Escuela Politécnica Superior de Linares

Master in Telecommunications Engineering

Universidad de Jaén

Escuela Politécnica Superior de Linares

MASTER THESIS

ACOUSTIC NOISE MONITORING WITH LOW POWER WIDE AREA NETWORKS

Student: Amine Alrharad

Supervisors: Prof. D. Manuel Ángel Gadeo Martos
Prof. D. Ulrich Birkel

Department: Telecommunication Engineering

September, 2022

Index

FIGURE INDEX 4
1. ABSTRACT 6
2. INTRODUCTION 7
3. OBJECTIVES 10
4. METHODOLOGY 11
4.1 Adding a security layer to the web server 11
4.1.1 Basic information about the web server 11
4.1.2 Updating the version of the TLS protocol 11
4.1.3 enabling SSL/TLS in AppServ 12
4.2 Sigfox 15
4.2.1 Architecture 15
4.2.2 Ultra Narrow Band 16
4.2.3 Random access: RFTDMA protocol 17
4.2.4 Sigfox messages 18
4.2.5 cooperative reception 18
4.2.6 Modulation 19
4.2.7 Security 19
4.2.8 Features 21
4.3 LoRa and LoRaWAN 23
4.3.1 LoRa 23
4.3.2 LoRaWAN 25
4.3.3 The Things Network 35
5. DEPLOYMENT OF THE PROPOSED NETWORK 37
5.1 Sigfox part 37
5.1.1 Used hardware 37
5.1.2 Sigfox backend 41
5.1.4 CALLBACKS 47
5.2 LoRa/LoRaWAN part 49
5.2.1 Used hardware 50
5.2.2 The Things Network TTN 58
6. Noise indicator 70
6.1 Software Implemented in the Sensor Nodes for Noise Monitoring 70
6.2 Evaluation of low frequency components. 73
6.2.1 Frequency weighting 73
6.2.2 Presence of low frequency components according to RD 1367/2007 74

6.2.3 Presence of low frequency components according to D176/2009 75
6.3 The proposed noise indicator 75
6.3 Results 76
To demonstrate the effectiveness of the noise indicator when it comes to the existence of low-frequency components, several tests have been carried out 76
6.3.1 Tones 76
6.3.2 White noise 77
6.3.3 Response to music 77
6.3.4 Responses in a street of an urban area 78
6.3.5 form of representation on the server 79
7. CONCLUSIONS 80
8. FUTURE LINES 81
9. REFERENCES 82

FIGURE INDEX

Figure 1. General scheme of the wireless sensor network 8
Figure 2. High-level architecture of the Sigfox network [10] 15
Figure 3. comparison of wireless technologies [10] 16
Figure 4. Resilience to interference provided by UNB [8] 17
Figure 5. Ultra Narrow Band Signal [9] 17
Figure 6. Sigfox message [11] 18
Figure 7. Sigfox cooperative reception [10] 18
Figure 8. BPSK modulation [12] 19
Figure 9. MAC verification to check the message integrity and device authentication [10] 20
Figure 10. The different checks executed along the uplink message treatment [10] 21
Figure 11. Combination of Sigfox specificities [10] 22
Figure 12. Low idle consumption increasing the battery life [10] 22
Figure 13. Sigfox coverage [15] 23
Figure 14. p2p mode [18] 24
Figure 15. European frequency band used for LoRa [19] 24
Figure 16. Hybrid mode scheme [18] 26
Figure 17. ABP [22] 26
Figure 18. OTAA [22] 27
Figure 19. LoRaWAN device classes [18] 29
Figure 20. security [23] 31
Figure 21. LoRaWAN message 31
Figure 22. The physical layer Payload 32
Figure 23. MAC header 32
Figure 24. MACPayload 32
Figure 25. The frame header 33
Figure 26. Maximum frame size in the 868 MHz band 33
Figure 27. Upstream communication Node/Gateway/Server with confirmation 34
Figure 28. Downstream Communication Server/Gateway 35
Figure 29. Evolution of LoRaWAN stack versions 36
Figure 30. Arduino DUE 38
Figure 31. Sigfox module and antenna 39
Figure 32. Communication shield 40
Figure 33. MICROPHONE MODULE WITH AMPLIFIER MAX4466 41
Figure 34. Login form 42
Figure 35. Sigfox backend main page 42
Figure 36. Sigfox backend - Device 43
Figure 37. Sigfox backend - Device information 44
Figure 38. Sigfox backend - Device messages 44
Figure 39. Sigfox backend - Device type list 45
Figure 40. Sigfox backend - Device type edition 46
Figure 41. Sigfox backend - Device type info 46
Figure 42. Sigfox backend - Device type asociated devices 47
Figure 43. Sigfox backend - callbacks 47
Figure 44. Email from Sigfox 48
Figure 45. Ngrok simplified architecture 49

Figure 46. LoRa Gateway 50
Figure 47. Connect via Wi-Fi 51
Figure 48. Login form Dragino 52
Figure 49. Wireless Overview - Dragino 52
Figure 50. Wireless Scan - Dragino 52
Figure 51. Joining Network - Dragino 53
Figure 52. Interfaces - Dragino 53
Figure 53. LoRa configuration - Dragino 54
Figure 54. LoRaWAN configuration - Dragino 55
Figure 55. LoRa shield SX1276 [30] 56
Figure 56. Frequency band – LoRa end node 56
Figure 57. pin mapping LoRa end node 57
Figure 58. ABP mode configuration end node 57
Figure 59. Live data example 57
Figure 60. TTN - sign up 58
Figure 61. TTN - plans 58
Figure 62. TTN cluster 59
Figure 63. TTN create an account 59
Figure 64. TTN activation 60
Figure 65. TTN console 60
Figure 66. TTN - register a gateway 61
Figure 67. TTN - gateway info 62
Figure 68. TTN - gateway status 62
Figure 69. TTN - gateway overview 63
Figure 70. TTN - add application 63
Figure 71. TTN - application list 64
Figure 72. TTN - add end device 64
Figure 73. TTN - end device registration 65
Figure 74. TTN - end device overview 65
Figure 75. TTN - payload formatters 66
Figure 76. TTN webhook 67
Figure 77. TTN - webhook templates 68
Figure 78. TTN - webhook info 68
Figure 79. The algorithm’s functional blocks. SPL—sound pressure level [32] 70
Figure 80. Frequency weighting 73
Figure 81. Results obtained playing tuba instrument 79

1. ABSTRACT

This master thesis aims to design and deploy a system to monitor the noise pollution using Low Power Wide Area Networks (LPWANs). Likewise, we are going to design, implement and deploy a more accurate noise indicator, adapted to resource-constrained devices, such as Arduino, in order to obtain a more precise real-time information about acoustic noise, which will help us to evaluate the presence of low frequency components that allows a detection more adjusted to reality of the degree of annoyance caused by the noise perceived by people.

For this purpose, a network topology will be designed and deployed aiming to monitor the noise pollution. In this master thesis, we are going to use two technologies: SIGFOX and LoRaWAN networks, with the goal of sending the noise indicators to our proprietary web server.

Finally, we are going to adapt the web server implemented by the Telematics Systems Engineering research group (TIC-220) to our requirements in order to achieve storing the noise indicators sent by the end nodes (SIGFOX and LoRaWAN) in a database, processing and visualizing them to the end user. As well, we are going to configure the TIC-220's web server so that all the communication among SIGFOX backend, LoRaWAN server (TTN) and the web server will be done through https protocol instead of http.

2. INTRODUCTION

Nowadays, population growth and changes in society's habits have given rise to major changes in urban environments, both visual and acoustics, creating increasingly less sustainable environments and a loss of quality of life in them. Noise pollution is inherent in these changes. Therefore, it is necessary to implement new tools to manage the environmental noise that allow problems to be identified quickly for their control.

Noise, as a harmful factor, is one of the most frequent complaints of the population living in large cities [1], which has caused that the authorities begin to consider noise as a threat that it is necessary to evaluate and propose new solutions to handle it.

Traditionally, noise monitoring, its control and evaluation have been carried out in an urban environment through Sound Level Meters (SLMs) and/or theoretical-practical models with the aim of creating noise maps in certain areas of cities [2]. Measurements with sound level meter have many drawbacks when implementing measurements in large period of time or with many repetitions, especially in outdoor environments [3]. These measurements are manual, where training and experience are required, and therefore a high cost of personnel [4].

Additionally, sound level meters are expensive equipment and whose values captured and subsequently analyzed are for a determined time interval and space. In addition, the results obtained are only for a unique target parameter.

However, these tools and results are limited in scope by not evaluating aspects as important as psychoacoustics, which take into account the subjective sensation that the noise generated produces upon people [5].

The current trend in the implementation of Technologies of Information and Communications (TIC), and specifically the Internet of Things (IoT), in all daily areas and not only in industry, have given rise to great advances that have made possible the improvement of electronic devices, both in features, such as size, and cost, for the monitoring of environmental data. As well, the possibility of transmitting the collected information in a simple way is added for later storage and analysis.

One of the most interesting fields of the internet of things is the Wireless Sensor Networks (WSNs). WSNs are formed of autonomous devices, distributed throughout an area of interest and whose objective is to monitor physical or environmental parameters such as acoustic noise, temperature, humidity, pressure, or movement. It is considered one of the key technologies nowadays to improve many industries.

Possible applications of Wireless Sensor Networks could be:

- Acoustic noise monitoring
- Industrial automation
- Smart and automated homes
- Video surveillance
- Traffic monitoring
- Medical device monitoring
- Monitoring of weather conditions
- Robot control

In the recent years, the creation of smart cities is growing, making use of the Internet of Things [6]. This technology is providing new tools for monitoring, management and evaluation of environmental parameters, including the acoustic noise, while reducing the costs of traditional techniques, such as SLMs, and providing real-time data that allow taking better decisions to mitigate possible adverse effects on different areas of the city.

Recently, the application of IoT technologies for the field of acoustics, and especially for acoustic noise monitoring [25], has been of great interest. Using these IoT technologies, it has been developed small and autonomous devices for carrying out measurement with good quality and low cost, allowing through a wireless communication with servers in the cloud, sharing information with end users.

In this master thesis, we are going to propose a solution for monitoring the noise pollution, in order to calculate a noise indicator and draw noise maps. By using Low Power Wide Area Networks, such as SIGFOX and LoRaWAN, we will transmit all the required data to the web server. For this purpose, we have proposed the scheme shown in the following figure:

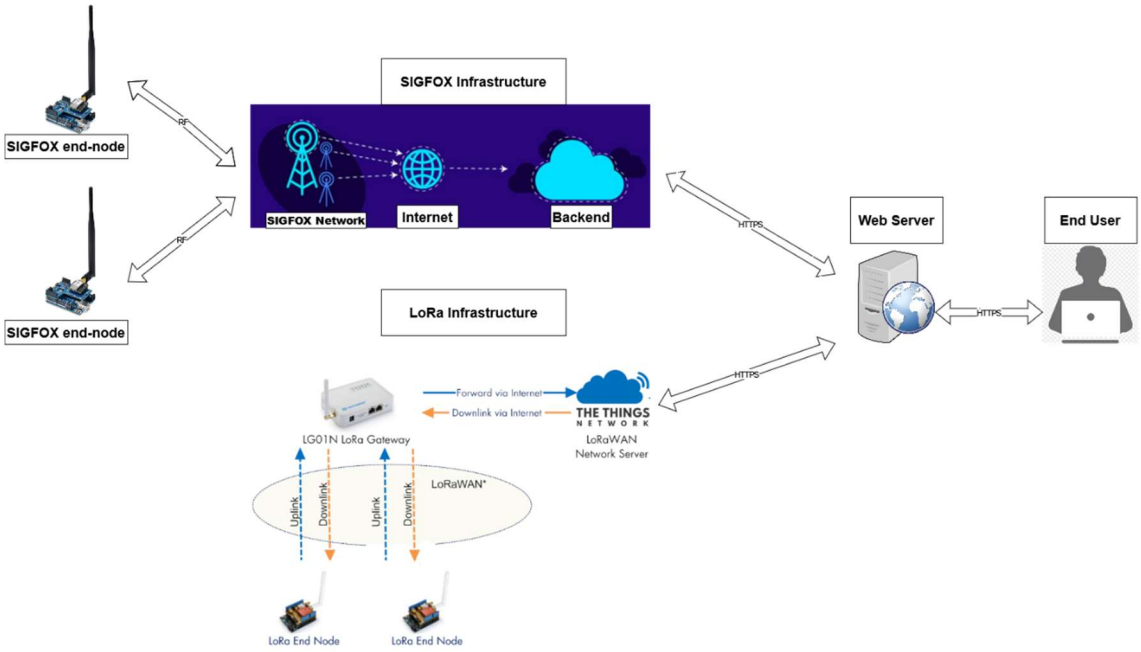


Figure 1. General scheme of the wireless sensor network

As it might be seen in figure 1, the proposed network is divided into two blocks. The first block will be operating using SIGFOX infrastructure. SIGFOX end-nodes, after reading the real-time urban acoustic noise, will calculate a noise indicator with the help of a frequency domain-based algorithm and will send it to the backend using the SIGFOX deployed infrastructure. Afterwards, a callback will be triggered and will send the noise indicator to the web server in a POST message using the HTTPS protocol.

The second block will be deployed using the LoRa/LoRaWAN infrastructure. In this master thesis, we have used two models of Lora shield. The first model is provided by **Dragino** and the second model is made by **Libelium**. Both models will play the same role: as mentioned previously, after

reading the real time urban acoustic noise, a noise indicator will be calculated with the help of a frequency domain-based algorithm and sent to the Lora Gateway using LoRaWAN protocol. Subsequently, the Gateway will forward the IP packets via internet to the LoRaWAN network server (in this master thesis we have used the things networks, TTN). Finally, a webhook, previously created in the things network, will send a json message to web server whenever an uplink message is received, using the HTTPS protocol.

Finally, we have configured the web server in order that all the communication, with the end users and internet, will be done by adding a security layer.

3. OBJECTIVES

The aim of this master thesis is to design and implement a system to monitor urban acoustic noise using Low Power Wide Area Networks, as SIGFOX and LoRaWAN networks.

To achieve the main objective a series of more specific objectives have been defined which, in combination will allow the final goal to be reached.

The specific objectives are listed below:

- Become familiar with LoRa, LoRaWAN, Sigfox and Arduino, Waspmote and/or Raspberry devices.
- Modify the web server in order to add a security layer when communicating with internet, as well as when interacting with end users.
- Choose the sensor node technology to be used, and then program the nodes to capture and process the acoustic noise in order to obtain a noise indicator.
- Design and implement a more accurate noise indicator that gives us an objective parameter to evaluate the acoustic noise.
- Send the noise indicator to the backend platform of Sigfox and to a LoRaWAN server (for example TTN).
- Do the integration to a private cloud platform for the real-time visualization of the noise indicator.
- Finally, we have to make different tests in order to validate the correct operation of the noise monitoring system, LPWAN networks, and the cloud platform.

4. METHODOLOGY

4.1 Adding a security layer to the web server

In this section, we are going to configure the web server used in this master thesis, in order to establish an encrypted connection between the web server and internet.

Note: We are going to use SSL certificates issued by the certifying entity GEANT OV RSA CA 4.

4.1.1 Basic information about the web server

As we have mentioned before, we are going to use the web server developed by the Telematics Systems Engineering research group which name is **maparuido.**

Maparuido is a web server implemented using the useful tool **AppServ** which is a full-featured of Apache, MySQL, PHP, phpMyAdmin.

The AppServ version used in order to implement the web server maparuio is the v.2.5.10. This version has many limitations, such as we cannot enable the SSL layer and make https requests using new versions of web browsers, also it gave us errors when trying to integrate it with others platforms.

These limitations are due to that the version 2.5.10 of AppServ is using an old version of the TLS protocol, 1.0 in this case. There are four versions of TLS protocol: 1.0, 1.1, 1.2 and 1.3. The versions 1.0 and 1.1 are obsolete and there is no support for them.

In order to solve this problem, we have two options:

- **Option 1**: Installing a new version of AppServ which is compatible with the versions 1.2 and/or 1.3 of the TLS protocol.
- **Option 2**: Updating the version of the TLS protocol in the current version of AppServ installed.

The first option is discard due to that some JavaScript functions are deprecated when using new versions of AppServ. Due to the time constraint, we are going to opt for the second option.

4.1.2 Updating the version of the TLS protocol

In order to update the version of the TLS protocol, we are going to make the following changes in the web server.

- Updating the libraries **libeay32.dll** and **ssleay32.dll** located inside the “bin” folder of Apache.
- Updating the executable file **openssl.exe** located inside the “bin” folder of Apache.
- Updating the configuration file **openssl.conf** located inside the “conf” folder of Apache.

4.1.3 enabling SSL/TLS in AppServ

In order to configure AppServ to use the https protocol instead of the default http protocol used, we have to implement the following configuration.

4.1.3.1 Adding a new domain name in localhost

The SSL certificates, given by the University of Jaen, are issued to secure a specific domain name, in this case **maparuido.ujaen.es**. So that, we have to add this domain name in the Windows server.

- Step 1:

Firstly, we have to open the "**hosts"** file (path: C:\windows\system32\drivers\etc\hosts) using the **administration mode** and we have to add the following code line in the bottom part of the file:

#127.0.0.1 maparuido.ujaen.es

Note: Some virus checkers block access to the **hosts** file, so we may need to disable our virus checker, or configure it not to block the hosts file temporarily.

- Step 2:

Afterwards, we have to restart the dnscache as follows, by opening a Windows's administrative command prompt as an administrator and typing the following command:

#ipconfig /flushdns

- Step 3

Now, we have to configure Apache to enable SSL/TLS protocol. To do so, we have to edit the "**httpd.conf**" file, located in "C:\AppServ\Apache2.2\conf", by uncommenting (remove the comment '#') the following line:

#LoadModule ssl_module modules/mod_ssl.so

We have to uncomment another line code and then move it after the SSL_module, as follow:

<IfModule ssl_module>

SSLRandomSeed startup builtin

SSLRandomSeed connect builtin

</IfModule>

Secure (SSL/TLS) connections

Include conf/extra/httpd-ssl.conf

- Step 4

Next, we have to configure PHP to activate SSL. In order to do so, we have to edit the file "**php.ini-recommended**", located in the following path: "C:\AppServ\php5", by uncommenting (remove the comment ;) the following line:

#extension=php_openssl.dll

- Step 5

Following, we have to configure the "**httpd-ssl.conf**" file, located in the following path: "C:\AppServ\Apache2.2\conf\extra". This file is released by Apache and contains some default file location. We can leave most of this file as it is, but we need to configure the virtual host in here to match our actual sites location and a few other things so, we have to find the following lines:

#DocumentRoot "C:/Apache2.2/htdocs"

#ServerName x:443

#ServerAdmin x@x.com

#ErrorLog "C:/Apache2.2/logs/error.log"

#TransferLog "C:/Apache2.2/logs/access.log"

And change them to:

#DocumentRoot " C:/AppServ/www"

#ServerName maparuido.es:443

#ServerAdmin jan@ujaen.es

#ErrorLog " C:/AppServ/Apache2.2/logs/error.log"

#TransferLog " C:/AppServ/Apache2.2/logs/access.log"

- Step 6:

Later, we have to change the paths of the SSL certificates to the correct ones. Before that, we have to create a new folder called "**certificados**" in the following path: "C:\AppServ\Apache2.2\conf", then, we have to find these lines:

#SSLCertificateFile "C:/Apache2.2/conf/server.crt"

#SSLCertificateKeyFile "C:/Apache2.2/conf/server.key"

And changing them by these lines:

#SSLCertificateFile " C:/AppServ/Apache2.2/conf/certificados/ maparuido_ujaen_es_cert.cer"

#SSLCertificateKeyFile " C:/AppServ/Apache2.2/conf/certificados/ maparuido.ujaen.es_privatekey.pem"

- Step 7:

Afterwards, we have to find these lines:

<Directory "C:/Apache2.2/cgi-bin">

SSLOptions +StdEnvVars

</Directory>

And depending on which version of Apache (2.2 in our case) we are using, we have to make the following changes:

- **Apache 2.2 Syntax**

```
<Directory " C:/AppServ/www ">
  SSLOptions +StdEnvVars
  Options Indexes FollowSymLinks MultiViews
  AllowOverride All
  Order Deny,Allow
  Deny from all
  Allow from 127.0.0.1 localhost ::1
</Directory>
```

- **Apache 2.4 Syntax**

```
<Directory "c:/wamp/www/wamphelpers">
  SSLOptions +StdEnvVars
  Options Indexes FollowSymLinks MultiViews
  AllowOverride All
  Require local
</Directory>
```

- Step 8:

Finally, we have to find the following lines:

```
SSLSessionCache        "shmcb:c:/Apache2.2/logs/ssl_scache(512000)"
SSLSessionCacheTimeout  300
CustomLog "c:/Apache2.2/logs/ssl_request.log" \
"%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

And change them by:

```
SSLSessionCache        "shmcb:c:/AppServ/Apache2.2/logs/ssl_scache(512000)"
SSLSessionCacheTimeout  300
CustomLog "c:/AppServ/Apache2.2/logs/ssl_request.log" \
"%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

- Step 9:

After making all the previous changes, we have to restart the Apache Server. If the Apache server is unable to start, it means that some of the changes did not set up correctly. We can check the possible errors by using this command in a Windows command prompt:

cd C:\AppServ\Apache2.2\bin

httpd –t

4.2 Sigfox

In this section, the Sigfox technology and its characteristics will be explained in details. Sigfox is a French global network operator founded in 2010 that builds wireless networks to connect low-power objects [7]. It is a technology designed for communications of devices that require a long range (up to 50 kilometers in the most favorable cases), at the same time that they are not connected to the electrical network and, therefore, require a very high autonomy time. in exchange for requiring little information to be sent and at wide intervals. Sigfox provides an ideal network for these devices, to which mobile phone technologies are not adapted and are not feasible.

4.2.1 Architecture

The Sigfox network is generally divided into two layers [8]:

- **Network equipments**: in this layer, there are all the devices in charge of transmitting the messages, along with the base stations that receive them.
- **Sigfox support systems**: it is in this layer where the messages are processed and the alerts to the clients are generated. It also offers functionalities for users (APIs) that allow them to generate response actions against the arrival of messages, control and monitor their devices, analysis tools, etc. It is the least visible layer but the one that encompasses the most complexity.

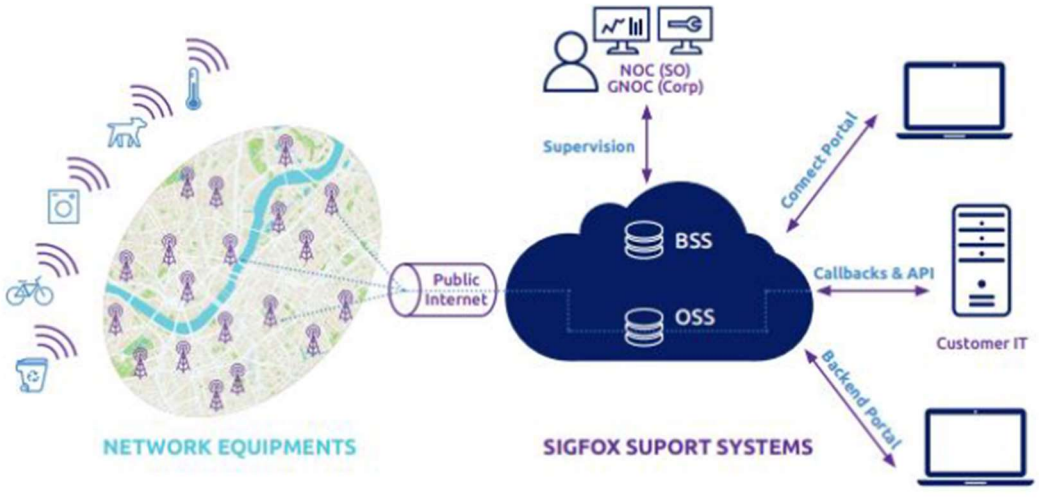


Figure 2. High-level architecture of the Sigfox network [10]

The information is sent from the devices to the base stations wirelessly. Once the data is received, the base stations are in charge of communicating with the Sigfox Cloud.

