achieve our objectives. In our case, *htop* and *python* (*python-all-dev*, *python-pip* and *python3-pip*). These ones will not be directly used by us, but by 6LoWPAN due to its complexity. With a little work around providing some helper scripts and a system service definition, it is possible to achieve a 6LoWPAN network interface up and running. Among other things, it is necessary to clone “wpan-raspbian” from *Github*.

As can be read, this section explain the way to go until our objectives, but hardly shows a command. What the reader has to keep in mind is not the amount of required commands to do this project, but the amount of tools and concepts to do that in a proper way.

To finish this subsection, the final step consist of modifying channel and PAN (personal area network) ID as needed. For instance, our case was as shown next image.

A terminal window titled "pi@raspberrypi: ~" with a black background and green text. The terminal shows the command "cat /etc/default/lowpan" and its output. The output includes configuration for channel (CHN="26"), PAN ID (PAN="0x23"), MAC address (MAC="18:C0:FF:EE:1A:C0:FF:EE"), IP6 address (IP6="fdaa:bb:cc:dd::1/64"), and ACKREQ=0. The prompt "pi@raspberrypi:~ \$" is visible at the end of the output.

```
pi@raspberrypi:~ $ cat /etc/default/lowpan
CHN="26"
PAN="0x23"
# set MAC to "" for random mac/hw address
MAC="18:C0:FF:EE:1A:C0:FF:EE"
# set IP6 to "" if not required; note: set a prefix length
IP6="fdaa:bb:cc:dd::1/64"
# Set ack requests (Only enable if all devices on your PAN
ACKREQ=0
pi@raspberrypi:~ $
```

Figure 3.4 Channel and PAN ID

It is advisable to activate the LowPAN autostart (in systemd) in order to be more efficient when using lowpan in the future.

CoAP support

Constrained Application Protocol (CoAP) is one of the major protocols for the Internet of Things; it implements a HTTP-like protocol to query sensors that

provide a RESTful interface. In other words, this protocol lets nodes from a network to connect each other, if and when these devices either are on the same constrained network e.g. low-power or are joined by an internet connection.

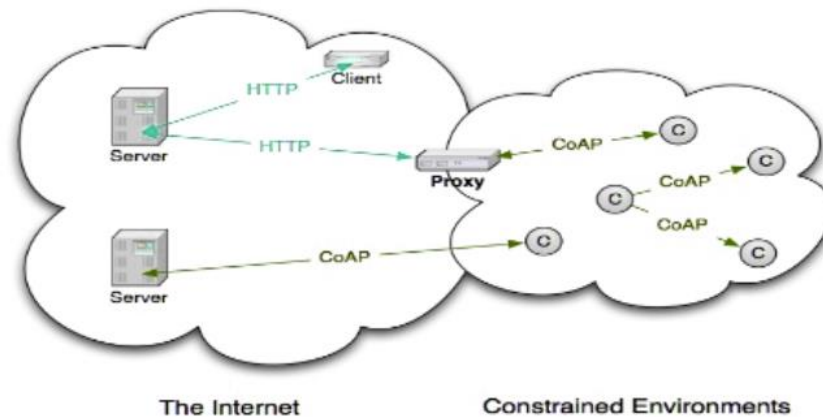


Figure 3.5 Constrained Application Protocol

This protocol is frequently used in *M2M* data exchange, and it is quite similar to HTTP. The main features of this protocol are:

- Low overhead and very simple to parse
- Proxy and caching capabilities
- Web protocol used in *M2M* with constrained requirements
- URI and content-type support
- Asynchronous message exchange

As can be seen, some features are very similar to *HTTP*. However, *CoAP* must not be considered a compressed *HTTP* protocol due to it is specifically designed for IoT. Concretely, for *M2M* and therefore it is very optimized for this task.

Hence, to take advantage of this protocol will be necessary to:

- Download *libcoap*
- Configure *libcoap*
- Build *libcoap*

We must not forget that it is required to update the dynamic linker cache before running *coap-client*. Once that the corresponding commands for doing this have been issued, we have also to install:

- *txThings*: is a *CoAP* library implemented with Python. One of the its main features is its asynchronous I/O framework and networking engine.
- *Aiocoap*: is a *CoAP* package implemented with Python. It facilitates concurrent operations while maintaining a simple to use interface and not depending on anything outside the standard library.

3.4.2. Raspberry Pi as an end-device

Once that raspberry is ready to establish IPv6 over BLE, final step is to check this topology works properly. First we are going to do is to check both 6LoWPAN and CoAP support. Therefore, the Laird will be connected with the RT-N16 via USB (Laird would be an end-device and will be controlled by the router) and the raspberry will be connected with a control PC via Ethernet.

The topology we are going to implement is like next one:

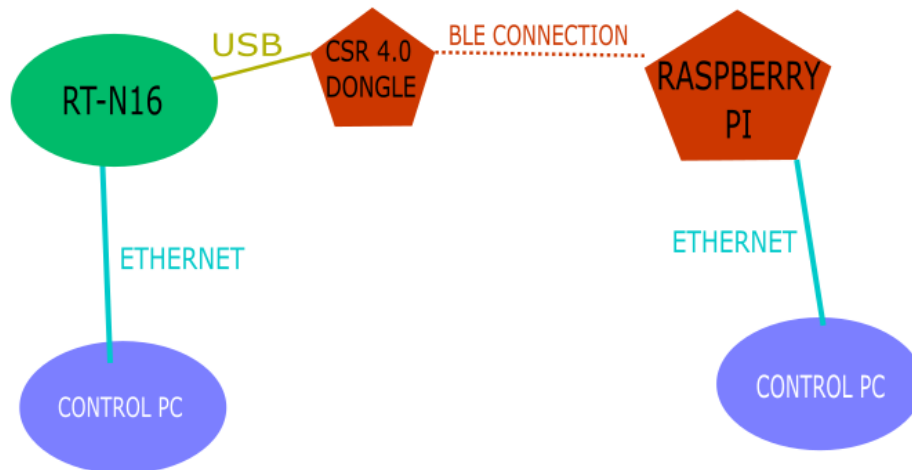


Figure 3.6 Topology to implement a BLE connection (I)

As can be seen in figure 3.6, raspberry will be an end-device, and a PC will control it. It is necessary to load and enable the IPv6-over-Bluetooth support on Raspberry Pi and it is advisable to verify that the modules required for 6LoWPAN over Bluetooth Low Energy (BLE) devices support are already loaded, just as to load the 6LoWPAN modules (even if they are already loaded). For doing this, next commands must be issued:

```
modprobe 6lowpan
modprobe bluetooth
modprobe bluetooth_6lowpan
```

The same as explained in 3.3.3, one of the final steps is to enable the Bluetooth 6lowpan module. In fact, the used commands are the same commands as the devices have the same operative system (Linux). Therefore, for doing that, next command must be issued:

```
echo 1 > /sys/kernel/debug/bluetooth/6lowpan_enable
```

Now it would be necessary to look for available HCI devices. If the HCI device to use is down, then it is required to reset the HCI device. After that, the HCI device must be *up* and *running*. The list with all connected Bluetooth Low Energy (BLE) devices just as if our raspberry has a BLE module may be known by next command:

```
hciconfig
```

Once that a BLE module of raspberry is ready to be controlled, it is possible to make a connection between this module and a BLE device. The connection process is as follows:

- One device (device 1) starts to do advertising
- Other device (device 2) starts to scan. A list of BLE devices, which are doing advertising, will appear and so their corresponding MAC addresses can be read.
- Device 2 can connect with one of these listed devices as a master. The chosen device will be a slave.

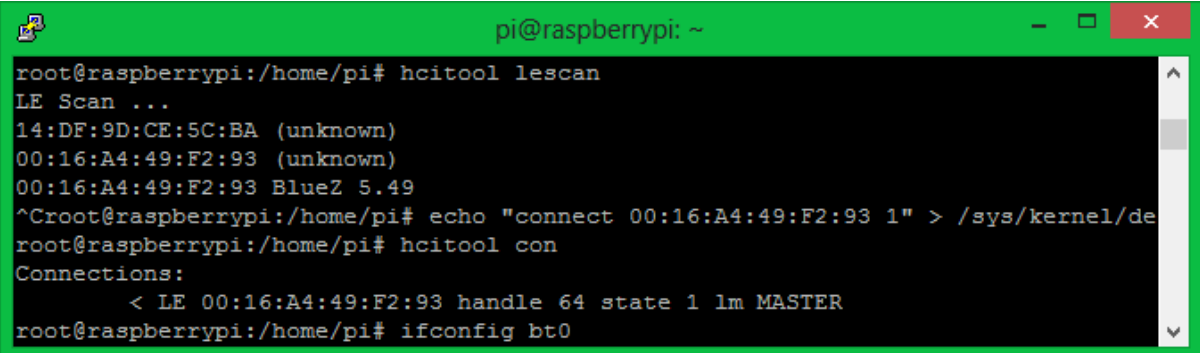
To connect a BLE module of the raspberry with a BLE device, next command must be issued:

```
echo "connect <MAC address> 1" > /sys/kernel/debug/bluetooth/6lowpan_control
```

It is important to understand what this command does. It does not only order to which device to connect with through BLE, but it also activates 6LoWPAN. Respect to the used number (which is "1"), is due to next reason: this number indicates the BLE address type, which are:

- 1; for public address
- 2; for random address

In our case, the address was not a random one but a public one. The obtained result is as shown next picture.



```
pi@raspberrypi: ~
root@raspberrypi:/home/pi# hcitool lescan
LE Scan ...
14:DF:9D:CE:5C:BA (unknown)
00:16:A4:49:F2:93 (unknown)
00:16:A4:49:F2:93 BlueZ 5.49
^Croot@raspberrypi:/home/pi# echo "connect 00:16:A4:49:F2:93 1" > /sys/kernel/de
root@raspberrypi:/home/pi# hcitool con
Connections:
  < LE 00:16:A4:49:F2:93 handle 64 state 1 lm MASTER
root@raspberrypi:/home/pi# ifconfig bt0
```

Figure 3.7 IPv6 over BLE connection process (I)

As figure 3.7 shows, the BLE connection has been correctly done. It is important to clarify that there are other ways to make a BLE connection, such as by using next command:

```
hcitool -i hci0 lecc <MAC address>
```

However, this command is not able to create an IPv6 over BLE connection, as it does not give any order to 6LoWPAN, which is what we want.

Having said that, we can give way to explain final step. When a BLE connection is made using 6LoWPAN, a new interface appears in each connected device,

whose name is *bt0*. This interface is created by 6LoWPAN, and it gives an IPv6 direction for each device. Therefore, to connect, at least, two BLE devices using IPv6 over BLE is already possible, as we will see later.

3.4.3. Setup a native 6LoWPAN router

From this point forward it is assumed that initial setup has been done on the raspberry Pi and WPAN has also been enabled in the Linux Kernel i.e. raspberry Pi 3 has been already *spiced up*. Therefore, we can do next step, which consist of raspberry become as a router, that is to say, a BLE device connected with raspberry to be able to connect with other BLE devices through the raspberry.

To achieve this, it is necessary to install and configure some things in the raspberry, as we will see later. In order to understand this better, this subsection explains how to:

- install router advertisement daemon (*radvd*)
- configure (and test) Raspbian as 6LoWPAN router

In order to install latest version of *radvd*, first some dependencies have to be installed. Though they are likely already installed or available via the apt repositories in raspbian, these version could be a bit old. Therefore, newer versions have to manually installed.

Flex

Fast LEXical analyzer generator is a tool for generating scanners, meaning lexical analyzers. It is also a free and open-source software. It works as follows:

- First, FLEX reads a specification of a scanner either from an input file **.lex*, or from standard input, and it generates as output a C source file *lex.yy.c*
- Then, *lex.yy.c* is compiled and linked with the "-lfl" library to produce an executable *a.out*.
- Finally, *a.out* analyzes its input stream and transforms it into a sequence of tokens.

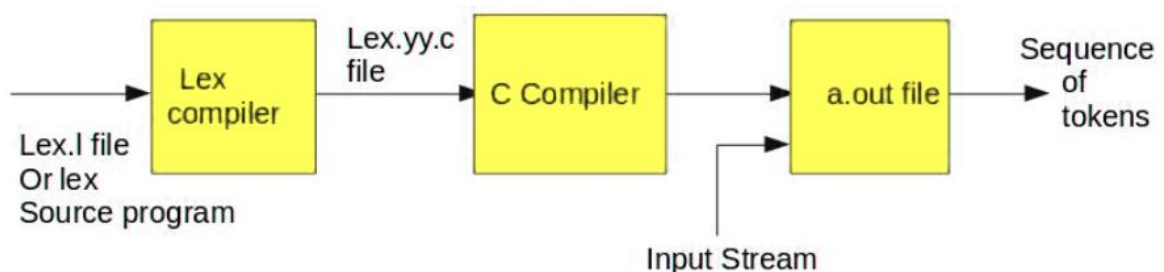


Figure 3.8 Fast Lexical Analyser Generator

This generator can be installed using the APT package system. Even we used FLEX in order to use *radvd* later, other generator can be used instead of e.g. *bison*.

