



**Escuela de  
Ingeniería y Arquitectura  
Universidad Zaragoza**

# Smart Street: “Proof of concept” Data processing

Author:  
Laura Guijarro Iguacel

Supervisor:  
Radek Fujdiak

Bachelor's Thesis – Bachelor's degree in Telecommunications Technology and Services Engineering

Brno, summer semester 2018

## Abstract

The Internet of Things (IoT) is forever changing the way people think, how they react, and what they want at their fingertips. IoT technology dives deeper into inter-connected data and sensing habits of consumers, inanimate objects, humans, and even animals with the ability to transmit data without human-to-computer or human-to-human interaction. The purpose of all this communicating between the two is to provide consumers smarter products and services and a better customer experience.

In this bachelor's thesis a Smart Street is designed and implemented in a Proof-of-Concept. Making the street smart allows monitoring in real time measured parameters that could be interesting such as air quality or temperature and have control of different things like streetlights and car park. Due to this control the consumption of resources can be controlled.

For this, a comparison of all the suitable technologies has been done, concluding with LoRaWAN technology as the network communication protocol, whose analysis has been carried out. A final application is elaborated to show to the end user the state of the street in real time.

# Contents

Abstract .....	ii
Contents .....	iii
List of figures .....	v
List of tables .....	vi
1. Introduction .....	1
2. Comparison of the communication technologies .....	2
2.1. Bluetooth.....	2
2.2. ZigBee .....	3
2.3. Wi-Fi HaLow .....	3
2.4. SigFox .....	4
2.5. LoRaWAN .....	4
2.6. Technology selected.....	5
3. The Things Network (TTN).....	8
3.1. Limitations.....	9
4. Design.....	11
4.1. Overview .....	11
4.2. Integration.....	12
4.3. Requirement specifications.....	13
4.3.1. Gateway .....	13
4.3.2. End device .....	13
4.3.3. Application .....	14
4.4. Hardware selected .....	14
4.4.1. Gateway .....	14
4.4.2. End device .....	16
4.5. Cost estimation .....	17
4.5.1. Gateway .....	17
4.5.2. End device .....	18
4.5.3. Total budget .....	18
5. LoRaWAN Overview .....	19
5.1. Lora.....	19
5.1.1. LoRa frequency bands .....	19
5.1.2. LoRa Modulation .....	19
5.2. LoRaWAN .....	23
5.2.1. LoRaWAN topology .....	24

5.2.2.	LoRaWAN Classes .....	25
5.2.3.	Regional Parameters .....	26
5.2.4.	Security.....	26
6.	“The Things Network” account .....	28
6.1.	Creating an application in TTN.....	28
6.2.	Adding a gateway into TTN. ....	29
7.	Setting up the gateway .....	31
7.1.	Hardware.....	31
7.2.	Software .....	31
7.3.	Registration into TTN .....	32
8.	Showing the data in TTN .....	34
8.1.	Gateway section.....	34
8.2.	Application section.....	34
9.	User application .....	37
9.1.	User Interface.....	39
9.1.1.	Home page .....	39
9.1.2.	Temp & Hum page.....	40
9.1.3.	Parking page .....	40
9.1.4.	Air & sound quality page.....	41
9.1.5.	Streetlights page .....	41
10.	Conclusions .....	43
10.1.	LoRa .....	43
10.2.	LoRaWAN.....	43
10.3.	The Things Network .....	43
10.4.	Gateway .....	43
10.5.	End-device .....	43
10.6.	Application .....	43
11.	References.....	44

## List of figures

Figure 1: TTN Network scheme .....	8
Figure 2: Coverage of TTN .....	8
Figure 3: Design of the project. ....	11
Figure 4: Scheme of the project. ....	12
Figure 5: Antenna and pigtail for IC880A concentrator. ....	15
Figure 6: Dragino LoRa shield pin-map. ....	16
Figure 7: Modulation/ Spreading Process [2].....	20
Figure 8: Demodulation /De-spreading Process [2].....	20
Figure 9: Chip signal in the time domain.....	21
Figure 10: Structure of the data.....	23
Figure 11: LoRa Modem Packet formatting .....	23
Figure 12: LoRaWAN topology .....	24
Figure 13: LoRaWAN classes .....	25
Figure 14: Overview of "console" section [1].....	28
Figure 15: "Application" section overview [4]. ....	29
Figure 16:" Gateways" section overview. ....	30
Figure 17: Description of the Gateway .....	32
Figure 18: Location of the Gateway and placement of the antenna .....	33
Figure 19: Gateway overview.....	33
Figure 20: Gateway data traffic.....	34
Figure 21: Payload format example. ....	35
Figure 22: APIs available in TTN to export data. [1].....	37
Figure 23: User interface of swagger data storage API.....	38
Figure 24: Root folder of website server in Plesk web host edition. ....	39
Figure 25: Preview of final application's home page [8]. ....	39
Figure 26: Preview of final application's Temp & Hum page [8]. ....	40
Figure 27: Preview of final application's Car park page [8]. ....	41
Figure 31: Preview of final application's Air & sound quality page [8]. ....	41
Figure 29: Preview of final application's Streetlights page [8]. ....	42

## List of tables

Table 1: Comparative of Bluetooth LE and BR/EDR. ....	3
Table 2: Overview of LPWAN technologies.....	5
Table 3: Costs of Sigfox and LoRa.....	6
Table 4: Summary of sensors used to measure real world parameters. ....	17
Table 5: Gateway hardware cost.....	17
Table 6: Bare end device hardware cost. ....	18
Table 7: Cost of the sensor and actuators used in the Smart street project. ....	18
Table 8: Total cost of the hardware in Smart street project.....	18
Table 9: EU863-870 Data rate [3].....	26
Table 10: Connections between IC880A and RPi. ....	31
<i>Table 11: Description of the payload sent.</i> .....	36

## 1. Introduction

Nowadays, the number of devices connected to the network is growing exponentially; this is where interest in Internet of Things (IoT) comes from. More and more devices are connected so that their data can be acquired in real time from a remote place; there are thousands of applications that can be done with this technology.

These applications are based on the information captured by sensors and the storage of this data. It enables to organize and structure this information to improve the safety and quality of life, reducing the waste of money and time.

In this work, a Proof-of-Concept of a Smart Street is going to be constructed to benefit from all the advantages of IoT and analyze up to what extent are useful. The project is based on two complementary reports, one focused on data acquisition which works on the sensors used to measure the real world parameters and converts them into data and, this one, which is based on how that data transmission is created by using LoRaWAN.

In the following chapter (chapter 2) a state of art is done where all the communication technologies are compared and analyzes which existing technology is the most suitable for the project. Chapter 3 introduces The Things Network, the startup company which will provide the data server to the project; all the LoRa packages will be collected there and using their APIs the data will be exported to create the final application. In chapter 4 the design of the project is introduced; it shows a general idea of what would be the physical intention of the project, construct a real smart street that collects all the data considered in the project; it is also included how the integration will be done analyzing all the parts that a LoRa application represents; the requirements of the project are explained and depending on them the hardware has been chosen and presented in this chapter. Chapter 5 explains the difference between LoRa and LoRaWAN introducing the protocols and technologies chosen in the project. Chapter 6 explains how to sign up in TTN, here the site’s user interface is presented and the steps follow to connect and register the gateway are explained. Chapter 7 explains how the gateway has been setting up. Chapter 8 is focused on showing the data between the end device and the gateway; it is centralized in introducing how data of each one is shown in TTN and how is organized the data string sent by the end device to the gateway, which is a key parameter to take into account when decrypting the data. Chapter 9 explains the process of creating a web application for the final user of the project where the whole data of the sensors is shown; it includes basic concepts of coding a website and explains the content of each page in the site. Finally, the report is finished with the conclusions made from the project.

## 2. Comparison of the communication technologies

In this chapter, the possible communication technologies for the project are introduced and compared. To establish a connection between the end devices and the gateway it is necessary to design a wireless communication between both of them. To do so, here are explained different communication models suitable for IoT. In the end the most suitable communication models are deeply compared and taking into account this comparison; finally, one of them has been chosen as the communication model for the design.

### 2.1. Bluetooth

Bluetooth is one of the most extended technology for low range data transmission and it has been very important in the consumer electronics field. It seems to be very useful for developing wearable devices, as it allows connection between the smart device and the consumer's smartphone.

Operating in the 2.4GHz unlicensed industrial, scientific and medical (ISM) frequency band, Bluetooth technology supports multiple radio options that enable developers to build products meeting the unique connectivity requirements of their market.

Whether a product streams high-quality audio between a smartphone and speaker, transfers data between a tablet and medical device, or sends messages between thousands of nodes in a building automation solution, the Bluetooth Low Energy (LE) and Basic Rate/Enhanced Data Rate (BR/EDR) radios are designed to meet the unique needs of developers worldwide.

The new low power Bluetooth, known as Bluetooth LE or Bluetooth Smart, is a protocol to take into account to develop IoT applications. It is characterized by offering a similar range as normal Bluetooth technology but with a much lower energy consuming. However, it should be taken into account that Bluetooth LE is not suitable for transferring files and it is more appropriate to send pieces of data (chunks).

The Bluetooth Low Energy (LE) radio is designed for very low power operation. To enable reliable operation in the 2.4 GHz frequency band, it leverages a robust frequency-hopping spread spectrum approach that transmits data over 40 channels. The Bluetooth LE radio provides developers a tremendous amount of flexibility, including multiple PHY options that support data rates from 125 Kb/s to 2 Mb/s, multiple power levels, from 1mW to 100 mW, as well as multiple security options up to government grade.

The Bluetooth BR/EDR radio is designed for low power operation and also leverages a robust Adaptive Frequency Hopping approach, transmitting data over 79 channels. The Bluetooth BR/EDR radio includes multiple PHY options that support data rates from 1 Mb/s to 3 Mb/s, and supports multiple power levels, from 1mW to 100 mW, as well as multiple security options. It supports a point-to-point network topology that is optimized for audio streaming.

Table 1 resumes all the parameters mentioned in the previous paragraphs.



Table 1: Comparative of Bluetooth LE and BR/EDR.

	<b>Bluetooth Low Energy (LE)</b>	<b>Bluetooth Basic Rate/ Enhanced Data Rate (BR/EDR)</b>
<b>Optimized for...</b>	Short burst data transmission	Continuous data streaming
<b>Frequency band</b>	2.4GHz ISM Band (2.402 – 2.480GHz utilized)	2.4GHz ISM Band (2.402 – 2.480GHz GHz utilized)
<b>Channels</b>	40 channels with 2MHz spacing (3 advertising channels/37 data channels)	79 channels with 1 MHz spacing
<b>Channel usage</b>	Frequency-Hopping spread spectrum (FHSS)	Frequency-Hopping Spread Spectrum (FHSS)
<b>Modulation</b>	GFSK	GFSK, $\pi/4$ DQPSK, 8DPSK
<b>Power consumption</b>	0.01x to 0.5x of reference (depending on use case)	1 (reference value)
<b>Data Rate</b>	LE 2M PHY: 2Mb/s LE 1M PHY: 1Mb/s LE Coded PHY (S=2) : 500Kb/s LE Coded PHY (S=8): 125Kb/s	EDR PHY (8DPSK): 3Mb/s EDR PHY ( $\pi/4$ DQPSK): 2Mb/s BR PHY (GFSK): 1Mb/s
<b>Max Tx Power</b>	Class 1: 100 mW (+20 dBm) Class 1.5: 10 mw (+10 dBm) Class 2: 2.5 mW (+4 dBm) Class 3: 1 mW (0dBm)	Class 1: 100 mW (+20 dBm) Class 2: 2.5 mW (+4 dBm) Class 3: 1 mW (0dBm)
<b>Network topologies</b>	Point-to-point (including piconet) Broadcast Mesh	Point-to-point (including piconet)

Communication range: 50-150m (Smart/LE).

Transfer velocity: 1Mbps (Smart/LE).

## 2.2. ZigBee

ZigBee is a wireless technology oriented to house automation, medical device data collection, and other low-power low-bandwidth needs and industrial applications. ZigBee PRO and ZigBee Remote Control (RF4CE) are based on protocol IEEE 802.15.4, a wireless technology to create personal area networks that works at 2.4GHz in applications which require a low data transmission rates and low-power digital radios, within an area of 100 meters, such as houses or buildings.

ZigBee/RF4CE has some significant advantages like low power consume in complex systems, top range security, robustness, high scalability and capacity to bear a big amount of nodes. Therefore, it is a well ranked technology for the wireless applications for IoT and M2M.

Communication range: 10-100m.

Transfer velocity: 250Kbps.

## 2.3. Wi-Fi HaLow

Wi-Fi HaLow refers to Low power, long range Wi-Fi. With industry momentum mounting around a low power Wi-Fi® solution, Wi-Fi Alliance® has introduced Wi-Fi HaLow™ as the designation for products incorporating IEEE 802.11ah technology. Wi-Fi HaLow operates in

frequency bands below one gigahertz, offering longer range, lower power connectivity to Wi-Fi CERTIFIED™ products. Wi-Fi HaLow will enable a variety of new power-efficient use cases in the Smart Home, connected car, and digital healthcare, as well as industrial, retail, agriculture, and Smart City environments.

Wi-Fi HaLow extends Wi-Fi into the 900 MHz band, enabling the low power connectivity necessary for applications including sensor and wearables. Wi-Fi HaLow’s range is nearly twice that of today’s Wi-Fi, and will not only be capable of transmitting signals further, but also providing a more robust connection in challenging environments where the ability to more easily penetrate walls or other barriers is an important consideration. Wi-Fi HaLow will broadly adopt Wi-Fi protocols and deliver many of the benefits that consumers have come to expect from Wi-Fi today, including multi-vendor interoperability, strong government-grade security, and easy setup.

Communication range: More than 60m.

Frequency band: 900 MHz.

Maximum speed: Up to 347Mbps.

#### 2.4. SigFox

An alternative of large range is SigFox, which in terms of range is between Wi-Fi and mobile communication. It uses ISM bands that can be used without the need of acquiring licenses.

SigFox responds to the needs of a lot of M2M applications, those that works with small battery and only require lower levels of data transfer, in scenarios where Wi-Fi range is too short and mobile communication is too expensive and consume a lot of energy.

SigFox uses a technology called Ultra Narrow Band (UNB) designed to work in low transfer velocities from 10 to 1000 bits per second. Hence, long distances can be achieved while being very robust against the noise.

SigFox has tailored a lightweight protocol to handle small messages. Less data to send means less energy consumption, hence longer battery life. It only consumes 50 microwatts (the mobile communication consumes 5000 microwatts) also It can stay in stand-by 20 years with a battery of 2.5Ah (0.2 years for mobile communications). They have already implemented a lot of end-devices and the network is being installed in the main cities of Europe.