The Sigfox backend is responsible for handling the received messages. It is very likely that the same message will be received multiple times due to the **cooperative reception** which will be explained later, so the backend must ensure that it only stores the received messages once. In addition to the message itself, additional data, or metadata, associated with it is also stored so that users can retrieve it. This includes attributes such as received signal strength, noise, etc.

Finally, users can access the messages through the web interface and the API, and even define HTTP callbacks in response to different events, such as the arrival of a new message. This is useful for using it together with our own database or an additional cloud service.

4.2.2 Ultra Narrow Band

The name itself of this technology is very descriptive. It is based on using very narrow channels of the spectrum, of the order of less than 1 KHz. Transmitting data on these channels has the advantage of that minimum power is needed and the signals reach long distances in the air [10]. Figure 3 shows a power vs bandwidth comparison of different technologies:

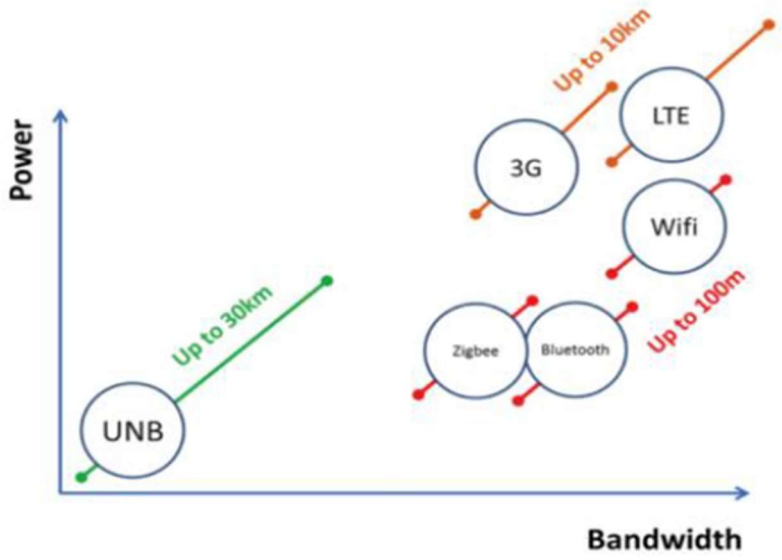


Figure 3. comparison of wireless technologies [10]

This technology (UNB) is usually used in unidirectional communications, in which sensors provide certain information to receivers, commonly called base stations, and do not wait for any response. However, it is also possible to implement a bidirectional communication. Since it is required that the energy consumption falls as minimum as possible in the end devices, the communication is asymmetric, so the data is not transmitted and received at the same speed.

In addition, another advantage that Ultra Narrow Band provides is that it does not interfere with broadband communications (which use wider channels), and it is very resistant to noise, making it perfect for low communication error rates and packet loss [8]. This can be seen in the following figures 4.

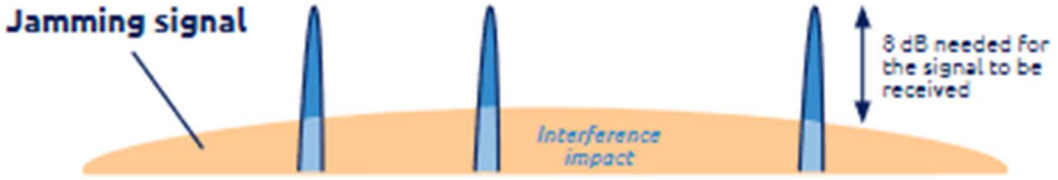


Figure 4. Resilience to interference provided by UNB [8]

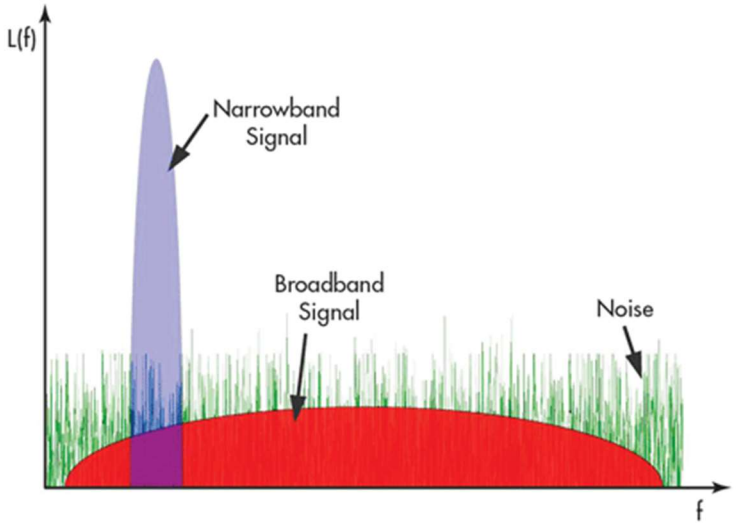


Figure 5. Ultra Narrow Band Signal [9]

Sigfox works in the frequency range of 868 - 868.6 MHz for Europe and uses channels with a width of 100 Hz.

4.2.3 Random access: RFTDMA protocol

Random access [10] is one of the keys that makes Sigfox have such a low collision rate in communications. When sending information, each device randomly chooses one of the available channels, and the data is sent without any type of synchronization between the sender and the receiver. In addition, it is also capable of sending up to two copies of the same message on different channels, for greater robustness. This is known as the RFTDMA (Random Frequency and Time Division Multiple Access) protocol.

The base stations are responsible for sampling the entire range of the spectrum (868-868.6 MHz) in search of UNB signals. Given that, there are around 60,000 channels available and each one of 100 Hz, the possibility that two nearby devices choose the same channel at the same instant of time is very small, so collisions hardly occur.

4.2.4 Sigfox messages

The Sigfox protocol uses messages of reduced size, whose data part ranges from 1 to 12 bytes. This space is enough to store any type of value obtained from a sensor, so larger messages are not required.

Figure 6. Sigfox message [11]

Figure 6 shows a graphical representation of the messages in Sigfox. European regulation indicates that this public frequency band (868 MHz) can only be used 1% of the total time. This means that a maximum of 140 messages of 12-byte can be sent per day, and although the legislation is not identical in this regard in all places, Sigfox offers a maximum of this number of messages at the moment in all countries equally [10].

4.2.5 cooperative reception

The devices are not associated with a specific base station, like in mobile network protocols, but when transmitting, all the base stations within the range receive the message. This provides a lot of robustness to the system, since, even if a base station fails for any reason, the rest will still be able to pick up the message and offer it to the backend [10].

The term backend refers to the infrastructure behind the system, which includes the logic of treatment, storage and processing of information. In the case of Sigfox, it offers through a Web portal, after proper authentication, access to the information held by its servers, where it shows users relevant information associated with their devices. Figure 7, represents graphically the concept of cooperative reception.

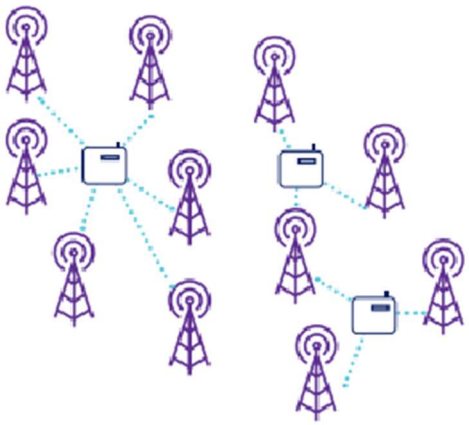


Figure 7. Sigfox cooperative reception [10]

4.2.6 Modulation

Sigfox uses phase shift keying modulation (PSK) [12]. This modulation is characterized in that the phase of the signal represents each digit of information.

Specifically, BPSK modulation is used, which is optimal for low-cost transmitters that do not require high speeds. It has two output phases for a single frequency, each 180 degrees:

- Phase 1: Logical "1"
- Phase 2: Logical "0"

An example of modulation for a digital input would be the one in figure 8.

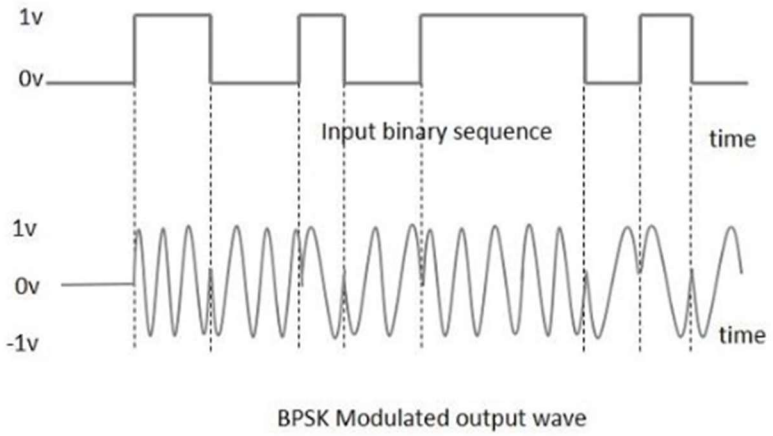


Figure 8. BPSK modulation [12]

4.2.7 Security

Sigfox is a secure technology, which employs security mechanisms at various levels, and which together make the Sigfox network a reliable option for the users.

4.2.7.1 Security in message processing

When processing the messages, Sigfox performs a series of preliminary checks [10]:

- **Sequence number**: This number is a mechanism to avoid duplication of messages. It is simply a number from 0 to 255 (an unsigned byte) that the Sigfox backend checks when a message is validated. In this way, if a message is captured and duplicated maliciously, it will be automatically discarded, since the sequence number will not have increased and it will become an invalid message.
- **Verification of MAC** (**Message Authentication Code**): each device has a symmetric authentication key that is given to it when the device is manufactured, and each message that is sent contains an encrypted part based on that key. In this way, knowing the device identifier, the backend is able to decrypt that part using the symmetric key, and thus verify both the authenticity and the integrity of the device that has issued the message. The process is showed in the next Figure.

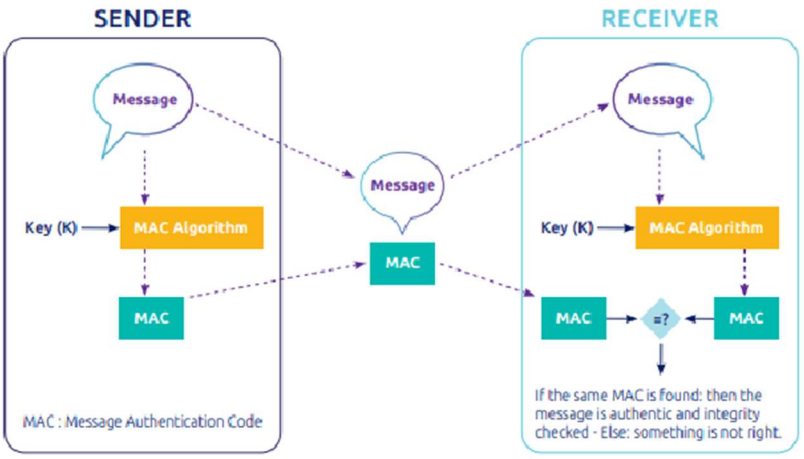


Figure 9. MAC verification to check the message integrity and device authentication [10]

- **Message encryption**: by default, the information is sent unencrypted. However, in some cases this information needs to be totally inaccessible and we cannot rely only on network security. Therefore, Sigfox offers the possibility to use its own encryption function, created in collaboration with CEA-LETI [13], a French research institute, which is intended for especially small messages, and is derived from the device's symmetric key.

4.2.7.2 Base station security

For security, several mechanisms have been implemented in the base stations in order to protect the integrity of the system [10]:

- Nobody can steal the software from the antennas.
- Nobody can alter the operating system of the antennas, since its integrity is checked in the boot sequence, and a periodic check is also carried out at run time.
- There is a relationship between the operating system and the hardware. The system can only boot on a base station built by Sigfox.

Additionally, communications between the antennas and the backend are carried out through a virtual private network (VPN) that prevents access by intruders.

4.2.7.3 Security in the creation and distribution of keys

The key creation process can only be carried out if the manufacturer has the certificates required by Sigfox to communicate with the network, and there are different levels of certifications. The process is the following [10]:

1. The manufacturer sends an email requesting a number of device identifiers to Sigfox, along with the certificates.
2. Once the request has been validated, Sigfox offers us a range of identifiers that we can use. The Central Registration Authority (CRA) creates a Network Authentication Key (NAK) and a Porting Authorization Code (PAC). The PAC will be used later to activate the device.

3. The manufacturer obtains the necessary files from the CRA: a binary file containing pairs (device identifier, key) encrypted with AES-ECB [14], and a text file containing pairs (device identifier, PAC).
4. The manufacturer uses the applications offered by Sigfox to decrypt and upload the keys to the devices.
5. The end user receives the device along with the device identifier and the PAC.
6. The end user accesses the Sigfox web portal and registers the device with this data.
7. The backend checks this data and registers the device, along with some additional internal operations.

4.2.7.4 Data center security

Sigfox servers are physically protected with biometric protection for physical access. In addition, each center is associated with different internet providers. Sigfox has a load balanced architecture, and all its layers are fully monitored.

Additionally, Sigfox uses its own solution to defend against DDoS (Distributed Denial of Service) attacks, which detects these types of attacks and mitigates them. An overview of Sigfox security is shown in Figure 10.

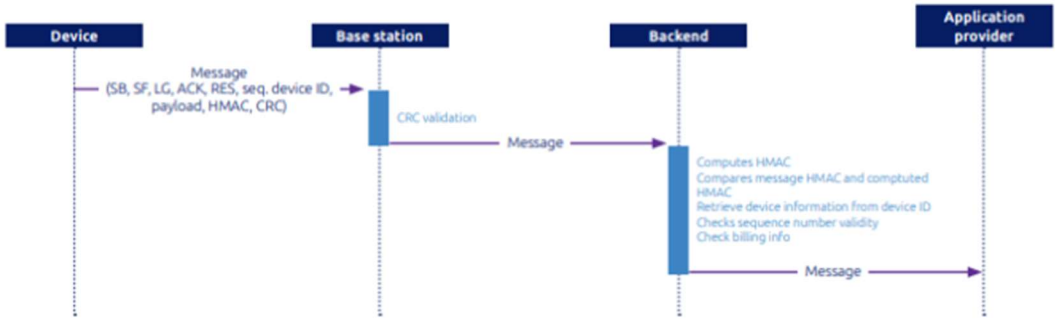


Figure 10. The different checks executed along the uplink message treatment [10]

4.2.8 Features

4.2.8.1 scalability

Sigfox is a highly scalable network, to which millions of devices can be connected. This is due to the sum of three factors that we have explained in the previous sections, also represented in figure 11:

- **Ultra Narrow Band**: allows great resistance against interference in addition to high scope.
- **Random Access**: The fact that the channels are chosen randomly helps a lot in avoiding collisions.
- **Spatial distribution**: the devices are placed in sites separated from each other, where the antennas that can receive the messages are highly different. This adds a lot of robustness to the system.

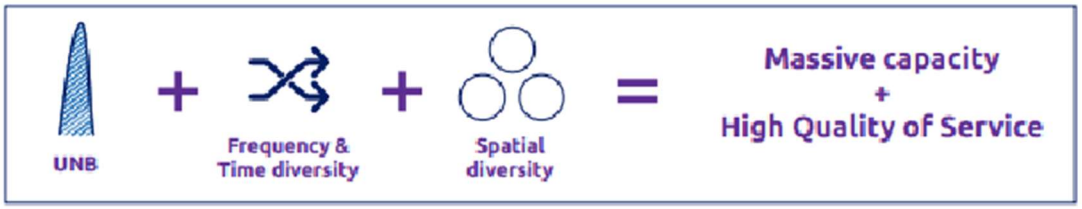


Figure 11. Combination of Sigfox specificities [10]

4.2.8.2 High energy efficiency

The energy cost, in addition to being low thanks to the UNB, also depends on the chip used. This cost is usually between 10 and 50 mA in transmission, and between 5 and 20 mA in reception, taking into account that in Europe a maximum transmission power of 14 dB is allowed, while in the United States it is 22 dB. The average airtime of a message is 6 seconds. [10]

Most of the time, the devices remain in an idle or rest state, in which they hardly consume any energy, and only wake up to transmit or to capture information from their sensors. The typical energy cost is represented in figure 12.

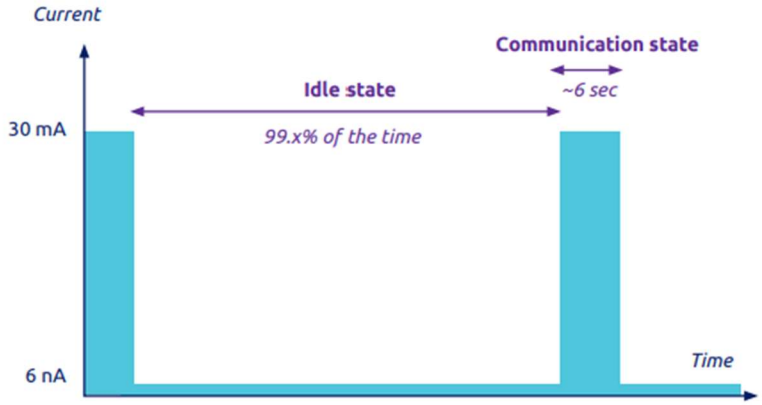


Figure 12. Low idle consumption increasing the battery life [10]

4.2.8.3 Network range

This is one of the best advantages that Sigfox offers. The maximum distance using this technology is defined by several parameters:

- The sending power (maximum 14 dB in Europe)
- The reception sensitivity of the antennas
- The topography of the place
- Whether the devices are indoors or outdoors. This point is the least affected because it uses frequencies below the GHz, and these are less affected for indoor cases.

By itself, Sigfox has a good scope thanks to UNB and the low transfer rate, which is generally around 100 bits per second [10]. Subsequently, the scope studies carried out and their comparison with the theoretical values are presented in the next section.

4.2.8.4 Coverage

Currently Sigfox is the technology oriented towards the internet of things with the infrastructure that covers the most territory in the world. It is possible to consult its coverage at any time through its website [15].

The coverage map, as of August 2022, is shown in figure 13, where the light blue color represents a place where the Sigfox antennas have already been fully deployed and the purple color represents the place that is still being covered, but it is already functional.

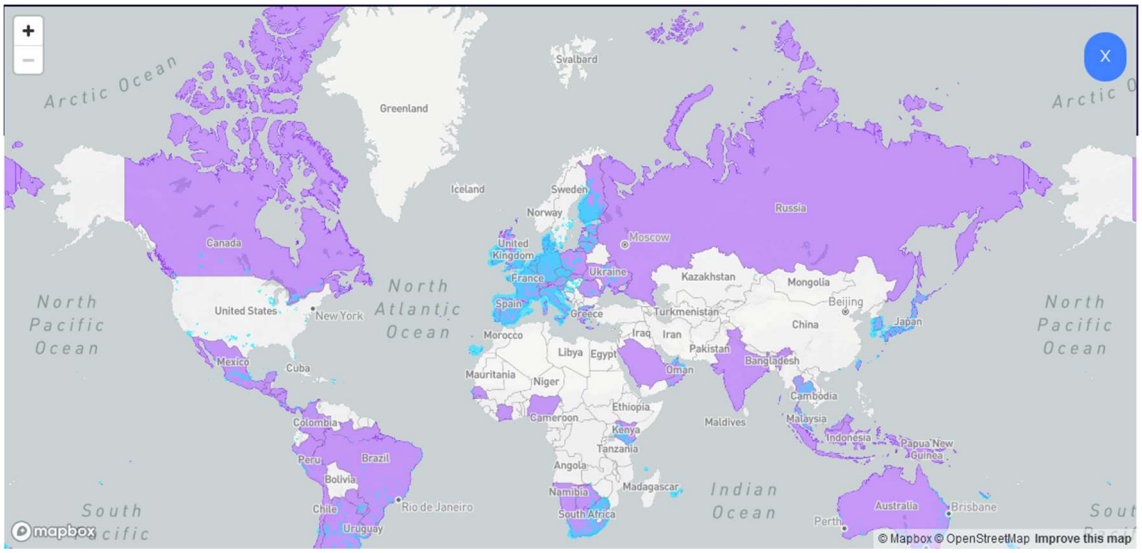


Figure 13. Sigfox coverage [15]

4.3 LoRa and LoRaWAN

In this section, a theoretical exposition of LoRa and LoRaWAN will be made, one of the technologies that are fundamental to understand this master's thesis.

4.3.1 LoRa

LoRa, stands for Long Range, is a modulation technique based on spread spectrum techniques and a variation of Chirp Spread Spectrum (CSS), which modulates data over different channels and speeds, with integrated Forward Error Correction (FEC). LoRa significantly improves receiver sensitivity and, as with other techniques spread spectrum modulation uses the entire bandwidth of the channel to transmit a signal, making it robust to channel noise and insensitive to frequency offsets caused by the use of low-cost crystals [16]. LoRa can demodulate signals 19.5 dB below noise floor, while most frequency shifting (FSK) systems need a signal power of 8-10 dB above noise floor to demodulate properly [17].

LoRa is described as a frequency modulated (FM) chirp, based on spread spectrum modulation, which maintains the same low power characteristics as FSK modulation but significantly increases the communication range. Use coding gain to increase receiver sensitivity and is capable of transmission speeds ranging from 0.3 kbps to 38.4 kbps. Therefore, it is suitable for P2P communications between nodes, i.e. communications where nodes can connect directly to each other.

4.3.1.1 LoRa network

In P2P mode (figure 14), the nodes can connect directly to each other without costs, since they do not use network infrastructure. This mode works without the need for a base station or a Cloud account, as in Sigfox, and therefore does not require the purchase of any license. This mode is the only one in which LoRa nodes can work.

Figure 14. p2p mode [18]

4.3.1.2 Channels and frequency ranges

LoRa can work in various frequency ranges in different regions of the world. The frequency band used in Europe is the 863-870 MHz ISM band regulated by ETSI (European Telecommunications Standards Institute). It uses 8 arbitrarily chosen channels, according to the UN-111 in force in the Spanish BOE A-2013-4845, with a bandwidth of 0.3 MHz per channel (figure 15). The frequency band used in the USA, Canada, Australia, Singapore or Israel is the 902-928 MHz ISM band, using 13 channels with a bandwidth of 2.16 MHz per channel. These channels were chosen arbitrarily adjusting to the channels used for XBee of 900 MHz [19].

