POLITECNICO DI TORINO

MASTER's Degree in ELECTRONICS ENGINEERING

MASTER's Degree Thesis

Exploiting LoRaWAN for IoT air pollution devices in urban areas

Supervisors

Prof. BARTOLOMEO MONTRUCCHIO

Prof. EDOARDO GIUSTO

Prof. MARIE-SANDRINE DENIS

Candidate

NAOURAS LATIRI

DECEMBER 2022

Summary

Air pollution is a major contributor to global warming, and efforts are being made to address this issue.
Monitoring air quality can be important for giving reliable information to assist in taking activities to enhance air quality.
The aim is to reduce health-threatening dangers and promote awareness about the impacts of air pollution exposure.

The purpose of this thesis is to investigate the functionality and performance of an environmental system, which includes air pollution sensors, specifically PM2.5 and PM10 sensors, temperature and humidity sensors, Internet of Things (IoT) communication protocols, and data acquisition and transmission via communication channels.
This low-cost, low-power node proved to be an excellent substitute for more expensive devices.

Acknowledgements

I would like to acknowledge and give my warmest thanks to all my supervisors who made this work possible and especially Edoardo Giusto and Gustavo Ramírez Espinosa for their guidance and advice that carried me through all the stages of my project.

I am also grateful to my family, my mother, my father, and my two brothers. Their confidence in me has maintained my spirits and motivation strong throughout the process. It would have been a much more difficult feat without their support.

I would also like to thank my cats for providing me with fun and emotional support.

This would not have been possible without you.

Table of Contents

List of Tables

List of Figures

Acronyms

LoRa
: Long Range

LoRaWAN
: Long Range Wide Area Network

LPWAN
: Low Power Wide Area Networks

WLAN
: Wide Local Area Networks

IoT
: Internet of Things

ISM
: Industrial, Scientific, and Medical

NB-IOT
: Narrowband IoT

LTE
: Long-Term Evolution

GSM
: Global System for Mobile communication

LAN
: Local Area Network

RF

Radio Frequence

bps

Bits per second

GPS

Global Positioning System

PM

Particulate Matter

LBT

Listen before talk

GFSK

Gaussian Frequency Shift Keying

RSSI

Received Signal Strength Indicator

SNR

Signal to Noise Ratio

ADR

Adaptive Data Rate

TOA

Time On Air

DR

Data Rate

CR

Coding Rate

SF

Spreading Factor

TTN

The Things Network

NS

Network Server

Tx

Transmission Power

PDR

Packet Delivery Ratio

LOS

Line of Sight

NLOS

Non Line of Sight

BW

Bandwidth

BPSK

Binary Phase Shift Keying

CSS

Chirp spread spectrum

QoS

Quality of service

CRC

Cyclic Redundancy Check

OTAA

Over-the-Air Activation

ABP

Activation by Personalization

MAC

Media Acess Control

LNS

LoRaWAN network server

PHDR

physical header

PHypayload

physical payload

MIC

Cryptographic Message Integrity

PHDRCRC

Cyclic redundancy check of the physical header

MHDR

MAC Header

Mtype

Message Type

FHDR

Frame Header

FPort

Port Field

FRMPayload

Frame Payload

DevAdress

address of the end device

FCtrl

Frame Control

ACK

Acknowledgement

NwkSKey

Network Session Key

AppSKey

Application Session Key

AppKey

Application Key

Chapter 1

Introdution

Nowadays, there are numerous sources of air pollution, including vehicular traffic, factories, and so on.
Because the high accuracy expected by the sensors increases both their cost and dimensions, air quality sensors that meet legal standards are typically collected inside static environmental monitoring stations. A monitoring station costs between $10,000 and $20,000. As a result, the stations cannot be placed in all locations. Therefore, a low-cost system is required.[1].
The system must acquire information such as Particulate Matter (2.5g/m3 and 10 g/m3), Temperature (C), Relative Humidity (%), and Pressure.
Particulate Matter (PM) sensing devices are low-cost devices that can produce beneficial records similar to the output of an expert and calibrated sensor.
New technology is needed to create a cheap and efficient system that can transmit measurement data over long distances and with low energy consumption. Thus, the use of Low Power Wide Area Networks (LPWAN)technology with LoRa allows the transmission of low-density data over long distances with low energy consumption. The LoRaWAN protocol was developed for the wireless connection of battery-powered devices to the Internet. LoRaWAN provides data security through authentication between nodes and the network server and end-to-end message encryption.

Chapter 2

Low Power Wide Area Network

Low Power Wide Area Networks (LPWAN) are a cutting-edge paradigm for communication that will function in conjunction with current cellular and short-range wireless technologies to meet the varied needs of Internet of Things (IoT) applications.
Wide-area connectivity for low-power, low-data-rate devices is one of the distinctive qualities that LPWAN technologies offer in contrast to older wireless technologies. Short-range wireless networks such as ZigBee and Bluetooth, legacy Wide Local Area Networks (WLAN) such as WiFi, and cellular networks such as the Global System for Mobile Communications (GSM), Long-Term Evolution (LTE)[2], and others offer different tradeoffs than LPWAN networks, which are distinctive in that they do not.
Older wireless methods are not ideal for connecting low-power devices spread over large areas. These technologies have a maximum range of several hundred meters. Therefore, the devices cannot be placed or moved randomly, which is necessary for many applications in logistics, personal health, and smart cities.[3].

2.1 Famous LPWAN Technologies

In both the permitted and unlicensed frequency bands, various LPWAN systems have arisen.
The top three LPWAN technologies competing for large-scale IoT deployment are Sigfox, NB-IoT, and LoRa.

2.1.1 Sigfox

Sigfox is an LPWAN network operator that offers an end-to-end IoT connectivity solution based on its patented technologies[4].
The cognitive radio base stations that Sigfox installs are connected to the back-end servers by an IP-based network.
End devices connected to these base stations use Binary Phase Shift Keying (BPSK) modulation on a sub-GHz ISM band carrier with an ultra-narrow band (100 Hz). Unlicensed ISM bands, such as 433 MHz in Asia, 868 MHz in Europe, and 915 MHz in North America, are used by Sigfox. This technology employs the ultra-narrow band to optimize frequency spectrum utilization and achieve very low noise levels, resulting in very low power consumption, high receiver sensitivity, and low-cost antenna design at the expense of a maximum throughput of only 100 Bits per second (bps)[5].
After initially supporting only uplink communications, Sigfox was developed as a bidirectional technology with strong link asymmetry.[3].
Downlink communication, which transfers data from base stations to terminals, can occur only after uplink communication and is limited to four per day, preventing the base station from acknowledging every uplink message[6].
The key information of Sigfox technology is shown in Table2.1

Number of messages over the uplink	140 message/day
Maximum payload length for every uplink messages	12 bytes
Maximum payload length for every downlink messages	8 bytes
Maximum throughput	100 bps

Table 2.1: key information of Sigfox technology[7].

2.1.2 NarrowBand IoT (NB-IoT)

Narrowband IoT (NB-IOT) technology coexists with Long-Term Evolution (LTE) and Global System for Mobile communication (GSM) in licensed frequency ranges. The key features of NB-IOT include[8]:

1. deployment with very low bandwidth.
2. more extensive coverage than current cellular networks.
3. Maximum terminal battery life (10 years).
4. support for large connections (50K devices).
5. Designed for incredibly low terminal cost.

NB-IOT should be able to function in three different modes:

1. Stand-alone: Use of a scattered spectrum as well as, for example, spectrum currently used by other systems to replace one or more GSM carriers[8].
2. Guard band: use of unused resource blocks within the guard band of an LTE operator[9].
3. Utilization of resource blocks within a typical LTE carrier (in-band)[10].

2.1.3 LoRa

LoRa is a physical layer technology that modulates signals in the sub-GHz ISM band using a proprietary spread spectrum technique.
Like Sigfox, LoRa also uses unlicensed ISM bands, i.e., 433 MHz in Asia, 868 MHz in Europe, and 915 MHz in North America[5].
Chirp spread spectrum (CSS) modulation, in which a narrow-band signal is spread over a wider channel bandwidth, enables bidirectional communication. The resulting signal has low noise, is highly resistant to interference, and is difficult to detect[5].
LoRa employs six spreading factors (SF7–SF12) to balance the trade-off between data rate and range. A higher spreading factor allows for a wider range at the cost of a lower data rate and vice versa.
LoRa data rates range from 300bps to 50 kbps, depending on the spreading factor and channel bandwidth. In addition, messages transmitted with different spreading factors can be received simultaneously by LoRa base stations. Each message has a

maximum payload length of 243 bytes[11].
The LoRa Alliance has standardized a LoRa-based communication protocol called LoRaWAN. Each message transmitted by a terminal is received by all base stations in range[4].
LoRaWAN uses redundant reception to increase the rate of correctly received messages. However, achieving this functionality necessitates a large number of base stations in the area, which can raise the cost of establishing the network[5]. Duplicate reception is checked in the Network Server's Network Server (NS) back-end system, which also checks security, sends acknowledgments to the end device, and forwards the message to the appropriate application server. Furthermore, LoRaWAN supports multiple End Device classes in order to meet the various requirements of a wide range of IoT applications, such as latency requirements[5].

2.2 LPWAN Comparison

Many factors should be considered when selecting an LPWAN technology for an IoT application, including quality of service, battery life, latency, scalability, payload length, coverage, range, deployment, and cost[5].
. Below is a comparison of Sigfox, LoRa, and NB -IoT with respect to these variables and technology differences[4].

2.2.1 Quality of service (QoS)

Sigfox and LoRa use unlicensed spectrum and asynchronous communication protocols and are able to bridge interference, multipath, and fading. However, they cannot provide the same QoS as NB-IoT[12]. NB-IoT employs a licensed spectrum and an LTE-based synchronous protocol, which are better for QoS but more expensive. Because of the trade-off between QoS and high spectrum costs, applications requiring QoS should use NB-IoT, while applications requiring low QoS should use LoRa or Sigfox[13].

2.2.2 Battery life and latency

End devices in Sigfox, LoRa, and NB-IoT remain in sleep mode most of the time outside of operation, which reduces the amount of spent energy, resulting in a long end-device lifetime[4].
The NB-IoT end device, on the other hand, consumes more energy due to the use of time-synchronized communication and QoS processing[12].
The increased energy consumption reduces the lifetime of NB-IoT end devices. LoRa, unlike Sigfox, has low bidirectional latency at the expense of higher energy

consumption[5].
Therefore, Sigfox and Class-A LoRa are the ideal solutions for applications where latency is not a concern and only tiny amounts of data must be transmitted.
NB-IoT and Class-C LoRa are better solutions for applications that require low latency[14].

2.2.3 Scalability and Payload length

Several technologies are being developed to address scalability issues, such as the efficient utilization of diversity in a channel, as well as in time and space.
However, NB-IoT has far greater scalability than Sigfox and LoRa. This technology supports up to 100K end devices per cell, compared to 50K for Sigfox and LoRa[5].
NB-IoT also has the advantage of allowing for maximum payload length, as it can transmit up to 1600 bytes of data. LoRa allows for data transmission of up to 243 bytes. Sigfox, on the other hand, has the shortest payload length of 12 bytes, limiting its use in many IoT applications that require large amounts of data[5].

2.2.4 Network coverage and Range

The main advantage of using Sigfox is that a single base station can cover an entire city (i.e., range $>$ 40 km).
The Sigfox network deployment in Belgium, a country with a total area of approximately 30500 km2, covers the entire country with only seven base stations.
LoRa, on the other hand, has a shorter range (i.e., 20 km), requiring only three base stations to cover an entire metropolis such as Barcelona. The most limited range and coverage are provided by NB-IoT. (i.e., range 10 km)[5].
Furthermore, NB-IoT deployment is limited to LTE base stations. As a result, it is unsuitable for rural or suburban areas with limited LTE coverage.

2.2.5 Deployment model

NB-IoT can be deployed in three ways by reusing and upgrading the existing cellular network: in stand-alone operation, guard-band operation, and in-band operation[13]. Sigfox and LoRa ecosystems are already in commercial use in a number of countries and locations. In addition, a key advantage of the LoRa ecosystem is its flexibility. Unlike Sigfox and NB-IoT, LoRa supports both the deployment of Local Area Network (LAN) via a LoRa gateway and the operation of public networks via base stations[5]. A hybrid operating model could be used in the industrial sector to deploy a local LoRa network in production areas while using the public LoRa network[4] to cover outdoor areas2.1.

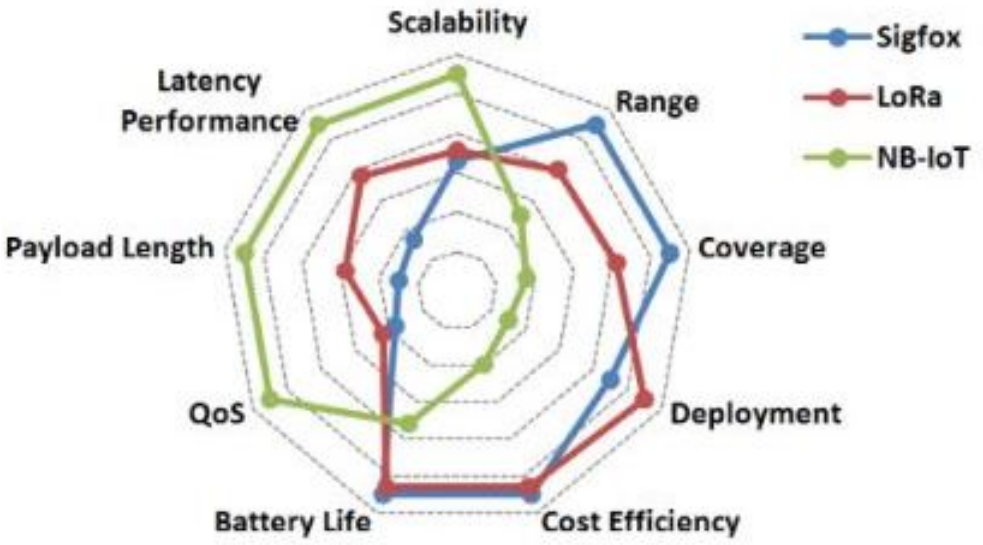

Figure 2.1: Respective advantages of Sigfox, LoRa, and NB-IoT in terms of IoT factors[4].

2.2.6 Cost

There are several cost factors to consider, including spectrum, network, device, and deployment costs. Sigfox (> 4000€/base station) and LoRa (> 100€/gateway > 1000€/base station) are clearly less expensive than NB-IoT (> 15 000€/base station).

Chapter 3

LoRa and LoRaWAN Overview

3.1 LoRa Technology

LoRa is a Low-power, wide-area network RF modulation technology. The name LoRa refers to the extremely long-distance data links enabled by this technology. LoRa, which was developed by Semtech to standardize LPWANs, enables long-distance communications of up to 5 kilometers in urban areas and up to 15 kilometers or more in rural areas[15].
The ultra-low power requirements of LoRa-based solutions enable the development of battery-powered devices that can last for up to ten years. A network based on the open LoRaWAN protocol, deployed in a star topology, is ideal for applications that require long-distance communication among a large number of low-power devices that collect small amounts of data[15].
LoRa is divided between two layers:

1. Physical Layer: To communicate between end devices and gateways, the Chirp spread spectrum (CSS) radio modulation technique is used. The LoRa physical layer is a Semtech-developed private technology[8].

2. MAC Layer: LoRaWAN is a multicast communication that employs the LoRa MAC layer protocol. It defines the rules that radio waves must follow in order to gain access to the LoRaWAN gateway and perform channel operations. LoRaWAN is a LoRa-Alliance-developed open standard[8].

