



POLITECNICO DI TORINO

Master degree course in Communications and Computer Networks  
Engineering.

Master Degree Thesis

**System Characterization for  
Cost-Effective and  
Energy-Efficient Food Safety  
Management with Wireless  
Communication and Sensing  
Integration**

**Supervisors**

prof. Micaela Demichela

**Candidate**

Jorge ROMERO

matricola: 301760

ANNO ACCADEMICO 2022-2023

This work is subject to the Creative Commons Licence

# Summary

Food is essential for the vitality of humans and other living organisms. Millions of people and even animals are exposed to health problems associated with food hygiene, storage, the supply chain, chemical additives, fungi and bacteria. The development of technologies that can enhance food safety control and monitoring has significant environmental benefits and reduced health risks for the population. Supply chains, increasingly extended and globalized, represent one of the most important challenges for food safety. It is necessary to overcome the concept of traceability to obtain more timely and targeted interventions, resulting in less production losses, less waste, less energy consumption, and high quality in the final consumer product.

This thesis will characterize new and innovative low-cost data acquisition tools that can be implemented along supply chains with block-based distributed control systems that will be part of a risk-based decision making support system. The integration of sensors with Wireless communication systems allows to collect, analyze and share data in an agile way, with continuous availability and allowing to cover large physical extensions.

The validation of the proposed system to be characterized was carried out by means of a case study for the hazelnut supply chain in the Piedmont region of Italy.

# Acknowledgements

I would like to express my heartfelt gratitude to my family for their unwavering support and understanding throughout this academic journey, especially for their patience during the times when I had to be away from home in pursuit of my studies.

I extend my sincere appreciation to my dear friend, Luis Alberto, whose constant presence and assistance were invaluable sources of motivation and encouragement during the challenges I encountered along the way.

I am deeply indebted to Professor Micaela Demichela and Engineer Lidmila Dobriakova for their exceptional guidance, unwavering support, and expert direction throughout the course of this project. Their mentorship and insights have been instrumental in shaping the outcome of this thesis.

Your contributions and support have played an integral role in my academic endeavors, and for that, I am truly grateful.

# Contents

List of Tables	7
List of Figures	8
<b>I Prima Parte</b>	<b>11</b>
<b>1 Introduction</b>	<b>13</b>
<b>2 Sensor</b>	<b>15</b>
2.1 Generality . . . . .	15
2.2 Biosensor . . . . .	16
2.2.1 Classifications of Biosensor . . . . .	16
2.3 Sensor Selection . . . . .	17
2.4 BME688 Sensor . . . . .	18
2.5 Pynode+ airQ . . . . .	20
2.6 Board Adafruit with BME688 . . . . .	21
<b>3 Low-power wide-area network (LPWAN)</b>	<b>23</b>
3.1 LPWAN Energy Efficiency . . . . .	25
3.2 LPWAN Long Range . . . . .	25
3.3 LPWAN Low Cost . . . . .	26
3.4 LPWAN Traffic Characteristics . . . . .	27
3.5 Security and privacy . . . . .	27
<b>4 LoRa Technology</b>	<b>29</b>
4.1 LoRa Characteristics . . . . .	30
4.1.1 Long Range of Lora . . . . .	30
4.1.2 Chirp spread Spectrum . . . . .	30
4.1.3 Coding Rate(CR) . . . . .	32

4.1.4	Link Budget . . . . .	32
4.1.5	Security . . . . .	33
4.1.6	Components of a LoRaWAN Network . . . . .	34
4.1.7	LoRaWAN link layer . . . . .	36
4.1.8	LoRa Packet Structure . . . . .	38
<b>5</b>	<b>Desing and Implementation</b>	<b>41</b>
5.1	The System Design . . . . .	41
5.1.1	Pycom BME688 Sensor Node . . . . .	42
5.1.2	Raspberry Pi Pico BME688 Sensor Node . . . . .	48
5.1.3	The Gateway . . . . .	53
5.1.4	LoRaWAN Network Server . . . . .	57
5.1.5	Visualization Dashboard . . . . .	60
5.2	The System Implementation . . . . .	61
<b>6</b>	<b>Conclusions and Future Developments</b>	<b>71</b>

# List of Tables

2.1	Sensor Specifications [15]	18
2.2	Index for Air Quality (IAQ) classification and color-coding	20
4.1	Bit Rates for Different Spreading Factors	32
5.1	Microcosm - Point Values	65

# List of Figures

2.1	Classifications of Biosensor [30]	17
2.2	BME688 Sensor	19
2.3	Pynode+ airQ	21
2.4	Board Adafruit with BME688	22
3.1	Wireless access geographic coverage [7]	24
4.1	LoRaWan architecture and Protocols [7]	34
4.2	Composition of the three LoRaWAN classes of end-devices [19]	35
4.3	LoRaWAN Network [7]	36
4.4	LoRa Physical structure of an uplink packet [28]	39
4.5	LoRa Physical structure of an Downlink packet [28]	39
4.6	LoRa Physical structure implicit mode [28]	40
5.1	LoRaWAN Desing and Implementation	42
5.2	Pycom LoPy4 [21]	43
5.3	Pycom FiPy [21]	44
5.4	LoRa Class A TX Window [21]	45
5.5	Pycom LoRaWAN BME688 Sensor Node	48
5.6	Pycom LoRaWAN BME688 Sensor Node	48
5.7	Pycom LoRaWAN BME688 Sensor Node	49
5.8	Pycom LoRaWAN BME688 Sensor Node	49
5.9	Pinout Pico-LoRa-SX1262-868M	52
5.10	Raspberry Pi Pico BME688 Sensor Node	53
5.11	Raspberry Pi Pico BME688 Sensor Node	53
5.12	Raspberry Pi Pico BME688 Sensor Node	54
5.13	Structure of a LoRa downstream package [13]	56
5.14	Fields of LoRa TX ACK [13]	57
5.15	LoRa Application server interface	59
5.16	Gateway summary interface	59
5.17	Configuration of the microcosm	62
5.18	Sensor BME688 Operational Inside	63



5.19	Microcosm Equipped with BME688 Sensor . . . . .	63
5.20	Screenshot depicting CO2 levels (ppm) recorded by the BME688 sensor from 15/05/23 to 23/05/23. . . . .	64
5.21	CO2 Concentration Display (15/05/23 - 30/05/23) . . . . .	66
5.22	Node-RED Backend Architecture . . . . .	67
5.23	The Things Network (TTN) Broker Connection . . . . .	68



Part I

**Prima Parte**



# Chapter 1

## Introduction

In recent years, the field of sensing, data acquisition, data transmission, data analysis and visualization has experienced significant growth due to its numerous applications and solutions in areas such as telecommunications, sensing and computing. Technologies such as the Internet of Things (IoT), 5G, big data analytics, and cloud computing are encountering significant challenges as they seek to disrupt established systems and processes with innovative solutions [4]. However, the constant growth of these applications and devices produces large amounts of electronic waste and at the same time also consumes a significant amount of energy [2], according to the latest Gartner study, there will be approximately 29 billion IoT connections on the web by 2027 [10], therefore, in the near future, this situation will present a challenge that requires addressing energy consumption, prompting the need for innovative approaches to establish environmentally-friendly communication throughout the network [2].

Food stands as a fundamental source of vitality for both humans and other living organisms. Health issues stemming from food hygiene, storage, the supply chain, chemical additives, fungi, and bacteria pose risks to millions of people, as evidenced by an 8.9% global population affected by food insecurity [23]. The food industry is an important part of the world's economies, production, processing and distribution are some of the multiple transformations and transactions between producers, distributors and consumers [11]. As trade relations globalize, trade agreements become more widespread and trade routes expand, the world food system is facing new challenges such as a growing world population, rapid urbanization, sustainability and climate change [22].

A food supply chain includes all stages from the harvesting of food to its consumption. With the accelerated development of disruptive technologies mentioned above, conventional ways of managing supply and delivery to the local consumer have also evolved. Supply chains that use technology to perform continuous monitoring, going beyond the concept of traceability, in order to preserve quality and optimize resources with more sustainable process approaches, which the UN Agenda 2030 [29] identifies as a central role for the private sector, will result in increased operational efficiency. Therefore food and quality control systems in the supply chain can find a meeting point between food safety requirements and at the same time maximizing financial returns for the producing companies [26].

This master thesis focuses on the study and Characterization for Cost-Effective and Energy-Efficient System for Food Safety Management with Wireless Communication and Sensing Integration. Especially through the use of new low-cost tools to collect more comprehensive information, to be shared throughout a supply chain, with block-based distributed control systems and a risk-based decision support system. The appropriate hardware available on the market will be identified to carry out data acquisition, on-site processing, that is, the identification of parameters that allow risk assessment, agile and effective decision execution. Determine the wireless transmission system to send data and exchange information between a node in the chain and the warehousing system. Prototyping, based on a case study (hazelnut supply chain), the integration of these systems, verifying the effectiveness of the technological solution through laboratory tests and a final cost-benefit analysis.

The present thesis is structured into six distinct chapters, each of which contributes to the overall objective of investigating and understanding the Food Safety Management with Wireless Communication and Sensing Integration.

# Chapter 2

## Sensor

### 2.1 Generality

Sensors represent an important part for automated and autonomous applications by measuring and processing data acquired by detecting changes in physical things. When there is a change in a physical condition for which the sensor is designed, this produces a response, usually electrical, which can be measured. There are different types of sensors that can be classified from the simplest to the most complex, this classification can be made taking into account different parameters such as manufacturing methods, specific materials, measurement ranges and accuracy. According to Sehwat [25], sensors designed for IoT are characterized by enabling intelligent environments and can be effectively used in health, water, transportation, home, and agriculture applications, obtaining sufficiently accurate answers taking into account low production costs, low energy consumption, portability and adaptation to environmentally unfavorable conditions, i.e. polluted environments and sometimes with extreme temperatures not suitable for human presence.

In the food industry there is a wide range of analytical methods to identify chemical and biological contaminants. Strict quality controls have allowed the development and evolution of conventional methods such as bacterial counts, immunological methods which employ antigens and antibodies, chromatographic analysis, and other analytical methods. Traditionally this type of identification of pathogens or agents hazardous to human health requires specialized instrumentation and dedicated laboratory testing. Recent advances in IoT technology have opened the door to the development of new and innovative devices that bring the laboratory closer to every stage involved

in food supply chains by allowing simple periodic analysis and immediate sharing of collected data with experts. [5]

## 2.2 Biosensor

Biosensors are developed in response to the limitations of conventional techniques used to detect potentially harmful substances for human consumption and/or exposure. The purpose is to have tools capable of identifying potentially harmful compounds in the food industry in an industrial, economical, sustainable and transportable way. A biosensor is presented as an analytical device that receives as input a physical alteration or change of biological or chemical character and whose response will be electrical. Essentially, it is composed of a bioreceptor (biological mediator in charge of selectivity) and a transducer. The bioreceptor interacts with the analytical target, converting the biological response into an electrical signal by means of the transducer. Subsequently, by means of signal processing techniques, such as matching, attenuation, amplification, filtering, etc. Interpretable values will be obtained to describe the phenomenon of interest [30].

### 2.2.1 Classifications of Biosensor

According to Velusamy biosensors can be classified according to the type of bioreceptor or by the type of transducer. A bioreceptor, in turn, is a molecular structure that uses a biochemical mechanism for the identification of physical changes, by means of which, after some interpretation, the measurement can be made. Bioreceptors can be classified at the same time into four categories including antibody, enzyme, DNA nucleic acids and cellular structures. The classification with respect to the transduction method used by the biosensor can be made taking into account the technology with which they are constructed, in that sense, they can be identified as optical, electrochemical, and mass based. Each of these categories includes some subclasses that differ essentially in the construction methods and materials involved in their conception [30].