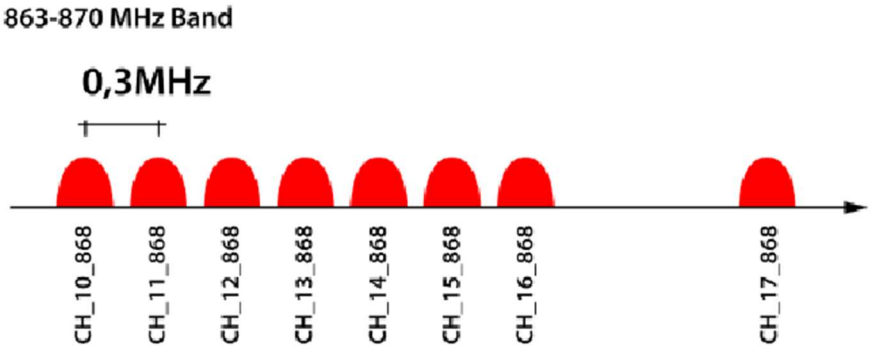


Figure 15. European frequency band used for LoRa [19]

The greatest advantage of LoRa is in the range of this technology and its scalability. A gateway or base station can cover entire cities or areas of hundreds of square kilometers.

Semtech is the manufacturer of the LoRa modules and they offer to the user a programmed library that allows communication between LoRa nodes through a simple link protocol. This library has been created by Libelium [20], a Zaragoza-based company that provides the necessary tools and libraries to operate with LoRa or even to integrate network security.

4.3.2 LoRaWAN

LoRa modulation offers physical layer (PHY) functionality, while LoRaWAN is a MAC protocol for a long-range, high-capacity star network that the LoRa Alliance has standardized for LPWANs [21]. LoRaWAN defines the MAC protocol itself and the architecture of the network system, while LoRa enables the long-range communication link at the physical level. The protocol and architecture of the LoRaWAN network decisively determine a node's battery life, network capacity, quality of service, security, and the variety of network applications. Unlike LoRaWAN, LoRa does not have rules or regulations beyond the physical layer limitations offered by the devices or those imposed by the library used. LoRaWAN can employ LoRa or FSK modulation at the physical level [21].

The LoRaWAN protocol is optimized for low-cost, battery-operated sensors and includes different classes of nodes to optimize the trade-off between network latency and battery life. It is fully bidirectional and was designed to ensure reliability and safety. The LoRaWAN architecture was also designed to easily locate and track moving objects. LoRaWAN is being deployed for national networks by major telecom operators, and the LoRa Alliance is standardizing LoRaWAN to ensure that different national networks are interoperable [18].

4.3.2.1 LoRaWAN network

Typically, LoRaWAN networks are spread out in a star-of-stars topology where gateways (a device that allows interconnecting networks with different protocols and architectures) relay messages between nodes (a device with the ability to process, gather information thanks to a sensor and communicate with other nodes or gateways in the network) and a network server located at the back end. The gateways connect to the server through a standard IP connection while the nodes and the gateway do so through a direct link using LoRa or FSK at the physical level. The communication can be unidirectional or bidirectional, although, in the latter case, the upstream traffic from a node to the gateway and from the latter to the server is the predominant.

LoRaWAN devices can operate in two ways: in P2P mode as in LoRa networks, and in hybrid mode. In hybrid mode (figure 16), a combination of LoRaWAN and P2P modes is used that allows messages to be sent using LoRaWAN networks. This mode requires a LoRaWAN license. In this case, a central node is used as a gateway working in hybrid mode and the rest of the nodes as P2P. The nodes can use a P2P star topology to reach the central node and make it access the LoRaWAN network to route the information. The basis of this operation is that the gateway listens for P2P packets and forwards them to the LoRaWAN infrastructure, as shown in the following figure:

Figure 16. Hybrid mode scheme [18]

4.3.2.2 Access modes to a LoRaWAN network

There are two methods for a module (device capable of collecting data from a sensor and transmitting it to the gateway) to become part of a LoRaWAN network [16].

- **Activation by Personalization** (ABP): The Network Session Key (NwkSKey) and Application Session Key (AppSKey), and the device address (DevAddr) are known to both the node and the server. In this way the packet transmission starts from the beginning.

Figure 17. ABP [22]

- **Over-the-Air Activation** (OTAA): The node and the network server negotiate the encryption keys at the time when the node joins the network. To do this, it is necessary for the node to send the device identifier (Device EUI), the application identifier (Application EUI), and the Application Key to the server. The server then sends the device address (DevAddr), the Network Session Key (NwkSKey) and Application Session Key (AppSKey). In this mode, this entire procedure is necessary before packet transmission begins.

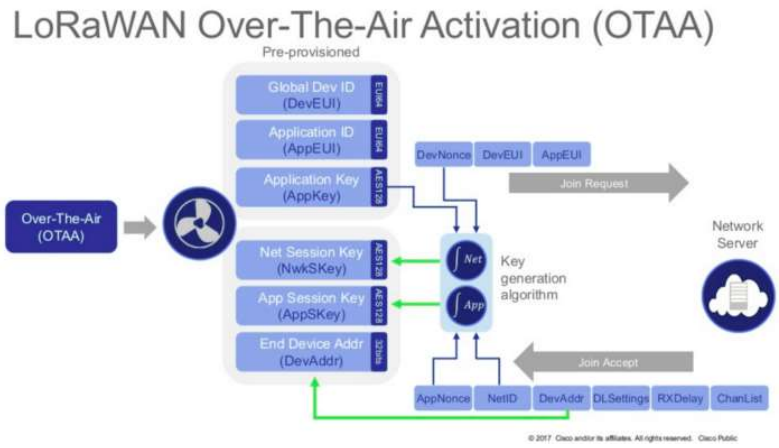


Figure 18. OTAA [22]

4.3.2.3 Transmission channels and data rates

The Communication between nodes and gateways is done using different frequency channels and data rates. Using the spread spectrum technique, communications with different data rates do not interfere with each other, creating a set of virtual channels that increase the gateway's capacity. To maximize the use of the batteries of the nodes and the capacity of the network, LoRa can manage the data rate and RF output for each node individually by means of an adaptive data rate (ADR) scheme [16].

Thus, nodes can transmit on any available channel at any time and using any available data rate as long as they comply with the following rules:

- The node switches channel pseudo-randomly for each transmission. The resulting frequency diversity makes the system more robust to interference.
- The node respects the maximum transmission duty cycle in relation to the sub-band used and local regulations.
- The node respects the maximum duration of transmission in relation to the sub-band used and local regulations.

The LoRaWAN modules have 16 channels in the 433 MHz and 868 MHz bands (Table 1), and 72 channels in the 900 MHz band. In Europe, the 433 MHz and 868 MHz bands are used, and in this master thesis we will work on the 868 MHz band, which has the following Parameters for LoRaWAN [21]:

Table 1. LoRaWAN frequency channels for the 868 MHz and 433MHz [18]. For more information about the Data Rate see Table 2

Channel	Parameters	frequency bands	
		868 MHz	433 MHz
0	Frequency	868100 kHz	433175 kHz
	Duty Cycle	0.33%	0.33%
	Data Rate	0 - 5	0 - 5
	State	On	On
1	Frequency	868300 kHz	433375 kHz
	Duty Cycle	0.33%	0.33%
	Data Rate	0 - 5	0 - 5
	State	On	On
2	Frequency	868500 kHz	433575 kHz
	Duty Cycle	0.33%	0.33%
	Data Rate	0 - 5	0 - 5
	State	On	On
3 - 15	Frequency	*	*
	Duty Cycle	**	**
	Data Rate	***	***
	State	Off****	Off****

* The first three channels have assigned frequencies. The rest of the channels can be configured within the range (868325 kHz - 869750 kHz) with a bandwidth of 125 kHz for the 868 MHz band and (433050 kHz - 434790 kHz) for the 433 MHz band (European ISM bands). LoRaWAN can use the 433 MHz frequency band as long as the EIRP of the devices is less than 10mW (10dBm) and, as in the 868 MHz band, the duty cycle is less than 1%. The 779 MHz ISM band (China) It has the same number of channels but with different center frequencies. On the other hand, the ISM band of 915 MHz (US) has another number of channels. The 915 MHz band is divided into 64 upstream channels numbered 0 to 63 using LoRa with a bandwidth of 125 kHz starting at frequency 902.3 MHz and increasing every 200 kHz up to 914.9 MHz, another 8 upstream channels numbered 64 to 71 using LoRa with a bandwidth of 500 kHz starting at frequency 902.3 MHz and increasing every 1.6 MHz up to 914.9 MHz, and 8 downstream channels numbered 0 to 7 using LoRa with a bandwidth of 500 kHz starting at frequency 923.3 MHz and increasing every 600 kHz up to 927.5 MHz.

** There are limitations that set how long the transmitter can be activated or the maximum time it can be transmitting. LoRaWAN imposes a duty cycle limitation per sub-band (Equation 1). Each time a frame is transmitted in a given sub-band, the broadcast time and on-air duration of the frame are recorded for this sub-band. The same sub-band cannot be used again during the next Toff seconds, where:

$$Toff = \frac{TimeOnAir}{DutyCycle\ (subband)} - \text{TimeOnAir}$$

Equation 1

*** The data rate accepts values from 0 to 5 for the 868 MHz band and from 0 to 7 for the 433 MHz band.

Table 2. Data Rates in LoRaWAN

Data Rate	Configuration	Bite Rate
0	LoRa: SF12 / 125 kHz	250 bps
1	LoRa: SF11 / 125 kHz	440 bps
2	LoRa: SF10 / 125 kHz	980 bps
3	LoRa: SF9 / 125 kHz	1760 bps
4	LoRa: SF8 / 125 kHz	3125 bps
5	LoRa: SF7 / 125 kHz	5470 bps
6	LoRa: SF7 / 250 kHz	11000 bps
7	FSK: 50 kbps	50000 bps

The SF corresponds to a Spreading Factor configuration, which determines the amount of redundant data that is sent in the transmission. The speed is related to the SF in such a way that if it is high, a lot of redundant data will be sent and therefore a maximum range will be achieved with low speed.

**** Channels 3 to 15 are in the Off state by default. When activating new channels, the network administrator must adapt the work cycle of the rest of the channels so that the total number of channels does not exceed the maximum 1% allowed in the frequency band. Furthermore, unlike the default channels, channels 3 to 15 can be configured with data rates 6 and 7 as long as they are compatible with the LoRaWAN module used.

4.3.2.4 Classes of LoRaWAN devices

There are three classes of LoRaWAN devices, named Class A, Class B and Class C (Figure 19). All LoRaWAN devices implement at least Class A functionality. Additionally, they may implement Class B or Class C options [21].

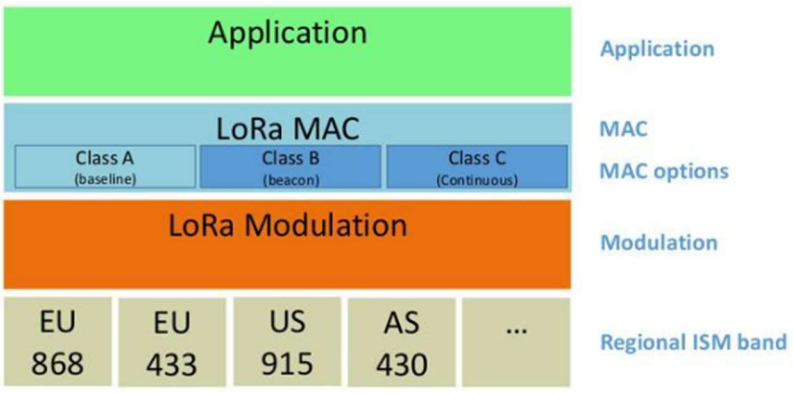


Figure 19. LoRaWAN device classes [18]

- **Class A**. Class A devices allow bidirectional communication, with the limitation that they can only receive data (downlink channel) if they have previously sent a packet (uplink channel). That's because every time the device sends a packet, two receive windows are opened, the first window one second after the transmission and the second window one second after the first window, with the opportunity to receive a packet back. This return packet contains the ACK of the sent packet as well as application data if necessary.

  These types of devices will be the ones with the lowest energy consumption of the specification, since they will be in sleep mode by default and will initiate all communication. They will serve for applications in which the devices should not receive data regularly. A LoRaWAN compliant device must always implement this basic class.

- **Class B**. Class B devices add the ability to receive data (downlink) without sending a packet (uplink), so the application can send data to the devices on a scheduled basis.

  This is achieved through the regular sending of beacons by the gateway. These beacons allow the devices to be synchronized with the gateway, and in this way they can negotiate packet reception times from the gateway to the device (downlink). Thus, the reception windows open at certain times. This class of devices will have a higher energy consumption than those of class A due to the periodic reception of the beacons from the gateway.

- **Class C**. Class C devices are permanently listening (that is, in receive mode), and therefore can receive data (downlink) at any time (except when they are sending data (uplink)). In other words, the receive windows are always kept open except when transmitting.

  This class provides the best response times and sending capacity from the server to the devices, at the cost of much higher energy consumption compared to classes A and B.

4.3.2.5 Security aspects of LoRaWAN

One of the main characteristics of this technology is the high level of security that offers and the viability secure delivery through the network. LoRaWAN uses two layers of security characterized by the protection of data at both, the link layer and the application layer.

In the application layer, the data is encrypted between the end-node and the application server while for the link layer works to ensure data integrity between the end-node and the network server [21].

- **mutual authentication**

Also known as bidirectional authentication, it refers to two parties authenticating each other at the same time. For that purpose, two keys are used:

  - A unique 128-bit **network session key** shared between the end-node and the server, known as **NwkSKey**. It is used to check the veracity of each message.
  - A unique 128-bit **application session key** shared end-to-end at the application level, known as an **AppSKey**. It is used to encrypt and decrypt the sent message.

- **Integrity and confidentiality**

The session keys discussed above are used to protect all the traffic in a LoRaWAN network. In addition, the AppSKey is used for the final encryption between the end-node and the server as well as to calculate the message integrity code (MIC verification) guaranteeing the integrity between the end-node and the server.

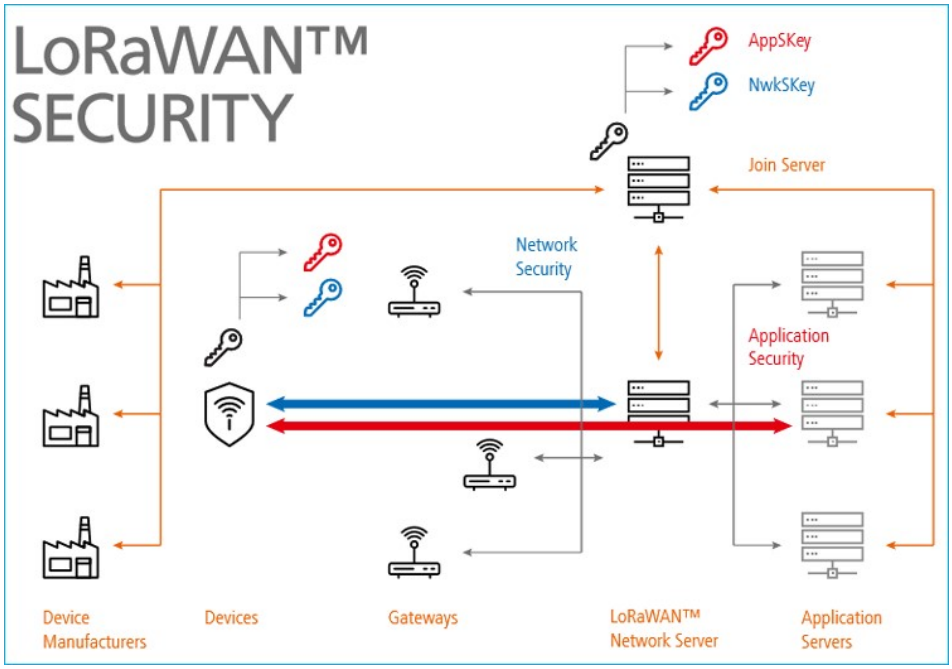


Figure 20. security [23]

In summary, using the benefits that LoRaWAN offers at the security level, it is possible to implement a shared network of multiple users without the network operator having visibility of the shared data.

4.3.2.6 LoRaWAN packet structure

LoRaWAN packets follow a structure defined in the LoRaWAN protocol specifications. All LoRaWAN messages (figure 21) have a preamble field of 8 bytes in length, a header (PHDR) and the Payload, the last two with their CRC (Close-Range Correction) [21]

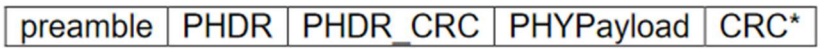


Figure 21. LoRaWAN message

* Only uplink messages

The physical layer Payload has a MAC header, the MAC Payload and a Message Integrity Code, a four-byte code that is calculated from the Network session key (NwkSKey) (figure 22).

MHDR	MACPayload	MIC[6]
1 byte	1-M bytes	4 bytes

Figure 22. The physical layer Payload

6: The Message Integrity Code (MIC) is a key to authenticate the message generated from the MHDR and MACPayload fields.

The MAC header specifies the message type and the version of the frame format of the LoRaWAN layer specification with which it has been encoded. There are eight types of MAC messages (figure 23).

MType	Descripción
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU[7]
111	Proprietary

Figure 23. MAC header

7: Reserved for Future Usage

The MACPayload includes a frame header, an optional port field, and an optional frame Payload field (figure 24).

FHDR	FPort	FRMPayload
7-23 bytes	0-1 bytes	0-N bytes

Figure 24. MACPayload

The frame header contains the address with which the device is identified within the network, an FCtrl field to enable the Adaptive data rate, a frame counter and a FOpts field in the event

that a MAC command is to be transmitted. The FPort field is used to determine whether the FRMPayload field contains MAC commands or application data (figure 25).

DevAddr	FCtrl	FCnt	FOpts
4 bytes	1 byte	2 bytes	0-15 bytes

Figure 25. The frame header

The seven most significant bits of the DevAddr field are used for the network identifier (NwkID) while the remaining twenty-five correspond to the network address (NwkAddr), which can be assigned by the network administrator.

The maximum size of the MACPayload field varies depending on the frequency band used, the data rate and the absence of the control field (FOpts), as can be seen in the next figure:

DataRate	M (bytes)	N (bytes)
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222

Figure 26. Maximum frame size in the 868 MHz band

4.3.2.7 Communication between nodes

Unlike what happens in LoRa networks where the only exchange of messages occurs between the nodes and the gateway, with LoRaWAN the network server communicates with the sensors through the gateway [16].

Although communication between modules and gateway is carried out in the same way as in LoRa networks, communication is different on the other side of the gateway since it is responsible for converting LoRaWAN packets into UDP packets and vice versa. In addition, there are the messages that the gateway receives from the sensors and retransmits to the network server (Upstream) and the network configuration messages that the network server sends to the sensors (Downstream).

In figure 27, we see that the node can transmit as many consecutive messages as the number of channels it has enabled (N). When the node transmits an acknowledged message, it reaches the gateway, which converts the message into a UDP packet and forwards it to the network server. Although the message does not require confirmation, the gateway does receive a response from

the network server. Once it has transmitted on the last available channel, the node will not be able to transmit again until the duty cycle of each channel has passed.

It is possible to transmit acknowledgment messages making that the gateway sends an ACK to the node after receiving it from the network server.

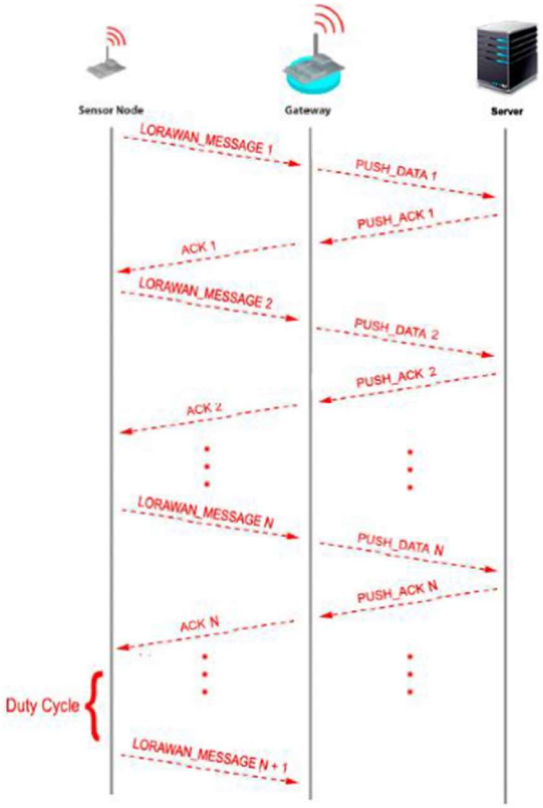


Figure 27. Upstream communication Node/Gateway/Server with confirmation

The PUSH_DATA packets include the information transmitted by the sensors to the gateway, a randomly generated token, and the MAC address of the gateway. The server will return a PUSH_ACK with the same token as the PUSH_DATA and will process the information from the gateway. The server processes the packets after sending the corresponding ACK.

On the other hand, when the network server wants to send messages to the gateway, the communication is somewhat peculiar (figure 28). Despite being downstream communication, it is the gateway that initiates it, because the opposite would not be possible if the gateway was behind a NAT. The gateway sends PULL_DATA packets periodically so that the NAT is always open and the server can send PULL_RESP messages through the same port that the PULL_DATA messages leave open.

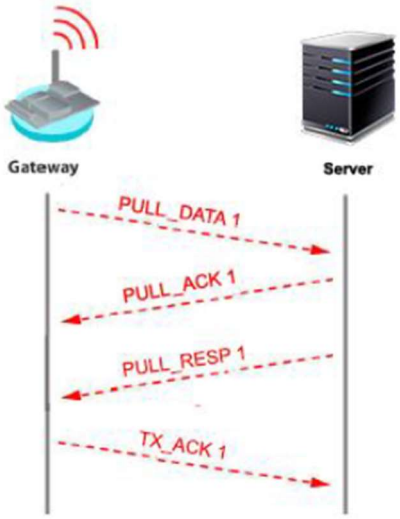


Figure 28. Downstream Communication Server/Gateway

4.3.3 The Things Network

The Things Network (TTN), is a global, decentralized, free, open IoT network in which anyone can contribute their knowledge about IoT in order to improve the platform more and more.

TTN was created in the summer of 2015 in Amsterdam, by its two founders Wienke Giezeman and Johan Stokking, among others, and whose objective was to facilitate the design and creation of networks, for which it supported gateways distributed throughout the world and also it offered the backend infrastructure to its users, that is, an implementation of the network server and application server functionalities.

This backend deals with message duplication, manages the transmission of downstream messages, manages integrations with platforms such as HTTP or MQTT and offers a series of APIs in different programming languages such as Java, Node.js or Node-Red.

4.3.3.1 The Things Stack

The Things Stack is an open source network stack available to a global, broad, and distributed public for deploying both private and public networks.

Since its creation, versions V.0, V.1, V.2 and V.3 have been published year after year, the evolution of which can be seen in figure 29 [24].

