POLITECNICO DI TORINO

Master Degree Course in ICT for Smart Societies

Master thesis

Design and development of a platform for workers' security monitoring in Industry 4.0 scenarios

Supervisors
Prof. Edoardo Patti
Ing. Saverio De Vito (ENEA)
Ing. Fabrizio Formisano (ENEA)
Ing. Sergio Ferlito (ENEA)

Candidate
Jacopo Braccio

Academic Year 2021-2022

To my beloved "Zietta".

Not a day went by without thinking about you.

Summary

This thesis presents the “SALVO” project, designed and developed with ENEA (Italian National Agency for New Technologies and Sustainable Development) and STMicroelectronics, that using IoT and Cloud Computing, aims at creating a distributed environment monitoring platform to improve workers’ security in Industrial environment. Its core component is a multi-sensor wearable device able to measure temperature, VOC, air composition, inertial parameters, sound, and location (both indoor and outdoor). The technology chosen for “SALVO” is LoRaWAN, through which the device sends, via radio frequencies in the unlicensed EU868 ISM band, two types of messages: a low priority telemetry message, used to construct an historical trend of the environmental conditions and a high priority message, triggered when some given thresholds are exceeded and representing a dangerous situation for the operator. A custom coding is introduced which allows to send all the information the node can provide in a smaller packet with respect to the standard coding, resulting in better battery performances and delivery rates in harsh environments. The radio messages sent by the node are demodulated and converted into an IP packet by a gateway and then forwarded to a Network Server which manages the whole LoRaWAN network. Finally, an application server displays the data collected in a dashboard addressed to the security managers under the form of maps, heatmaps and tiles.

The proposed project allows for fast intervention when needed and allows the collection of environmental parameters that in future version of the platform can be used to perform targeted AI algorithms for predicting dangerous situation, possibly reducing injuries and saving lives.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Work is an essential part of every person. It allows to have a financial freedom, but also to develop skills and human connections. The average person will spend 90.000 hours at work over a lifetime, making clear that jobs can have a huge impact on the quality of life and health [1]. According to the International Labour Organization 2019 survey , 22.3% of the world employment, defined as persons of working age who were engaged in any activity to produce goods or provide services during the reference period, is in the industry sector, which consists of mining and quarrying, manufacturing and construction. Modern industries require to cope with big batches of high-quality products while respecting deadlines to keep the pace with an always increasing demand. The rush to produce may possibly lead to a crack in the equilibrium that should exist between operational efficiency and employee safety. Indeed, the number of accidents in industry is impressive: in 2019, 421.400 non-fatal injuries were recorded in USA and 503.790 in Italy, whereas the number of serious accidents (which mainly involve death) is 300 for USA and 974 for Italy [2] [3].

The causes of accidents can be classified into three categories:

1. Environmental causes: accidents happen because of the working environment. In particular, the triggering factors can be both natural, or man-made, and can include poor lighting, ambient temperature, air pollution, sound pollution.

2. Mechanical causes: accidents happen because of the machinery, equipment or failures. Common causes are power failure, explosions, broken machines.

3. Human causes: accidents happen because of human factors such as fatigue, stress, overexertion, carelessness.

Guaranteeing the absence of accidents is impossible, but proper planning and training can drastically reduce those negative numbers. Additionally, a new industrial revolution is taking place since the last decade, in which new technologies are exploited to improve the industrial sector and can have a huge impact in both in production and safety management.

1.1 Industry 4.0

The invention of the term "Industry 4.0" is commonly attributed to Henning Kegermann, Wolf-Dieter Lukas and Wolfgang Wahlster which presented the concept at the Hannover Fair in their work entitled on a mandate from the Research Union Economy-Science of the German Ministry of Education and Research. The term was used to cover two different meanings: as a synonym for a claimed "fourth industrial revolution", following those triggered by a steam-powered mechanization, electricity and information and communication technologies, and also as a label referring to a strategic plan pursued by Germany to promote computerization of manufacturing in order to strengthen its international competitive position [4] . In contrast with the previous industrial revolution, which were driven by a single technology invention, the ongoing forth revolution is based on 9 pillars defined by the Boston Consulting Group [5]:

1. *Big Data and Analytics*
   Big Data consists of 4 dimensions: Volume of data, Variety of Data, Velocity of generation of new data, Value of Data. The analysis of these large, heterogeneous, and complex dataset will be crucial for supporting organizational decision making. Big data analytics could be used in different areas such as

marketing (by understanding customers' preferences, trends, and other information) or maintenance of machinery, which help in reducing costs and improving safety.

2. *Autonomous Robots*

   Robots have been used in industry to carry out complex and tedious work that cannot be solved easily by human. They are nowadays becoming more autonomous, flexible and cooperative, and may eventually communicate one with another and work side by side with humans, possibly learning from them.

3. *Simulation*

   Simulation will play a supportive role in production. By exploiting real-time data, an almost identical virtual representation of reality can be created, which can include machines, products, and humans. Thanks to that, testing and optimization can be performed without any physical changes, reducing costs and the production failures.

4. *Horizontal and Vertical System integration*

   Cooperation and communication among all the actors of the production chain, such as companies, departments, and functions.

5. *Additive Manufacturing*

   Additive manufacturing, also known as 3D-printing, will be more used in production rather than just for prototyping or realizing custom objects. It allows the creation of small batches of customized products that offer construction advantages, such as complex, lightweight designs. High-performance, decentralized additive manufacturing systems will reduce transport distances and stock on hand [5].

6. *Augmented Reality*

   These are systems that, through a mobile device, add multimedia information to the reality already normally perceived by humans. In the future, these technologies will be used to provide information in real time to improve work processes and decision making, enhancing human-machine interaction and remoting control.

7. *Internet Of Things*
   The Internet of Things refers to a network of connected objects communicating using standard protocols among themselves (machine to machine interaction) or with humans (machine to human interaction). The Industrial Internet of Things devices' main task is to collect data via sensors on the environment in which they are placed and make use of them for real-time decision making, leading to increased efficiency and productivity.

8. *Cyber Security*
   Due to the higher digitalization level and the use of standard communication protocols, the risk for companies of being victims of cyberattacks grows. Therefore, it is mandatory to improve protection against cyber threats of industrial systems and manufacturing lines by investing in cybersecurity.

9. *Cloud Computing*
   Cloud computing is the way of virtualizing resources and services. There are three types of Cloud Computing: Infrastructure as a Service (IaaS) where the provider offers pay-as-you-go basic activities such as storage and computing capabilities; Platform as a Service (PaaS) where customers use hardware and software tools provided by vendor to deploy their custom applications; Software as a Service (PaaS) where software is made available to the user over the internet. The main advantages of Cloud Computing are flexibility, scalability, and accessibility.

As reported by Forbes [6], the expected outcomes of Industry 4.0 are reductions in downtime (35%-40%), improvements in production (15%-20%), improvements in quality (35%-40%), improvements in overall productivity (65%-70%) and improvements in asset utilization (35%-40%). Other expected outcomes include improvements in power utilization, overall cost reduction, avoidance of unnecessary capital expenses, and reduction in wastage, among others.

Figure 1.1 illustrates the size of the global smart factory market between 2019 and 2024. It is expected that the smart factory market will grow at a compound annual growth rate (CAGR) of 9.6 percent to reach around 244.8 billion U.S. dollars by 2024.

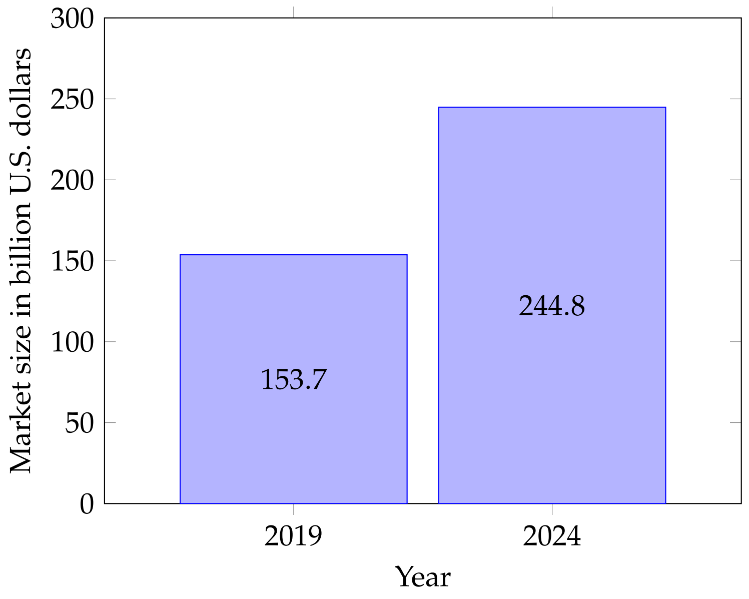


Figure 1.1. Industry 4.0: smart factory market size worldwide 2019/2024 (Statista, 2021)

1.2 Industry 4.0 in safety management application

The advent of Industry 4.0 does not only bring economic benefit, but it also aims at increasing safety levels, thus raising workers' satisfaction. Focina et al. conducted a systematic literature review to answer the question "What are the impacts did Industry 4.0 enabling technologies have on safety management?" by analyzing 68 papers published between 2010 and 2020. Figure 1.2 shows the temporal distribution of the papers reporting a significant increase in the number of publications from 2017 that highlights a growing interest in the topic since then.

Of the nine pillars mentioned earlier, the two most useful technologies for security management are the Industrial Internet of Thing and Cloud, whose use

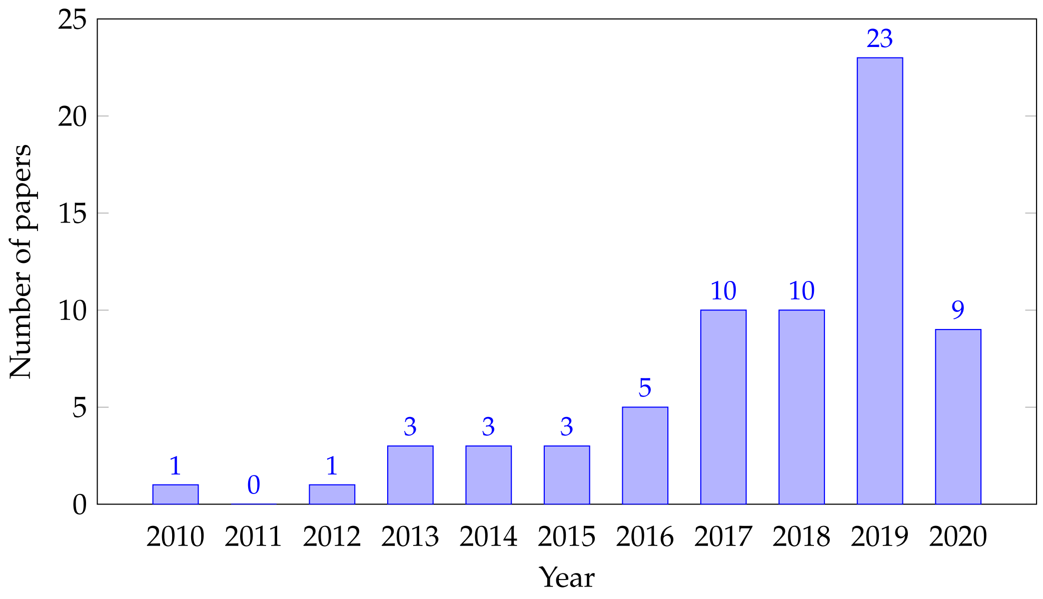


Figure 1.2. Temporal distribution of papers dealing with the improvements Industry 4.0 enabling technologies can have on safe management. (Focina et al, 2021)

represents 80% of the papers analyzed in [7]. Thanks to those two technologies new tools and frameworks for the implementation of safety management principles are introduced. Cloud computing and GPS are used in [8] to create a Physical Security Information Management (PSIM) system to provide custom and real-time instructions to operators in construction sites, raising operators' awareness of safety hazards and creating a safer environment as well. The use of sensors is one of the factors that made IoT one of the leading technologies in the modernization of safety management since it allows advanced monitoring systems by the remote and automatic collection of environmental data. For example, in [9], a system of sensors for measuring CO levels is introduced, with an alert time up to 7 times faster than the conventional CO sensors. The combination of sensors results in a Wireless Sensor Network (WSN) whose information is useful to study the trend of the environmental surroundings and possibly detect some anomalies. In [10], a Wireless Sensor Network and Building Information Model (the integration of database and geometry into a digital model which provides a visualized way in all construction lifecycle

management) are integrated into a unique system which enables the construction site to visually monitor the safety status via a spatial, colored interface and remove any hazardous gas automatically. A novel system that infers and renders safety context on construction sites by fusing data from portable devices, distributed sensing infrastructure and video is presented in [11].
Wearable IoT (WIoT) is a technological infrastructure that interconnects wearable sensors to enable monitoring human factors including health, wellness, behaviors, and other data useful in enhancing individuals' everyday quality of life [12]. Even though those kinds of devices are generally associated with mass-market electronic devices such as fitness trackers and smartwatches, they offer different advantages such as portability, non-intrusiveness, and closeness to the body, that make them suitable for some safety applications such as the ones discusses as follows.

Physiological Monitoring: industrial workers may face different health risks due to the often-difficult tasks that they have to face. Body data like hearth rate, blood pressure, breathing rate, can be automatically monitored using different sensors such as electrocardiogram sensor, electromyography sensor, skin temperature, blood pressure sensor, breathing sensor and so on. The captured data are then used to assess the health status and stress level of the workers as done in [13],[14] and [15], where an IoT wearable device, able to identify and prevent dangerous situations for operators, is introduced. It monitors heart rate and body temperature and, when those values exceed a given threshold, operators are notified by the system. Wearables equipped with accelerometer, gyroscope, and magnetometer are also able to analyze human motion. For instance, by equipping safety belts and vests with accelerometers, Fang et al. developed an accelerometer-based fall portent detection system [16].

Environmental Monitoring: the exposure not only to weather elements but also to hazardous materials and elements, might be a problem for industrial workers. It is now possible to use environmental sensors to measure a range of concerns including air quality, barometric pressure, carbon monoxide, capacitance, color, gas leaks, humidity, hydrogen sulfide, temperature, and light [17].

Proximity detection: especially in construction sites, there is a high rate of injuries due to contact. WSDs for proximity detection are capable of alerting construction personnel and equipment operators during hazardous proximity situation [18]. Many systems have been developer using different technologies such as ultrasonic-sensor and radar [19], RFID [20], GPS [21]. According to the task being carried out, most of these devices provide some form of warning signals to workers (visual, sound, light) when they are close to danger.

Location Tracking: useful to locate people and resources both in indoor environments using technologies such as Ultra-Wide-Band, and outdoor using GPS. For instance, Zhang et al. introduced a smart hard hat to solve the problem of people not wearing a hard hat on construction sites. The device is developed using an IoT-based architecture including multiple) sensors, GPS receiver, infrared rays, a smartphone companion application, and a web application for managers and guarantees real-time alarming, monitoring, and locating non-hard-hat use [22].
Summing up, safety levels can be increased by the key enabling technologies of Industry 4.0, whose introduction in this field foresees a reduction in injuries, illness, and an overall increase in workers' satisfaction.

1.3 Objective of the thesis

A new platform for safety management in industrial applications, that exploits IoT and Cloud, is presented. The platform is the core of the "*SALVO*" (nodo multiSensore per il monitoraggio degli Ambienti di LaVOro) project, Italian research whose participants involved are STMicroelectronics Srl, the University of Catania, and ENEA (the Italian national agency for new technologies, energy, and sustainable development) with whom the thesis has been carried out.
The "*SALVO*" project aims at realizing a multi-sensors wearable device for the continuous monitoring of the workers in the environment in which they operate. The novelty of the platform is that thanks to the high number of sensors, it enables to monitor physiological parameters (such as body motion), environmental parame-

ters (such as particular matters, VOC, gases, and air quality), and location (both indoor and outdoor), in a package that can be easily worn. New communication technologies are also exploited to guarantee high scalability and reliability while consuming low power for long-lasting battery devices. Nodes generate telemetry messages, useful to construct a trend of the overall working environment and run some AI algorithms, and alerts whenever a threshold indicating a possibly dangerous situation is exceeded. Those messages are received by an IoT Cloud device platform and their semantic value is enriched so that security clerks have a complete overview of what is happening and can intervene in a faster and more organized way when it is needed.
In this work, we focused on defining the architecture of "*SALVO*", choosing its components to create a functional prototype of the platform. A particular emphasis is put on the research of the optimal format of the messages sent by the nodes to maximize the probability of received alarms.

Chapter 2

Description and designing of *SALVO*

In this chapter, the *SALVO* project will be presented. By discussing the different reasons that justify its existence, the goal of the project is formally defined. Finally, we propose an original architecture of the platform, taking a closer look at each of its components.

2.1 Description of the project

Environmental causes are one of the main factors of accidents in the industry sector. For instance, exposure to Volatile Organic Compounds can, in the short term, cause irritation of the eyes and breathing difficulties, headaches, dizziness and visual disorder, and liver, kidney and central nervous system damages in the long term. Temperature and humidity can cause discomfort, thus reducing workers' performance. Recently, pandemic impacts have reshaped the approach to safety's work emphasizing the role of occupancy, ventilation, and eventually indoor air quality. Indoor Air Quality levels with specific reference to CO2 and PM levels are strong candidates to be viral infectivity danger proxies since their concentrations has been correlated with increased viral transmission in poorly ventilated environments which do not resort to recirculating air filtering [23] [24].