3.1.1 LoRa Physical Layer

Cycleo developed the LoRa technology, which was later acquired and patented by Semtech. LoRa operates in Industrial, Scientific, and Medical (ISM) bands which span 169, 433, 868 MHz in Europe and 915 MHz in the United States.
The European standards[16] specified a duty cycle ranging from 0.1% to 1% depending on the used sub-band to limit interferences. Unlike other IoT technologies, LoRa network management is open.
While other IoT technologies are proprietary, LoRa network management is open[17].

3.1.2 LoRa Parameters

The LoRa transmission distance can be influenced by selecting four main parameters with a noticeable effect on the LoRa signals.
These parameters are listed below:

Spreading Factor (SF)

In LoRa, the spreading factor is a value that determines how far the chirp will spread. The number of bits packed into a single chirp is primarily responsible for this degree of spreading[18]. For instance, SF7 denotes that each chirp is worth seven bits. The LoRa chirp is modulated by adjusting the start/end frequency of each chirp, there must be 2^{SF} positions for each start/end frequency[19].
A chirp has to sweep through a certain bandwidth and its time duration can be determined by the following equation 1:

$$T_{sym} = \frac{2^{SF}}{BW} \tag{1}$$

A chirp is created by iterating through all possible f_{SF}, yielding a number of 2^{SF} chips. If all other parameters remain constant, each increase in SF doubles the time required to transmit a chirp[18].
By doubling the chirp duration, the receivers have more opportunities to sample the signal power, resulting in a higher Signal to Noise Ratio (SNR). Since the definition of SNR is $\frac{P_{Signal}}{P_{Noise}}$, the higher the signal power P_{Signal} compared to the noise power P_{Noise}, the higher the probability that each chirp will be received correctly.

Bandwidth (BW)

According to equation 1, the bandwidth determines the width of the transmitted signal, as well as the duration of the chirp. Changing the BW affects the chip

duration, which in turn affects the SNR of that particular chirp.
LoRa gateway chipsets are optimized for 125kHz transmissions, with the ability to receive transmissions at a fixed SF with varying BW (250 and 500kHz) and GFSK transmissions. Changing the BW would change the chip duration accordingly, which in turn affects the SNR of that particular chirp.
LoRa gateway chipsets are optimized to receive transmissions at 125kHz, with the additional capability to receive transmissions at a fixed SF with varying BW (250 and 500kHz) and GFSK transmissions.

Transmission Power (Tx)

Transmission power directly affects the amount of energy used to transmit a chirp. By increasing TX Pow, the signal has a better chance of surviving the attenuation caused by the environment, which effectively increases the signal power P_{Signal} received by the receivers[18].

Coding Rate (CR)

Code Rate refers to the forward error correction (FEC) added to a packet before transmission. LoRa utilizes Hamming Code as FEC in CR.
A CR of 4/5 means that one bit of correction code is added to every four bits of data[18]. In LoRaWAN, CR equals $\frac{4}{4+n}$, with $n \in 1,2,3,4$.
The coding rate increases the overhead in transmission by increasing the number of bits to be transmitted. This overhead allows the receiver to verify the correctness of the received chirps and provides the opportunity to correct some erroneous bits from a chirp [20].

3.1.3 LoRa Characteristics

LoRa technology has various characteristics that make it a popular technology.

1. Ultra-long distance: In Line of Sight (LOS) communications, the longest SF12 can reach a distance of up to 5 km and the smallest SF7 can reach a distance of 2 km. The longest distance reached in of 2 km in Line of Sight (LOS) communications. The longest distance achieved in communications with structures is less than 2 kilometers. The communication distance is affected by factors such as bandwidth, SF, transmission power, and coding rate.

2. Low cost and complexity: LoRa devices are designed to be simple and inexpensive to manufacture. The end devices operate in such a way that they do not use the complex Listen before talk Listen before talk (LBT) protocol but instead begin transmitting when necessary. As a result, only the ALOHA

system is used.
The pure ALOHA system is a random access protocol where the end device transmits whenever it has data to send to the gateway, at any time. In this system, the collided frames are destroyed. In LoRa, the probability of collisions is low due to the duty cycle being limited to 1% [8].

3. Insensitivity to the Doppler effect: LoRa transmissions are highly resistant to the Doppler effect due to the CSS modulation technique. Moving LoRa terminals operating in LOS and at a constant speed can achieve a Packet Delivery Ratio (PDR) of more than 85%.

3.1.4 LoRa Packet Format

There are two modes of LoRa packet format:
- The explicit header mode includes a short header that contains information about the length of the payload, the coding rate, and whether the packet uses a Cyclic Redundancy Check (CRC)[21].

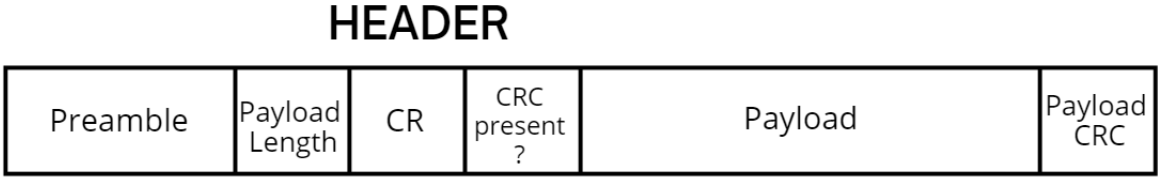

Figure 3.1: Explicit LoRa Packet Format.

- The packet's header is removed, which reduces transmission time. Both sides of the radio link must manually configure the payload length, error coding rate, and presence of the payload CRC[21].

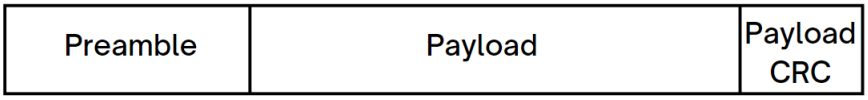

Figure 3.2: Explicit LoRa Packet Format.

In both types of LoRa packet format, the preamble is used by the receiver to detect the start of the packet, and the payload is a variable-length field that contains the actual data coded at the forward error correction code rate, either as specified in the header in explicit mode or fixed in implicit mode.
An optional payload CRC may be appended.

3.1.5 LoRa End Device Activation Methods

Devices must be commissioned and activated on the network during startup to ensure security and quality of service.
During commissioning, the network and each device are securely aligned in terms of critical deployment parameters like identifiers, encryption keys, and server locations. The LoRaWAN specification allows for two types of activation: recommended Over-the-Air Activation (OTAA) and Activation by Personalization (ABP)[22].

Over the Air Activation (OTAA)

In OTAA, a device is given a DevEUI, an AppEUI, and an AppKey. The AppKey is used to generate the session keys NwkSKey and AppSKey.
To activate, the device sends a join request and uses the join response to derive the session keys NwkSKey and AppSKey. The device can store these keys and continue to use them for communication. If they are lost or the network decides to let them expire, the device must rejoin to generate a new key[22].

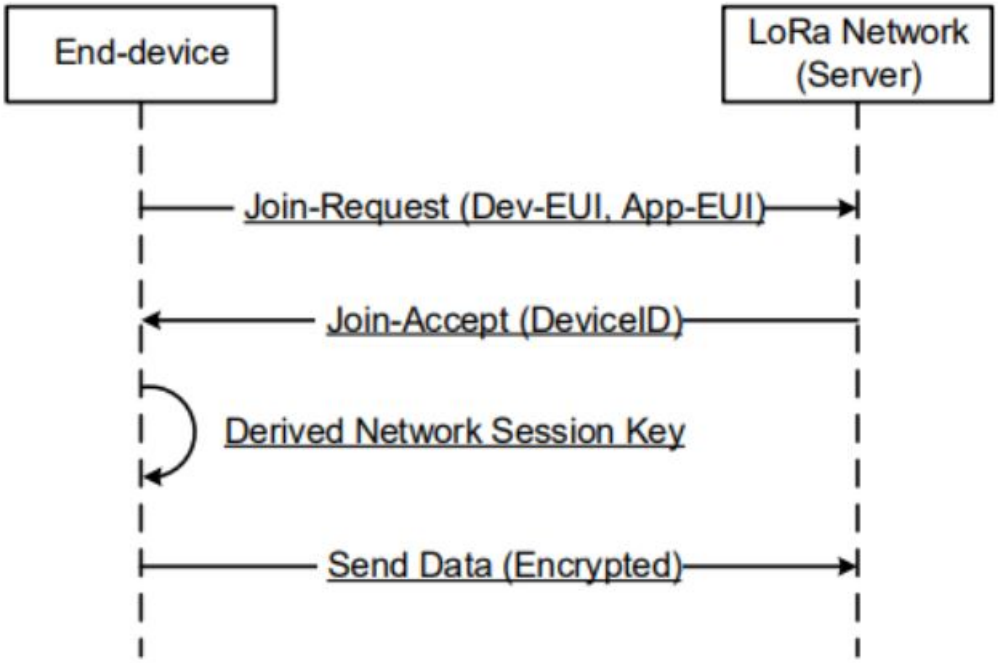

Figure 3.3: Overview of the OTAA Join procedure[23].

Activation by Personalization (ABP)

A DevEUI, AppEUI, or AppKey are not required for ABP activation.
Instead, the session keys NwkSKey and AppSKey are preprogrammed into the device and the device is preregistered on the network. In this mode, an end device skips the join procedure, which appears to be simpler, but it has some security implications[22].

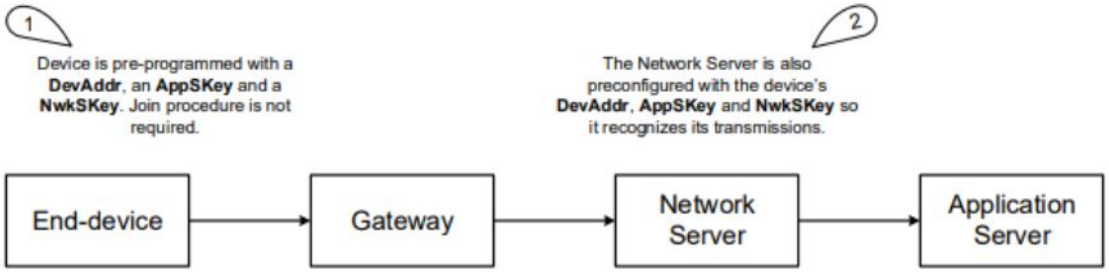

Figure 3.4: ABP End Device Activation[23].

3.1.6 LoRa Limitations

LoRa long-distance transmission flexibility comes at the expense of low throughput and limited channel activity time[24]

ISM Band Limitations

Each of the frequency bands imposes limits on the maximum amount of time devices are allowed to transmit[25].
ETSI[26] regulations impose a maximum duty cycle of 1% per cycle on each node, or 36 seconds per hour. This restriction can preclude important applications where some sensors occasionally require more than 36 seconds per hourly cycle to transmit their data.
The other restriction concerns the maximum transmission power for uplink messages, which is limited to 25mW (14 dBm), and for downlink messages the maximum transmit power is limited to 0.5W (27 dBm).

3.2 LoRaWAN Technology

LoRaWAN is a communication protocol and system architecture defined by the LoRa Alliance. LoRa describes the physical layer, i.e., the wireless modulation required to establish the long-range communication link, and LoraWAN describes the Media Acess Control (MAC) layer [27]. The LoRaWAN technology stack is shown below in the figure.

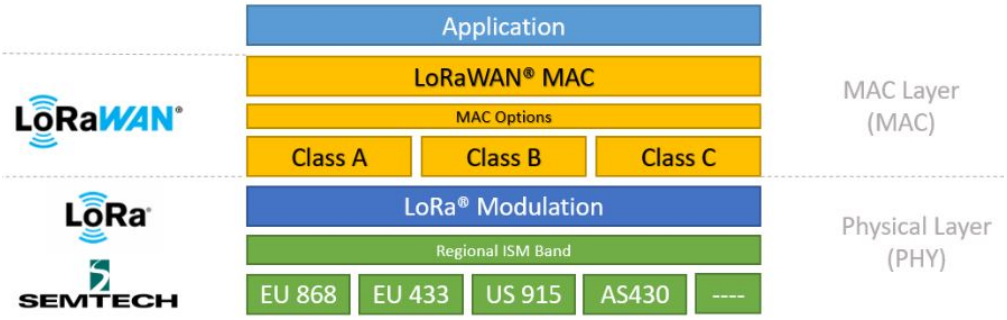

Figure 3.5: The LoRaWAN technology stack[27].

3.2.1 LoraWAN Architecture

The LoRaWAN network architecture is based on a star topology, which means that end devices can only communicate with LoRaWAN gateways and not with one another.

A central network server connects multiple gateways.
LoRaWAN gateways are only in charge of forwarding raw data packets from end nodes to network servers and encapsulating them in UDP/IP packets.
The network server is responsible for sending downlink packets and MAC commands to the end devices if necessary[28].
Furthermore, communication terminates at application servers, which can be linked to a single network server.

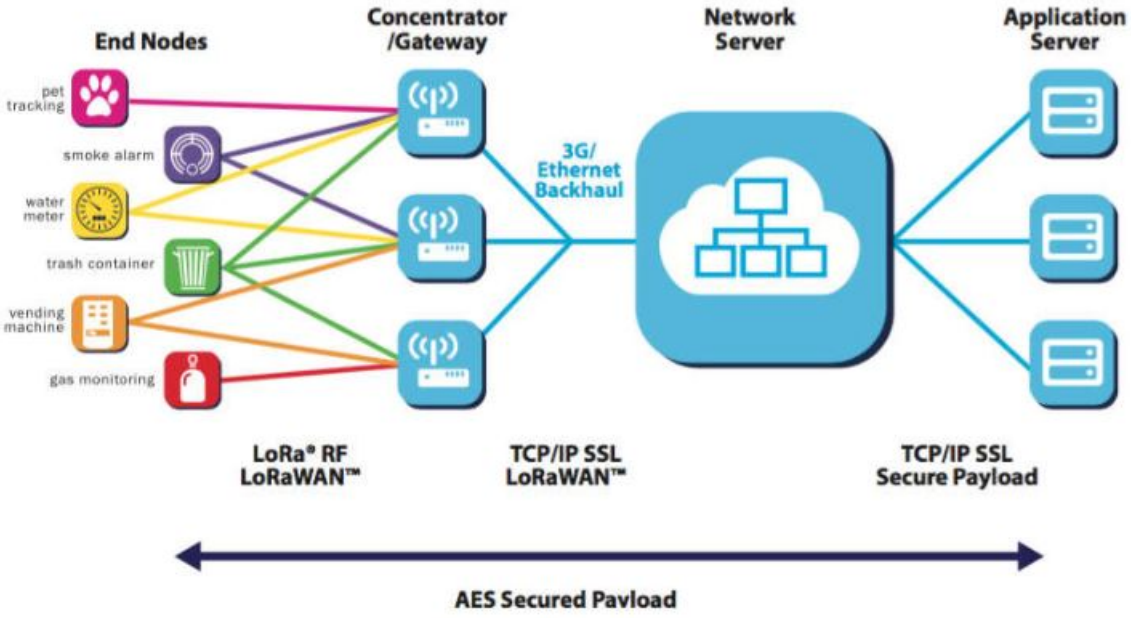

Figure 3.6: LoRaWAN network topology[29].

As shown in the figure3.6, different elements are part of the network.