This technology is robust, low connectivity fee, high network capacity, energetically efficient and it works as a scalable network where it can communicate hundreds of end-devices in many kilometers. For that, is suitable for applications M2M like; smart counters, medical monitors, security devices, street lighting and environmental sensors.

The system uses wireless transceivers that works in the band sub-1GHz offering an exceptional efficiency, long range and minimal consume.

Communication range: 30-50Km (rural environments), 3-10Km (urban environments).

Transfer velocity: 10-1000bps.

#### 2.5. LoRaWAN

This technology is similar to Sigfox, the Table 2 gives an overview of both technologies. LoRaWAN is designed to implement Wide Area Networks (WAN) with specific characteristics to

put up with mobile communications, bidirectional, economical and safe for IoT, M2M, smart cities and industrial applications.

Table 2: Overview of LPWAN technologies.

	<b>Sigfox</b>	<b>LoRaWAN</b>
<b>Modulation</b>	PBSK	CSS
<b>Frequency</b>	Unlicensed ISM bands (868MHz in Europe, 915MHz in North America and 433MHz in Asia)	Unlicensed ISM bands (868MHz in Europe, 915MHz in North America and 433MHz in Asia)
<b>Bandwidth</b>	100 Hz	250 KHz and 125 KHz
<b>Maximum data rate</b>	100 bps	50 kbps
<b>Bidirectional</b>	Limited / Half-duplex	Yes / Half-duplex
<b>Maximum messages/day</b>	140 (UL), 4 (DL)	Unlimited
<b>Maximum payload length</b>	12 bytes (UL), 8 bytes (DL)	246 bytes
<b>Range</b>	10 km (urban), 40 km (rural)	5 km (urban), 20 km (rural)
<b>Interference immunity</b>	Very high	Very high
<b>Authentication &amp; encryption</b>	Not supported	Yes (AES 128b)
<b>Adaptive data rate</b>	No	Yes
<b>Handover</b>	End-devices do not join a single base station	End-devices do not join a single base station
<b>Localization</b>	Yes (RSSI)	Yes (TDOA)
<b>Allow private network</b>	No	Yes
<b>Standardization</b>	Sigfox company is collaborating with ETSI on the standardization of Sigfox-based networks	LoRa-Alliance

LoRaWAN technology is optimized for low power energy using and offers wide networks with millions and millions of devices.

Communication range: 2-5km (city), 15km (countryside).

Transfer velocity: 0.3-50kbps.

## 2.6. Technology selected

Many factors should be considered when choosing the appropriate technology for an IoT application including quality of service, battery life, latency, scalability, payload length, coverage, range, deployment and cost.

For the project, although the representation of the Smart Street will only have 2-3 meters, the chosen technology has to cover a real Smart Street so it has to cover about 1-2 Km. Therefore, a Wide Area Networks (WAN) is needed. That leads into two possible options SigFox or LoRaWAN networks.

Temperature, humidity, movement, lightening, air quality and noise quality sensors of the Smart Street alert property managers to prevent damages and instantly respond to requests without having a manual street monitor. These sensors require low cost and long battery lifetime. They do not require quality of service, therefore Sigfox and LoRa are suitable for this application.

In the following points, both technologies are deeply analyzed and compared.

- Quality of service

Sigfox and LoRaWAN employ unlicensed spectra and asynchronous communication protocols. They can bounce interference, multipath, and fading. However, they cannot offer the same QoS provided by other technologies that use licensed spectrum and synchronous protocol like NB-IoT but, as mentioned before, quality of service guarantee is not required.

- Latency

Unlike Sigfox, LoRaWAN provides class C to also handle low-bidirectional latency at the expense of increased energy consumption.

- Deployment model

LoRaWAN has the advantage that allows it to be currently deployed in 42 countries versus 31 countries for Sigfox. Nevertheless, the world wide deployments of LoRa and Sigfox are still under rollout.

In addition, one significant advantage of LoRaWAN ecosystem is its flexibility. Unlike Sigfox, LoRaWAN offers local network deployment, i.e., LAN using LoRa gateway as well as public network operation via base stations.

- Cost

Various cost aspects need to be considered such as spectrum cost (license), network/deployment cost, and device cost Table 3 shows the cost of Sigfox and LoRaWAN.

*Table 3: Costs of Sigfox and LoRa*

	<b>Spectrum cost</b>	<b>Deployment cost</b>	<b>End-device cost</b>
<b>Sigfox</b>	Free	>4000€/base station	<2€
<b>LoRa</b>	Free	>100€/gateway >1000€/base station	3-5€

- Authentication and encryption

LoRaWAN permits to create a private network with encryption based in AES 128b. Sigfox does not permit to have a private network and it has no encryption.

- Messages and Payload length

LoRaWAN allows sending unlimited messages per day with a maximum of 243 bytes of data to be sent. In contrary, Sigfox only allows to send 140 messages (UL) and 4 messages (DL) with the maximum payload of 12 bytes in uplink and 8 bytes in downlink. In the project, is needed to send messages from the sensors all the time.

- Network coverage and Range

The major utilization advantage of Sigfox is that an entire city can be covered by one single base station (i.e., range >40 km). By contrast, LoRaWAN has a lower range (i.e., range <20 km) that requires only three base stations to cover an entire city such as Barcelona. For this application, the range of LoRaWAN is enough to cover the Street.

Taking into account all the parameters analyzed in this section, it is concluded to use **LoRaWAN** as communication technology for this project.

### 3. The Things Network (TTN)

The Things Network is a non-profit organization that is building a network for the Internet of Things by creating abundant data connectivity for the development of applications. They are a contributor member of the LoRa Alliance and the technology that they use for their network is LoRaWAN.

They are creating a perfect backend for the developing of IoT applications. They provide the users with everything that they need to construct their applications; nodes, gateways and a network server. They also have a lot of different integrations; MQTT, HTTP, Amazon webservices etc. Their network scheme is shown in Figure 1.

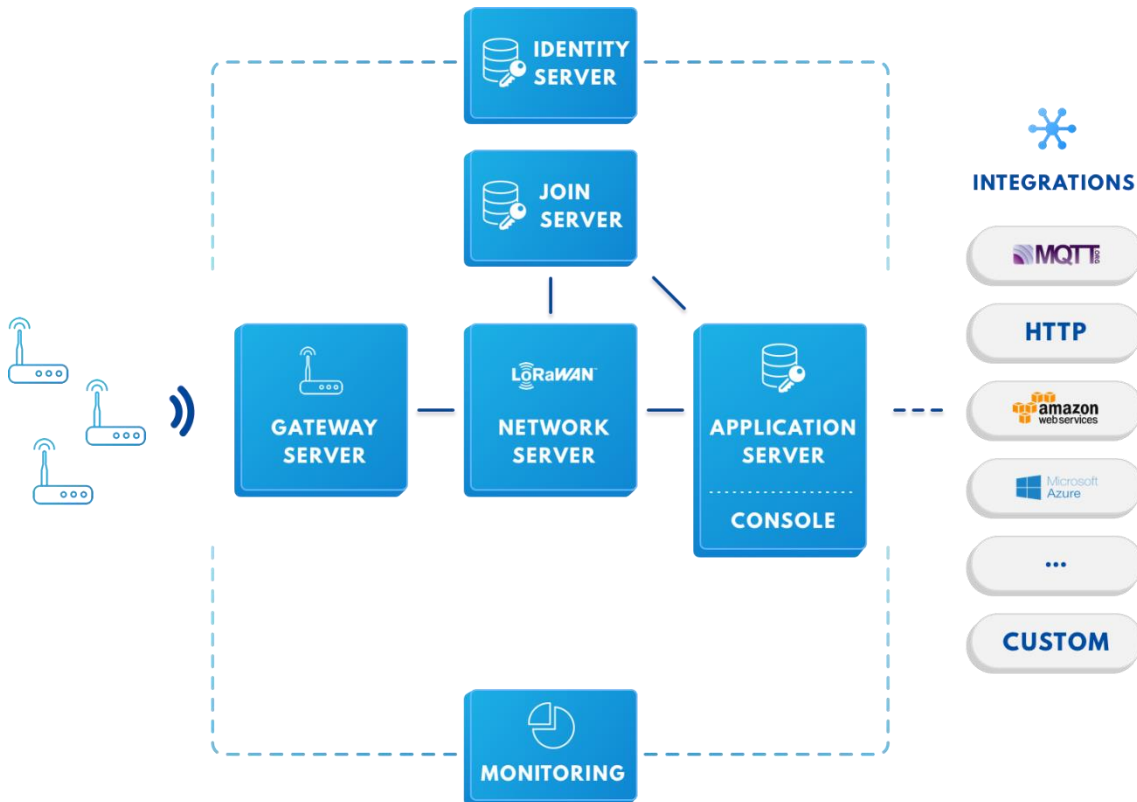


Figure 1: TTN Network scheme [1].

Their network is created by people; everyone can contribute by placing a gateway and expanding their network. The more gateways are placed, the larger the coverage. Figure 2 shows their actual coverage.



Figure 2: Coverage of TTN [1].

A network server that understands LoRaWAN technology is needed for the project, The Things Network backend is going to be used to provide the network server to the project.

The network server performs several task related to the gateway, end-devices and the application; such as integrity check, frame counter update, downlink template generation, etc.

The gateway related functions are scheduling and managing the utilization of the gateways. Scheduling is needed because a gateway can only do one transmission at the same time. The utilization information is used to evenly distribute load over different gateways and to be compliant with the European duty cycles. Another important feature is monitoring the status of each gateway.

The end device related function is to manage the state of devices in the network. As device address is non-unique, the network has to keep track of which addresses are used by which devices in order to map a message to the correct device and application. Other things the network must keep track of are the security keys and frame counters.

They also provide some functionality related to applications. For example, the Brokers and Handlers need to know to which server traffic for a specific application needs to be forwarded. The Handlers need to know how to interpret binary data, and bridge to higher-layer protocols, such as AMQP and MQTT.

Finally The Things Network is a distributed network, there has to be functionality that supports this distribution. Service discovery functionality helps components to determine where traffic should be routed to. Currently, this is implemented as a centralized Discovery server, giving The Things Network Foundation control over which components are allowed to announce specific services.

### 3.1. Limitations

The Things Network has limitations due to their “Fair access policy” limiting the number of packets that the end-device can send to the gateway. This policy states are the following:

- Maximum of 30 seconds of uplink time on air, per day and per device.
- Maximum of 10 downlink messages per day, per device.

Technically the maximum bytes that are able to send are 51 bytes. But, the more bytes sent, the more airtime needed to send the package. Therefore it does not depend in the number of bytes sent, in depends on no overcome the maximum allotted time.

The following regulations apply as well:

#### **LoRaWAN data rate and application packet sizes**

The data rate and maximum packet size roughly depend on the distance to the nearest gateway and the type of data to be sent, and are also defined in the specification for each region. For the European 863-870MHz band, the *application* packet size varies between 51 bytes for the slowest data rate, and 222 bytes for faster rates. Note that the LoRaWAN protocol adds at least 13 bytes to the application payload. Also, the library used to transmit the data to the Gateway LMIC only support 51 bytes for all SF.

Devices with fixed hardcoded data rates of SF12 or SF11 are not allowed to join the network.

**LoRaWAN transmit duty cycles and dwell times**

To avoid network congestion, LoRaWAN defines some maximum transmit duty cycles 320 and maximum transmit times (dwell times). These depend on many factors including the region and the type of operation (like sending data, or broadcasting a request to join a network).

For the European EU 863-870MHz ISM Band the specification 306 limits the duty cycle to 1% for data:

All of this means that the more distance between the node and the Gateway the lower data rate has to be chosen to ensure that the Gateway will receive the data. But a lower data rate implies a longer air time for each byte. This limits the maximum packet size, to ensure other end-devices get time to use the network as well. And when transmitting takes longer, a device cannot send as often as it might want to.

The uplink limit of the TTN Fair Access Policy is based on the following:

- 8 frequencies
- 5% receive duty cycle on the gateway
- 86,400 seconds in a day
- 1,000 nodes

Or:  $(8 \times 86,400 \times 0.05) / 1,000$  yields approximately 30 seconds per node per day.

In the project, the unique device use SF7 (due to the device is close to the Gateway) and 29 bytes of payload are sent. It takes 87.3ms on air. This means that only 343 messages can be sent without overcoming the limit of air-time per device per day.



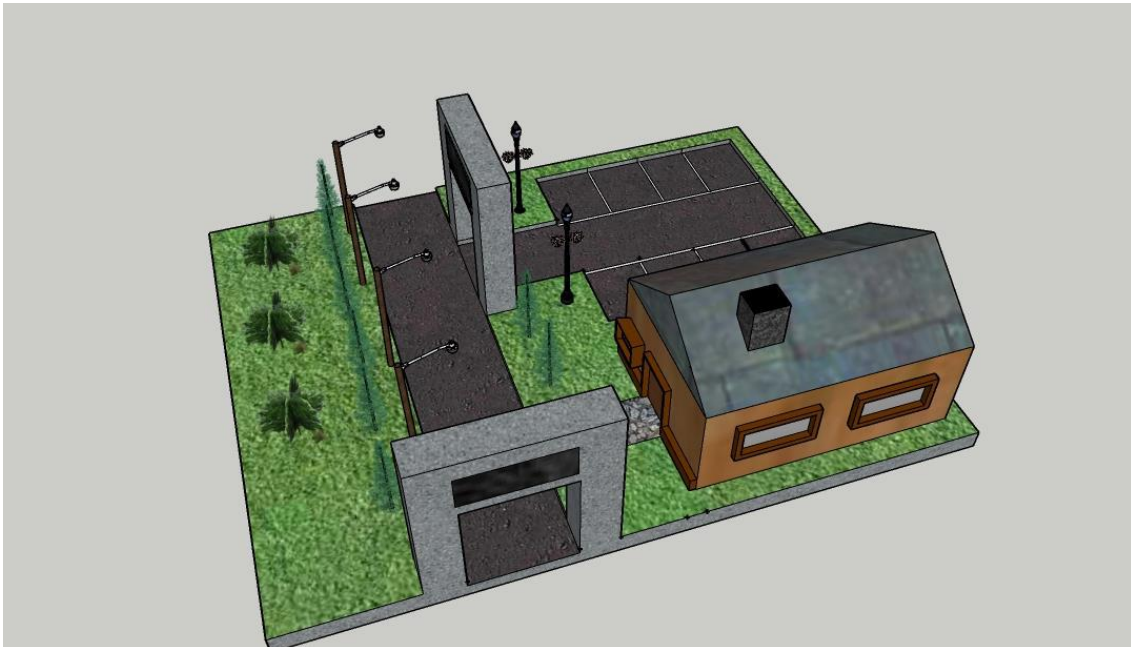
## 4. Design

This chapter is divided in five sections; in the first section, the design of the project is presented. The applications and objectives are explained. In the following section, the network architecture based on end-device, Gateway and Server is described. The section 4.3 is used to specify the requirements of the hardware that is going to be used. With all the requirements presented in the third section, in the next section the hardware is selected. Finally a cost estimate is done.