Sensors will be a vital component for the proposed IoT system. These devices can detect and monitor external environments and then replace that information with a signal that humans and machines can read and distinguish [1]. Sensors can be active or passive, analog or digital. The most commonly used sensors in IoT systems are the following: Temperature sensors, Humidity



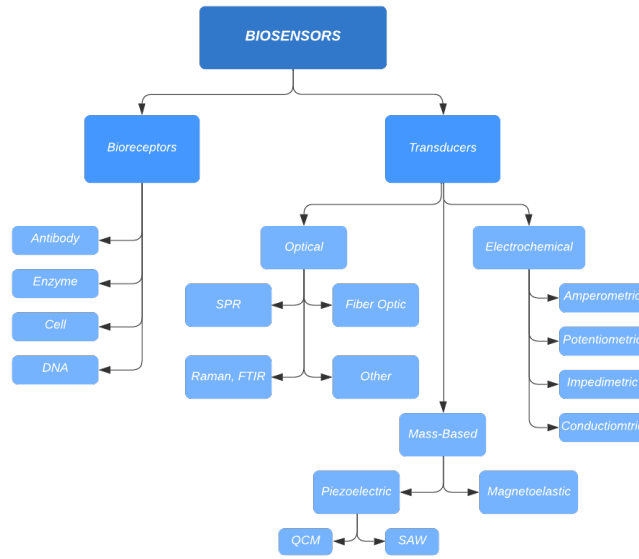


Figure 2.1. Classifications of Biosensor [30]

sensors, Pressure sensors, Proximity sensors, Level sensors, Accelerometers, Gyroscopes, Gas sensors, Infrared sensors and Optical sensors. They play a key role in improving operational efficiency, reducing costs and improving worker safety and efficiency [24]. The choice of the transducer to be adopted depends on the type of biochemical modification; in fact, the bioreceptor and the transducer must be compatible in order to have an electrical output signal.

## 2.3 Sensor Selection

According to the technical requirements previously identified and taking into account the main objective of the project, i.e. an IoT system capable of managing supply chains for food safety by integrating low-cost and energy efficient technologies. As recommended by Marin [15] four different sensors (CCS811, MICS-VZ-89TE, BME680, BME688) for AQ assessment and ease of data acquisition were evaluated to complete such purpose at hardware level and as a first step in the development stage.

To make the choice, the methodology proposed by Das [8] was used, which consists of making comparisons in the principles of operation, consigned in

the data sheets available in the official documentation of each one, and giving positive value weights to the advantages and negative value weights to the disadvantages of each device. The comparative results at the physical construction level, operating ranges, consumption and suitability for the final solution are shown in the table 2.1.

Specifications	Sensor			
	CCS811	MICS-VZ-89TE	BME680	BME688
Manufacturer	AMS	SGX Sens	Bosh	Bosh
Range tVOC (ppb)	0–1200	0-2000	NA	0.3-10000
Range eCO2 (ppm)	400-8192	400-2000	NA	400-10000
Additional parameters	-	-	T, RH, p	T, RH, p
Operating voltage, (V)	1.7-3.6	1.65–3.6	1.7-3.6	1.7-3.6
Dimensions, mm	2.7x4x1.1	6.9×5×1.55	3×3×0.93	3×3×0.93
Response time, ts	< 5	< 5	< 3	< 3

Table 2.1. Sensor Specifications [15]

Following the selection process according to the methodology and with respect to the results observed in the table 2.1, the sensor identified for the development process is the BME688.

## 2.4 BME688 Sensor

The BME688 2.2 is a complex digital environmental sensor that can detect gases, volatile organic compounds and measure the temperature, pressure, and humidity of the targeted area. "It is housed in a robust yet compact 3.0 x 3.0 x 0.9 mm<sup>3</sup> package and especially developed for mobile and connected applications where size and low power consumption are critical requirements.

The Gas sensor can detect Volatile Organic Compounds (VOCs), volatile sulfur compounds (VSCs) and other gases such as carbon monoxide and hydrogen in the part per billion (ppb) range".



Figure 2.2. BME688 Sensor

Regarding the gas value, the sensor is tested with hydrogen sulfide, ethanol, carbon monoxide and volatile organic compounds produced during exhalation. Also through the BSEC (Bosch Software Environmental Cluster) library it is possible to obtain, from the resistive value, an index for air quality (IAQ), a value for CO<sub>2</sub> and by default it is possible to obtain the probability for the presence of H<sub>2</sub>S. The IAQ is a complex concept because it is determined by different variable parameters, some of them are indoor and outdoor penetration of pollutants, chemical reactions, air exchange, ventilation characteristics, temperature and relative humidity [20]. The IAQ output delivered by the BME688 sensor is an index that can have values between 0 and 500 with a resolution of 1 to indicate or quantify the quality of the air available around the sensor. The detailed classification and color coding for the IAQ index is described in the table 2.2.

The sensor is sensitivity when testing with chemicals such as benzene, toluene and ethylbenzene. The BME 688 includes artificial intelligence based software features that provide the ability to adjust selectivity levels and sensitivity towards specific compounds in ppb ranges, however, the lengthy initial self-calibration and adjustment time prior to making measurements or

obtaining data should be noted. In addition, the sensor will provide concentrations with a slow response time, i.e. the measurement result is provided over time after the change in concentration. This response time is dependent on the time taken for self-calibration. [15]

According to the manufacturer’s official documentation, the main use cases evaluated for Hardware-Software systems with the BME688 sensor are:

- Selectivity to target gas classes
- Provision for custom use case development
- Calculation of index for air quality (IAQ) level outside of the device
- Calculation of ambient air temperature outside of the device
- Calculation of ambient relative humidity outside of the device

IAQ index values	Air Quality
0-50	Excellent
51-100	Good
101-150	Lightly polluted
151-200	Moderately polluted
201-250	Heavily polluted
251-350	Severely polluted
> 351	Extremely polluted

Table 2.2. Index for Air Quality (IAQ) classification and color-coding

For measuring environmental values, the choice was initially based on the availability of the sensor BME688 in the market. Two options were evaluated. The first, supplied by Pycom (Pynode+ airQ) and the second, the Board Adafruit with Bosch BME688.

## 2.5 Pynode+ airQ

The Pynode+ airQ is a Bluetooth Low Energy (BLE) enabled sensor, Air Quality and Temperature sensor featuring Dialog Semiconductor DA14531 BLE 5.1 and Bosch BME688 Air Quality sensor.

When operating in a central role for initial configuration, the Pynode+ airQ sensor exhibits an average power consumption of 446 $\mu$ A, with a maximum peak of 1.67mA. In peripheral and advertising mode, where the sensor sleeps for one minute intervals, it showcases an average power consumption of 7.16 $\mu$ A, with a peak consumption of 1.00mA. These power consumption figures highlight the sensor’s ability to maintain efficient operation even in scenarios where energy conservation is critical.

Serves as a dependable solution for a wide range of applications that demand precise temperature and humidity monitoring. Whether it’s integrated with Pycom’s WiPy, GPy, LoPy4, FiPy, OEM modules, or other BLE 5.1-compatible devices, this sensor stands out as a reliable choice for collecting critical environmental data with efficiency and accuracy.



Figure 2.3. Pynode+ airQ

## 2.6 Board Adafruit with BME688

The Board Adafruit with BME688 gives the environmental sensing in one package. Contains temperature, humidity, barometric pressure, and VOC gas sensing capabilities, all over SPI or I2C. The Board Adafruit with BME688 has a 3.3V regulator and some level shifting so it can be easily used with different kind of microcontrollers. Allows integration with MCU boards such as Raspberry Pi Pico or Arduino.

The precision sensor developed by Bosch can accurately measure humidity to within  $\pm 3$  percent, provide barometric pressure measurements with an absolute accuracy of  $\pm 1$  hPa, and provide temperature readings with an accuracy level of  $\pm 1.0$  °C. In addition, due to the strong correlation between

pressure and altitude, the sensor allows its use as an altimeter, achieving an altitude measurement accuracy of  $\pm 1$  meter [15].

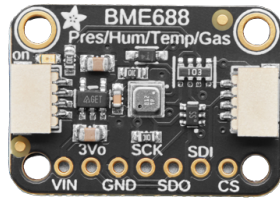


Figure 2.4. Board Adafruit with BME688

## Chapter 3

# Low-power wide-area network (LPWAN)

Low-power wide-area network (LPWAN) is an interesting solution for long-range and low-power communication applications like Internet of things (IoT) and machine-to-machine communications (M2M).

The LPWANs are networks with demanding requirements for extended coverage, high scalability, low power consumption and low device and deployment costs, which means that LPWANs are resource-constrained networks.

Currently, wireless technologies such as IEEE 802.11 wireless local area networks (WLAN), IEEE 802.15.1 Bluetooth, IEEE 802.15.3 ZigBee, Low-rate personal area networks (LR-WPAN), are used in applications for short-range environments, unlike cellular wireless networks such as 2G, 3G, 4G and 5G that can be used for long-range applications. Initially networks such as WLAN and Bluetooth were designed for high speed data transmission, while ZigBee and LR-WPAN were designed as local area networks and are used for low rate applications and distances between a few meters and a few hundred meters, depending on the line of sight, obstacles present in the environment, interference, transmission power, etc. Cellular wireless networks such as 2G, 3G and 4G, although designed for voice and data communication, have been occasionally used for sensing applications, as a solution to the main problem of reaching long distances, this practice may have applicability in some scenarios but is not attractive in terms of power consumption, performance and implementation costs or use with third party service providers.[7].

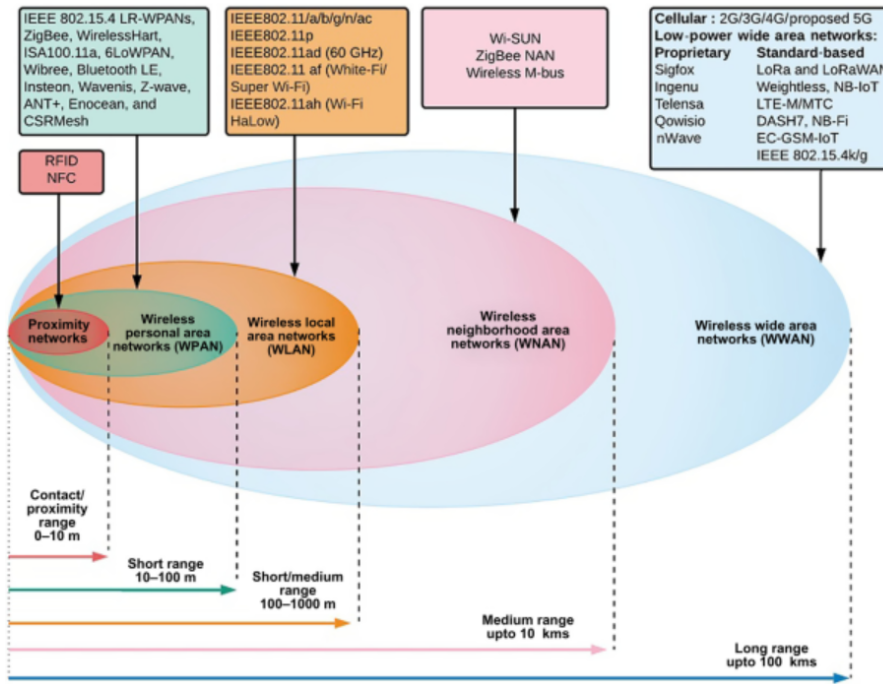


Figure 3.1. Wireless access geographic coverage [7]

LPWANs have to meet certain requirements in terms of reachability and retainability, some of which relate to energy efficiency, transmission integrity in a long-range communication scenario, low-cost operation and support for uplink-dominated connections (UL).