The *SALVO* project aims at creating an innovative technology realizing a multi-sensors wearable device, that enables the continuous monitoring of the worker and of the environment in which it operates. Devices can send both telemetry messages and alarms messages. The regular telemetry messages are periodic transmissions of the measurements recorded by the device's sensors, whereas the alarm messages are sent whenever a sensor is triggered by a rarer event, requiring, therefore, higher reliability. The information sent is received by an IoT Cloud platform addressed to qualified staff, for rapid intervention in case of danger or a data collection and analysis of the received parameters.
The application requires low power, low complexity, and low costs nodes, deployed on a scalable network, that can communicate over long distances and in different scenarios.
Wireless technologies such as wireless local area networks (WLANs), Bluetooth, and ZigBee are not adapted for a scenario requiring long-range transmissions. In contrast, wireless networks based on cellular technologies such as 2G, 3G, 4G, and LTE are used primarily for covering wide areas with high-speed communication resulting in excessive use of energy and higher costs as the network scales, since each device should have its contract with the mobile operator.
The requirements needed by the project are satisfied by an emerging wireless communication technology called low power wide area network (LPWAN).

2.2 Proposed architecture

LPWANs are resource-constrained networks (low transmission data rates and small packets sizes) and have the critical requirements for long battery life, extended coverage, high scalability capabilities, low device cost, and low deployment cost [25]. They provide long communication ranges (up to 40 km in rural zones and 5 km in urban environments [26]), with support of the simultaneous connection of thousands of end-devices to the infrastructure [27]. Additionally, the cost of a radio chipset is between 1$ and 5$ with an ideal battery life of the end device of several

years [28]. The usual packet size can range between 10 to 1000 Bytes at uplink speeds of up to 200 kbps [29].
LPWAN technologies can be deployed for a broad range of smart and intelligent applications as given in Table 2.1 [30].

Field	Applications
Smart Cities	Smart parking, air quality measurement, sound noise measurements, traffic light control, road toll control, smart lighting, waste management, utility meters, fire detection, elevator monitoring
Smart Environment	Water quality, air pollution reduction, forest fire, animal tracking, snow level monitoring, earthquake detection
Smart Water	Water quality, water leakage, river flood monitoring, swimming pool management, chemical leakage
Smart Metering	Smart electricity meters, gas meters, pipeline monitoring, warehouse monitoring
Smart Grid and Energy	Network control, load balancing, remote monitoring, windmills/solar power plants monitoring
Security and Emergencies	Perimeter access control, radiation levels, and explosive and hazardous gases detection
Automotive and Logistics	Insurance, security and tracking, car sharing management, item location, fleet tracking, smart trains and mobility as a service
Smart Manufacturing	Production line monitoring, machine health monitoring, preventive maintenance, energy management
Smart Agricolture and Farming	Temperature, humidity measurements, smart greenhouses, agricoltural automation and robotic, meteorological station network, livestock monitoring and tracking
Smart Homes	Energy and water use, temperature, humidity, fire/smoke detection, remote control of aplicances
eHealth	Healthcare wearables, patients surveillance, telemedicine, fall detection, tracking chronic disease

Table 2.1. Application of low power wide area networks (LPWANs).

The reasons why *SALVO* is a perfect application for LPWAN are listed below.

1. *Capacity and densification*

   At the beginning of the working shift, and for all its duration, each worker will be equipped with a multi-sensors node. Therefore, *SALVO* may involve a high number of simultaneously connected devices, whose support is one of the essential requirements for LPWANs. Furthermore, an LPWAN also ensures scalability, referred to the ability to seamlessly grow from a network of a small number of devices to massive numbers of devices without compromising the quality and provisioning of existing services [27], particularly useful for the security managers in charge of controlling the platform, who may not have broad technological knowledge.

2. *Coverage*

   SALVO is addressed to all the possible industry sectors. For instance, it could be used in a small paint production plant or a large construction site. For some scenarios, coverage may need to be provided for indoor or hard-to-reach locations like underground locations and basements. It is mandatory to have a reliable signal that allows both long-range as well as short-range communications. The use of the sub-GHz band helps most of the LPWANs to achieve robust communication because the lower frequencies of the sub-GHz band have better propagation characteristics as compared with the 2.4 GHz band and higher bands. Additionally, the low modulation rates used for LPWAN put more energy for each bit, and hence increase the coverage [25].

3. *Energy Efficient operations and Low Power Sources*

   Having a long-lasting battery is life is crucial for *SALVO* end nodes. Some scenarios would not allow workers to recharge the device, leaving them without the possibility of reporting dangerous situations. In LPWANs, the battery is expected to last over ten years without charging. LPWANs characteristics such as power-optimized radio network, a simplified network topology, frame sizes in the order of 50 bytes transmitted a few times per day at ultra-low speeds, and a mostly upstream transmission pattern that allows the devices to spend most of their time in low-energy deep-sleep mode, enable long battery lifetimes, possibly ten years of operation on a single coin-cell [31].

4. *Security and privacy*
   It is important to prevent malicious intrusions into the platform, possibly due to the high number of devices and their simplicity. The essential attributes of authorization, authentication, trust, confidentiality, data security, are supported by LPWANs. Over-the-Air Activation (OTAA) is a key facility to assure that end devices are not exposed to security risks over a prolonged duration. Devices perform a join-procedure with the network, during which a dynamic address is assigned and security keys are negotiated with the device. In most networks, strong encryption and authentication schemes such as advanced encryption suite (AES) are used for encrypting the application payload and the network admission requests [25].

5. *Reduced hardware complexity*
   A *SALVO* end node is a wearable device, so a small design and low complexity become an essential requirement. LPWAN's simplified and lightweight protocols reduce complexity in hardware design and lower device costs. The reduction of the hardware complexity reduces the power consumption of the device, without sacrificing too much performance, given by simple MCUs.

The fundamental LPWANs architecture requires wireless access and connectivity to the internet and the Cloud. Messages are sent on a specific radio link, whose frequency depends on the technology used, to a wireless access station and on to the IoT network. The access stations maintain the consistency of the radio link and interface with a gateway/concentrator that supports the access station protocols on one side and the network protocols on the other. The IoT network server is responsible for managing all the entities of the LPWAN and can be deployed locally, hosted in the cloud, or hosted by a third party. The network server has an impact in terms of scalability and reliability of the network. All the collected date are made available to users by an application server, to visualize and analyze them or perform AI algorithms.

Many LPWANs technologies exists both in the licensed and the unlicensed frequency bandwidth, with the leading ones being Sigfox, LoRa and NB-IoT.

Sigfox is a proprietary technology that operates in an unlicensed ISM band: in Europe, it operates at 868 MHz, in North America at 915 MHz, and 433 MHz in China. The end devices are connected to some base stations using a binary phase-shit keying (BPSK) modulation in the ultra-narrow band (100Hz) that gives low noise levels, low power consumptions, high receiver sensitivity and, therefore, larger area coverage. However, the use of ultra-narrow bands limits the maximum throughput to only 100bps. The number of messages over the uplink is bounded to 140 messages per day with a maximum payload length of 12 byes with 14 bytes of overhead. The number of downlink messages that a device can receive is limited to four per day, with a maximum payload of 8 bytes, meaning that it is not possible to receive an acknowledgement for every uplink message. To improve uplink communication reliability, a device transmits the message three times in different frequency and period assuring a message delivery rate of 95% [32].

LoRaWAN modulates the signals in the unlicensed sub-GHz ISM band using exploiting a proprietary spread spectrum technique. Like Sigfox, LoRaWAN operates at 868 MHz in Europe, in North America at 915 MHz, and 433 MHz in China. Different spreading factors are used to adapt the data rate and range tradeoff. The data rate is between 300bps and 50kbps according to the spreading factor used. Higher spreading factors corresponds to low data rate but longer coverage. The maximum payload length for each message is 243 bytes. Due to the regulation policy, LoRaWAN has 1% duty cycle, which can be translated to 36 sec/hour transmission per device per channel.

NB-IoT is a Narrow Band IoT technology standardized in 3GPP release 13. It can coexist with GSM and LTE under licensed frequency bands (700MHz, 800MHz, 900MHz). NB-IoT occupies a frequency band width of 200 KHz, which corresponds to one resource block in GSM and LTE transmission [33]. The data rate is limited to 200 kbps for the downlink and to 20 kbps for the uplink. The maximum payload size for each message is 1600 bytes. The technology can achieve 10 years of battery lifetime when transmitting 200 bytes per day on average [34]. The technical differences of the previous technologies are summarized in Table 2.2.

	Sigfox	**LoRaWAN**	**NB-IoT**
Technology	Proprietary	Proprietary	Open LTE
Frequency	Unlicensed ISM bands	Unlicensed ISM bands	Licensed LTE frequency
Maximum data rate	100bps	50kbps	200kbps
Maximum messages/day	140 Uplink, 8 Downlink	Unlimited	Unlimited
Maximum payload length	12 bytes Uplink, 8 bytes Downlink	243 bytes	1600 bytes
Duty cycle limit	Yes	Yes	No
Range	10km urban, 40km rural	5km urban, 20km rural	1km urban, 10km rural
Security	Low (AES-128)	Medium (AES-128)	High (LTE encryption)
Localization	Yes (RSSI)	Yes (TDOA)	No
Interference immunity	High	High	Low

Table 2.2. Technical overview Sigfox, LoRaWAN, NB-IoT

Given the differences among these LPWAN technologies, many factors should be considered when choosing one over the other. NB-IoT is preferred for applications that require guaranteed quality of service and low latency, whereas, for applications that are insensitive to the latency and don't need to send a large amount of data, Sigfox and LoRaWAN are a better choice. Due to the higher quality of the communication, battery life on NB-IoT devices is slightly inferior to its competitors. The major advantage of Sigfox is its large coverage: in Belgium, a country with a total surface area of approximately 30 500 km2, the Sigfox network deployment covers the entire country with only seven base stations [27]. By contrast, LoRa has a lower range of < 20km, and NB-IoT has the lowest coverage in the 10 km range. NB-IoT is also limited to LTE base stations, so it is not suitable for regions without LTE coverage and used mainly in deep indoor scenarios. Table 2.3 represents the cost associated with each technology [35], highlighting the higher cost effectiveness of Sigfox and LoRaWAN compared to NB-IoT.

	Spectrum Cost	**Deployment cost**	**End device cost**
Sigfox	Free	>4000€/base station	<2€
LoRaWAN	Free	>100€/gateway >1000€/base station	3-5€
NB-IoT	>500M€/MHz	>15000€/base station	20€

Table 2.3. Different costs of the major LPWANs technologies

LoRaWAN is the technology that best fits the requirements of *SALVO*: the use of an unlicensed band, the lower deployment costs, and the longer battery life of the end device make it a better alternative than NB-IoT [36], whereas the possibility of sending a longer payload at a faster data rate makes it a better alternative to Sigfox. Furthermore, the minimum 5 km range in an urban environment is sufficient for most of the industrial applications and can be easily increased by adding new gateways with a small expense.
Figure 2.1 shows the proposed architecture for *SALVO*, modeling the platform as a LoRa LPWAN. A LoRa network, and therefore *SALVO*, is made of four building blocks:

1. *End-nodes*: the element of the network that gathers sensor data and transmits them using Radio Frequency;

2. *Gateway*:it receives the RF messages sent by the end-nodes and forward them via high-bandwidth networks such as Ethernet or cellular networks;

3. *Network Server*: it is the core of the network. It gathers data from the gateway, filters out duplicate messages, sends downlink and acknowledgment mes-

sages, and adapts data rate to increase the battery life of the end nodes. *SALVO* will be "The Things Network"[1] (TTN) that is an open-source LoRaWAN network server suitable for distributed public and private global deployment as well as for small and local networks.

4. *Application Server*: uses the collected device for reaching the purpose of the application such as data visualization and analytics. *SALVO* uses an IoT Cloud platform called "Tago.io"[2] to create a reactive dashboard.

In the following sections, a closer look at LoRa and each of the components of *SALVO* is given.

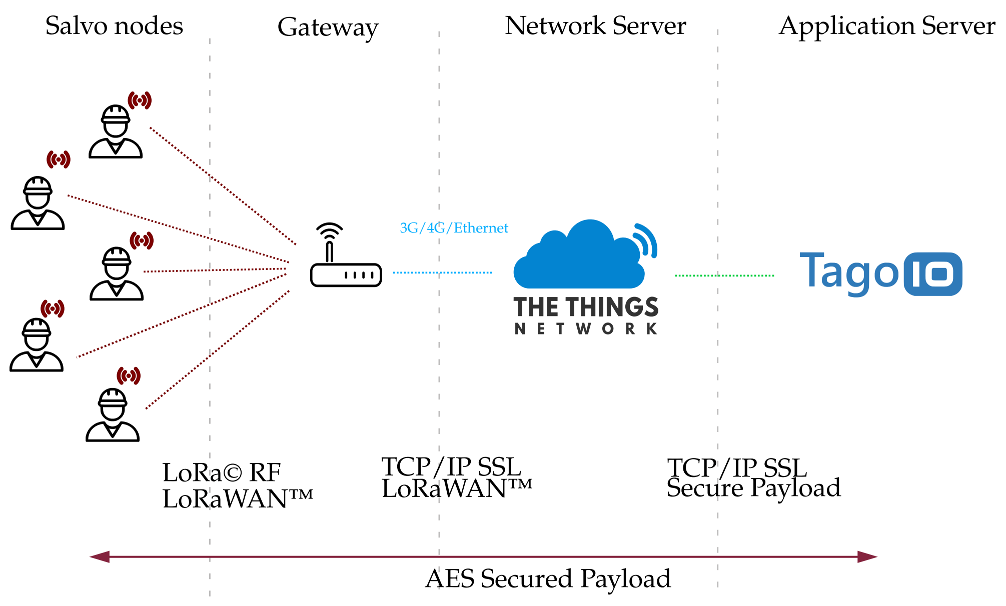


Figure 2.1. *SALVO* proposed architecture based on LoRa LPWAN technology.

2.3 LoRa and LoRaWAN

The technology was originally introduced by Cycleo in France but then acquired by the Semtech Corporation for $5 million in 2012. LoRa is the physical layer of the

[1]https://www.thethingsnetwork.org
[2]https://tago.io

protocol while LoRaWAN represents the MAC layer (Figure 2.2).
In this section, the fundamentals of LoRa and LoRaWAN useful to understand the main points of the protocol and architecture are presented.

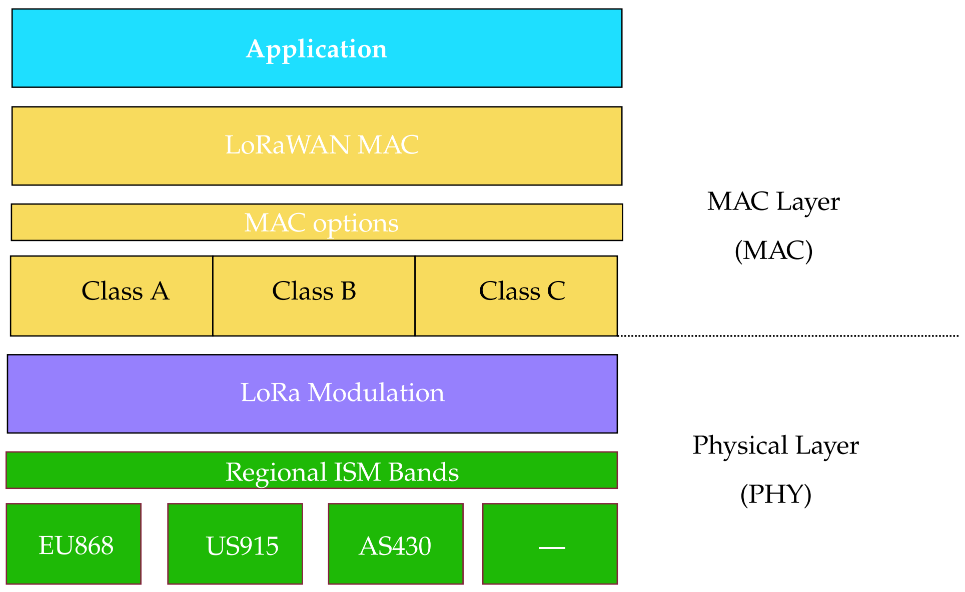


Figure 2.2. LoRa and LoRaWAN protocol stack.

2.3.1 LoRa physical layer

LoRa is the physical layer, or "bit" layer, of the LoRaWAN stack that manages modulation, power, radios, and signal conditioning. LoRa uses a proprietary modulation derived from the existing Chirp Spread Spectrum (CSS). CSS was first used in 1940 for military long-range communication by using sinusoidal waves that increase or decrease over time, called chirps, to encode data. Since they use all the entire fixed channel bandwidth for communication, they are robust in terms of interference. The number of bits that LoRa encodes in a symbol is a tunable parameter, called Spreading Factor. The SF controls the chirp rate and therefore the speed of the transmission. The higher the spreading factor, the slower the transmission but the longer is the transmission range. LoRa spreading factors are orthogonal among them, meaning that multiple frames can be sent simultaneously if each one of them is sent with a different SF.

The nominal bitrate used by LoRa Rb can be defined as [37]:

$$Rb = SF \cdot \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} \tag{2.1}$$

where:

- SF is the spreading factor, SF ∈ {7,8,9,10,11,12};
- BW is the bandwidth, BW ∈ {125,250,500} kHz;
- CR is the coding rate used for the Forward Error Correction (FEC) and represents the proportion of the transmitted bits that carries information, CR ∈ {1,2,3,4}.

The bitrate has an impact on the Time on Air (ToA) of the transmission, which is the time needed to transmit a packet. The ToA at a given SF is computed using the following equations [38]:

$$T_{symbol}(SF) = \frac{2^{SF}}{BW} \tag{2.2}$$

$$T_{preamble}(SF) = (L_{preamble} + 4.25) \cdot T_{symbol}(SF) \tag{2.3}$$

$$L_{payload} = 8 + [\frac{8B - 4SF + 28 + 16 - 20H}{4SF}] \cdot (CR + 4) \tag{2.4}$$

$$T_{payload}(SF) = L_{payload} * T_{symbol}(SF) \tag{2.5}$$

$$ToA(SF) = T_{preamble}(SF) + T_{payload}(SF) \quad [s] \tag{2.6}$$

where:

- B is the payload size in bytes;
- H is equal to zero if the header is present, 0 otherwise;

- $L_{payload}$ length in symbols of the payload;
- $L_{preamble}$ length in symbols of the preamble.