Radvd

Router ADVERTISEMENT Daemon is an open-source software product that implements link-local advertisements of IPv6 router addresses. It is run by Linux (or Berkeley Software Distribution systems, BSD), acting as IPv6 routers.

It listens to router solicitations and sends router advertisements as described in "Neighbour Discovery for IP Version 6 (IPv6)". With these advertisements hosts can automatically configure their addresses and some other parameters. They also can choose a default router based on these advertisements. After a proper configuration, the daemon sends advertisements through specified interfaces and clients are hopefully receive them and auto-magically configure addresses with received prefix and the default route.

To use this software it is necessary to start by getting the source code and run initial configuration. In other words:

- Download it
- Configure it
- Build it
- Install it

Next step consists of creating or editing the file `/etc/radvd.conf`, whose content is shown here to clarify what it is doing:

```
interface lowpan0
{
    AdvSendAdvert on;
    UnicastOnly on;
    AdvCurHopLimit 255;
    AdvSourceLLAddress on;

    prefix fd00:1:2:3::/64
    {
        AdvOnLink off;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    abro fd00:1:2:3:a:b:c:d
    {
        AdvVersionLow 10;
        AdvVersionHigh 2;
        AdvValidLifeTime 2;
    }
}
```

```
};  
};
```

The Unique Local Address (ULA) prefix (fd00:1:2:3::/64) must be replaced with our random prefix network, just as the IP address with the global IP address of the LoWPAN interface. Keep in mind that for multi hop wireless networks it is imperative not to use the link local address of the interface. In our case, both the ULA prefix and the global IPv6 address of the LoWPAN interface that worked properly are the same as shown in before code.

Finally, it is required to enable forwarding for IPv6 interfaces, otherwise Linux will not send *neighbour advertisements* with the router flag set. This last step can be permanently set as it is more efficient that enabling forwarding for IPv6 interfaces whenever we need it.

3.4.4. Raspberry as a router

Once that both raspberry is ready to establish IPv6 over BLE and LoWPAN and CoAP support works properly, final step is to create a BLE network. Therefore, the CSR 4.0 Dongle will be connected with the RT-N16 via USB and the Laird DVK-BT850 will be connected with the raspberry, which will be controlled by a PC via Ethernet.

The topology we are going to implement in this subsection is like next one:



Figure 3.9 Topology to implement a BLE connection (II)

As can be seen in figure 3.9, raspberry will be used as a router given that it will control the Laird, which will be the end-device. As before subsection, it is necessary to load and enable the IPv6-over-Bluetooth support on Raspberry Pi and it is advisable to verify that the modules required for 6LoWPAN over Bluetooth Low Energy (BLE) devices support are already loaded, just as to load the 6LoWPAN modules (even if they are already loaded). For doing this, the same three commands must be issued:

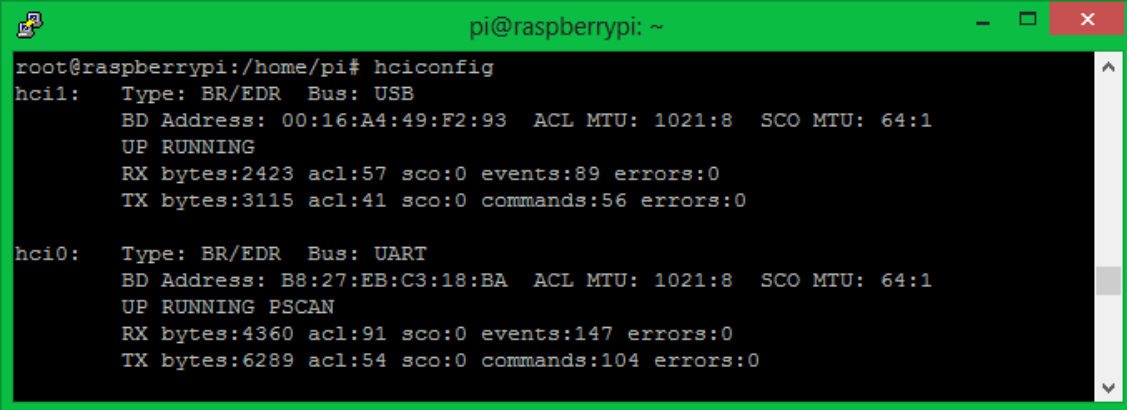
```
modprobe 6lowpan
modprobe bluetooth
modprobe bluetooth_6lowpan
```

The same as explained in 3.3.3, one of the final steps is to enable the Bluetooth 6lowpan module. The used commands are actually the same commands as the devices use Linux, that is, the same operative system. Hence, next command must be issued:

```
echo 1 > /sys/kernel/debug/bluetooth/6lowpan_enable
```

Now it would be necessary to look for available HCI devices. If the HCI device to use is down, then it is required to reset the HCI device, as it was previously said. After that, the HCI device we want to use must be *up* and *running*. The list with all connected Bluetooth Low Energy (BLE) devices with our raspberry may be known by next command:

```
hciconfig
```



```
pi@raspberrypi: ~
root@raspberrypi:/home/pi# hciconfig
hci1:  Type: BR/EDR  Bus: USB
      BD Address: 00:16:A4:49:F2:93  ACL MTU: 1021:8  SCO MTU: 64:1
      UP RUNNING
      RX bytes:2423 acl:57 sco:0 events:89 errors:0
      TX bytes:3115 acl:41 sco:0 commands:56 errors:0

hci0:  Type: BR/EDR  Bus: UART
      BD Address: B8:27:EB:C3:18:BA  ACL MTU: 1021:8  SCO MTU: 64:1
      UP RUNNING PSCAN
      RX bytes:4360 acl:91 sco:0 events:147 errors:0
      TX bytes:6289 acl:54 sco:0 commands:104 errors:0
```

Figure 3.10 BLE modules controlled by raspberry

As this picture shows, there are two BLE modules that may be controlled by the raspberry; the intern module of the raspberry and the recent connected dongle (Laird). In addition, both module and dongle are *up running*, that is, both of them can be used to make a BLE connection with another BLE device.

Once that a BLE device is connected with the raspberry and can be controlled, it is possible to make a connection between this device and another BLE device. The connection process is as follows:

- One device (device 1) starts to do advertising
- Other device (device 2) starts to scan. A list of BLE devices, which are doing advertising, will appear and so their corresponding MAC addresses can be read.
- Device 2 can connect with one of these listed devices as a master. The chosen device will be a slave.

To connect a BLE device controlled by the raspberry with another BLE device, next command must be issued:

```
echo "connect <MAC address> 1" >
/sys/kernel/debug/bluetooth/6lowpan_control
```

As can be realized, the used commands are the same as before subsection. It is due to we are controlling the raspberry. In other words, the real difficulty to use the raspberry as a router is not to control the dongle through the raspberry, but to turn raspberry into a router, as many protocols and software have to be used just as many parameters must be well configured.

Therefore, once that the controlled dongle is connected, next result is obtained, as shown next picture.



```
pi@raspberrypi: ~
root@raspberrypi:/home/pi# hcitool -i hci1 con
Connections:
root@raspberrypi:/home/pi# echo "connect 00:1A:7D:DA:71:13 1" > /sys/kernel/debug
root@raspberrypi:/home/pi# hcitool -i hci1 con
Connections:
    < LE 00:1A:7D:DA:71:13 handle 64 state 1 lm MASTER
root@raspberrypi:/home/pi# hcitool -i hci0 con
Connections:
```

Figure 3.11 IPv6 over BLE connection process (II)

As figure 3.11 shows, the BLE connection has been correctly done. It is important to clarify that only one of two possible modules is connected; hci1 module is connected with 00:1A:7D:71:13, whereas hci0 module is not connected. In other words, the Laird and another BLE device are connected and so the raspberry is being a router.

4.Results

4.1.LoRa

4.1.1. First test to clarify

Before measuring throughput, it is important to understand how LoRa parameters such as *spreading factor* can affect the test. In other words, if one of these parameters modified, we should understand how that fact may affect to the result. Therefore, a test was done that consisted of transmitting from different points of the building (ELDI) in order to verify both the relationship between RSSI and distance and how some parameters can alter the obtained results. The used configuration was:

- Spreading Factor: 7 (it will automatically change if necessary)
- Bandwidth: 125 kHz
- C/R: 4/5
- Data Rate: 5
- Power (DUT): 14 dBm

Packets were sent with RN2483 mote, from the marked points in next picture.

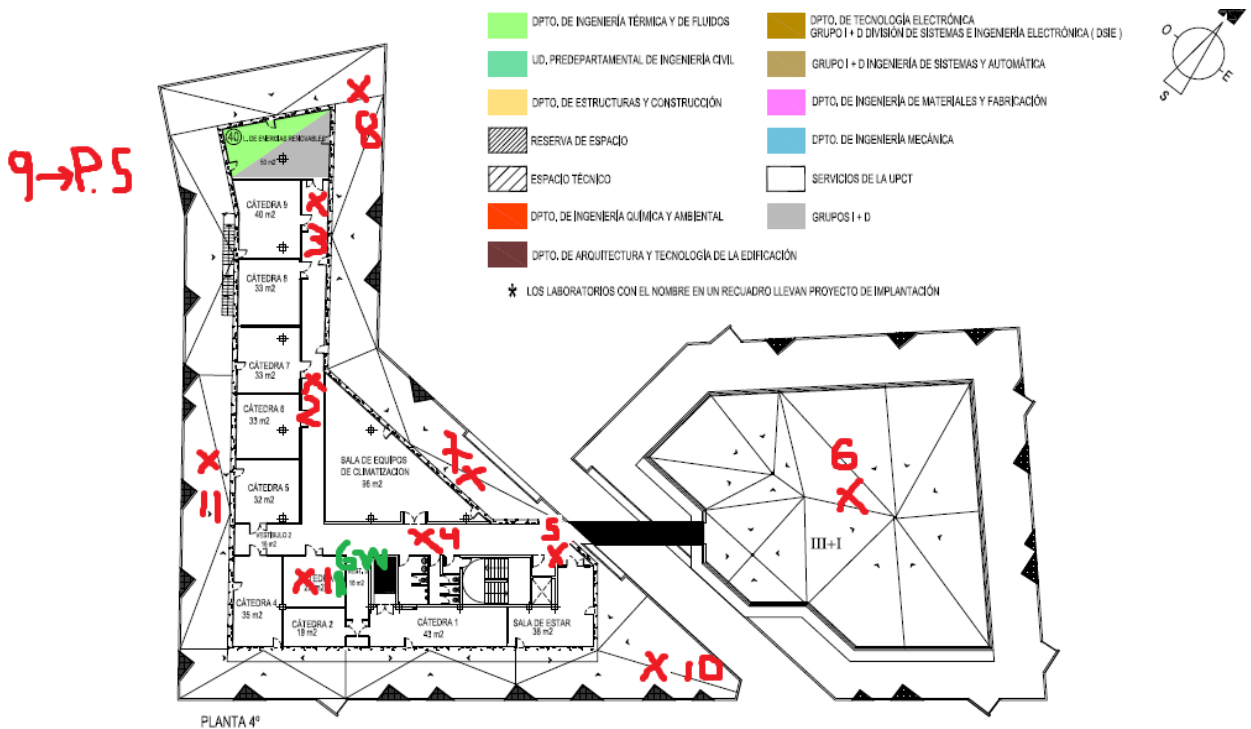


Figure 4. 1 Map for LoRa measurements at ELDI

Looking at above map, it is easy to realize that, for instance, point 8 is much further from LoRa IoT Station (point 1) than point 4, that is, the obtained RSSI when sending packets from point 8 will be less than when sending from point 4. Then, the obtained results are in next table:

Table 4-1 LoRa measurements in ELDI

Measurement index	RSSI (dBm)	SNR (dB)	SF
1	-34	8	7
2	-43	6.5	7
3	-54	7.8	7
4	-53	5.8	7
5	-66	10	9
6	Mac Error Occurred	Mac Error Occurred	10/11/12
7	-58	7.2	7
8	-81	8	7
9	-89	9.5	7
10	Mac Error Occurred	Mac Error Occurred	7/9/11
11	-67	6.2	7

As it was to be expected, RSSI is greater when sending packets from a near place, and it decreases just as packets are sent from further places. We must also keep in mind that other circumstances can alter RSSI i.e. whether there is line of sight or not.

Nevertheless, the most noted thing is not the relation between RSSI and distance, but relation between SNR and SF. As we can easily see, all the measurements were done using seven as a value of SF. However, the only time that SF was different (that value was nine, that is, the SF was larger) the obtained SNR was better (10 dB). This fact is due to SF definition i.e. the bigger SF, a better SNR and a lower data rate will be obtained (see 2.1.1).

In fact, when sending packets from a really far place, these packets could not be received by LoRa gateway. Then, RN2483 mote tried to send the same packets again, but using a bigger SF as a better SNR can be reached and so the packet could be correctly received. This process does not stop until RN2483 mote used a value of SF as big as possible, that is to say, twelve.

4.1.2. Testing in a RC

As table 4-1 showed us, obtained SNR changes depending on SF and distance between devices when transmitting packets. Despite of this, it is possible to realize that SNR value can be larger even large distance between distances. In other words, LoRa achieves a good SNR value as it uses some transmission techniques.