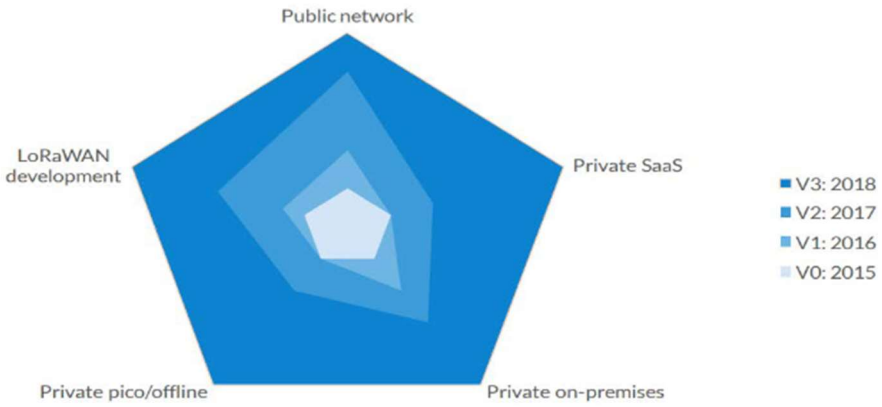


Figure 29. Evolution of LoRaWAN stack versions

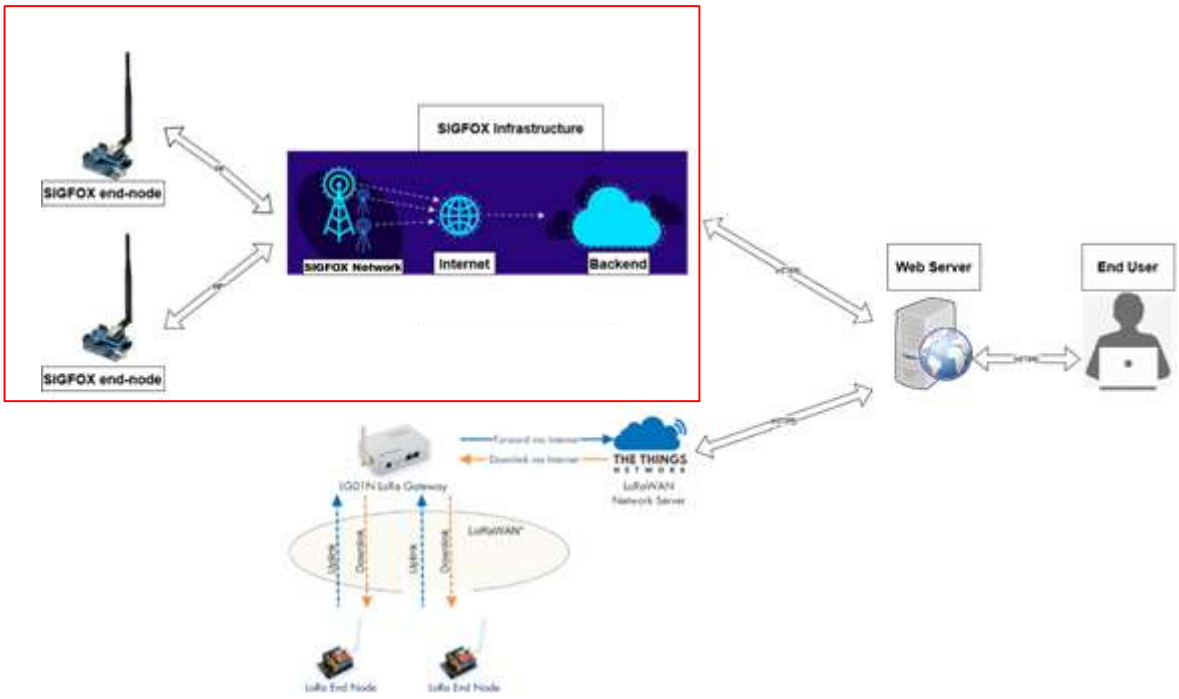
Observing figure 29, a great development can be seen from V.2 to V.3 since in V.2 the area of the TTN public network stood out above the others, but there has been a great advance not only in that aspect, but all the blocks have been optimized giving rise to a very powerful stack.

5. DEPLOYMENT OF THE PROPOSED NETWORK

In this section, we are going to design, configure and deploy the network proposed in the introduction of this master's thesis. Firstly, we are going to introduce the characteristics of the components that are part of the network which will monitor the noise pollution, both in Sigfox and in LoRaWAN parts, as well as, we are going to configure the backend of Sigfox and The Things stack (TTN) of LoRaWAN.

5.1 Sigfox part

In reminder mode, we are going to put the general scheme of the proposed network.



5.1.1 Used hardware

5.1.1.1 Arduino

Arduino is an open source platform, based on flexible and easy-to-use hardware and software. It has several analog and digital inputs that allow it to read information from its environment, as well as connect to the Internet through external modules.

The microcontroller on the board is programmed using the “Arduino Programming Language” and the “Arduino Integrated Development Environment, IDE”. The software can be downloaded free of charge from the official web page of Arduino. [26]

Arduino has several advantages, the most important are:

- **Cheap**: Arduino boards are relatively cheap compared to other microcontroller platforms. The least expensive version of the Arduino module costs less than 20 euros [9].
- **Multi-platform**: The Arduino software runs on Windows, Mac, and Linux operating systems. Most microcontroller systems are limited to Windows.

- **Simple and clear programming environment**: The Arduino programming environment is easy to use and intuitive.
- **Open source and extensible software**: The Arduino software is published as open source tools. The language can be expanded using C++ libraries.
- **Open source and extensible hardware**: Arduino is based on ATMEGA8 and ATMEGA168 microcontrollers from Atmel. The plans for the modules are released under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending and improving it. Even relatively inexperienced users can build the board version of the module to understand how it works and save money.

Arduino has several types of boards for various purposes. In this master's thesis, we have chosen to use the Arduino DUE board due to its high computational capacity, which will allow us to execute the algorithm to calculate the acoustic noise index.

- Arduino DUE

This microcontroller is similar to other Arduino boards, but it has a higher computational power, which makes it perfect to use in this master's thesis.

Features:

- Microcontroller: AT91SAM3X8E
- Voltage at which it works: 3.3 V
- Recommended input voltage: 7-12 V
- Digital input/output pins: 54
- Analog input pins: 12
- Analog output pins: 2
- Continuous output current on the input/output lines: 130 mA
- Flash Memory: 512KB
- SRAM: 96KB
- Clock frequency: 84MHz

Figure 30. Arduino DUE

5.1.1.2 Sigfox module

This module, provided by Libelium, makes possible to send information to the Sigfox network. The module identifier allows the cloud to know which device is sending the information and it

will be included in the messages transmitted to it, without affecting the payload limit of 12-byte messages.

Figure 31. Sigfox module and antenna

The main features are:

- Frequency: ISM 868 MHz
- Transmission power: 16 dBm.
- Consumption:
    - Transmission: 51 mA at 16 dBm.
    - Receive: 16mA.
- Reception sensitivity: -126 dBm.
- ETSI limitation: 140 messages of 12 bytes, per module per day.
- Sigfox Certificate: Class 0u (Sigfox radio solution certificate [Sigfox Verified] and finished product certificate [Sigfox Ready]).

The module is powered at 3.3V. The consumption for different states is:

- Off 0mA.
- Transmitting 52 mA data.
- Receiving data 13 mA.

The consumption time until the sending process is completed is a few seconds. During the course of that time, the current consumption specified above will occur.

Table 3. Elapsed time depending on the transmission mode of the Sigfox module.

Transmission Mode	Time elapsed
Send from power on state.	[- 6 seconds] 6 seconds to send package.
Send from power off state.	[- 12 seconds] 6 seconds to power on and 6 seconds to send to the network.
Send with ACK from power on state.	[-39 seconds] 6 seconds to send to the network, 14 seconds to enter the data waiting state, and 19 seconds to receive the packet.
Send with ACK from power off state.	[- 45 seconds] 6 seconds to power on, 6 seconds to send to the network, 14 seconds to enter the data waiting state, and 19 seconds to receive the packet.

5.1.1.3 Communication shield

To send data from the Arduino DUE, the "Communication Shield" will be needed. This board allows communication between devices using XBee, Bluetooth, RFID or Sigfox.

As mentioned, this board is compatible with Arduino DUE, which will be used in this master's thesis. Thus, the Arduino will be able to communicate with the Sigfox module.

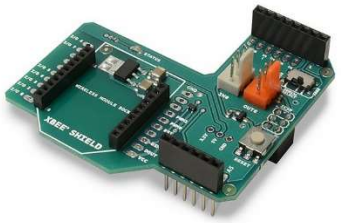


Figure 32. Communication shield

In this way, and after assembling all the hardware, compatibility is achieved.

5.1.1.4 Sensor to monitor noise pollution

The module incorporates a 20KHz electret microphone amplified by the MAX4466, a chip designed specifically for the task of preamplifying microphones. This op-amp has excellent power supply noise rejection. On the back it has a small potentiometer to adjust the gain from x25 to x125, the op-amp output is rail to rail, so at maximum gain it can reach 5Vpp. The output is not amplified for a higher power than for small hearing aids, in case we need more power it is necessary to connect a power amplifier.

Technical specifications:

- Supply Voltage Operation: 2.4 - 5.5V DC
- electret microphone
- Pre-amplifier chip: MAX4466
- Power Supply Rejection Ratio: 112dB
- Common-Mode Rejection Ratio: 126 dB
- Output: Rail-to-Rail - up to 5vp-p
- Frequency Response: 20Hz - 20 KHz
- Adjustable Gain 25x-125x

Figure 33. MICROPHONE MODULE WITH AMPLIFIER MAX4466

5.1.1.5 Arduino DUE programming

The programming of the Arduino DUE will be done through "Arduino IDE" (Integrated Developed Services), an open source programming environment that allows easy writing of code and uploading it to the Arduino.

The Sigfox module has a C++ library, which allows it to be managed in a simple way through a set of functions. It is a free code system, which can be accessed through the Internet [20].

5.1.2 Sigfox backend

The Sigfox backend provides a web application interface for device management and configuration of data integration, as well as standards based web APIs to automate the device management and implement the data integration.

The APIs are based on HTTPS REST requests, as GET or POST and the payload format is JSON.

5.1.2.1 Activation process

The first thing we need to do is to contact Sigfox in order to buy the connectivity service for the modules with which we want to transmit to the Sigfox network.

Sigfox will give instructions to us about how to do it [15]. Basically, we have to create an account with a valid email address and a password. Then we will create our group, user, device type and device units. We need to pair the **Serial Number** and **PAC number** of every module with a license.

5.1.2.2 Login

Once we complete the activation process, we will have the credentials (email address and password) needed to connect to the system in https://backend.sigfox.com/auth/login

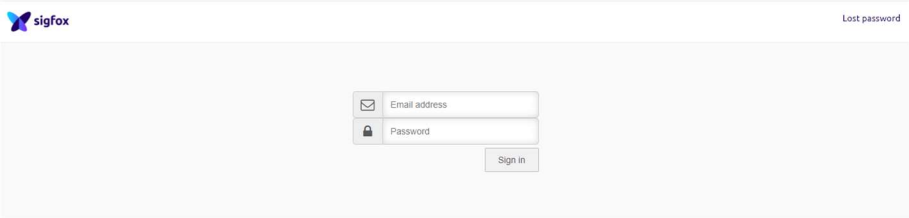


Figure 34. Login form

Once logged in, we arrive to the main page of the Sigfox backend with the latest news and updates, events, coverage map and known issues.

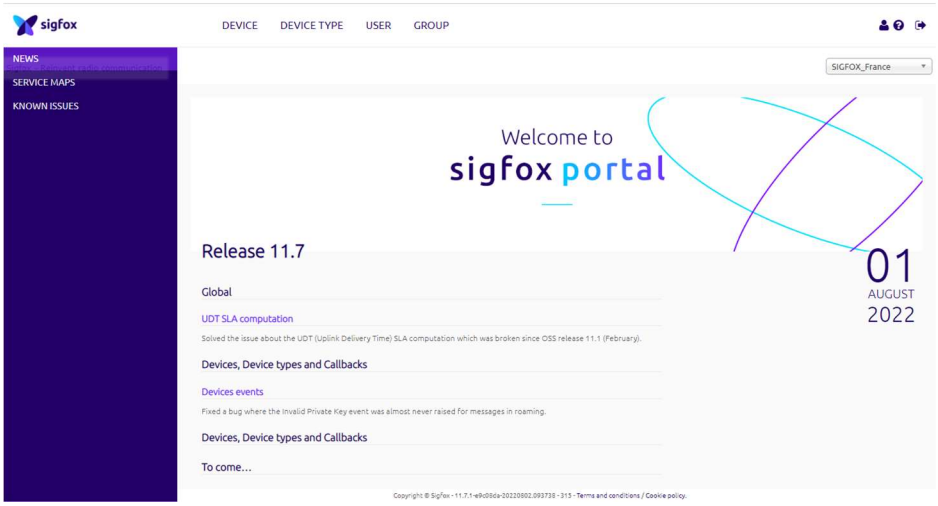


Figure 35. Sigfox backend main page

5.1.2. Device

By clicking on “Device” tab (top menu), we will get information related to all devices associated to our account. The operations allowed on this list are:

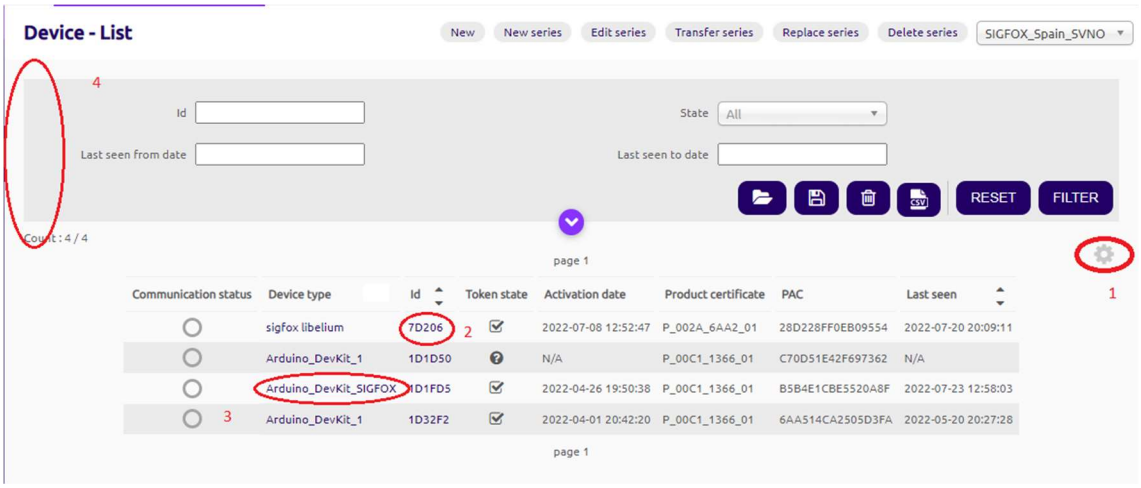


Figure 36. Sigfox backend - Device

1. By clicking on this icon we can select the columns we want to see on the table.

2. This is the most important operation because it will take us to all the information related to this specific device. On the left side of the window, a menu will appear with all the possibilities available for a single module.

3. Device type associated to the device.

4. The gray zone is a filter form where we can perform our devices searches.

Once a specific device ID is clicked, the backend leads us to that specific device information area.

- **Information**

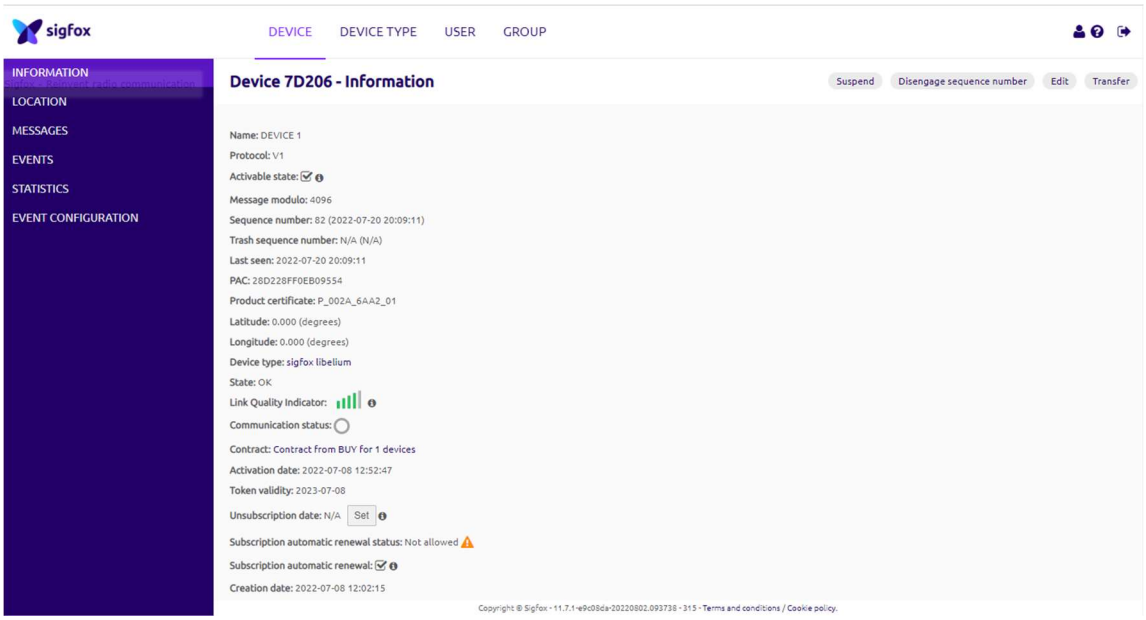


Figure 37. Sigfox backend - Device information

- **Messages**

In this section we can find all information about all messages received on the Sigfox platform. It is possible to see several columns for:

- The packet arrival timestamp
- The packet data
- Location
- Signal strength
- Packet transmission status depending if callback or ACK is required

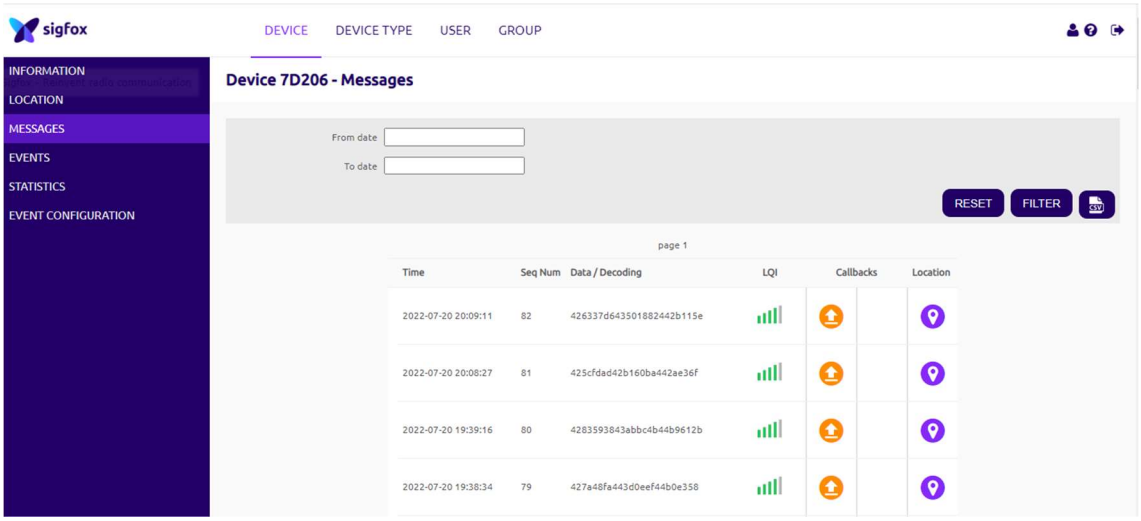


Figure 38. Sigfox backend - Device messages

5.1.3. Device type

In the “Device Type” tab, we will find information about the characteristics of our device. Firstly, we will find the same table as in Device tab. Besides, there is a “New” button to create new Device Types.

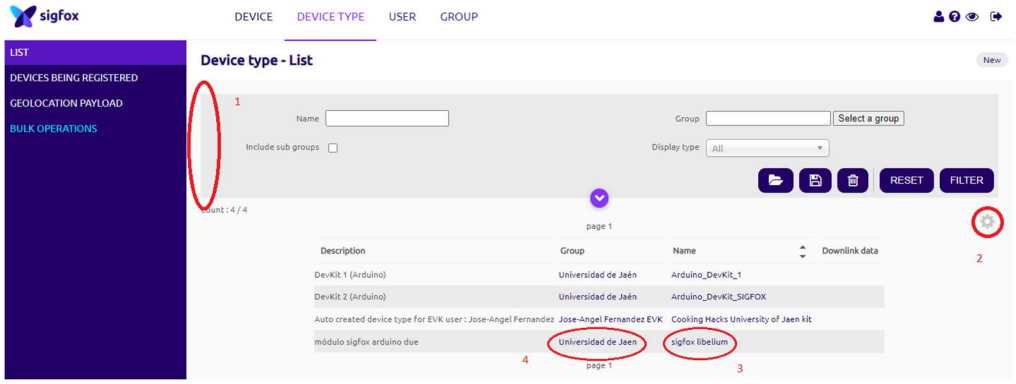


Figure 39. Sigfox backend - Device type list

1. The gray zone is a filter form where we can perform our device-type searches.

2. By clicking on this icon we can select the columns we want to see on the list.

3. This link takes to the Device Type Information area. Information is shown divided in sections that we can access clicking on the left menu.

4. This link shows the group information page which the device type selected belongs to.

Once a specific device type is clicked, the backend leads us to that specific device type information area.

- **Edit**

On the top right side of the website we find the “Edit” button in order to change the settings of the device type as we consider appropriate for our requirements.

In the “Edition” window it is possible to change:

- **Device type information**: in this section, we can change the device-type name, add a description to it, configure keep-alive, choose the contract associated to this device-type, enable/disable the subscription automatic renewal option and add the alert email in case a callback fail.
- **Downlink data:** we can choose the downlink mode between DIRECT or CALLBACK, as well as we can configure the downlink data in hexadecimal. In our case, we have chosen the CALLBACK type.
- **Display type:** here we can create the format of our messages. In our case, we have chosen the regular format (raw payload).

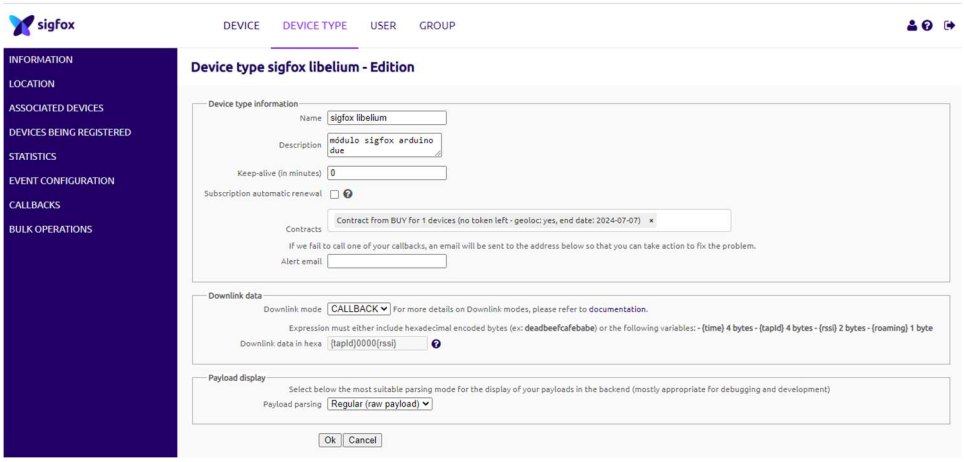


Figure 40. Sigfox backend - Device type edition

- **Information**

On the left side of the device-type window it is possible to select the information area. It shows all information of the selected Device Type.