LoRa networks have a powerful feature called Adaptive Data Rate (ADR) that enables dynamically scalable capacity according to the condition of the network. ADR is performed by the network server and assigns to nodes that have a better communication channel, such as nodes close to the base station, a lower spreading factor (thus higher data rate) because of the signal confidence.
In [39], LoRaWAN regional parameters are specified:in Europe, LoRa operates at 868 MHz, regulated by the norm UE868MHz: the maximum transmission power must be below 14dBm, and the maximum duty cycle is ad 1%. Table 2.4 reports the LoRa parameters in the EU868 band. It can be noticed that the payload size varies according to the spreading factor used. This aspect will be crucial in Chapter 3 for the design of the optimal payload format.

Data Rate	**SF**	**Bandwidth (KHz)**	**Throughput (bps)**	**Payload (Bytes)**
DR6	SF7	250	11.000	222
DR5	SF7	125	5470	222
DR4	SF8	125	3125	222
DR3	SF9	125	1760	115
DR2	SF10	125	980	51
DR1	SF11	125	440	51
DR0	SF12	125	250	51

Table 2.4. LoRa parameters using the frequency of 868 MHz.

2.3.2 LoRaWAN MAC layer

LoRaWAN represents the MAC layer built on top of LoRa PHY. In contrast with the lower level, LoRaWAN is open and described in [40] by the LoRa Alliance, formed in 2015, that is a group of vendor and institution interested in spreading the technology. LoRaWAN is based on a star-of-stars topology, in which end devices

send/receive messages to/from one or more gateways, which in turn forward them to a dedicated network server in the Cloud via a backhaul (such as 4G LTE, Ethernet or Wi-Fi). A single device is not paired to just one gateway, but rather messages are simply sent on the wireless channel, assuming that at least one gateway will receive them and forward them to the network server. This allows improved network capabilities and range. The network server has the responsibility of filtering the duplicated messages and choose the most appropriate route, through which send downlink messages, and source of information when an end-node is associated with more than one gateway.
LoRaWAN is an asynchronous, ALOHA-based protocol, which means that end-nodes can transmit messages without knowing if other clients are simultaneously transmitting. Collisions only occur if more than one transmission use the same channel and spreading factor. If an acknowledgment is envisioned, a node will wait and retransmit the packet for a fixed number of times.
LoRaWAN defines three classes of devices [41], that balance latency and energy efficiency:

- *Class-A* is primarily intended for devices that have constraints on power consumption such as battery-based devices. This kind of device will spend most of its time in an idle state. Class-A optimizes power consumption by setting various receive delays during transmission, as shown in Figure 2.3. After an endpoint has sent a packet of data to a gateway, it will enter a sleep state until a receive delay timer expires (usually one second). Once the timer has expired, the endpoint opens a receive window (RX1), awaits a downlink message for some time, and then reenters the sleep state. If no downlink messages are received in the first receive window, the device will wake up again as another delay timer expires to listen to a second receive window (RX2). If no response is received during this second receive window, the device goes back into sleep mode until it needs to transmit again. Class-A device can't be woken up by the application, so it's not suitable for actuators.

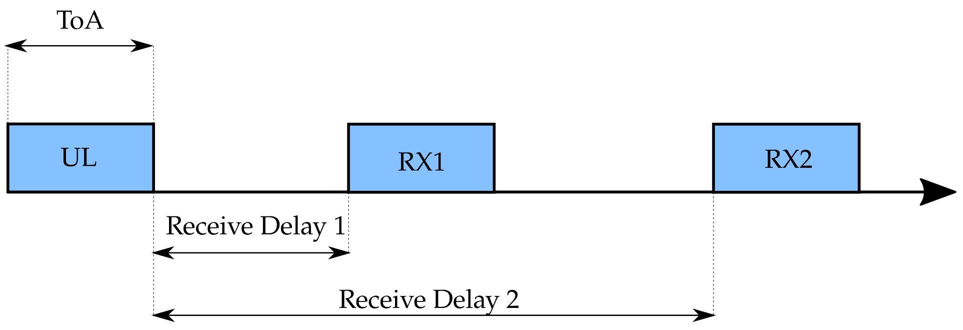


Figure 2.3. Class A reception windows

- *Class-B* balances power and latency. It offers regularly scheduled, fixed-time opportunities for an end device to receive downlink messages from the network. It relies on a beacon sent by the gateways at fixed intervals that synchronizes all the endpoints in the network. Once the beacon is received, the device opens a slot in which it can send and receive messages after which it goes back to idle mode.
- *Class-C* provides the best latency at the expense of energy consumption. These devices have a continuously open receive window, except for the time they are transmitting uplink messages. This makes them always reachable from the network and ideal for actuators having stricter delay requirements, but they need to be constantly powered by an external source such as connecting them to the power grid.

Two transmission modes are defined by LoRaWAN specifications: a confirmed mode where the end device expect to receive an ACK packet from the network server after each transmission and an unconfirmed mode where no ACK is sent by the network server, so the end device doesn't know if the packet has been correctly received. The impact on the performance of each mode, applied to *SALVO*, will be studied in Chapter 3.
LoRaWAN security encrypts data using AES128 model and an end device must be activated to participate in a LoRaWAN network. The activation process gives the following information to the device [42]:

- Device Address (*DevAddr*): a 32-bit ID use to identify the device. The first seven bits identify the network, whereas the remaining identify the device within the current network
- Application Identifier (*AppEUI*): a 64-bit application identifier that allows to uniquely identity the owner of the device globally.
- Network Session Key (*NwkSKey*): used by the end device and the network server to calculate and verify the Message Integrity Code of all the messages
- Application Session Key (*AppSKey*): use to encrypt and decrypt user's payload

There are two ways to proceed with the activation of a device: Over The Air Activation (OTAA) and Activation By Personalization (ABP). For OTAA, a *join-request* is sent by the device to join a LoRa network. A gateway will respond with a device address and an authentication token. During the join procedure, the AppSKey and NwkSKey will be derived. For the ABP, the device address and the security keys are hardcoded in the end device.

2.4 Multi-sensors node

The development of the multi-sensors node is the main goal of *SALVO*. The device will be low cost with a long-lasting battery in a package small enough to be worn easily during the working shift. The device is provided with different sensors (such as temperature and humidity, gas, and particular matter) whose data are going to be sent via LoRaWAN to an IoT Cloud Platform. A *SALVO* end-node can understand possible dangerous situations (mainly checking that all the measured parameters don't overpass some critical thresholds) and send alarms messages along with the position of the worker.
The development of the device is complex and requires lots of time, so just a prototype to assess the feasibility of the project will be presented in this work.
The prototype is made of two main components: a Master Board (Figure 2.4a) represented by a STEVAL-STRKT01 LoRa IoT tracker by STMicroelectronics, and

a Slave Board (Figure 2.4b) constituted of an STM32 MCU and different sensors. The two boards work independently and communicate periodically via the serial protocol UART. The main role of the Master Board is to ask for the measurements of the Slave Board every interval of time and send data to the LoRaWAN network. On the other hand, the Slave Board acquires measurements from various sensors and compares them with some predefined threshold. Its main functions are:

- Monitoring of temperature, humidity, and atmospheric pressure
- Monitoring CO_2, gases and particular matter concentrations
- Monitoring of the environmental sound pollution
- Detect falls of the operator wearing the device
- Localize the operator inside delimited areas
- Allow the operator to send an alarm not detectable by the device
- Warn the user of a dangerous situation via feedback such as light (LED), sound (buzzer) or physical (vibration)

The information on the measurements is completed with additional information about on the position of the operator. In addition to the GNSS position obtained thanks to the module GNSS Teseo-LIV3F, the device is also capable of being localized in indoor areas thanks to RTLS (Real-Time Locating System).
After the device is turned on, it starts the initialization of the MCUs and sensors. Once this phase is over, the acquisition of the parameters is done during regular intervals. All the components that support sleep mode are in that state until it's time to make a measurement, so the battery of the device is even more preserved. After the collection of the data, if no anomalies are found, information about the position is acquired and the packet is sent to the LoRaWAN network. After that, all the system goes back to sleep until the next capturing. On the contrary, a reduced payload is sent, to increase the probability of being delivered. A complete study about the format of the messages, the probability of being received, and the right

amount of time between each message sufficient to guarantee the reception of the alarms is done in Chapter 3.

(a) Master Board

(b) Slave Board

Figure 2.4. The two main components of the *SALVO* end node: the Master Board used to send data via LoRaWAN and the Slave Board used to take measurements.

A list of the sensors connected to the Slave Board is reported in Table 2.5.

Measurement	**Sensor**
Temperature	STTS22
Humidity	HTS221
Pressure	LPS22DF
Inertial	LMS6DSOX
VOC	GHT25S
CO_2	CMD7160
Particular Matter (PM_1, $PM_{2.5}$, PM_{10})	PMS7003

Table 2.5. List of sensors included in the current prototype of the *SALVO* node.

An additional gas sensors and sound sensors are still development, but will be included in the final version of the device.

2.5 Gateway

Gateways are the bridge that links the end device to the NS. Nodes use LoRa RF to connect to the gateway, while the gateway uses high bandwidth connections to

connect to the NS. All the gateways within the reach of the device will receive its message and forward it to the NS.
A gateway integrates a LoRa concentrator that is the part enabling the transmission and reception of LoRa packets and it's made of a baseband chip (provided by Semtech) and front-end radios. Through the use of packet forwarding software running in the background, the messages from the devices just received are sent to the NS.
LoRaWAN gateways can be grouped into two categories: single-channel and multi-channel. Single-channel gateways are less expensive and are very limited in terms of functionality. They can only receive one single SF and one channel at the same time, whereas multi-channel gateways can receive up to 10 channels and 6 SFs at the same time.
The prices of the gateways range between 100$ for the basic model, to more than 1000$ for more advanced models featuring a complete operating system and a resistant outdoor enclosure.
In this work, for the construction of the prototype of *SALVO*, the gateway we used is the *MultiTech Conduit® AP*[3] which is easy to deploy and good enough to cover buildings like hotels, convention centers and offices. It offers two development environments: mLinux for advanced configuration, and an AEP (Application Environment protocol) with a GUI for a more straightforward set-up. Ethernet is used as IP backhaul. The price of the gateway is around 300$.

[3]https://www.multitech.com/documents/publications/data-sheets/86002212.pdf

Figure 2.5. MultiTech Conduit® AP

2.6 Network server

The network server is the core of every LoRaWAN Network. It enables the connection, management, and monitoring of the end devices and gateways. The NS controls dynamically the parameter of the network to always guarantee the best performance and establishes a secure connection for the end-to-end data transport (from the nodes to the application in the Cloud and vice versa).
In general, all LoRaWAN network servers share the following features [16]:

- Device address checking
- Frame authentication and frame counter management
- Acknowledgment of receive messages when confirmation mode is set on the device
- Adapting data rates using ADR protocol
- Responding to all MAC layer requests coming from the device
- Forward uplink application payload to the appropriate application servers

- Queuing of downlink payload coming from any Application Server to any device connected to the network
- Forwarding join-request and join-accepted messages between the device and the join server

For the current phase of the development of *SALVO*, the network server used is "*The Thing Network*", actively maintained by "The Thing Industries" . It offers a free and user-friendly start with LoRaWAN technologies. TTN is a global collaborative Internet of Things ecosystem that creates networks, devices, and solutions using LoRaWAN. The Things Network runs The Things Stack Community Edition, which is a crowdsourced, open, and decentralized LoRaWAN network. This network is a great way to get started testing devices, applications, and integrations, and get familiar with LoRaWAN [43]. Once the proof of concept is fully functional, the network can be scaled to a commercial level which the price varies according to the services requested. A Fair Use Policy must be respected when using the community edition of TTN: the uplink airtime for each device is limited to 30 seconds per day and the downlink messages can't be more than 10 per day per node. Even though these limits would be strict in a real-world *SALVO* implementation, they are not a problem for the goals of this thesis.

2.7 Application server

The last piece of a working and fully complete LoRaWAN network is the Application Server. It contains all the business logic and rules to run a custom user application that exploits the data sent by the end devices, forwarded by the NS via some standard communication protocols such as HTTP webhooks or MQTT. An application can include storing, visualization, and analysis of data. *SALVO* foresees a control application addressed to qualified staff. The application should at a minimum allow to: visualize in real-time measurements of each device, visualize a map with the exact position of each device, provide a form of alert in case of alarms received by the nodes and store the data for some period. To create a proof of concept of the control platform, we employed the IoT Cloud Platform Tago.io that offers the tools to manage devices, store data, run analytics, and integrate services. Its free version allows the registration of a maximum of 5 devices, the creation of 5 dashboards, and 30 days of free storage.

Chapter 3

End to end data transmission

In this chapter, we present *SALVO*'s data transmission chain. After discussing LoRaWAN's limitations, two different methods of encoding the messages will be presented and compared to find the most reliable way to transmit messages. Finally, we present an initial version of the monitoring dashboard, in which the data are collected and visualized.

3.1 Types of messages

In this section, we take a closer look at the kind of information sent by the end devices, i.e, the periodic telemetry message and a rarer, but more important, alarm message.

3.1.1 Telemetry messages

Telemetry is the process of recording and transmitting the readings of an instrument. In *SALVO*, telemetry messages are low priority messages sent by the node every certain period. They are used to build a location aware historical records of the measured parameters and, in the future development of the platform, to run artificial intelligence algorithms for predicting dangerous situations. The members of the telemetry messages, derived from the sensors listed in Section 2.4, are reported in Table 3.1 along with their unit of measure. Each message includes information about the position of the node, GPS or RTLS according to if it is in an open or indoor environment. A careful design of the message and a correct study of the transmission period is needed to maximize the goodput of the LoRaWAN network.

Measurement	**Unit of measure**
Temperature	°C
Humidity	%rH
PM_1	$\mu g/m^3$
$PM_{2.5}$	$\mu g/m^3$
PM_{10}	$\mu g/m^3$
Loudness	db
VOC	ppm
CO	ppm
O_2	ppm
CO_2	ppm

Table 3.1. Members of the telemetry message along with their unit of measure.

3.1.2 Alarm messages

Alarms messages are sent as soon as a sensor is triggered by a rarer event representing a possible danger. These kinds of events can be thresholds being exceeded, sensors not working, discharged battery, falls detection, or alerts sent by the user by pressing a button on the device. Even though the probability of occurrence should be low, the delivery and the correct reception of the alarms must be ensured since they are strictly related to the safety of the operators.
The current implementation of the alarms is done using a 16-bit register where each bit is associated with a different kind of alarm as reported in Table 3.2. The register is then sent in a packet that also contains information about the position and the actual value measured by the sensor that triggered the alarm.

Alarm type	**Register value**
Temperature alarm	`xxxxxxxxxxxxxxx1`
Humidity alarm	`xxxxxxxxxxxxxx1x`
Pressure alarm	`xxxxxxxxxxxxx1xx`
VOC alarm	`xxxxxxxxxxxx1xxx`
CO_2 alarm	`xxxxxxxxxxx1xxxx`
PMS alarm	`xxxxxxxxxx1xxxxx`
Audio alarm	`xxxxxxxxx1xxxxxx`
BLE broadcast alarm	`xxxxxxxx1xxxxxxx`
Fall alarm	`xxxxxxx1xxxxxxxx`
User alarm	`xxxxxx1xxxxxxxxx`
- Unused -	`xxxxx1xxxxxxxxxx`
- Unused -	`xxxx1xxxxxxxxxxx`
- Unused -	`xxx1xxxxxxxxxxxx`
POZYX timeout	`xx1xxxxxxxxxxxxx`
POZYX sensor failures	`x1xxxxxxxxxxxxxx`
PMS Sensor failures	`1xxxxxxxxxxxxxxx`
No alarms - No sensor failures	`0000000000000000`

Table 3.2. Alarm status register structure.

3.2 From the end node to the gateway

The first step in the transmission chain of a LoRaWAN network is the Radio Frequency transmission between the end device and the gateway. LoRaWAN is based on the ALOHA access scheme which means that whenever a node has data to transmit, it can transmit without listening if the channel is busy or not. The success of the transmission is only determined by the absence of collisions with transmissions from other nodes, which is verified when different channels and different SF are used. Intuitively, as the number of messages grows, and the air becomes more saturated with transmissions, the number of collisions is going to increase. The combination of the number of end devices and the transmission period must be carefully studied to fit each use case. The number of collisions can also be reduced by decreasing the Time on Air of each transmission, since the shorter a signal stays on the air, the earlier it frees that channel and that specific spreading factor. Application's payloads are, therefore, sent using a specific encoding that minimizes the package size. In *SALVO*, telemetry messages are going to be coded, since they are made of many data and are frequently sent, so to reduce collisions even with alarms messages and reduce the power consumption of the end node. In the next section the standard CayenneLLP coding is presented.

3.2.1 CayenneLLP

Cayenne Low Power Payload is an easy and simple way to send data via LPWAN such as LoRaWAN. The Cayenne LPP is compliant with the payload size restriction and allows the end device to send multiple sensor data at one time. To achieve that, each measure is anticipated by two bytes:

- *Data Channel*: used to identify to which sensor of the device the frame is associated to
- *Data Type*: used to identify to which sensor of the device the frame is associated to

Data Types conform to the IPSO Alliance Smart Objects Guidelines, which identifies each data type with an "Object ID". However, a conversion is made to fit the Object ID into a single byte [44].
All the possible CayenneLLP Data Types are defined in Table 3.3.

Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Lat : 0.0001 ° Signed MSB Long : 0.0001 ° Signed MSB Altitude : 0.01 m Signed MSB

Table 3.3. CayenneLLP allowed Data Types along with their size in bytes, HEX representation and resolution.

The typical structure of a payload is shown in Figure 3.1.

1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

Figure 3.1. CayenneLLP payload structure: each measure is anticipated by two bytes referring to its type and its unique sensor in the device.