LPWAN applications require connections between various devices, spanning varying distances and different power sources in diverse environments. These solutions typically involve small and infrequent messages, are delay tolerant, and prioritize energy efficiency and low cost. Coverage requirements vary by application, from indoor deployments to global needs for asset tracking. These applications are divided into bulk and mission-critical IoT categories, based on latency and reliability needs. Key features include M2M traffic management, massive capacity, energy efficiency, extended coverage, security and interoperability. [15]



## 3.1 LPWAN Energy Efficiency

To ensure the availability of a system in LPWAN technologies, special attention must be paid to power consumption during the transmission process. Careful transmission planning and the use of efficient radio interfaces are essential. In addition, the implementation of lightweight protocols is essential, as they not only minimize the number of transmissions, but also their duration. [17]

In some situations, devices operate on limited power, often through solar panels, highlighting the importance of energy efficiency in network management. In many LPWAN applications, it is not possible to recharge the batteries, and these batteries, such as AA-type batteries, are expected to last more than 10 years without recharging. Power loss or even battery replacement may not be feasible in short time intervals. Therefore, it is crucial to keep battery source costs low and operate with a strict and very low duty cycle to maximize the lifetime of the nodes in LPWAN. [15]

## 3.2 LPWAN Long Range

To better understand LPWANs, it is important to consider their reach and coverage capabilities. In general, these networks are designed to provide long distance communications, covering up to 10-40 kilometers in rural or desert areas, and around 15 kilometers in urban areas, outperforming traditional cellular networks with a gain of 120 dB.

The need for coverage is not only limited to outdoors, as a robust presence is also required indoors, including hard-to-reach places such as subways and basements. This is essential for monitoring and data collection applications, and coverage must be matched to expectations for data rate and controlled data error rates.[17]

A key element for LPWAN performance is the use of the sub-GHz band, which enables robust and reliable communication with lower power consumption. The lower frequencies in this band offer better propagation compared to the 2.4 GHz band. In addition, LPWANs take advantage of slow modulation techniques, which means higher power consumption per bit, but in return improves coverage and allows receivers to accurately demodulate signals.

The very name, LPWAN, indicates the importance of long-range support in this technology. In practice, these networks establish links ranging from  $10^3$  to  $10^4$  meters, with records up to  $10^5$  meters in experimental configurations. The  $10^3$  meter range is typical in urban or suburban environments, while the  $10^4$  meter range is achieved in areas without obstacles and clear line of sight, such as rural environments. In addition,  $10^5$  meter range has been demonstrated to be possible, even in experiments that have reached impressive altitudes, such as transmissions from weather balloons at 766,000 meters.[15]

### 3.3 LPWAN Low Cost

To better understand costs in LPWAN networks, it is important to consider both the access side and the server side. On the access side, Gateways (GW) are deployed in limited numbers due to their range and function as packet forwarders, transferring most of the functionality to the server side. This strategy minimizes the costs associated with Gateways.

On the other hand, End Devices are deployed in large quantities, driving the need to minimize implementation costs. This is achieved by restricting the functionality of the devices and limiting hardware and software requirements. Therefore, it is possible to build these devices directly using suitable radio modules and common development boards, such as Arduino, Raspberry Pi, or Beagle Board.[15] [17] [27]

On the server side, addressing operational and device costs in LPWAN applications is crucial. In addition to the usual requirements of low deployment and operating costs, the large number of connected devices imposes significant constraints in terms of costs, operational expenses, and the imperative for low power consumption. The ability to update software without changing the hardware is a key attribute that must be supported.[17]

Furthermore, the solution must be scalable, easy to install and maintain, and cost-effective within the total cost of ownership. These elements are essential to meet the needs of LPWAN applications at a reasonable cost.

## 3.4 LPWAN Traffic Characteristics

For a better understanding of LPWAN networks, it is essential to consider the access part, which supports wireless sensor networks (WSN) [3]. It is mainly characterized by being dominant in upstream communication (UL), that is, most transmissions originate from the End Devices to the Gateways (GW). End Devices are typically deployed in field environments and are designed to provide information collected by their sensors to the central server. This can be based on a periodic analysis scheme or on specific events. Unlike conventional wireless sensor networks that use a mesh or ad-hoc topology, LPWANs follow a star system approach [18].

The inherent communication mechanism in LPWAN networks is traffic generated by distributed sensors. In addition to the possible presence of traffic generated by devices such as smartphones, LPWAN traffic itself can vary across a wide range of attributes, such as the number of messages, message size, and reliability requirements. LPWAN technologies span various application categories, each with different requirements. Some applications are delay-tolerant, such as smart metering, while others, such as fire detection, nuclear radiation, and home security, require immediate, priority transmissions. In certain situations, it may be necessary to schedule priority messages for event-triggered broadcasts. With the large number of active devices, there is a possibility that each application's service level (SLA) requirements may not be met. Therefore, mechanisms are required that allow different types of traffic to coexist, provide the necessary quality of service (QoS), and meet SLAs. [18] [15] [17] [27]

In LPWAN applications, it is important to consider managing multiple classes of End Devices according to their communication needs, both in the uplink and downlink. Some applications may require device mobility support, which involves the ability to maintain connectivity at any location.

## 3.5 Security and privacy

LPWAN devices face especially rigorous security requirements due to their large number, vulnerabilities, and simplicity. To ensure the integrity and protection of these devices, it is essential to support a number of security attributes, including authorization, authentication, trust, confidentiality, data security, and non-repudiation.

Security must be able to address malicious attacks, such as worms, as well as hacking of LPWAN devices and systems. You must also manage eavesdropping, sniffing, and denial-of-service attacks. Protecting device identity and location privacy is essential.

Additionally, security must be flexible and tailored to the requirements of specific applications, providing both forward and reverse transmission security as needed.[15]

LPWANs, or Low Power Wide Area Networks, are networks with limited resources and demanding requirements, including long battery life, expanded coverage, scalability, and low device and deployment costs. They face numerous challenges, such as network virtualization, simplification of media access control, dynamic spectrum management, link optimization, energy harvesting, network restrictions. duty cycle, scalability, localization, coexistence, interference mitigation, mobility, higher data rates and QoS. support, security, privacy, congestion control, SLA compliance, integration with data analytics, and leveraging artificial intelligence and machine learning to improve performance. Addressing these issues and expanding the range of LPWAN applications requires extensive research and development to make these networks more competitive with other cellular technologies.

# Chapter 4

## LoRa Technology

LoRa, which is a registered trademark of Semtech, represents a modulation technique rooted in Chirp Spread Spectrum (CSS) modulation. It primarily defines the physical layer, often referred to as the 'bits layer,' within the OSI Model. The core objective behind this protocol is to cater to the needs of IoT devices powered by batteries, emphasizing low power consumption.

An outstanding attribute of CSS is its capability to establish long-range communication while demonstrating remarkable resilience to interference. Remarkably, a single gateway has the potential to cover vast areas, ranging from entire cities to hundreds of square kilometers. [13]

LoRa technology offers the flexibility to operate within public, private, or hybrid network environments, delivering an extended communication range when compared to cellular networks. It seamlessly integrates with existing infrastructure, making it a cost-effective solution, particularly for low-power Internet of Things (IoT) applications.

LoRa, as a physical layer technology, employs a proprietary spread spectrum technique to modulate signals within the Sub-GHz ISM band supports different frequencies, namely 868 (Europe), 915 (North America), and 433 MHz (Asia) [27]. Data transmission rate that can range from 300 bps to 50 kbps can varies depending on the chosen spreading factor (SF) and channel bandwidth settings. In the context of the Internet, LoRa has been managed by the LoRa Alliance, which has developed the Long-Range Wide-Area Network, or LoRaWAN. LoRaWAN join both network infrastructure and upper-layer functionalities, making it a solution for long-range communication. One of LoRaWAN's advantages is the ability to link to diverse IoT applications. To accommodate a wide range of needs, LoRaWAN provides

three classes of end devices. These classes are in charge to address different application requirements like latency [15]. This unique approach allows for the concurrent exchange of multiple frames utilizing various spreading factors.

LoRaWAN adopts a star architecture, with gateways serving as intermediaries for transmit messages between end-devices and a central core network. In the LoRaWAN architecture, messages transmitted by end-devices reach all accessible base stations. These base stations are interconnected with core network services, and they manage the packets through addressing IP connections. [27]

## 4.1 LoRa Characteristics

### 4.1.1 Long Range of Lora

LoRa, is a protocol that can send data over really long distances. With just one device, it can cover a really big area, as big as hundreds of square kilometers. This long-distance capability is because of two things: the link budget and a special kind of signal modulation called chirp spread spectrum.

### 4.1.2 Chirp spread Spectrum

The chirp spread spectrum doesn't need much power to transmit signals. Chirp is a signal whose frequency increases or decreases over time. Thus, a chirp signal can be up-chirp and down-chip.

In chirp spread spectrum modulation, the desired data signal gets combined with the chirp signal. This enhance the frequency range beyond that of the initial data signal. When the signal reaches the receiver, it undergoes multiplication with a locally generated copy of the chirp signal. This action restores the modulated signal to its original frequency range, effectively minimizing unwanted noise and interference [9].

The LoRa modulation Bit Rate can be expressed as:

$$R_b = SF \cdot \left( \frac{1}{2^{SF}} \right) \text{ bit/sec} \quad (4.1)$$

$R_b$	Bit Rate
$SF$	Spreading Factor
$BW$	Bandwidth

Expanding the signal bandwidth enables the successful transmission of error-free data over long distances. This can be illustrated by comparing the sensitivity of LoRa modulated signals to frequency-shift key modulated signals. The chirp spread spectrum modulation exhibits significantly greater sensitivity compared to frequency-shift key modulation.

Every Chirp in the LoRa Signal contains  $2^{SF}$  Chip, if we want to know how to calculate the Time a LoRa symbol would take then we can use the following equation:

$$T_c(sec) = \left( \frac{1}{BW} \right) \quad (4.2)$$

Understanding the selected Spreading Factor (SF) used in transmission allows us to determine the duration of a Chirp (Symbol):

$$T_c(sec) = \left( \frac{2^{SF}}{BW} \right) \quad (4.3)$$

Examining equation 4.3 shows the clear influence of both SF and BW on symbol timing. This factor plays a significant role in managing the Time on Air (TOA) for a LoRaWAN Packet, which is a crucial parameter and a recognized limitation when utilizing the LoRaWAN network.

The Spreading Factor (SF), defined as  $SF = \log_2 \left( \frac{R_c}{R_s} \right)$ , represents the relationship between the symbol rate ( $R_s$ ) and chip rate ( $R_c$ ). In LoRa, there are six distinct spreading factors ranging from 7 to 12. SF plays an important role in balancing data rate and transmission range by enhancing the receiver's sensitivity. Additionally, LoRa incorporates forward error correction (FEC) techniques to further enhance receiver sensitivity [15].

In Table 4.1. spreading factor has a significant effect on bit rate. These values should be considered for the applications.