Testing with a RC, we are able to obtain a SNR value in a very real environment, as a reverberation chamber creates a large amount of reflected beams, which are considered as interferences by receiver. In a real environment, our devices will receive many packets that are not desired. In fact, these packets will deteriorate our communication as they may create interferences. Therefore, it is really important to verify that our devices are able to communicate each other in a real environment. In other words, to verify a good communication with our devices, they must be tested in a RC, not in an AC. To know better why that is like this, see Annex A: Different Environments to Test a Device.

Positions and stirrer

For doing this, it is necessary to use the software *MIMO Analyser GUI*. In our case, we are going to attenuate the transmitted signal from 0 dB to 60 dB, where the step is 5, meaning the transmitted signal will be attenuated 0 dB, 5 dB, 10 dB, 15 dB, 20 dB, 25 dB, 30 dB, 35 dB, 40 dB, 45 dB, 50 dB, 55 dB and 60 dB. What we want to see is how RSSI and SNR values change depending on attenuation.

It is important to know there is another attenuation, which is 26.3 dB. Due to antennas of the used reverberation (remember, E500 RC), size of RC and other things, there is an added attenuation. This attenuation varies depending on used frequency. Hence, we have to add this attenuation to the known attenuation (from 0 dB to 60 dB, step 5 dB). The value of this attenuation is, for the work frequency (868 MHz), 26.3 dB.

For testing in RC, the position of the stirrer is changing every time the whole values set is already used, that is, when all the attenuations (from 0 dB to 60 dB) are used. Respect to the stirrer, it has two positions (left and right), so that position changes every time the whole attenuation set is tested. Then, when the stirrer is back to its first position, the DUT is spun (turned 90°). To sum up, there are two stirrer positions and four DUT positions (or positions of table, as the DUT is on the table), as next table clarifies.

Table 4-2 Modes for testing in a RC

Mode number	Stirrer position	Table position (0°)
1	Left	0
2	Right	0
3	Left	90
4	Right	90
5	Left	180
6	Right	180
7	Left	-90
8	Right	-90

As table 4-2 verifies, there are eight different modes when testing in a RC. It is truly important to understand how to test as best as possible and so a picture is shown below.

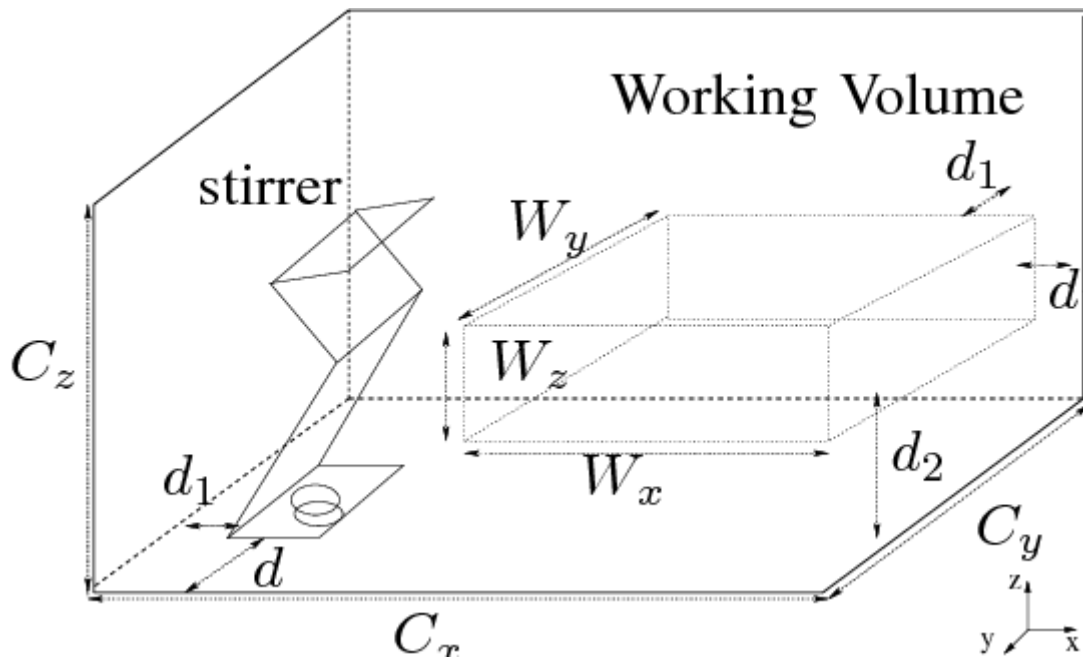


Figure 4. 2 Inside of a RC

Depending on d value, the positions of the stirrer will change. To know E500 features in detail, see Annex B: Utilized Reverberation Chamber.

Obtained SNR

SNR value was quite constant despite of RSSI was different depending on the distance between devices, among other things. This fact may be assumed as LoRa technology is made for large distances. In other words, this technology is able to fix the received signal even it has an extremely low power.

In order to get that across, some of the obtained results are going to be shown below. It is important to keep in mind that the devices were tested in eight different modes. In addition, there are many used attenuations (from 0 dB to 60 dB, step 5). Therefore, to show all the obtained data is not feasible. Having said that, the most noted results will be shown in order to clarify how LoRa technology works.

Table 4-3 Obtained data for mode 6 (DUT inside of RC)

Attenuationn	RSSI (dBm)	SNR (dB)
0	-29	7.2
5	-35	9
10	41	9.8
15	45	7.5
20	-47	8.5

25	-53	10.2
30	-54	8.8
35	-61	9
40	-69	10.8
45	75	8.5
50	-81	8.8
55	-85	6.5
60	-91	10.2

NOTE: attenuation results from used attenuator, that is, it would be necessary to add 26.3 dB (see Positions and stirrer) to that attenuation in order to know real attenuation.

As can be seen, SNR may be high (10 dB approximately) independent of RSSI. So the maximum distance, which can be covered using this technology, may be worked out by studying these results. It is also important to know the minimum required SNR to be able to fix the received data, which is from -7.5 dB to -20 dB, depending on SF. In our case, SF was seven and so the minimum required SNR would be -20 dB. Despite of that, we are going to choose 0 dB as a minimum required SNR in order to be on the safe side.

Once both minimum required SNR and RSSI are known, next step would be to analyse the situation. In other words, we must check the maximum reachable distance given by RSSI and SNR. What is more, we know the added attenuation so we can turn that attenuation into a distance, which attenuates the transmitted signal in the same way. In addition, gateway sensitivity is given that by its manufacturer and so we know how many more decibels the transmitted signal may be attenuated.

To sum up, we are going to work out the maximum reachable distance and so we have to base on the shown data in Table 4-3, among other things. Then, next data must be considered:

- Applied attenuation: 60 dB
- Attenuation results from used RC (868 MHz): 26.3 dB
- Total attenuation: 86.3 dB (60 dB + 26.3 dB)
- Obtained SNR: 10.8 dB

Having into account obtained SNR and the minimum required SNR (we assume 0 dB), the maximum attenuation that transmitted signal can suffer would be: $86.3 \text{ dB} + 10.8 \text{ dB} = 97.1 \text{ dB}$. Picture below shows attenuation depending on the distance in order to calculate the maximum reachable distance.

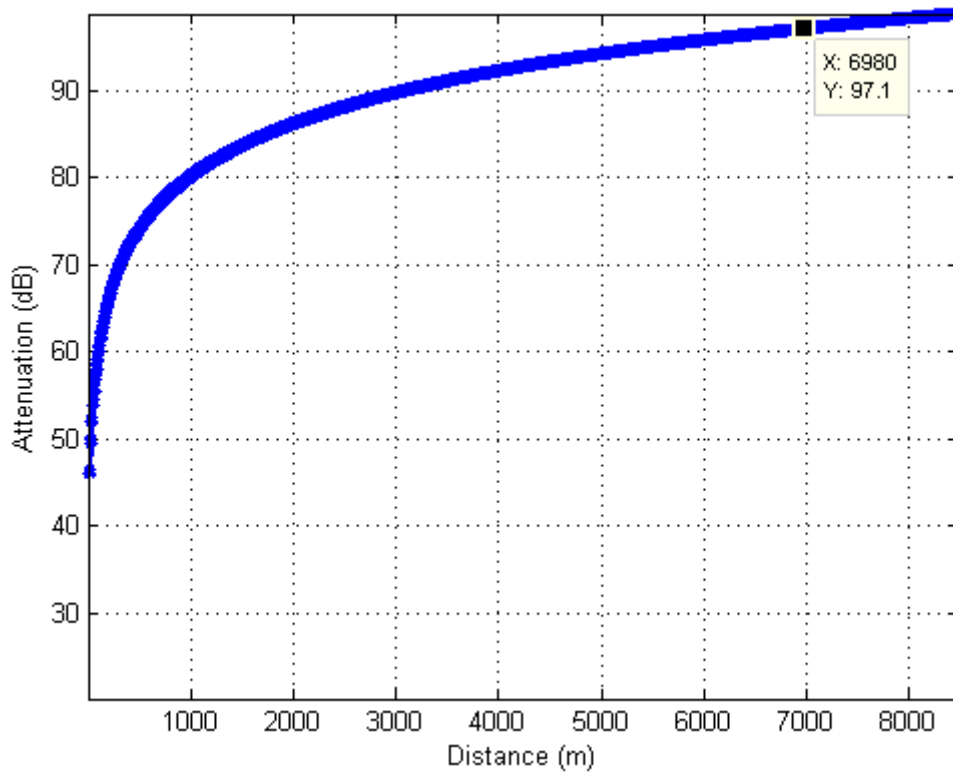


Figure 4.3 Attenuation depending on distance

As can be seen, it is possible to cover until 6980 meters (in a rural environment). The used equation to create Figure 4.3 is:

$$Attenuation = 10 \log_{10} \left(4\pi \left(\frac{distance}{\lambda} \right)^2 \right)$$

where λ is wavelength, which is:

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8}{868 \cdot 10^6} = 34.56 \text{ cm}$$

Throughput

Once we know LoRa technology better and so we know some features such as maximum reachable distance (based on obtained data), we can do next step, that is to say, to measure throughput. It is necessary to have into account that here we are going to evaluate the amount of packets that are correctly forwarded to the server (TTN), so measured throughput is from 0% (zero packets are forwarded) to 100% (all the sent packets are correctly forwarder).

To get this process across, the process is going to be explained in detail. First, a device will be the DUT, meaning it will be properly installed inside of the RC. Therefore, the other used device will be out of the RC. Due to this, this device will be connected with the RC via cable, and so it will use antennas of the inside of the RC to be able to communicate with the DUT.

Having said that, next step is to communicate the devices each other. We did that by OTAA, as it more secure than ABP. Once both devices are already connected, a large amount of packets have to be sent from the RN2483 to LoRa gateway. The bigger amount of packets, better precision to work out the throughput we will obtain.

Packets from RN2483 must be sent one by one, that is to say, packets must be manually sent, as there is not a process for doing that automatically. When a packets is received, we can know some features of the packets such as RSSI, SNR; used SF and so on. However, what we want to know is whether the packet was correctly forwarder or not. This information is reachable as when forwarding packets, all the information is shown.

Final step to work out the throughput. For doing that, the obtained information is analysed and used in order to know both how many packets were received and how many packets were forwarder. Hence, some *sh* codes to achieve this objective. To know more about that, see *Annex C: How to calculate obtained throughput*.