Assuming a progressive number identifying each sensor, a possible *SALVO* telemetry message encoded using CayenneLLP could be the following:

Payload (Hex)	`01 67 01 13 02 68 41 03 03 00 14 03 03 00 19 03 03 00 32 04 00 72 05 03 01 F4 06 03 01 2C 07 03 00 C8 08 03 07 D0 09 88 06 76 5F F2 96 0A 00 03 E8`	
Data Channel	**Type**	**Value**
01	67 $\longrightarrow$ Temperature	00 13 $\longrightarrow$ 27.5 °C
02	68 $\longrightarrow$ Humidity	41 $\longrightarrow$ 65%
03	03 $\longrightarrow$ Analog Input	00 14 $\longrightarrow$ 20 μg/m^3 (PM$_1$)
03	03 $\longrightarrow$ Analog Input	00 19 $\longrightarrow$ 25 μg/m^3 (PM$_{2.5}$)
03	03 $\longrightarrow$ Analog Input	00 32 $\longrightarrow$ 50 μg/m^3 (PM$_{10}$)
04	00 $\longrightarrow$ Digital Input	72 $\longrightarrow$114 db (Loudness)
05	03 $\longrightarrow$ Analog Input	01 F4 $\longrightarrow$ 500 ppm (CO$_2$)
06	03 $\longrightarrow$ Analog Input	01 2C $\longrightarrow$ 300 ppm (CO)
07	03 $\longrightarrow$ Analog Input	00 C8 $\longrightarrow$ 200 ppm (VOC)
08	03 $\longrightarrow$ Analog Input	07 D0 $\longrightarrow$ 2000 ppm (O$_2$)
		06 76 5F $\longrightarrow$ 42.3519°
09	88 $\longrightarrow$ Location (GPS)	F2 96 0A $\longrightarrow$-87.9094°
		00 03 E8 $\longrightarrow$10 m

Table 3.4. SALVO's telemetry message example using CayenneLLP.

The correspondent decoded payload in JSON format is:

```
{"decoded_payload": {
      "temperature_1": 27.5,
      "humidity_2": 41,
      "analog_input_3": 20,
      "analog_input_3": 25,
      "analog_input_3": 50,
      "digital_input_4": 114,
      "analog_input_5": 500,
      "analog_input_6": 300,
      "analog_input_7": 200,
      "analog_input_8": 2000,
      "location_9": {
          "latitude": 42.3519,
          "longitude": -87.9094,
          "altitude": 10,
      },
  }
}
```

The previous example highlights some critical aspects of exploiting CayenneLLP in *SALVO*. Since the *Data Types* allowed are only those reported in Table 3.3, measurements that do not belong to any of them must be referred to as one of those available that fits their sizes and that is possibly closer to their real type to avoid confusion. For instance, concentration measurements are defined as “Analog Input” since they take two bytes each, whereas loudness is defined as “Digital Input” since its size is only 1 byte. Furthermore, a sensor measuring multiple things always uses the same *Data Channel*, resulting in ambiguities when the measurements are of the same type. This can be seen in the decoded JSON payload: since the PM sensor (DataCh 03) measures three different values, three “analog_input_3 ” fields are present, but it is impossible to distinguish which is related to PM_1, $PM_{2.5}$, or PM_{10}. The total size of the payload is 49 bytes, only 2 bytes smaller than the maximum allowed dimension transmittable using SF12, meaning that expanding it with new information results in violation of the EU regulations, which is a limiting factor since the project has still time to be developed and the end node may have additional sensors to the current implementation.
To solve these problems, a new encoding is introduced in the next section.

3.2.2 Proposed encoding

A new proprietary encoding (that we will call *Optimized* from now on in the thesis) is proposed with the intent of fitting all the information to be sent by the nodes in the smallest packet possible. The reduced dimension of the payload allows a faster transmission, a reduction in radio packets collision and makes it possible to expand the message in future developments of the board.
The Optimized encoding is obtained by removing from CayenneLLP the *Data Channel* and *Data Type* for each value: indeed, those fields have no actual advantages in *SALVO* and only represent a waste of space and a cause of ambiguities. The message is represented as an ordered buffer containing only the values measured by the sensor so that each measure is identified by its position inside the buffer.
The message in Table 3.4 can be therefore encoded as follows:

Payload (HEX)	`01 13 41 03 14 00 19 00 32 72 01 F4 01 2C 00 C8 07 D0 06 76 5F F2 96 0A 00 03 E8`	
Position in buffer	**Mesaure**	**Value**
[1, 2]	Temperature	00 13 → 27.5 °C
[3]	Humidity	41 → 65%
[4, 5]	PM_1	00 14 → 20 µg/m3
[6, 7]	$PM_{2.5}$	00 19 → 25 µg/m3
[8, 9]	PM_{10}	00 32 → 50 µg/m3
[10]	Loudness	72 →114 db
[11, 12]	CO_2	01 F4 → 500 ppm
[13, 14]	CO	01 2C → 300 ppm
[15, 16]	VOC	00 C8 → 200 ppm
[17, 18]	O_2	07 D0 → 2000 ppm
[19, 20, 21]	Pos X	06 76 5F → 42.3519°
[22, 23, 24]	PosY	F2 96 0A →-87.9094°
[25, 26, 27]	Pos Z	00 03 E8 →10m

Table 3.5. SALVO's telemetry message example using Optimized encoding.

The payload size is now 27 bytes, 45% smaller than CayenneLLP's payload. As a consequence of the reduction of the size, the Time On Air, calculated with Equation 2.6, is going to be lower: using the lowest Data Rate, correspondent to SF12, the ToA is going to be 1974,3 ms against the 2793,5 ms of CayenneLLP. Each message is staying almost 1 second less on air, which will result in benefits in a congested network or when the environmental conditions are harsh.

3.2.3 Experimental results

A possible *SALVO* network was simulated with the intent of studying how the network behaves when the number of devices grows or when the transmission period changes, how the two encodings described in Section 3.2.1 and Section 3.2.2

perform, and finally to understand the influence that the introduction of the acknowledgment would have on the overall network performances.

Thanks to the large interest that the research community has toward LoRaWAN, many open-source simulators have been developed over recent years [45] but most of them are not capable of describing the LoRaWAN protocol stack as a whole and rather focus only on one target. One of the first and most famous LoRaWAN simulators is LoRaSim [46], a discrete-event simulator based on SimPy, that can emulate a network of devices and gateways randomly positioned on a grid implementing a long-distance path loss channel model. However, LoRaSim has not been updated since 2017 and still lacks some features such as downlink traffic and duty cycle limitation. A LoRaWAN module based on ns-3 discrete-event network simulator, implementing only Class A devices is presented in [47], but it also lacks downlink communication and has a steep learning curve.

LoRaWANSim, presented in [48] and available for free at https://github.com/kvmikhayl/LoRaWAN_simulator, is a LoRaWAN simulator implemented in MATLAB, whose completeness and simplicity made it the perfect choice for our goals. It characterizes the behavior of LoRaWAN networks accounting for physical, medium access control, and network aspects.

A complete LoRa transceiver is implemented at the PHY-layer, thereby generating the modulated signal, and performing demodulation tasks, whereas the MAC layer manages the data traffic, accounting for mutual interference and the presence of multiple gateways. Class A operating in the EU863-870MHz ISM band is considered and supports several features such as uplink and downlink communication, downlink communication, uplink-downlink interference, and DC limitation. A large number of parameters can also be tuned to make the simulator flexible and adaptable to many situations. The results of the simulation are given in terms of *Packet Delivery Rate*, for both uplink and downlink. The uplink delivery rate is defined as the ratio between the total uplink messages sent and the total number of packets received by the gateways, considering noise and interference that may prevent the correct delivery of a packet, whereas the downlink delivery rate is then defined as the ratio between the number of downlink packets correctly delivered and those generated. The simulator's block scheme is shown in Figure 3.2.

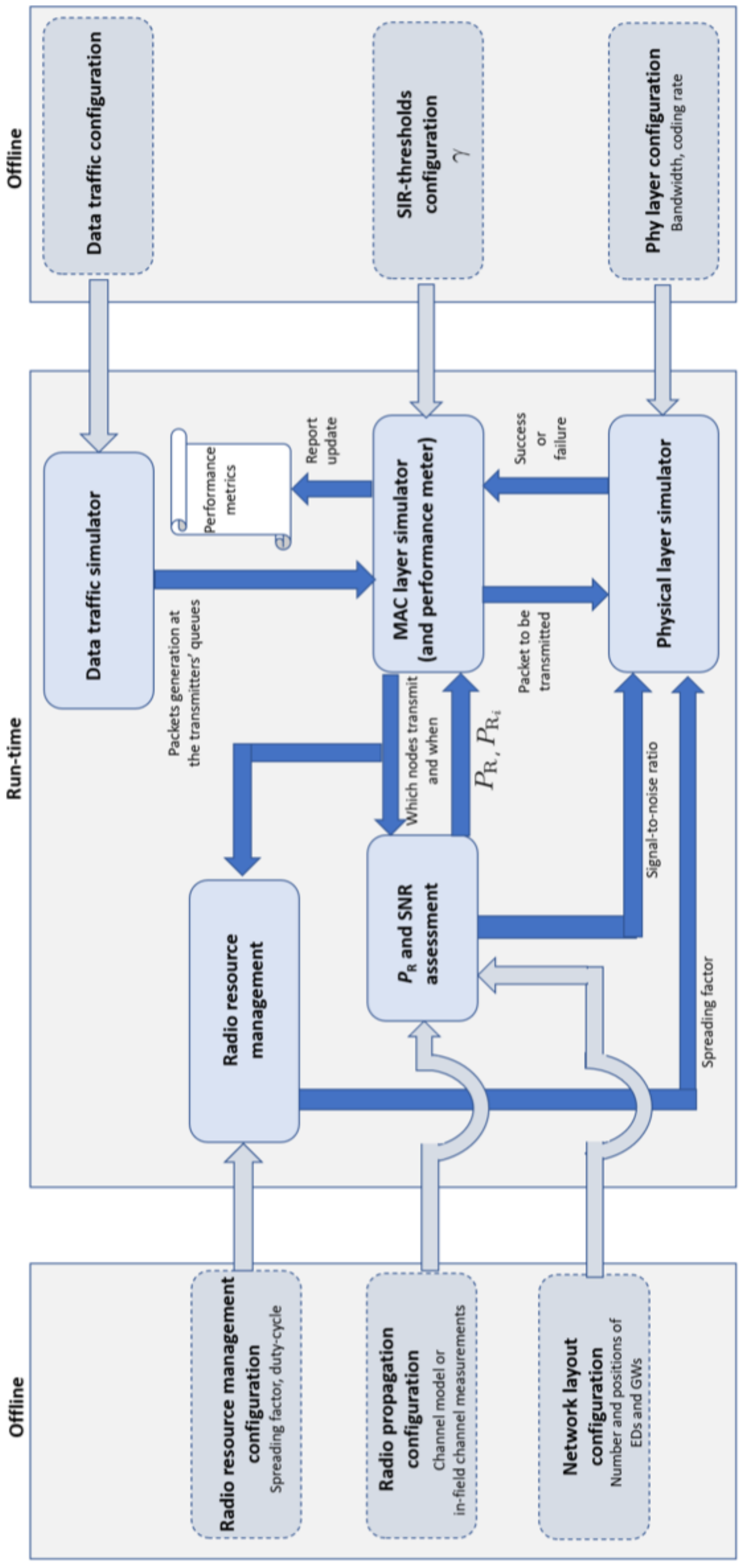

Offline
Data traffic configuration
SIR-thresholds configuration
γ
Phy layer configuration
Bandwidth, coding rate
Run-time
Data traffic simulator
Packets generation at the transmitters' queues
Performance metrics
Report update
MAC layer simulator (and performance meter)
Success or failure
Physical layer simulator
Packet to be transmitted
Which nodes transmit and when
P_R, P_{R_i}
Signal-to-noise ratio
Radio resource management
P_R and SNR assessment
Spreading factor
Offline
Radio resource management configuration
Spreading factor, duty-cycle
Radio propagation configuration
Channel model or in-field channel measurements
Network layout configuration
Number and positions of EDs and GWs

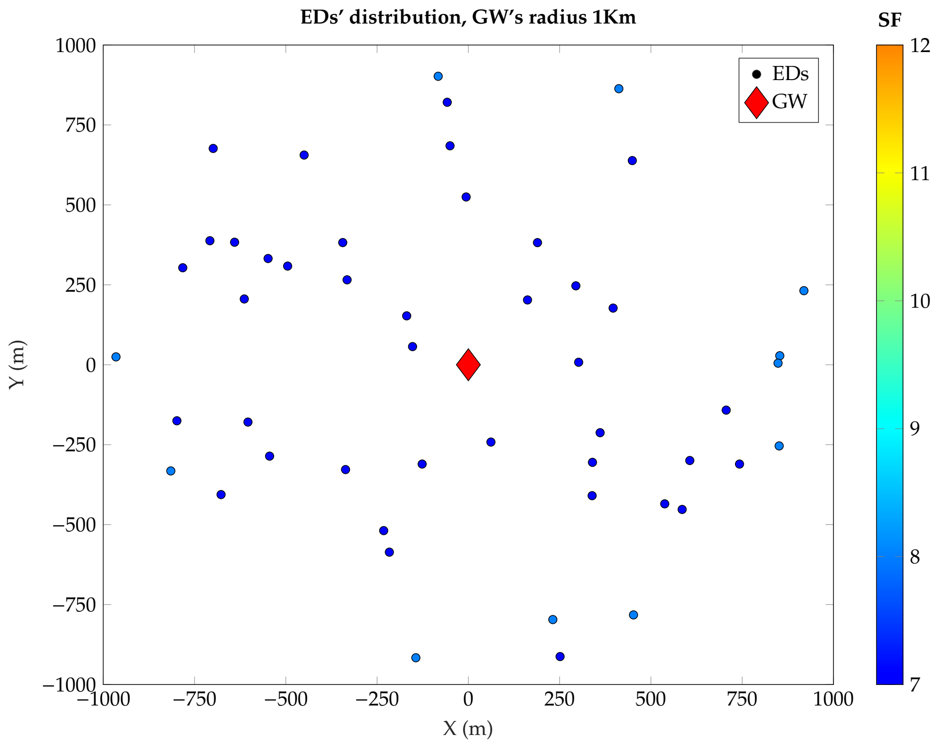
Figure 3.2. Simulator's block scheme.

Parameter	**Value**
Number of GWs	1
Circular area radius	[1,2,3] km
Number of Devices	[50,100,200,500]
Transmission period τ	[60,120,300,600] s
TX Power Gateway	14dbm
GW's antenna gain	3dbi
Node's antenna gain	3dbi
Alarm payload	15 bytes
Duty Cycle limitation	1%

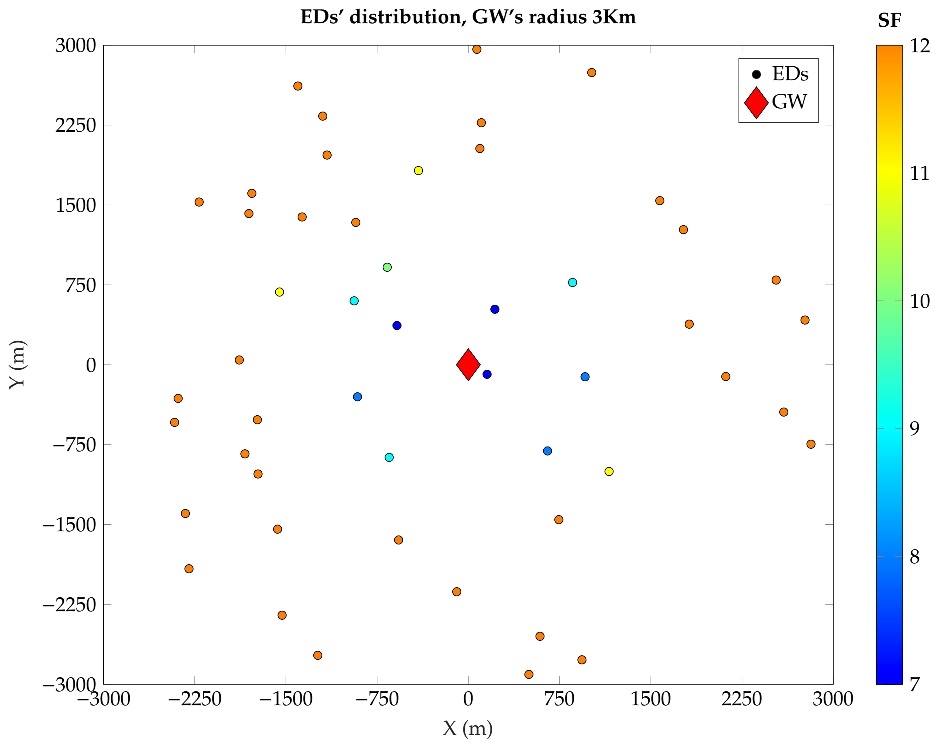
Table 3.6. List of parameters defined for the simulation

Before starting, different simulation parameters should be tuned so that the simulated scenario reflects what the reality of interest should look like. Our simulation parameters are reported in Table 3.6, in which those regarding devices (end nodes and gateway) are directly taken from the official datasheets.

Once the simulation starts, several nodes, whose number is a user defined parameter, are spread in the gateway area of radius R, and to each of them, a specific spreading factor is given according to the channel conditions (example Figure 3.3). After that, each node starts transmitting after a short random time and the transmission is repeated every user set time interval until the simulation is over. The simulator has been modified to implement alarms messages which are sent with a probability of one every 5000 messages. The results of the simulation are given in terms of *Packet Delivery Rate*, for both uplink and downlink. The *uplink delivery rate* is defined as the ratio between the total number of uplink messages sent and the total number of packets received by the gateways, considering noise and interference that may prevent the correct delivery, whereas the *downlink delivery rate* is then defined as the ratio between the number of downlink packets correctly delivered and those generated.



(a) Gateway radius 1km



(b) Gateway radius 3km

Figure 3.3. Example of the distribution of 50 end devices: the closer to the gateway, the lowest the Spreading Factor assigned and hence the higher the data rate.