Spreading Factor	Bit rate (bit/s)
7	5469
8	3125
9	1758
10	977
11	537
12	293

Table 4.1. Bit Rates for Different Spreading Factors

### 4.1.3 Coding Rate(CR)

Code rate CR defines the amount of forward error correction (FEC) in LoRa frame. LoRa offers CR = 0, 1, 2, 3, and 4, where CR = 0 means no encoding. The general formula for the coding rate is  $CRC = \left(\frac{4}{4+CR}\right)$  where CRC stands for Cyclic Coding Rate.

Select higher Spreading Factor (SF) and Coding Rate (CR) values significantly extends the Time on Air (ToA),  $T_{air}$ . A larger bandwidth (BW) value will decrease  $T_{air}$  but at the cost of reduced receiver sensitivity. LoRa offers various bandwidth options, ranging from 7.8 to 500 kHz, with 125, 250, and 500 kHz being the most commonly employed choices. Considering these factors, the effective bit rate  $R_b$  is calculated as follows [9]:

$$R_b = \left( \frac{4 \cdot SF \cdot BW}{(4 + CR) \cdot 2^{SF}} \right) \text{ bit/sec} \quad (4.4)$$

### 4.1.4 Link Budget

LoRa technology has a stronger link budget compared to other technologies, which is a major reason for its long-range capability. A link budget is like a balance sheet that adds up all the gains and losses in a communication system. We can express the link budget of a network as follows [17]:

$$\begin{aligned} PRX(\text{dBm}) &= PTX(\text{dBm}) + G_{\text{SYSTEM}}(\text{dB}) \\ &\quad - L_{\text{SYSTEM}}(\text{dB}) - L_{\text{CHANNEL}}(\text{dB}) - M(\text{dB}) \end{aligned} \quad (4.5)$$

Where,



$PRX(dBm)$ : received power

$PTX(dBm)$ : transmitted power

$GSYSTEM(dB)$ : system gains such as those associated with directional antennas, etc.

$L_{SYSTEM}$ : losses associated with the feed-lines, antennas, etc.

$L_{CHANNEL}$ : losses due to the propagation channel

$M$ : fading margin

The link budget is the power received by a device. LoRa has a strong link budget, which makes it very sensitive. Many IoT technologies use a method called frequency shift key (FSK) modulation. When the data speed of LoRa is four times faster than FSK, it has the same sensitivity. As a result, LoRa can reach farther than other methods.

### 4.1.5 Security

This technology uses AES encryption and IEEE 802.15.4/2006 Annex B for security and making sure a device is who it says it is. Most technologies have one layer of security, but LoRa uses two: one for the network and another for your data. The network security checks if a device belongs in the network, while the application security guards your data from network operators. LoRa uses two keys for security: NwkSKey and AppSKey [19].

To join a network, a device must be activated and verified. This tech has two ways to do this:

- Over the air activation (OAA)
- Activation by personalization (ABP)

#### **Over the air activation (OAA)**

In this kind of device activation, the device isn't set up with any specific details. When it wants to join a network, it goes through a joining process. Before joining, the device gets the necessary information. But this process has to be done every time it sends data over the network if it loses its session information. This way, the device isn't tied to any one service provider and can connect to any network service provider, even when roaming.

## Activation by personalization (ABP)

In this activation type, the device already has the necessary information to connect to a specific network. When the device starts up, it automatically joins the network defined in that information.

This type of personalization is not commonly used and is reserved for special cases. The more common method is Over the Air Activation (OAA). In ABP, the device simply sends a request to join the network, and the network accepts it. Every device has its own specific NetSKey and AppSKey. Besides being an activation method, both of these methods also provide security and make sure the device is genuine on the network [19].

### 4.1.6 Components of a LoRaWAN Network

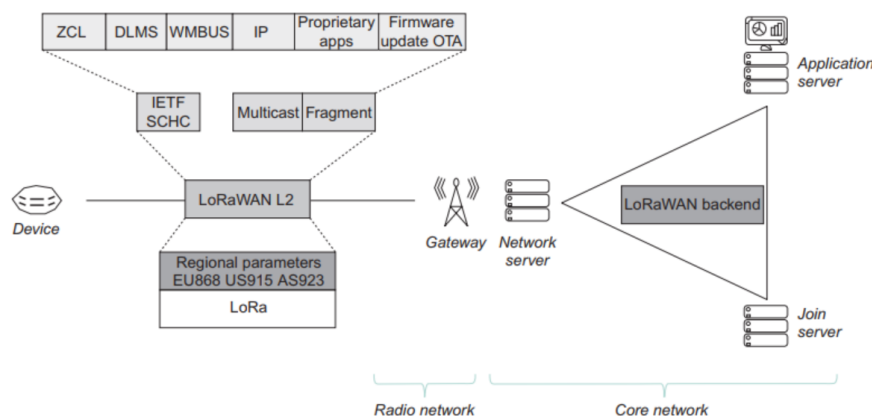


Figure 4.1. LoRaWAN architecture and Protocols [7]

A LoRaWAN network is made up of three main parts: end-devices, gateways, and the network server, as shown in Fig 4.1

#### End-devices

These are small, low-power sensor devices that use LoRa to communicate with gateways.

- Class A End-devices: These devices can send data whenever they want, but after each of their messages, they have two short moments when

they're listening for a reply. If the server wants to send a message at any other time, it has to wait until the next message from the device. Class A devices use the least power but are not very flexible for receiving messages.

- Class B End-devices: These devices have set times when they listen for messages from the server, and they need the gateway to send a signal at regular times to stay in sync. This way, the server knows when the device is ready to receive.
- Class C End-devices: These devices are almost always ready to listen, so they use the most power. The server can send messages to them whenever it needs to.

LoRaWAN only allows communication between devices and the network server. If devices want to talk to each other, it has to go through the network server and use two gateways to make it work. Fig 4.2 shows the differences between the different types of end devices in terms of battery consumption and latency.

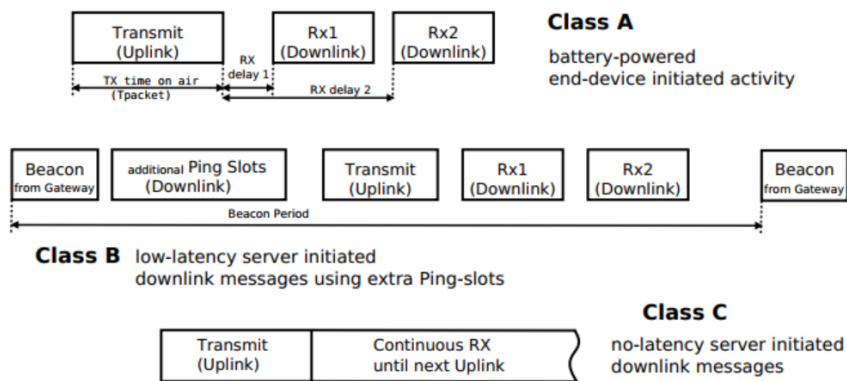


Figure 4.2. Composition of the three LoRaWAN classes of end-devices [19]

## Gateways

Gateways are like middlemen. They receive data from end-devices and send it to the network server over an internet connection like Ethernet or cellular. A single data packet can be received and sent by more than one gateway.

## Network Server

The network server is like the boss. It manages the entire network. It keeps track of which end-devices are active. When a new end-device wants to join, the network server checks with the application server to see if it's allowed. It also figures out the best settings for that device. The network server makes sure data is sent securely and deals with any data that comes from multiple gateways. It also talks to the application server to decide if a response is needed. Besides handling data, it's in charge of the end-device's settings, like data rate and channels, using something called MAC-commands.

### 4.1.7 LoRaWAN link layer

LoRaWAN serves as the data-link protocol responsible for overseeing the entry of LoRa devices into the network. It manages data exchange, data transmission rates, addresses, encryption, and various packet configurations. LoRaWAN is similar to the IP protocol, while LoRa can be likened to the physical layer, much like Ethernet's role in connectivity [7].

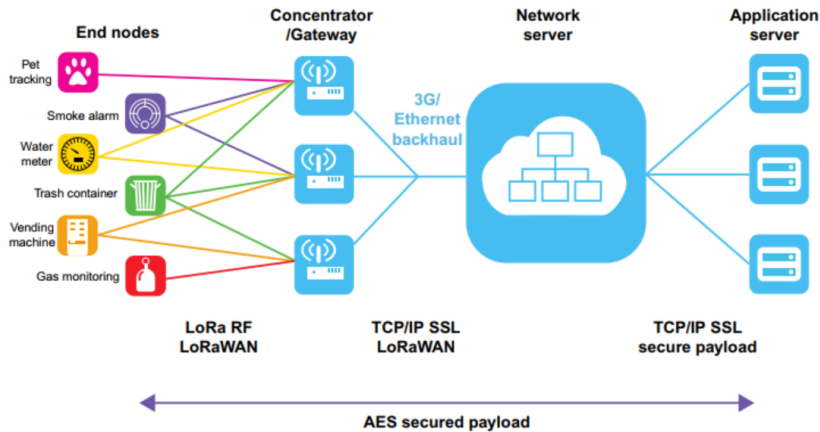


Figure 4.3. LoRaWAN Network [7]

LoRaWAN networks are organized in a star-of-stars topology, where gateways act as intermediaries between devices and a central network server (NS) as shown in Fig 4.3. This central NS routes data between network gateways and their corresponding application servers. Communication is mostly two-way, with a stronger emphasis on uplink communication, where devices send

data to both the network and application servers. Moreover, multiple gateways can receive data from devices, and there's no fixed pairing between devices and gateways.

Communication between devices and gateways involves using single-hop LoRa or frequency-shift keying (FSK) radio transmissions. LoRa transmissions spread across different frequency channels and data rates. The choice of data rate, ranging from 0.3 to 50 kbps, involves balancing communication range with packet transmission time. Lower data rates offer greater coverage but consume more airtime, and importantly, different data rates do not interfere with one another.

To optimize both device battery life and overall network capacity, the LoRaWAN network infrastructure can dynamically manage data rates and radio-frequency (RF) output power for each device through an adaptive data rate (ADR) scheme. Depending on gateway distribution, ADR can also influence how many gateways receive uplink data from a device, affecting its connectivity and redundancy within the network.

Devices have the flexibility to transmit data on any available channel and at any data rate, provided they adhere to the following rules [14]:

- Devices switch channels in a pseudo-random manner for each radio transmission, enhancing system resilience against interference.
- Devices must respect the maximum transmit duty cycle, which is specific to the subband they operate in, ensuring compliance with local regulations.

Additionally, the device's adherence to maximum transmit duration, known as dwell time, aligns with local regulations and varies by region. Specific maximum transmit duty cycles and dwell times for each subband are detailed in the LoRaWAN Regional Parameters specification, which may include other regulatory requirements like listen before talk (LBT) when applicable.

To secure radio transmissions, the LoRaWAN protocol relies on symmetric cryptography, employing session keys derived from device-specific root keys. During over-the-air activation, a join server (JS) stores the device's root keys and associated key derivation operations in the backend. Alternatively, device-specific session keys can be directly embedded in the device, a method known as activation by personalization [9].

### 4.1.8 LoRa Packet Structure

A LoRa packet has several key components, a preamble, synchronization signals, and payload chirps. It begins with a sequence of reiterated chirps, serving as a symbolic representation of the preamble. To mark the preamble's conclusion to the receiving end, it culminates with synchronization signals and two-and-a-quarter down-chirp symbols [12], resulting in a frequency shift from positive to negative. Subsequently, the packet may incorporate an elective header featuring bitrate information, followed by a payload encoded in CSS.