3.2.2 End Nodes

A LoRaWAN-enabled end device is a sensor or actuator that is wirelessly connected to a LoRaWAN network using LoRa RF modulation via wireless gateways.
An end device in most applications is a self-contained, often battery-powered sensor that automates physical conditions and environmental events. Several unique identifiers are assigned to LoRa-based devices during manufacturing. These identifiers are used to securely activate and manage the device, ensure the secure transport of packets over a private or public network, and deliver encrypted data to the cloud[30].

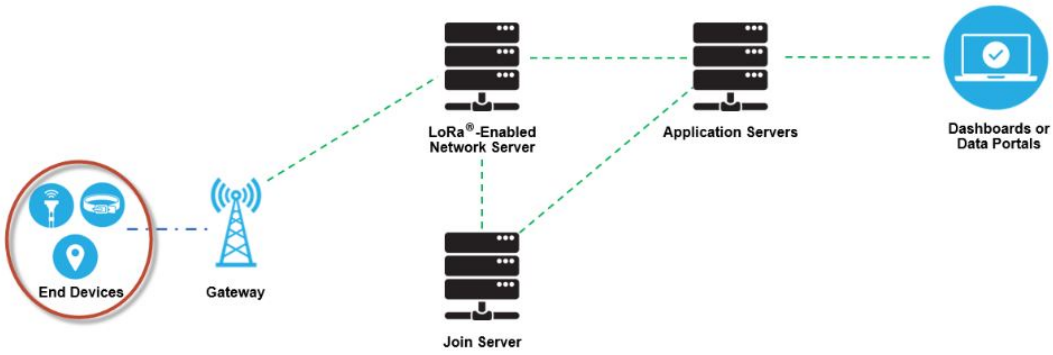

Figure 3.7: End devices in LoRaWAN network deployment[29].

3.2.3 Gateways

LoraWAN Gateways serve as a link between end devices and theLoRaWAN network server (LNS). Devices connect to the Gateway via low-power networks such as LoRaWAN, while the Gateway connects to The Network server via high-bandwidth networks such as WiFi, Ethernet, or Cellular[31].
All gateways within reach of an end device will receive the device's messages and forward them to the Network Server. This arrangement significantly reduces the packet error rate (since the probability that at least one gateway receives the message is very high).
Due to the possibility of multiple gateways receiving the same LoRa RF message from a single end device, the LNS performs data deduplication and deletes all copies. LoRaWAN gateways are nothing more than LoRa radio message forwarders that operate at the physical layer. They check the data integrity of each incoming LoRa RF message. If the integrity is not intact, i.e. if the CRC is incorrect, the message is discarded. If it is correct, the gateway forwards it to the LNS, along with some metadata that includes the Received Signal Strength Indicator (RSSI) of the message and an optional timestamp.
In LoRaWAN downlinks, a gateway executes the transmission requests coming from the LNS without interpreting the payload.
When transmitting a downlink message, the network server usually chooses the gateway that received the message with the best RSSI value based on the RSSI values of the identical messages.

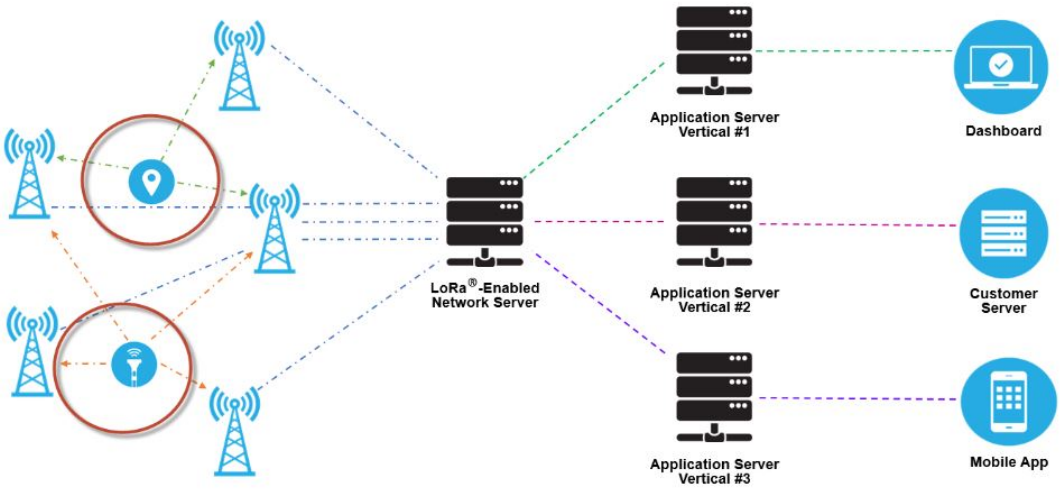

Figure 3.8: Gateways receiving and transmitting messages from end devices[29].

3.2.4 Network Server.

The LoRaWAN network server (LNS) manages the entire network, dynamically controls network parameters, and establishes secure 128-bit connections AES key called Network Session Key: NwkSKey[8].
For transporting both end-to-end data and for controlling traffic flowing from the LoRaWAN end device to the LNS (and back), the network server ensures the authenticity of each sensor and the integrity of each message. At the same time, the network server cannot see or access the application data.
This type of network where all gateways can send the same packet to the network server eliminates the need for handoff or handover. This is useful for asset-tracking applications that move from one location to another[15].

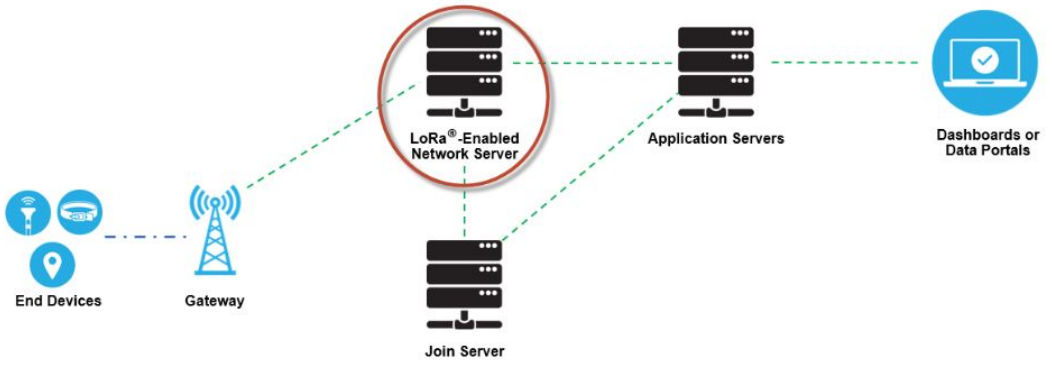

Figure 3.9: LoRaWAN Network Server in a LoRaWAN network deployment[29].

3.2.5 LoRaWAN classes

There are three types of devices in LoRaWAN: class A, class B, and class C devices.

- Class A: After each uplink transmission, class A end devices insert two receive windows. At the beginning of these receive windows, only downlink transmissions are allowed. In the first window, downlink transmissions use the same channel and data rate settings as the preceding uplink transmission[32]. However, in the second window, a fixed setting (DR0 (SF12/125 kHz)) at 869.525 MHz is used.
Class A is the most energy-efficient for applications that require only a short downlink connection after the terminal sends an uplink message.
At any other time, downlink communication must wait until the terminal's next uplink message.

- Class B: This is an optional mode for the end devices, which requires additional information from the base station as acknowledgment (ACK) that using this mode is preferable.
The End device receives a time-synchronized beacon from the base station to open receive windows at a predetermined time. This informs the network server when a terminal is listening.
Class B allows for a greater number of receiving windows than Class A, but at higher power consumption.
The additional regular receiving windows are opened after each uplink transmission, in addition to the synchronized opening of the two normal receiving windows, called ping slots.
To ensure synchronization, the Gateways beacons are transmitted every 128 seconds[32].

- Class C: The nodes keep their reception windows open all the time and close them only when the uplink sends a packet to one of them. This increases the end device's power consumption because it is constantly ready to receive[33].
Figure 3.10 depicts the Receive Windows of Class A, B, and C operating modes[32].

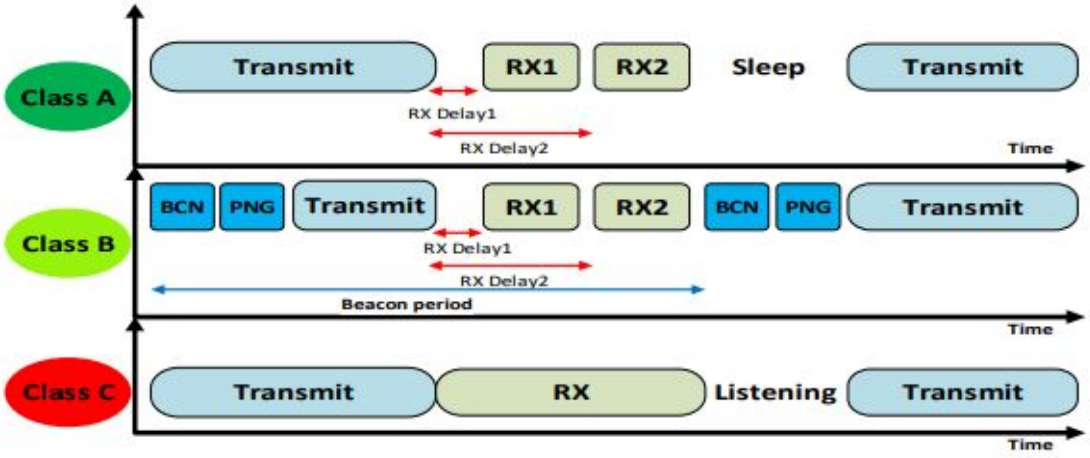

Figure 3.10: Class time diagram of LoRaWAN devices [34].

Device-to-device communication is not possible with LoRaWAN. Packets can only be sent from an End device to a network server or the other way around.

3.2.6 LoRaWAN Message Format

LoRaWAN employs the physical packet format outlined in Section 3.1.4
Because the header and Cyclic Redundancy Check (CRC) are required for uplink messages, a spreading factor of six (SF6) is not possible in LoRaWAN.
Downlink messages have the header but not the CRC[35].
The code rate to be used is not specified, nor is when the End devices should use low data rate optimization. The message format is detailed in the figure below.

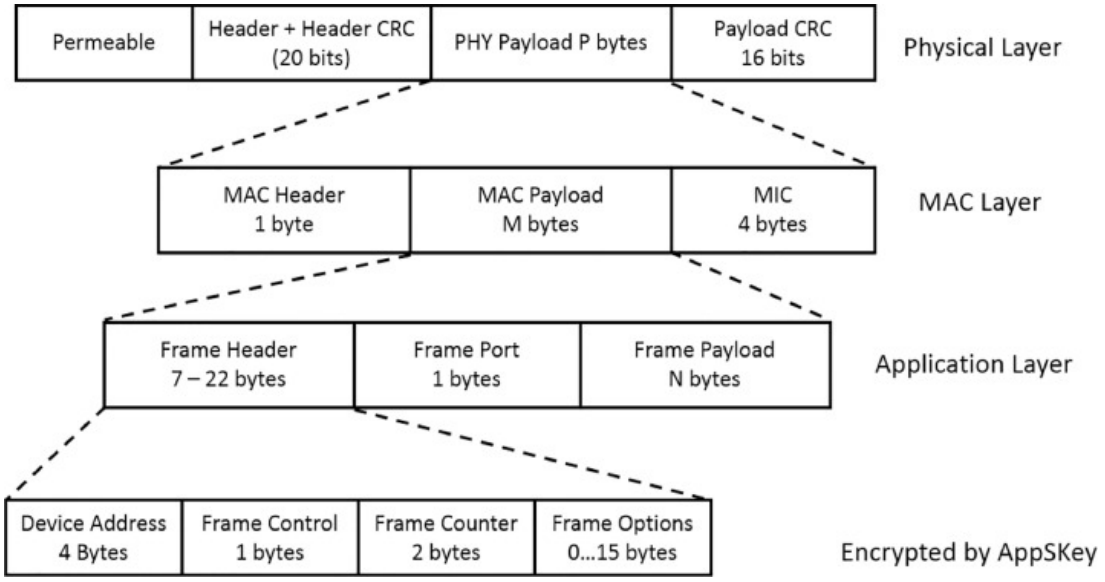

Figure 3.11: LoRaWAN frame format[36].

On the physical layer, the LoRa packet is composed of:

1. The preamble.
2. physical header (PHDR) and the Cyclic redundancy check of the physical header (PHDRCRC).
3. physical payload (PHypayload).
4. The Cyclic Redundancy Check (CRC) of the packet.

The PHDR is required for both uplink and downlink messages, whereas the CRC is only required for uplink communications[35].

The PHYPayload contains:

1. The MAC header.
2. The MAC payload.
3. Cryptographic Message Integrity (MIC).

The MAC header contains information about the LoRaWAN version used (v1 or v2) and the Message Type (Mtype).
The Mtype field allows registration packets (Join-Request/Accept) to be distinguished from Unconfirmed-data and Confirmed-data packets.

MIC is a code calculated using the MAC Header (MHDR).
The MAC payload contains Frame Header (FHDR), Port Field (FPort), and Frame Payload (FRMPayload).

The FHDR field contains information about the address of the end device (DevAddress) assigned by the Network Server to this End device once it has successfully joined the network, and other control information contained in the Frame Control (FCtrl) field, such as the Adaptive Data Rate (ADR) status for the communication[37].

The Port Field (FPort) field has a value of 0 in the case of Frame Payload (FRMPayload), which contains only MAC commands, while it is used by the application to distinguish the contents of the payload, so the value of the packet is application-specific.
FRMPayload is encrypted with AES with a key length of 128 bits[37].

Each End device has a packet counter (FCnt field) to number subsequent data packets that are sent to the Network Server.

Along with the physical layer messages, the Network Server also receives additional information about the physical parameters of the communication, such as Signal to Noise Ratio (SNR) and Received Signal Strength Indicator (RSSI)[38].

3.2.7 Adaptive Data Rate (ADR)

An Adaptive Data Rate (ADR) mechanism is built into LoRaWAN to dynamically manage the link parameters of an End device to improve the Packet Delivery Ratio (PDR). The ADR mechanism manages the data rate and transmission power on the network[39].
When ADR is enabled, the network server instructs the end device to reduce transmit power or increase the data rate. End devices near gateways should have a lower spreading factor and a higher data rate, whereas devices further away should have a higher spreading factor because they require a larger connection budget[8]. The ADR algorithm at the end node is straightforward. It has two parameters, ADR ACK LIMIT and ADR ACK DELAY, which are set to 64 and 32, respectively. For each uplink packet transmitted, the end device will increment ADR ACK CNT. When ADR ACK CNT exceeds ADR ACK LIMIT, the end device sets the ADRACKReq bit and waits for network Acknowledgement (ACK) for the next ADR ACK DELAY uplink packets[40].
If no acknowledgment is received before ADR ACK DELAY uplink packets, the end device will reduce its data rate in an attempt to reconnect to the network.

The end device initially attempts to connect by raising its transmit power. If this is insufficient, the data rate will be reduced further[40]. The following Figure depicts the ADR mechanism implemented in the end node.