### 4.1. Overview

The task of the project is to create a real Smart Street to show the advantages of the Smart Grids. Smart Grid technology allows city officials to interact directly with both community and city infrastructure and to monitor what is happening in the city and how the city is evolving. This means **real time control, automatization, consumption reduction and better quality of service.**

The design of the project is shown in Figure 3.



*Figure 3: Design of the project.*

The project will be focused on four applications; the measuring of the temperature and humidity, the detection of air pollution and noise and the car park and the light control. These functionalities are going to reduce the consumption and improve the quality of life of the citizens (users).

Measuring temperature and humidity are going to permit a better planning of the heating and air conditioner reducing the consumption of electricity.

Air Pollution monitoring can give to the city an up-to-date and accurate data to help in the pollution reduction; reducing the traffic of the cities and controlling the contamination of the companies. Street disturbances would be monitored using noise detection, and this will help the community to have a comfortable environment.

Car parking is one of the biggest complaints and issues for cities and is directly tied to retail traffic and tax revenue. To solve that, a system is going to be created to show the availability of the parking, showing the number of places that are empty or full in the parking. This application allows to know in real time the status of the car park and this can help both sides; the user and the manager of the application. The manager of the application can know the usage of the parking, knowing the most requested hours and doing a better organization of the traffic. The user knows if there is a place for his car or not, saving him the time of looking for a new parking place.

The light control is going to reduce the consumption of electricity, adapting the amount of light from streetlights depending on the sunlight.

#### 4.2. Integration

For doing all the applications mentioned in section 4.1, sensors are needed to measure the different physical parameters; temperature, humidity, air pollution, level of sound, amount of light and proximity for knowing if a car is in the parking or not.

After, all these sensors have to send the data to a network architecture to be able to process the data and finally an application has to show these results to a final user.

The network architecture chosen for this project is LoRa Network Architecture. The election of this technology is explained in the following chapters.

To understand the magnitude of the project Figure 4 represents the schema of all the parts.

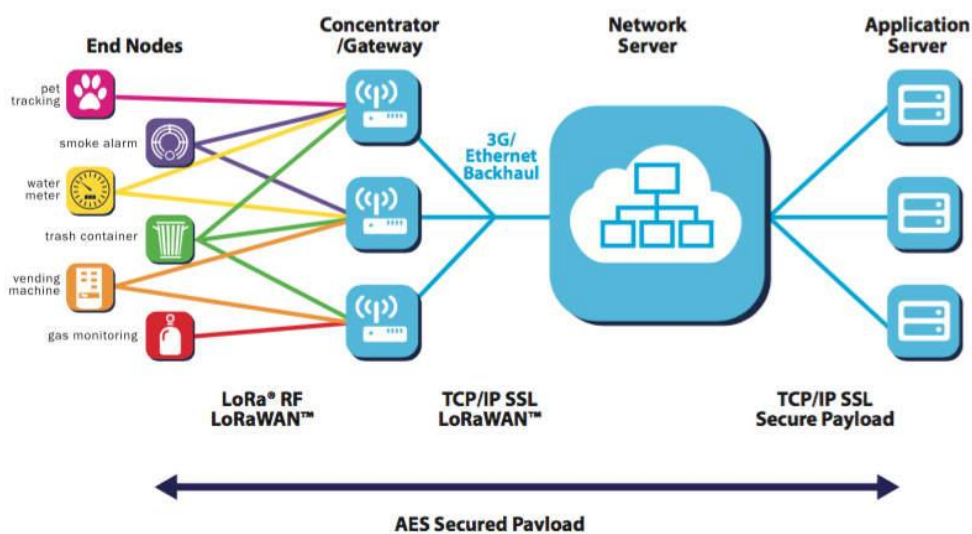


Figure 4: Scheme of the project.

The end nodes are going to be the sensors measuring in real time. These sensors are going to send the data to a gateway using LoRaWAN communication protocol with LoRa for the physical layer. This gateway is going to send the data to a server via Ethernet, this gateway is going to act as a repeater and the data is going to be processed on the server. Finally the processed data is going to be shown in a website using data storage implementation.

The first steps of the project are to choose and buy the needed sensors. Then, when the sensors will be measuring correctly they have to communicate with the gateway. This project designs a way to communicate the sensors using LoRaWAN technology. A gateway is designed

and built to work with LoRaWAN technology and to be able to communicate with the server. The server is given by The Things Network (TTN) as it is explained in chapter 3.

Due to the magnitude of this project, it is made by two people: Koldo Antolinez Jimenez and Laura Guijarro Iguacel forming two complementary reports. The report done by Antolinez is called *Smart Street “Proof-of-concept”- Data Acquisition* and is focused in the part of the sensors and therefore the data acquisition; while report done by Guijarro is called *Smart Street “Proof-of-concept”- Data Processing* and is focused in the communication protocol and final application.

### 4.3. Requirement specifications

This section will talk about the functional and non-functional requirements the gateway and the end device have to fulfill in order for the project to be considered successful.

Furthermore, it must be pointed out that the final destination of the data, sent by the end device and forwarded by the gateway, is an application. Similarly to the end device and the gateway, the application has to meet some requirements in order to recover useful information. Therefore, the application requirements will also be described in this chapter.

#### 4.3.1. Gateway

Here the requirements that the gateway must meet are defined. Although some of the requirements are not strictly necessary for the gateway to work, all of them should be met for a quality experience when using it.

##### **Functional requirements**

- Ability to receive packets over the LoRa physical layer.
- Ability to forward the received packets over to TTN via either Wi-Fi or Ethernet connection in order to reach the application.

##### **Non-functional requirements**

- Fully LoRaWAN compatible.
- Supported by TTN, since the aim is to forward the packets through it.
- Wide coverage.
- Reasonably cheap. Only the necessary features.
- Must provide security throughout the entire communication process.
- Availability. It should not have long periods of downtime.
- Easy to access and maintain.
- Efficient, as to make the power consumption as low as possible.
- Must support interoperability, as to be able to receive and forward packets from a wide range of different end devices.
- Must support concurrency, the gateway should be able to handle several nodes transmitting simultaneously.
- Scalable. The gateway must be able to handle the exponential growth of IoT devices.

#### 4.3.2. End device

In this section, the requirements the end device after the implementation phase should meet will be listed. Again, all of the requirements do not need to be completed so that the end device can work, in fact as the project will end in a simulation, all of them will not be fulfilled.

However, the idea is to build an end device that closely resembles one that would be used in a real scenario, hence why all of the requirements should be met.

**Function requirements:**

- Able to transmit and receive packets over LoRa as the physical layer.
- Must be able to encrypt the packets before sending them.
- Will allow customizable payloads, such as a simple string or data from a sensor.
- Will have the option to change the spreading factor used, as to adapt to different distances and data rates.
- Capable of changing the transmission frequency. In order to comply with TTN’s fair access policy.

**Non-functional requirements:**

- Must be cheap and affordable.
- Power efficient. A very important aspect, since in a real scenario the end device will be powered by a battery.

#### 4.3.3. Application

Finally, here the requirements that the application must meet are listed. Moreover, it is important that the application meets these basic requirements, as that will help to perform the testing phase properly and show data to the manager and users of the application.

**Functional requirements**

- Able to decrypt the payload and display content in plain text.
- Allows changing the encryption keys in case it is needed.
- Must not be restricted to one device. It needs to be capable of receiving packets from different end-devices. Therefore, it also must be able to distinguish packets from different end-devices.

**Non-functional requirements**

- Availability. Similarly to the gateway, it must have a little downtime as possible.
- Readability. The data should be easy to interpret. It must be intuitive no matter what is the technical knowledge of the final user of the application; it is directed to everyone.
- Informative. It should display the information for different parameters, such as RSSI or SNR.

#### 4.4. Hardware selected

This section will explain the choice of the hardware that will be used to develop the gateway and the end-device taking into account the requirements analyzed in the previous part (4) of this report.

##### 4.4.1. Gateway

The gateway used will be a DIY multi-channel Raspberry Pi gateway, therefore to construct it the following hardware will be needed:

- Embedded Linux board.
- Concentrator.
- Antenna.

- Pigtail for the antenna.

### **Embedded Linux board**

There are several embedded Linux boards to choose from such as Raspberry Pi Beagle Bone and Banana Pi among others. Raspberry Pi is the one chosen for this project as it is the most extended Linux board, therefore is the one most documented for constructing things.

In this project Raspberry Pi 3 model B is used, which is well over the minimum requirements but it will help improve the overall performance of the gateway.

### **Concentrator**

As mentioned in previous chapters, the concentrator is the piece of hardware that allows the gateway to be multi-channel.

Regarding to the specifications, the concentrator needs to be LoRaWAN compatible and be able to handle packets with different spreading factors and data rates. It is found that IMST IC880A meets all of the requirements the gateway needs and so it will be the one used in this project.

Lastly, this concentrator allows for female jumper wire as interface between it and the board. This means that double female jumpers wires will be used to connect the concentrator and the Raspberry Pi.

### **Antenna**

An antenna is the last and crucial part needed to have a functional gateway, whose function consists of transmitting and receiving the signal over the 868MHz frequency range, in charge of transmitting and receiving the LoRa packets over the ISM band.

The concentrator’s manufacturer sells the antenna and pigtail suitable for the concentrator used in this project (see Figure 5). Therefore and mainly for convenience, will be using those ones. However, any antenna that supports the adequate frequency range, 868MHz in this case, should work too.



*Figure 5: Antenna and pigtail for IC880A concentrator.*

#### 4.4.2. End device

In this section, the design of the end device is explained. It is divided into two different parts; first it is explained which hardware is used to create the communication between the end device and the gateway using LoRa communication, and secondly the sensors used to measure the physical parameters introduced at the beginning of the chapter are commented.

##### Development board

The microprocessor that will be the head which will manage all the sensors and the communication parameters will be Arduino DUE. Taking into account the amount of pins required for the project because of the communication module and the quantity of sensors, Arduino UNO’s amount of pins was not enough.

##### Transceiver module

For creating the communication, Dragino LoRa shield (shown in Figure 6, which demonstrates that the amount of free pins is very limited if connected to Arduino UNO) is used. This is the part in charge of taking the packets and modulate them using LoRa.

It is compatible with Arduino DUE. This shield has embedded the Semtech SX1276 chip which is the chip that allows doing the connectivity LoRaWAN. As being in Europe the module has to support the 868MHz frequency range, the frequency range is limited by the passive hardware the shield has, so when buying it is important to buy the correct version so that it transmits in the required frequency range.

#### Pin Mapping For LoRa

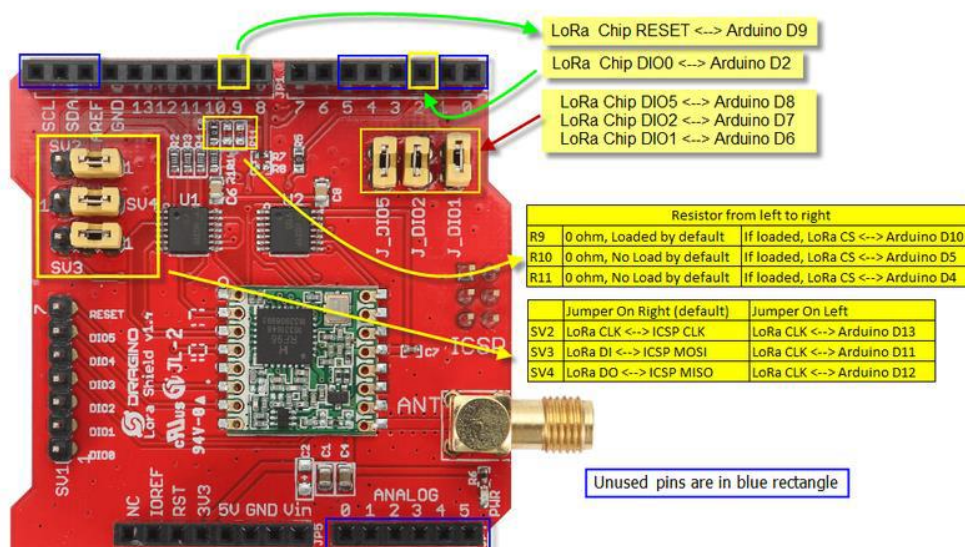


Figure 6: Dragino LoRa shield pin-map [2].

##### Antenna

Similarly to the gateway, the antenna is a crucial part of the communication part in the end device, in charge of transmitting and receiving the LoRa packets over the ISM band.

When buying the Dragino LoRa shield, this is sold with the required antenna, so to that is the antenna used in the project. It is an antenna based on an 868MHz frequency.

With this last piece all the necessary communication hardware to create the end device is covered so the only last necessary thing is to assemble all the parts together, which is not a difficult task as using a shield for Arduino simplifies a lot the wiring.

## Sensors

Measuring the physical parameters of real world into electrical signals that the microprocessor can understand and manage is done by sensors. Nowadays there are sensors that are able to measure every parameter in the world. Once the physical parameters to measure are defined is time to see what sensor is able to convert that state into electrical signals. In Table 4 is summarized what sensor is used to measure every parameter defined in this project.

*Table 4: Summary of sensors used to measure real world parameters.*

Real world parameter	Sensor used
Street luminosity	Light Dependent Resistor (LDR)
Presence of a car in the parking slot	Ultrasonic sensor, HC-SR04
Presence of a car in front of the barrier	Passive Infrared Sensor (PIR), HC-SR501
Air quality	Sharp GP2Y1010AU0F Optical Air Quality sensor
Temperature and humidity	Sensor DHT22
Sound quality	KY-037 sound detection module

## Actuators

Once the sensors measured the parameters to analyze in the project, it is usually interesting to have something that will act so that depending on the lecture of the sensor a desired state is achieved, i.e. turn on the heating when the temperature in the room is below 21°C.

In the case of this project, most of the sensors' purpose is merely informative; however, there are a couple of lectures that should be interactive with the real world:

- Streetlights must turn ON when the luminosity in the street is low.
- Parking center barrier must open when a car is detected in front of it.

## 4.5. Cost estimation

This section analyses the cost of the resources needed to develop the gateway and end device concerning the hardware.

### 4.5.1. Gateway

Taking into account the hardware needed (introduced in section 4.4.1) Table 5 details the approximate cost of each part of the hardware and the total approximate cost of it, which would be around 185.78€.