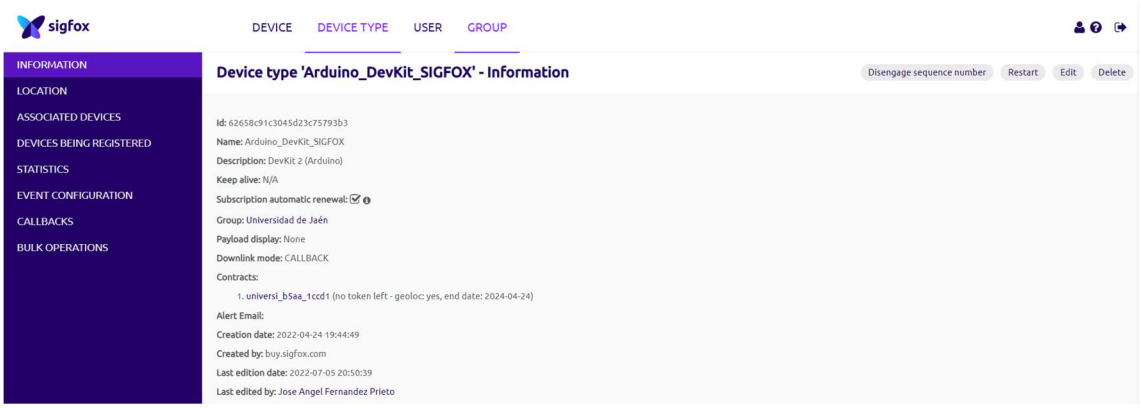


Figure 41. Sigfox backend - Device type info

- **Associated devices**

Shows a device list (same as *Devices* section) with all devices associated to this specific device type.

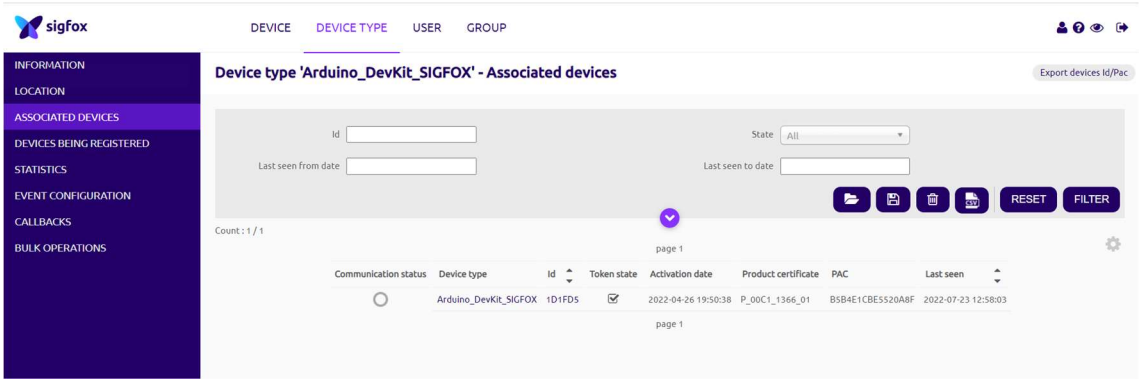


Figure 42. Sigfox backend - Device type asociated devices

5.1.4 CALLBACKS

The backend can automatically forward some events using the callback system. The “Callbacks” button in the “Device Type” section permits to create new callbacks associated to that device type. The callbacks are triggered when a new device message is received or when a device communication loss has been detected.

There are three different callback types:

- DATA
- SERVICE
- ERROR

There is a set of available variables for each type of callback. These variables are replaced by their value when a callback is called.

Firstly, in this page we will see all the callbacks we configured.

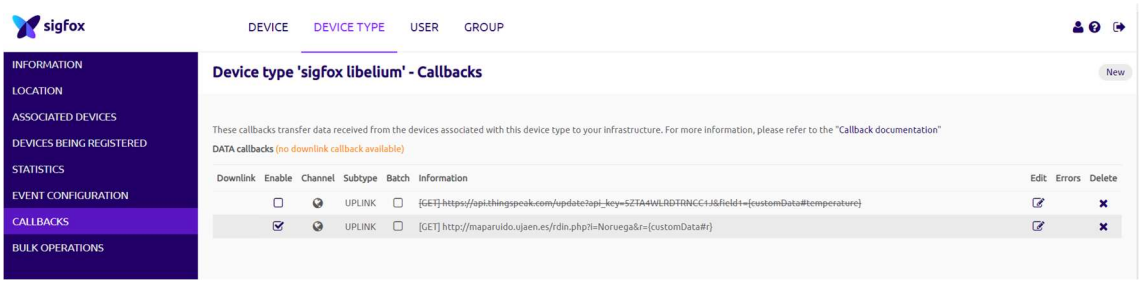


Figure 43. Sigfox backend - callbacks

In this master thesis, we have to configure a callback in order to be triggered whenever a new message is received. This callback should send three parameters: the noise indicator (r), the frequency (freq) and the maximum noise (rmax). As we stated before, the maximum payload that we can send are 12 bytes. The three values mentioned before are of type float, so each one of them is 4-byte length.

The configuration of the callback will be the following:

- **Type**: DATA, UPLINK
- **Channel**: URL
- **Custom payload config**: r::float:32 freq::float:32 rmax::float:32

The used pattern is as follows: variable-name::data-type:number of bits used

- **URL pattern**: http://maparuido.ujaen.es/rdin.php?i=Noruega&r={customData#r}

  As we can notice in the previous URL, we are using the http protocol. Nevertheless, we have stated before that one of our goals is to make all the communications secured using the https protocol instead of http.

  In the previous URL pattern, if we use https instead of http, the callback will give an error message. We have contacted the technical support of Sigfox in Spain and the gave us the following answer:

"Our Cloud team confirmed that the issue is caused by the certificate not being trusted by our systems.

They have been conducted tests using a very up-to-date CA database from Mozilla (https://curl.se/ca/cacert.pem) and got the same result, leading us to conclude that this CA is not widely trusted."

Figure 44. Email from Sigfox

As we could read in the previous email, the Sigfox backend does not trust our Certification Authority. Finally, we have decided to use a provisional solution until this CA problem is solved.

The proposed solution is the use of “ngrok” tunnel [https://ngrok.com/]. Ngrok is a tool that allows us to make our localhost public in internet and establish https connections to it using SSL/TLS.

The way that ngrok is working is very simple (see figure 45 and steps below).

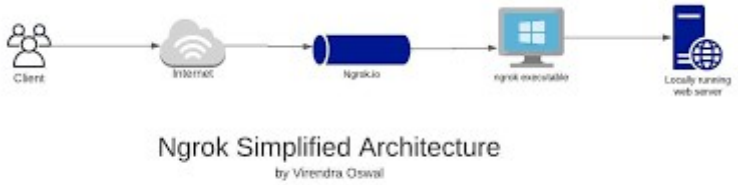


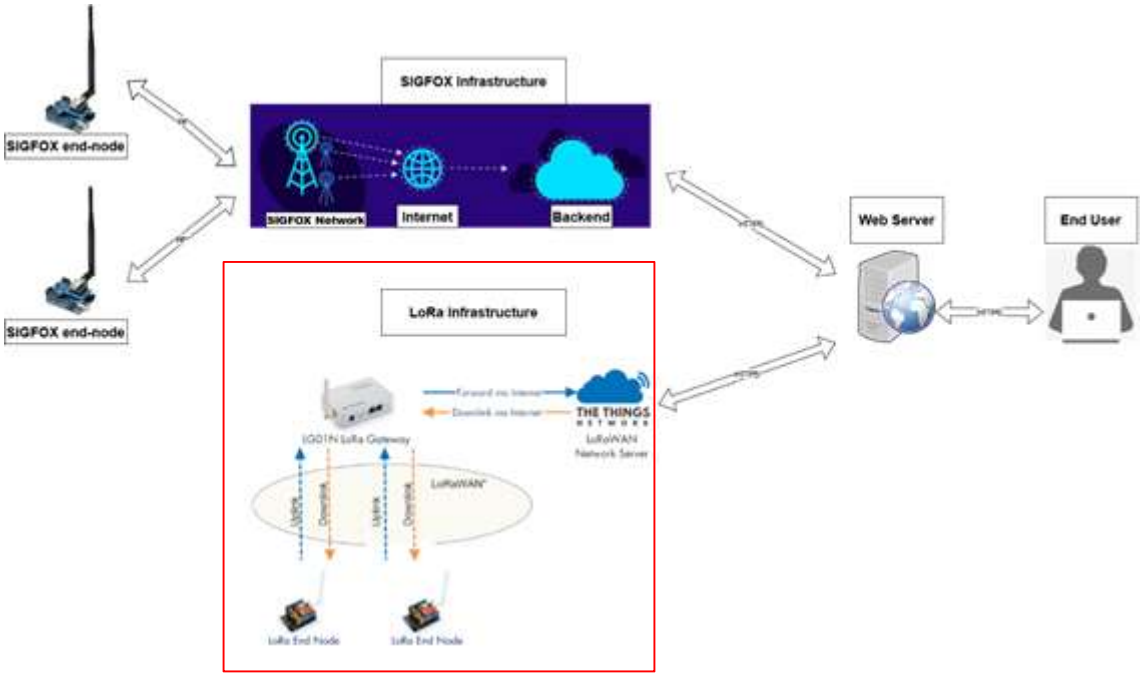
Figure 45. Ngrok simplified architecture

- **Step 1**: Create a new account in ngrok.com
- **Step 2**: Download ngrok for Windows
- **Step 3**: Extract the ZIP file and execute the .exe
- **Step 4**: copy the link for https and paste it in the url pattern of the backend of Sigfox.

- **Use HTTP Method:** GET
- **Send SNI (Server Name Indication):** Checked (for SSL/TLS connections)

5.2 LoRa/LoRaWAN part

In reminder mode, we are going to put the general scheme of the proposed network.



5.2.1 Used hardware

5.2.1.1 LG01 – N Single Channel LoRa IoT Gateway

LG01N is an open-source LoRa Gateway with a single channel. LG01N have a variety of internet connectivity options, including a Wi-Fi interface, an Ethernet port, and optional 3G/4G cellular. These interfaces enable users to connect their sensor networks to the Internet in a variety of ways. LG01N can support multiple working modes, including MQTT mode and TCP/IP Client mode, to meet a variety of IoT connectivity needs. [27]

Some important technical Specification for Dragino Gateway are as followed [28]:

- Processor: 400MHz, 24K MIPS
- Flash: 16MB
- RAM: 64MB
- 10M/100M RJ45 Ports x 2
- Wi-Fi: 802.11 b/g/n
- USB 2.0 host connector x 1
- USB 2.0 host internal interface x 1
- 1 x LoRa Interface

Figure 46. LoRa Gateway

5.2.1.2 Access and configuration for LG01-N Gateway

The LG01N is configured as a Wi-Fi Access Point by default. We can access and configure the LG01N after connecting to its Wi-Fi network, or via its Ethernet ports.

we can configure the LG01-N via Ethernet port or after connecting to its Wi-Fi network. By default, LG01-N is configured as a Wi-Fi Access Point. Following, are the different methods to configure LG01-N. [28]

- Connect via Wi-Fi
- Connect via Wan port with DHCP IP from router.
- Connect via LAN port with direct connection from PC.
- Connect via Wi-Fi with DHCP IP from router.
- Connect via LAN port by fall back IP.

In this master thesis, we are going to use the LG01-N as LoRa Gateway. Since we need to connect to our proprietary server, we need to configure the LG01-N to have internet access.

Figure 47. Connect via Wi-Fi

Configure LG01-N for internet connection

Below steps show how to set up LG01-N to use WiFi for internet access.

- Step 1

Connect PC to the wifi AP netwok (SSID: dragino-1ed808, password: dragino+dragino). The PC will then get IP from LG01-N. The ip range is 10.130.1.xx

- Step 2

Open any browser and type this url: http://10.130.1.1/luci, afterwards, we will see the interface of LG01-N.

The credentials for web login are:

- User Name: root
- Password: dragino

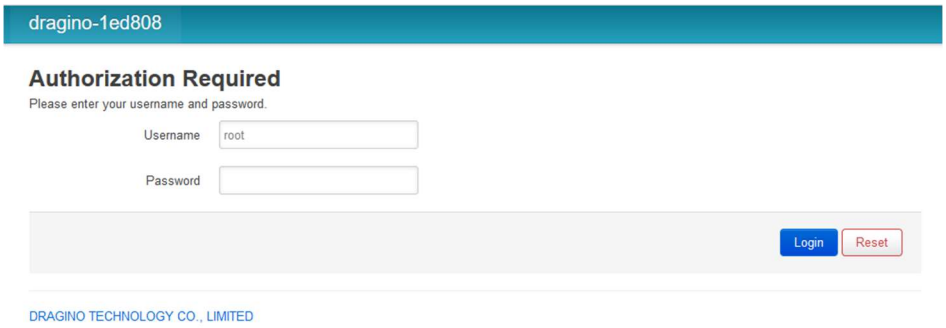


Figure 48. Login form Dragino

- Step 3

Inside, Network → Wireless, select "radio0" interface and scan.

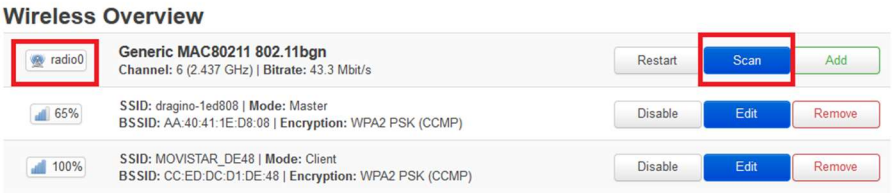


Figure 49. Wireless Overview - Dragino

- Step 4

Select the desired AP and join the Wi-Fi network. Afterwards, only we have to introduce the password to join this network.

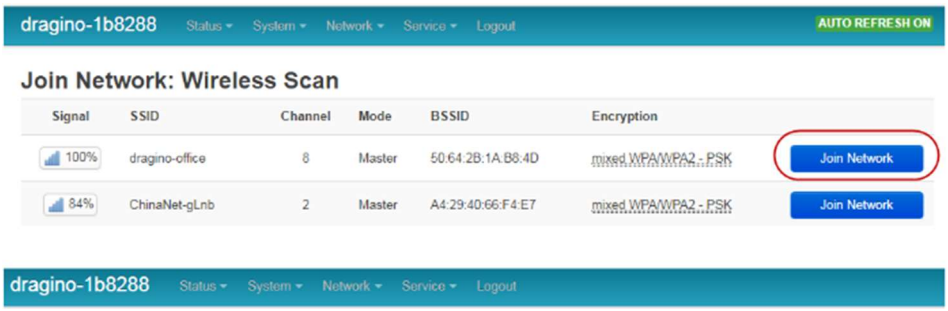


Figure 50. Wireless Scan - Dragino

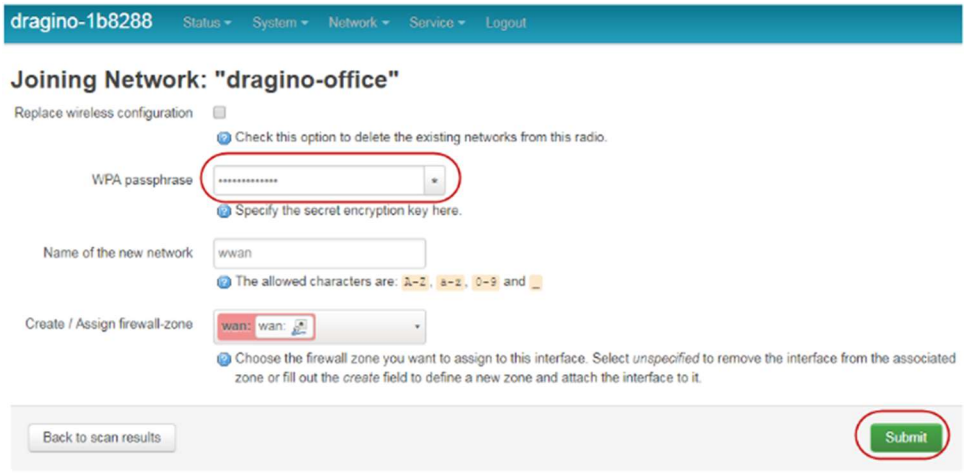


Figure 51. Joining Network - Dragino

After every change we make, we have to save and apply.

Finally, going to Network → Interfaces, we can see that LG01-N is connected to internet through the router with SSID: MOVISTAR_DE48. Also, we can notice the IP addresses are assigned through DHCP.

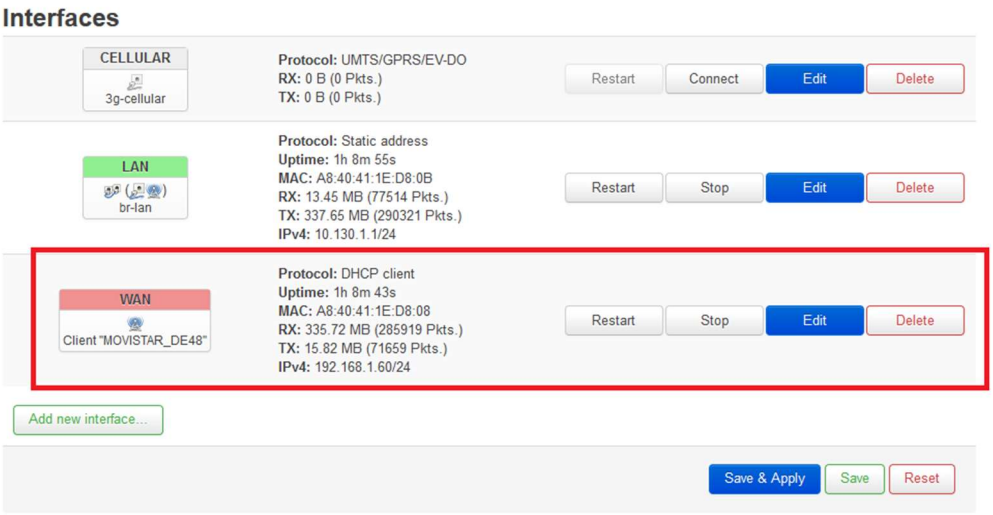


Figure 52. Interfaces - Dragino

Configure LG01-N as a LoRaWAN Gateway

In order to send the monitored data from the end node to the IoT server via LG01-N, we need to follow one of the methods given below so that communication between the gateway and the IoT server is established. [28]

- MQTT transfer mode.
- TCP/IP client mode.
- Write customized script.
- Communicate using HTTP server.
- Configure as a LoRaWAN gateway.

In this master thesis, we have configured the LG01-N as a LoRaWAN gateway. LG01-N sends uplink LoRa packets from the end node to the LoRaWAN server. It will also send the LoRaWAN server's downlink LoRa packet to the end node. In the LoRaWAN protocol, end nodes can use the ABP or OTAA modes [28]. As shown in the following figure, frequency selected here is 868.1MHz which should be the same as in the end node device as well as the same frequency plan should be selected in IoT server. Spreading factor selected here is SF7, Bandwidth 125Khz which must match with end node and same should be selected in IoT server.

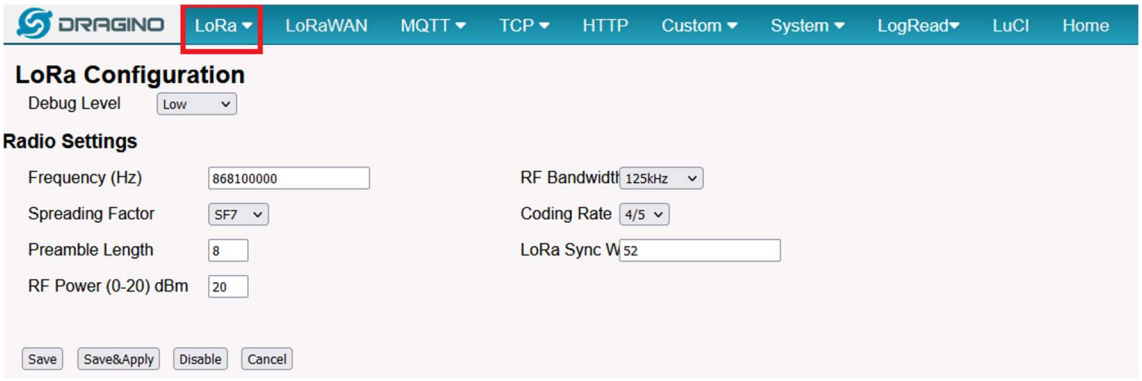


Figure 53. LoRa configuration - Dragino

As shown in the following figure, the gateway ID generated in this section, should be used in the IoT server as an identification of the respective gateway. LoRaWAN Service Provider is custom and Server Address which is shown in the figure should be kept same in the IoT server.

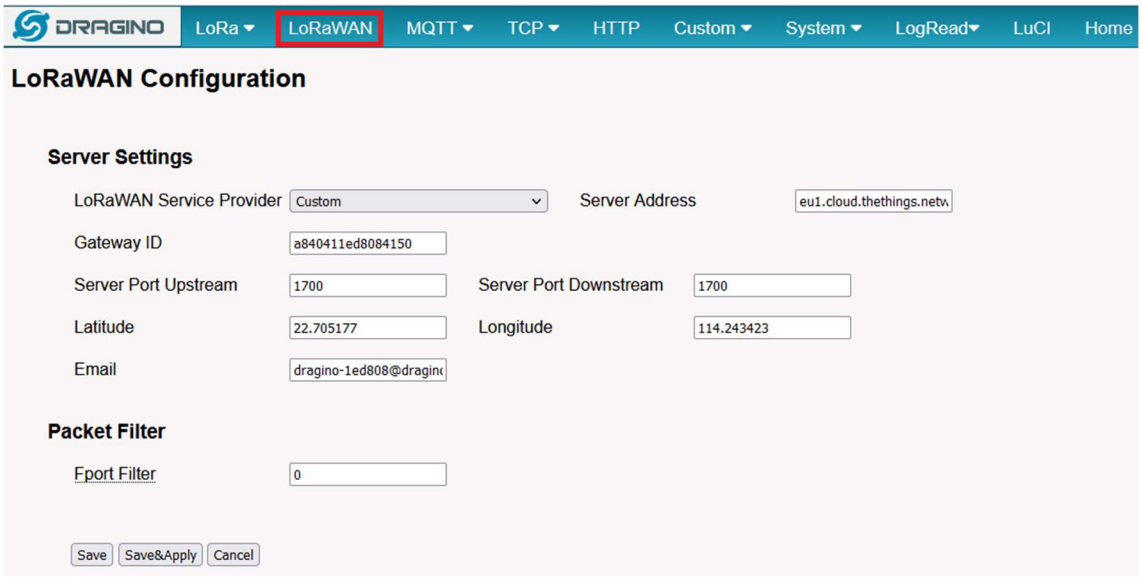


Figure 54. LoRaWAN configuration - Dragino

Limitations for using LG01-N as LoRaWAN.

The single-channel gateway LG01-N can only operate at a single frequency and data rate. In case of single channel gateway if end node is not set at fix frequency than it LG01-N will lose the majority of packets. To solve this problem, the end node must be set to the fix frequency. A basic LoRaWAN sensor may operate at various frequencies (3 - 72 depending on frequency bands) and data speeds. [29]

5.2.1.3 LoRaWAN module

5.2.1.4 End-node programing

Arduino DUE

As in Sigfox part, we are going to use Arduino DUE as a microcontroller along with LoRaWAN shield.