Finally, the obtained results are shown in next graphics:

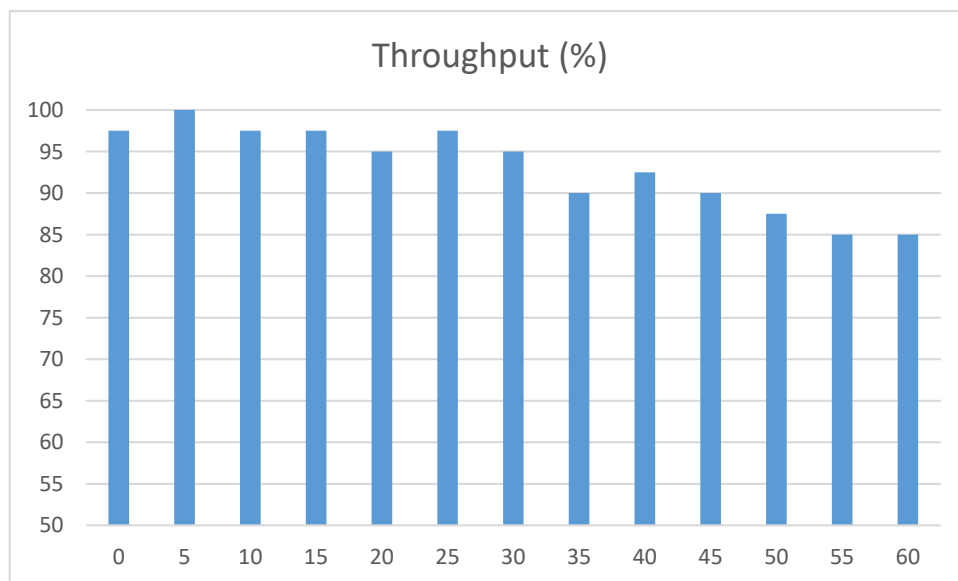


Figure 4.4 Obtained throughput with the DUT inside of the RC

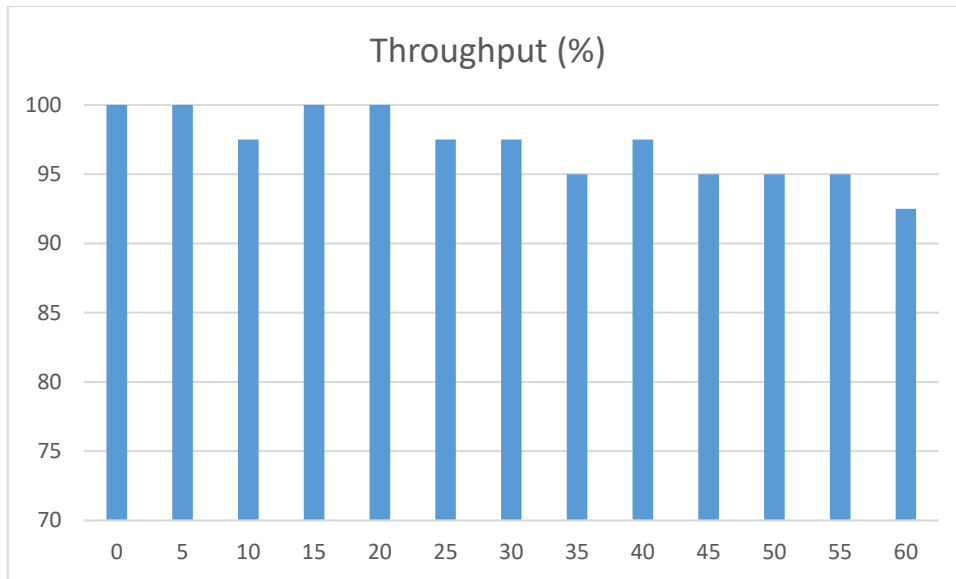


Figure 4.5 Obtained throughput with the LoRa gateway inside of the RC

As can be seen, a high percentage of correctly forwarded packets are obtained despite of a large attenuation is used.

4.2. IPv6 over BLE

4.2.1. Raspberry as an end-device

In this section we may check a IPv6 over BLE connection is properly created. Hence, to verify all the done work (see *Raspberry Pi as an end-device*), we must do a ping using the IPv6 directions result from bt0 interfaces. For doing that, the used command is next one:

```
ping6 -c 4 -I bt0 <IPv6 direction>
```

“ping6” indicates that the direction to use is an IPv6 direction; “-c 4” is due to four packets will be sent, and “-I bt0” indicates that the used interface to send packets will be *bt0*.


```
pi@raspberrypi: ~
bt0: flags=4161<UP,RUNNING,MULTICAST> mtu 1280
    inet6 fe80::b827:ebff:fec3:18ba prefixlen 64 scopeid 0x20<link>
    unspec B8-27-EB-C3-18-BA-00-00-00-00-00-00-00-00 txqueuelen 1000
    RX packets 12 bytes 1296 (1.2 KiB)
    RX errors 0 dropped 5 overruns 0 frame 0
    TX packets 20 bytes 1739 (1.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@raspberrypi:/home/pi# ping6 -c 4 -I bt0 fe80::16:a4ff:fe49:f293
PING fe80::16:a4ff:fe49:f293(fe80::16:a4ff:fe49:f293) from fe80::b827:ebff:fec3:
64 bytes from fe80::16:a4ff:fe49:f293%bt0: icmp_seq=1 ttl=64 time=66.1 ms
64 bytes from fe80::16:a4ff:fe49:f293%bt0: icmp_seq=2 ttl=64 time=88.1 ms
64 bytes from fe80::16:a4ff:fe49:f293%bt0: icmp_seq=3 ttl=64 time=61.5 ms
^C
--- fe80::16:a4ff:fe49:f293 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 61.520/71.919/88.113/11.604 ms
```

Figure 4.6 bt0 interface and ping6

It is easy to check the connection is correctly done, as shown in figure 4.6. Once connection is correctly created, it is possible to make throughput tests. For doing that, we used *iperf3*, which is a free open source, cross-platform command line based program for performing real time network throughput measurements. It is one of the powerful tools for testing the maximum achievable bandwidth in IP networks (supports IPv4 and IPv6).

Therefore, the first step is to install *iperf3* into the router and raspberry pi. Then, we run *iperf3* for IPv6 on both RT-N16 router (which will be the server, that is, the receiver) and raspberry (which will be the client, that is, the transmitter). The obtained results are shown in figures below; first picture is RT-N16 console and next one is raspberry console.

```

root@EMITE_PAN: ~
root@EMITE_PAN:~# iperf3 -6 -s fe80::16:a4ff:fe49:f293%bt0
-----
Server listening on 5201
-----
Accepted connection from fe80::b827:ebff:fec3:18ba, port 56032
[ 5] local fe80::16:a4ff:fe49:f293 port 5201 connected to fe80::b827:ebff:fec3:18ba port 56034
[ ID] Interval           Transfer             Bitrate
[ 5]  0.00-1.00    sec  1.18 KBytes        9.66 Kbits/sec
[ 5]  1.00-2.00    sec  1.18 KBytes        9.66 Kbits/sec
[ 5]  2.00-3.00    sec  2.36 KBytes       19.3 Kbits/sec
[ 5]  3.00-4.00    sec  1.18 KBytes        9.66 Kbits/sec
[ 5]  4.00-5.00    sec  1.18 KBytes        9.66 Kbits/sec
[ 5]  5.00-6.00    sec  2.36 KBytes       19.3 Kbits/sec
[ 5]  6.00-7.00    sec  1.18 KBytes        9.66 Kbits/sec
[ 5]  7.00-8.00    sec  2.36 KBytes       19.3 Kbits/sec
[ 5]  8.00-9.00    sec  1.18 KBytes        9.66 Kbits/sec
[ 5]  9.00-10.00   sec  2.36 KBytes       19.3 Kbits/sec
[ 5] 10.00-11.00   sec  1.18 KBytes        9.66 Kbits/sec
[ 5] 11.00-12.00   sec  2.36 KBytes       19.3 Kbits/sec
[ 5] 12.00-13.00   sec  2.36 KBytes       19.3 Kbits/sec
[ 5] 13.00-13.64   sec  1.18 KBytes       15.1 Kbits/sec
-----
[ ID] Interval           Transfer             Bitrate
[ 5]  0.00-13.64   sec  23.6 KBytes       14.2 Kbits/sec
-----
Server listening on 5201
-----
^Ciperf3: interrupt - the server has terminated

```

Figure 4.7 RT-N16 console when using Iperf3 (I)

```

pi@raspberrypi: ~
root@raspberrypi:/home/pi# iperf3 -6 -c fe80::b827:ebff:fec3:18ba%bt0 -c fe80::16:a4ff:fe49:f293%bt0
Connecting to host fe80::16:a4ff:fe49:f293%bt0, port 5201
[ 4] local fe80::b827:ebff:fec3:18ba port 56034 connected to fe80::16:a4ff:fe49:f293 port 5201
[ ID] Interval           Transfer             Bandwidth          Retr  Cwnd
[ 4]  0.00-1.00    sec  46.0 KBytes        377 Kbits/sec      1   13.0 KBytes
[ 4]  1.00-2.00    sec  15.3 KBytes       126 Kbits/sec      0   14.2 KBytes
[ 4]  2.00-3.00    sec  2.36 KBytes       19.3 Kbits/sec     0   14.2 KBytes
[ 4]  3.00-4.00    sec  1.18 KBytes        9.66 Kbits/sec     0   14.2 KBytes
[ 4]  4.00-5.00    sec  1.18 KBytes        9.66 Kbits/sec     0   14.2 KBytes
[ 4]  5.00-6.00    sec  2.36 KBytes       19.3 Kbits/sec     0   14.2 KBytes
[ 4]  6.00-7.00    sec  1.18 KBytes        9.66 Kbits/sec     0   14.2 KBytes
[ 4]  7.00-8.00    sec  2.36 KBytes       19.3 Kbits/sec     0   16.5 KBytes
[ 4]  8.00-9.00    sec  1.18 KBytes        9.66 Kbits/sec     0   16.5 KBytes
[ 4]  9.00-10.00   sec  2.36 KBytes       19.3 Kbits/sec     0   16.5 KBytes
-----
[ ID] Interval           Transfer             Bandwidth          Retr
[ 4]  0.00-10.00   sec  75.5 KBytes       61.8 Kbits/sec      1
[ 4]  0.00-10.00   sec  23.6 KBytes       19.3 Kbits/sec
-----
iperf Done.

```

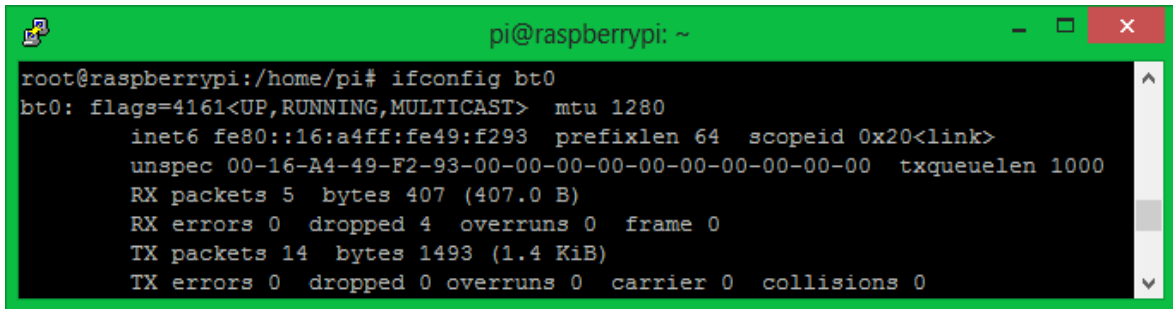
Figure 4.8 Raspberry console when using Iperf3 (I)

When having a look at both figure 4.7 and figure 4.8, we can find out about bitrate, amount of transferred bits and bandwidth.

Finally, when this BLE connection is finished, bt0 interface will disappear in every device. This part has been done in order to check that 6LoWPAN and CoAP support works as we hoped. Next step is to do the same using a BLE device connected with the raspberry via USB, instead of a BLE module from the raspberry. In that way, the raspberry will be a router in the IPv6 over BLE network, as we want.

4.2.2. Raspberry as a router

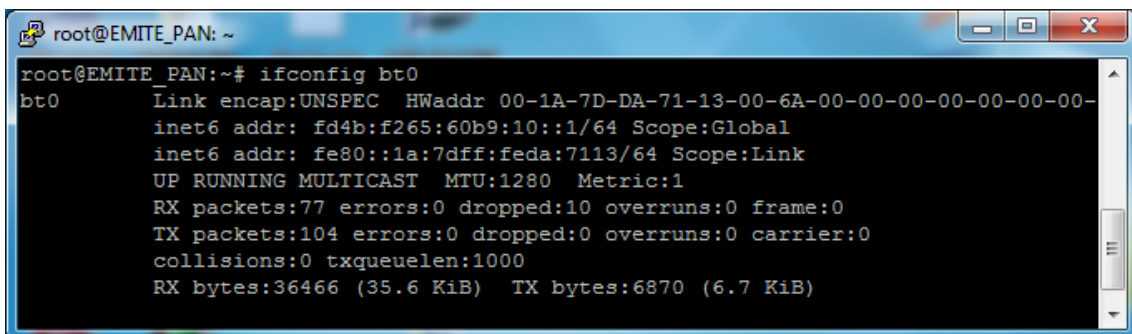
Having said that, we can give way to explain final step. When a BLE connection is made using 6LoWPAN, a new interface appears in each connected device, whose name is *bt0*. This interface is created by 6LoWPAN, and it gives an IPv6 direction for each device. If raspberry works correctly as a router, a *bt0* interface must come up.



```
pi@raspberrypi: ~
root@raspberrypi:/home/pi# ifconfig bt0
bt0: flags=4161<UP,RUNNING,MULTICAST> mtu 1280
    inet6 fe80::16:a4ff:fe49:f293 prefixlen 64 scopeid 0x20<link>
    unspec 00-16-A4-49-F2-93-00-00-00-00-00-00-00-00-00-00 txqueuelen 1000
    RX packets 5 bytes 407 (407.0 B)
    RX errors 0 dropped 4 overruns 0 frame 0
    TX packets 14 bytes 1493 (1.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 4.9 *bt0* interface in raspberry

The obtained result verifies that raspberry is running adequately, as shown in above image. What is more, we can know new IPv6 direction is correct as it is made from MAC address of the controlled dongle, which is DVK-BT850 Laird. The same interface (*bt0*) comes up in the RT-N16 router.