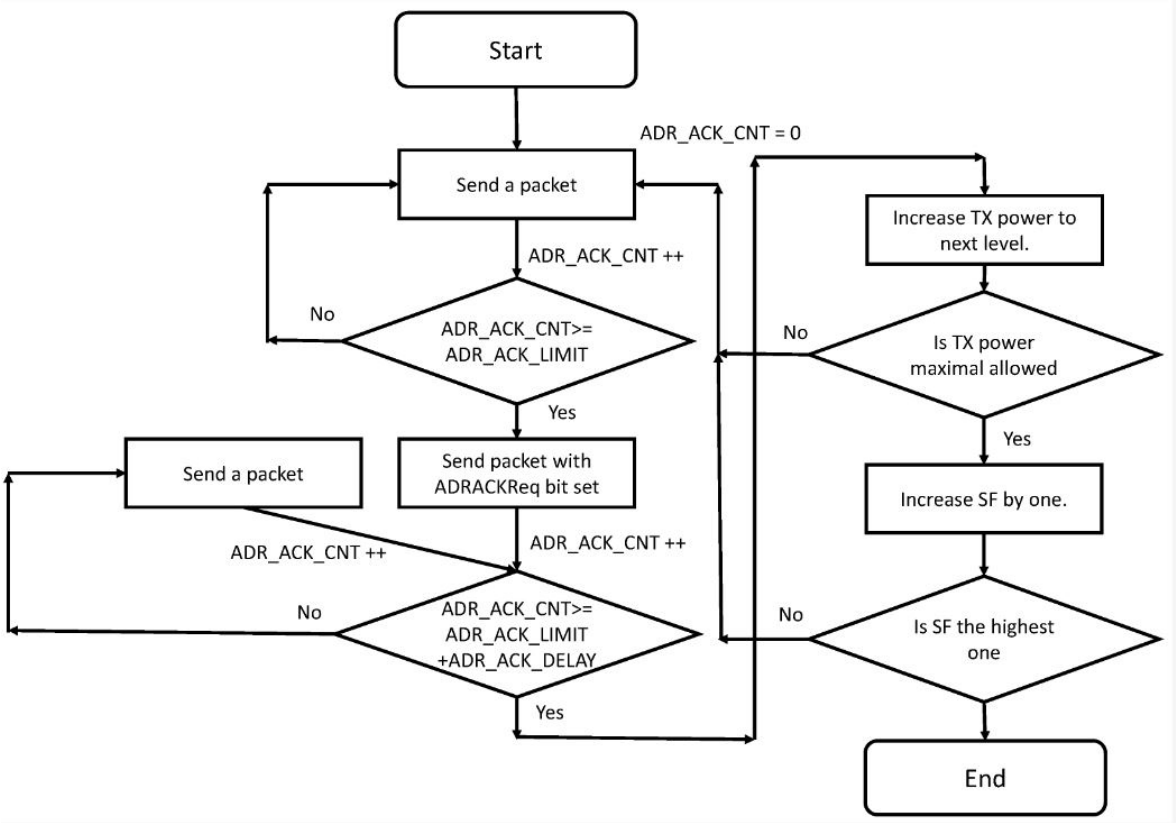

Figure 3.12: ADR mechanism[41].

3.2.8 Authentication And Encryption

LoRaWAN specifies some security keys: NwkSKey, AppSKey, and AppKey. All keys have a length of 128 bits. The algorithm used for this is AES-128.
When a device joins the network, an Application Session Key (AppSKey) and a Network Session Key (NwkSKey) are generated. The Network Session Key (NwkSKey) is used to provide authentication between the LoRa device and the Network Server. To perform this authentication, a Cryptographic Message Integrity (MIC) field is inserted into the frame. It is calculated based on the data sent and the NwkSKey. The same calculation is performed on the receiver. If the NwkSKey in the device and in the network server is identical, the two estimated MICs must be identical[8].

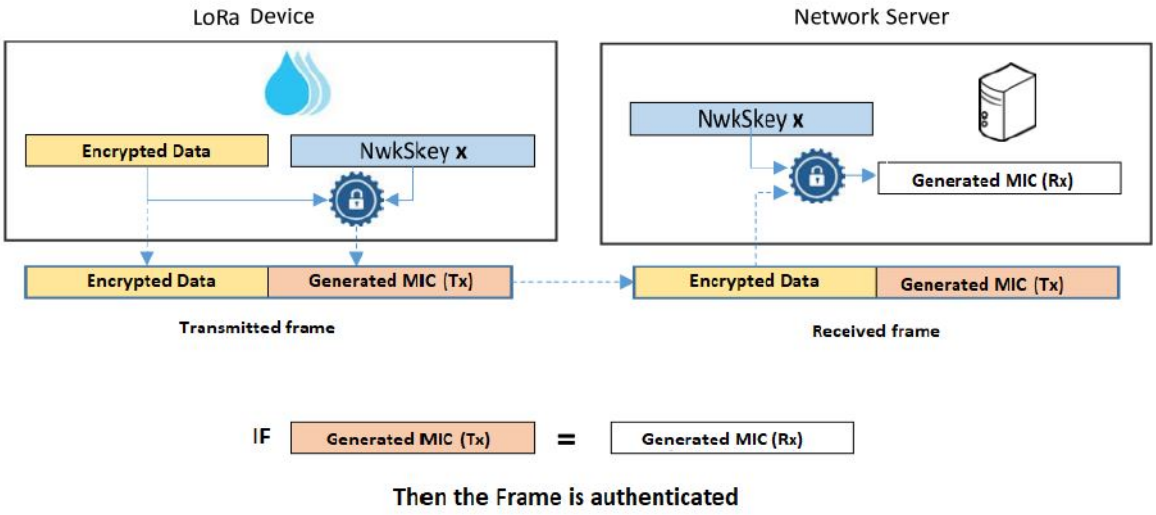

Figure 3.13: Frame authentication by the Network Server[42].

The Application Session Key (AppSKey) is used to encrypt data between the end device and the Application Server. Only if the data has the same key, it will be decrypted.

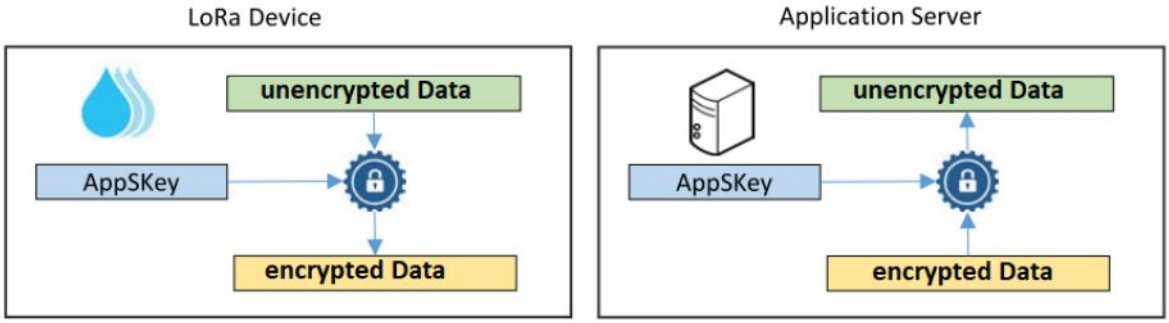

Figure 3.14: Data encryption/decryption.[42]

The Application Key (AppKey) is known only to the device and the application. Dynamically activated devices (OTAA) use the AppKey to derive the two session keys during the activation process[42].
In The Things Network (Network Server (NS)), a default AppKey can be used to activate all devices, or one AppKey can be customized per each device[42].

Chapter 4

System Design and implementation

4.1 Proposed System Model

In this chapter, we will present the system model and the components used for this approach.

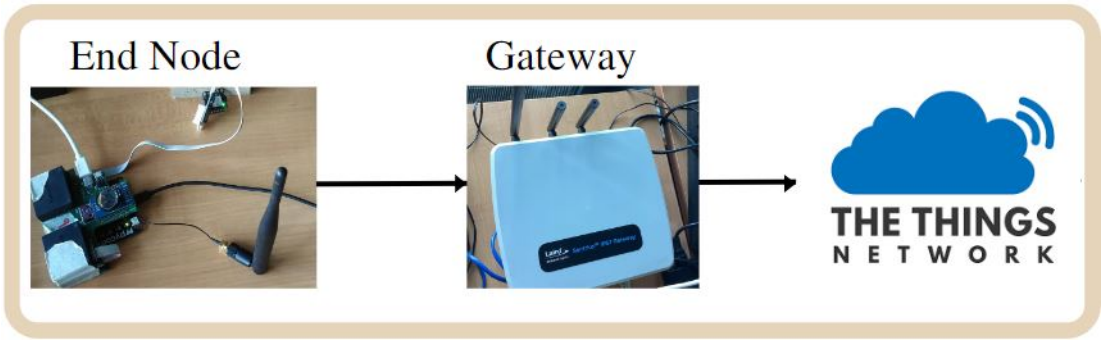

Figure 4.1: system model.

4.2 System Architecture and components

4.2.1 Gateway

Each LoRa gateway consists of the following two components:
1. a signal demodulation processor.
2. a single or two TX /RX radios.
The gateway used in this experiment is Sentrius RG186. Based on Semtech's

SX1301/SX1257 chipset designs, it offers up to 10 miles of LoRa range and preloaded LoRa Packet Forwarder software, perfect for highly scalable and flexible IoT networks[43]. The gateway can connect to the Internet via WLAN and Ethernet, and can receive on 8 channels in parallel in the 868 MHz and 867 MHz sub-bands and with all spreading factors. Since the spreading factors are orthogonal to each other, two packets with different SF can be delivered on the same channel at the same time[44].

Figure 4.2: Sentrius RG186 Gateway.

4.2.2 LoRa End Device

We have worked with the LoPy4 board and its expansion board. The LoPy4 is a quad MicroPython-enabled development board (LoRa, Sigfox, WiFi,Bluetooth).

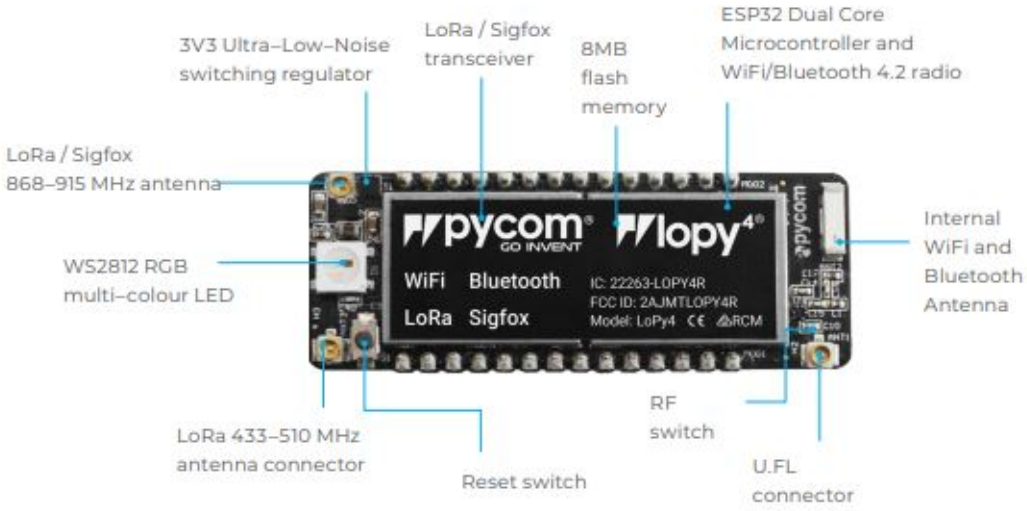

Figure 4.3: LoPy4 board [45].

With the latest Espressif chipset, the LoPy4 offers a perfect combination of performance, ease of use, and flexibility. The ESP32 microcontroller, powered by 3.3V-5.5V, is connected via a high-speed SPI link to Semtech's SX1276 chip, which is responsible for performing LoRa modulation[45]. The block diagram of LoPy4 is shown in the Figure below.

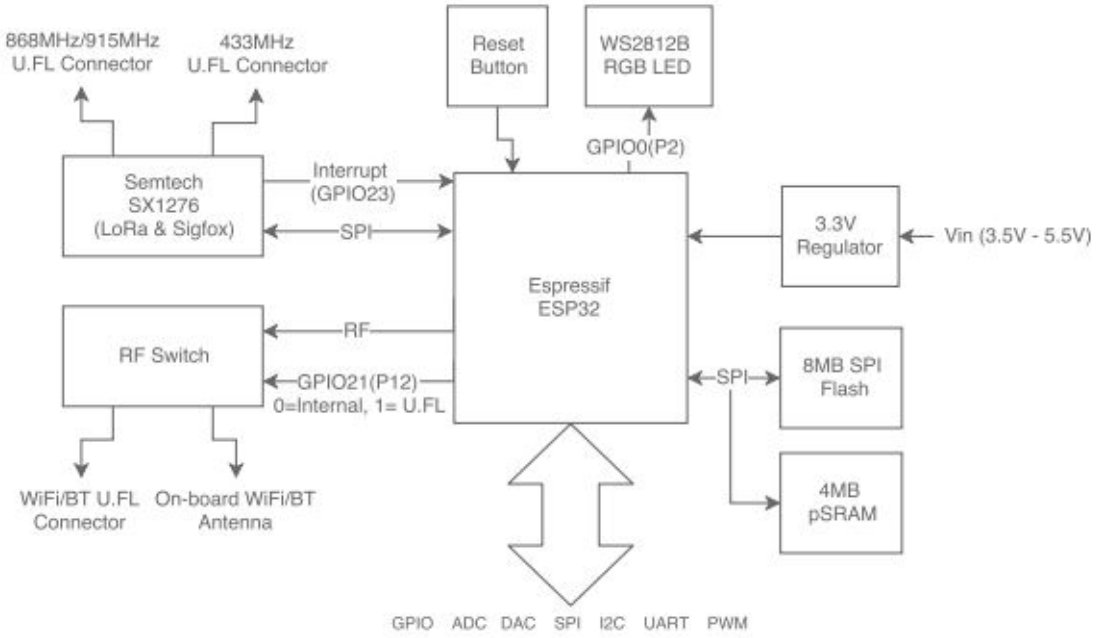

Figure 4.4: Block Diagram of LoPy4.[45]

Sensors

The sensor node is composed of:

1. 4 PM2.5 and PM 10 sensors: ranges from 0 µg/m3 to 1000 µg/m3 with ±15% accuracy[46].

2. Digital-output relative humidity and temperature sensor, also known as DHT22. Temperature readings range from -40 to $80°C$, with $\pm 0.5°C$ accuracy and humidity readings range from 0% to 100% with 2% to 5% accuracy[47].

3. Atmospheric pressure sensor: barometric pressure sensing range: from 300 to 1100 hPa; resolution up to 0.03hPa/0.25m[48].

4. Global Positioning System (GPS) sensor:-165 dBm sensitivity, up to 10 Hz updates[49].

4.2.3 Firmware Flowchart

An adapted Firmware was developed in order to control the sensing function and the LoRa transmission of the data.
The data was delivered in a specific format that enables the sensors' accuracy to

be preserved while limiting payload length as much as possible.
Below is a flow chart that gives an idea about the methodology used to assure the transmission of the data.