*Table 5: Gateway hardware cost.*

Concept	Unit cost	Quantity	Total
<b>Raspberry Pi 3 model B</b>	39.13€	1	39.13€
<b>IC880A concentrator</b>	132.25€	1	132.25€
<b>Antenna</b>	6.50€	1	6.50€
<b>Pigtail for antenna</b>	6.50€	1	6.50€
<b>Board to IC880A interface</b>	0.20€	7	1.40€
<b>Total</b>			<b>185.78€</b>

#### 4.5.2. End device

Taking into account the hardware needed (introduced in section 4.4.2) Table 6 and Table 7 detail the approximate cost of each part of the hardware and the total approximate cost of it. In this case the costs are divided into two parts: the cost of adding a new end-device and the cost of the sensors used in this project.

Table 6: Bare end device hardware cost.

Concept	Unit cost	Quantity	Total
Arduino DUE	31.77 €	1	31.77€
Dragino LoRa shield	24.28€	1	24.28 €
<b>Total</b>			<b>56.05 €</b>

Table 7: Cost of the sensor and actuators used in the Smart street project.

Concept	Unit cost	Quantity	Total
LDR	0.10€	1	0.10€
HC-SR04	3.30€	6	19.80€
HC-SR501	2.91€	2	5.82€
Sharp GP2Y1010AU0F Optical Air Quality Sensor	10.68	1	10.68
SOUND	2.25€	1	2.25€
DHT 22	7.53€	1	7.53€
LEDs	1.12€	14	15.68€
Servo motors	4.66€	2	9.32€
<b>Total</b>			<b>71.18€</b>

The whole cost of the end device in this project is around 127.23€.

#### 4.5.3. Total budget

To wrap up, it is now possible to make an estimate cost of the project’s hardware based on the three tables of budgets in this section.

Table 8: Total cost of the hardware in Smart street project.

Concept	Total
Hardware gateway	185.78€
Hardware end device	127.23 €
<b>Total</b>	<b>313.01€</b>

Finally, from Table 8, the total cost estimated for this project is 313.01€, THREE HUNDRED AND THIRTEEN EUROS WITH ONE CENT.



## 5. LoRaWan Overview

The purpose of this chapter is to introduce the protocols and technologies chosen in the project. This chapter is divided into two parts:

1. **LoRa**. It is the short for "Long Range". It is the physical layer of LoRaWAN and enables extremely long range communication with optimized energy efficiency and the robust signal.
2. **LoRaWAN**. It also known as LPWAN, stands for low power, wide area network. It is the network protocol and the core of the project.

### 5.1. Lora

The following section is based on Semtech's LoRa description [2].

LoRa technology was developed by a company called Semtech and it is a wireless protocol designed specifically for long-range, low-power communications. LoRa stands for Long Range Radio and is mainly targeted for M2M and IoT networks. This technology will enable public or multi-tenant networks to connect a number of applications running on the same network.

LoRa Alliance was formed to standardize LPWAN (Low Power Wide Area Networks) for IoT and is a non-profit association which features membership from a number of key market shareholders such as CISCO, actility, MicroChip, IBM, STMicro, SEMTECH, Orange mobile and many more. This alliance is the key to provide interoperability among multiple nationwide networks.

Each LoRa gateway has the ability to handle up to millions of nodes. The signals can span a significant distance, which means that less infrastructure is required, making constructing a network much cheaper and faster to implement.

LoRa also features an adaptive data rate algorithm to help maximize the nodes battery life and network capacity. The LoRa protocol includes a number of different layers including encryption at the network, application and device level for secure communications.

#### 5.1.1. LoRa frequency bands

The LoRa wireless system makes use of the unlicensed frequencies that are available worldwide. The most widely used frequencies / bands are:

- 868 MHz for Europe
- 915 MHz for North America
- 433 MHz band for Asia

Using lower frequencies than those of the 2.4 or 5.8 GHz, ISM bands enables much better coverage to be achieved especially when the nodes are within buildings.

#### 5.1.2. LoRa Modulation

The modulation of LoRa is a unique spread spectrum technique, which provides robustness against interferences and a very low minimum SNR for the receiver to be able to demodulate the signal.

In this section, first the traditional way of applying the spread spectrum technique, Direct Sequence Spread Spectrum (DSSS) is introduced to then be able to understand better LoRa modulation technique.

### Direct Sequence Spread Spectrum (DSSS)

In this modulation, the carrier phase of the transmitter changes in accordance with a code sequence. For this, the wanted data signal is multiplied with a spreading code, also known as a chip sequence. The chip sequence has a higher data rate than the data signal, which divides data based on a spreading ratio. As a result, this expanded signal has a greater bandwidth than the original one. This process is shown in Figure 7.

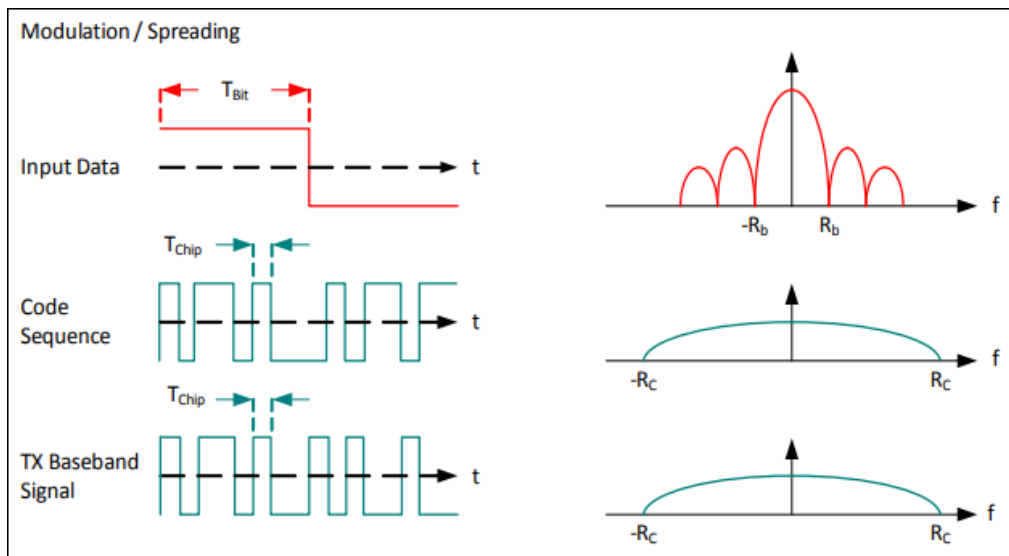


Figure 7: Modulation/ Spreading Process [2].

In the Figure 7  $R_b$  is referred to the bit rate (bits/seconds) and  $R_c$  is referred to the chip rate (Chips/seconds). The amount of spreading is dependent on the ratio "chips per bit".

At the receiver, the expanded signal received is multiplied with the same spreading sequence that it was generated in the transmitter to recover the wanted data signal. This multiplication process in the receiver effectively compresses the spread signal back to its original un-spread bandwidth. If the chip sequence is not the same in the receiver and in the transmitter, the information will not be recovered. This recovery process is shown in the Figure 8.

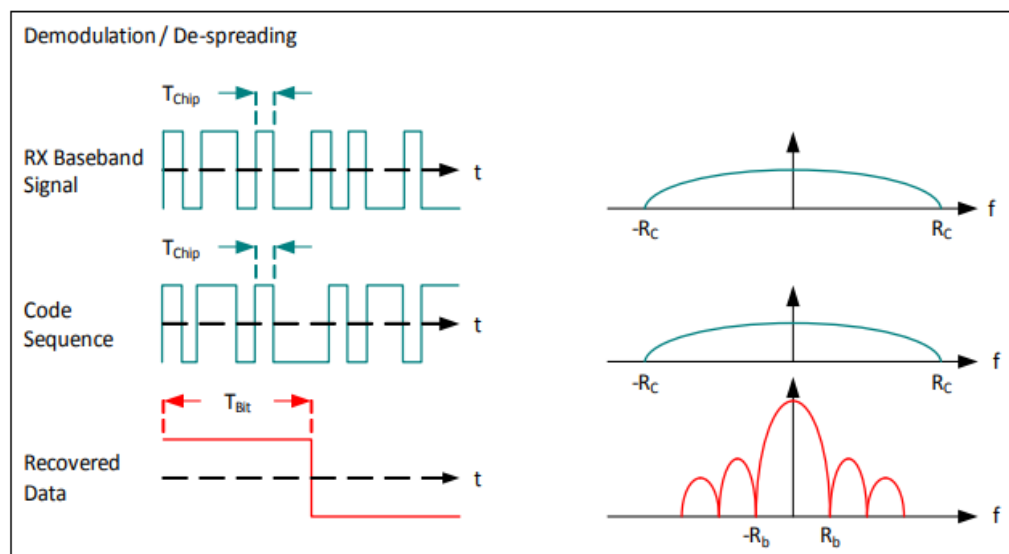


Figure 8: Demodulation /De-spreading Process [2].

This method helps to increase the signal's resistance to interference increasing the receiver sensibility (which enables the receiver to correctly recover the data signal even when the SNR of the channel is a negative value). If any bits are damaged during transmission, the original data can be recovered due to the redundancy of transmission.

This modulation presents some problems for low-cost or power-constrained devices, as they require a highly accurate and expensive reference clock source. Furthermore, the longer the spreading code, the longer the time required by the receiver to perform a correlation over the entire length of the code sequence. This is especially of concern for power-constrained devices that cannot be “always-on” and thus need to repeatedly and rapidly synchronize.

### LoRa Spread Spectrum

LoRa modulation is designed for low-cost and low-power requirements. The difference with the traditional modulation is that the chirp signal used to expand the data signal is continuously varying on frequency, as is shown in the Figure 9.

Due to the linearity of the chirp pulses, the frequency offsets created between the receiver and the transmitter are equivalent to timing offset, greatly reducing the complexity of the receiver design. The frequency offset between the transmitter and the receiver can reach 20% of the bandwidth without impacting decoding performance. This helps with reducing the price of LoRa transmitters, as the crystals embedded in the transmitters do not need to be manufactured to extreme accuracy. Also, this frequency offset tolerance makes this modulation immune to the Doppler Effect which makes LoRa ideal for mobile data communications.

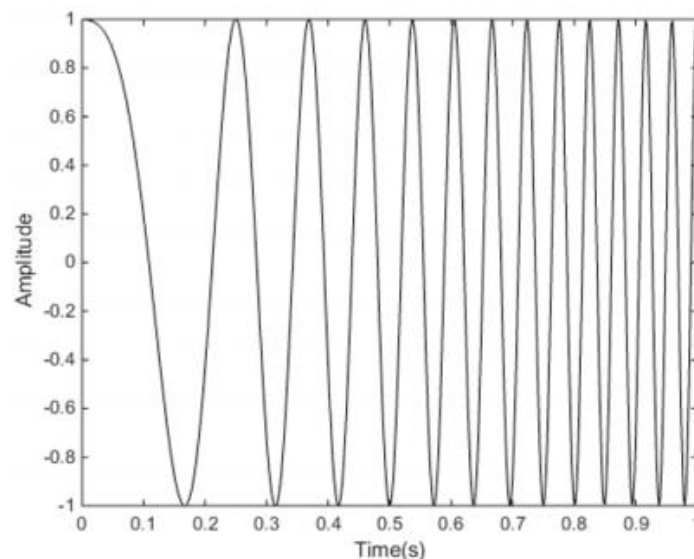


Figure 9: Chip signal in the time domain

The LoRa symbol period is longer than the usual short-duration burst of FHSS systems so the errors generated by such interface are easily corrected through Forward Error-correction Codes (FECs). This gives robustness to the modulation. Also we have protection to multipath and fading thanks to the chirp pulse being relatively broadband.

LoRa modulation has three parameters to customize and depending how you configure them the characteristics of the link are going to change. These parameters are:

1. Bandwidth (BW)
2. Spreading Factor (SF)
3. Code Rate (CR)

We can define the nominal bit rate of the signal depending on these parameters:

$$R_b = SF * \frac{CR}{2^{SF}/BW}$$

Where:

SF = spreading factor (7 up to 12)

CR = code rate (4/5, 4/6, 4/7, 4/8)

BW = modulation bandwidth (Hz)

$$CR = \frac{4}{4 + n} \text{ where } n \in (1,2,3,4)$$

One of the principle design compromises that the designer must manage in the selection of spreading factor is that of time on air (packet duration) versus occupied bandwidth. The representation of a single bit by many chips implies that the chips must either be sent faster than the original bitrate – increasing the occupied bandwidth of the signal, or in the same bandwidth – increasing the time taken to transmit the information.

LoRa modulation sends the spread data stream at a chip rate equal to the programmed bandwidth in chipsper-second-per-Hertz. So a LoRa bandwidth of 125 kHz corresponds to a chip rate of 125 kcps.

Furthermore, we can also adapt our data rate depending on what we need, using different spreading factors. There is a compromise, if a robust communication against interferences is wanted; a high spreading factor has to be used, at the expense of lowering the data rate. In addition, using a higher spreading factor also translates into a longer airtime for the signal. LoRa modulation employs orthogonal spreading factors which enables multiple spread signals to be transmitted at the same time and on the same channel without minimal degradation the RX.

The receiver sensibility also depends on these three factors and is shown in the following equation:

$$S = -174 + 10\log_{10}BW + NF + SNR$$

The first term is due to thermal noise in 1 Hz of bandwidth and can only be influenced by changing the temperature of the receiver. The second term, BW, is the receiver bandwidth. NF is the receiver noise figure and is fixed for a given hardware implementation. Finally, SNR represents the signal to noise ratio required by the underlying modulation scheme. It is the signal to noise ratio and bandwidth that are available as design variables to the LoRa designer.

### LoRa Packet Format

A physical frame format is specified and implemented in Semtech’s transmitters and receivers. The bandwidth and spreading factor are constant for a frame.

A LoRa frame begins with a preamble. The preamble starts with a sequence of constant upchirps that cover the whole frequency band. This preamble is used for the synchronization. The last two upchirps encode the sync word. The sync word is a one-byte value that is used to differentiate LoRa networks that use the same frequency bands. A device configured with a given sync word will stop listening to a transmission if the decoded sync word does not match its configuration. The structure of the preamble can be seen in Figure 10.