LoRa employs a dual approach to packet formats in data transmission, categorized as explicit and implicit modes. When operating in explicit mode, a LoRa packet incorporates the subsequent components [28]:

- Preamble, which serves the vital function of synchronizing the receiver with the transmitter. It is mandated to encompass 8 symbols across all regions, as stipulated in the LoRaWAN Regional Parameters document. Notably, the radio transmitter appends an additional 4.25 symbols, ultimately yielding a preamble length of 12.25 symbols.
- PHDR (Physical Header) is a facultative element exclusive to the explicit mode. It imparts essential information regarding payload size and includes a CRC (Cyclic Redundancy Check) for error detection.
- $PHDR_{CRC}$  (Header CRC) represents an optional field with the role of an error-detection code for rectifying errors within the header.
- Both the PHDR and  $PHDR_{CRC}$  are encoded with a Coding Rate of  $4/8$ , ensuring data integrity.
- PHYPayload constitutes the comprehensive frame generated by the MAC layer. The maximum payload size varies contingent on the Data Rate (DR) and is region-specific.
- An additional optional field, CRC, is integrated into the payload to facilitate error correction in uplink messages. Similar to the PHDR and  $PHDR_{CRC}$ , the PHYPayload and CRC are encoded with one of the Coding Rates ( $4/5$ ,  $4/6$ ,  $4/7$ , or  $4/8$ ). The complete frame is subsequently transmitted utilizing one of the Spreading Factors ( $SF = 7$  to  $12$ ).

Figure 4.4 shows the physical layer structure of uplink and downlink packets that uses explicit mode.

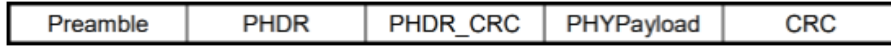


Figure 4.4. LoRa Physical structure of an uplink packet [28]

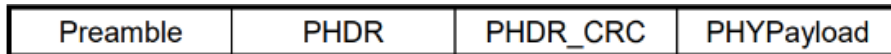


Figure 4.5. LoRa Physical structure of a Downlink packet [28]

In the implicit mode, the packet omits the header, assuming that both the payload size and Coding Rate are predetermined or previously established.

Beacons use LoRa radio packet implicit mode for sending time synchronizing information from gateways to the end devices [28].

Figure 4.6 shows the structure of a LoRa packet that uses the implicit mode.

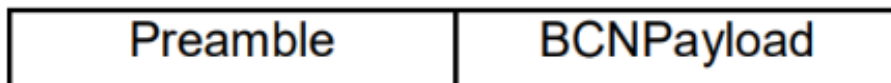


Figure 4.6. LoRa Physical structure implicit mode [28]



# Chapter 5

## Design and Implementation

### 5.1 The System Design

The system architecture can be described in terms of its key components:

**Sensor Node:** The Sensor Node is the primary data source within the system. It collects data from its environment using the selected sensor (BME688). The Sensor Node initiates the data collection process and prepares it for transmission.

**The Gateway:** The Gateway serves as the intermediary between the Sensor Nodes and the broader network. It receives data packets from Sensor Nodes and forwards them to the network server. Importantly, LoRaWAN architecture allows for multiple Gateways to receive the same packet, ensuring robust coverage and transmission reliability.

**Chirpstack:** Within this system, Chirpstack functions as the network server. It manages the communication between Gateways and Application Servers, facilitating data packet routing, security, and seamless integration with other services.

**TTN Web Servers:** The Things Network (TTN) web servers provide critical services for LoRaWAN communication. They serve as a platform for device management, data visualization, and application integration. TTN allows users to efficiently manage their Sensor Nodes and access data from

the network.

**Node-Red:** Node-Red is an essential tool within the system, offering a flow-based development environment for processing data. It is able to do the data transformations, filtering, and integration the services.

**Grafana Dashboard:** Data visualization is paramount in this system. It provides an intuitive and customizable interface for real-time visualization of collected data. Dynamic graphs, charts, and dashboards for in-depth data analysis and sensor monitoring could be created.

The system architecture seamlessly integrates these components to create a comprehensive and efficient IoT solution, in order to reach the main objective about Cost-Effective and Energy-Efficient Food Safety Management with Wireless Communication and Sensing Integration. Sensor Nodes collect real-world data, which is transmitted through Gateways and managed by Chirpstack. TTN web servers facilitate device management and application integration, while Node-Red allows for customized data processing. Finally, Grafana provides a user-friendly interface for insightful data visualization.

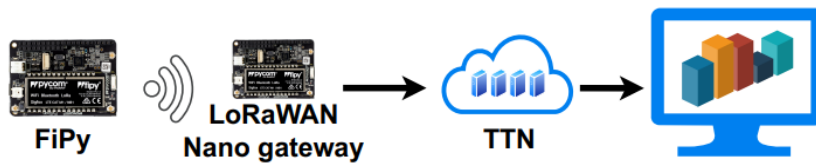


Figure 5.1. LoRaWAN Desing and Implementation

Figure 5.1 illustrates the standard connectivity of these components. The Sensors transmit data packets to The Chirpstack Network via pre-registered gateways (gateways that are officially registered on the web-server). Following this transmission, we leverage the integration feature, which enables seamless connectivity with the Back-End using the MQTT Protocol. Once the data packet reaches the Back-End platform, it can be subsequently stored, analyzed, and visualized through Grafana Dashboard.

### 5.1.1 Pycom BME688 Sensor Node

The initial phase of the project entailed the utilization of two Pycom device variants: LoPy4 boards and FiPy boards. LoPy4 boards are essentially

ESP32 microcontrollers equipped with an sx1272 hat, enabling them to harness LoRa communication frequencies. On the other hand, the FiPy board is also founded on an ESP32 platform, boasting a transceiver for LoRa and Sigfox, as well as a transceiver for LTE, along with a slot for a nano SIM card. Additionally, the ESP32s feature integrated WiFi and Bluetooth radios, which further enhance their connectivity capabilities.

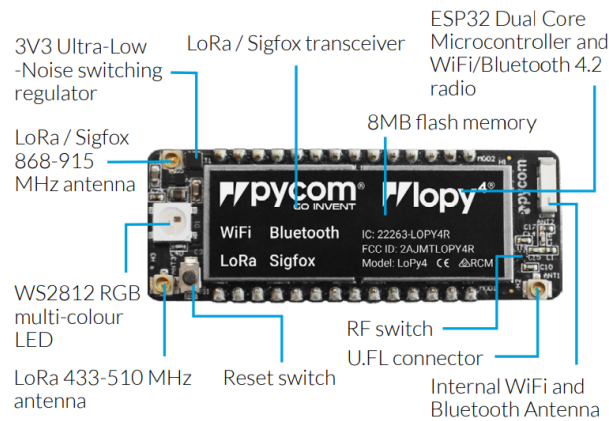


Figure 5.2. Pycom LoPy4 [21]

Pycom provides firmware specifically tailored to its boards, allowing them to interpret code written in MicroPython. Moreover, Pycom offers a range of libraries that implement common IoT features. These libraries significantly streamline the development process for communication, enabling the exploitation of various protocols, including LoRaWAN. Furthermore, Pycom provides comprehensive usage examples, serving as a valuable starting point for developing custom code.

The Pycom FiPy board is a versatile programmable device, boasting an extensive range of connectivity options, including WiFi 802.11 b/g/n, Bluetooth (both LE and classic), LoRa, Sigfox, and dual LTE-M (supporting CAT-M1 and NB-IoT) [21]. It is powered by the Espressif ESP32 chipset and is equipped with 4 MB of RAM and 8 MB of flash memory. The FiPy board comes preloaded with firmware based on MicroPython, which is an optimized implementation of the Python 3 programming language. Notably, Pycom boards are compatible with standard Python libraries, MicroPython-specific libraries, and modules specifically designed for Pycom devices [16].

The Pycom FiPy Fig 5.3 board’s features make it exceptionally well-suited for performing the role of a Gateway, owing to its versatile connectivity options, including LTE. Conversely, the LoPy4 board, with its simpler and more streamlined components, is primarily designed for use as a node. It’s worth noting that, with appropriate configuration, one can re purpose the LoPy4 board as a gateway and the FiPy board as a node, thus offering flexibility in network design.

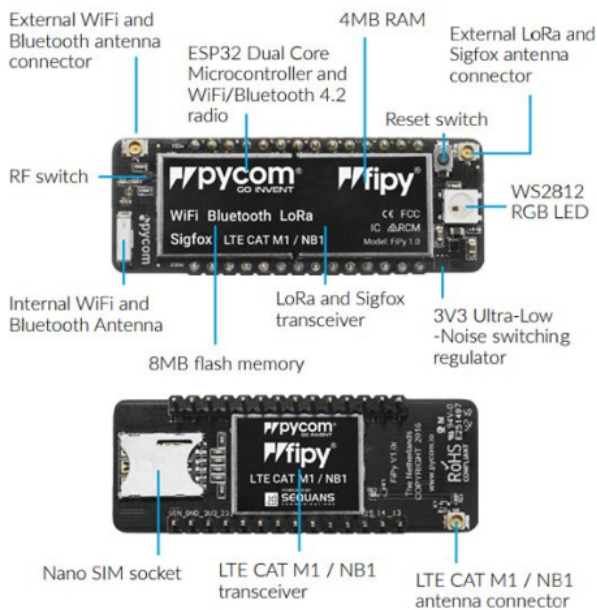


Figure 5.3. Pycom FiPy [21]

In the context of implementing the LoPy4 node, there are several critical considerations to keep in mind. First and foremost is the activation method of the node, which is initially defined within ChirpStack. Depending on whether the activation is accomplished over the air (OTAA) or through activation by personalization (ABP), different configuration parameters must be set in the device’s configuration file.

For OTAA, it’s essential to specify the DevEUI, a unique device identifier provided by the manufacturer, and the application key. The join function will then automatically populate any necessary fields left blank. In contrast, with ABP, you need to copy the values of the device address, network session key, and application session key defined in ChirpStack into the node’s configuration file. The choice between OTAA and ABP also influences how

the join process unfolds. Specifically, in the case of ABP, the join process is essentially redundant, as there’s no need for pre-data exchange between the server and the node before data transmission.

This choice of activation method also aligns with the node class. The device profile chosen during node registration on ChirpStack dictates the appropriate class setting in the node’s configuration file. In the case of a class A node, as depicted in Figure 5.12, the node initiates two consecutive downlink windows immediately following an uplink transmission, during which downlink packets can be received. This introduces stringent timing constraints, highlighting the importance of gateway capabilities in sending these downlinks with precision.

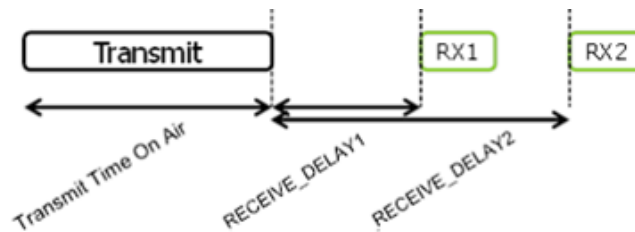


Figure 5.4. LoRa Class A TX Window [21]

In contrast, in the case of Class C, the node remains in a constant listening state, except for the brief intervals when it’s actively transmitting an uplink signal.

This distinction in listening behavior arises from the fundamental purpose behind the Class distinction. Class A is primarily designed for battery-powered devices with limited power availability, promoting power-efficiency by implementing specific time windows for communication. On the other hand, Class C devices are characterized by a continuous and uninterrupted power source, enabling them to maintain a persistent listening mode.