The first studied scenario scenario consider the use of CayenneLLP, therefore considering a telemetry message payload of 49 bytes, with nodes around the gateway area with a radius equal to 1km. As shown in Figure 3.4, the Packet Delivery Rate decreases as the number of messages increases, that is when the transmission period is shorter, or the number of devices grows. For instance, when the transmission period is 60 seconds and 500 devices are in the network, the Packet Delivery Rate is only 63% compared to the 99.3% obtained considering a transmission period of 600 seconds and 50 devices. Furthermore, in these crowded conditions, not every alarm message sent by the network was received by the gateway, such as the case with the period of 60 seconds with 200 or 500 devices, and the case of a period of 120s and 500 devices.

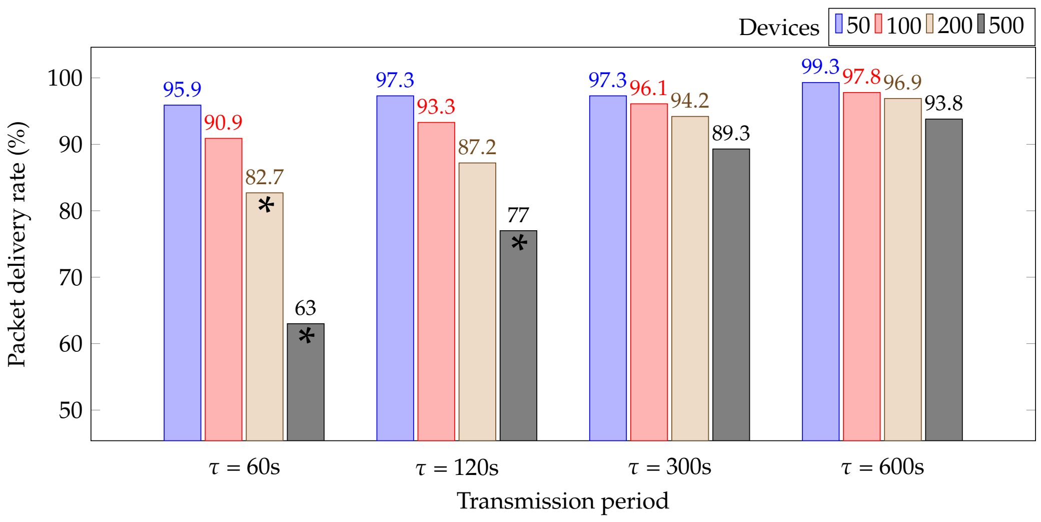
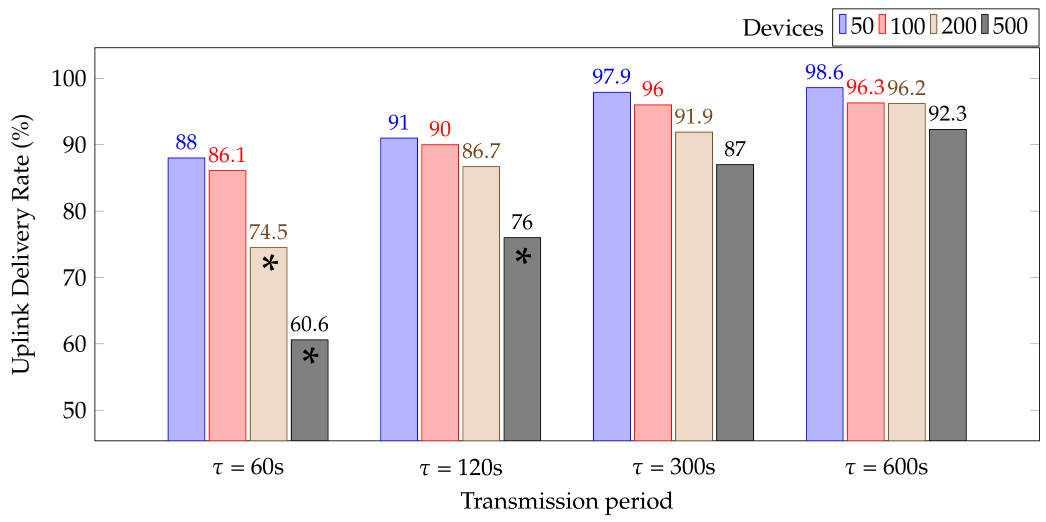


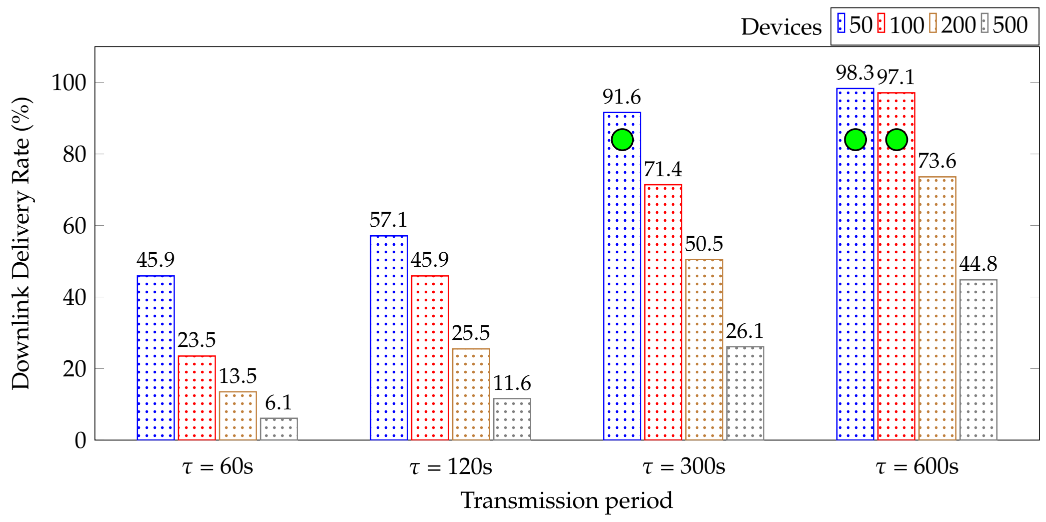
Figure 3.4. CayenneLPP Packet Delivery Rate vs Transmission Period and Number of devices. Not all the alarms sent were received by the gateway in the simulations marked with $*$.

As mentioned in Section 2.3.2, LoRa end nodes can send messages in a *confirmed mode*, which requires the LoRaWAN network to confirm the reception of the message with an acknowledgment. Acknowledgment is sent back using the same channel and SF of the uplink message and may be very useful in case of alarms: whenever an alarm is correctly received by the gateway, the *SALVO* end node receives an ACK

and may notice the worker of the correct reception with feedback, such as a buzzer beeping.
Figure 3.5 shows how the previous simulated network behaves when the confirmed mode is set for all the messages, both telemetry, and alarms messages. Due to the presence of the downlink messages representing the ACK, the *uplink delivery rate* (Figure 3.5a) diminishes compared to the case of no ACK envisioned, especially with a low transmission period and many devices in the network. For instance, the uplink delivery rate when τ is 60 seconds, and 50 devices are in the network is 8% smaller when the confirmed mode is activated compared to when it is turned off. Furthermore, the downlink delivery rate shown in Figure 3.5b, has low values most of the time. This is because the downlink communication is subject to the Duty Cycle limit of 1% imposed by the EU, meaning that, after a transmission, the gateway must be in silence for at least 99 times the duration of the transmission. As the number of messages grows, the gateway is requested to send more downlinks messages, but these cannot be scheduled since the limitation would be violated. Only in a few cases, the alarms received by the gateway correctly receive an acknowledgment back. (τ = 300s with 50 devices, and τ = 600 seconds with 50 or100 devices), so in this condition, their use is meaningless and even counterproductive.



(a) Uplink Delivery Rate



(b) Donwlink Delivery Rate

Figure 3.5. Results when confirmed mode is activated for each message. Only in the simulation marked with ● all the alarms received by the gateway, received an ACK themselves.

To reduce the packet loss, we tested the confirmed mode only for alarms messages since telemetry messages have a low priority an ACK is not necessary, whereas, as said before, an acknowledgment for alarms allows to send feedback to the operator of the correct reception of the message. Figure 3.6 shows the uplink packet delivery rate for this case: the results are similar to the one shown in Figure 3.4 because being the probability of receiving an alarm message is very low, the probability for the gateway of sending an acknowledgment is low as well, avoiding any Duty Cycle infringements and collision with other packets. Indeed, all the alarms that were correctly received by the gateway, received an ACK back.

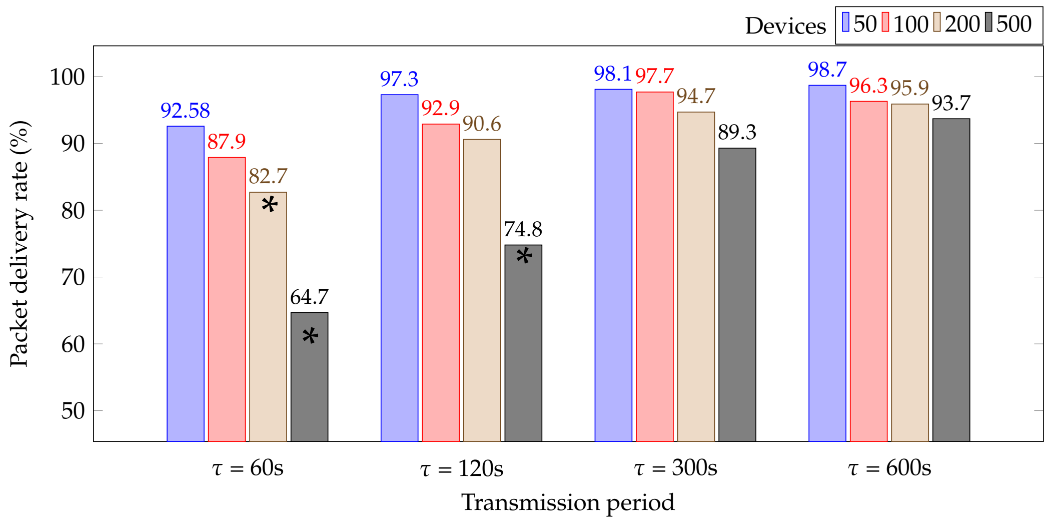


Figure 3.6. Results when only alarms requires ACK.

The last simulated scenario was carried out considering the use of the Optimize encoding, which reduces the telemetry message payload to only 27 bytes. Figure 3.7 shows a similar trend to CayenneLLP's reported in Figure 3.4, i.e., a packet delivery rate decreasing with a growing number of messages, however, the new encoding has always better results, with some improvements of up to 9% in more critical scenarios, such as the case of τ = 60s and 500 devices.

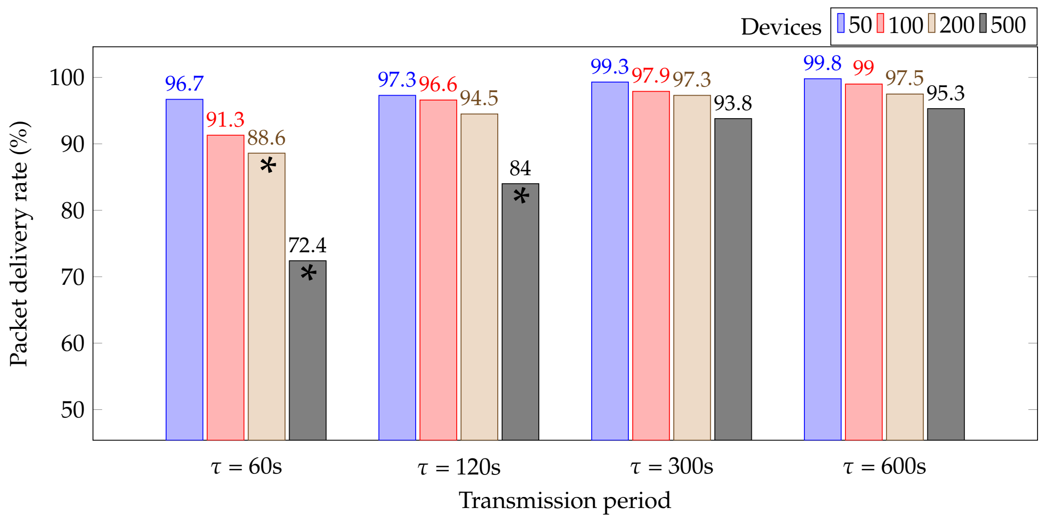
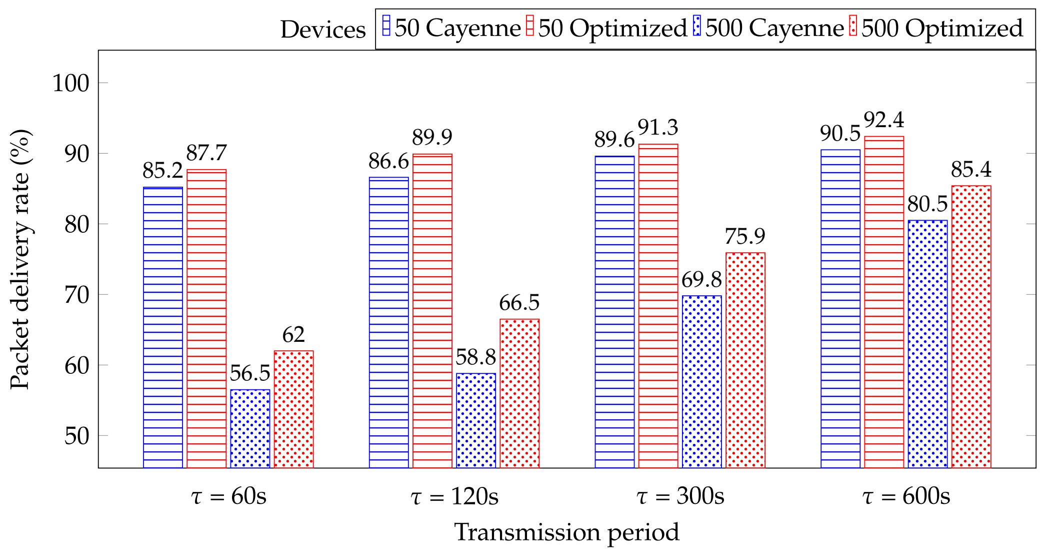
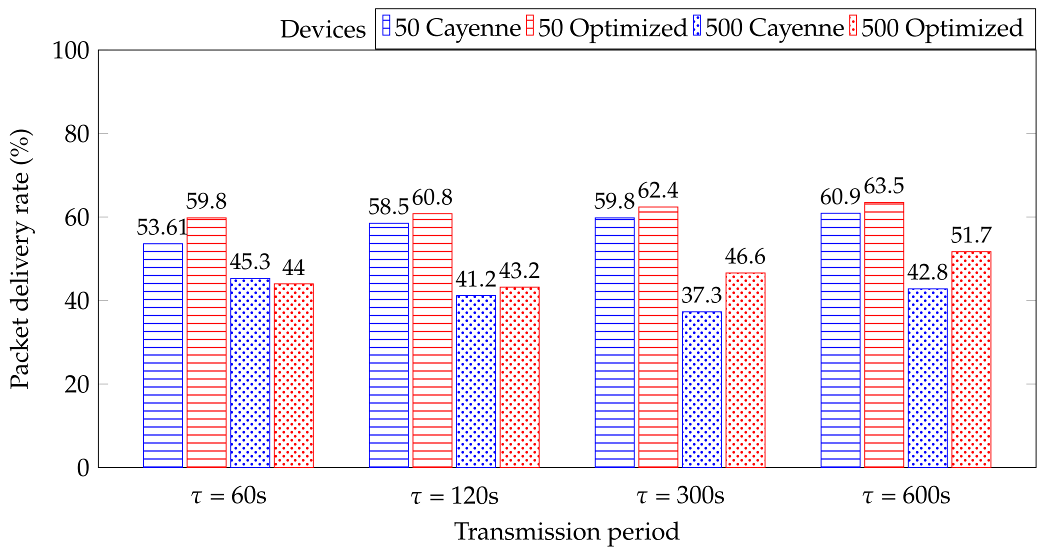


Figure 3.7. Optimized Encoding

Being the payload of the new encoding format is 45% smaller than the one encoded with CayenneLLP, the transmission time is reduced up to 0.8 seconds when transmitting with Spreading Factor 12 (lowest data rate possible), hence better performances in harsher environments are guaranteed. For instance, Figure 3.8 shows the packet delivery rate of both encoding formats, using either 50 or 500 devices (the best and worst case), when the gateway range is 2km (Figure 3.8a) and 3km (Figure 3.8b). In both cases, the new encoding has better performances, especially when the number of devices is large. In most of the scenarios, not all the alarms were received, but, hopefully, by collecting more telemetry messages, a safety manager could be able to discover some abnormal trends and proceed with a check. Note that in Figure 2b the similar values of packet delivery rate for 500 devices when τ is 60 or 120 are due to the violation of the end devices of the EU duty cycle of 1%.



(a) Gateway's radius 2km



(b) Gateway's radius 3km

Figure 3.8. Comparison of CayenneLLP and Optimized coding considering 50 or 500 devices in harsher environments, i.e. when placed further from the gateway.

In conclusion, the optimal implementation for *SALVO* would be to use the Optimized encoding with nodes transmitting messages every 300s or 600s (which were proved to be the most reliable configurations even with a large number of devices) and the use of confirmed mode only for alarms messages. The project will move in this direction in the next steps of the development.