```
root@EMITE_PAN: ~
root@EMITE_PAN:~# ifconfig bt0
bt0    Link encap:UNSPEC HWaddr 00-1A-7D-DA-71-13-00-6A-00-00-00-00-00-00-00-00-
    inet6 addr: fd4b:f265:60b9:10::1/64 Scope:Global
    inet6 addr: fe80::1a:7dff:feda:7113/64 Scope:Link
    UP RUNNING MULTICAST MTU:1280 Metric:1
    RX packets:77 errors:0 dropped:10 overruns:0 frame:0
    TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:36466 (35.6 KiB) TX bytes:6870 (6.7 KiB)
```

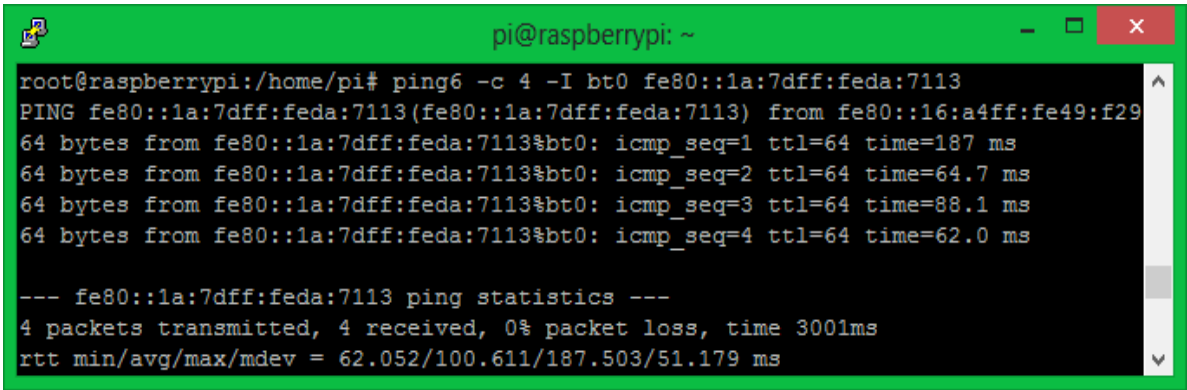
Figure 4.10 *bt0* interface in RT-N16

As shown in figure 4.7 and figure 4.10, we already have two *bt0* interfaces, that is to say, we have two IPv6 directions and so it is possible to communicate both BLE devices each other.

To verify all the done work, we must do a ping using the IPv6 directions result from *bt0* interfaces. For doing that, the used command is next one (if we do it with the raspberry, meaning the Laird is sending):

```
ping6 -c 4 -I bt0 fe80::1a:7dff:feda:7113
```

“ping6” indicates that the direction to use is an IPv6 direction; “-c 4” is due to four packets will be sent, and “-I *bt0*” indicates that the used interface to send packets will be *bt0*.

A terminal window titled 'pi@raspberrypi: ~' with a black background and white text. The window shows the execution of a ping6 command: 'root@raspberrypi:/home/pi# ping6 -c 4 -I bt0 fe80::1a:7dff:feda:7113'. The output displays four successful ping attempts with varying response times: 187 ms, 64.7 ms, 88.1 ms, and 62.0 ms. Below the individual results, a summary line reads: '--- fe80::1a:7dff:feda:7113 ping statistics ---'. The final line of output provides statistics: '4 packets transmitted, 4 received, 0% packet loss, time 3001ms' and 'rtt min/avg/max/mdev = 62.052/100.611/187.503/51.179 ms'. The terminal window has a green title bar and standard window control buttons (minimize, maximize, close) on the right side.

```
pi@raspberrypi: ~
root@raspberrypi:/home/pi# ping6 -c 4 -I bt0 fe80::1a:7dff:feda:7113
PING fe80::1a:7dff:feda:7113(fe80::1a:7dff:feda:7113) from fe80::16:a4ff:fe49:f29
64 bytes from fe80::1a:7dff:feda:7113%bt0: icmp_seq=1 ttl=64 time=187 ms
64 bytes from fe80::1a:7dff:feda:7113%bt0: icmp_seq=2 ttl=64 time=64.7 ms
64 bytes from fe80::1a:7dff:feda:7113%bt0: icmp_seq=3 ttl=64 time=88.1 ms
64 bytes from fe80::1a:7dff:feda:7113%bt0: icmp_seq=4 ttl=64 time=62.0 ms

--- fe80::1a:7dff:feda:7113 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 62.052/100.611/187.503/51.179 ms
```

Figure 4.11 ping6 from the Laird to Dongle CSR 4.0 Dongle

It is easy to check the connection is correctly done, as shown in figure 4.11. Next step is to measure throughput once that a BLE network has already been correctly created. Hence, we are already able to use *iperf3*, which is one of the most powerful tools for testing the maximum achievable bandwidth in IP networks (supports IPv4 and IPv6).

The first step would be to install *iperf3* into the router and the raspberry pi. Then, just run *iperf3* for IPV6 on both RT-N16 router (which will be the server, that is, the receiver) and raspberry (which will be the client, that is, the transmitter), as before. The main difference consist of raspberry is a router now, meaning it has a dongle connected with one of its USB interfaces. Then, that dongle is which is going to transmit whereas the BLE module of the raspberry will be inactive. The obtained results are shown in figures below; first picture is RT-N16 console and next one is raspberry console.

```

PuTTY (inactive)
root@EMITE_PAN:~# iperf3 -6 -s fe80::1a:7dff:feda:7113%bt0
-----
Server listening on 5201
-----
Accepted connection from fe80::16:a4ff:fe49:f293, port 60054
[ 5] local fe80::1a:7dff:feda:7113 port 5201 connected to fe80::16:a4ff:fe49:f29
[ ID] Interval          Transfer      Bitrate
[ 5]  0.00-1.00      sec  1.18 KBytes  9.66 Kbits/sec
[ 5]  1.00-2.00      sec  1.18 KBytes  9.66 Kbits/sec
[ 5]  2.00-3.00      sec  2.36 KBytes  19.3 Kbits/sec
[ 5]  3.00-4.00      sec  2.36 KBytes  19.3 Kbits/sec
[ 5]  4.00-5.00      sec  0.00 Bytes   0.00 bits/sec
[ 5]  5.00-6.00      sec  2.36 KBytes  19.3 Kbits/sec
[ 5]  6.00-7.00      sec  1.18 KBytes  9.66 Kbits/sec
[ 5]  7.00-8.00      sec  2.36 KBytes  19.3 Kbits/sec
[ 5]  8.00-9.00      sec  2.36 KBytes  19.3 Kbits/sec
[ 5]  9.00-10.00     sec  1.18 KBytes  9.66 Kbits/sec
[ 5] 10.00-11.00     sec  1.18 KBytes  9.66 Kbits/sec
[ 5] 11.00-12.00     sec  2.36 KBytes  19.3 Kbits/sec
[ 5] 12.00-13.00     sec  2.36 KBytes  19.3 Kbits/sec
[ 5] 13.00-13.54     sec  1.18 KBytes  17.8 Kbits/sec
-----
[ ID] Interval          Transfer      Bitrate
[ 5]  0.00-13.54     sec  23.6 KBytes  14.3 Kbits/sec
-----
Server listening on 5201
-----
receiver

```

Figure 4.12 RT-N16 console when using Iperf3 (II)