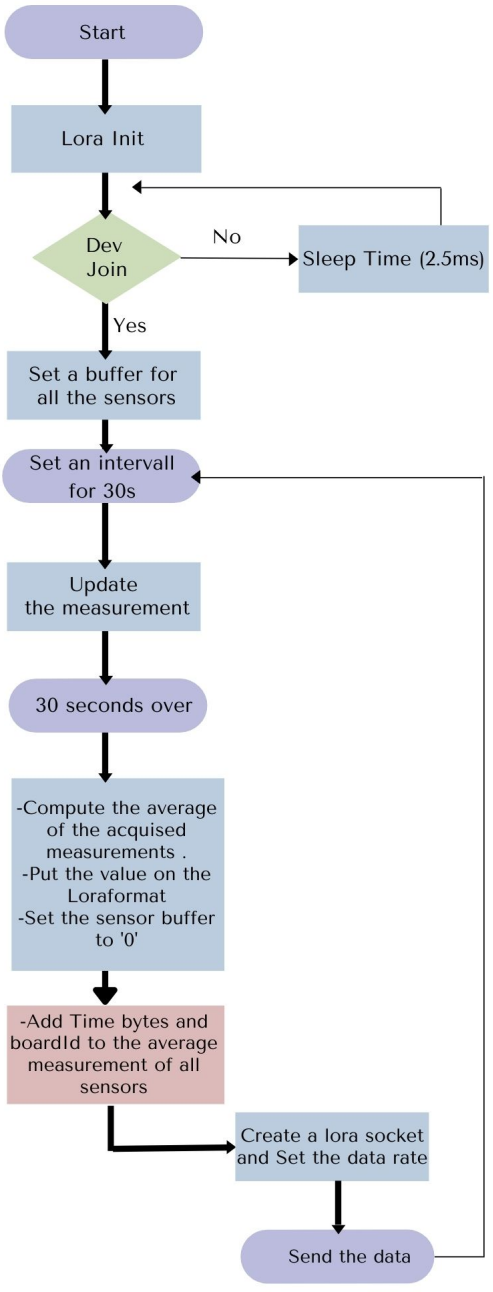

Figure 4.5: Flow chart.

4.2.4 Network Server: The Things Network TTN

The Things Stack is a full-featured, open-source LoRaWAN network server intended for a variety of deployment scenarios.
It supports all existing LoRaWAN versions, device classes A, B, and C, as well as all LoRa Alliance regional parameters.
This network server consists of an architecture of loosely coupled components that provide security, data routing, and battery optimization for LoRaWAN endpoints[50]. Gateways are required as a bridge between specific radio protocols and the Internet in a typical Internet of Things network. Non-IP protocols, on the other hand, such as LoRaWAN, necessitate some form of routing and processing before messages can be delivered to an application.
These routing and processing steps are handled by the Things Network, which sits between the gateways and the applications[51].
The Things Network is a highly secure public network that provides true end-to-end encryption, protection against attacks, and support for multiple 128-bit encryption keys for each endpoint.

4.2.5 LoraWAN and TTN Limitations

LoraWAN Limitations

LoRaWAN is not suitable for every use case, so it is important to know its limitations.

- Uplink messages: Sending data from a Node to an Application:

The payload should be as small as possible. This means that instead of sending JSON or plain (ASCII) text, data should be encoded as binary.
The interval between messages should be chosen wisely with respect to the length of the payload and the data rate used.
Data Rate should be the fastest possible to minimize the airtime.

- Downlink messages: Sending data from an Application to the node:
Downlink messages should be avoided if possible in order to preserve the reliability of the network. If it's necessary to use the downlinks, the payload should have a short length.
Downlinks use the same Data Rate as the uplinks so it needs to be as efficient as possible[52].

TTN Limitations

On The Things Network, a fair use policy applies that:

- The maximum airtime of uplink messages is 30 seconds per day (24 hours) per node.
- The maximum number of downlink messages is 10 messages per day (24 hours) per node.
- Devices with hard-coded spreading factors of SF12 or SF11 are not allowed to join the network[53].

Chapter 5

Tests and result analysis

This chapter will provide a description of the tests carried out utilizing the system architecture discussed in the previous chapter.
We will investigate the system's performance under various conditions in order to study its functionality and analyze the performance of LoRaWAN in a tough environment.

5.1 Initial Test Parameters

During the measurements, each node sends a packet of 33 bytes of application payload to the base station every 30 seconds.

Time 4Bytes	BoardID 2Bytes	Temperature 2Bytes	Humidity 1Byte	Atmpress 2Bytes	PM2.5 8Bytes	PM10 8Bytes	GPSlg 3Bytes	GPSlt 3Bytes

Figure 5.1: Application Payload Format.

As mentioned in 3.2.7, when ADR is in use, the network server adjusts the data rate and TX Power using the Received Signal Strength Indicator (RSSI) of the uplinks messages[39].
ADR benefits include preserving an End Device's battery life and reducing interference, allowing all End Devices on the network the best opportunity of connecting successfully.
In Europe, LoRaWAN operates in the 863-870 MHz frequency band. This is the Regional ISM band. As specified in the LoRa specification, the 3 main channels

that should be used, of bandwidth equal to 125 kHz, are : 868.10 MHz, 868.30 MHz,868.50 MHz. The Things Network which is our Network Server adjusts the remaining 5 frequencies[54].
Below is the figure of the list of frequencies used in The Things Network for Europe.

Figure 5.2: TTN frequency plan [55].

The Transmission Power (Tx) for the End nodes is fixed by TTN to 14 dBm (25mW).

RSSI and SNR

Received Signal Strength Indicator (RSSI) and Signal to Noise Ratio (SNR) are the two components that determine the quality of the received radio signal. These values are measured by the LoRa Gateway for each communication that it receives[56].
RSSI is the received signal power in milliwatts and is measured in decibels (dBm). RSSI values range from -30 dBm to -120 dBm. RSSI = -30 dBm indicates a very strong signal, whereas RSSI = -120 dBm indicates a very faint signal
SNR is the ratio between the received signal power and the noise floor power level. SNR values range from -20 to +10 dB. A value closer to +10 dB means that the received signal is less corrupted. LoRa can demodulate signals ranging from -7.5

dB to -20 dB below the noise floor.

5.2 Test Scenarios

5.2.1 Indoor tests

In order to validate the functionality under a controlled environment using our own Gateway 4.2.1, a test was done in two different rooms.

Room 1

The End node is in the same room as the Gateway.
In order to establish uplink and downlink communication, the End Device should be registered to an Application on The Things Network (TTN). So we add an Application.

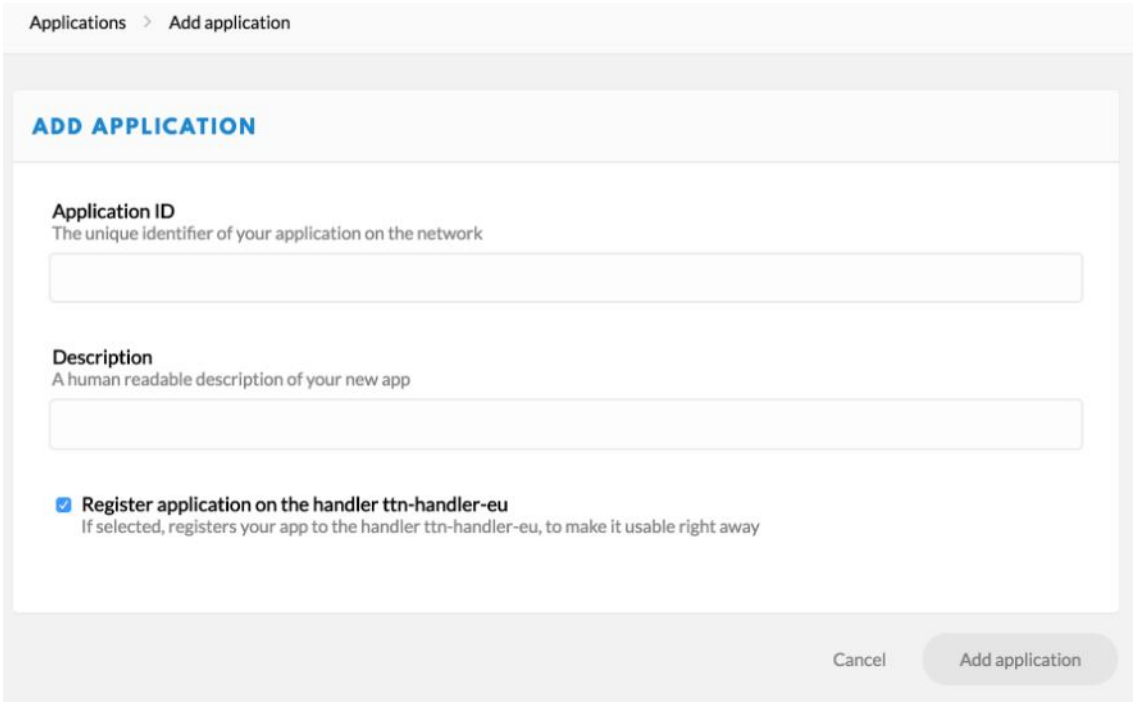

Figure 5.3: Application creation on TTN .

When clicking on Add application, an App EUI and Access Keys are created. Those keys are used to add a device within this application. They are then used for the security of the data transmitted, as mentioned in 3.2.8.

TTN decrepit the data received and a decoder function is developed in order to reverse the encoding done by the device.

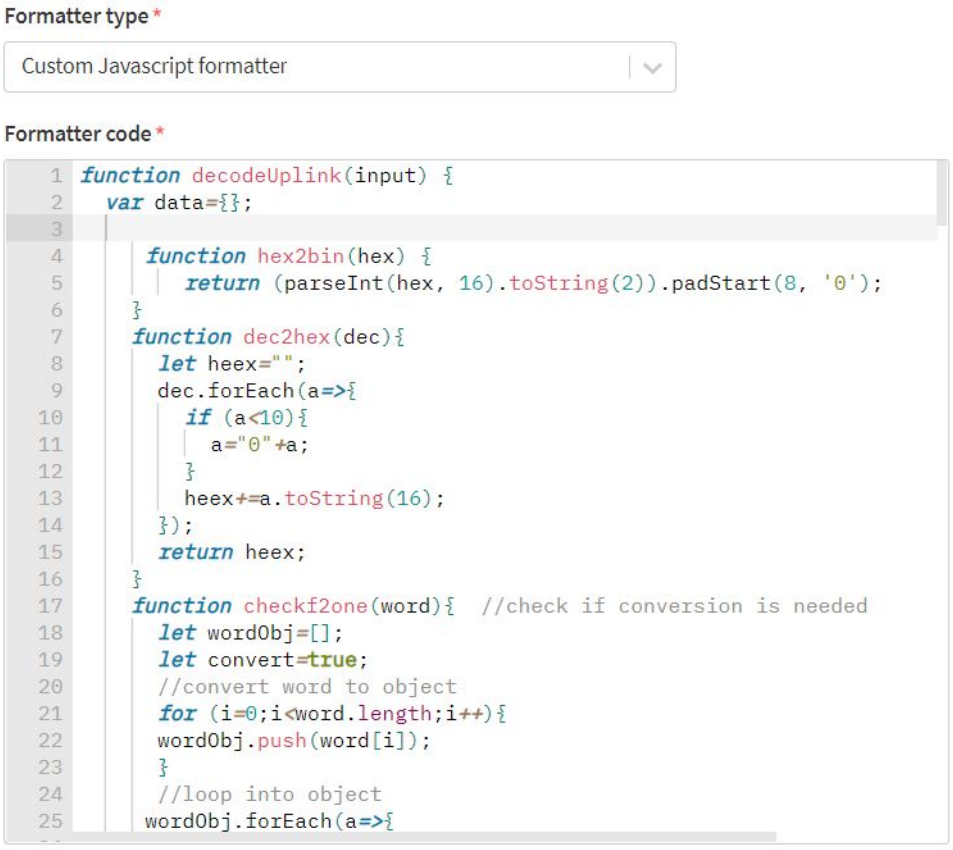

Figure 5.4: Decoder function on TTN.

The complete function can be found in the appendix A.
The initial trial took place inside a small range, room1. The information about the RSSI and SNR of one of the received uplinks is in the figure below:

Figure 5.5: RSSI and SNR values for an uplink message for room1.

Based on the RSSI and SNR values, we may conclude that the communication

quality is almost adequate. A large RSSI value was predicted given the short distance between the Gateway and the End node.

Room 2

The End node is two floors below the Gateway. The figure below shows a capture from the gateway interface.

Figure 5.6: Gateway capture for room2.

The amount of RSSI and SNR decreased significantly when the distance and obstacles between the receiver and the transmitter were increased compared to the values obtained for the experiment in room1. Positive SNR indicates that the signal power is greater than the noise power, allowing the receiver to demodulate the signal.
The signal cannot be demodulated if the RSSI is less than the noise floor.
LoRa can, however, demodulate signals that are below the noise floor.
The figure below depicts the minimum SNR required for demodulation at various spreading factors.

Spreading Factor	Spreading Factor (chips/symbol)	LoRa Demodulator SNR
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Figure 5.7: Minimum SNR for demodulation.

5.2.2 Outdoor tests

The outdoor tests took place in the Italian city of Turin. The Turin metropolitan area has high buildings, which makes communication more challenging. To evaluate the performance of our system under the requirements of Lora, LoRaWAN, and TTN, different locations were chosen for the End node.

Test points

The test points are shown in the Figure below:

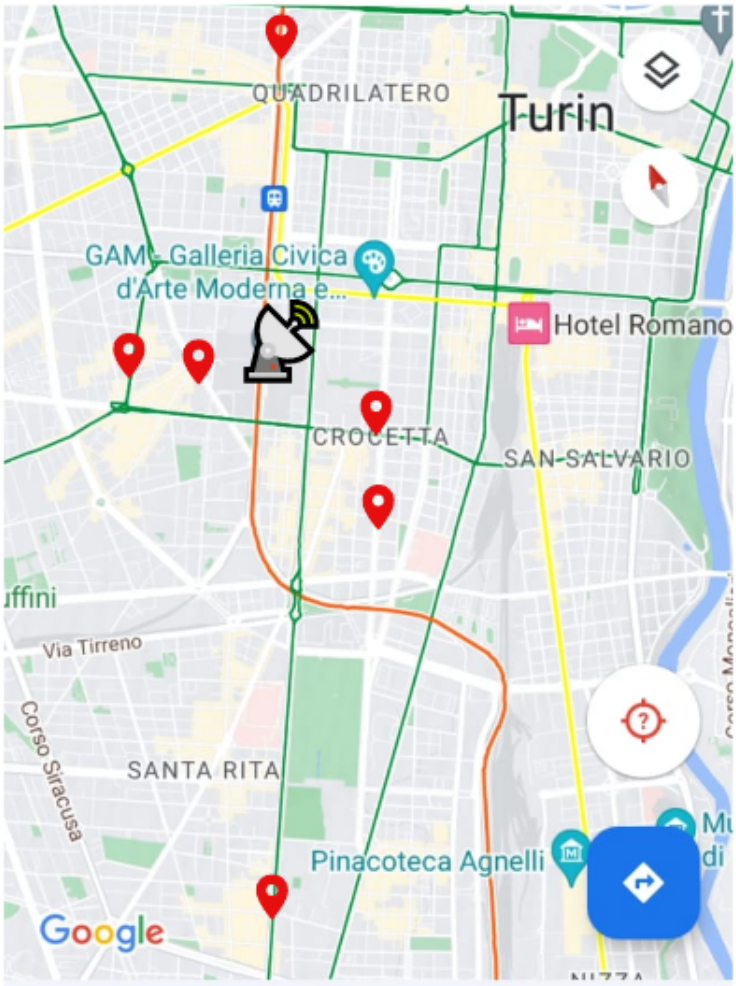

Figure 5.8: Test positions over Turin city.

Six positions were chosen to emulate the Line of Sight (LOS), and Non Line of Sight (NLOS) at relatively short and long distances. LOS, and NLOS are characteristics that explain whether the sender of a signal and the receiver have an unobstructed vision of another.
For LoraWAN, having a Line of Sight (LOS) between the sender and the receiver will lead to a good performance.