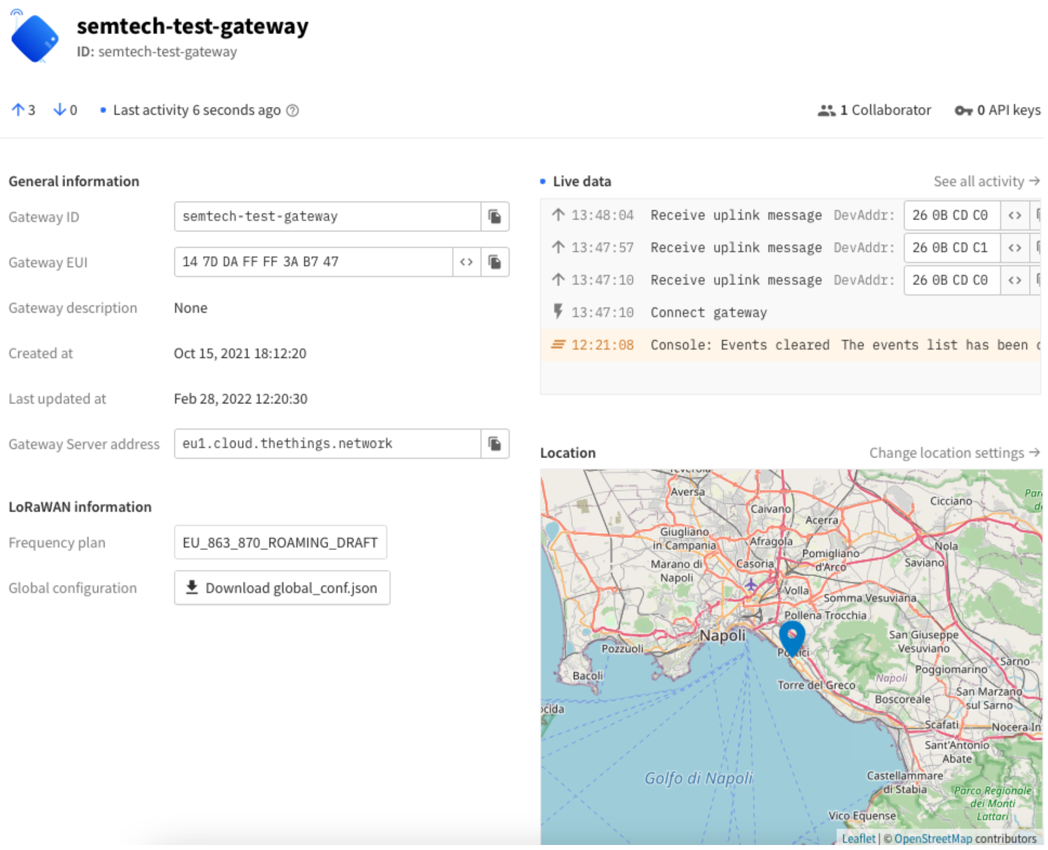
3.3 From the gateway to the user

In this section, we explore the second phase of the transmission of a message, which is the part using the internet. Once the Radio Frequency message is received by the gateway, it is converted into an IP packet forwarded via UDP protocol to the network server, which must be correctly configured beforehand. The packet is then sent via webhook to the application server, which displays it via the dashboard addressed to the qualified staff.
The real end-node is still in development and currently misses most of the foreseen sensors. To overcome this problem and develop a prototype of the platform the most complete possible, we used a LoRaWAN node simulator, available at https://github.com/kartben/lorawan-node-simulator, by Benjamin Cabé, principal project manager at Microsoft Azure IoT. It simulates LoRaWAN gateways, and endpoints regularly emitting LoRaWAN radio packets ("Unconfirmed Data Uplink" by default). The simulator is intended to stress test a LoRaWAN network server, only uplink traffic is generated at this point, and devices are expected to be provisioned via ABP instead of OTAA, but it is sufficient for our needs.
A custom payload generator function was created and reported in Appendix A. It generates both alarms messages and telemetry messages using the optimized coding. The values of each measurement, as well as the kind of position (indoor or outdoor) are randomly chosen.

3.3.1 Network Server configuration

To set an application using “The Things Network”, after creating an account in the platform, two steps are necessary: the registration of the gateway and the registration of the end devices that are going to be used.

A virtual gateway that uses the Semtech UDP Packet Forwarder protocol [1] is added to the TTN console by providing a Gateway name, a Gateway ID, and a Gateway EUI (Extended Unique Identifier) to uniquely identify it. With this protocol, uplinks, statuses, and downlinks are exchanged in a pseudo-JSON format, through UDP, between the gateway and the network server. Because of its simplicity, it is easy to reproduce it for testing purposes or for bootstrapping, and therefore it is implemented in the simulator to simulate a gateway. The gateway EUI is also hardcoded in the LoRaWAN packet simulator since it is where the packets get picked up and forwarded to the network server.


semtech-test-gateway
ID: semtech-test-gateway

↑3 ↓0 • Last activity 6 seconds ago

1 Collaborator 0 API keys

General information

Gateway ID	semtech-test-gateway
Gateway EUI	14 7D DA FF FF 3A B7 47
Gateway description	None
Created at	Oct 15, 2021 18:12:20
Last updated at	Feb 28, 2022 12:20:30
Gateway Server address	eu1.cloud.thethings.network

LoRaWAN information

Frequency plan	EU_863_870_ROAMING_DRAFT
Global configuration	Download global_conf.json

Live data See all activity →

```
↑ 13:48:04  Receive uplink message  DevAddr: 26 0B CD C0
↑ 13:47:57  Receive uplink message  DevAddr: 26 0B CD C1
↑ 13:47:10  Receive uplink message  DevAddr: 26 0B CD C0
  13:47:10  Connect gateway
  12:21:08  Console: Events cleared  The events list has been
```

Location Change location settings →

Leaflet | © OpenStreetMap contributors

Figure 3.9. Test gateway registered to the "TTN" console.

As mentioned before, the simulated devices must be registered in the TTN

[1]https://www.thethingsindustries.com/docs/gateways/udp/

console using the Activation by Personalization procedure (already described in Section 2.3.2), by providing a Device EUI, a Device Address, a Network Session Key, and an Application Session Key. The device address of each device is hardcoded in the simulator for the creation of the packets.
The raw buffer received from each device, representing the coded message, must be decoded in a human-readable way. This is done by specifying a "Payload Formatter" which decodes new messages as soon as they are received.
Our custom JavaScript payload formatter is reported in Appendix A and converts the hexadecimal message into a JSON object. For instance, the raw telemetry message sent by a node:

```
<00301d00330024000f950327000f001d05bc0639fb02300a00000c>
```

would be converted in the following JSON:

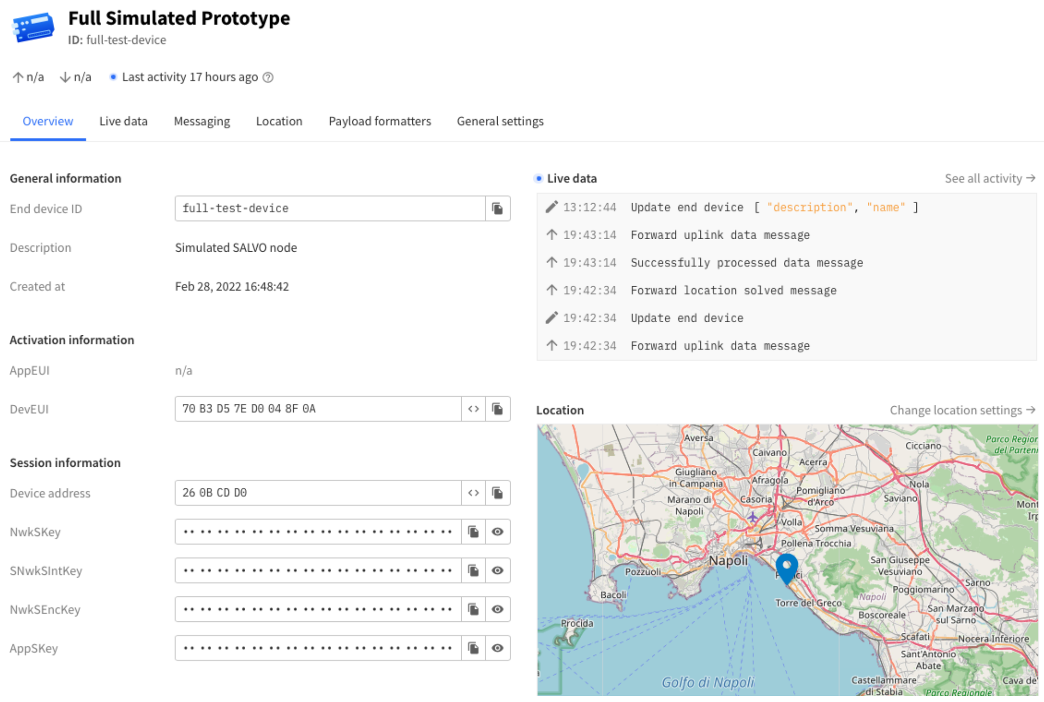
```json
{
  "altitude": 0.012,
  "co": 15,
  "co2": 807,
  "humidity": 29,
  "latitude": 40.8059,
  "longitude": 14.337,
  "loudness": 149,
  "o2": 1468,
  "pm1": 51,
  "pm10": 15,
  "pm25": 36,
  "temperature": 48,
  "voc": 29
}
```

wheras the following raw message corresponding to a user activated alert:

```
<02000010006e80007ff0008f7>
```

would be converted in the following JSON:

```
{ "alarms_status": 512,
  "indoor_x": 1768,
  "indoor_y": 2047,
  "indoor_z": 2295,
  "value": 1 }
```



Full Simulated Prototype
ID: full-test-device

↑ n/a ↓ n/a • Last activity 17 hours ago

Overview | Live data | Messaging | Location | Payload formatters | General settings

General information

End device ID	full-test-device
Description	Simulated SALVO node
Created at	Feb 28, 2022 16:48:42

Activation information

AppEUI	n/a
DevEUI	70 B3 D5 7E D0 04 8F 0A

Session information

Device address	26 0B CD D0
NwkSKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••
SNwkSIntKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••
NwkSEncKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••
AppSKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••

Live data See all activity →

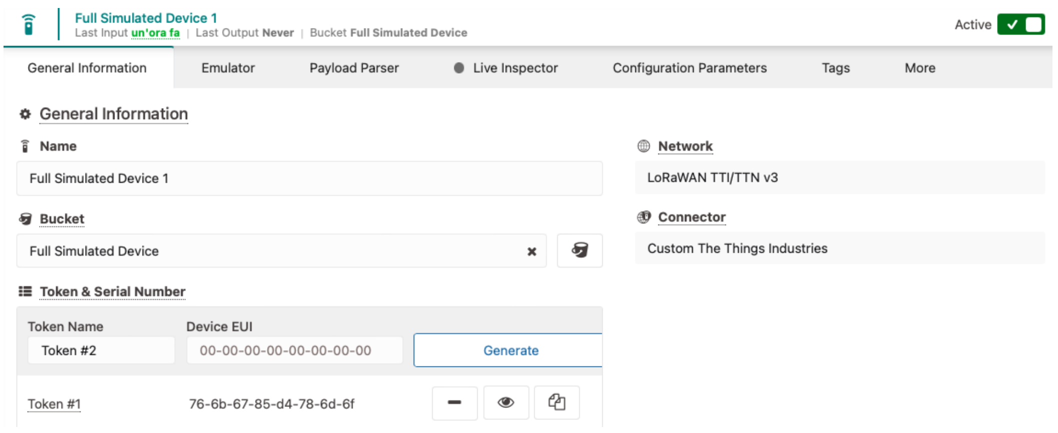
```
13:12:44  Update end device [ "description", "name" ]
19:43:14  Forward uplink data message
19:43:14  Successfully processed data message
19:42:34  Forward location solved message
19:42:34  Update end device
19:42:34  Forward uplink data message
```

Location Change location settings →

Figure 3.10. Test device registered to the "TTN" console.

3.3.2 Dashboard

The end nodes already registered in the "The Thing Network" must, in turn, be registered in "Tago.IO", with the same Device EUI, to create a link between the two platforms using a webhook. Through the webhook, TTN forwards every message of each node to "Tago.IO", that will store the data just received in the correspondent device's *bucket*, a place where variables are stored, created at the registration of the device. In the free version, a bucket will temporary contain all its data for 30 days.



Full Simulated Device 1
Last Input un'ora fa | Last Output Never | Bucket Full Simulated Device

Active

General Information | Emulator | Payload Parser | Live Inspector | Configuration Parameters | Tags | More

General Information

Name: Full Simulated Device 1

Bucket: Full Simulated Device

Network: LoRaWAN TTI/TTN v3

Connector: Custom The Things Industries

Token & Serial Number

Token Name	Device EUI	
Token #2	00-00-00-00-00-00-00-00	Generate
Token #1	76-6b-67-85-d4-78-6d-6f	

Figure 3.11. Test device registered to the "TagoIO" console.

At the current stage of the project, the minimum dashboard's requirements are the visualization of real-time measurements of each device, the visualization of the location (both indoor and outdoor) of each device, an alerting in case of alarm, and the visualization of the history of the received alarms.
These requirements can be met combining three useful features of "Tago.IO":

- *Predefined set of widgets*: it consists of a collection of already defined widgets (Figure 3.12) that allows the visualization of data coming from devices' buckets in different ways. Each widget can be customized in appearance and behavior.

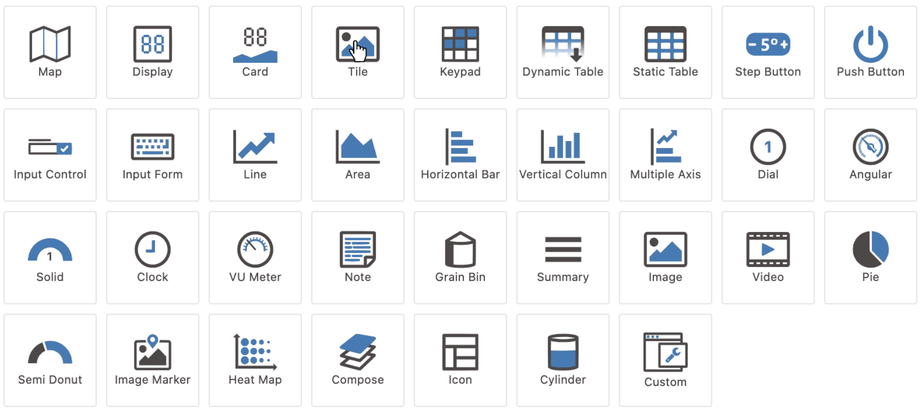


Figure 3.12. Tago.IO predefined set of widgets.

- *Analysis*: used to implement scripts to analyze and manipulate data coming from any device in real-time. Scripts are coded using the "Node.js Tago.IO SDK" and can be run internally by the platform itself (each analysis execution is billed in the monthly cost of the platform) or externally using the developer's own server. By using Analysis, users have access to all their data, devices, and even to external services using APIs. An analysis is useful for manipulating variables (such as transforming them or performing mathematical conversions), for adding new values in another data bucket and read the data from them, or to set up actions.

- *Actions*: are functions triggered when a selected variable meets certain conditions, or when a given resource (device, buckets, . . .) changes, or are triggered automatically from time to time by specifying a custom range (for example, every 13th of the month) or specifying a fixed time range (for example, every ten minutes). Actions include sending emails, SMS, making HTTP post requests, publishing to MQTT topic, notifications to the user's account or run custom analysis.

The created Dashboard is shown in Figure 3.13 and it's made up of the following widgets:

- *Device List*: list of all the devices registered in the platform.

- *Active nodes*: number of nodes that were active in the last hour. The number is updated using an *Analysis* that runs every five minutes and evaluates how much time has passed since the last message of each device. Since each bucket is associated to a device, a new fictitious device, called *Auxiliary*, is created to store data useful for the platform such as this counter. The *Analysis*'s code is reported in Appendix A.

- *Outdoor Location*: shows the location of the device currently located outdoor on a Map using the GPS coordinates received via message. "Tago.IO" can automatically extract the information about the position from the message forwarded by the network server. Each node is represented by a pin on the Map that when clicked opens an info box with all the measurements sent by the node. If the received message is an alarm, the pin will change color to red and the infobox will show information about the just received alarm.

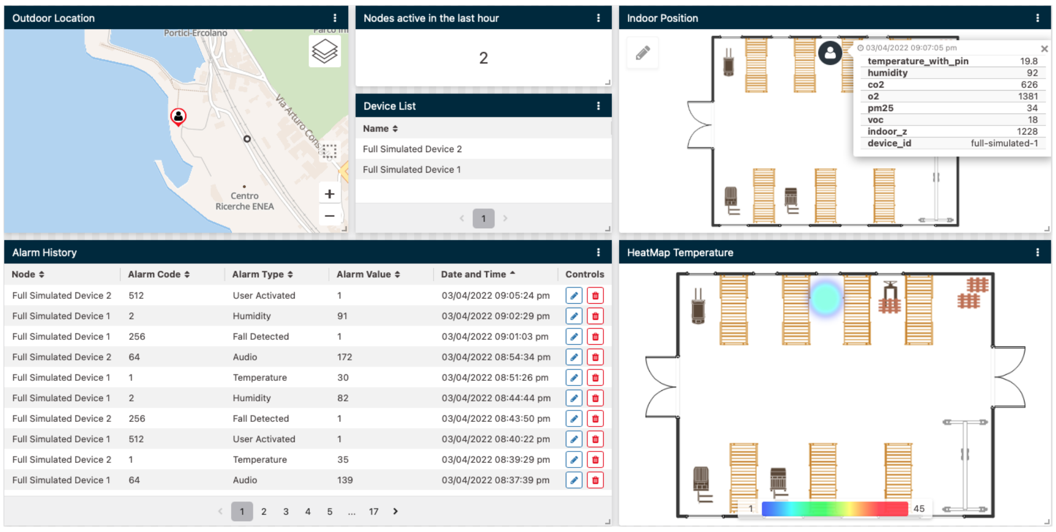
- *Indoor Position*: shows the devices that are currently located indoors. This widget is based on the "Image Marker" Tago.IO's predefined widget, which allows adding pins to different images. The center (0,0) is in the top left corner of the given image and coordinates (1,1) in the bottom right corner. Therefore, the indoor position's x and y coordinates, provided by the Real-Time Locating System, must be normalized to be correctly displayed. The operation is done exploiting an *Analysis* (whose code is reported in Appendix A) that is triggered as soon a message with information about indoor positioning is received. The *Analysis* applies the following formulas to obtain the pin's coordinate:

$$x_{pin} = \frac{x_{\text{indoor received}}}{x_{\text{max RTLS}}}, y_{pin} = \frac{y_{\text{indoor received}}}{y_{\text{max RTLS}}} \tag{3.1}$$

  where $\mathrm{x}_{\text{max RTLS}}$ and $\mathrm{y}_{\text{max RTLS}}$ are the maximum values measured by the RTLS, which ideally correspond to the dimensions of the indoor environments. The pin's coordinates are then provided to as metadata (additional piece of information) to the measurements of the node. Similar to the outdoor location widget, if the received message is an alarm, the pin changes color and information about the alarm is shown.