```

pi@raspberrypi: ~
root@raspberrypi:/home/pi# iperf3 -6 -c fe80::16:a4ff:fe49:f293%bt0 -c fe80::1a:
Connecting to host fe80::1a:7dff:feda:7113%bt0, port 5201
[ 4] local fe80::16:a4ff:fe49:f293 port 60056 connected to fe80::1a:7dff:feda:7
[ ID] Interval          Transfer      Bandwidth    Retr  Cwnd
[ 4]  0.00-1.00      sec  46.0 KBytes  377 Kbits/sec  1    13.0 KBytes
[ 4]  1.00-2.00      sec  15.3 KBytes  126 Kbits/sec  0    14.2 KBytes
[ 4]  2.00-3.00      sec  2.36 KBytes  19.3 Kbits/sec  0    14.2 KBytes
[ 4]  3.00-4.00      sec  2.36 KBytes  19.3 Kbits/sec  0    14.2 KBytes
[ 4]  4.00-5.00      sec  0.00 Bytes   0.00 bits/sec  0    14.2 KBytes
[ 4]  5.00-6.00      sec  2.36 KBytes  19.3 Kbits/sec  0    15.3 KBytes
[ 4]  6.00-7.00      sec  1.18 KBytes  9.66 Kbits/sec  0    16.5 KBytes
[ 4]  7.00-8.00      sec  2.36 KBytes  19.3 Kbits/sec  0    16.5 KBytes
[ 4]  8.00-9.00      sec  2.36 KBytes  19.3 Kbits/sec  0    16.5 KBytes
[ 4]  9.00-10.00     sec  1.18 KBytes  9.66 Kbits/sec  0    16.5 KBytes
-----
[ ID] Interval          Transfer      Bandwidth    Retr
[ 4]  0.00-10.00     sec  75.5 KBytes  61.8 Kbits/sec  1
[ 4]  0.00-10.00     sec  23.6 KBytes  19.3 Kbits/sec
-----
iperf Done.
sender
receiver

```

Figure 4.13 Raspberry console when using Iperf3 (II)

As can be seen, results are like before (see *Raspberry as an end-device*). When having a look at both figure 4.7 and figure 4.8, we can find out about bitrate, amount of transferred bits and bandwidth. In fact, obtained values for those parameters are, in reality, the same.

Finally, when this BLE connection is finished, bt0 interface will disappear in every device. This part has been done in order to check that *radvd* and *flex* works as well as we wanted.

5. Conclusions

5.1. LoRa technology

Now we are going to analyse both obtained results and limitations in this technology. First, it is important to remember LoRa was created to communicate devices that are far from each other. Despite of the transmitted signal is a long time on the air (at least, it is compared to BLE), this technology is one of the most secure solutions we have ever seen for IoT.

This idea may be easily seen when transmitting by OTAA, as a mote have to know some parameters such as *eui* of the device. In other words, when the transmitted packet arrives to a undesired gateway LoRa (a gateway that is not the ours one), that LoRa gateway will not forward the packet to a server, as its *eui* and the *eui* of our LoRa mote will not match.

Nevertheless, when using ABP the situation is quite different. It is true that this type of connection is really good when a device have not a good connection. In addition, ABP offers a more flexible connectivity as it is not required to join the network to send data. However, if the encryption key is found, it can be extracted and cloned. Hence, anyone could know the information that is being transmitted.

One of the most important advantages offered by LoRa is distance. Two LoRa devices can communicate each other for distances up to various kilometres. In fact, we attenuated a lot the transmitted signal and the devices could be still connected with each other.

This fact is due to many factors; bandwidth is one of them. LoRa uses a little bandwidth and so it achieves an incredible sensitivity. A little bandwidth offers a minor bitrate, but a minor noise power, too. Then, it makes to fix the received signal possible, even the signal has a really low power.

Having said that, it is important to find out the reliability of this technology, that is to say, the success percentage of the transmitted packets. In other words, we want to know whether LoRa has an amount of bad forwarder packets high or not. To know that we tested a LoRa device in a RC, and we got some throughput values (see *Throughput*), in percentage. These results proved its reliability, as the success percentage always was near 100%.

5.2. IPv6 over BLE

BLE is a well-known technology whose main advantages are its incredible low power consumption. It is necessary to modify other parameters such as bandwidth in order to achieve that goal, meaning BLE also has a certain number of disadvantages. One of these ones is its connectivity, as it had to be obligatorily created by a slave-master relation.

Hence, this obligatory nature implies a series of advantages and disadvantages. For instance, some of these advantages are:

- One master may control some slaves
- A master is able to synchronise with each slave in order to establish a communication correctly

On the other hand, the main disadvantages are:

- Slaves cannot communicate each others
- It is necessary a spare master device, as the communication will not work if the master device is broken.
- Slaves are able to communicate with an unique master

The arrival of 5G implies the necessity of communicate every device with every device, that is, master-slave communication is not enough. Therefore, it is required every device has internet access and so IP direction, either IPv4 or IPv6. Hence, to give an IP direction to BLE devices is one of the most efficient ways to reach internet connection for every BLE device.

This advance creates a huge number of functionalities for BLE devices. In other words, thanks to give IP direction to BLE devices an incredible number of applications can be carried out. To sum up, we are sure this advance will be the cornerstone of many future applications.

6.Future Lines of Investigation

6.1.LoRa

As was said in LoRa, to create a LoRa network is not an impossible objective. In fact, all the LoRa devices (gateways and motes) are already prepared for that. A quite noted part was to study the reliability of the created LoRa network, that is to say, success percentage of forwarded packets.

This measurement is one of the most important parameters, as we will be able to find out whether our LoRa network works as we desire. In other words, if that parameter is near zero, our network is not working as best as possible. However, if that parameter is near a hundred percent implies our network works correctly.

To measure this parameter we have to send a large amount of packets. In this project, this process was manually done and so we had to issue one command for each sent packet. Therefore, one of the possible future lines to LoRa technology would be automating this process, meaning to be able to send as many packets as we want by issue just one command.

Another process that was manually done is extraction data. When a packet is forwarded, we can know some features such as RSSI and SNR through TTN, among other ones. The issue is as follows; to copy that information and to paste it in a text document for instance, we have to do it manually, that is, copy by selecting the desired text and paste it in the text document. Therefore, we would already have the desired information in our PC.

Next step is to write the values of every feature (e.g. RSSI) in an excel file in order to represent it by means of a graphic. This whole process implies a lot of time and it could be improved if it was automatically done. For doing that, some software such as *Matlab* or *Python* can be used as they offer a large amount of functionalities.

Hence, a program (using either *Matlab* or *Python*) would have to be created in order to choose the desired information and copy it in a text file, in a organized way. Thus, all this data could be easily accessible using a program such as *Matlab* and then we would already have our properly sorted information, meaning we could represent it easily by means of a graphic.

6.2.BLE with IP

When creating a network with BLE devices, a big problem came up; BLE devices have not any IP direction, neither IPv4 nor IPv6. This fact is a large brake for communication, as BLE devices are not able to communicate with all the devices. This problem could be solved by generating an IPv6 direction for every BLE

device. Therefore, now a BLE device may communicate with internet and so every device, which has internet access.

In conclusion, we have a huge amount of future lines due to that. Now that BLE devices can communicate with the entire network (we must remember 5G is coming and so this advantage is more important over time), a vast number of functionalities may be conducted and we have a future line for each new functionality. Hence, we are going to name some of the possible future lines, although we have to keep in mind that there are infinite future lines of investigation.

For example, future IPv6-based BLE mesh solutions will be able to leverage existing end-to-end security functionality for IP-based protocols. A first option is Transport Layer Security (TLS), which is commonly used to secure HTTP communications. Non-IP-based BLE mesh networks will need to develop end-to-end security functionality that might be equivalent to the approaches described.

Another possible future line would be that IPv6-based BLE mesh solutions require the use of data channels for the transmission of both routing and data messages. Among these, static solutions do not appear to be solid candidates given the intrinsic dynamics of mesh networking. Therefore, solutions based on dynamic routing that use data channels for the transmission of both routing and data messages, are currently the most promising ones to unleash the potential of BLE mesh networks.

7. Bibliography

- [1] “Basic Methods in Antenna Measurements”, http://acad-antenna.co.il/pics/ACAD/_thumbs/08_Measurements.pdf, 2012, [Online accessed 12-april-2019].
- [2] “Testing in a Reverberation Chamber”, <https://www.nts.com/services/testing/emc/reverberation-chamber-testing/>, 2019, [Online accessed 15-april-2019].
- [3] 3GPP TR 37.976, “Measurement of radiated performance for Multiple Input Multiple Output (MIMO) and multi-antenna reception for High Speed Packet Access (HSPA) and LTE terminals”, 2014.
- [4] “Wirnet Station”, <https://www.kerlink.com/product/wirnet-station/>, 2019, [Online accessed 9-july-2019].
- [5] “RN2483 LoRa Technology Module Command Reference User’s Guide”, 2018.
- [6] “Graphic User Interface (GUI) reference guide”, EMITE. MIMO Analyzer F-Series Handbook v3.12, 2019.
- [7] “Internet of Things for Smart Cities”, <https://ieeexplore.ieee.org/abstract/document/6740844>, 2019, [Online accessed 11-July-2019].
- [8] “Bluetooth IoT Applications: From BLE to Mesh”, <https://www.iotforall.com/bluetooth-iot-applications/>, 2018, [Online accessed 12-july-2019].
- [9] “Wireless Technologies for IoT”, <https://electronicsforu.com/technology-trends/tech-focus/wireless-technologies-iot>, 2018, [Online accessed 15-july-2019].
- [10] A. Woods, “USQ ePrints,” n.d. June 2006. [Online]. Available: https://eprints.usq.edu.au/2367/1/WOODS_Andrew-2006.pdf. [Accessed 10-april-2019].
- [11] “E500”, <http://emite-ing.com/en/solutions/test-systems/e-series>, 2018, [Online accessed 16-july-2019].

Annexes

Annex A: Different Environments to Test a Device

The path towards a realistic and cost-effective 5G over-the-air (OTA) testing scenario is not clear yet. With a tremendous pressure on 5G standards development, network deployments and device manufacturing, the realistic answers that a 5G Over-The-Air (OTA) test system should provide are far from being obtained to date. This article identifies some of the challenges ahead with possible solutions.

In the race towards satisfying the expected growth, connectivity, availability and reliability, 3GPP and CTIA have emerged as the standardization bodies that will enable adequate OTA testing of the new technologies prior to massive deployment. With the experience of 4G OTA testing standardization history in mind, there are more questions about the capabilities of consensus-driven OTA testing standardization and how it works to solve the real challenges of 5G deployment and operation.

With new engineering concepts in 5G like MIMO (Multiple Input, Multiple Output), beamforming and the generalized use of millimetric wave's frequencies, 5G OTA testing is clearly the challenge of the decade for wireless communications and a key milestone for 5G deployment and operational success. Nowadays, there are two main kinds of chamber where all devices are tested, which are anechoic and reverberation chamber (RC).

A.1 Anechoic Chamber (AC)

An anechoic (which means non-reflective, non-echoing, or echo-free) chamber is a room that is designed to completely absorb reflections of electromagnetic waves.

Anechoic chambers are enclosed by an external metallic shielding, which provides isolation from the outside environment. Used to conduct experiments, this electromagnetically quiet room is constructed with echo suppression features and with effective isolation from RF noises that are present in the external environment. This combination allows workers to exclusively receive direct signals (no reverberant sounds), simulating being inside an infinitely large room. This ensures the integrity of the experiment carried out inside as it is not influenced by external or internal reflected noise.

Next figure shows the plain view of the rectangular anechoic chamber that uses pyramidal absorber as main RF absorbing material.

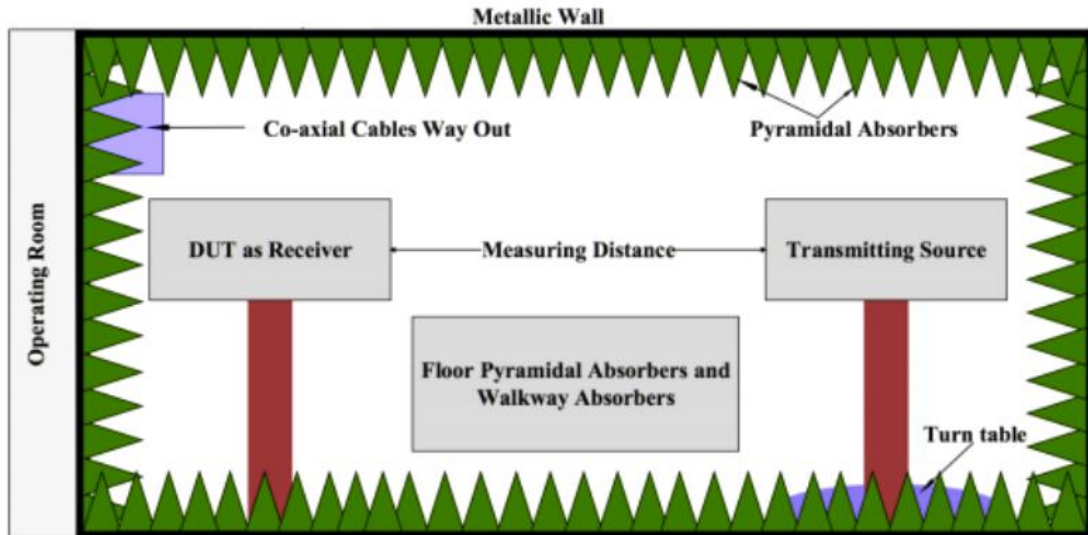


Figure A.1 View of an Anechoic Chamber with pyramidal absorber

To test an equipment inside the anechoic chamber, a transmitting source that emits EM signal and a device under test (DUT) as a receiver is required to receive the transmitted signal. The transmitter and receiver could be any type of antenna. The transmitted signal field strength is sampled by the receiver antenna that can be used for calculating the desired results [10].

One of the most important requirements for testing is that the AUT to be in the quiet zone (QZ), that is to say, in a place inside of the AC where the amplitude and the phase ripples are minimum. The QZ is typically 50% to 60% the aperture of the reflector. The imperfections of the field in the QZ are measured in terms of phase errors, ripple-amplitude deviations, and taper-amplitude deviations. Acceptable deviations for a QZ are:

- less than 10% phase error
- less than 1 dB ripple
- taper amplitude deviations

The ideal condition for measuring the far-field pattern and antenna gain is an illumination by a uniform plane wave. This is a wave, which has a plane wave front with the field vectors being constant over an area that extends well beyond the aperture of the antenna under test (AUT). For example, the E field vector of a uniform non-attenuating plane wave propagating in the +z-direction is described by the 1-D wave expression

$$E(z) = \rho E_i e^{-jkz} \quad (1)$$

Here, ρ is the wave polarization vector, which must remain constant within the volume of the AUT. The same holds for the magnitude E_i , which must remain constant across the AUT aperture.

In practice, antennas generate far fields in 3-D space which are closely approximated by spherical wave fronts when the observation point is sufficiently far from the source. In addition, at large distances from the source antenna, the curvature of the phase front is small at the aperture of the AUT and it is well approximated by a uniform plane wave.

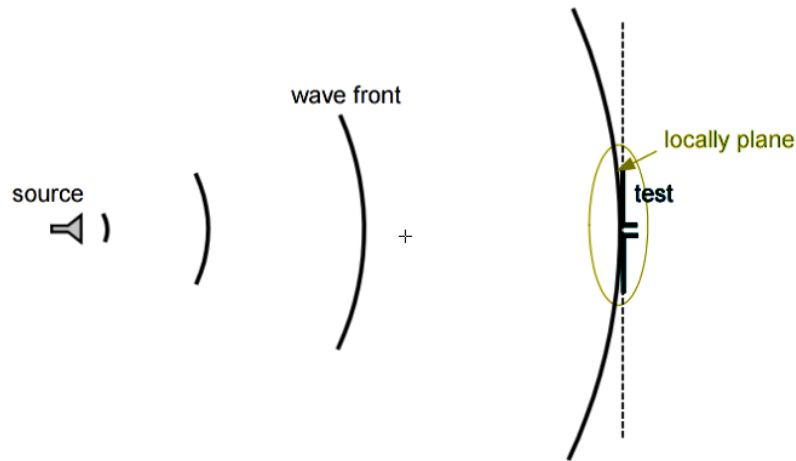


Figure A.2 Locally plane wave front

If the distance from the source is equal or greater than the inner boundary of the far-field region $R_{min} = 2D_{max,Tx}^2/\lambda$, then the maximum phase difference between the actual incident field and its far-zone approximation (remember the 1st order binomial approximation $R \approx r - D_{max,Tx}/2$) does not exceed $e_{max} \approx 22.5^\circ = \frac{\pi}{8} rad$. Here, $D_{max,Tx}$ is the maximum dimension of the source (or transmitting) antenna. Conversely, we can show that if $D_{max,Rx} \equiv D_{max}$ is the maximum dimension of the receiving AUT, a distance

$$R_{min} = 2D_{max}^2/\lambda \quad (2)$$

from the source of a spherical wave ensures that the maximum phase difference between a plane wave and the spherical wave at the aperture of the AUT is $e_{max} \approx 22.5^\circ = \frac{\pi}{8} rad$. Consider a source of a spherical wave and an AUT located a distance R away.

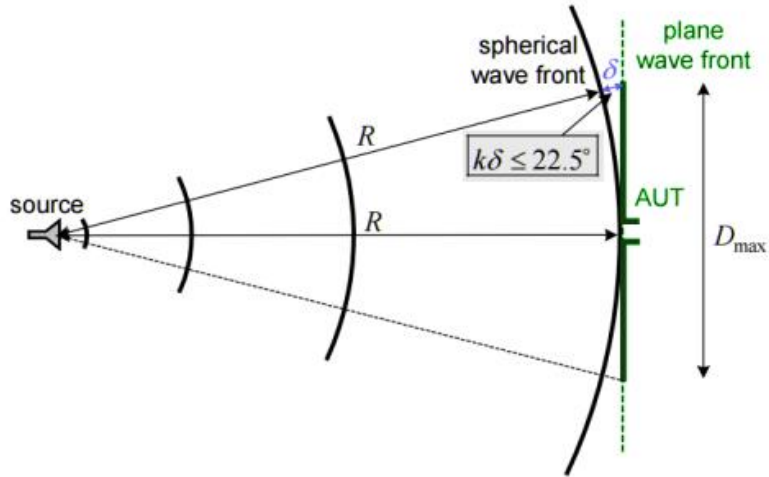


Figure A.3 Phase difference between a spherical wave and a plane one

The largest phase difference between the spherical wave and the plane wave appears at the edges of the AUT, which corresponds to the difference in the wave paths δ . This phase difference must fulfil the requirement:

$$k\delta \leq \pi/8 \quad (3)$$

The difference in the wave paths δ is determined by noticing that

$$(R + \delta)^2 = R^2 + (D_{max}/2)^2 \quad (4)$$

The real-positive solution of this quadratic equation for δ is

$$\delta = \sqrt{R^2 + (D_{max}/2)^2} - R \quad (5)$$

Next, the above expression is approximated by the use of the binomial expansion (the first two terms only) as

$$\delta = R \left[\sqrt{1 + \left(\frac{D_{max}}{2R}\right)^2} - 1 \right] \approx R \left[1 + \frac{1}{2} \left(\frac{D_{max}}{2R}\right)^2 - 1 \right] = \frac{D_{max}^2}{4R} \quad (6)$$

The minimum distance from the source of the spherical wave is now determined from the requirement in (3),

$$k \frac{D_{max}^2}{4R} = \frac{2\pi}{\lambda} \frac{D_{max}^2}{4R} \leq \frac{\pi}{8} \quad (7)$$

Thus,

$$R_{min} = 2 \frac{D_{max}^2}{\lambda} \quad (8)$$

It is now clear that the antenna far-field characteristics must be measured at a sufficiently large distance between the source antenna and the AUT [1].