Location	Characteristic	Distance(m)
Loc1	LOS	900
Loc2	LOS	1300
Loc3	LOS	2210
Loc4	NLOS	646
Loc5	NLOS	1300
Loc6	NLOS	2600

Table 5.1: Positions information.

The payload length is 33bytes when using a customized LoRa format5.1 to communicate this data along with the time and date..
We sent 60 data packets with 30 seconds between each.
The gateway status and performance can be evaluated by analyzing RSSI and SNR values.
TTN takes the 20 most recent uplinks, starting at the moment the ADR bit is set. These measurements contain the frame counter, SNR, and the number of gateways that received each uplink. For each of these measurements, the SNR of the best gateway is taken and the margin is computed in order to determine the optimal Data Rate (DR). The margin is the difference between the measured SNR and the required SNR to demodulate a message given the DR:

$$Margin = SNR - SNR_{demodultaion} \tag{2}$$

This margin is used to determine how much the DR could be increased or the transmit power lowered. As a result, Time On Air (TOA) will be used more efficiently.

Testing

We start with Line of Sight (LOS)positions: uplinks sent from position 1 were received by more than one gateway.

Figure 5.9: Uplink message for Location1.

As shown in Figure 5.9, two gateways have received the signal: our Sentrius gateway which is called on TTN "mineloragateway" and Polito gateway "polito". As previously stated, the LoRaWAN architecture employs a star of stars topology, allowing multiple gateways to receive the same packet.
To quantify the Packet Delivery Ratio (PDR) measurements, we will only consider packets received by the Polito gateway. The positive SNR indicates that the RSSI was greater than the noise power, allowing the receiver to demodulate the signal. As a consequence, all the packets sent were received.

The second and the third positions are in LOS and far from the Gateway with respectively 1300m, and 2210m.
The figure below is a capture of one of the uplinks details for both locations:

Figure 5.10: Information of uplink message for position2 and position3.

The ADR was activated which allowed decreasing the Data Rate from DR5 (=SF7) to DR4 (=SF8) respectively. As a result, both SNR levels are positive. The SNR is equal to 6 for location2 and equal to 3 for location 3. Although, having a positive SNR doesn't mean that the packets sent were all received. We have some packets that were lost.
Non Line of Sight (NLOS)positions:
We began the experiment 646m away from the gateway.

Figure 5.11: Information of uplink message for position4.

As shown in the Figure above, the RSSI is low and close to the minimum value. This is a consequence of being out of the Line of Sight (LOS) and surrounded by some high buildings.
Since the range is not very high (646m), the SF7 didn't get increased and the SNR is almost good.
Going further away from the gateway by 1300m and 2600m yields the following result.:

Figure 5.12: Information of uplink message for position5 and position6.

Placing the End nodes far from the gateway and surrounded by many obstacles, such as buildings, autos, and buses involves an SF adjustment for both sites. Position 5 (1300m): from SF7 to SF8 and position6: from SF7 to SF9 (2600m). The RSSI falls as the range rises, and so does the SNR. As a consequence, some packets get lost and don't reach the gateway.

5.2.3 Result Analysis

Packet Delivery ratio (PDR)

The Packet Delivery Ratio is the ratio of the number of packets received at the gateway to the number of packets sent from the End node.

$$PDR\% = \frac{R}{T} \tag{3}$$

with R: the number of received packets
T: the total number of packets sent.
The table below summarizes the PDR for each node as well as the efficient spreading factor for each.

Location	SF	Number of received packets	PDR(%)
Loc1 (LOS/430m)	7	60	100%
Loc2 (LOS/876m)	8	54	90%
Loc3 (LOS/2210m)	8	48	80%
Loc4 (NLOS/646m)	7	47	78%
Loc5 (NLOS/1700m)	8	46	76%
Loc6 (NLOS/3000m)	9	44	73%

Table 5.2: PDR values.

The LOS positions have better PDR values compared with the NLOS ones. Having a straightforward line of sight between the Gateway and the End node and having fewer impediments such as trees, cars, and buildings, as well as interference from other radio systems, provides for the reception of a higher number of packets. The amount of received packets is affected by the distance between the receiver and the message's emitter. The longer the distance, the lower PDR.
However, with ADR enabled, TTN instructs the End Device to raise the Data Rate (DR) in order to ensure the delivery of all packets at the expense of Time On Air (TOA) for each of them.

Time On Air (TOA)

Time On Air (TOA) refers to the amount of time it takes for a signal sent from an End Device to reach the Gateway.

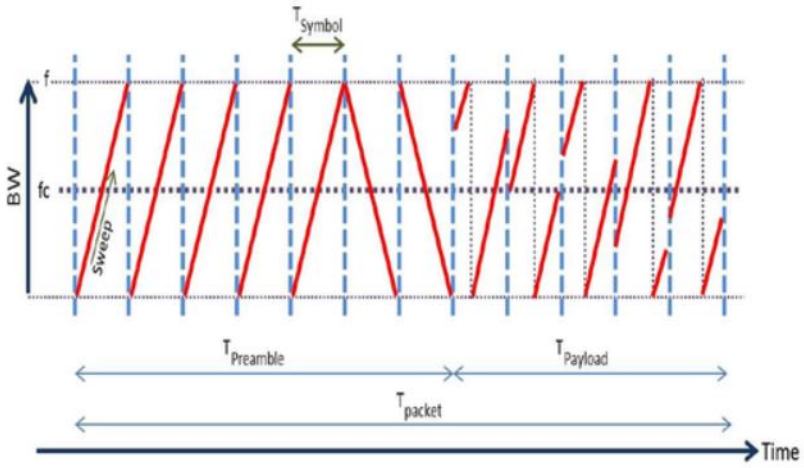

Figure 5.13: Chirps containing preamble and payload[57].

Since LoRa packet consists of preamble and payload symbols as seen in 3.1.4. The total transmission time of a LoRa packet is calculated as follows:

$$TOA = T_{packet} = T_{preamble} + T_{payload} \tag{4}$$

$T_{preamble}$ is a function of T_S which represents the symbol period.
The chip rate R_c and symbol rate R_S are expressed in the following equations:

$$R_c = \frac{1}{BW} \tag{5}$$

$$R_S = \frac{BW}{2^{SF}} \tag{6}$$

where: BWstands for Bandwidth and SF for Spreading Factor.
Using 5 and 6, T_S can be defined as shown in the following equation:

$$T_S = \frac{1}{R_c} \tag{7}$$

Therefore, the transmission time for the preamble is:

$$T_{preamble} = (n_{preamble} + 4{,}25) * T_S \tag{8}$$

with $n_{preamble}$ representing the length of the preamble, that is fixed by LoRawAN to 8 for all regions.

$$T_{payload} = n_{payload} * T_S \tag{9}$$

With

$$n_{payload} = 8 + max(\frac{8PL - 4SF + 28 + 16CRC - 21IH}{4(SF - DE)}(C_r + 4){,}0) \tag{10}$$

where PL: the number of bytes of the payload.
SF: the spreading factor.
IH: the implicit header mode when (0) or explicit header mode when (1).
DE: Low data rate, disabled when (0) or enabled when (1).
CRC: is not present in payload when (0) or is present in payload when (1). CRC goes from 1 to 4.
There is some online TOA calculator that allows knowing the exact TOA taken by each packet depending on its length[58].

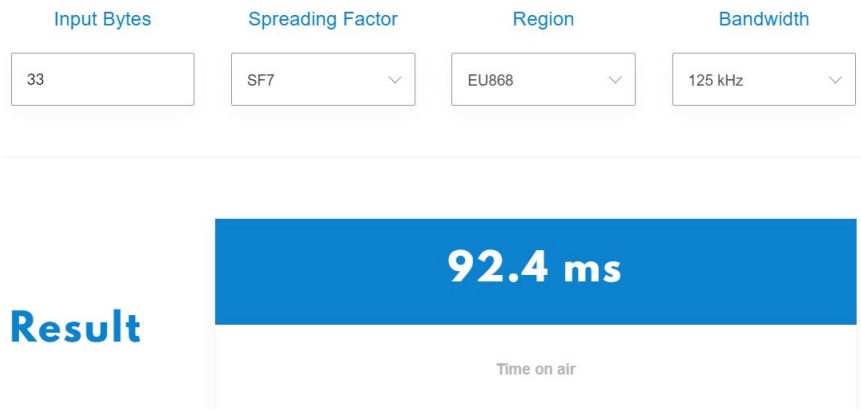

Figure 5.14: Online Air Time calculator[59].

The values calculated online can be then compared to the amount of time we have on TTN console for each uplink and they should be equal. Considering the table 5.3 below:

Mode	TOA (ms)	Max payload size (bytes)
SF7/125KHz	92,4	222
SF8/125KHz	164,4	222
SF9/125KHz	308,2	115
SF10/125KHz	575,5	51
SF11/125KHz	1232,9	51
SF12/125KHz	2302	51

Table 5.3: TOA and maximum payload sizes for different SF.

Sending a fixed size of data with a higher Spreading Factor (SF) and a fixed bandwidth requires more TOA than with a lower Spreading Factor (SF).
A signal modulated with a higher SF can be received with reduced errors than a signal modulated with a lower SF.
Signals using a high SF are able to travel long distances but consume more time to reach their destinations. This is explained by the payload size constraint imposed by the SF utilized5.3.

Packet length

For LoRaWAN, it is recommended to send data with as fewer bits as possible in order to cover a larger area and to conserve the TOA for each End node.
The following figure represents theTOA and the number of messages allowed to be transmitted for a higher payload length for all the possible SFs:

Output	SF6	SF7	SF8	SF9	SF10	SF11	SF12	spreading factor; higher means more range and better reception, but also more airtime
Tsym	0.512	1.024	2.048	4.096	8.192	16.384	32.768	ms
Tpreamble	6.272	12.544	25.088	50.176	100.352	200.704	401.408	ms
payloadSymbNb	103	88	78	68	63	68	63	number of symbols
Tpayload	52.736	90.112	159.744	278.528	516.096	1114.112	2064.384	ms
Tpacket	59.008	102.656	184.832	328.704	616.448	1314.816	2465.792	ms total airtime to send the full packet
TTN Fair Access Policy		292	162	91	48	22	12	average messages/day for maximum of 30 seconds airtime on The Things Network
		12.2	6.8	3.8	2.0	1.0	0.5	average messages/hour when sending all day

Figure 5.15: TOA for 50 bytes payload length.

When compared to the TOA of the 33bytes payload 5.3, it is evident that the length of the data is critical to ensuring that less time is spent transmitting.
To accomplish this, appropriate data formatting was implemented. A specific payload format was developed in order to satisfy these requirements and recommendations. The payload format used was mentioned in 5.1.

The sensing function works as follows: each sensor takes different measurements for 30 seconds, compute the average, and transmits only that value. To do this, a buffer for all the sensors was created and then updated each time a sensor has a new measurement.

```
def update_lora_buffer(self, sensor, measurement):
    self.sensor_buffer[sensor]['value'] += measurement
    self.sensor_buffer[sensor]['count'] += 1
```

Figure 5.16: Update LoRa buffer.

To satisfy the accuracy of each sensor, some operations were made. The figure below represents the function that encodes each measurement on the exact number of bits while conserving the accuracy. The opposite was done on TTN side, inside the payload formatter function in order to extract the data.

```
def LoraFormat(value, type):

    if type is 'T': # Temperature
        if value != None:
            sensor_value = (int((value+40)*10)).to_bytes(2,'big')        #-40<temp<80
        else:
            sensor_value = int(0xffff).to_bytes(2,'big')
    elif type is 'H': # Relative Humidity
        if value != None:
            sensor_value = (int(value *2)).to_bytes(1,'big')             #0<humid<100
        else:
            sensor_value = int(0xff).to_bytes(1,'big')
    elif type is 'A': # Atmospheric Pressure
        if value != None:
            sensor_value = (int(value*10)).to_bytes(2,'big')
        else:
            sensor_value = int(0xffff).to_bytes(2,'big')
    elif type is 'P': # Particulate Matter Concentration
        if value != None:
            sensor_value = (int(value)).to_bytes(2,'big')                #0<pcm<1000
        else:
            sensor_value = int(0xffff).to_bytes(2,'big')
    elif type is 'G': # GPS Latitude and Longitude
        if value != None:
            sensor_value = int((value + 180)*10000).to_bytes(3,'big')    #-180<gps<180
        else:
            sensor_value = int(0xffffff).to_bytes(3,'big')
    return sensor_value
```

Figure 5.17: Update LoRa buffer function.

The Micropython code containing all of the functions necessary for ensuring this is included in the appendix.B.

For the time and date, the UNIX timestamp was used which gives the maximum information on 4 bytes only.
In this way, we are being more precise about the information sent and we are sending more or less on a regular basis.

Throughput

Evaluating the throughput in terms of the maximum data rate with which the LoRa packets can be transmitted.
Below is the equation representing the Data Rate:

$$DR = SF * \frac{BW}{2^{SF}} * \frac{4}{(4 + CR)} \tag{10}$$

CR represents the coding rate

$$CR = \frac{4}{4 + n} with n \in 1,2,3,4 \tag{11}$$

.
DR determines how fast bytes are transmitted.
If DRincreases (make the bandwidth wider or the spreading factor lower), bytes are transmitted in a shorter time.

Mode	Data Rate	Bit Rate (bit/sec))
SF7/125KHz	DR5	5470
SF8/125KHz	DR4	3125
SF9/125KHz	DR3	1760
SF10/125KHz	DR2	980
SF11/125KHz	DR1	440
SF12/125KHz	DR0	250

Table 5.4: Bit rates for different SF.

The ADR mechanism makes a trade-off by decreasing the DR (raising SF) in order to cover more area, so that End nodes positioned all over the city, even those far from the gateway can communicate their measurements, at the expense of TOA.

Chapter 6

Conclusion and Improvements

6.1 Conclusion

The low-cost environmental system designed is intended to sample environmental data such as PM (PM2.5 and PM10), temperature, and humidity for outdoor monitoring applications.
Using LoRa technology, covering a big area with low energy consumption and low-cost sensors, this system is well suited to be distributed in the cities.
The measurements showed that LoRaWAN could achieve acceptable ranges of data communications which were 2210 meters for LOS positions and 2600 meters for NLOS positions using respectively DR2 and DR3.
Therefore, there is still the ability to cover more area while sending a total of 33 bytes of the data payload.
The Packet Delivery Ratio (PDR) ranges from 100% to 73%. Such a result shows that the performance of a system using a LoRaWAN network is highly dependent on various parameters such as the Spreading Factor (SF) and Coding Rate (CR). Not only that, but the environment has a significant impact on data transmission.

6.2 Improvements

This project may benefit from some future enhancements.
One possibility is that we construct our own Network Server in order to evade The Things Network's fair use policy.
The fair use policy places various constraints on both uplinks and downlinks in terms of Time On Air and message count.

Using our own Network Server could greatly improve system performance.
Another possible improvement could be the use of an Application Server that processes the data messages received from end devices. The collected data can be interpreted by applying techniques such as machine learning and artificial intelligence to solve various problems.
An improvement regarding the lack of feedback from the part of the gateway could be to exploit the GPS data (GPSlt, GPSlg) sent by the End node in order to determine the distance separating the device from the Gateway. The distance should be compared to the maximum reachable range by the End device using a certain Spreading Factor. If the packet can't reach the destination, this packet won't be sent before setting the needed Spreading Factor for such a range.
This should allow for achieving a better Packet Delivery Ratio (PDR).