Specifically, the LoPy4 nodes were initially configured as Class C nodes due to the device profile settings within ChirpStack, which designated them as such.

A critical factor to consider is the combination of frequencies and data rates used for transmitting data from the node to the gateway. Initially, the code involved the configuration of channels for the three specified frequencies

in the EU868 band. However, this approach resulted in non-deterministic behavior on the part of the gateway, as it was uncertain which frequency to monitor for incoming signals.

```
self.lora.add_channel(0, frequency=868100000, dr_min=0, dr_max=5)
self.lora.add_channel(1, frequency=868300000, dr_min=0, dr_max=5)
self.lora.add_channel(2, frequency=868500000, dr_min=0, dr_max=5)
```

Hence, the optimal solution was configuring all channels to operate at a uniform frequency. Furthermore, it was considered judicious to clear all existing channels and subsequently re-add them, rather than overwriting the existing configurations.

```
for i in range(0, 16):
    self.lora.remove_channel(i)

    self.lora.add_channel(0, frequency=868100000, dr_min=0, dr_max=5)
    self.lora.add_channel(1, frequency=868100000, dr_min=0, dr_max=5)
    self.lora.add_channel(2, frequency=868100000, dr_min=0, dr_max=5)
```

In the original code, the for loop was positioned following the channel additions and solely targeted channels with an index equal to 3, which were not previously configured. After successfully introducing the desired channels with a frequency of 868.1 MHz (although the frequency choice can be customized based on the gateway’s settings), the final step involved configuring the LoRa socket’s data rate to align with the gateway’s reception settings.

```
self.sock.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
```

Regarding the data packets transmitted by the LoPy4 node, these are precisely defined in the configuration file as 34-byte packets. Within this structure, the initial two bytes serve as a unique device identifier, while the subsequent 32 bytes represent the values obtained from an imaginary sensor in the form of eight floating-point numbers.

For the application server to decode these packets effectively, a JavaScript decoding function was crafted. It’s worth noting that the JavaScript interpreter utilized adheres to the ES5 revision. This interpreter, implemented in Go language, has certain limitations, particularly in regard to features introduced in ES6, such as typed arrays. Therefore, the decoding functions

were meticulously developed using this interpreter, ensuring compatibility and functionality.

The primary decoding function for the packet transmitted by the LoPy4 node primarily focuses on translating the initial two bytes into an integer.

```
function uintToInt(uint, nbit) {
    nbit = +nbit || 32;
    if (nbit > 32) throw new RangeError;
    uint <<= 32 - nbit;
    uint >>= 32 - nbit;
    return uint;
}
function returnInt(bytes){
    var num = bytes[2] | bytes[3] << 8;
    return uintToInt(num, 16);
}
```

and a function that translates the 4 bytes into a float-type number with two decimal places, where start idx and end idx are the first byte and the last byte.

```
function bytesToFloat(bytes, start_idx, end_idx) {
    var bits = 0;
    var shift_idx = 3;
    for (var i=start_idx-1; i>=end_idx; i--){
        bits |= (bytes[i] << (8*shift_idx))
        shift_idx -= 1;
    }
    var sign = (bits>>>31 === 0) ? 1.0 : -1.0;
    var e = bits>>>23 & 0xff;
    var m = (e === 0) ? (bits & 0x7fffff)<<1;
    var f = sign * m * Math.pow(2, e - 150);
    return f.toFixed(2);
}
```

Another noteworthy observation regarding the LoPy4 node pertains to its activation method. When the device is activated using the Over-the-Air Activation (OTAA) approach, it exhibits the capability to successfully receive downlink messages. However, a striking contrast emerges when the device is activated through Activation by Personalization (ABP); under this method,

the node appears incapable of receiving any downlink communications. The root cause of this discrepancy, whether it emanates from the gateway, node configuration, or ChirpStack system, remains unclear and warrants further investigation within the context of this thesis.

The completed physical implementation of the End Node system is illustrated as follows.



Figure 5.5. Pycom LoRaWAN BME688 Sensor Node

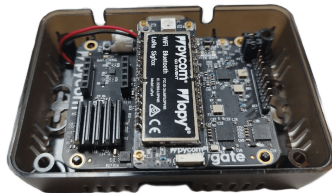


Figure 5.6. Pycom LoRaWAN BME688 Sensor Node

### 5.1.2 Raspberry Pi Pico BME688 Sensor Node

Pycom boards have a significant setback, arising from the lack of firmware and library support, largely attributed to Pycom’s bankruptcy. In light of this challenge, the quest for an alternative solution emerged, one that not





Figure 5.7. Pycom LoRaWAN BME688 Sensor Node



Figure 5.8. Pycom LoRaWAN BME688 Sensor Node

only matched the cost-effectiveness of Pycom boards but also encompassed a similar array of functionalities. The choice of this alternative hinged on factors such as cost-effectiveness, versatility, computational prowess, and affordability.

The Raspberry Pi Pico stood out as a compelling choice in this quest. However, transitioning to Raspberry Pi Pico introduced an initial fundamental shift in the programming language. While Pycom boards were equipped with firmware capable of interpreting MicroPython code, the Raspberry Pi Pico's flash memory lacked a native firmware. To tackle this, two primary options presented themselves:

- Use the SDK: This option involves setting up the Software Development Kit (SDK) for Raspberry Pi Pico. While programming in languages like C, C++, or assembly may be less straightforward, it provides extensive

access to the Pico’s capabilities and ensures compatibility with various onboard components. It also allows for the utilization of standard headers and libraries tailored for RP2040-based boards.

- **Install a MicroPython Interpreter:** Alternatively, the route of installing a MicroPython interpreter offers the simplicity of writing code in MicroPython, which is often more user-friendly and easier to comprehend. However, it comes with limitations, particularly in terms of library support and functionality for specific modules.

Opting for the first approach, setting up the SDK entailed an initial configuration phase, which subsequently provided access to standard headers and libraries, facilitating programming in languages like C and C++. This method, although potentially less intuitive, optimally harnessed the Raspberry Pi Pico’s capabilities and provided a wider array of libraries compatible with various integrated components.

Next, in pursuit of devices enabling LoRa-compatible radio connections within the EU868 frequency band, a meticulous evaluation of options was conducted. The RFM95 module, initially explored, supported LoRa frequencies but presented complexities in terms of installation and exhibited communication challenges, often rooted in configuration disparities when interfacing with Pycom boards.

Consequently, the preferred alternative emerged in the form of the Pico-LoRa-SX1262 868M module. Leveraging Semtech’s sx1262 chip, this module exhibited superior compatibility and performance, solidifying its position as the prime choice for enabling LoRa capabilities within the EU868 frequency band.

The official documentation underscores the Pico-LoRa-SX1262-868M module’s robust support for the LoRaWAN protocol, opening the gateway to seamless connectivity with versatile platforms like ChirpStack. Notably, the Waveshare website sheds further light on the module’s capabilities, encompassing:

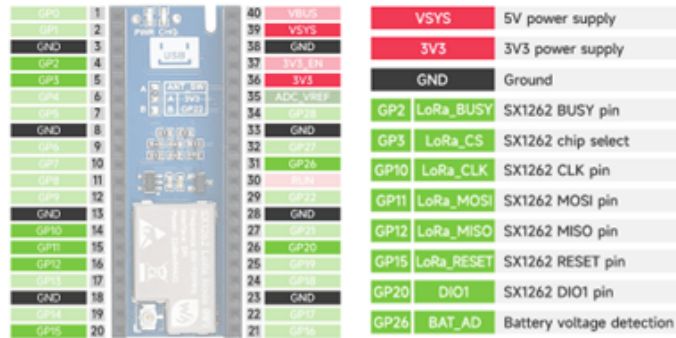
- **Compatibility with Raspberry Pi Pico Series:** The module seamlessly interfaces with Raspberry Pi Pico, equipped with standard headers to facilitate effortless integration.

- **Diverse Frequency Band Support:** Offering versatile support for the LoRaWAN protocol across multiple frequency bands, ensuring adaptability to varying operational requirements.
- **Modulation Flexibility:** The module boasts support for various modulation schemes, including FKS, GFSK, and LoRa modulation, expanding the horizons of communication possibilities.
- **Exceptional Receive Sensitivity:** With an impressive receive sensitivity of up to -148dBm, the module demonstrates an acute capability to capture even faint signals, bolstering reliable data reception.
- **Programmable Transmit Power:** Empowering users with the ability to finely tune transmit power up to 22dBm, granting control over the communication range and efficiency.
- **CRC Preamble Recognition:** Implementation of CRC preamble recognition enhances the module's data validation and integrity assurance.

Furthermore, the module comes equipped with a dedicated library that seamlessly implements the functions prescribed by the LoRaWAN protocol, streamlining the development process.

The integration of Raspberry Pi Pico with the Pico-LoRa-SX1262-868M module emerges as a straightforward endeavor, necessitating only the connection of Pico's pins to the module header. However, delving into the code demands attention to a couple of essential considerations. First, along to the configuration process for the LoPy4 board, it entails the meticulous setting of parameters for the connection via Activation by Personalization (ABP), aligning them with the predefined configurations in ChirpStack. Second, it requires diligent validation to ensure that the struct defining the pins utilized by the Pico-LoRa-SX1262-868M adheres to the correct pin assignments, as illustrated in Figure 5.9. These meticulous steps are vital to enable the Pico for seamless LoRaWAN communication, ensuring the module's optimal functionality in the broader network ecosystem.

Evidently, the pins exclusively allocated for the Pico-LoRa-SX1262-868M module must be safeguarded against any alternative usage, thereby precluding their availability for connecting additional devices, such as data-reading sensors integral to the application.



GP0	1	40	VBUS	VSYS	5V power supply
GP1	2	39	VSYS	3V3	3V3 power supply
GND	3	38	GND	GND	Ground
GP2	4	37	VSYS		
GP3	5	36	3V3		
GP4	6	35	ADC_VREF	GP2	LoRa_BUSY SX1262 BUSY pin
GP5	7	34	GP23	GP3	LoRa_CS SX1262 chip select
GND	8	33	GND	GP10	LoRa_CLK SX1262 CLK pin
GP6	9	32	GP27	GP11	LoRa_MOSI SX1262 MOSI pin
GP7	10	31	GP26	GP12	LoRa_MISO SX1262 MISO pin
GP8	11	30	GP25	GP15	LoRa_RESET SX1262 RESET pin
GP9	12	29	GP22	GP20	DIO1 SX1262 DIO1 pin
GND	13	28	GND	GP26	BAT_AD Battery voltage detection
GP10	14	27	GP21		
GP11	15	26	GP20		
GP12	16	25	GP19		
GP13	17	24	GP18		
GND	18	23	GND		
GP14	19	22	GP17		
GP15	20	21	GP16		

Figure 5.9. Pinout Pico-LoRa-SX1262-868M

A second imperative involves the utilization of a function designed to clear the Non-Volatile Memory (NVM) during the MCU’s startup phase. The NVM serves as the repository for crucial LoRa context data, including packet counters, among other essential information. The significance of this function lies in its capacity to prevent potential corruption of the NVM. This corruption risk materializes due to the inherent challenge of determining when the NVM is being written by the library. In the event of an untimely MCU shutdown during NVM writing, the NVM can be left in a corrupted state, impeding a seamless reboot. This function plays a pivotal role in mitigating such risks.