Therefore there a series of challenge in antenna measurements, which we must be capable of confronting as well as possible, and they are:

- affected by unwanted reflections
- the instrumentation is expensive
- very complicated when a whole antenna system (e.g., on-craft mounted antenna) is to be measured
- often require too large separation distances
- indoor sites cannot accommodate large antenna systems
- outdoor sites have uncontrollable EM environment, which, besides all, depends on the weather

A.2 Reverberation Chamber (RC)

An electromagnetic reverberation chamber (RC) (or Mode-Stirred Chamber, MSC) is where the whole chamber is made to resonate. They are predominantly used as a cavity resonator to perform radiated immunity testing. Due to the high Q-factor of the chamber and the almost complete reflectivity of the walls and floor, an electromagnetic field of a given strength may be created using a much smaller power amplifier compared to that needed in a SAC.

Reverb chambers are useful for radiated susceptibility, radiated emissions (total radiated power), shielding effectiveness, and many other troubleshooting scenarios. Reverb chambers are random in polarity, which makes it challenging in determining directivity of RF energy.

Testing multiple field levels on a system, such as outside the pressure vessel level and inside the pressure vessel level, can be difficult; all equipment in the chamber is exposed to the same field. There are limitations on pulse width due to a high Q (efficient) chamber having large amount of stored energy. If you have small, simple equipment, single aspect angle tests may be faster and sufficient for test coverage [2].

For a future Multi-antenna OTA measurement standard it is important to have a fast and repeatable test method to evaluate and compare multi-antenna devices in the environments and under the conditions where most people will use them.

The overwhelming majority of calls/data connections with mobile phones are made indoors and in urban areas which can be very well represented by the RC. These environments are well characterised by multipath and 3D distribution of the communication signals and it makes sense to use the RC for optimizing/evaluating devices with both single and multiple antenna configurations to be used indoors and in urban areas [3].

Having said that, the best place to test a device will be a place where the test conditions to be just like real conditions, that is to say, a place that multipath and 3D distribution of the communication signals are kept in mind. Therefore, RC

represent one of the best ways to achieve this purpose. Figure 1.9 shows us how the inside of a reverberation chamber is, including all the required devices for testing.

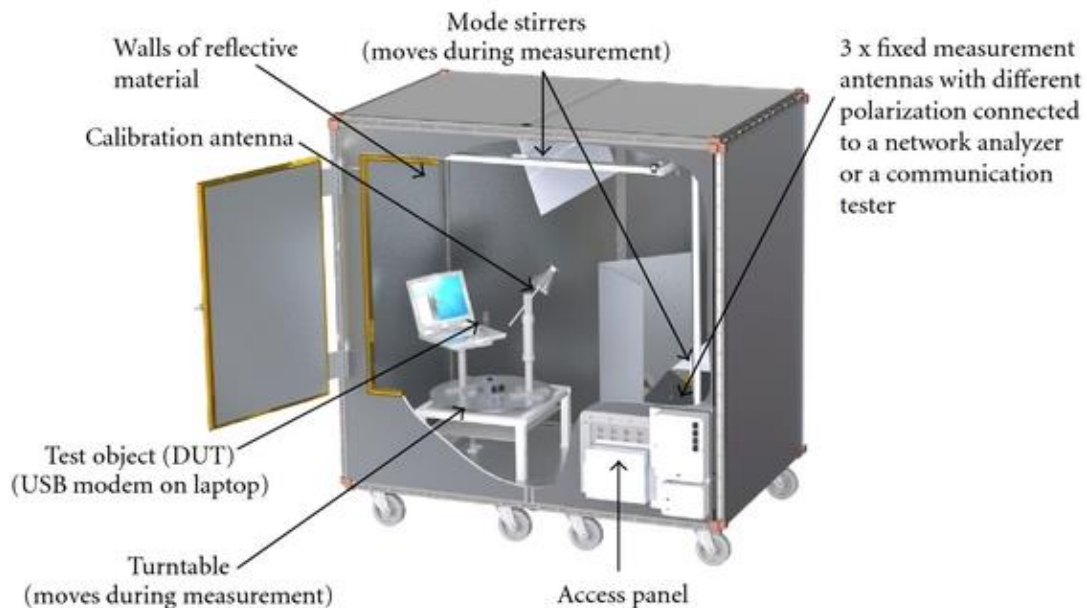


Figure A.4 Inside of a reverberation chamber for measurements

The test setup for testing UE receiver diversity performance is composed of a base station emulator, a RC equipped with fixed BS wall-mounted antennas, a switch to direct the base station signal to/from one of the BS wall mounted antennas, mechanical metallic stirrers and a rotating platform to hold the DUT (figure 1.9).

Reverberation chambers have no quiet zone. As long as the DUT is placed at least 0.5 wavelengths from the wall or metallic stirrers the result will be the same within the standard deviation of the chamber.

Mechanical stirrers and switching among different fixed BS wall-mounted antennas (monopoles used for polarization stirring) allow simulating the Rayleigh fading at each antenna of the terminal inside the chamber. Accuracy may even be increased by rotating the platform holding the device.

Each position of the mechanical stirrers for each position of the platform and each fixed BS antenna, represents a point of the Rayleigh distribution in terms of receive power on the device antennas. In that way a Rayleigh fading is artificially created.

In order to calibrate the RC a broadband antenna can be used to measure the losses in the chamber with a network analyser. This takes less than 10 minutes.

Because of these advantages, CTIA RCSG is working on a standard methodology for reverberation chamber calibration.

It is also necessary to say that there are not any active electronics in the measurement path that needs to be calibrated. Reflections in turntables, cables, doors and so on, do not degrade accuracy. Reflections increase the richness of the channel in the RC.

In order to create a Rayleigh fading environment, there are three types of parameters, which can be set using a tool on a computer plugged to the chamber [3]:

- Antenna among the 3, installed at the top of the cavity with different polarizations, is chosen
- Turning the platform that holds the DUT
- The 2 metallic stirrers near the walls can be moved on their axes

A.3 Advantages and disadvantages of anechoic and reverberation chamber

The benefits to reverberation testing are numerous. RF is applied to all exposed sides of the device under test (DUT) during a full 360° turn of the paddle, instead of a single side. For direct illumination testing, many standards require that all apertures of the DUT be illuminated. On complex items, this can be difficult – even impossible. Window effects testing required when applying direct illumination, is not required during reverb testing because the field intensities are constantly changing.

Test repeatability is much easier to obtain in a reverb chamber with proper processes, and running the test is much less complex than a single aspect angle test. Antenna distance, aim (focus), 3 dB beam width, location of the field probe, EUT layout, and location of the EUT in the working volume are all less of a factor in the repeatability of test.

Because of all these features, a RC is not considered as the best environment for testing some KPIs such as Effective Isotropic Radiated Power (EIRP) or Effective Isotropic Sensitivity (EIS). Perhaps, it is not too easy to realize why this affirmation is said. To understand it, it is necessary to see what inside of a RC is happening when a test is carried out.

The DUT that is being tested in a RC receives the signal sent by the BS from everywhere due to reflections. Then, it is impossible to know whether the received energy is coming from beam peak direction or not. Until now, it was impossible to know both EIRP and EIS using a RC but, thanks to some algorithms that allow for distinguishing between energy coming from beam peak direction and energy coming from reflection in a wall, we actually can do that.

Having said this, the main problem with RC to know EIRP and EIS consist in these algorithms need a large amount of measurements, in order to do all the

required mathematician operations. Hence, to test a device with a RC can take too much time compared to using an AC, at least, for these two KPIs.

As it is seen, RC does not seem very useful if and when distinguishing between the energy coming from the beam peak direction and other directions is required, although it is not impossible. However, if we want to know a KPI and to work out it before condition is not necessary, RC may be incredibly useful.

This happens with some KPIs, such as Total Isotropic Sensitivity (TIS) and Total Radiated Power (TRP). The first parameter let us to know the amount of energy that a device requires to demodulate it correctly. The second one is the total energy that a device transmit when it is sending a signal, whatever it is. Both TRP and TIS are important parameters, as it will explain later.

TRP informs us about how much energy a device transmits, and this data indicates whether a device can be come out on the market or not, as there is a legal limit. What is more, if a TRP is known, it is possible to work out other parameters of the device such as antenna gain, among others. Therefore, we can also know the EIRP of our device is its TRP and its antenna gain are already known. This whole thing makes to know our devices easier.

With TIS, the same thing occurs; it is necessary to know it as not only a device is not always receiving a signal from beam peak direction, but also the arrival signal can arrive our device by a direction, where the antenna gain is minimum. In this case, EIS is not important as our device can have a really good EIS, but the signal does not arriving by that direction. Because of this, TIS has become one of the most important parameter for all devices.

We also have to keep in mind that signal from every single directions is arriving our device, due to reflections which are made in our environment (we usually are surrounded by walls, trees, buildings, people and so on, meaning we are in Rayleigh environment, not in Rice environment). Therefore, we may affirm that TIS offer a better information or, at least, a more realistic information about our device than EIS and so, knowing the TIS of our device we can assume whether it our device will work properly or not, for an arrival signal power.

Once that the importance of these parameters has already been explained, it will be easier to understand why a RC can be very useful in spite of this kind of chamber is slower to know some parameters, as it was previously said. To know TRP and TIS, a RC is much faster than an AC as the amount of measurements that a RC needs is much smaller. This advantage is due to reflections; when a signal is generated, the energy of this signal always reach the DUT, either with or without reflections.

When calculating TRP, all the energy arrive at the receiving antennas. If this amount of energy is known, it is possible to work out the TRP of the DUT as we

also control the environment features. Nevertheless, using an AC this situation change a lot. For instance, to measure a TRP of a device, it is necessary to take into account a series of measurements, which give us the value of the EIRP of each measured point. It seems quite easy to understand why an AC is much slower than a RC to measure KPIs such as TRP and TIS.

In order to get this across, we are going to probe it mathematically for both TRP and TIS. Evaluating TRP with a RC we need to make a number of measurements, which will not be larger than 50. In fact, the real number of measurements is much smaller, but we are going to use 50 to clarify this situation. With an AC, the number of measurements will depend on the amount of points that we use to evaluate EIRP, as we will do the measurements on a sphere, as next image shows.

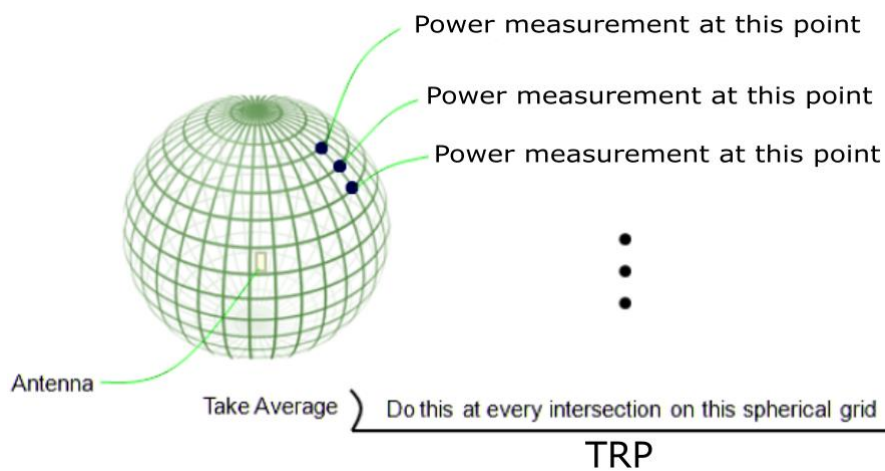


Figure A.5 Measuring TRP with an AC based on EIRP measurements

As we can see, the measurements must be done on an imaginary sphere that is composed of a set of points, where the receiver antennas will be. The DUT will be in the centre of this sphere, and the distance of this sphere will be, at least, the minimum distance for the measurements to be done in far field.

If we want to know how many measurements have to be done or, in other words, how many points the sphere contains, we can find out it by next equation, which establish the relation between TRP and EIRP.

$$TRP = \frac{1}{4\pi} \int_0^{2\pi} \int_0^{\pi} EIRP(\theta, \varphi) \sin(\theta) d\theta d\varphi$$

Where θ is the azimuth variation between two measurement points that are adjacent and φ is the elevation variation. Obviously, this equation related TRP with EIRP in an exact way, but it does not give us the number of equation we would have to do to know the TRP value of a device. To find out that, the equation must be considered in a discrete dimension and so, it would be like next one:

$$TRP \approx \frac{\pi}{2NM} \sum_0^{N-1} \sum_0^{M-1} (EIRP_{\theta}(\theta_n, \varphi_m) + EIRP_{\varphi}(\theta_n, \varphi_m)) \sin(\theta_n)$$

Where $EIRP_{\theta}$ is the obtained value of EIRP in horizontal polarization and $EIRP_{\varphi}$ is the obtained value of EIRP in vertical polarization. As we can see, we must do $N \times M$ measurements to get a TRP value. For example, if an azimuth and elevation variation of 5° are done, we will have $180/5 - 1 = 35$ horizontal measurement circles surrounding the DUT, just as $360^{\circ}/5^{\circ} = 72$ measurement points for each horizontal circle. In addition, we must add up two more points, which are: the point on top of the DUT and the point under the DUT, although there is not any azimuth variations for these both cases. Finally, we may calculate how many measurements we have to do, and it is: $35 \times 72 + 2 = 2522$ measurement points, a number much larger than 50, which is the obtained value using a RC. In other words, an AC to find a TRP value takes too much time, what is unfeasible.

Nevertheless, it is not the unique KPI that is more fastly measured with RC than AC. The same thing happens with TIS. As it is already known, we can obtain TIS from EIS following next expression:

$$TIS = \frac{4\pi}{\int_0^{2\pi} \int_0^{\pi} \left(\frac{1}{EIS_{\theta}(\theta, \varphi)} + \frac{1}{EIS_{\varphi}(\theta, \varphi)} \right) d\theta d\varphi}$$

As before, this is the equation that relates TIS with EIS in a exact way. However, what we want to know is the number of measurements that we have to do to obtain TIS by calculating EIS several times. This number can be known if we work in discrete dimension and so, before equation would become like next one:

$$TIS = \frac{4\pi}{\sum_0^{N-1} \sum_0^{M-1} \left(\frac{1}{EIS_{\theta}(\theta_n, \varphi_m)} + \frac{1}{EIS_{\varphi}(\theta_n, \varphi_m)} \right) d\theta d\varphi}$$

Where EIS_{θ} is the obtained value of EIS in horizontal polarization and EIS_{φ} is the obtained value of EIS in vertical polarization. Next picture clarifies how all these points would be surrounding the DUT.

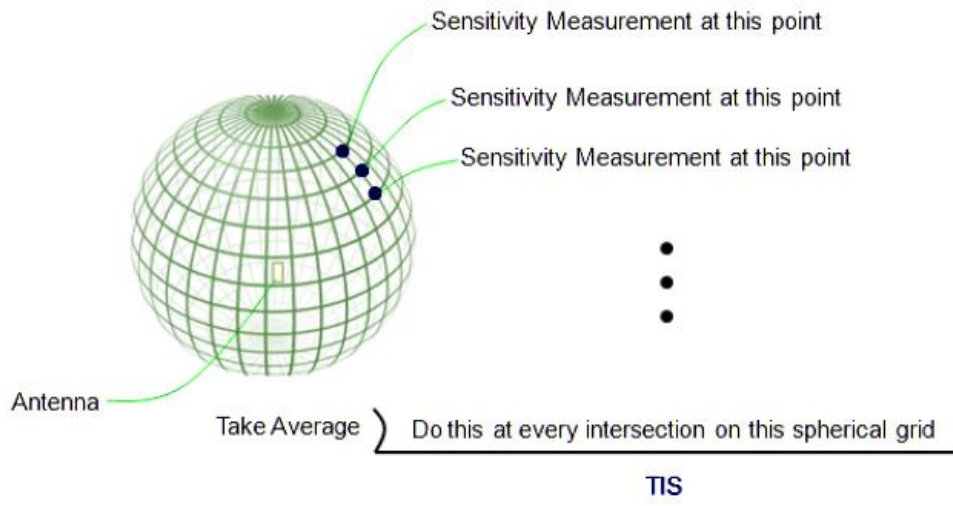


Figure A.6 Measuring TIS with an AC based on EIS measurements

Following before example, we would have a variation of 5° for both vertical and horizontal polarizations and so, we would have 35 horizontal measurement circles surrounding the DUT, just as 72 measurement points for each horizontal circle. In addition, we would have to add up two more points, which are: the point on top of the DUT and the point under the DUT, having into account there is not any azimuth variations for these both cases. Finally, we calculate the required number of measurements to calculate TIS, and it is: $35 \times 72 + 2 = 2522$ measurement points. From here, we can conclude that this way to find a TIS value takes an amount of time, what is unfeasible.

Annex B: Utilized Reverberation Chamber

To analyse some devices, a reverberation chamber was required. In our case, we used a chamber known as E500, as it is able to measure throughput (that is what we want) for frequencies from 690 MHz to 6 GHz.



Figure B.1 E500 Reverberation Chamber

Its dimensions are: 2750 mm (length) x 1545 mm (width) x 2000 mm (height). The E500 is a multicavity mode-stirred source-stirred reverberation chamber that provides unique up to 8x8 passive mode MIMO measurement capabilities, active MIMO Over-The-Air (OTA) 3GPP/CTIA figures of merit for WLAN (including ac), such as [11]:

- Total Radiated Power (TRP)
- Total Isotropic Sensitivity (TIS)
- MIMO Throughput (TPUT)
- Power Delay Profile (PDP)
- Path Loss (PL)
- TCP/UDP Throughput versus time

Furthermore, this parameters may be calculated for a wide variety of Rayleigh, Rician and other fading environments, all united in one single and intuitive interface.

In addition, E500 efficiency is very similar to anechoic chamber efficiency, as shown in next picture.

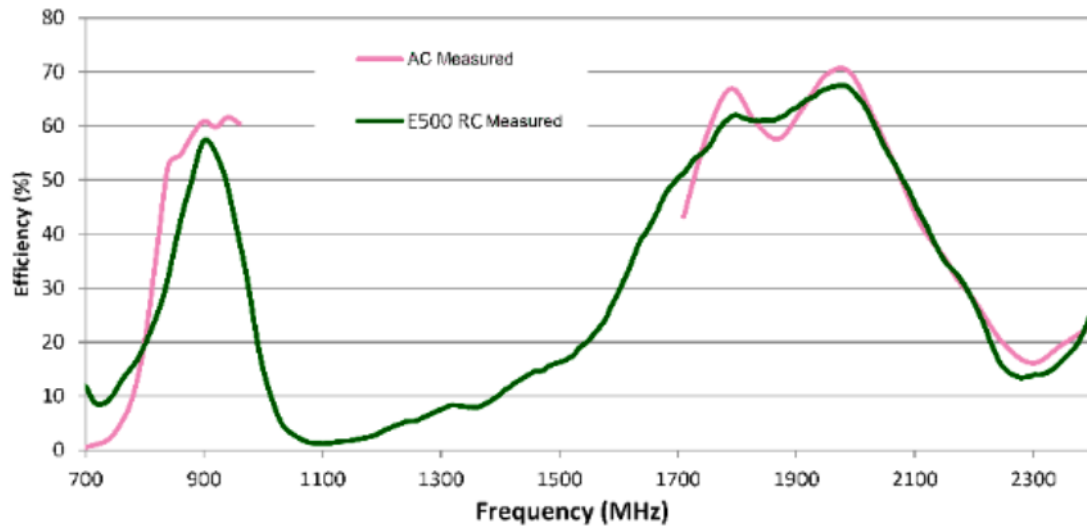


Figure B.2 Efficiency(%) depending on frequency for AC and RC

As can be seen in Figure B.2, E500 reverberation chamber can be considered a suitable environment to measure throughput, at least, for the frequencies used by LoRa, in other words, 868 MHz.

Annex C: How to calculate obtained throughput

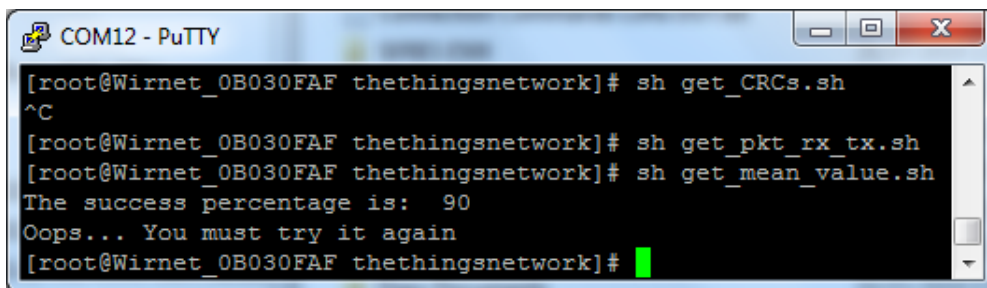
In this annex, how to analyse all the generated information in order to work out the obtained throughput is going to be explained. Before anything else, all the codes that were required to do this were made using *bash*.

First, it is necessary to save the generated information when executing *packet forwarder* in a new file, as after we will need it. Hence, first of all, we must execute packet forwarder. Once that this program is running, we have to realize what the most important information is, that is, among all generate lines there are some lines which are the really important lines.

As we already know this, we can “filter” and choose the really important information, that is to say, the value percentage of correctly sent packets for every retransmission, just as the number of forwarded packets. Therefore, the important information is saved in other file. Thus, this desired information will be easily reachable.

Next step is to calculate the required values to find out the total number of sent packets that were correctly transmitted, and so we need to calculate the addition of the percentages (where percentage indicates the percentage of packets that were correctly transmitted) and the total number of sent packets.

In this way, we are already able to work out the success mean percentage, by dividing the addition of the percentages by the total number of sent packets. To get this across, picture below is shown.