- *Alarm History*: shows a table containing records about the past received alarms. The predefined widget used is "Dynamic Table" which only allows showing data coming from a single device. To overcome this problem, whenever an alarm has been received an *Analysis* (reported in Appendix A) is triggered that stores the alarm information and its origin node as a new record in the previously mentioned *Auxiliary Device*'s bucket.

- *Heatmap*: displays a heatmap with bubbles representing the live intensity of the variables on top of your image (such as temperature in Figure 3.13). The higher the value of the variable, the redder the bubble will be. The size of the radius of the bubble is defined by the user, which makes this heatmap not accurate and useful only to have a global idea of the trend in the indoor environment. The predefined heatmap widget doesn't have layers, which means that a heatmap must be created for each measure (for instance, a CO_2 heatmap, an O_2 heatmap. . .).



Name
Full Simulated Device 2
Full Simulated Device 1

⊙ 03/04/2022 09:07:05 pm

temperature_with_pin	19.8
humidity	92
co2	626
o2	1381
pm25	34
voc	18
indoor_z	1228
device_id	full-simulated-1

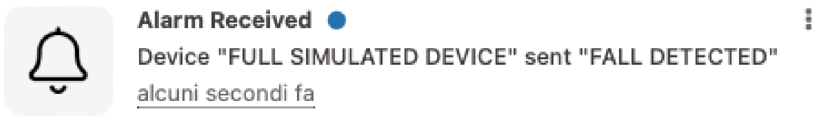
Alarm History

Node	Alarm Code	Alarm Type	Alarm Value	Date and Time	Controls
Full Simulated Device 2	512	User Activated	1	03/04/2022 09:05:24 pm	
Full Simulated Device 1	2	Humidity	91	03/04/2022 09:02:29 pm	
Full Simulated Device 1	256	Fall Detected	1	03/04/2022 09:01:03 pm	
Full Simulated Device 2	64	Audio	172	03/04/2022 08:54:34 pm	
Full Simulated Device 1	1	Temperature	30	03/04/2022 08:51:26 pm	
Full Simulated Device 1	2	Humidity	82	03/04/2022 08:44:44 pm	
Full Simulated Device 2	256	Fall Detected	1	03/04/2022 08:43:50 pm	
Full Simulated Device 1	512	User Activated	1	03/04/2022 08:40:22 pm	
Full Simulated Device 2	1	Temperature	35	03/04/2022 08:39:29 pm	
Full Simulated Device 1	64	Audio	139	03/04/2022 08:37:39 pm	

1 2 3 4 5 ... 17

Figure 3.13. Global dashboard with two simulated devices.

A notification alert, such as the one in Figure 3.14, appears at the reception of an alarm, indicating the original device and the type of alarm.



Alarm Received

Device "FULL SIMULATED DEVICE" sent "FALL DETECTED"

alcuni secondi fa

Figure 3.14. Dashboard's alarm notification.

Finally, a *blueprint* "Tago.IO" dashboard (Figure 3.15), a special type of dashboard that allows users to dynamically link widgets to multiple devices, is created to display information about every single node. After choosing a device, in addition to information about its location, graphs with the past values of its measurements of are shown.

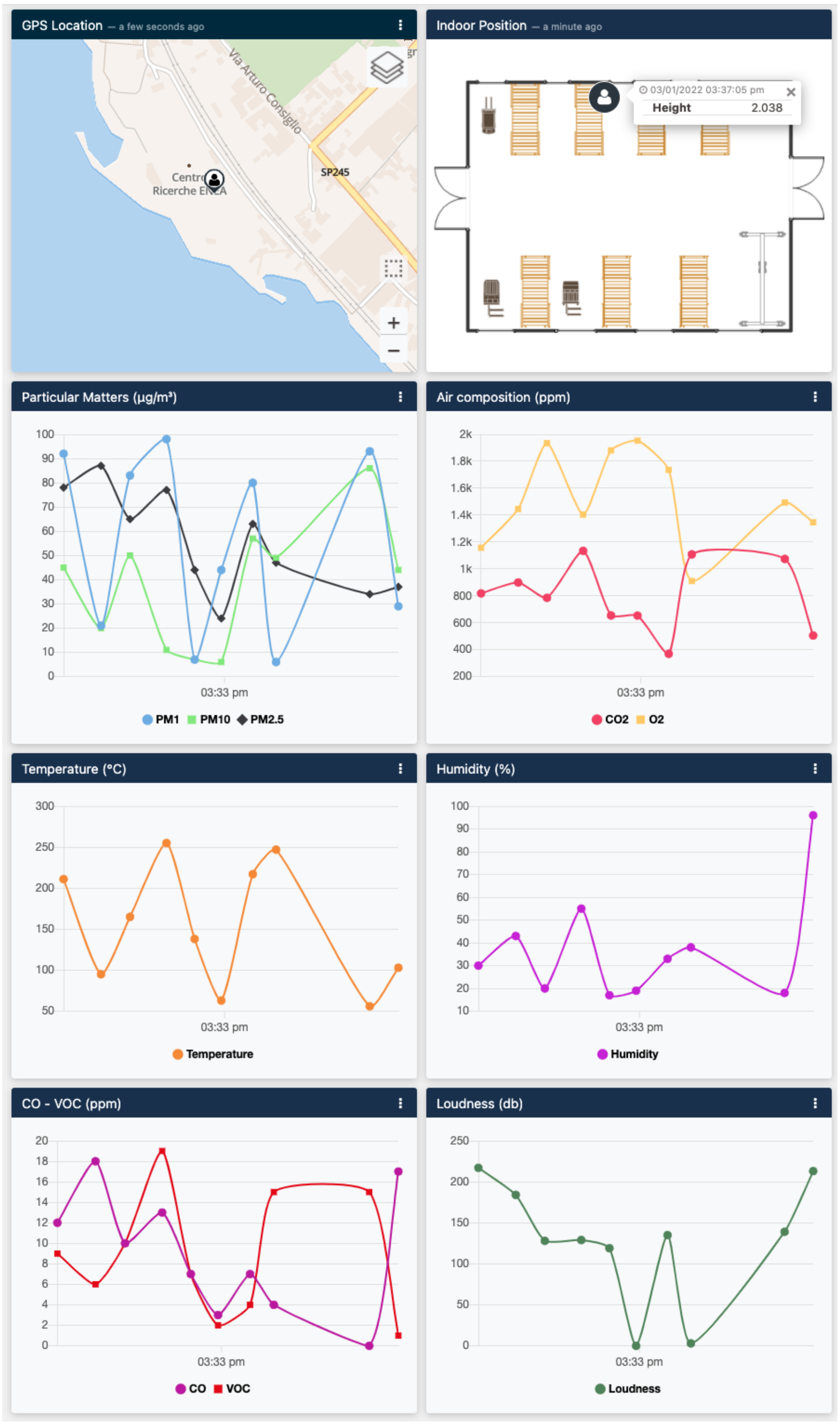


Figure 3.15. Specific Dashboard to each device.

Chapter 4

Conclusions

This Chapter will conclude the thesis by summarizing its main objectives reached and discussing its value and contribution. It will also review the current limitations of the discussed project and lay the foundations for future research.
In this work, we introduced the fundamental architecture of a platform exploiting IoT and Cloud to improve workers' security in an industrial environment. The platform is part of the National Operative Program (PON) project SALVO, and we specifically studied the design and development of the associated IoT ecosystem.
The fundamental core of the platform is a wearable device, addressed to each operator, able to sense environmental parameters, inertial parameters, and location (both indoor and outdoor). The application requires low power, low complexity, and low costs nodes, deployed on a scalable network, that can communicate over long distances and in different scenarios. After reviewing the common communication technologies, highlighting the downsides of exploiting them in this project, we deeply illustrated how the use of low power area networks fulfills each of the previous requirements.
We compared the main LPWANs technologies, concluding that LoRaWAN is the most suitable for our needs. Then, we defined the components of the LoRaWAN network to build a prototype of the platform choosing the open-source 'The Thing Network' as manager of the network and 'Tago.io' for the data visualization.
To improve the performance and reliability of the nodes' RF transmissions, we

proposed a new coding that reduces the payload of the telemetry messages up to 45% compared to the standard low power payload used in LoRaWAN applications, CayenneLLP. We proved that the new coding does bring benefits to the overall network performances, especially in harsh environments. We also stated through experiments that the introduction of acknowledgment is not useful to the network if the message received is not an alarm and that a correct transmission period for the telemetry message should be greater than 300 seconds.
Finally, we built a data visualization platform using a simulated full working node, that satisfies the minimum user requirements exploits the new coding. This will make easier the implementation of the real device as soon as it is complete.
The transmission chain was also tested with the current node prototype. Some changes had to be made since the format of the message is still provisional, but the tests were positive, and the data were correctly sent and visualized on the dashboard. Furthermore, the Confirmed Mode when an alarm is received by the network server was proved to be working, with a LED turning on when the alarm's ACK was correctly delivered back to the node.

4.1 Limitation and future developments

The architecture proposed in this thesis was thought to assess the feasibility of the platform but has some flows if implemented in a real-world scenario.
Both the network server and the application server are used in their free versions. To scale to a commercial-like level, business plans must be subscribed, which starting prices are 2500€/year for "The Things Stack" and 600€ for "Tago.io" and increases with the number of devices and functionalities. Furthermore, "Tago.io", is limited to its set of predefined widgets, that are not suitable when many devices coexist in the same application and can't be customized, which is a significant limitation. For instance, the maximum number of devices that could be shown in the GPS widget is only 10.
Future developments of the platform could go in the direction of a more proprietary solution that foresees a local private LoRaWAN network server for each application

of the project and resorting to cloud infrastructures only for to the collection of data in a temporal database such as InfluxDB to reduce the overall costs. However, the most important update that SALVO could receive is the implementation of a front-end control platform coded from the ground up. Most of the network servers provide REST APIs to interface with them. The new control platform could exploit them to improve the user experience, such as managing the devices directly from there, instead of accessing the IoT Cloud Platform provider settings, or creating more powerful widgets such as a precise heatmap exploiting the Inverse Distance Weighting algorithm.
Finally, the collected data could be used to enable the execution of targeted AI algorithms for predictive safety/maintenance (e.g., to predict malfunctioning of the HVAC systems, identify gas spills, etc.).
SALVO has still time to become a valuable tool to reduce accidents, injuries and, possibly, save lives for a safer Industry.

Appendix A

Scripts

Payload Generator

This function has been implemented in the Node.js LoRaWAN Node Simulator and returns a payload either of an alarm message or a telemetry message using the Optimized coding. The values of the measurements and the kind of position (indoor or outdoor) are random.

```

/* Generate Random Telemetry */
function generateRandomTelemetry(){
    let payload = Buffer.alloc(27);

    let temperature = Math.floor(Math.random() * 300) + 10; //Random [10,30]  C
    let humidity = Math.floor(Math.random() * 100); //Random [0,100] \%rH
    let pm1 = Math.floor(Math.random() * 100); //Random [0,100] ug/m3
    let pm25 = Math.floor(Math.random() * 100); //Random [0,100] ug/m3
    let pm10 = Math.floor(Math.random() * 100); //Random [0,100] ug/m3
    let loudness = Math.floor(Math.random() * 200) + 100; //Random [100,200] db
    let co2 = Math.floor(Math.random() * 1000) + 350; // Random [350,1000] ppm
    let o2 = Math.floor(Math.random() * 1250) + 750; // Random [750,1250] ppm
    let voc = Math.floor(Math.random() * 30); // Random [0,30] ppm
    let co = Math.floor(Math.random() * 25); // Random [0,25] ppm


    //GPS Positioning
    let lat = Math.floor(Math.random() * (408080 - 408057 + 1) + 408057) //
        Coordinates lat Portici
    let long = Math.floor(Math.random() * (143381 - 143350 + 1) + 143350) //
        Coordinates long Portici ENEA
    let alt = Math.floor(Math.random() * 10) + 10;

    //indoor positioning
    let indoor_x = Math.floor(Math.random() * (10000)); // 0-10000 mm
    let indoor_y = Math.floor(Math.random() * (5000)); // 0-3000 mm
    let indoor_z = Math.floor(Math.random() * (3000)); // 0-3000 mm

    payload[0] = temperature >> 8;
    payload[1] = temperature & 0xFF;
    payload[2] = humidity;
    payload[3] = pm1 >> 8;
    payload[4] = pm1 & 0xFF;
    payload[5] = pm25 >> 8;
    payload[6] = pm25 & 0xFF;
    payload[7] = pm10 >> 8;
    payload[8] = pm10 & 0xFF;
    payload[9] = loudness;
    payload[10] = co2 >> 8;
    payload[11] = co2 & 0xFF;
    payload[12] = co >> 8;
    payload[13] = co & 0xFF;
    payload[14] = voc >> 8;
    payload[15] = voc & 0xFF;
    payload[16] = o2 >> 8;
    payload[17] = o2 & 0xFF;

    let indoor = Math.floor(Math.random() * (5)) > 2 ? 0 : 1;

    if (indoor == 0){
        payload[18] = lat >> 16;
        payload[19] = lat >> 8;
        payload[20] = lat & 0xFF;
        payload[21] = long >> 16;
        payload[22] = long >> 8;
        payload[23] = long & 0xFF;
        payload[24] = alt >> 16;
        payload[25] = alt >> 8;
        payload[26] = alt & 0xFF;
    } else {
        payload[18] = indoor_x >> 16;
        payload[19] = indoor_x >> 8;
        payload[20] = indoor_x & 0xFF;
        payload[21] = indoor_y >> 16;
        payload[22] = indoor_y >> 8;
        payload[23] = indoor_y & 0xFF;
        payload[24] = indoor_z >> 16;
        payload[25] = indoor_z >> 8;
        payload[26] = indoor_z & 0xFF;
    }
    console.log(payload);
    return payload;
}

/*Generate Random Alarm */
function generateRandomAlarm() {
    const ALARMS: { [key: string]: number } = {
        "TEMPERATURE": 1,
        "HUMIDITY": 2,
        "CO2": 16,
        "AUDIO": 64,
        "FALL":256,
        "USER":512
    }

    let keys = Object.keys(ALARMS);
    let alarm = keys[ keys.length * Math.random() << 0];

    console.log(‘Sending alarm ${alarm}‘);

    let value: number = 0;

    switch (alarm) {
        case "TEMPERATURE":
            value = Math.floor(Math.random() * (45 - 30) + 30);
            break;
        case "HUMIDITY":
            value = Math.floor(Math.random() * (100 - 70) + 70)
            break;
        case "CO2":
            value = Math.floor(Math.random() * (2000 - 1500) + 1500);
            break;
        case "AUDIO":
            value = Math.floor(Math.random() * (200 - 110) + 110);
            break;
        case "FALL":
            value = 1;
            break;
        case "USER":
            value = 1;
            break;
    }

    let payload = Buffer.alloc(13);
    let alarmCode: number = ALARMS[alarm];

    payload[0] = alarmCode >> 8;
    payload[1] = alarmCode & 0xFF;
    payload[2] = value >> 8;
    payload[3] = value & 0xFF;

    let indoor = Math.floor(Math.random() * (5)) > 2 ? 0 : 1;

    if (indoor == 0){
        let lat = Math.floor(Math.random() * (408080 - 408057 + 1) + 408057) //
            Coordinates lat Portici
        let long = Math.floor(Math.random() * (143381 - 143350 + 1) + 143350) // Coordinates long Portici ENEA
        let alt = Math.floor(Math.random() * 10) + 10;

        payload[4] = lat >> 16;
        payload[5] = lat >> 8;
        payload[6] = lat & 0xFF;
        payload[7] = long >> 16;
        payload[8] = long >> 8;
        payload[9] = long & 0xFF;
        payload[10] = alt >> 16;
        payload[11] = alt >> 8;
        payload[12] = alt & 0xFF;
    } else {

        let indoor_x = Math.floor(Math.random() * (10000)); // 0-10000 mm
        let indoor_y = Math.floor(Math.random() * (5000)); // 0-3000 mm
        let indoor_z = Math.floor(Math.random() * (3000)); // 0-3000 mm

        payload[4] = indoor_x >> 16;
        payload[5] = indoor_x >> 8;
        payload[6] = indoor_x & 0xFF;
        payload[7] = indoor_y >> 16;
        payload[8] = indoor_y >> 8;
        payload[9] = indoor_y & 0xFF;
        payload[10] = indoor_z >> 16;
        payload[11] = indoor_z >> 8;
        payload[12] = indoor_z & 0xFF;
    }
    console.log(payload);
    return payload
}

/* Return a Telemtry message 70\% of time */
function generateRandomPayload()  {
    return (Math.random() < 0.7) ? generateRandomTelemetry() : generateRandomAlarm()
```

```
}



export { generateRandomPayload };
```