LoRa shield SX1276

The LoRa Shield is a long-range transceiver Arduino shield based on Open-source library. The LoRa Shield enables users to communicate data over great distances at low data rates. It provides long range spread spectrum communication with high interference immunity while consuming the least amount of current. The LoRa Shield has Semtech SX1276/SX1278 chips used for various applications such as irrigation systems, smart metering, smart cities, smartphone detection, building automation etc. [30]

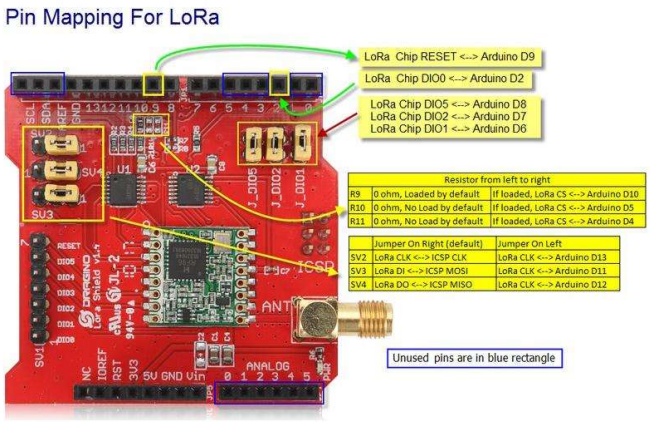


Figure 55. LoRa shield SX1276 [30]

LMIC library

Following are the steps to establish the communication between LoRa shield, Arduino and TTN.

- **Step 1**: Install the LMIC library in the Arduino IDE. Go to Tools and then search LMIC in manage libraries and then install.
- **Step 2**: Open the LMIC configuration file. In that file, uncomment the frequency band which is provided by manufacturer behind the LoRa Shield. In this master thesis, the frequency band is 868MHz.

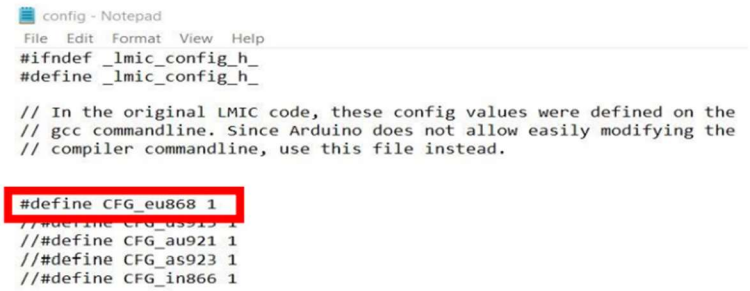


Figure 56. Frequency band – LoRa end node

- **Step 3**: In order to make the LMIC library work, we need to connect the Arduino to LoRa Shield. The connections depend on the type of Arduino board. In this master thesis, Arduino DUE is used. Following figure represents the pin configuration.
- **Step 4**: With respect to above connections following Pin mapping should be done in Arduino code.

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};
```

Figure 57. pin mapping LoRa end node

- **Step 5**: As we are using LG01-N as a LoRaWAN gateway in ABP mode, it is necessary to define the **NwkSKey**, **AppSKey** and **DevAddr.** These unique codes are generated while adding end device in TTN (will be explained in the next section). Copy these codes and paste in the Arduino IDE. Refer to highlighted part.

```
const float dry = 25;//we'll choose this number during experiment
const int pumpPin = 31;