Appendix A

Decoder Function

```
function decodeUplink(input) {
  var data={};
    function hex2bin(hex) {
       return (parseInt(hex, 16).toString(2)).padStart(8, '0');
    }
    function dec2hex(dec){
      let heex="";
      dec.forEach(a=>{
        if (a<10){
          a="0"+a;
        }
        heex+=a.toString(16);
      });
      return heex;
    }
    function checkf2one(word){  //check if conversion is needed
      let wordObj=[];
      let convert=true;
      //convert word to object
      for (i=0;i<word.length;i++){
      wordObj.push(word[i]);
      }
     wordObj.forEach(a=>{
       if (a!=="f".toUpperCase()&&a!=="f"){
          convert=false;
       }
      });
      return convert;
    }
    function f2one(word){  //convert f to 1
      let final="";
        for (i=0;i<word.length;i++){ final=final+1;}
      return final;
    }
```

```
    function checkpm(pm){
      var check=checkf2one(pm);
      if (check ){
        pm=f2one(pm);
      }
      else{
     pm=parseInt(pm,16) ;
    }
    return pm ;
    }
    let timesTamp = input.bytes.slice(0,4);
    let timesTampHexa=dec2hex(timesTamp);
    var binaire=hex2bin(timesTampHexa);
    var decimal = parseInt(binaire, 2);
    var date = new Date(decimal * 1000);
    var finalDate = date.toLocaleDateString("en-US");
    finalDate+=` Time ${date.getHours()}:${date.getMinutes()}:${date.getSeconds()}`;
data.timesTamp=finalDate;
  var boardIdBytes= input.bytes.slice(4,6);
  var boardIdHexa=dec2hex(boardIdBytes);
  var check=checkf2one(boardIdHexa);
  if (check){
   boardIdDecim=f2one(boardIdHexa);
  }else{
      boardIdDecim=parseInt(boardIdHexa,16)  ;
  }
data.boardId=boardIdDecim;
  var temperatureone =Number(input.bytes.slice(6,7));
  if (temperatureone<10){
    temperatureone="0"+temperatureone;
  }
  temperatureone=temperatureone.toString(16);
  var temperaturetwo=Number(input.bytes.slice(7,8));
```

```
   if (temperaturetwo<10){
    temperaturetwo="0"+temperaturetwo;
  }
  temperaturetwo=temperaturetwo.toString(16);
  var temperature=temperatureone+temperaturetwo;
  var check=checkf2one(temperature)
  if (check){
   temperature =f2one(temperature);
  }
  else{
   temperature=parseInt(temperature,16);
   temperature=(temperature/10)-40;
   temperature=Number(temperature.toFixed(1))
  }
data.temperature=temperature;
 var humidity = input.bytes.slice(8,9);
 var humidityDec=dec2hex(humidity);
 var check=checkf2one(humidityDec)
 if (check ){
   humidity=f2one(humidityDec);
 }else{
   humidity = humidity/2 ;
   humidity=humidity.toFixed(1);
 }
data.humidity =humidity;
   var atmpressone= Number(input.bytes.slice(9,10));
   atmpressone=atmpressone.toString(16);
   var atmpresstwo=Number(input.bytes.slice(10,11));
   atmpresstwo=atmpresstwo.toString(16);
   var atmpresshexadecimal=atmpressone+atmpresstwo;
   var atmpressdecimal="";
   var check=checkf2one(atmpresshexadecimal)
```

```
  if (check){
      atmpressdecimal=f2one(atmpresshexadecimal);
  }
  else{
   atmpressdecimal=parseInt(atmpresshexadecimal,16) ;
   atmpressdecimal=atmpressdecimal/10;
  }
data.atm=atmpressdecimal;
 var pm10bytesone=Number(input.bytes.slice(11,12));
  pm10bytesone=pm10bytesone.toString(16);
    if(pm10bytesone.length===1){
    pm10bytesone="0"+pm10bytesone  ;
    }
 var pm10bytestwo=Number(input.bytes.slice(12,13));
    pm10bytestwo=pm10bytestwo.toString(16);
      if(pm10bytestwo.length===1){
      pm10bytestwo="0"+pm10bytestwo  ;
      }
  var pm10_1=pm10bytesone+pm10bytestwo;
  var pm10_1=checkpm(pm10_1)
data.pm10_1=pm10_1;
  var pm10bytesthree=Number(input.bytes.slice(13,14));
    pm10bytesthree=pm10bytesthree.toString(16);
      if(pm10bytesthree.length===1){
       pm10bytesthree="0"+pm10bytesthree  ;
        }
  var pm10bytesfour=Number(input.bytes.slice(14,15))
     pm10bytesfour=pm10bytesfour.toString(16)
       if(pm10bytesfour.length===1){
      pm10bytesfour="0"+pm10bytesfour
     }
  var pm10_2=pm10bytesthree+pm10bytesfour
  var pm10_2=checkpm(pm10_2)
data.pm10_2=pm10_2
```

```
    var pm10bytesfive=Number(input.bytes.slice(15,16))
      pm10bytesfive=pm10bytesfive.toString(16)
        if(pm10bytesfive.length===1){
      pm10bytesfive="0"+pm10bytesfive
      }
    var pm10bytessix=Number(input.bytes.slice(16,17))
      pm10bytessix=pm10bytessix.toString(16)
        if(pm10bytessix.length===1){
      pm10bytessix="0"+pm10bytessix
      }
      var pm10_3=pm10bytesfive+pm10bytessix
        pm10_3=checkpm(pm10_3)
data.pm10_3=pm10_3
      var pm10bytesseven=Number(input.bytes.slice(17,18))
        pm10bytesseven=pm10bytesseven.toString(16)
          if(pm10bytesseven.length===1){
      pm10bytesseven="0"+pm10bytesseven
      }
      var pm10byteseight=Number(input.bytes.slice(18,19))
        pm10byteseight=pm10byteseight.toString(16)
          if(pm10byteseight.length===1){
          pm10byteseight="0"+pm10byteseight
        }
      var pm10_4=pm10bytesseven+pm10byteseight
      pm10_4=checkpm(pm10_4)
data.pm10_4=pm10_4-
      var pm25bytesone=Number(input.bytes.slice(19,20))
          pm25bytesone=pm25bytesone.toString(16)
        if(pm25bytesone.length===1){
        pm25bytesone="0"+pm25bytesone
      }
      var pm25bytestwo=Number(input.bytes.slice(20,21))
      pm25bytestwo=pm25bytestwo.toString(16)
```

```
    var pm25bytesseven=Number(input.bytes.slice(25,26))
    pm25bytesseven=pm25bytesseven.toString(16)
        if(pm25bytesseven.length===1){pm25bytesseven="0"+pm25bytesseven}
    var pm25byteseight=Number(input.bytes.slice(26,27))
    pm25byteseight=pm25byteseight.toString(16)
        if(pm25byteseight.length===1){ pm25byteseight="0"+pm25byteseight}
    var pm25_4=pm25bytesseven+pm25byteseight
      pm25_4=checkpm(pm25_4)
 data.pm25_4=pm25_4
    var pregplg = input.bytes.slice(27,30)
    var pregplgDec=dec2hex(pregplg)
    var check=checkf2one(pregplgDec)
    var gplgdec=""
    if (check){gplgdec=f2one(pregplgDec)}
    else{
    gplgdec=parseInt(pregplgDec,16)
    gplgdec=(gplgdec/10000)-180
    gplgdec=Number(gplgdec.toFixed(4))
    }
 data.gplg= gplgdec
   var pregplt = input.bytes.slice(30,33)
   var pregpltDec=dec2hex(pregplt)
   var check=checkf2one(pregpltDec)
   var gpltdec=""
   if (check){gpltdec=f2one(pregpltDec)}
   else{
   gpltdec = parseInt(pregpltDec,16)
   gpltdec = (gpltdec/10000)-180
   gpltdec=Number(gpltdec.toFixed(4))    }
 data.gplt = gpltdec
  return {
    data: data
  };
}
```

Appendix B

Lora Sensing Function

```
class SDHandler:
    current_date = ()
    write_buffer = ''
    buffer_size = 0
    fileHandler = ''
    sensor_buffer = {}
    def __init__(self, lorasettings, lora=True, buffer_limit=80, boardID=1010):
        self.current_date = time.localtime()
        self.fileHandler = get_file_handle(self.current_date, log=True)
        self.fileHandler.close()
        self.buffer_limit = buffer_limit
        self.boardID = boardID
        self.lora = lora
        self.loraFramesCounter = 0
        self.loraID = boardID.to_bytes(2,'big')

        self.loraObj = LoraHandler(lorasettings["LoraSettings"], lorasettings["dr"],
            lorasettings["adr"], lorasettings["Sleep_Time"])
        if lora:
            if self.loraObj.join_lora():
                print("lora joined")
            self.sensor_buffer = { 'T': {'value': 0, 'count':0},
                                   'H': {'value': 0, 'count':0},
                                   'A': {'value': 0, 'count':0},
                                   'P1_25': {'value': 0,'count':0},
                                   'P1_10': {'value': 0,'count':0},
                                   'P2_25': {'value': 0,'count':0},
                                   'P2_10': {'value': 0,'count':0},
                                   'P3_25': {'value': 0,'count':0},
                                   'P3_10': {'value': 0,'count':0},
                                   'P4_25': {'value': 0,'count':0},
                                   'P4_10': {'value': 0,'count':0},
                                   'P5_25': {'value': 0,'count':0},
                                   'P5_10': {'value': 0,'count':0},
```

```
                                    'P6_25': {'value': 0,'count':0},
                                    'P6_10': {'value': 0,'count':0},
                                    'P7_25': {'value': 0,'count':0},
                                    'P7_10': {'value': 0,'count':0},
                                    'P8_25': {'value': 0,'count':0},
                                    'P8_10': {'value': 0,'count':0},
                                    'GPS_LT':{'value': 0,'count':0},
                                    'GPS_LG':{'value': 0,'count':0}
                                }
        self.alarm = Timer.Alarm(self.LoraWrite, 30, periodic=True)
            def update_lora_buffer(self, sensor, measurement):
    self.sensor_buffer[sensor]['value'] += measurement
    self.sensor_buffer[sensor]['count'] += 1
def LoraWrite(self, alarm):
    if  self.sensor_buffer['T']['count']!=0:
        temp = self.sensor_buffer['T']['value'] / self.sensor_buffer['T']['count']  # after each
            30 seconds of measurements
        print('temp: {}'.format(temp))
        temp = LoraFormat(temp,'T')                                                  #return the
            average value in the right format
        print('temp: {}'.format(temp))
        self.sensor_buffer['T']['value'] = 0                                         #set to 0
            after each 30 sec
        self.sensor_buffer['T']['count'] = 0
    else:
        temp = LoraFormat(None,'T')
    if self.sensor_buffer['H']['count']!=0:
        humi = self.sensor_buffer['H']['value'] / self.sensor_buffer['H']['count']
        print('humi: {}'.format(humi))
        humi= LoraFormat(humi,'H')
        print('humi: {}'.format(humi))
        self.sensor_buffer['H']['value'] = 0
        self.sensor_buffer['H']['count'] = 0
```

```
else:
    humi = LoraFormat(None,'H')
     if self.sensor_buffer['A']['count']!=0:
    Atmpress = self.sensor_buffer['A']['value'] / self.sensor_buffer['A']['count']
    print('atmpress: {}'.format(Atmpress))
    Atmpress = LoraFormat(Atmpress, 'A')
    print('atmpress: {}'.format(Atmpress))
    self.sensor_buffer['A']['value'] = 0
    self.sensor_buffer['A']['count'] = 0
else:
    Atmpress = LoraFormat(None,'A')
if self.sensor_buffer['GPS_LT']['count']!=0:
    GPLt = self.sensor_buffer['GPS_LT']['value'] / self.sensor_buffer['GPS_LT']['count']
    print('GPlt: {}'.format(GPLt))
    GPLt = LoraFormat(GPLt, 'G')
    print('GPlt: {}'.format(GPLt))
    self.sensor_buffer['GPS_LT']['value'] = 0
    self.sensor_buffer['GPS_LT']['count'] = 0
else:

    GPLt = LoraFormat(None,'G')
    print('GPlt0: {}'.format(GPLt))
     if self.sensor_buffer['GPS_LG']['count']!=0:
    GPLg = self.sensor_buffer['GPS_LG']['value'] / self.sensor_buffer['GPS_LG']['count']
    print('GPlg: {}'.format(GPLg))
    GPLg = LoraFormat(GPLg, 'G')
    print('GPlg: {}'.format(GPLg))
    self.sensor_buffer['GPS_LG']['value'] = 0
    self.sensor_buffer['GPS_LG']['count'] = 0
else:
    GPLg = LoraFormat(None,'G')
    print('GPlg0: {}'.format(GPLg))
pm_array25 = [0, 0, 0, 0]
pm_array10 = [0, 0, 0, 0]
```

```
for x in range(4):
            if self.sensor_buffer['P{}_25'.format(x + 1)]['count']!=0:
                pm_array25[x] = round(self.sensor_buffer["P{}_25".format(x + 1)]['value']/ self
                    .sensor_buffer["P{}_25".format(x + 1)]['count'])
                print('pm25{}: {}'.format((x + 1), pm_array25[x]))
                pm_array25[x] = int.from_bytes(LoraFormat(pm_array25[x], 'P'), "big")
                print('pm25{}: {}'.format((x + 1), pm_array25[x]))
                self.sensor_buffer["P{}_25".format(x + 1)]['value'] = 0
                self.sensor_buffer["P{}_25".format(x + 1)]['count'] = 0
            else:
                pm_array25[x] = LoraFormat(None,'P')
            if self.sensor_buffer['P{}_10'.format(x + 1)]['count']!=0:
                pm_array10[x] = round(self.sensor_buffer["P{}_10".format(x + 1)]['value']/ self
                    .sensor_buffer["P{}_10".format(x + 1)]['count'])
                print('pm10{}: {}'.format((x + 1), pm_array10[x]))
                pm_array10[x] = int.from_bytes(LoraFormat(pm_array10[x], 'P'), "big")
                print('pm10{}: {}'.format((x + 1), pm_array10[x]))
                self.sensor_buffer["P{}_10".format(x + 1)]['value'] = 0
                self.sensor_buffer["P{}_10".format(x + 1)]['count'] = 0
            else:
                pm_array10[x] = LoraFormat(None,'P')
        pm_bytes25 = pack('!hhhh', pm_array25[0],pm_array25[1],pm_array25[2],pm_array25[3])
        pm_bytes10 = pack('!4h', pm_array10[0],pm_array10[1],pm_array10[2],pm_array10[3])
print(pm_bytes25)
        print(pm_bytes10)
        print("PM sensors formated")
        print('time: {}'.format(timeToUnix()))
        time_byte= timeToUnix().to_bytes(4,'big')
        print('time: {}'.format(time_byte))
        print('boardid: {}'.format(self.boardID))
        board_byte =self.boardID.to_bytes(2,'big')
        print('boardid: {}'.format(board_byte))
        self.buf= time_byte + board_byte + temp+ humi+ Atmpress+  pm_bytes25+ pm_bytes10+ GPLg+ GPLt
```

```
    print(self.buf)
    print ('Lora frame Length is: {}'.format(len(self.buf)))
    self.loraObj.lora_socket(self.buf)
    print('transmitted')
    self.loraFramesCounter = self.loraFramesCounter+1
    print('The number of LoRa frames sent:{} '.format(self.loraFramesCounter))
```

```
class LoraHandler:
    lorasettings = {}
    datarate = 3
    sleeptime = 0
    adaptivedr = True
    sensorsettings = {}
    def __init__(self, LoraSettings, dr, adr, Sleep_Time):
        self.lorasettings = LoraSettings
        self.datarate = dr
        self.adaptivedr = adr
        self.sleeptime = Sleep_Time

    def join_lora(self, force_join = False):
        lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868, adr=self.adaptivedr, tx_retries=0,
            device_class=LoRa.CLASS_A)
        # create an OTAA authentication parameters
        print('The ADR is: {}'.format(self.adaptivedr))
        self.dev_eui = ubinascii.unhexlify(self.lorasettings["dev_eui"])
        self.app_eui = ubinascii.unhexlify(self.lorasettings["app_eui"])
        self.app_key = ubinascii.unhexlify(self.lorasettings["app_key"])
        if not force_join:
           lora.nvram_restore()
            if not lora.has_joined() or force_join == True:
           # join a network using OTAA (Over the Air Activation)
           lora.join(activation=LoRa.OTAA,auth=(self.dev_eui, self.app_eui, self.app_key), timeout=0,
               dr=self.datarate)
           # wait until the module has joined the network
           while not lora.has_joined():
            print("trying to connect to LoRa Gateway")
            time.sleep(self.sleeptime)
            # saving the state
           lora.nvram_save()    if lora.has_joined():
            return True
```

```
def lora_socket(self, msg, confirmed= False):
    print("lora socket executed")
    # create a LoRa socket
    s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
    # set the LoRaWAN data rate
    s.setsockopt(socket.SOL_LORA, socket.SO_DR, self.datarate)
    # DR(5) --> SF7BW125 /DR(4) --> SF8BW125 /DR(3) --> SF9BW125 /DR(2) --> SF10BW125 /DR(1) -->
        SF11BW125 /DR(0) --> SF12BW125 /
    # choose confirmed or unconfirmed uplinks
    s.setsockopt(socket.SOL_LORA, socket.SO_CONFIRMED, confirmed)
    s.setblocking(True)
    s.send(msg)
    s.setblocking(False)
```

Bibliography

[1] Edoardo Giusto, Renato Ferrero, Filippo Gandino, Bartolomeo Montrucchio, Maurizio Rebaudengo, and Mingyang Zhang. «Particulate matter monitoring in mixed indoor/outdoor industrial applications: a case study». In: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2018, pp. 838–844 (cit. on p. 1).