```
COM12 - PuTTY
[root@Wirnet_0B030FAF thethingsnetwork]# sh get_CRCs.sh
^C
[root@Wirnet_0B030FAF thethingsnetwork]# sh get_pkt_rx_tx.sh
[root@Wirnet_0B030FAF thethingsnetwork]# sh get_mean_value.sh
The success percentage is: 90
Oops... You must try it again
[root@Wirnet_0B030FAF thethingsnetwork]#
```

Figure C.1 Example of obtained throughput

The minimum percentage to pass the test was established as 95%. Due to this, we see 90% and the software says ‘You must try it again’.

Annex D: Connecting raspberry Pi with a Wireless Network

You can communicate with your Raspberry Pi from your control PC through an Ethernet connection, meaning that Raspberry is not able to have an internet connection using the same via (ethernet). Therefore, the device will reach this kind of connection by WiFi. In order to get it, some files must be modified.

One of these files is “interfaces” (/etc/network/interfaces). This file gives an static IP for an Ethernet connection, and a net mask. It also lets the Raspberry to obtain an IP by DHCP protocol for a WLAN connection. Furthermore, this files point at other file, whose name is “wpa_supplicant.conf” (/etc/wpa_supplicant/wpa_supplicant.conf), which contains the required configuration to get a wireless connection.

Both files interfaces and wpa_supplicant.conf must be modified in order for raspberry pi to be able to obtain a Wifi connection. The most important thing is to keep in mind that both parameters ssid and key of the networks have to be introduced in that file, as shown in next two lines:

```
ssid="raspi"  
psk="upct"
```

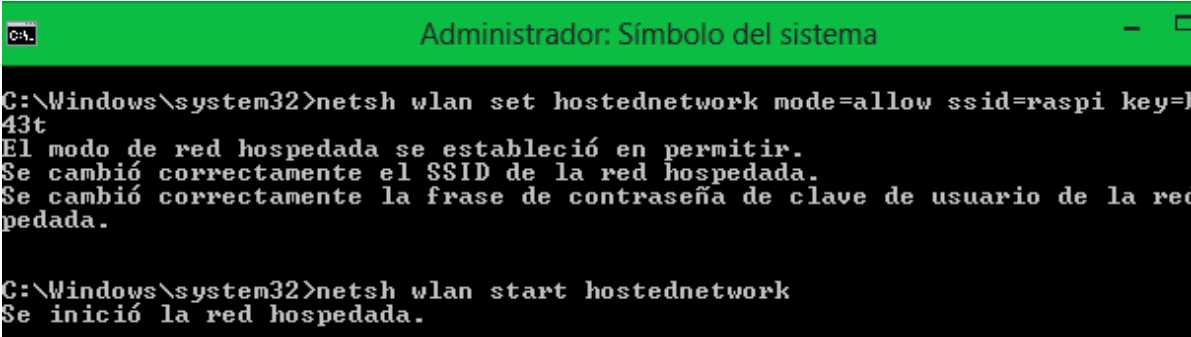
Once that these files have been already configured, next step is to create a network. Remember that this network must have “raspi” as SSID and “upct” as a key. Having said that, we must open command prompt as an administrator. Then, write this command:

```
netsh wlan set hostednetwork mode=allow ssid=raspi key=upct
```

After that, the created network must be habilitated. So, issue next command:

```
netsh wlan start hostednetwork
```

Next image shows how these steps must be done:



```
Administrator: Símbolo del sistema  
C:\Windows\system32>netsh wlan set hostednetwork mode=allow ssid=raspi key=upct  
43t  
El modo de red hospedada se estableció en permitir.  
Se cambió correctamente el SSID de la red hospedada.  
Se cambió correctamente la frase de contraseña de clave de usuario de la red  
hospedada.  
C:\Windows\system32>netsh wlan start hostednetwork  
Se inició la red hospedada.
```

Figure D.1 Creating a wireless network in windows

To sum up, the control PC can be also used as a wifi router and so our raspberry Pi will reach a internet connection. In this way, we will be able to control our raspi just as our raspi will be able to install and update as many programs as required, using only one PC.