// LoRaWAN NwkSKey, network session key from TTN - Change this with your Unique Key
// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
//ttn
static const PROGMEM u1_t NWKSKEY[16] = { 0x29, 0xDE, 0x26, 0x68, 0x7E, 0x25, 0x82, 0x18, 0xD9, 0xCB, 0x2F, 0x10, 0x3B, 0xD9, 0x28, 0x52 };
// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
//ttn
static const u1_t PROGMEM APPSKEY[16] = { 0x94, 0xF6, 0xA8, 0xD2, 0x33, 0x24, 0x81, 0xA9, 0x66, 0x10, 0xBB, 0xD7, 0x2A, 0xF7, 0x01, 0xDA };
//
// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace
// ttn
static const u4_t DEVADDR = 0x260B34DE;
```

Figure 58. ABP mode configuration end node

- **Step 6**: After executing above steps, compile and upload the code to Arduino, we will be able to send the data from LoRa node to TTN. Live data received by TTN is shown in following figure.

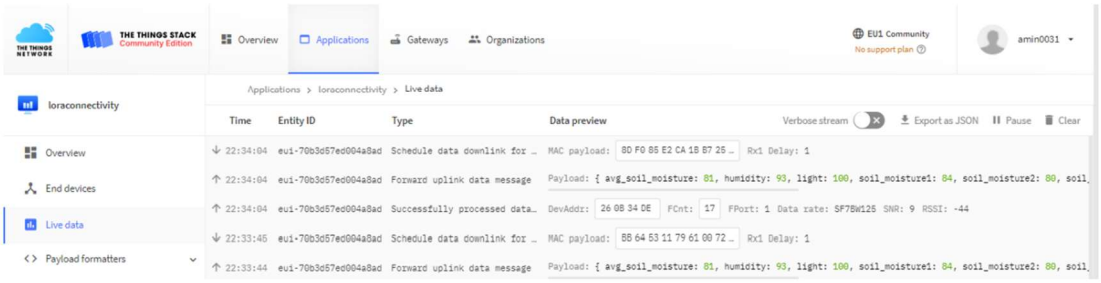


Figure 59. Live data example

5.2.2 The Things Network TTN

5.2.2.1 TTN configuration

Step 1: Open this URL “https://www.thethingsnetwork.org” and create an account (Sign up).

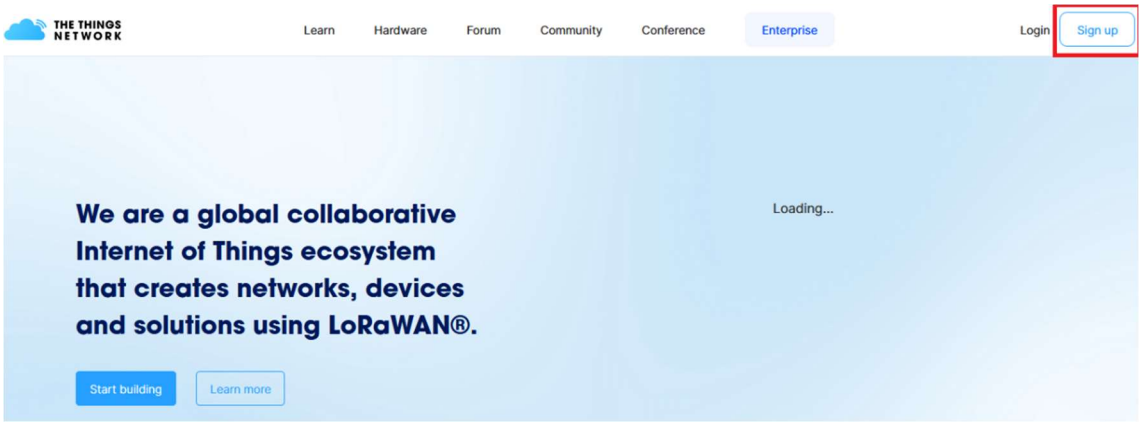


Figure 60. TTN - sign up

Step 2:

The things network has multiple plans. In this project, we will choose “**Community**” option with “**Student**” plan. The features that offers this plan will help us to fulfill the objectives of this project.

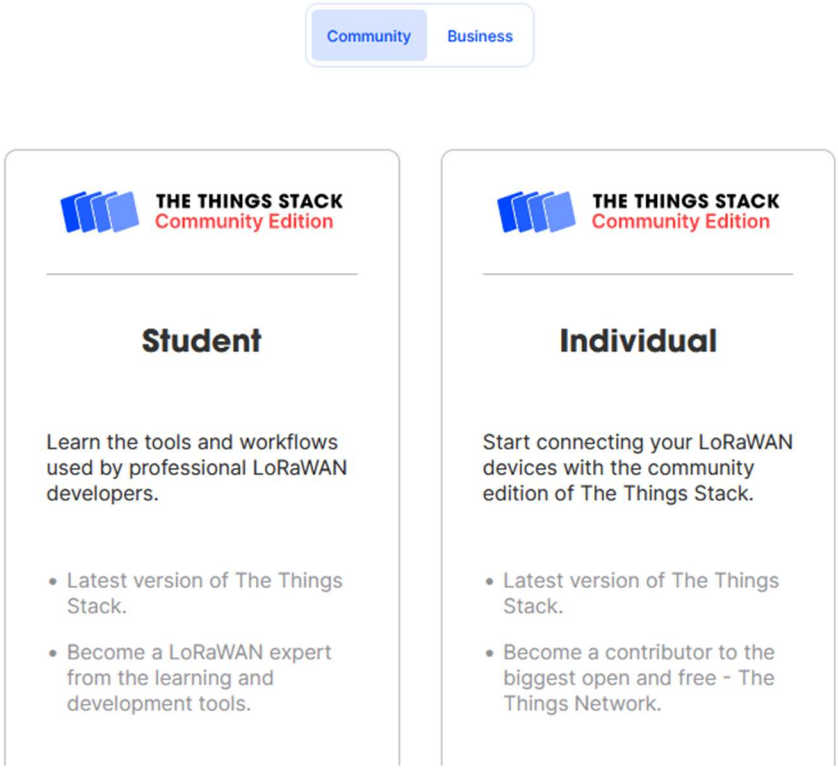


Figure 61. TTN - plans

Step 3:

The next step is to select a cluster to start adding devices and gateways. In our case, it will be "Europe 1, eu1 – Dublin, Ireland".

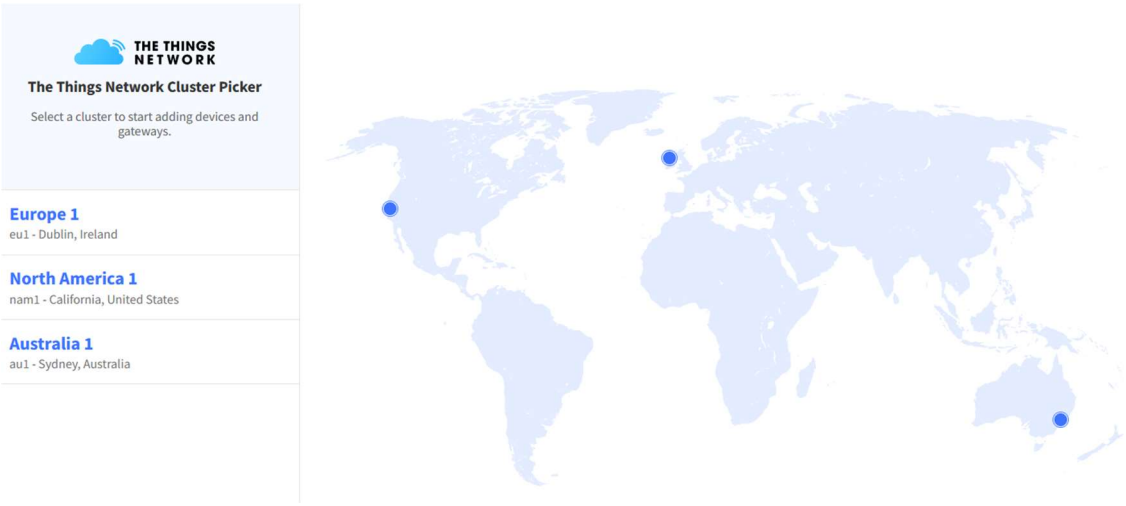


Figure 62. TTN cluster

Step 4:

Later, we will be asked to register introducing our username, email address and the password.

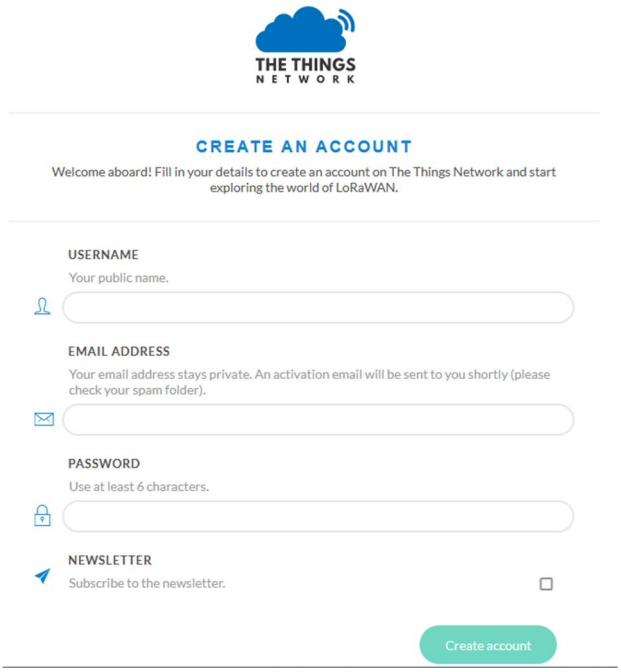


Figure 63. TTN create an account

We will not be able to use our TTN account until our account has been validated.

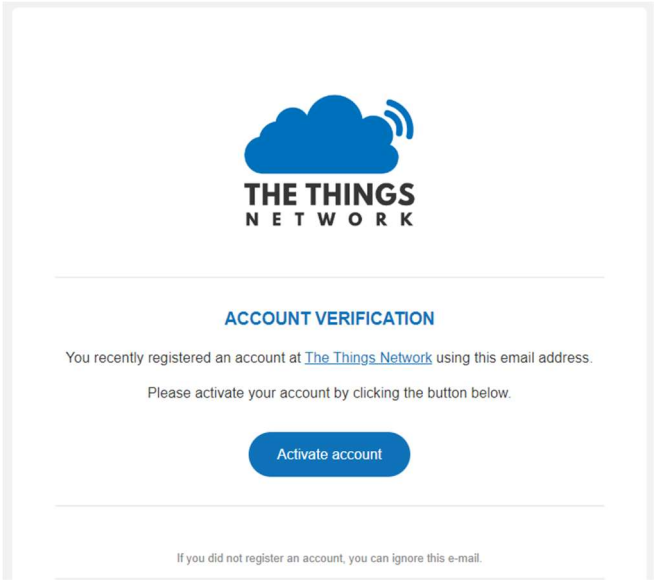


Figure 64. TTN activation

Step 5:

After activating our account, we can use our account. To do so, we have to click on “console”.

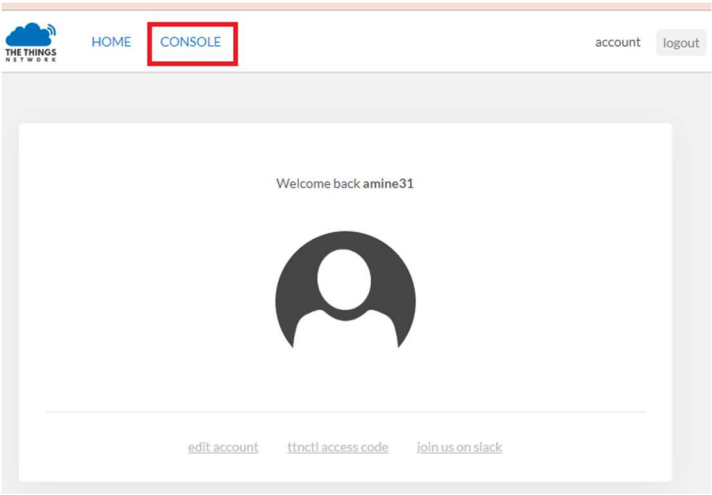


Figure 65. TTN console

Step 6:

Further, we will have two options: Create an application and register a gateway. Firstly, we will register our Dragino Gateway.

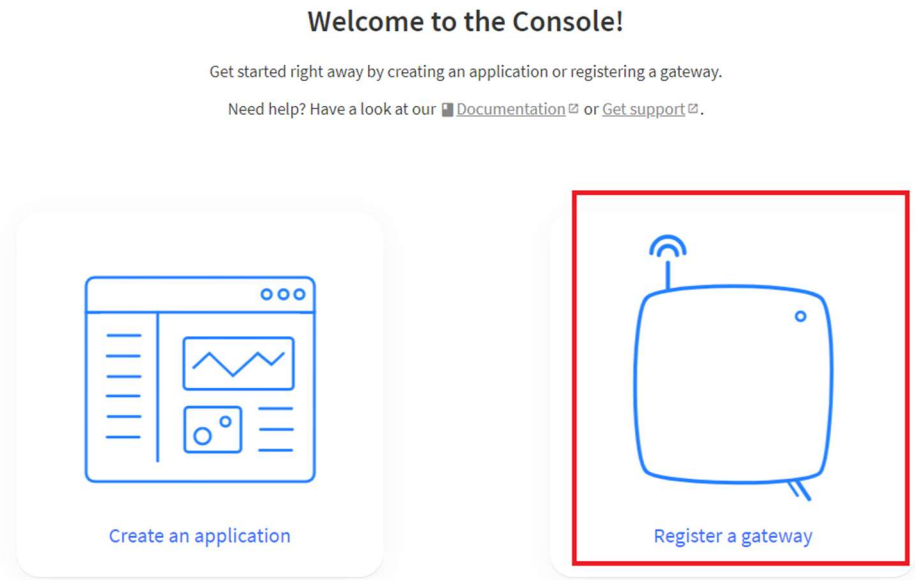


Figure 66. TTN - register a gateway

Step 7:

After clicking on "Register a gateway", we will have to fill some information related to the new gateway:

- **Owner**: It will be similar to the username which was defined initially.
- **Gateway ID**: It is a unique identifier; we can put any name as per our desire.
- **Gateway EUI**: It is a 64-bit extended unique identifier. It is provided by manufacturer. It is also found in LoRa Gateway User interface. Refer to the gateway ID, in figure 21.
- **Gateway Server address**: Server address is "eu1.cloud.thethings.network", which is same in LoRa Gateway User Interface, refer figure 21.
- **Frequency Plan**: In the frequency plan select the desired plan. The frequency plan should be same as frequency of end node and frequency plan configured in LoRa Gateway User Interface, refer figure 20. In this project frequency plan selected is EU_863_870_TTN.

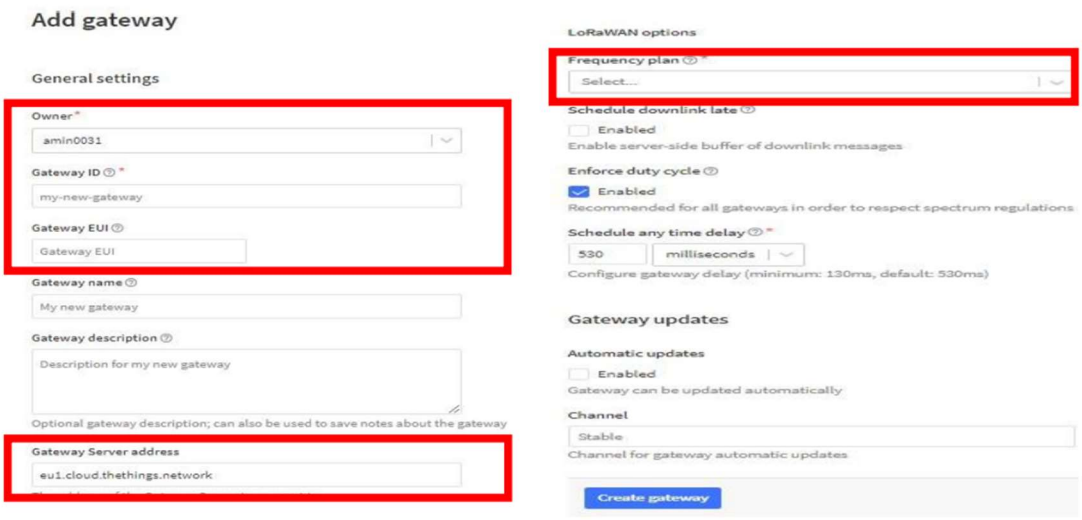


Figure 67. TTN - gateway info

Step 8:

After configuring the gateway, we can see the status of the gateway as shown in the figure below.

Figure 68. TTN - gateway status

Step 9:

If we click on the configured gateway we can see the general information about our gateway as well as the last activity status of the gateway.

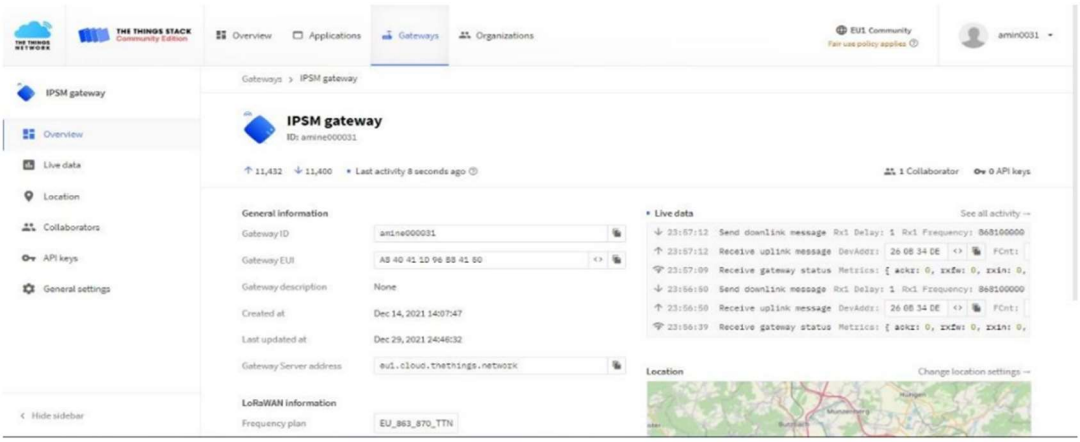


Figure 69. TTN - gateway overview

Step 10:

To configure a new application, select go to application (refer figure 22) and press on add application. Owner name is same as username, specify application ID, in this master thesis application ID is “prueba000031”, specify application name and description if required.

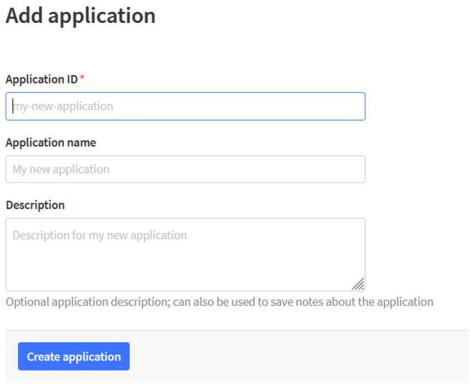


Figure 70. TTN - add application

Step 11:

After configuring the gateway, we can see the added applications as shown in the figure below. Click on the created application to add end device.

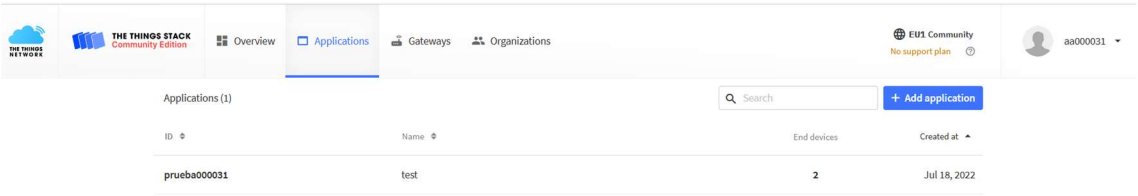


Figure 71. TTN - application list

Step 12:

Enter in to the created application and click on add end device.

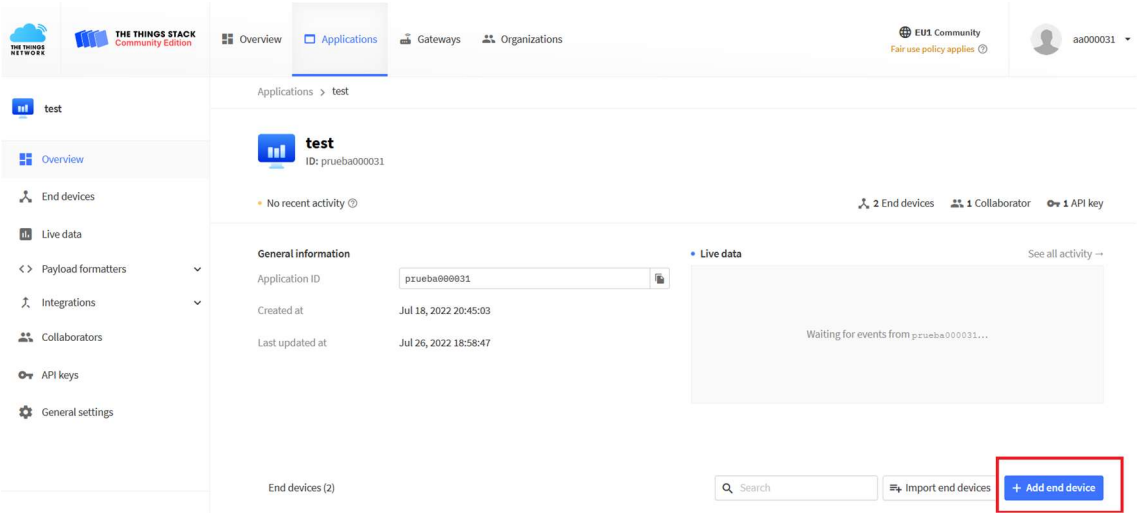


Figure 72. TTN - add end device

Step 13:

Now, we will register the end device **manually**.

- **Frequency Plan**: In the frequency plan, we will select the desired plan. The frequency plan should be same as the frequency of the end node and the frequency plan configured in LoRa Gateway User Interface (refer figure 20). In this project frequency plan selected is EU_863_870_TTN.
- **LoRaWAN Version**: Enter LoRaWAN version of end device provided by manufacturer.
- **Activation Mode**: The activation mode should be selected as ABP.
- **DevEUI**: Click on generate in order to generate DevEUI.
- **Device Address**: It is a non-unique identifier assigned by network server during end device activation.
- **AppSKey**: Click on generate to get the AppSKey, it is an encrypted key used to secure the messages which carry a payload.
- **NwkSKey**: Click on generate to get the NwkSKey, it is an encrypted key used to secure the messages which do not carry a payload.

- **End Device ID**: Unique identifier of the end device, this value is automatically prefilled using the DevEUI.

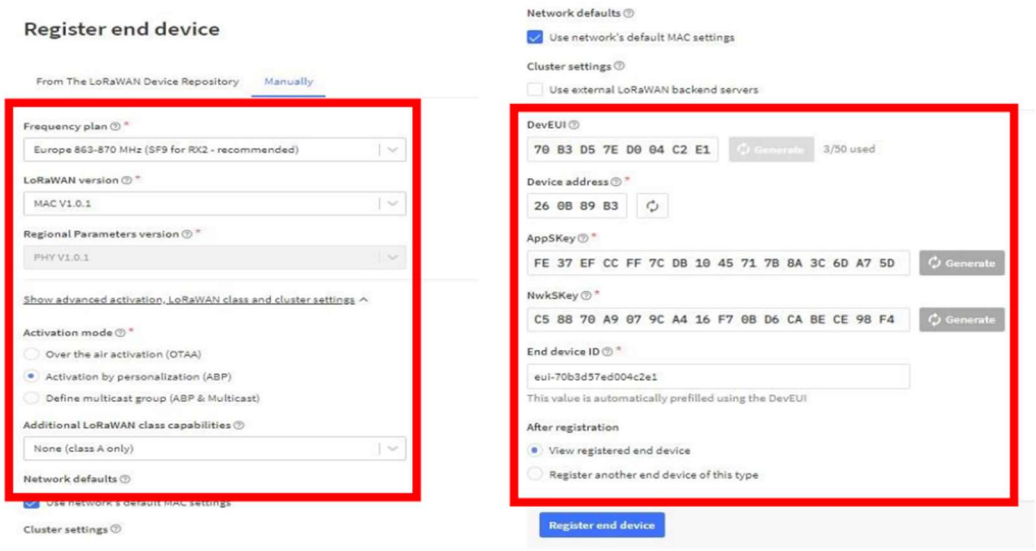


Figure 73. TTN - end device registration

Step 14:

Registered end device overview.

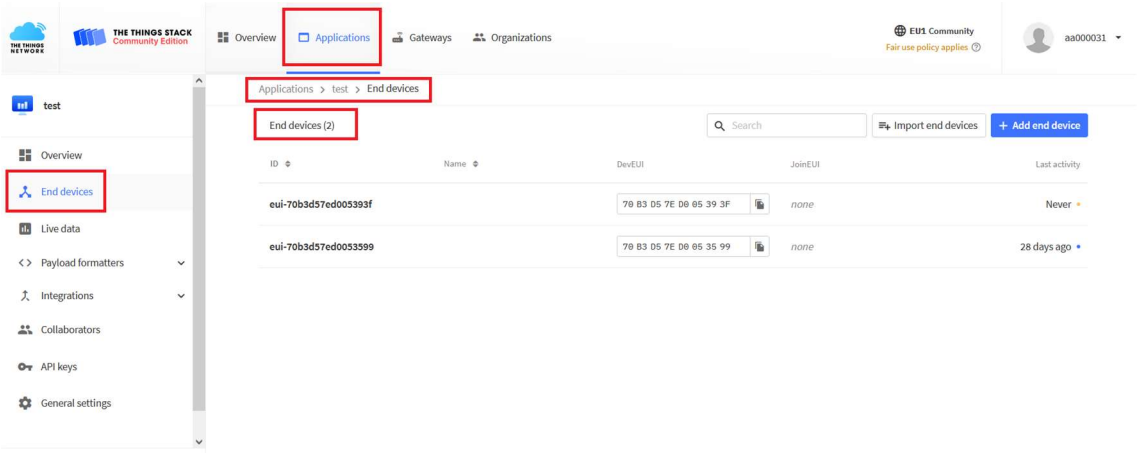


Figure 74. TTN - end device overview

Payload formatters

Payload formatters allow us to process data going to and from end devices. This is useful for converting binary payloads to human readable fields, or for doing any other kind of data conversion on uplinks and downlinks.

In this section we will explain how to create a payload formatter for a specific end device. To do so, we should follow the steps bellow.

- **Step 1**:

To create a device specific payload formatter, navigate to the Applications tab. Choose one application.

- **Step 2**:

Within the Application Overview, select End Devices in the left menu.

- **Step 3**:

Choose your End Device.

- **Step 4**:

Within the End Device Overview, select the Payload Formatters tab in the top menu.

- **Step 5**:

Choose Uplink.

- **Step 6**:

Choose a Formatter type. In our case, custom JavaScript formatter.

- **Step 7**:

Write the formatter code (see annexed files).

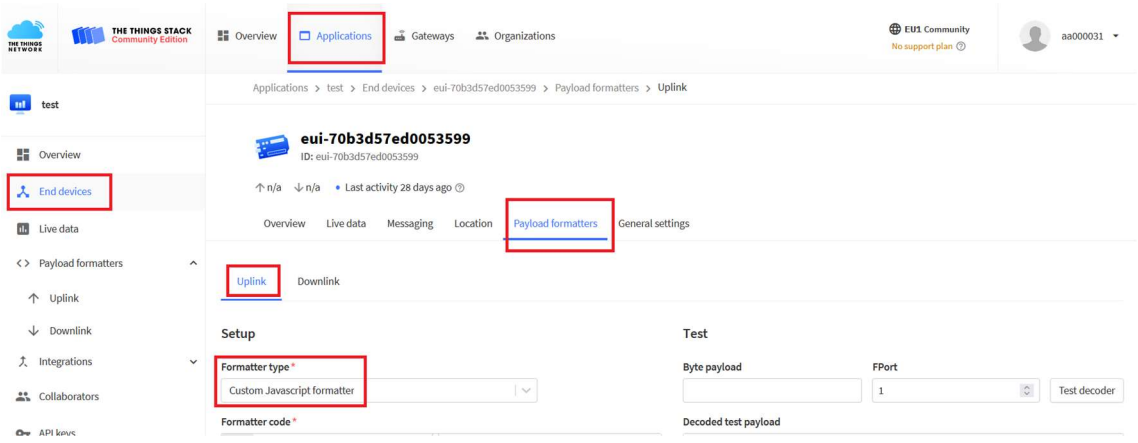


Figure 75. TTN - payload formatters

Webhooks

The Webhooks feature allows The Things Stack to send application related messages to a specific HTTP(S) endpoints. Webhooks can be used to build integrations between The Things Stack and any third party service.

In this master thesis, The Things Network will send a webhook every time an uplink message is received, which will be visualized in an external server (maparuido).

Webhooks access an HTTP(S) endpoint and pass relevant data as JSON. The third party service must expose this endpoint and listen for data, or post messages to The Things Stack. For uplinks, this is encrypted or unencrypted application data, metadata about gateways and signal strength, timestamps, etc. For downlinks, The Things Stack expects an application payload. (write with your own words, and explain how it works)

A user can choose from wide range of third party Webhooks for example Datacake, ThingSpeak etc. also there is a custom webhook which helps to build custom application (integrated in our project).

- **Creating Webhooks**

In this subsection, it will be explained how to create a webhook from the TTN web platform.

Creating a webhook requires us to have an HTTP(S) endpoint available, in our case, it will be maparuido server. In order to integrate a **custom webhook**, our LoRa Gateway must be connected with TTN.

In the application tab, select the Webhooks submenu from the Integrations side menu. Clicking on the "+ Add Webhook" button, a list of Webhook templates will be opened.

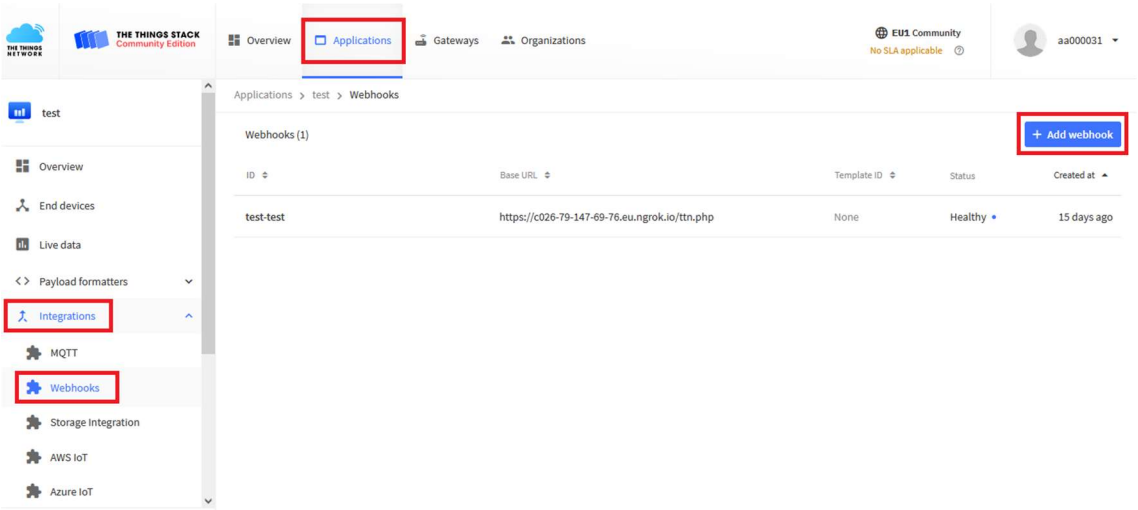


Figure 76. TTN webhook

The next step is selecting "Custom webhook" template. This will open a new window. Now, just we have to enter the Webhook ID, which can be any unique alphanumeric character and to choose JSON as default webhook format. The Base URL should be our HTTP/HTTPS end point

URL, and finally, we have to check the “Uplink message” box to trigger the webhook once any uplink message is received in the TTN platform.

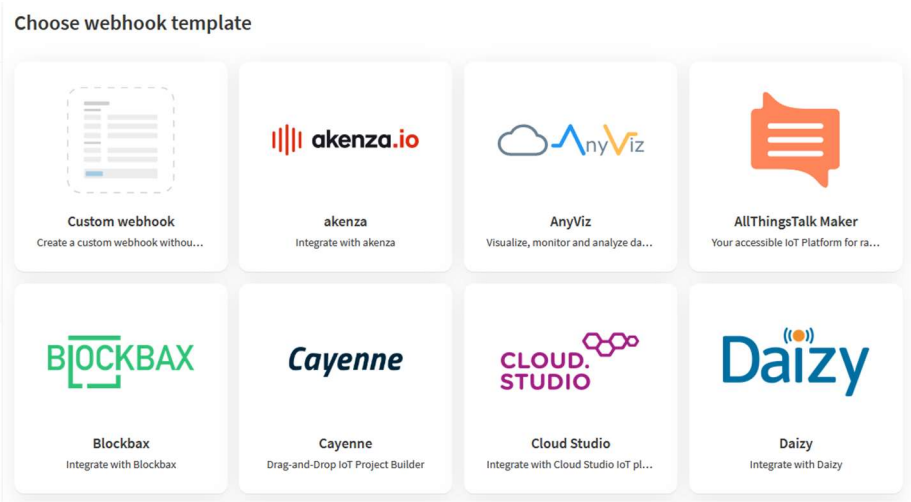


Figure 77. TTN - webhook templates

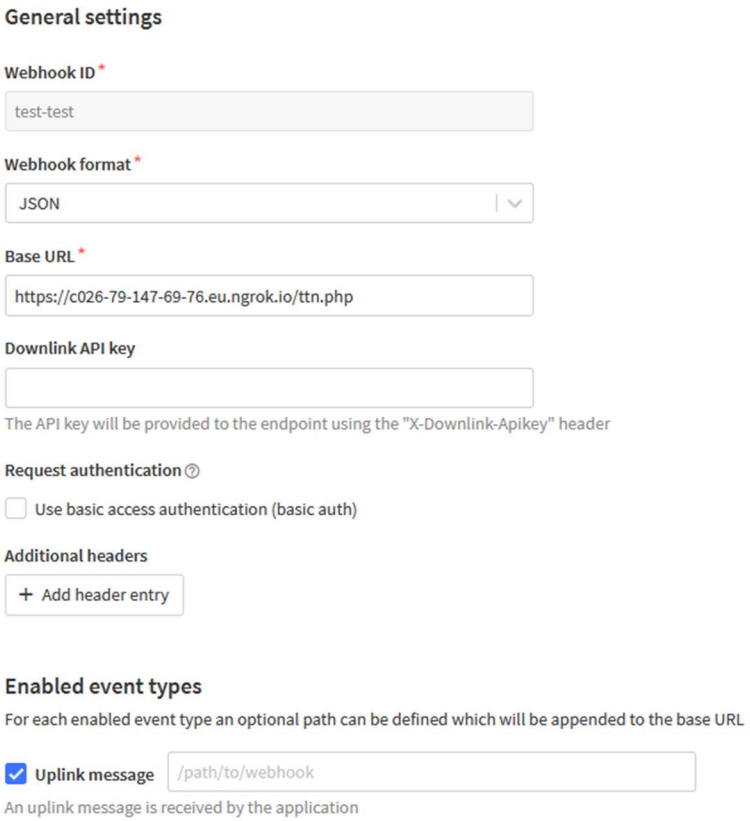


Figure 78. TTN - webhook info

After integrating the webhook, whenever a message is received in TTN server, the webhook will be triggered, and the data will be sent to the web server in JSON format.

After that, the script ttn.php will parse the data and save it inside a database, in order to be visualized to the end user.

5.2.2.2 Web server configuration

In order to process the data coming from the Things Stack, we have developed a new script in php code (ttn.php). This file could be found annexed along with the report of this master thesis.

As we have mentioned before, for data storage and visualization, we will integrate the new script inside the same web server used in the Sigfox part.

6. Noise indicator

In this section, we are going to explain the noise indicator designed and implemented by the Telematics System Engineering research group of the University of Jaen (TIC-220) for resource-constrained devices, as well, we are going to introduce to the calculation of a noise indicator that help us to get a more accurate noise maps, by adding the evaluation of low frequency components.

6.1 Software Implemented in the Sensor Nodes for Noise Monitoring

For the monitoring of real-time acoustic noise and to integrate the calculated "L_{AeqT}" parameter into the sensor nodes, we will use an algorithm designed and implemented by **TIC-220 group** and run it on sensor nodes as a library.

This algorithm is an improvement of a previous one [31] designed and implemented by the same group (TIC-220). The new improvement was the introduction of a new module to determine the frequencies with a higher energy and their degree of importance with respect to background noise or less significant frequencies. In this manner, we obtain the information from the predominant frequency of the noise. The optimized architecture used by the algorithm consists of four functional blocks, as shown in figure 79 [32]:

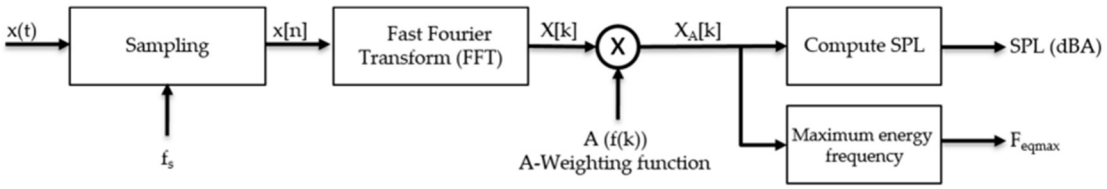


Figure 79. The algorithm's functional blocks. SPL—sound pressure level [32]

The **sampling block** is responsible for sampling the acoustic signal x(t). The IEC 60651 Type-2 SLM acoustic standard [33], superseded by IEC 61672 [33], established the measurement of environmental noise between 0–8 kHz. As is well known, in urban areas, the acoustic signal energy is concentrated in a low-frequency region (<10 kHz). Based on this, we configured the Arduino DUE with a frequency sampling rate, fs, of 33 kHz, by a software function with a resolution of 12 bits. Thus, it is not possible to measure frequencies higher than 16.5 kHz. Related to the time-constant of the integration or time capture, there are two time-weightings that have been internationally standardized, namely: slow response (S) of one second; and fast response (F) of 125ms.

The second block receives the audio samples from the first block. The algorithm is based on a frequency analysis, which uses the discrete Fourier transform (DFT) to determine the frequency spectrum of a segment of audio samples. Let us denote X[k] as the DFT of a windowed signal x[n], at the digital frequencies 2πk/N radians, where N denotes the number of samples and k = 0, . . ., N − 1. To determine the samples of the DFT, we use the following equation:

$$f(k) = \frac{fs.k}{N} \quad (1)$$

The difference between two consecutive samples is given by the following expression:

$$\Delta f = \frac{fs}{N} \quad (2)$$

Regarding the CPU and memory requirements, this block is the most demanding. For a computationally efficient implementation, we used the **fast Fourier transform** (FFT) to evaluate the DFT. Some analyzers have been designed to determine the optimal FFT length and thus achieve efficient implementation. Additionally, an exhaustive analysis has been performed using software functions to the reduce memory requirements and the execution time. The FFT length depends on the frequency-sampling rate chosen and the time capture (Tw), as follows:

$$N = fs * Tw \quad (3)$$

The next table shows the FFT length for the two time-weightings that have been standardized and for the frequency-sampling rate of 33 kHz.

Frequency Sampling Rate	Type of Response	Time Capture	Sample Length
33 kHz	Fast	125ms	4125
33 kHz	Slow	1s	33000

Taking into account that the FFT is more runtime efficient if a base-two sample length is selected, to reduce the execution time, we propose a length of 4096 samples (instead of 4125) for the fast response and 32,768 samples (instead of 33,000) for the slow response. This means a slight decrease in the time window at 124.1ms for the fast response, and at 0.993 s for the slow response. Our proposal is to perform an FFT calculation for the fast response only, which uses a time window of 124.1ms. This approximation of the time window produces an error of 0.7% (29 samples less than using a time window of 125ms), which could be considered as negligible. Larger FFT sizes provide a higher spectral resolution, but take more resources to compute. A smaller window size means a shorter runtime, and, therefore less resource consumption.

Once the frequency components of the acoustic signal are available, an **A-weighted filtering** is implemented. This filtering stage allows for weighting of the different frequency components of the acoustic signal, consistent with a typical human ear response. The mathematical function used to obtain the value of the attenuation depends on the frequency, and is given by the normative IEC 61672 [33], as follows:

$$A(f) = 10log_{10}\left[\frac{1.562339^2.f^4}{(f^2+107.65265^2)(f^2+737.86223^2)}\right] + 10log_{10}[\frac{2.242881.10^{16}.f^4}{(f^2+20.598997^2)(f^2+12194.22^2)}] \quad (4)$$

In the above equation, f is the frequency and A(f) is the associated attenuation. However, some resource-constrained devices have an anomalous behavior for math operations with large numbers. Therefore, we propose the use of an equivalent expression to reduce the complexity of the mathematical operations [32].

$$A(f) = 2 + 20log_{10}\big(R_A(f)\big) \quad (5)$$

$$R_A(f) = \frac{12200^2.f^4}{(f^2.20.6^2)\sqrt{(f^2+107.7^2)(f^2+737.9^2)}(f^2+12200^2)}$$

Using Equations (1) and (2), we can determine the frequency for each sample of the FFT. After applying the filter, the obtained signal is as follows:

$$X_A[k] = \text{A } (\Delta\text{f. k}).\ \text{X[k]} \quad (6)$$

Finally, the last block computes the total energy of the weighted frequency components to obtain the sound pressure levels (SPLs) in dBA. Using the Parseval's relation [34], the total energy of the waveform can be summed across all of its frequency components, as follows:

$$\in_x = \frac{1}{N}\sum_{k=0}^{N-1}|X[k]|^2 \quad (7)$$

Regarding the properties of the FFT, the samples have symmetry because they are complex conjugates, as follows:

$$X\left[\frac{N}{2}+k\right] = X^*\left[\frac{N}{2}-k\right] \quad 1 \le k \le \left(\frac{N}{2}\right)-1 \quad (8)$$

The input samples in the time domain are real values. In the frequency domain, they are symmetrical from the sample N/2 (N is even). Therefore, we can calculate the energy of the signal using only the first N/2 + 1 samples (filtered spectrum with A-weighting filter). The expression is the following:

$$\epsilon_x = \frac{N}{2} \sum_{k=1}^{\left(\frac{N}{2}\right)-1} |X_A[k]|^2 + |X_A[0]|^2 + |X_A[\frac{N}{2}]|^2 \qquad (9)$$

Equation (9) is used to determine the total energy of the signal in the time capture. Taking Tw seconds, the instantaneous average power of the signal is given by the following:

$$P_x = \frac{\epsilon_x}{T_w} \qquad (10)$$

To obtain the SPL in dba, we applied the following expression:

$$SPL\ (dBA) = 10.log_{10}(P_x) + C \qquad (11)$$

where C is the calibration constant, which will be calculated in the calibration process.

6.2 Evaluation of low frequency components.

6.2.1 Frequency weighting

Frequency weighting are used if we are going to talk about sound pressure level. This weighting is necessary because the sensitivity of the human ear to sound varies as a function of frequency. The IEC 61672-1 standard [33] defines A, C, and Z frequency weightings. Other frequency weightings are occasionally used in specialized applications.