TTN Decoder

The Thing Network implementation of a custom decoder for optimized coding. Returns a human readable JSON object.

```
 function decodeUplink(input) {
   var data = {};

   if (input.bytes.length == 27) {
       data.temperature = (input.bytes[0] << 8) + input.bytes[1];
       data.humidity = (input.bytes[2]);
       data.pm1 = (input.bytes[3] << 8) + input.bytes[4];
       data.pm25 = (input.bytes[5] << 8) + input.bytes[6];
       data.pm10 = (input.bytes[7] << 8) + input.bytes[8];
       data.loudness = (input.bytes[9]);
       data.co2 = (input.bytes[10] << 8) + input.bytes[11];
       data.co = (input.bytes[12] << 8) + input.bytes[13];
       data.voc = (input.bytes[14] << 8) + input.bytes[15];
       data.o2 = (input.bytes[16] << 8) + input.bytes[17];

       let position_x = (input.bytes[18] << 16) + (input.bytes[19] << 8) + input
           .bytes[20];
       let position_y = (input.bytes[21] << 16) + (input.bytes[22] << 8) + input
           .bytes[23];
       let position_z = (input.bytes[24] << 16) + (input.bytes[25] << 8) + input
           .bytes[26];

       if (position_x > 40000){
       //it means GPS is being sent
       data.latitude = position_x/10000;
       data.longitude = position_y/10000;
```

```
        data.altitude = position_z/1000;
        } else {
        //it means rtls transmission
        data.indoor_x = position_x;
        data.indoor_y = position_y;
        data.indoor_z = position_z;
        }
    } else {
        data.alarms_status = (input.bytes[0] << 8) + input.bytes[1];
        data.value = (input.bytes[2] << 8) + input.bytes[3];

        let position_x = (input.bytes[4] << 16) + (input.bytes[5] << 8) + input.
            bytes[6];
        let position_y = (input.bytes[7] << 16) + (input.bytes[8] << 8) + input.
            bytes[9];
        let position_z = (input.bytes[10] << 16) + (input.bytes[11] << 8) + input
            .bytes[12];

        if (position_x > 40000){
        //it means GPS is being sent
        data.latitude = position_x/10000;
        data.longitude = position_y/10000;
        data.altitude = position_z/1000;
        } else {
        //it means rtls transmission
        data.indoor_x = position_x;
        data.indoor_y = position_y;
        data.indoor_z = position_z;
        }
    }

    return {
      data: data,
      warnings: [],
      errors: []
    };
  }
```

Indoor position normalization

This script is implemented as a Tago.IO analysis and used to normalized the indoor coordinates received by the nodes to display a user pin in the dashboard's widget.

```
const { Analysis, Utils, Account, Device } = require('@tago-io/sdk');

// The function myAnalysis will run when you execute your analysis
async function myAnalysis(context,scope) {
 if (!scope.length) return context.log('Analysis must be triggered by an action/widget');

  const env_vars = Utils.envToJson(context.environment);
  if (!env_vars.account_token) {
    return context.log('Missing account_token environment variable');
  }

  const account = new Account({ token: env_vars.account_token });

  // Get the device ID that triggered the Analysis;

  const device_id = scope[0].origin;
  const device_token = await Utils.getTokenByName(account, device_id);
  const device = new Device({ token: device_token });

  context.log(device)


  // create the filter options to get the data from TagoIO
  const indoorX = {
    variable: "indoor_x",
    query: "last_item",
  };

  const indoorY = {
    variable: "indoor_y",
    query: "last_item",
  };

  const indoorZ = {
    variable: "indoor_z",
    query: "last_item",
  };

  const temperatureQuery = {
    variable: "temperature",
    query: "last_item",
  }

  const resultIndoorX = await device.getData(indoorX).catch(() => null);
  const resultIndoorY = await device.getData(indoorY).catch(() => null);
  const resultIndoorZ = await device.getData(indoorZ).catch(() => null);
  const resultTemperature = await device.getData(temperatureQuery).catch(() =>
      null);


  const indoor_position_x = resultIndoorX[0].value;
  const indoor_position_y = resultIndoorY[0].value;
  const indoor_position_z = resultIndoorZ[0].value;
  const temperature = resultTemperature[0].value;

  context.log(
    ‘Indoor X: ${indoor_position_x} - Indoor Y ${indoor_position_y} - Indoor Z ${
        indoor_position_z}‘
  )

  //Test Position Salvo
  //Normalized and referencing to RTLS

  indoor_x_normalized = (indoor_position_x) / 10000;
  indoor_y_normalized = (indoor_position_y) / 5000;

  const obj_to_save_pin = [{
    "variable": "indoor_position",
    "value": indoor_position_z / 1000,
    "metadata": {
      "x": indoor_x_normalized,
      "y": indoor_y_normalized,
      "icon":"user",
    }
  }
  ];

  try {
    await device.sendData(obj_to_save_pin);
    context.log("Successfully Inserted");
  } catch (error) {
    context.log("Error when inserting:", error);
  }

}

module.exports = new Analysis(myAnalysis);
```

Active in last hour

This script is implemented as a Tago.IO Analysis and used to count the number of devices active in the last hour. The variable is stored in a new auxiliary device.

```
const { Analysis, Device, Account, Utils} = require("@tago-io/sdk");

async function countActiveDevice(context) {
  const account = new Account(env_vars.account_token);

  // Gell only full prototypes devices
  const filter = {
    tags: [
      {
        key: "type_device", // change by your key name
        value: "full_prototype", // change by your key value
      },
    ],
    // You also can filter by: name, last_input, last_output, bucket, etc.
  };

  // Searching all devices with tag we want
  const devices = await account.devices.list({
    fields: ["id", "tags"],
    filter,
  });

  if (!devices.length) {
    return console.log("Devices not found");
  }

  const number_active_last_hour = devices.map(async device => {
    let device_token = await Utils.getTokenByName(account, device.id);
    let accesible_device = new Device({ token: device_token });

    let filter = {
        variable: "fcnt",
        query: "last_item",
      };


  try{
    const lastMessage = await accesible_device.getData(filter).catch(() => null);
    let now = new Date();
    let diff_hours =  Math.floor(Math.abs(now - lastMessage[0].time)/(3600 *
        1000));
    let active = ( diff_hours <= 1) ? 1 : 0;
    return active

  } catch {
    return 0;
  }
  });

```

```
  Promise.all(number_active_last_hour).then(async (values) => {
        context.log(values);
        const number_active = values.reduce((a, b) => a + b, 0);

        let device = new Device(\{token: env.auxiliary_device_token\});

        let obj_to_save = {
            variable: "number_active_last_hour",
            value: number_active
          };

        context.log(number_active)

          try {
            await device.sendData(obj_to_save);
            context.log("Successfully Inserted");
          } catch (error) {
            context.log("Error when inserting:", error);
          }
      });
}

module.exports = new Analysis(countActiveDevice)
```

Alarm retrieve

This script is implemented as a Tago.IO Analysis and used to save the received alarm as a new record in the Auxiliary Device.

```
const { Analysis, Utils, Account, Device } = require('@tago-io/sdk');

// The function myAnalysis will run when you execute your analysis
async function myAnalysis(context,scope) {
 if (!scope.length) return context.log('Analysis must be triggered by an action/
     widget');

  const env_vars = Utils.envToJson(context.environment);
```

```
if (!env_vars.account_token) {
  return context.log('Missing account_token environment variable');
}

const account = new Account({ token: env_vars.account_token });

// Get the device id from the first variable available;
const device_id = scope[0].origin;
const device_token = await Utils.getTokenByName(account, device_id);
const device = new Device({ token: device_token });

const device_info = await device.info();

console.log(device_info.name);


  // create the filter options to get the data from TagoIO
const alarmStatus = {
  variable: "alarms_status",
  query: "last_item",
};

const alarmValue = {
  variable: "value",
  query: "last_item",
};


const resultAlarmStatus = await device.getData(alarmStatus).catch(() => null);
const resultAlarmValue = await device.getData(alarmValue).catch(() => null);

const alarm_status = resultAlarmStatus[0].value;
const alarm_value = resultAlarmValue[0].value;

const auxiliaryDevice = new Device({ token: env.auxiliary_device_token});

const obj_to_save_alarm_status = {
  "variable": "alarm_status",
  "value": alarm_status
};

const obj_to_save_alarm_value = {
  "variable": "alarm_status",
  "value": alarm_value
};

let string_value;
switch(alarm_status) {
  case 1:
  string_value = "Temperature";
  break;
  case 2:
  string_value = "Humidity";
  break;
  case 16:
  string_value = "CO2";
  break
  case 64:
  string_value = "Audio";
  break
  case 256:
  string_value = "Fall Detected";
  break;
  case 512:
  string_value = "User Activated"
}

const obj_to_save_alarm_string = {
  "variable": "alarm_string",
  "value": string_value
};

const obj_to_save_alarm_device = {
  "variable": "alarm_device",
    "value": device_info.name
  }


  const obj_to_save_alarm = [
    {
    "variable": "alarm_string",
    "value": string_value
    },
    {
    "variable": "alarm_device",
    "value": device_info.name
    },
    {
    "variable": "alarm_value",
    "value": alarm_value
    },
    {
    "variable": "alarm_status",
    "value": alarm_status
    }
  ]

  try {
    await auxiliaryDevice.sendData(obj_to_save_alarm);
    context.log("Successfully Inserted");
  } catch (error) {
    context.log("Error when inserting:", error);
  }

}

module.exports = new Analysis(myAnalysis);
```

Bibliography

[1] J. Pryce-Jones, *Happiness at work: maximizing your psychological capital for success*. Malden, MA: Wiley-Blackwell, 2010.

[2] B. O. L. Statistics, "Employer-Reported Workplace Injuries and Illnesses – 2019," p. 9, 2020.

[3] INAIL, "Andamento degli infortuni sul lavoro e delle malattie professionali," 2020.

[4] G. Culot, G. Nassimbeni, G. Orzes, and M. Sartor, "Behind the definition of Industry 4.0: Analysis and open questions," *International Journal of Production Economics*, vol. 226, p. 107617, Aug. 2020.

[5] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch, "Industry 4.0: The Future of Productivity and Growth in Manufacturing," p. 20.

[6] A. Damani, "Council Post: The Fundamentals And Impact Of Industry 4.0." Section: Small Business.

[7] A. Forcina and D. Falcone, "The role of Industry 4.0 enabling technologies for safety management: A systematic literature review," *Procedia Computer Science*, vol. 180, pp. 436–445, 2021.

[8] N. Tang, H. Hu, F. Xu, and F. Zhu, "Personalized safety instruction system for construction site based on internet technology," *Safety Science*, vol. 116, pp. 161–169, July 2019.

[9] C.-Y. Chang, K.-S. Ko, S.-J. Guo, S.-S. Hung, and Y.-T. Lin, "CO Multi-Forecasting Model for Indoor Health and Safety Management in Smart Home," *Journal of Internet Technology*, vol. 21, no. 1, pp. 273–284, 2020.

[10] W.-F. Cheung, T.-H. Lin, and Y.-C. Lin, "A Real-Time Construction Safety Monitoring System for Hazardous Gas Integrating Wireless Sensor Network and Building Information Modeling Technologies," *Sensors*, vol. 18, p. 436, Feb. 2018.

[11] B. Mayton, G. Dublon, S. Palacios, and J. A. Paradiso, "TRUSS: Tracking Risk with Ubiquitous Smart Sensing," in *2012 IEEE Sensors*, (Taipei, Taiwan), pp. 1–4, IEEE, Oct. 2012.

[12] S. Hiremath, G. Yang, and K. Mankodiya, "Wearable Internet of Things: Concept, Architectural Components and Promises for Person-Centered Healthcare," in *Proceedings of the 4th International Conference on Wireless Mobile Communication and Healthcare - "Transforming healthcare through innovations in mobile and wireless technologies"*, (Athens, Greece), ICST, 2014.

[13] W. Lee, K.-Y. Lin, E. Seto, and G. C. Migliaccio, "Wearable sensors for monitoring on-duty and off-duty worker physiological status and activities in construction," *Automation in Construction*, vol. 83, pp. 341–353, Nov. 2017.

[14] P. Arpaia, N. Moccaldi, R. Prevete, I. Sannino, and A. Tedesco, "A Wearable EEG Instrument for Real-Time Frontal Asymmetry Monitoring in Worker Stress Analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 8335–8343, Oct. 2020.

[15] A. Costin, A. Wehle, and A. Adibfar, "Leading Indicators—A Conceptual IoT-Based Framework to Produce Active Leading Indicators for Construction Safety," *Safety*, vol. 5, p. 86, Dec. 2019.

[16] Y.-C. Fang and R.-J. Dzeng, "Accelerometer-based fall-portent detection algorithm for construction tiling operation," *Automation in Construction*, vol. 84, pp. 214–230, Dec. 2017.

[17] M. Swan, "Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0," *Journal of Sensor and Actuator Networks*, vol. 1, pp. 217–253, Nov. 2012.

[18] E. Marks and J. Teizer, "Proximity Sensing and Warning Technology for Heavy Construction Equipment Operation," in *Construction Research Congress 2012*, (West Lafayette, Indiana, United States), pp. 981–990, American Society of Civil Engineers, May 2012.

[19] S. Choe, F. Leite, D. Seedah, and C. Caldas, "Evaluation of Sensing Technology for the prevention of Backover Accidents in Construction Work Zones," p. 19.

[20] J. Park, E. Marks, Y. K. Cho, and W. Suryanto, "Performance Test of Wireless Technologies for Personnel and Equipment Proximity Sensing in Work Zones," *Journal of Construction Engineering and Management*, vol. 142, p. 04015049, Jan. 2016.

[21] J. Wang and S. N. Razavi, "Low False Alarm Rate Model for Unsafe-Proximity Detection in Construction," *Journal of Computing in Civil Engineering*, vol. 30, p. 04015005, Mar. 2016.

[22] H. Zhang, X. Yan, H. Li, R. Jin, and H. Fu, "Real-Time Alarming, Monitoring, and Locating for Non-Hard-Hat Use in Construction," *Journal of Construction Engineering and Management*, vol. 145, p. 04019006, Mar. 2019.

[23] L. Morawska, J. W. Tang, W. Bahnfleth, P. M. Bluyssen, A. Boerstra, G. Buonanno, J. Cao, S. Dancer, A. Floto, F. Franchimon, C. Haworth, J. Hogeling, C. Isaxon, J. L. Jimenez, J. Kurnitski, Y. Li, M. Loomans, G. Marks, L. C. Marr, L. Mazzarella, A. K. Melikov, S. Miller, D. K. Milton, W. Nazaroff, P. V. Nielsen, C. Noakes, J. Peccia, X. Querol, C. Sekhar, O. Seppänen, S.-i. Tanabe, R. Tellier, K. W. Tham, P. Wargocki, A. Wierzbicka, and M. Yao, "How can airborne transmission of COVID-19 indoors be minimised?," *Environment International*, vol. 142, p. 105832, Sept. 2020.

[24] J. Kurnitski, M. Kiil, P. Wargocki, A. Boerstra, O. Seppänen, B. Olesen, and L. Morawska, "Respiratory infection risk-based ventilation design method," *Building and Environment*, vol. 206, p. 108387, Dec. 2021.

[25] B. S. Chaudhari, M. Zennaro, and S. Borkar, "LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations," *Future Internet*, vol. 12, p. 46, Mar. 2020.

[26] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios," *IEEE Wireless Communications*, vol. 23, pp. 60–67, Oct. 2016.

[27] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Express*, vol. 5, pp. 1–7, Mar. 2019.

[28] R. K. Singh, P. P. Puluckul, R. Berkvens, and M. Weyn, "Energy Consumption Analysis of LPWAN Technologies and Lifetime Estimation for IoT Application," *Sensors*, vol. 20, p. 4794, Aug. 2020.

[29] S. Farrell, "Low-Power Wide Area Network (LPWAN) Overview," May 2018. Issue: 8376 Num Pages: 43 Series: Request for Comments Published: RFC 8376.

[30] B. S. Chaudhari and M. Zennaro, eds., *LPWAN technologies for IoT and M2M applications*. London: Academic Press, 2020. OCLC: on1127120750.

[31] P. Thubert, A. Pelov, and S. Krishnan, "Low-Power Wide-Area Networks at the IETF," *IEEE Communications Standards Magazine*, vol. 1, pp. 76–79, Mar. 2017.

[32] M. I. Hossain and J. I. Markendahl, "Comparison of LPWAN Technologies: Cost Structure and Scalability," *Wireless Personal Communications*, vol. 121, pp. 887–903, Nov. 2021.

[33] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A Primer on 3GPP Narrowband Internet of Things," *IEEE Communications Magazine*, vol. 55, pp. 117–123, Mar. 2017.

[34] A. Adhikary, X. Lin, and Y.-P. E. Wang, "Performance Evaluation of NB-IoT Coverage," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, (Montreal, QC, Canada), pp. 1–5, IEEE, Sept. 2016.

[35] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, (Athens), pp. 197–202, IEEE, Mar. 2018.

[36] "Analyzing NB IoT and LoRaWAN® Sensor Battery Life." https://tech-journal.semtech.com/analyzing-nb-iot-and-lorawan-sensor-battery-life.

[37] Semtech Corporation, "LoRa Modulation Basics." https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf.

[38] Semtech Corporation, "LoRa Design Guide." https://www.rs-online.com/designspark/rel-assets/ds-assets/uploads/knowledge-items/application-notes-for-the-internet-of-things/LoRa%20Design%20Guide.pdf.

[39] LoRa Alliance, "LoRaWAN® Regional Parameters." https://lora-alliance.org/resource_hub/lorawan-regional-parameters-v1-1ra/, 2017.

[40] LoRa Alliance, "LoRaWAN Specification." https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf.

[41] Semtech Corporation, "LoRa and LoRAWAN: A Technical Overview." https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf, Feb. 2020.

[42] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, p. 1466, Sept. 2016.

[43] The Things Industries, "What is the things stack?." https://www.thethingsindustries.com/docs/getting-started/what-is-tts/.

[44] myDevices, "Cayenne docs." https://developers.mydevices.com/cayenne/docs/lora/.

[45] J. M. Marais, A. M. Abu-Mahfouz, and G. P. Hancke, "A Review of LoRaWAN Simulators: Design Requirements and Limitations," in *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, (Vanderbijlpark, South Africa), pp. 1–6, IEEE, Nov. 2019.

[46] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, (Malta Malta), pp. 59–67, ACM, Nov. 2016.

[47] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *2017 IEEE International Conference on Communications (ICC)*, (Paris, France), pp. 1–7, IEEE, May 2017.

[48] R. Marini, K. Mikhaylov, G. Pasolini, and C. Buratti, "LoRaWANSim: A Flexible Simulator for LoRaWAN Networks," *Sensors*, vol. 21, p. 695, Jan. 2021.