It is worth noting that the library does present certain constraints, particularly in the context of uplink transmissions. The library exclusively supports the transmission of unconfirmed uplinks, which do not necessitate acknowledgment receptions. This limitation, while inherent, is an important aspect to consider when devising communication strategies.

The ArmDeveloperEcosystem GitHub repository houses a repository of diverse illustrative examples to provide insights into the library’s functioning, serving as a valuable resource for users seeking a comprehensive understanding of its capabilities and applications.

Furthermore, an additional consideration revolves around the imperative to enforce a single frequency and a specific data rate when intending to leverage the Pycom nanogateway. To achieve this objective, specific parameters within the LoRaMac-node library require modification, specifically within the EU868 header file. These adjustments are instrumental in aligning the

node’s communication behavior with the intended configurations and operational requirements.

The completed physical implementation of the End Node system is illustrated as follows.



Figure 5.10. Raspberry Pi Pico BME688 Sensor Node



Figure 5.11. Raspberry Pi Pico BME688 Sensor Node

### 5.1.3 The Gateway

The initial aspect that necessitates consideration when examining the gateway established using the FiPy board pertains to its compliance with the LoRaWAN gateway specifications. In essence, it is more appropriate to refer to it as a LoRa forwarder rather than a gateway, given the parameters stipulated by LoRa and the requirements outlined by the LoRaWAN protocol for Europe. The designated frequencies for transmitting uplink packets in

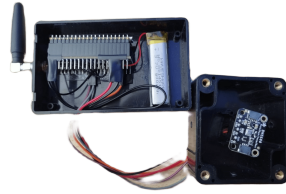


Figure 5.12. Raspberry Pi Pico BME688 Sensor Node

this context are 868.1, 868.3, and 868.5 MHz, while the broader frequency spectrum available for communication extends from 863 MHz to 870 MHz.

A commercial gateway possesses the capability to manage these frequencies and accommodate multiple channels operating within this range. As a result, this kind of gateway can concurrently handle multiple uplinks on different channels and effectively use this channel multiplicity for transmitting down-link packets while still receiving uplinks on unoccupied channels. In contrast, the FiPy board operates as a forwarder and is limited to monitoring a single frequency and channel. Consequently, it does not align with the LoRaWAN protocol, as it forfeits the capacity to utilize the three specified frequencies for communication, leading to an increased likelihood of conflicts, particularly in scenarios involving multiple nodes and a genuine gateway.

The code for the gateway is sourced from the example provided by Pycom. The implementation of the nanogateway, which is the code executed by the forwarder, exhibits certain inaccuracies and errors. The code's overarching structure involves:

- Establishing a UDP socket for communication with the network server.
- Creating a Lora socket for managing the reception and transmission of downlinks and uplinks.
- Defining a function linked to a timer for sending periodic keepalive signals from the gateway to the server.
- Designing a function tied to a timer for regularly requesting discovery packets from the server to facilitate the discovery of new gateways.
- Creating a separate thread to handle the various events specified in the gateway bridge.

- Defining a function for dispatching downlinks.

The gateway is responsible for managing various categories of packets, with the primary ones including:

- "stat" refers to the packet structure used for transmitting various gateway information to the server.
- "rx" defines the packet structure for data received by the gateway from connected nodes.
- "tx" specifies the packet structure for data originating from the network server and forwarded by the gateway.

The initial issue identified in the source code, subsequently rectified in the version supplied by Pycom, pertained to the handling of a specific type of downlink packet. According to the documentation from Semtech, accessible through The Things Network (Corporation, n.d.a), downstream packets include the fields depicted in Figure 5.13. As depicted in the table, it is explicitly stated that if the "imme" field is set to true, the gateway is expected to transmit the packet immediately. Furthermore, it is clarified that if "imme" is not true, it is essential to verify the presence of the "tmst" field, and if it is present, the packet should be transmitted following the defined protocol.

In the source code, a consistent issue was encountered with the utilization of the "tmst" value, which was being used to send packets without verifying its presence. This led to an exception when "imme" was set to true. Subsequently, the code was rectified to handle these two cases separately.

The code employs two distinct functions for sending in these cases. The functions differ in that, in the first scenario, the timer is given the timestamp for scheduling the downlink transmission (the timestamp is logged), whereas, in the second case, the downlink is sent immediately.

Another error, identified through an examination of the gateway bridge logs, pertained to the "TX ACK" packet type. This message serves as an acknowledgment sent from the gateway to the network server in response to a "PULL RESP" message. In Figure 5.14, it becomes evident that the "TX ACK" identifier assumes a value of 0x05 under the V2 protocol. However,

Name	Required	Type	Function
imme	No	Boolean	If true, the gateway is commanded to transmit the frame immediately
tmst	No	unsigned integer < 2 <sup>32</sup>	If "imme" is not true and "tmst" is present, the gateway is commanded to transmit the frame when its internal timestamp counter equals the value of "tmst". Section 6.2.2 contains a description of the gateway timestamp counter.
time	Y	string	UTC time. The precision is one microsecond. The format is ISO 8601 ( [3] ) 'compact' format. If "imme" is false or not present and "tmst" is not present, the gateway is commanded to transmit the frame at this time.
freq	N	unsigned float, Hz precision	The centre frequency on when the frame is to be transmitted in units of MHz.
rfch	Y	unsigned integer	The antenna on which the gateway is commanded to transmit the frame.
powe	N	signed integer	The output power which what the gateway is commanded to transmit the frame
modu	N	string	The modulation technique to be used: <ul style="list-style-type: none"> <li>• "LORA", representing LoRa modulation</li> <li>• "FSK", representing FSK modulation</li> </ul>
datr	N	string	Datarate identifier. When "modu" equals "LORA", "datr" comprises "SFnBWm", where 'n' is an integer representing the frame's 'spreading factor' and 'm' is an integer representing the frame's bandwidth in units of kHz. When "modu" equals "FSK" "datr" comprises an integer representing the frame's bit rate in units of Hz
codr	Yes, if "modu" equals "LoRa"	string	ECC code rate. "codr" comprises the string "k/n", where 'k' represents the carried bits and 'n' the total number of bits transmitted, including those added by the error checking/correction algorithm. Transmitted only when "modu" equals "LORA"
ipol	Y	bool	If true, commands gateway to invert the polarity of the transmitted bits. LoRa Server sets value to true when "modu" equals "LORA", otherwise the value is omitted.
size	N	unsigned integer	The number of octets in the received frame.
data	N	string	The frame payload, encoded into Base64, [4]. Base64 padding characters shall not be not added.
ncrc	N	bool	If not false, disable physical layer CRC generation by the transmitter.

Figure 5.13. Structure of a LoRa downstream package [13]

in the original code, the field was initialized with a value of 0x04, leading to an error where the gateway bridge failed to acknowledge the sent packet as requested. The solution was simply to introduce a new variable set to 0x05 to correctly define the identifier for the packet.

Subsequently, efforts were made to resolve the issue of not receiving packets from the LoRa node, despite correcting the errors recorded in the logs.



Offset (from start)	Number of octets	Function	Value or description
0	1	Protocol version	0x02
1	2	Token	If protocol version is 1, transmit as zero, ignore on receipt. If protocol version is 2, the
3	1	TX_ACK identifier	0x05
4		Payload	If no error is reported, the 'Payload' field comprises one octet of value '\0'. If an error is reported, the field contains a JSON "error" object.

Figure 5.14. Fields of LoRa TX ACK [13]

A review of the configuration file revealed that the nanogateway contained three distinct parameters configured for use in accordance with the EU868 standard. These parameters delineate the frequency and the datarate at which the nanogateway listens for incoming packets. As a result, it is imperative to configure the node to transmit data exclusively on that specified frequency, with the designated spreading factor and bandwidth, to ensure successful packet reception by the nanogateway.

Lastly, an ongoing challenge with the nanogateway relates to its inconsistent uptime, which is attributed to occasional crashes of undetermined origin. There is a likelihood that these crashes stem from unresolved issues within the board firmware, compounded by the FiPy's limitations in handling all incoming packets.

#### 5.1.4 LoRaWAN Network Server

Various solutions exist for implementing a LoRaWAN Network Server, with two prominent options being TTN and ChirpStack. ChirpStack offers a Docker Compose solution tailored for Linux operating systems, granting users the capability to operate their own LoRaWAN Network Server. On the other hand, TTN provides a fully managed service by The Things Network, simplifying end-user adoption. However, this ease of use comes at the cost of complete reliance on an external entity with its own policies and restrictions. For instance, TTN enforces adherence to the LoRaWAN protocol, imposing limitations similar to those of a public service aimed at accommodating numerous devices.

In light of these considerations, the decision was made to implement a self-owned LoRaWAN Network Server utilizing ChirpStack. This choice affords greater autonomy and alleviates the instability issues that were encountered with TTN at the time.

The backend's role involves receiving, filtering, and processing data from various nodes, and this function is carried out by the ChirpStack LoRaWAN Network Server, commonly referred to as ChirpStack. Specifically, ChirpStack V3 implemented in Docker is utilized, supporting the LoRaWAN protocol's three defined classes, incorporating the adaptive data rate algorithm, and offering additional features such as real-time frame logging. The ChirpStack stack comprises:

- The network server, responsible for implementing the LoRaWAN network stack.
- An application-specific server implementation.
- The gateway bridge.

Taking a top-down view of the ChirpStack architecture, two pivotal components stand out: the Application Server, which presents the user interface for configuring and monitoring devices within the application network, and the Network Server, primarily tasked with eliminating duplicate packets received from the LoRa gateway.

The ChirpStack application server offers a user interface through which all devices participating in the application network can be supervised and configured, facilitating their registration for the application. The main screen, Figure 5.15 provides an overview of the active devices and gateways, along with the primary data rates employed for transmissions.

The gateways screen serves as the interface for device registration and provides a summary page for users to assess traffic statistics associated with each gateway. This summary includes details on the volume and frequencies of received and transmitted packets, as well as the count of packets with "ok" status or those encountering collisions.

Conversely, the devices screen furnishes various summary data for each node, encompassing metrics like signal-to-noise ratio and received packet strength, Figure 5.16. It also facilitates in-depth analysis of these parameters.

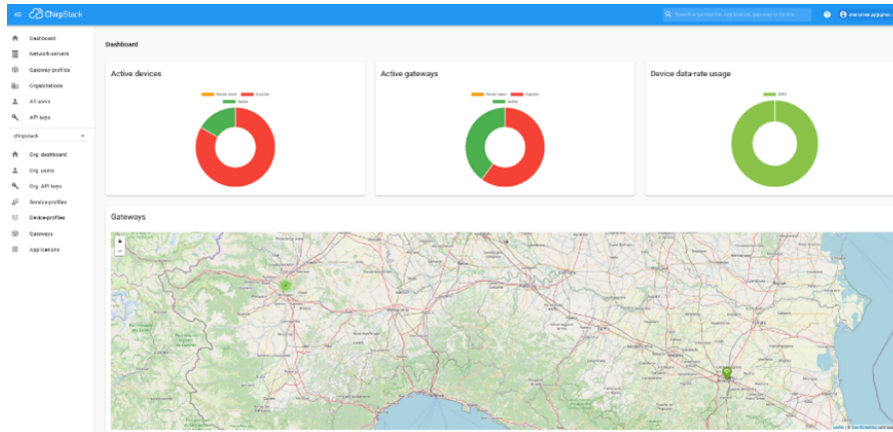


Figure 5.15. LoRa Application server interface

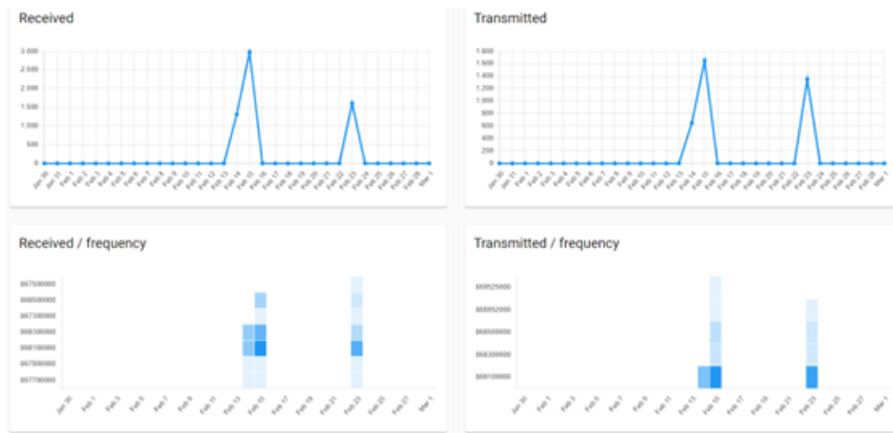


Figure 5.16. Gateway summary interface

Within the LoRaWAN frames window, users can scrutinize the type of LoRa frame sent or received by the device. Meanwhile, in the device data section, users can access not only general packet information but also the decoded payload sent by the node in the form of a JSON object.