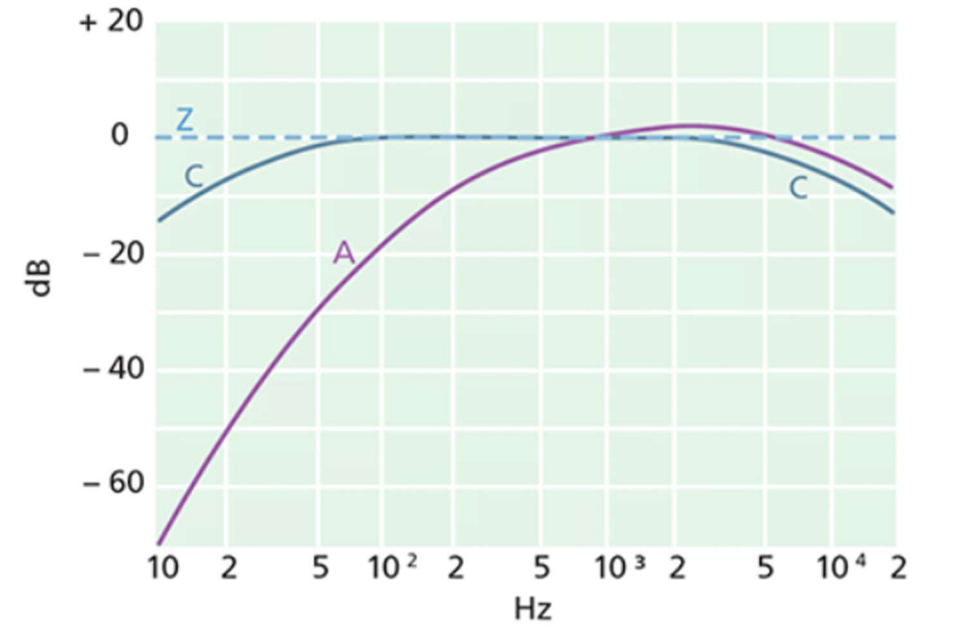


Figure 80. Frequency weighting

6.2.1.1 A-weighting – dba/dB(A)

A-weighting adjusts the signal to most closely match the response of the human ear at average sound levels. It is based on the 40 dB equal loudness curve. Noise parameter symbols typically include the letter "A" (for example, LAeq) to indicate that the measurement includes frequency weighting.

The A-weighting is used in almost all measures of ambient noise and noise in the workplace, and is specified by various national and international standards and guidelines. A-weighted filters cover a spectrum from 10 Hz to 20 kHz; that is, the entire auditory spectrum that the human being perceives.

6.2.1.2 C-weighting – dbc/dB(C)

The response of the human ear varies depending on the sound level. C-weighting corresponds to a 100 dB equal loudness curve and roughly matches the response of the human ear to relatively high sound levels. It is mainly used to evaluate peak values at high sound pressure levels. It can also be used, for example, for noise measurements at shows, where transmission of bass noise can be a problem.

6.2.2 Presence of low frequency components according to RD 1367/2007

The methodology described in Annex IV starts from the simultaneous measurement of noise levels expressed in dB(A) and dB(C). In general, the greater the difference between both indicators, greater proportion of low frequency with respect to medium and high frequencies and therefore greater perception of bass frequencies. The method set out in Annex IV determines Lf, as the difference between LeqC - LeqA duly corrected for background noise. When this difference is less than 10 dB, it is considered that there is no perception of low frequency components. When this indicator is between 10 and 15 dB, it is considered that there is a net perception, and when it exceeds 15 dB, it is considered that the perception is strong. This procedure does not take into account the acoustic conditions of the environment or the spectrum of residual noise, which alters the sensory capacity of the ear.

For the detailed evaluation of noise due to the presence of low-frequency components, the following procedure will be taken as a reference:

a) The sound pressure levels with frequency weightings A and C will be measured, preferably simultaneously.

b) The difference between the values obtained, duly corrected for background noise, will be calculated:

$$Lf = LCeq, Ti - LAeq, Ti$$

c) The presence or absence of low-frequency components and the value of the correction parameter Kf are determined by applying the following table:

Table 4. Correction parameter Kf

Lf in dB	Low frequency component Kf in dB
If Lf <= 10	0
If 10 <Lf <=15	3
If Lf >15	6

6.2.3 Presence of low frequency components according to D176/2009

The basic methodology used is the same as that of RD 1367/2007, but it limits the frequency band between 20 Hz and 160 Hz. The method has two parts: in a first step, it is considered that there is no perception of low frequency if the unevenness LeqC – LeqA (20Hz-160Hz) is less than 20 dB. If a higher difference in level is obtained, a second criterion based on the audibility of the sound according to ISO 226:2003 standard is applied to the unweighted sound level. The LB index is determined as the sum of the energy content of the bands that exceed the audible threshold. Finally, it is considered that there is no perception of low frequency if LB is less than 25 dB. Is considered net perception when this indicator is between 25 and 35 dB, and it is considered a strong perception for LB levels above 35 dB.

Both methods start from the energy value of the noise, corrected for background noise, to assess the presence of low-frequency components. These methods are based on the absolute energy value, and do not take into account the background noise level as a reference. Regarding the human sensation, the sensation of annoyance of a noise is governed rather by the difference in levels, that is to say of the immission with respect to the background noise. Therefore, although the spectrum presents a high absolute energy value, it does not mean that its low frequency content is high.

In this master thesis, in order to evaluate the presence of low frequency component we will use the methodology based in the **RD 1367/2007**.

6.3 The proposed noise indicator

The standard [35] defines weights (A(f), C(f)) in units of dB by tables with tolerance limits (to allow for a variety of implementations). In addition, the standard describes weighting functions Rx(f) [35] to compute the weights. The weighting function Rx(f) is applied to the amplitude spectrum (not the intensity spectrum) of the unweighted sound level. Offsets ensure normalization to 0 dB at 1000 Hz.

In order to calculate the Sound Pressure Level SPL using filter of type C, we will use the same procedure used before for SPL with weighting A with some changes, as follows.

$$R_c\,(f) = \frac{12194^2 * f^2}{(f^2 + 20.6^2) * (f^2 + 12194^2)}$$

$$C_{dB} = 0.06 + (20 * log_{10}(R_c))$$

6.3 Results

To demonstrate the effectiveness of the noise indicator when it comes to the existence of low-frequency components, several tests have been carried out.

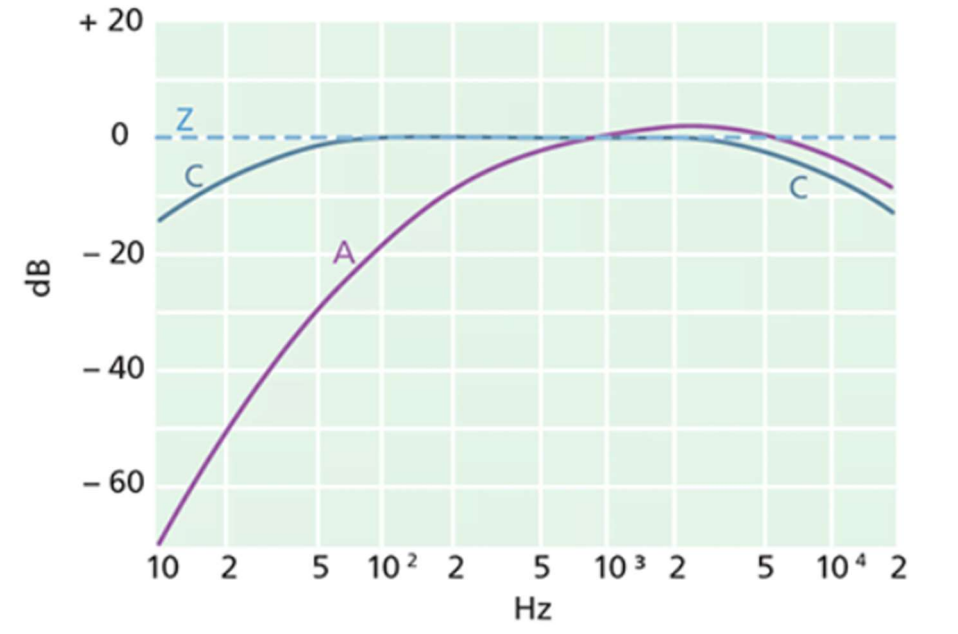
6.3.1 Tones

In this section, we have made a comparative test of the standard noise indicator with the noise indicator with low frequency evaluation. In this test, we have used multiple tones with the same intensity. Firstly, we have used a tone of 1 kHz, later, a composed signal of two tones, 1 kHz and 2 kHz. Afterwards, we have used a 1 kHz tone along with some low frequency tones, 20 Hz, 80 Hz and 120 Hz respectively. The obtained results are shown in Table 5.

Note: The duration of each reading is 5s and the values of both indicators are the average of 12 readings.

Table 5. Noise indicator with low frequency evaluation

Tone frequency	standard noise indicator (dB)	Lf in dB	Kf in dB	Noise indicator with low frequency evaluation (dB)
1 kHz	80.74	-1.12	0	80.74
1 kHz + 2kHz	87.22	-3.01	0	87.22
1 kHz + 20 Hz	73.37	9.63	0	73.37
1 kHz + 80 Hz + 20 Hz	57.14	13.94	3	60.14
1 kHz + 80 Hz + 20 Hz + 120 Hz	67.23	15.64	6	73.23



6.3.2 White noise

In this section, we have made a comparative test, in the same conditions like in the previous section. In this experiment, we have used white noise along with pure tones. Firstly, we have used pure white noise (30 Hz – 16 kHz). Later, we have added low frequency components (pure tones of 100 Hz and 40 Hz respectively) to the white noise. The obtained results are shown in the table 6.

Note: The duration of each reading is 5s and the values of both indicators are the average of 12 readings.

Table 6. Evaluation of low frequency components using white noise

Signal	standard noise indicator (dB)	Lf in dB	Kf in dB	Noise indicator with low frequency evaluation (dB)
White noise	76.62	-3.31	0	76.62
White noise + 100 Hz	61.13	10.71	3	64.13
White noise + 100 Hz + 40 Hz	67.61	14.23	3	70.61

6.3.3 Response to music

In this section, we are going to reproduce two music tracks using the same intensity. The first track will be a person playing Tuba instrument and the second track a person playing piano. As it is known, most of Tuba sounds are placed in low frequencies, nevertheless the Piano sounds cover a more extended range of frequencies. The obtained results are shown in tables 7 and 8.

Note: The duration of each reading is 5s and the values of both indicators are the average of 12 readings.

Table 7. Tuba instrument

fundamental frequency (Hz)	standard noise indicator (dB)	Lf in dB	Kf in dB	Noise indicator with low frequency evaluation (dB)
531.06	58.54	6.33	0	58.54
372.02	61.04	5.02	0	61.04
187.04	56.17	14.00	3	59.17
160.59	49.71	11.42	3	52.71
255.84	69.42	3.20	0	69.42
393.24	54.57	7.59	0	54.57
46.19	46.66	12.18	3	49.66
610.31	48.62	9.03	0	48.62
39.99	42.75	17.11	6	48.75
123.11	46.72	10.94	3	49.72
150.20	43.65	13.88	3	46.65
142.33	48.55	10.53	3	51.55
110.54	43.01	14.11	3	46.01

Table 8. Piano instrument

fundamental frequency (Hz)	standard noise indicator (dB)	Lf in dB	Kf in dB	Noise indicator with low frequency evaluation (dB)
773.24	67.75	1.10	0	67.75
797.45	74.38	0.09	0	74.38
685.10	68.33	0.76	0	68.33
643.72	68.76	1.72	0	68.76
722.75	75.71	-0.35	0	75.71
688.88	73.89	1.13	0	73.89
664.67	75.95	2.12	0	75.95
646.54	71.70	1.31	0	71.70
728.90	76.14	-0.44	0	76.14
707.83	73.43	0.78	0	73.43
690.71	73.41	1.31	0	73.41
719.70	76.03	-0.57	0	76.03
727.79	88.53	0.06	0	88.53

6.3.4 Responses in a street of an urban area

In this section, we have monitored street noise in two different periods of time. The first experiment, was from 12:00 hours to 15:00 hours and the second one was from 00:00 hour to 03:00. The obtained results are shown in table 9.

Note: The duration of each reading is 5s and the values of both indicators are the average of 12 readings.

Table 9. Monitoring street noise from 12:00 – 15:00

fundamental frequency (Hz)	standard noise indicator (dB)	Lf in dB	Kf in dB	Noise indicator with low frequency evaluation (dB)
882.19	73.42	2.84	0	73.42
819.92	75.16	2.20	0	75.16
799.32	73.36	2.24	0	73.36
794.15	74.87	2.68	0	74.87
807.51	70.54	2.54	0	70.54
792.51	68.31	2.39	0	68.31
785.58	68.91	2.77	0	68.91
797.56	72.50	2.61	0	72.50
791.13	75.81	2.65	0	75.81
796.13	75.49	2.72	0	75.49
802.67	70.80	2.43	0	70.80
807.52	73.95	2.63	0	73.95

Table 10. Monitoring street noise from 00:00 - 03:00

fundamental frequency (Hz)	standard noise indicator (dB)	Lf in dB	Kf in dB	Noise indicator with low frequency evaluation (dB)
627.09	74.04	-0.52	0	74.04
612.73	74.43	-0.51	0	74.43
596.37	44.08	16.01	6	50.08
650.57	79.28	-0.19	0	79.28
684.12	73.79	0.67	0	73.79
667.05	92.33	-0.46	0	92.33
633.72	92.01	-0.48	0	92.01
605.04	92.33	-0.28	0	92.33
692.80	93.94	0.47	0	93.94
677.81	92.13	0.87	0	92.13
557.68	52.72	23.57	6	58.72
766.97	63.44	1.23	0	63.44

6.3.5 form of representation on the server

In this section, we are going to show how the results are represented on the server side. As we can notice in the figure 81, we have multiple graphs showing different parameters (frequency, noise, maximum noise, minimum noise, maximum frequency, minimum frequency). In this case, we have represented only the noise indicator along with the frequency. As well, we can filter the results by time intervals.

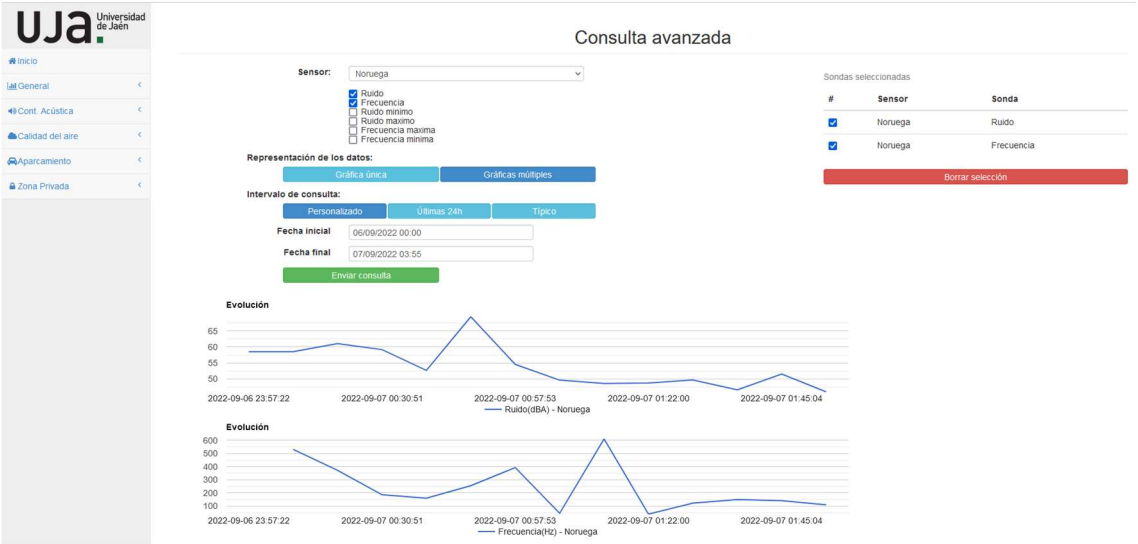


Figure 81. Results obtained playing tuba instrument

7. CONCLUSIONS

During the execution of this master thesis, a series of conclusions have been reached:

- Using Arduino as a sensor network node allows us, on one hand, offering an economic solution, and on the other hand, the possibility of integrating different technologies, such as SIGFOX and LoRaWAN, as well it allows executing the algorithms used to obtain the noise indicator.
- A complete and functional system can be realized using components of low cost, being useful for normal users who cannot afford some of the traditional options presented in the introduction.
- The use of open source hardware allows the access to documentation developed by other users, being able to reuse free code as well as find solutions to problems in different forums.
- The use of the web server developed by the research group of EPSL (TIC-220) has been of great interest due to its similarity of our master thesis.
- A noise indicator has been implemented that allows a more precise measurement of the degree of annoyance caused by the noise perceived by people.

8. FUTURE LINES

We consider that in the development of this master's thesis, the objectives set at the beginning of it have been met, in general terms. However, there are several aspects to take into account and that could be worked on in the future.

- design and deploy a proprietary web server that meets general and specific requirements.
- design and deploy a mobile application that performs the same role as the web application.
- Improved efficiency in the evaluation of low-frequency components.
- Evaluate other aspects when calculating the noise indicator, such as the presence of emerging tonal components and the presence of impulsive components.
- Deploy various nodes on the street in real scenarios.
- Conducting a survey to assess the degree of annoyance caused by low-frequency noise perceived by people.

9. REFERENCES

[1] Noise pollution, web page: https://www.eea.europa.eu/articles/noise-pollution-is-a-major (available on 05/08/2022)

[2] Measuring and Assessment the Noise Level, web page: https://www.researchgate.net/publication/327907263_Measuring_and_Assessment_the_Noise_Level_in_Different_Regions_in_Baghdad_City_And_Compare_it_with_The_Allowable_Levels (available on 10/08/2022)

[3] Drawbacks of SLMs, web page: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0053649 (available on 06/08/2022)

[4] SLMs use, web page: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0053649 (available on 06/08/2022)

[5] 46º CONGRESO ESPAÑOL DE ACÚSTICA ENCUENTRO IBÉRICO DE ACÚSTICA EUROPEAN SYMPOSIUM ON VIRTUAL ACOUSTICS AND AMBISONICS, source: https://www.academia.edu/34295988/46o_CONGRESO_ESPA%C3%91OL_DE_AC%C3%9ASTICA_ENCUENTRO_IB%C3%89RICO_DE_AC%C3%9ASTICA_EUROPEAN_SYMPOSIUM_ON_VIRTUAL_ACOUSTICS_AND_AMBISONICS_RUIDO_DE_IMPACTOS_NUEVAS_HERRAMIENTAS_TE%C3%93RICAS_Y_PR%C3%81CTICAS (available on 06/08/2022)

[6] IoT for Smart cities, source: https://www.researchgate.net/publication/260540297_Internet_of_Things_for_Smart_Cities

(available on 07/08/2022)

[7] Sigfox: https://en.wikipedia.org/wiki/Sigfox (available on 24/08/2022)

[8] An Introduction to Sigfox Technology – Basics, Architecture and Security Features: https://circuitdigest.com/article/what-is-sigfox-basics-architecture-and-security-features (available on 03/08/2022)

[9] Sigfox: Strengths, technological developments and IoT trends: https://enless-wireless.com/en/sigfox-network-technological-developments-iot/ (available on 09/08/2022)

[10] Sigfox. Technical overview: https://www.ismac-nc.net/wp/wp-content/uploads/2017/08/sigfoxtechnicaloverviewjuly2017-170802084218.pdf (available on 01/08/2022)

[11] Sigfox. Message structure: https://www.avnet.com/wps/portal/apac/resources/article/sigfox-is-the-worlds-leading-provider-of-connectivity-for-the-internet-of-things/ (available on 06/08/2022)

[12] Sigfox modulation: https://www.tutorialspoint.com/digital_communication/digital_communication_phase_shift_keying.htm# (available on 06/08/2022)

[13] Research Center: https://www.leti-cea.fr/cea-tech/leti (available on 06/08/2022)

[14] Encriptación AES-ECB: https://infocenter.nordicsemi.com (available on 06/08/2022)

[15] Sigfox coverage: https://www.sigfox.com/en/ (available on 06/08/2022)

[16] LoRa Alliance: https://lora-alliance.org/ (available on 10/08/2022)

[17] LoRa overview: https://www.semtech.com/products/wireless-rf (available on 10/08/2022)

[18] Waspmote LoRaWAN. Networking Guide: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf (available on 11/08/2022)

[19] Libelium. Waspmote-LoRa-868MHz_915MHz-SX1272 — Networking Guide: http://www.libelium.com/downloads/documentation/waspmote_lora_868mhz_915mhz_sx1272_networking_guide.pdf (available on 31/07/2022)

[20] Libelium. http://www.libelium.com

[21] Specification LoRa v1.1: https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf (available on 01/08/2022)

[22] ABP and OTAA: https://medium.com (available on 01/08/2022)

[23] LoRaWAN security: https://www.pinterest.com/pin/467600373810565938/ (available on 01/08/2022)

[24] Announcing The Things Network Stack V3: https://www.youtube.com/watch?v=dvYREtWc1lA (available on 15/08/2022)

[25] Integración de indicadores borrosos en redes de sensores inalámbricos. Aplicación para la monitorización acústica: https://ruja.ujaen.es/handle/10953/947 (available on 29/08/2022)

[26] Trabajo Fin de Grado, Amine Alrharad: https://tauja.ujaen.es/bitstream/10953.1/9718/1/Memoria_TFG.pdf (available on 09/08/2022)

[27] Dragino Single channel gateway LG01-N: https://www.dragino.com/products/lora-lorawan-gateway/item/143-lg01n.html

[28] LoRa Configuration and access: https://www.dragino.com/downloads/downloads/LoRa_Gateway/LG01N/LG01N_LoRa_Gateway_User_Manual_v1.4.0.pdf

[29] LG01-N Limitations: https://wiki.dragino.com/index.php?title=Limitation_of_Single_Channel_Gateway

[30] LoRa shield: https://wiki.dragino.com/index.php?title=Lora_Shield#What_is_the_Dragino_LoRa_Shield

[31] A new algorithm to monitor noise pollution adapted to resource-constrained devices: https://link.springer.com/article/10.1007/s11042-014-2074-3

[32] Wireless Acoustic Sensor Nodes for Noise Monitoring in the City of Linares (Jaén): https://www.mdpi.com/1424-8220/20/1/124

[33] The IEC 60651 Type-2 SLM acoustic standard. https://iec.ch/homepage

[34] Parseval's relation: https://www.tutorialspoint.com/parseval-s-theorem-and-parseval-s-identity-of-fourier-transform

[35] Frequency weighting: https://hmong.es/wiki/A-weighted