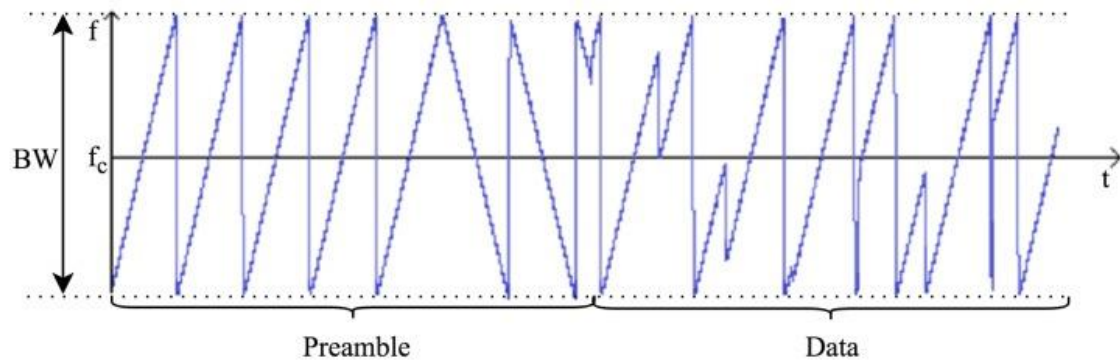


Figure 10: Structure of the data

After the preamble, there is an optional header. When it is present, this header is transmitted with a code rate of 4/8. This indicates the size of the payload (in bytes), the code rate used for the end of the transmission and whether or not a 16-bit CRC for the payload is present at the end of the frame. The header also includes a CRC to allow the receiver to discard packets with invalid headers. The payload size is stored using one byte, limiting the size of the payload to 255 bytes. The header is optional to allow disabling it in situations where it is not necessary, for instance when the payload length, coding rate and CRC presence are known in advance.

The payload is sent after the header, and at the end of the frame is the optional CRC. A schematic summarizing the frame format can be seen in the Figure 11.

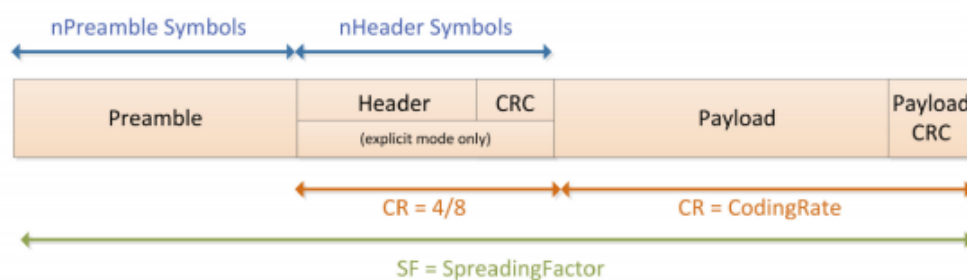


Figure 11: LoRa Modem Packet formatting

## 5.2. LoRaWAN

This section is based on the LoRaWAN specification defined by the LoRa Alliance [3].

LoRaWAN is a MAC protocol, built to use the LoRa physical layer. The LoRaWAN specification is a Low Power, Wide Area (LPWA) designed mainly for battery-powered end-devices that are either mobile or static which requirements are bi-directional communication, end-to-end security, mobility and localization services.

### 5.2.1. LoRaWAN topology

LoRaWAN networks typically are laid out in a star-of-stars topology, which includes three different types of devices, as shown in Figure 12.

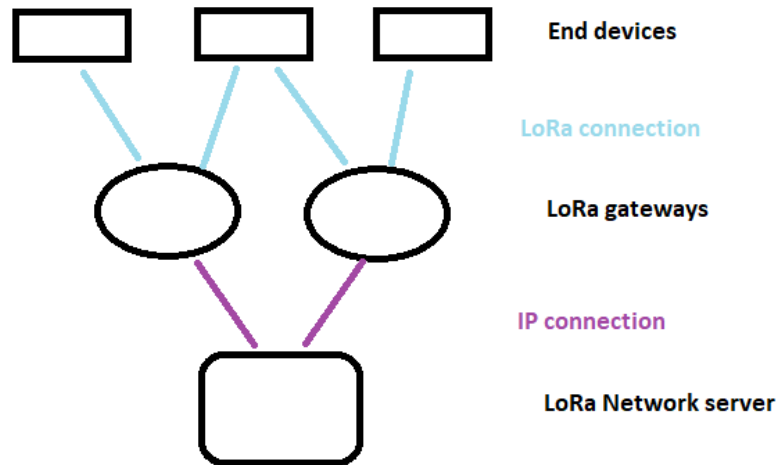


Figure 12: LoRaWAN topology

The basic architecture of a LoRaWAN network is as follows: end-devices communicate with gateways using LoRa with LoRaWAN. Gateways forward raw LoRaWAN frames from devices to a network server over a backhaul interface with a higher throughput, typically Ethernet or 3G.

Consequently, gateways are only bidirectional relays, or protocol converters, with the network server being the server the responsible for decoding the packets sent by the devices and generating the packets that should be sent back to the devices.

In a LoRaWAN network nodes are not associated with a specific gateway. Instead, data transmitted by a node is typically received by multiples gateways. The nodes in a LoRaWAN network are asynchronous and communicate when they have data ready to send whether event-driven or scheduled. This type of protocol is typically referred to as the Aloha method. In a mesh network or with a synchronous network, such as cellular, the nodes frequently have to ‘wake up’ to synchronize with the network and check for messages. This synchronization consumes significant energy and is the number one driver of battery lifetime reduction.

The wireless communication takes the advantage of the Long Range characteristics of the physical layer LoRa, allowing a single-hop link between the end-device and one gateway or different gateways.

In order to make a long range star network viable, the gateway must have a very high capacity or capability to receive messages from a very high volume of nodes. High network capacity in a LoRaWAN network is achieved by utilizing adaptive data rate and by using a multichannel multi-modem transceiver in the gateway so that simultaneous messages on multiple channels can be received. The selection of the data rate depends on the communication range and message duration, and also different data rates are selected not to interfere with each other. LoRa data rates range from 0.3 kbps to 50 kbps. In order to maximize both battery life of the end-devices and overall network capacity, the LoRa network infrastructure can manage the data rate and RF output for each end-device individually by means of an adaptive data rate (ADR) scheme.

When an end-device wants to transmit, it has to respect the following rules, in order to transmit on an available channel at any time:

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

### 5.2.2. LoRaWAN Classes

LoRaWAN has three different classes of end-point devices to address the different needs reflected in the wide range of applications. All LoRaWAN devices have to implement at least the Class A that is described in this section. In addition, they can implement Class B or Class C also described in this part. It is shown in Figure 13.

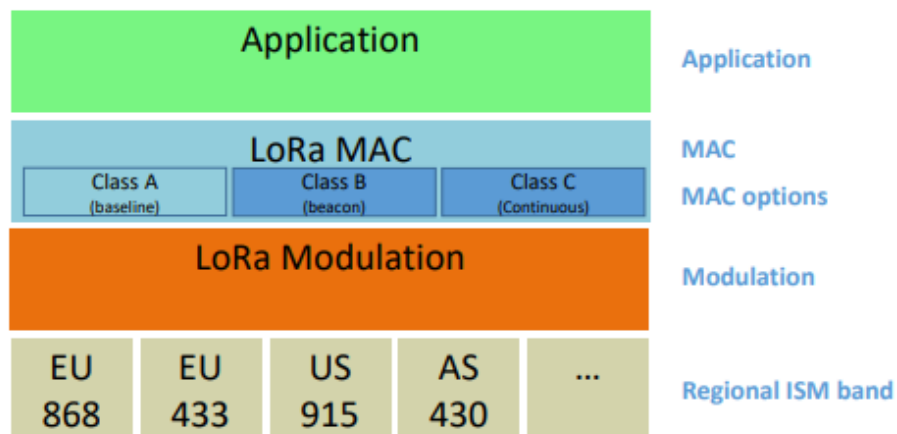


Figure 13: LoRaWAN classes

#### Class A - Lowest power, bi-directional end-devices:

Class A end-devices can schedule an uplink transmission based on their own needs, with a small jitter (random variation before transmission). This class of devices allows bi-directional communications, whereby each uplink transmission is followed by two short downlink receive windows. Downlink transmission from the server at any other time has to wait until the next uplink transmission occurs. This is an ALOHA type of protocol.

The end-device is able to enter low-power sleep mode for as long as defined by its own application: there is no network requirement for periodic wake-ups. This makes class A the lowest power operating mode, while still allowing uplink communication at any time.

#### Class B – Bi-directional end-devices with deterministic downlink latency:

Class B, bi-directional with scheduled receives slots: Class B end-devices open extra receive windows at scheduled times. This provides the network the ability to send downlink communications with a deterministic latency, but at the expense of some additional power consumption in the end-device. A synchronized beacon from the gateway is thus required, so that the network server is able to know when the end-device is listening.

**Class C – Lowest latency, bi-directional end-devices:**

Class C, bi-directional with maximal receives slots: Class C end-devices have almost continuous receive windows. Based on this, the network server can initiate a downlink transmission at any time on the assumption that the end-device receiver is open, so no latency. The compromise is the power drain of the receiver (up to ~50mW) and so class C is suitable for applications where continuous power is available.

For battery powered devices, temporary mode switching between classes A & C is possible, and is useful for intermittent tasks such as firmware over-the-air updates.

It should be noted that LoRaWAN does not enable device-to-device communications: packets can only be transmitted from an end-device to the network server, or vice-versa. Device-to-device communication, if required, must thus be sling-shot through the network server (and consequently, by way of two gateway transmissions).

**5.2.3. Regional Parameters**

To be able to ensure interoperability between different LoRaWAN networks, the LoRaWAN specification defines the parameters that should be followed for each region.

Due to the project is going to be constructed in Czech Republic the 863-870 MHz ISM band is going to be used. This band permits different data rates depending on the bitrates that the transmitter wants to have. These data rates are directly related to the spreading factors seen previously in LoRa modulation. This is presented in Table 9.

*Table 9: EU863-870 Data rate [3].*

Data Rate	Configuration/Bandwidth	Bit Rate (bits/s)
<b>0</b>	LoRa: SF12 / 125kHz	250
<b>1</b>	LoRa: SF11 / 125kHz	440
<b>2</b>	LoRa: SF10 / 125kHz	980
<b>3</b>	LoRa: SF9 / 125kHz	1760
<b>4</b>	LoRa: SF8 / 125kHz	3125
<b>5</b>	LoRa: SF7 / 125kHz	5470
<b>6</b>	LoRa: SF7 / 250kHz	11000
<b>7</b>	FSK = 50kbps	50000
<b>8..14</b>	RFU	
<b>15</b>	Defined in LoRaWAN Alliance	

In order to access the physical medium of these free-license bands, the European Telecommunications Standards Institute (ETSI) imposes some restrictions such as maximum time the transmitter can be on or the maximum time a transmitter can transmit per hour. Currently, the LoRaWAN specification uses duty-cycled limited transmissions to comply with the ETSI regulations.

**5.2.4. Security**

In order to participate in a LoRaWAN network, an end-device must be activated. LoRaWAN provided two ways to activate an end-device: **Over-The-Air Activation** (OTAA) and **Activation By Personalization** (ABP).

The activation process should give the following information to an end-device:



- **End-device address (DevAddr):** A 32-bit identifier of the end-device. Seven bits are used as the network identifier, and 25 bits are used as the network address of the end-device.
- **Application identifier (AppEUI):** A global application ID in the IEEE EUI64 address space that uniquely identifies the owner of the end-device.
- **Network session key (NwkSKey):** A unique 128-bit key used by the network server and the end-device to calculate and verify the message integrity code of all data messages to ensure data integrity.
- **Application session key (AppSKey):** A unique 128-bit key used by the network server and end-device to encrypt and decrypt the payload field of data messages.

AES algorithms are used to provide authentication and integrity of packets to the network server and end-to-end encryption to the application server. By providing these two layers of cryptography (*AppSKey* and *NwkSKey*), it becomes possible to implement ‘multi-tenant’ shared networks without the network operator having visibility of the users payload data.

For OTAA, a join procedure with a join-request and a join-accept message exchange is used for each new session. Based on the join-accept message, the end-devices are able to obtain the new session keys (*NwkSKey* and *AppSKey*). For the ABP, the two session keys are directly stored into the end-devices. It means that the keys become static and do not change over different sessions.

## 6. “The Things Network” account

This chapter will describe the steps followed to sign up in The Things Network and shows what hardware can be added to their network. Everything is managed by their website [www.thethingsnetwork.org](http://www.thethingsnetwork.org).

When login in to the site, the first thing the users sees is the console where it gives the user the option to manage his applications or gateways as can be seen in Figure 14.

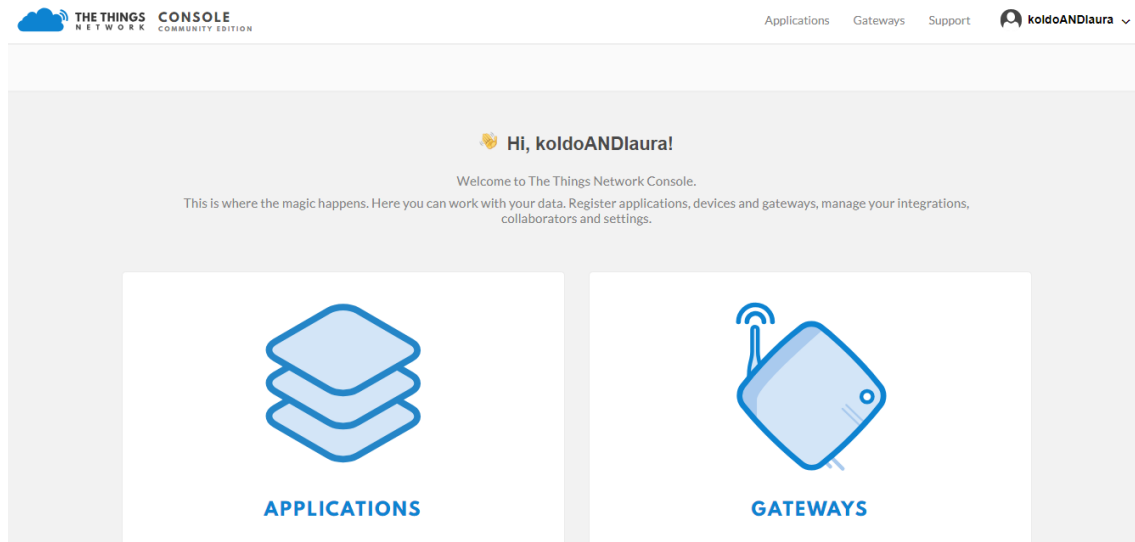


Figure 14: Overview of “console” section [1].

### 6.1. Creating an application in TTN.

When entering to the “Applications” section once a first application is created, the general overview is depicted in Figure 15. The information about the application is shown; the Application ID, the AppEUI and the AppKey are the most important parameters in this section. These parameters have been explained in section 5.2.4.