[2] Zhirong Zhang, Xuetian Zhu, Zhijun Li, and Yong Zeng. «Analysis of the Impact of eMTC on Legacy LTE». In: June 2019, pp. 1133–1138. DOI: 10.1109/IWCMC.2019.8766733 (cit. on p. 2).

[3] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. «Low Power Wide Area Networks: An Overview». In: *IEEE Communications Surveys and Tutorials* 19.2 (2017), pp. 855–873. DOI: 10.1109/COMST.2017.2652320 (cit. on pp. 2, 3).

[4] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. «A comparative study of LPWAN technologies for large-scale IoT deployment». In: *ICT Express* 5.1 (2019), pp. 1–7. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.12.005. URL: https://www.sciencedirect.com/science/article/pii/S2405959517302953 (cit. on pp. 3, 5–7).

[5] Jie Ding, Mahyar Nemati, Chathurika Ranaweera, and Jinho Choi. «IoT Connectivity Technologies and Applications: A Survey». In: *IEEE Access* PP (Apr. 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.2985932 (cit. on pp. 3–6).

[6] Niemah I. Osman and Esra B. Abbas. «Simulation and Modelling of LoRa and Sigfox Low Power Wide Area Network Technologies». In: *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*. 2018, pp. 1–5. DOI: 10.1109/ICCCEEE.2018.8515816 (cit. on p. 3).

[7] Ala Khalifeh, Khaled Aldahdouh, Khalid Darabkh, and Waleed Al-Sit. «A Survey of 5G Emerging Wireless Technologies Featuring LoRaWAN, Sigfox, NB-IoT and LTE-M». In: Mar. 2019, pp. 561–566. DOI: 10.1109/WiSPNET45539.2019.9032817 (cit. on p. 3).

[8] Marco Allegretti and Luca Mattiauda. «LoRa©: applications and validations in complex urban environment». In: 2020 (cit. on pp. 4, 8, 11, 17, 21, 22).

[9] Haider A. H. Alobaidy, Mandeep Jit Singh, Rosdiadee Nordin, Nor Fadzilah Abdullah, Cheong Gze Wei, and Marvin Liong Siang Soon. «Real-World Evaluation of Power Consumption and Performance of NB-IoT in Malaysia». In: *IEEE Internet of Things Journal* 9.13 (2022), pp. 11614–11632. DOI: 10.1109/JIOT.2021.3131160 (cit. on p. 4).

[10] Roberto Fantini. «Roberto Fantini». In: (2020) (cit. on p. 4).

[11] João P.D. Faria, José A.N. Pombo, Maria R.A. Calado, and Sílvio J.P.S. Mariano. «Development of an IoT communication module for energy sharing communities management and monitorization». In: *2022 IEEE International Conference on Environment and Electrical Engineering and 2022 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*. 2022, pp. 1–5. DOI: 10.1109/EEEIC/ICPSEurope54979.2022.9854688 (cit. on p. 5).

[12] Seyed Mousavi, Ahmad Khademzadeh, and Amir Rahmani. «The role of low-power wide-area network technologies in Internet of Things: A systematic and comprehensive review». In: *International Journal of Communication Systems* 35 (Feb. 2022). DOI: 10.1002/dac.5036 (cit. on p. 5).

[13] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. «A survey on LPWA technology: LoRa and NB-IoT». In: *ICT Express* 3.1 (2017), pp. 14–21. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.03.004. URL: https://www.sciencedirect.com/science/article/pii/S2405959517300061 (cit. on pp. 5, 6).

[14] Ivo Andrić, Adrijana Vrsalović, Toni Perković, M. Čuvić, and Petar Šolić. «IoT approach towards smart water usage». In: *Journal of Cleaner Production* 367 (Sept. 2022), p. 133065. DOI: 10.1016/j.jclepro.2022.133065 (cit. on p. 6).

[15] tech-papers-and-guides. «lora-and-lorawan». In: *lora-developers.semtech.com* (). URL: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan (cit. on pp. 8, 17).

[16] ETSI. «Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW». In: *European Telecommunications Standards Institute* 2 (2017) (cit. on p. 9).

[17] Jansen C. Liando, Amalinda Gamage, Agustinus W. Tengourtius, and Mo Li. «Known and Unknown Facts of LoRa: Experiences from a Large-Scale Measurement Study». In: 15.2 (Feb. 2019). ISSN: 1550-4859. DOI: 10.1145/3293534. URL: https://doi.org/10.1145/3293534 (cit. on p. 9).

[18] Akram Jebril, Aduwati Sali, Alyani Ismail, and Mohd Rasid. «Overcoming Limitations of LoRa Physical Layer in Image Transmission». In: *Sensors* 18 (Sept. 2018), p. 3257. DOI: 10.3390/s18103257 (cit. on pp. 9, 10).

[19] Martin Bor and Utz Roedig. «LoRa Transmission Parameter Selection». In: *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2017, pp. 27–34. DOI: 10.1109/DCOSS.2017.10 (cit. on p. 9).

[20] Minar El-Aasser, Tallal Elshabrawy, and Mohamed Ashour. «Joint Spreading Factor and Coding Rate Assignment in LoRaWAN Networks». In: *2018 IEEE Global Conference on Internet of Things (GCIoT)*. 2018, pp. 1–7. DOI: 10.1109/GCIoT.2018.8620147 (cit. on p. 10).

[21] *Lora Packet Format, lora Alliance*. http:lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf (cit. on p. 11).

[22] Fredrik Mårlind and Ismail Butun. «Activation of LoRaWAN End Devices by Using Public Key Cryptography». In: *2020 4th Cyber Security in Networking Conference (CSNet)*. 2020, pp. 1–8. DOI: 10.1109/CSNet50428.2020.9265530 (cit. on pp. 12, 13).

[23] *Otaa activation, Microship Company*. http:microchipdeveloper.com/lora:join-types (cit. on p. 13).

[24] Samuel Pierre, Michel Barbeau, and E. Kranakis. *Ad-Hoc, Mobile, and Wireless Networks*. Oct. 2003 (cit. on p. 14).

[25] Martijn Saelens, Jeroen Hoebeke, Adnan Shahid, and Eli De Poorter. «Impact of EU Duty Cycle and Transmission Power Limitations for Sub-GHz LPWAN SRDs: An Overview and Future Challenges». In: *EURASIP J. Wirel. Commun. Netw.* 2019.1 (Dec. 2019), pp. 1–32. ISSN: 1687-1472. DOI: 10.1186/s13638-019-1502-5. URL: https://doi.org/10.1186/s13638-019-1502-5 (cit. on p. 14).

[26] ETSI. «Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz Part 2: Harmonised Standard for access to radio spectrum for nonspecific radio equipment». In: *European Telecommunications Standards Institute* (2018). URL: https://https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_30/en_30022002v030201v.pdf (cit. on p. 14).

[27] LoRa. «What are LoRa and LoRaWAN?» In: *LoRa technical documents* (). URL: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/ (cit. on p. 14).

[28] Biswajit Paul. «An Overview of LoRaWAN». In: *WSEAS TRANSACTIONS ON COMMUNICATIONS* 19 (Jan. 2021), pp. 231–239. DOI: 10.37394/23204.2020.19.27 (cit. on p. 15).

[29] *LoRaWAN network topology*. http:medium.com/coinmonks/lpwan-lora-lorawan-and-the-internet-of-things-aed7d5975d5d (cit. on pp. 15–17).

[30] *Understanding LoRaWAN® End Devices CHRIS FRUCI*. http://https://radiobridge.com/blog/understanding-lorawan-end-devices. Accessed: 2010-09-30 (cit. on p. 15).

[31] *Gateways*. https://https://www.thethingsnetwork.org/docs/gateways// (cit. on p. 16).

[32] Phui San Cheong, Johan Bergs, Chris Hawinkel, and Jeroen Famaey. «Comparison of LoRaWAN Classes and their Power Consumption». In: Nov. 2017. DOI: 10.1109/SCVT.2017.8240313 (cit. on p. 18).

[33] Hassan Noura, Tarif Hatoum, Ola Salman, Jp A. Yaacoub, and Ali Chehab. «LoRaWAN Security Survey: Issues, Threats and Possible Mitigation Techniques». In: *Internet of Things* (Oct. 2020). DOI: 10.1016/j.iot.2020.100303 (cit. on p. 18).

[34] Mukarram A. M. Almuhaya, Waheb A. Jabbar, Noorazliza Sulaiman, and Suliman Abdulmalek. «A Survey on LoRaWAN Technology: Recent Trends, Opportunities, Simulation Tools and Future Directions». In: *Electronics* 11.1 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11010164. URL: https://www.mdpi.com/2079-9292/11/1/164 (cit. on p. 19).

[35] Emekcan Aras, Gowri Ramachandran, Piers Lawrence, and Danny Hughes. «Exploring the Security Vulnerabilities of LoRa». In: June 2017, pp. 1–6. DOI: 10.1109/CYBConf.2017.7985777 (cit. on pp. 19, 20).

[36] Hassan Noura, Tarif Hatoum, Ola Salman, Jean-Paul Yaacoub, and Ali Chehab. «LoRaWAN security survey: Issues, threats and possible mitigation techniques». In: *Internet of Things* 12 (2020), p. 100303. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100303. URL: https://www.sciencedirect.com/science/article/pii/S2542660520301359 (cit. on p. 20).

[37] F. Cuomo, Manuel Campo, Alberto Caponi, Giuseppe Bianchi, Giampaolo Rossini, and Patrizio Pisani. «EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations». In: Oct. 2017, pp. 1–8. DOI: 10.1109/WiMOB.2017.8115779 (cit. on p. 21).

[38] Domenico Garlisi, Alessio Martino, Jad Zouwayhed, Reza Pourrahim, and F. Cuomo. «Exploratory approach for network behavior clustering in LoRaWAN». In: *Journal of Ambient Intelligence and Humanized Computing* (Mar. 2021). DOI: 10.1007/s12652-021-03121-z (cit. on p. 21).

[39] Rachel Kufakunesu, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. «A Survey on Adaptive Data Rate Optimization in LoRaWAN: Recent Solutions and Major Challenges». In: *Sensors* 20.18 (2020). ISSN: 1424-8220. DOI: 10.3390/s20185044. URL: https://www.mdpi.com/1424-8220/20/18/5044 (cit. on pp. 21, 31).

[40] *ADR in The Things Stack*. https://https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate/ (cit. on pp. 21, 22).

[41] Jetmir Haxhibeqiri, Eli De Poorter, Ingrid Moerman, and Jeroen Hoebeke. «A Survey of LoRaWAN for IoT: From Technology to Application». In: *Sensors* 18 (Nov. 2018), p. 3995. DOI: 10.3390/s18113995 (cit. on p. 22).

[42] Hassan Noura, Tarif Hatoum, Ola Salman, Jean-Paul Yaacoub, and Ali Chehab. «LoRaWAN security survey: Issues, threats and possible mitigation techniques». In: *Internet of Things* 12 (2020), p. 100303. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100303. URL: https://www.sciencedirect.com/science/article/pii/S2542660520301359 (cit. on p. 23).

[43] *Semtech Gateway*. https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rg1xx-lorawan-gateway-wi-fi-ethernet-optional-lte-us-only (cit. on p. 25).

[44] *Laird's Sentrius RG1xx*. https://www.lairdconnect.com/documentation/868-mhz-compatible-antennas-rg186-datasheet/ (cit. on p. 25).

[45] *Lopy datasheet*. https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf (cit. on pp. 26, 27).

[46] *PM sensor*. https://sps.honeywell.com/us/en (cit. on p. 27).

[47] *DHT datasheet*. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf (cit. on p. 27).

[48] *BME sensor*. https://pdf1.alldatasheet.com/datasheet-pdf/view/1132060/BOSCH/BME280.html (cit. on p. 27).

[49] *gps datasheet*. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-mini-gps-pa1010d-module.pdf (cit. on p. 27).

[50] *Network Server*. https://www.thethingsindustries.com/docs/getting-started/what-is-tts// (cit. on p. 29).

[51] *THe Things Network Server*. https://https://www.thethingsnetwork.org/docs/network/architecture/ (cit. on p. 29).

[52] *LorAWAN Limitations*. https://www.thethingsnetwork.org/docs/lorawan/limitations/ (cit. on p. 29).

[53] *The Thnigs Network Server Limitations*. https://www.thethingsnetwork.org/docs/lorawan/limitations/ (cit. on p. 30).

[54] *The Thnigs Network Server frequency plans*. https://www.thethingsnetwork.org/docs/lorawan/limitations/ (cit. on p. 32).

[55] *TTN frequency plan*. https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/ (cit. on p. 32).

[56] *RSSI and SNR, LoRa*. https://lora.readthedocs.io/en/latest/ (cit. on p. 32).

[57] Mohammed Alenezi, Kok Keong Chai, Yue Chen, and Shihab Jimaa. «Ultra-dense LoRaWAN: Reviews and challenges». In: *IET Communications* 14.9 (2020), pp. 1361–1371 (cit. on p. 42).

[58] Mohammed Alenezi, Michael Chai, Yue Chen, and Shihab Jimaa. «Ultra-dense LoRaWAN: Reviews and challenges». In: *IET Communications* 14 (June 2020). DOI: 10.1049/iet-com.2018.6128 (cit. on p. 43).

[59] *TTN air time online calculator*. https://www.thethingsnetwork.org/airtime-calculator/ (cit. on p. 43).