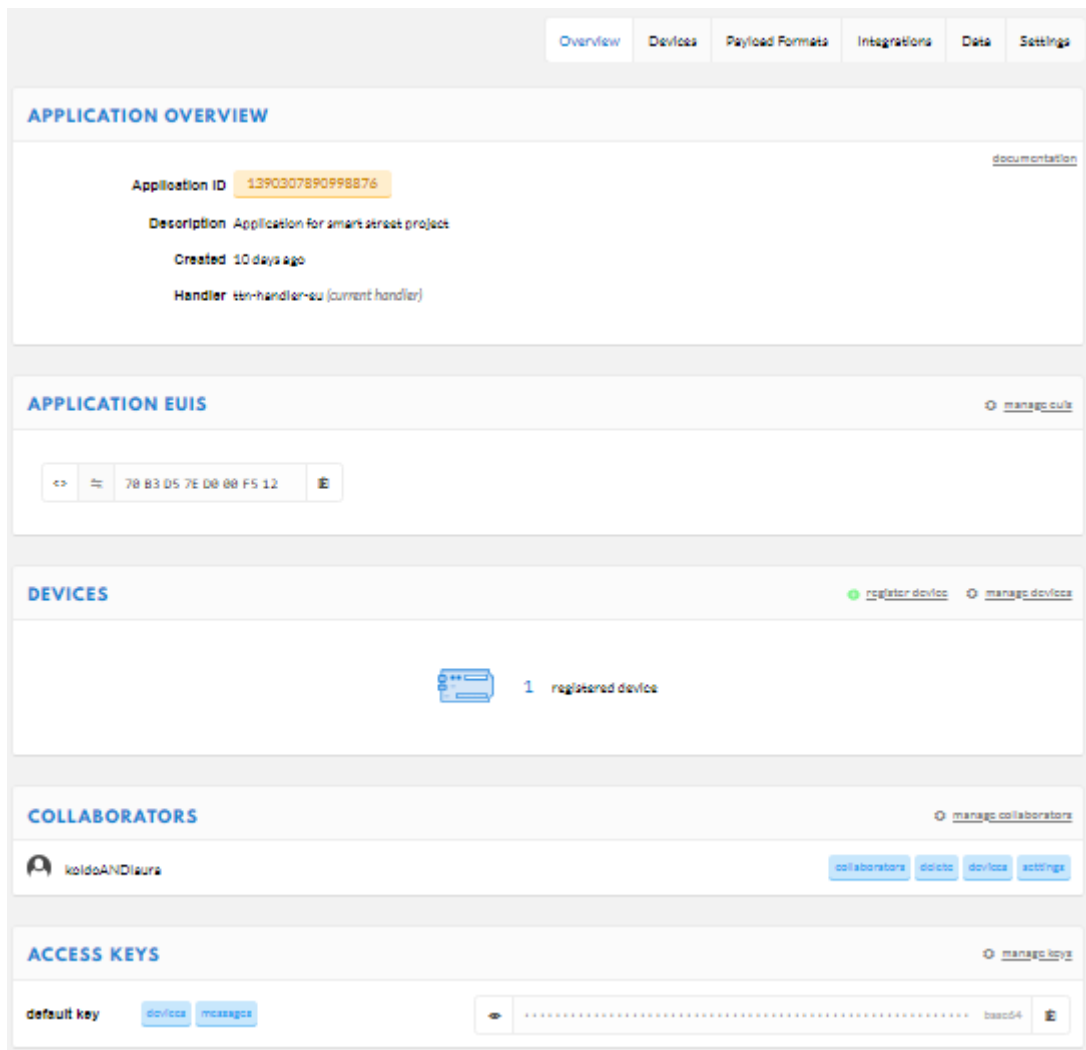


Figure 15: "Application" section overview [4].

## 6.2. Adding a gateway into TTN.

When entering to the "Gateways" section once a first application is created, the general overview is depicted in Figure 16. There information about the registered gateways is shown; the Gateway ID and the gateway key are the most important parameters in this section.

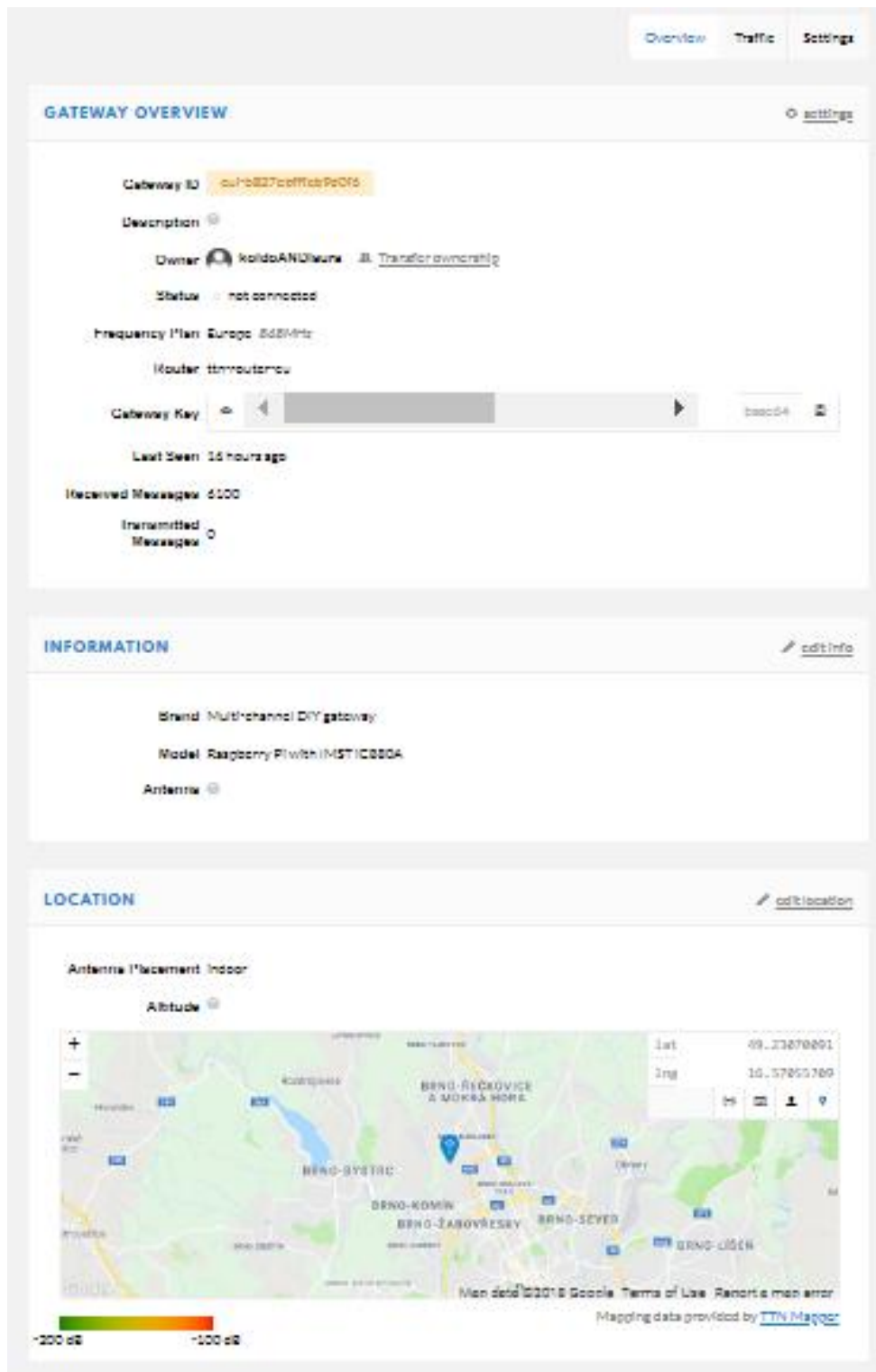


Figure 16: "Gateways" section overview.

The **gateway ID** is the unique identifier of each user's gateway. While the **gateway key** is the parameter that enables the network to connect the end device to the gateway once they are both registered in TTN.

## 7. Setting up the gateway

This chapter will explain how the gateway was constructed. This chapter has three sections; the hardware used, the software installed and how was made the registration in to TTN.

### 7.1. Hardware

As it was said in the section 4.4.1 the hardware needed to construct the gateway is the following:

- Raspberry Pi 3 model B.
- IMST IC880A.
- Antenna for 868MHz.
- Pigtail for the antenna.

The first step is choosing the backhaul. It refers to how the Raspberry Pi is going to be connected to the Internet. In the project this connection is done by Ethernet due to it is more stable than WiFi and it is less noisy.

The second step is to make the physical connections between the Raspberry Pi and the IMST IC880A. For that, 7 Dual female jumper wires are used. The connections are described in Table 10.

Table 10: Connections between IC880A and RPi.

IC880A pin	Description	RPI physical pin
21	Supply 5V	2
22	GND	6
13	Reset	22
14	SPI CLK	23
15	MISO	21
16	MOSI	19
17	NSS	24

### 7.2. Software

Firstly, The Raspberry has to be configured. For that, the OS Raspbian Stretch Lite has to be downloaded in a SD card. The SD card has to have at least 4 GB. Once the Raspberry has a OS, it is the time to configure the gateway with the following stages;

1. The Raspberry has to be powered up and a login and a password have to be introduced. The default values for Raspbian are; user=pi and password=raspberrypi.
2. To enter to the main menu of Raspbian and be able to change some parameters, this command must be executed:

```
$ sudo raspi-config
```

Parameters are changed to enable SPI ([5] Interfacing options -> P4 SPI) and expand the filesystem ([7] Advanced options -> A1 Expand filesystem)

Then, the system has to be rebooted.

- The next step is to configure the time zone and locales, the corresponding commands are:

```
$ sudo dpkg-reconfigure locales
$ sudo dpkg-reconfigure tzdata
```

- As Raspbian Stretch Lite is used, *git* has to be installed.

```
$ sudo apt-get install git
```

- A new user for TTN has to be created and add it to sudoers, also the user pi has to be deleted. This is shown in the next commands:

```
$ sudo adduser ttn
$ sudo adduser ttn sudo
```

```
$ sudo userdel -rf pi
```

- To continue, the installation of the gateways has to start, for that, TTN has created a script that install everything. This is done with the commands:

```
$ git clone -b spi https://github.com/ttn-zh/ic880a-gateway.git ~/ic880a-gateway
$ cd ~/ic880a-gateway
$ sudo ./install.sh spi
```

- The last step is to register it to TTN, for that, the MAC address of the concentrator is needed since it is going to be the EUI of the Gateway. In the project:

**Gateway ID** eui-b827ebfffeb9d0f6

### 7.3. Registration into TTN

From the console of TTN a new Gateway can be added. In the project, the Gateway has to be configured with TTN Packet Forwarded and as a result, the Gateway ID is going to be the MAC address of the concentrator.

In the registration the frequency has to be configured, 868 MHz due to Czech Republic is placed in Europe. Also it is needed; a description of the Gateway (Figure 17), the placement of the antenna and the location of the Gateway (Figure 18). When it is registered the status can be shown as it is shown in Figure 19.

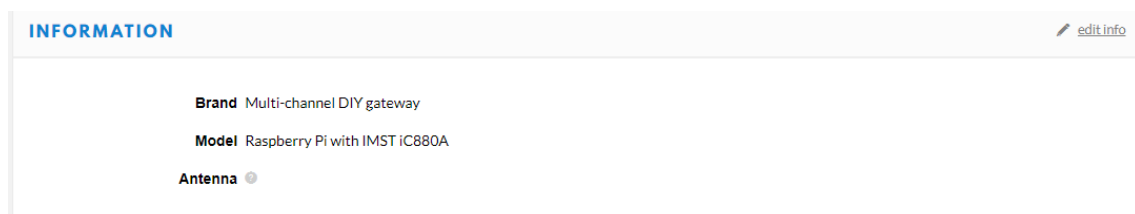


Figure 17: Description of the Gateway

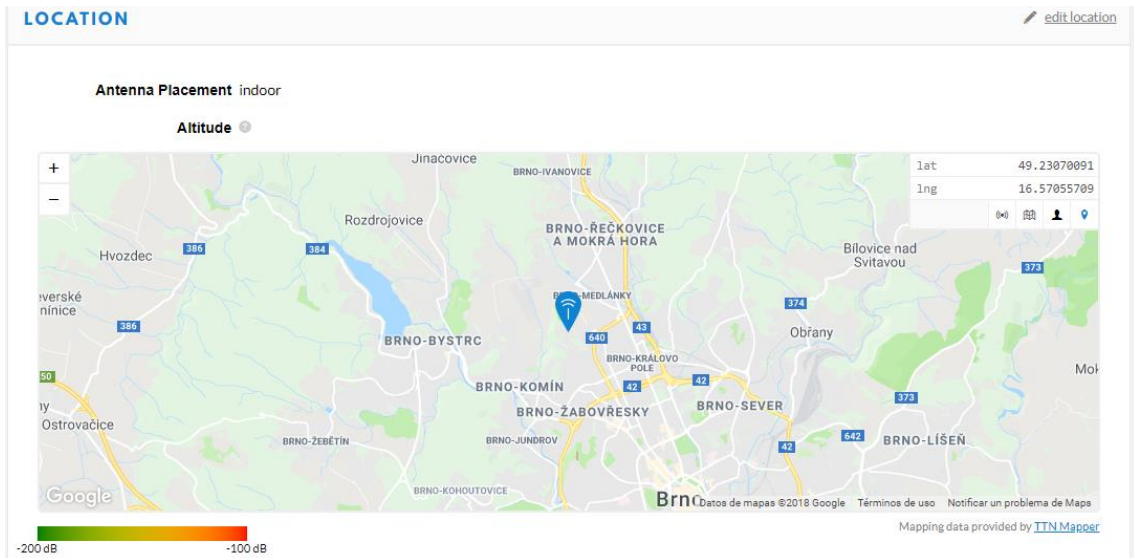


Figure 18: Location of the Gateway and placement of the antenna

For the location of the Gateway (Figure 18) the positioning parameters are obtained from Google Maps;

- Latitude: 49.23070091.
- Longitude: 1657055709.

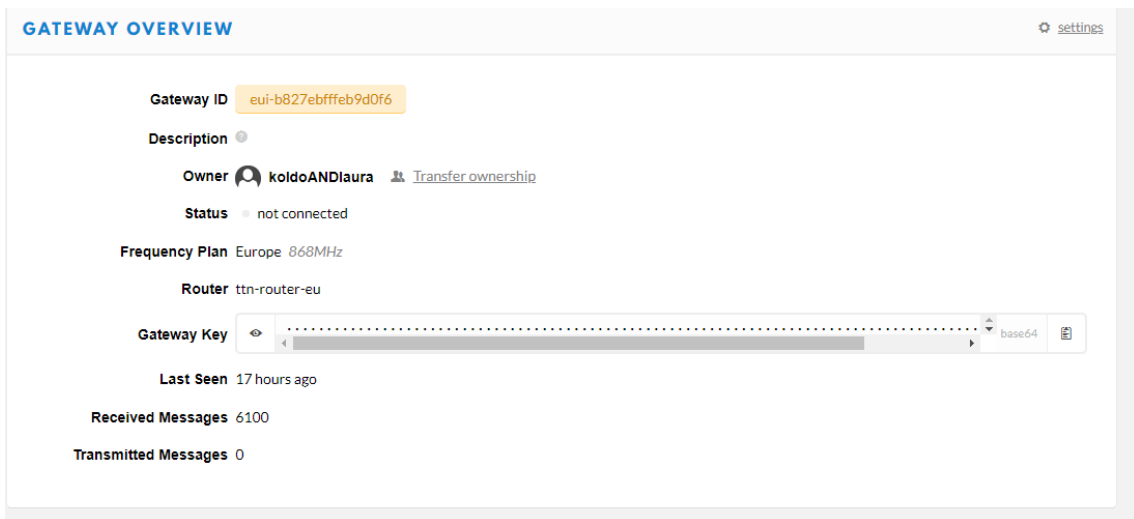


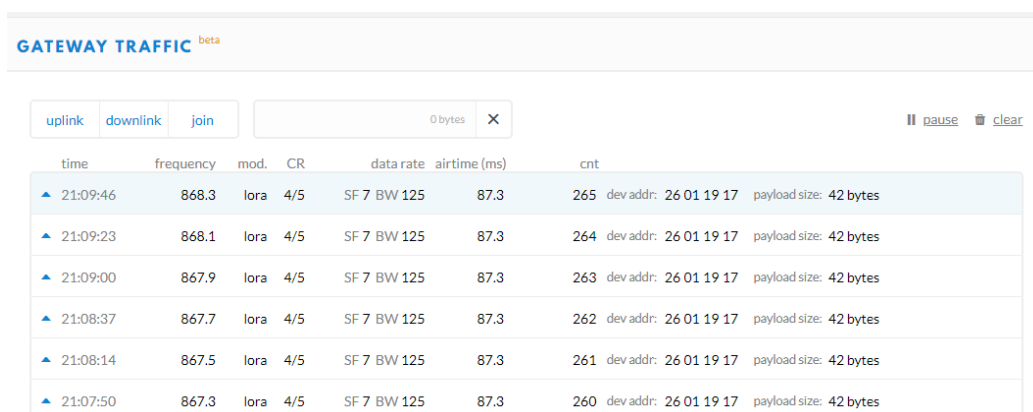
Figure 19: Gateway overview

## 8. Showing the data in TTN

First interaction of data from Arduino and the developer is done thanks to TTN’s console section. Here the sent data can be seen in two different parts; in the device’s part and on the gateway’s part, depending on what is registered in TTN. In the case of this project both are constructed and registered into the TTN’s network so information can be analysed in both sections.

### 8.1. Gateway section

Information about the packages received by the gateway is presented as is shown in Figure 20. The information is divided in several parts: **Time**, which refers to the time that the gateway has received the package. **Frequency**, which talks about the frequency used to transmit the received package, it is perceptible that the concentrator used in the project can receive data from different frequencies; that way the gateway is not collapsed. Type of **modulation** is also included; as it is obvious, LoRa is used. The **code rate** which is being used for transmit the data is 4/5. This high data rate is possible due to the proximity of the gateway to the end device. Data is transmitted with a **Spreading Factor** of 7 and a **Bandwidth** of 125kHz. This two parameters determine the **airtime** of the package which in this case is 87.3 ms. It is also possible to see the **frame counter** of the packages sent by the end node, the **address** of the end device and the **amount of bytes** sent it to that gateway. The end device is transmitting 30 bytes, to this; it must be added 12 bytes due to LoRaWAN.



The screenshot shows the 'GATEWAY TRAFFIC beta' interface. It has tabs for 'uplink', 'downlink', and 'join'. The 'uplink' tab is selected, showing a table of traffic data. The table has columns for time, frequency, mod., CR, data rate, airtime (ms), and cnt. The data rows show several packages received by the gateway, all with a payload size of 42 bytes.

time	frequency	mod.	CR	data rate	airtime (ms)	cnt
21:09:46	868.3	lora	4/5	SF 7 BW 125	87.3	265 dev addr: 26 01 19 17 payload size: 42 bytes
21:09:23	868.1	lora	4/5	SF 7 BW 125	87.3	264 dev addr: 26 01 19 17 payload size: 42 bytes
21:09:00	867.9	lora	4/5	SF 7 BW 125	87.3	263 dev addr: 26 01 19 17 payload size: 42 bytes
21:08:37	867.7	lora	4/5	SF 7 BW 125	87.3	262 dev addr: 26 01 19 17 payload size: 42 bytes
21:08:14	867.5	lora	4/5	SF 7 BW 125	87.3	261 dev addr: 26 01 19 17 payload size: 42 bytes
21:07:50	867.3	lora	4/5	SF 7 BW 125	87.3	260 dev addr: 26 01 19 17 payload size: 42 bytes

Figure 20: Gateway data traffic.

Regarding to the airtime parameter and taking into account the network usage fair policy of TTN which says that every day maximum airtime of all packages sent is thirty, the end device is able to send 343 packages per day, which will limit the number of sent packages.

### 8.2. Application section

In the application section the information sent by the end device is shown. It is given the **time** when the package was sent, the **number of the frame**, the **port** through it is transmitting, **the device ID** and the **payload bytes**.

The payload bytes has to be transformed into human understandable data. For that, a *payload format* is applied as shown in Figure 21.



**Payload**

```
0A 96 13 7E FF FD 00 00 00 00 47 00 00 07 01 00 00 00 00 9A 01 00 00 01 00 00 01 00 1F
```

**Fields**

```
{
  "Distance1": 71,
  "Distance2": 7,
  "Distance3": 0,
  "Distance4": 154,
  "Distance5": 0,
  "Distance6": 0,
  "DustDensity": 655.33,
  "Humidity": 49.9,
  "Luminosity": 31,
  "LuzOn": 1,
  "Parking1": 0,
  "Parking2": 0,
  "Parking3": 1,
  "Parking4": 0,
  "Parking5": 1,
  "Parking6": 1,
  "Polluted": 0,
  "Sonido": 0,
  "Temperature": 27.1
}
```

Figure 21: Payload format example.

As it is perceptible, the string is transformed from hexadecimal values to labeled decimal values. It is important to format the data in the proper way as it is how it will be stored in the data base integration. For formatting this is important to know how the end device organizes the data string to send the information of the sensors.

The string sent from the end device to the gateway contains 29 positions which are explained in Table 11. Although there is data in the string not represented in the web application, it is convenient to send it to know if the interpretation of the data is correct or not.

Table 11: Description of the payload sent.

Position of the array of data	Data from the sensors
0	8 MSB of temperature
1	8 LSB of temperature
2	8 MSB of humidity
3	8 LSB of humidity
4	8 MSB of dust density
5	8 LSB of dust density
6	8 bits of air pollute or not
7	8 bits of sound overcome
8	8 bits of park slot 1
9	8 MSB of park slot 1 distance
10	8 LSB of park slot 1 distance
11	8 bits of park slot 2
12	8 MSB of park slot 2 distance
13	8 LSB of park slot 2 distance
14	8 bits of park slot 3
15	8 MSB of park slot 3 distance
16	8 LSB of park slot 3 distance
17	8 bits of park slot 4
18	8 MSB of park slot 4 distance
19	8 LSB of park slot 4 distance
20	8 bits of park slot 5
21	8 MSB of park slot 5 distance
22	8 LSB of park slot 5 distance
23	8 bits of park slot 6
24	8 MSB of park slot 6 distance
25	8 LSB of park slot 6 distance
26	8 bits "streetlights are on?"
27	8 MSB of amount of light
28	8 LSB of amount of light

## 9. User application

In this chapter, the development of the user’s final application is explained. First, the integration in TTN is explained which defines the way the information is exported. It is also explained the different servers used to complete the whole application as they are the data base server and the server where the website is hosted. This chapter is focused on describing how the website looks like, explaining the content of each page.

Once the information is uploaded to TTN server, to create the final application for the user of the smart street, the data is exported by using an API. TTN calls “integrations” to the APIs they have developed; it has different ways to show the data externally. Integrations are the easiest way to connect with the devices registered in TTN to an application. An integration uses the same APIs or SDKs an application could use directly. Together with the private or public APIs of the platform, it ties up the application running on the platform with The Things Network [1].

It is possible to add any of the integrations shown in Figure 22. In this project, “data storage” integration is used; this API allows the user to query data and list devices for which data is stored by the Storage Integration. The API is available over HTTPS and serves data in JSON format.



Figure 22: APIs available in TTN to export data [6].

The data access is done by going to the Swagger server’s interface, the base URL for the storage API is: <https://<app-id>.data.thethingsnetwork.org/>, which redirects to the site shown in Figure 23. This site allows the user to do different callings to the base URL depending on the information they would like to receive from the database. In the case of this project it is used query function explained in section 9.1.

The screenshot shows the Swagger API interface for 'The Things Network Data Storage'. At the top, there is a green header with the Swagger logo, the API URL 'https://1390307890998876.data.thethingsnetwork.org/swagger', and 'Authorize' and 'Explore' buttons. Below the header, the title 'The Things Network Data Storage' is displayed, followed by the description 'Stores data and makes it available using a REST API'. The creator information 'Created by The Things Industries B.V.' and a link to 'https://www.thethingsindustries.com' are also present. The main content area lists three endpoints:

- devices**: GET /api/v2/devices. Description: Query the devices for which data has been stored.
- query**: GET /api/v2/query. Description: Query data.
- query/{device-id}**: GET /api/v2/query/{device-id}. Description: Query data for a specific device.

At the bottom, there is a status bar showing '[ BASE URL: , API VERSION: 2.0.0 ]' and a 'VALID' button with a Swagger logo.

Figure 23: User interface of swagger data storage API.

By using *query* function, the data stored in the server is accessed and by using a php script this data is used and managed in a website; the data is saved in JSON format.

For the user’s interface, a web application was created. The website is hosted by the domain *antolinez.org* and it works under the subdomain *smartstreet.antolinez.org*. The site points to a server supported by Plesk web host edition which allows to manage the documents and folders added to the server which contain information about the website. In Figure 24 root folder of the website is introduced. It is possible to see how the site is organized; it has a *.php* document for every page of the site and different folders to add style with *.html* and *.css* documents and downloadable documents to the site. The documents in *.html*, *.css* etc. are usually called in *.php* documents as they are giving overall format to the site so that it stays the same no matter what is the opened page.

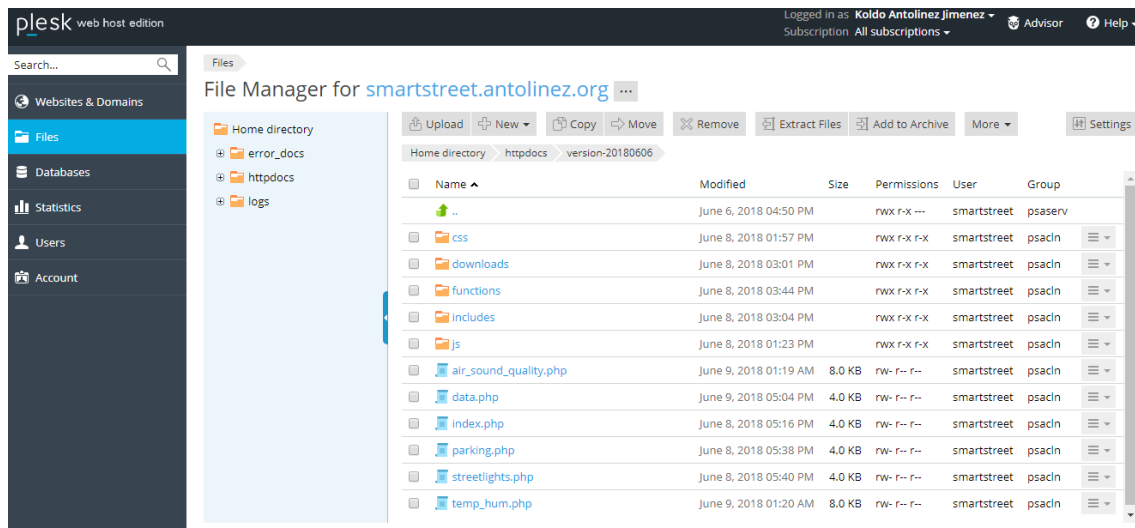


Figure 24: Root folder of website server in Plesk web host edition.

### 9.1. User Interface

The user interface is divided into five different pages: Home page (depicted in Figure 25), Temperature and humidity page, parking page, air and sound quality page and streetlight page. In each page. All the pages follow the same structure; they all have the same header and footer. The header is the green part on the top of the page and it contains the buttons to travel through all the pages in the site before mentioned. The footer is the black part on the bottom of the page and it has information about the copyright and creators of the site.

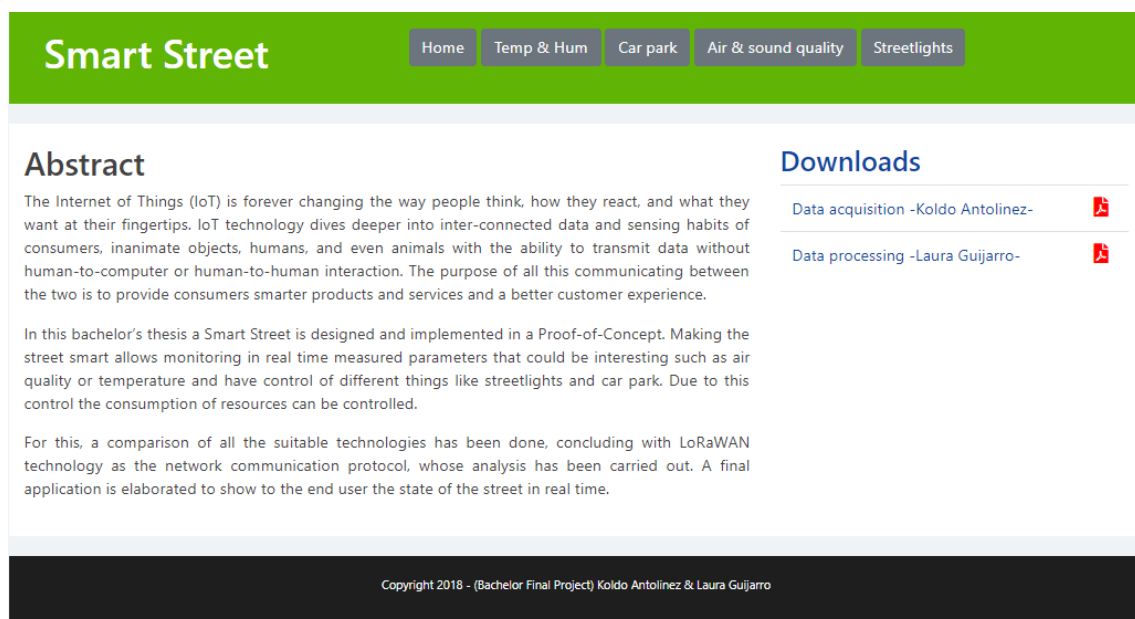


Figure 25: Preview of final application's home page [8].

#### 9.1.1. Home page

The home page is divided into two parts, on the left side of the page the project is introduced via the abstract of it. There is explained the aim of the project and an overview of how it is going to be developed. On the part downloads, on the right side of the page, the downloadable link to project's report is included. There are a report that focuses on the

development of the end device (data acquisition) and another one that explains the communication channel and talks about how creating the gateway and connecting it to TTN.

### 9.1.2. Temp & Hum page

In this page information about the temperature and humidity sensor is shown in a graph each. In this case, the page is divided in two equal parts as shown in Figure 26.

The graph on the left shows the evolution of the temperature for the last 160 lectures of the sensor while the one in the right shows the evolution of the humidity during the last 160 lectures of the sensor. They both depend on the sensor DHT22 which measures temperature and humidity.

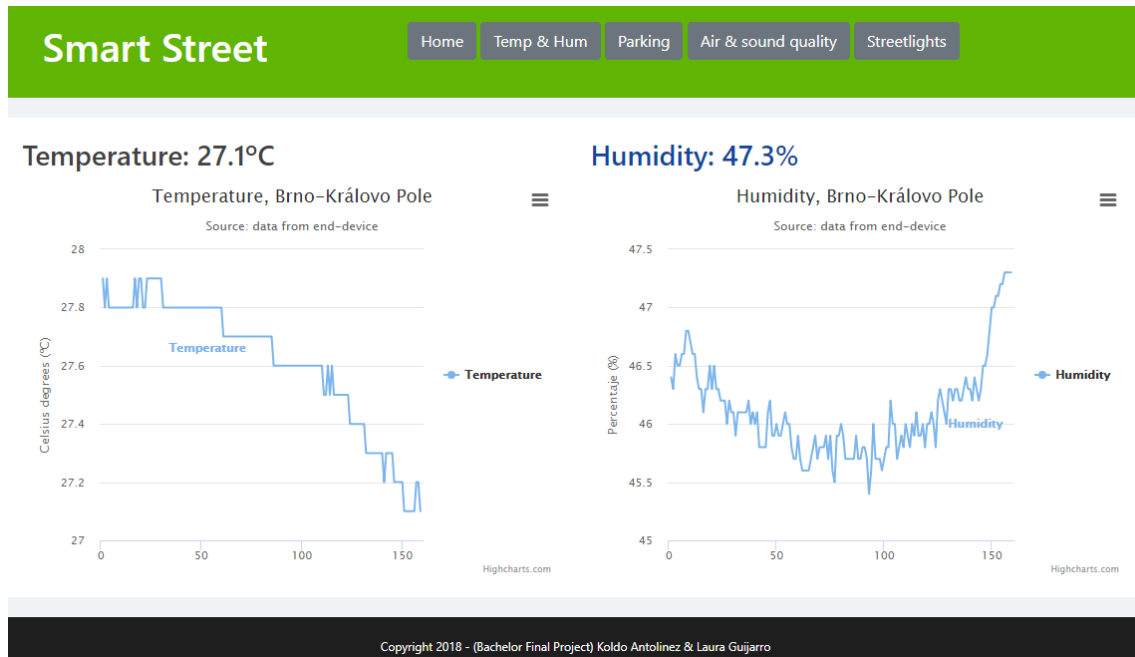


Figure 26: Preview of final application's Temp & Hum page [8].

### 9.1.3. Parking page

In this page information about the car park is shown. To do so, a car park has been simulated showing two rows of three park slots. In each slot a car is pictured and it will change the color depending on if it is free or occupied as it can be appreciated in Figure 27.

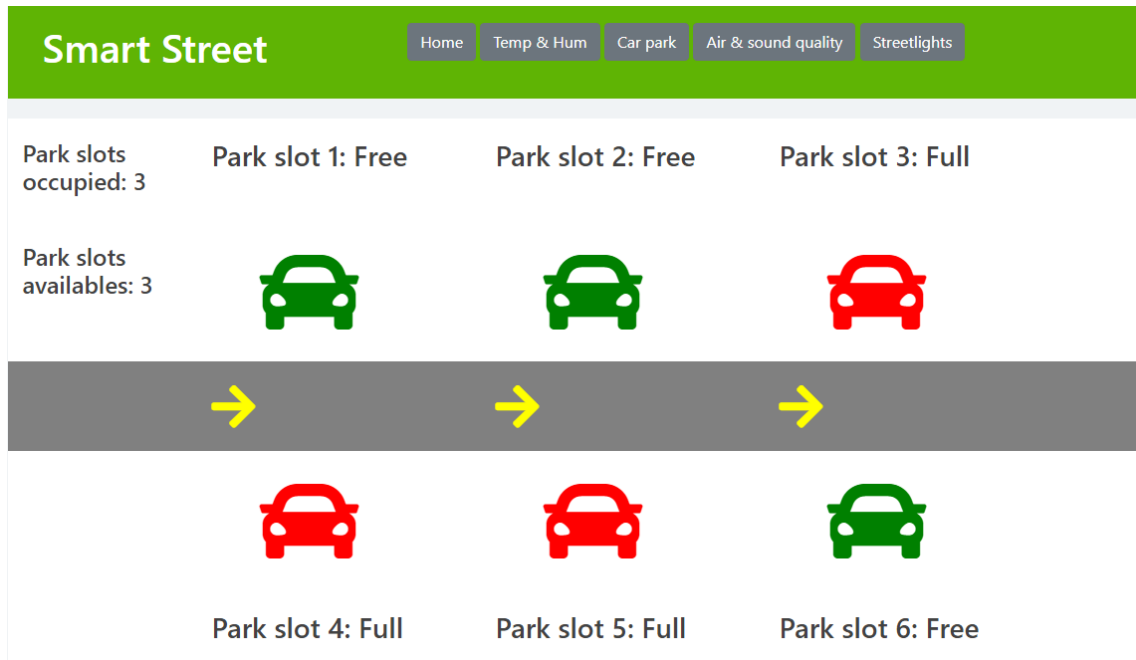


Figure 27: Preview of final application's Car park page [8].

#### 9.1.4. Air & sound quality page

In this page information about the air and sound quality in the neighborhood is shown. As it can be seen in Figure 28, the page template is the same as the one used for “Temp & Hum” page. It also shows two graphs; the one on the left shows the evolution of the dust density in the air in  $\mu\text{g}/\text{m}^3$  and the one in the right tells whether the sound limit has been overcome or not and in what moment that happened.

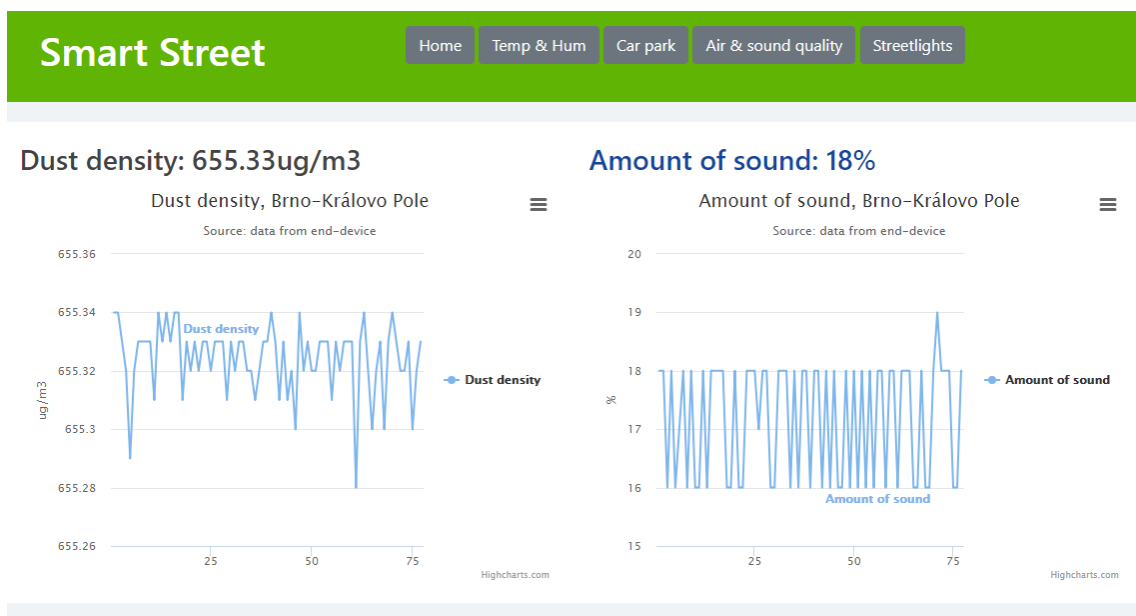


Figure 28: Preview of final application's Air & sound quality page [8].

#### 9.1.5. Streetlights page

In this page information about the streetlight’s state is presented. The page is divided in two parts as can be seen in Figure 29; on the left part a graph is shown where the amount of light’s

evolution is perceptible, while on the right it is presented whether the lights are ON or OFF with the light bulb's picture.

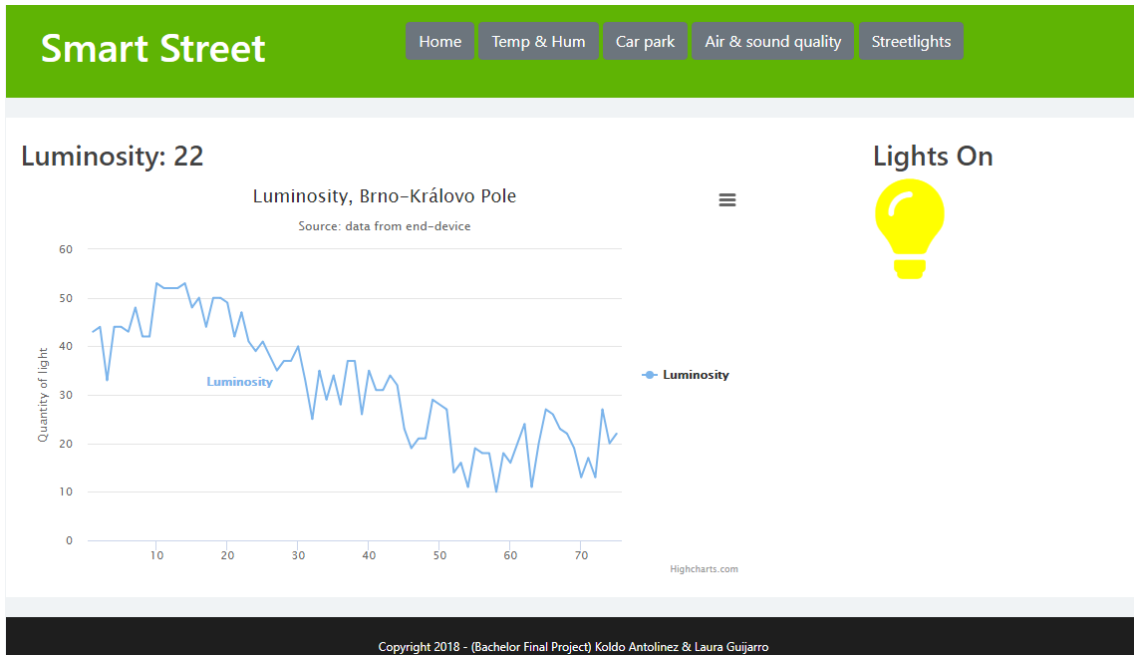


Figure 29: Preview of final application's Streetlights page [8].



## 10. Conclusions

In this final chapter, the conclusions of the project will be presented.

### 10.1. LoRa

As it was expected LoRa works well for the purpose of the project, giving a long range communication link. The constructed gateway can cover the entire city of Brno supporting hundreds of nodes, a breakthrough compared with any other standardized communication technology. An entire network can be created using a minimal amount of infrastructure.

### 10.2. LoRaWAN

Due to the architecture of LoRaWAN network, other end-nodes can be added easily for increasing the amount of measured data. This gives scalability to the project. Also, in the case that a future end- device will be moving, for example a car tracing application, a handover would not be necessary from gateway to gateway because in a LoRaWAN network nodes are not associated with a specific gateway; data transmitted by a node is received by multiple gateways. This gives to LoRaWAN a very high potential.

### 10.3. The Things Network

TTN allows the users to make use of their network structure, so that only the gateway and an end-device are needed in order to have a fully operational LoRaWAN network. The interface of TTN is very understandable and easy to use, that gives a great knowledge of all the options that you can use. TTN has a lot of integrations which makes possible a lot of different applications.

### 10.4. Gateway

The gateway has been a great success; it is able to receive packets from different frequencies with different data rates and spreading factors. It was implemented with a Raspberry Pi and an IC880A concentrator; whose price is much lower if it is compared with a gateway that is already done. TTN facilitates the connection of new gateways to their network, making the registration of them in a simple and intuitive way.

### 10.5. End-device

The end-device is able to send all the measured data from the sensors. The interval in which the information is sent can be selected as well as the spreading factor and the frequency in which it is sent. To be able to use LoRaWAN technology the end-device it is constructed with the Arduino Due and a Dragino LoRa shield which is one of the cheapest options.

The problem that the end-device has is that due to the large number of sensors incorporated in the Arduino Due, the manipulation and connection of them are very difficult.

### 10.6. Application

The application shows properly the measured data from the sensors, giving graphs and styling format to it, which helps the user to easily understand the data shown [7].

## 11. References

- [1] K. Shaw, «Network world,» IDG, 30 01 2018. Available: <https://www.networkworld.com/article/3238664/wi-fi/80211-wi-fi-standards-and-speeds-explained.html>. [Accessed: 12 June 2018].
- [2] Semtech, «LoRa Modulation Basics,» 2015.
- [3] L. Alliance, «LoRaWAN 1.1 Specification,» 2017.
- [4] T. T. Network, «The Things Network,». Available: <https://www.thethingsnetwork.org>.
- [5] Brocaar, "Lora server," . Available: <https://forum.loraserver.io/t/application-eui/34>.
- [6] T. T. Network, «Integrations: The Things Network,» The Things Network. Available: <https://console.thethingsnetwork.org/applications/1390307890998876/integrations/create>. [Accessed: 05 June 2018].
- [7] w3schools, «w3schools.com,». Available: <https://w3schools.com>. [Accessed: 08 06 2018].
- [8] K. Antolinez Jimenez y L. Guijarro Iguacel, «Smart street,» 09 June 2018. Available: <https://smartstreet.antolinez.org>. [Accessed: 09 June 2018].
- [9] HighCharts, «HighCharts,». Available: <https://www.highcharts.com>.
- [10] Dragino, «Wiki Dragino,». Available: [http://wiki.dragino.com/index.php?title=Lora\\_Shield](http://wiki.dragino.com/index.php?title=Lora_Shield). [Accessed: 23 March 2018].