This visibility into the JSON object containing the decoded payload is made possible by the capacity of the ChirpStack application server to define uplink decoding functions and downlink encoding functions. While the option to send packets via JSON encoding, which can then be translated into bytes (hexadecimal encoding) or transmitted directly as bytes, is noteworthy, the capability to decode uplinks directly into a JSON object at the backend

level holds distinct advantages, allowing for early data inspection.

### 5.1.5 Visualization Dashboard

Grafana, stands as a widely adopted open-source tool for querying, visualizing, and setting up alerts for time-series data. Its flexible data source model accommodates a range of time-series-based data sources, Grafana enables data querying through a query editor, data visualization through dashboards, and the implementation of alerts using its alerting functionality [6].

Grafana’s open-source nature and extensive compatibility with various data sources, including time-series data from sensors, ensure that we have the freedom to select the most suitable data storage solutions for our specific needs. This adaptability is particularly advantageous when working with diverse types of sensors and data formats.

By harnessing the data collected from the sensors and effectively processing it within the backend system, we unlock the potential to perform a comprehensive data extraction, transformation, and loading (ETL) process. This transformation process allows to convert raw sensor data into valuable insights and key performance indicators (KPIs).

For the project and the central objective of this thesis, the primary focus lies in the continuous monitoring of crucial environmental parameters within a test environment. These parameters include temperature, humidity, pressure, and CO2 levels. By utilizing queries and data processing tools, is possible to extract, transform, and load the sensor data, thereby creating informative and dynamic visualizations that provide valuable insights into the environmental conditions.

Through these visualizations, the project gain the ability to track, analyze, and respond to variations in temperature, humidity, pressure, and CO2 levels. This data-driven approach not only enhances our understanding of the test environment but also supports more informed decision-making and proactive actions to ensure optimal conditions and safety.

In essence, the use of data analytics and visualization, powered by the ETL process, is pivotal for achieving the objectives of this project and thesis, as it empowers to maintain a continuous and detailed watch over the critical environmental parameters that impact the test environment.

## 5.2 The System Implementation

A controlled microcosm is designed to emulate real-world conditions and address critical challenges, particularly in the realm of food security. Understanding the intricate dynamics of carbon dioxide (CO<sub>2</sub>) and volatile organic compounds (VOCs) in a controlled environment is crucial for advancing technologies that contribute to sustainable agricultural practices.

As global concerns about food security intensify, innovative solutions that enhance agricultural productivity, resource efficiency, and, especially for the purpose of this thesis, efficient monitoring in supply chains become paramount. By analyzing the system's performance in the microcosm, we aim to uncover insights that can inform strategies to optimize the growing environment. This research aligns with the broader mission of ensuring food security by refining technologies that directly impact the health and productivity of agricultural ecosystems. The controlled microcosm approach not only improves our understanding of system dynamics but also aligns with the development of solutions that have tangible implications for the future of sustainable and safe food production.

The configuration of the microcosm, distinguished by its unique geometry and integral to the experimentation, is an Airtight cylindrical flask with a volume of 1.062 liters, equipped with a gasketed lid, Figure 5.17

The preparation of the microcosm, involved the introduction of 20 ml of sterile water at the base to maintain humidity at 40 percent. Additionally, a monolayer of hazelnuts, with a dry weight of 37 grams (8 percent moisture content) and selected through random sampling, was incorporated. A support system featuring a plastic tripod connected to a net was introduced to accommodate the hazelnut monolayer.

Prior to microcosm setup, all materials mentioned underwent sterilization (20 minutes at 121°C, 2 atm). Furthermore, as part of the experimental controls, an abiotic microcosm was established for each test. In these control microcosms, hazelnuts underwent the same sterilization process to ensure a comparative baseline.

To assess the efficacy of the system, a comprehensive evaluation was undertaken through the implementation of two distinct tests. In the initial test phase, meticulous configuration of the acquisition system was carried out to



Figure 5.17. Configuration of the microcosm

capture precise carbon dioxide (CO<sub>2</sub>) values, measured in parts per million (ppm). Subsequently, the second test was designed to extend the monitoring capabilities by encompassing not only carbon dioxide but also volatile organic compounds (VOCs) emanating from the biomass present within the microcosms.

Throughout the experimental proceedings, the BME688 sensor, was strategically positioned within the biotic microcosms. This critical placement of the sensor within the experimental setup, illustrated in Figures 5.18, 5.19 allowed for a nuanced exploration of the dynamic interactions and fluctuations in both CO<sub>2</sub> levels and VOCs, offering a comprehensive insight into the system's performance under varying conditions.

The CO<sub>2</sub> levels (ppm) recorded by the sensor system before opening the microcosm were examined, emphasizing that the initial value of 500 ppm corresponds to the measurement taken just 5 minutes after closing the microcosm. This value indicates the concentration of CO<sub>2</sub> in the environment outside the microcosm. The sensor, when detecting changes in carbon dioxide, does not provide an accurate value without a reference with respect to the previous value. In other words, if the current value remains constant compared to the immediately preceding one, this data is not sent to the

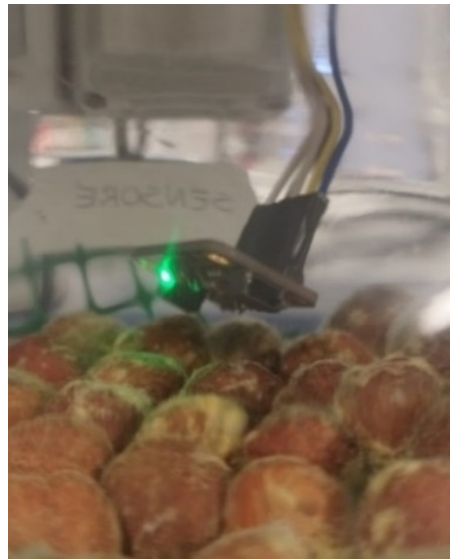


Figure 5.18. Sensor BME688 Operational Inside



Figure 5.19. Microcosm Equipped with BME688 Sensor

server, optimizing energy consumption and utilization of the transmission data channel.

The screenshot illustrating the CO<sub>2</sub> values of the system at the first measurement on 05/15/23 at 12:00 noon is detailed in Figure 5.20.

After opening the microcosm that housed the sensor, a decrease in the CO<sub>2</sub> value was observed until the initial condition of approximately 500 ppm was reached, followed by an increase after closure. In the first 24 hours, the recorded CO<sub>2</sub> value doubled from the initial value of 500 ppm, and by 48 hours, this value rose to approximately five times the initial value.

However, on 05/18/23, starting at 16:00 hours, due to a failure in the general power supply of the laboratory, no value was recorded until 12:00 hours the following day when manual measurements were taken. Table 6.4 details the point values recorded by the sensor every 24 hours before opening the microcosm, providing a more comprehensive view of the evolution of CO<sub>2</sub> levels in the system.

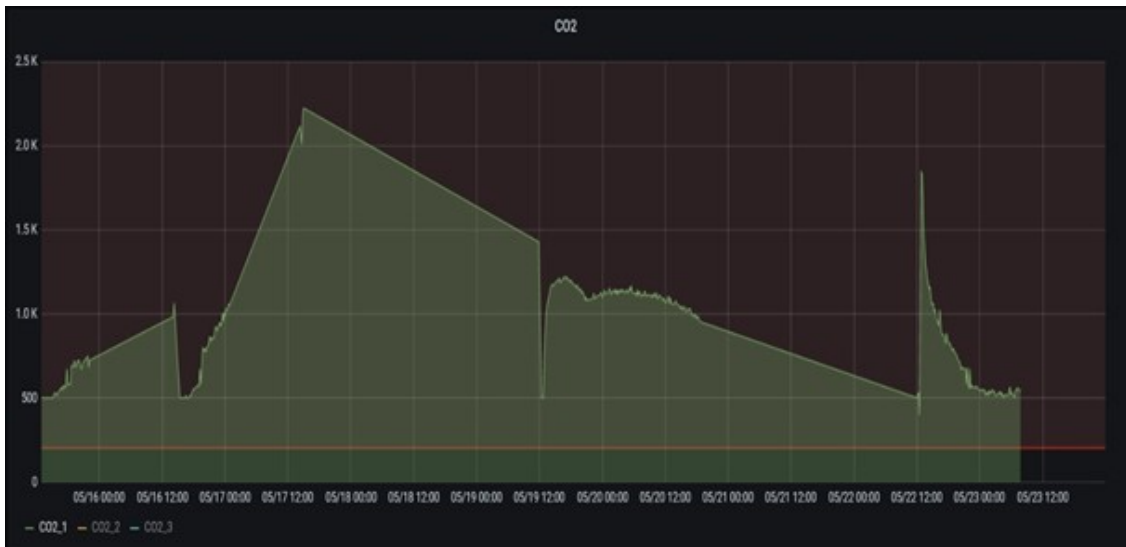


Figure 5.20. Screenshot depicting CO<sub>2</sub> levels (ppm) recorded by the BME688 sensor from 15/05/23 to 23/05/23.

After the weekend, during which the microcosm remained unopened, a value comparable to the initial measurement (531 ppm) was obtained on 22/05/23. The following day, the lowest recorded CO<sub>2</sub> value of 400 ppm was observed. Subsequently, the sensor was repositioned approximately 1 cm from the hazelnut monolayer, in contrast to its previous placement. Five minutes after closure, the CO<sub>2</sub> levels exhibited a sharp increase, reaching 12,500 ppm (Figure 6.6), and then gradually decreased to 4,130 ppm (the value recorded prior to opening the microcosm 24 hours later).



In the subsequent days, the recorded CO<sub>2</sub> values maintained an average of 1,600 ppm, with the exception of the final day of incubation.

Table 5.1. Microcosm - Point Values

<b>Data</b>	<b>CO<sub>2</sub> [ppm]</b>
15-05-2023	500
16-05-2023	1063
17-05-2023	2223
18-05-2023	\
19-05-2023	1427
22-05-2023	531
23-05-2023	400
24-05-2023	4130
25-05-2023	1830
26-05-2023	1750
29-05-2023	1360
30-05-2023	512

Given the significance associated with the study of fungal Volatile Organic Compounds (VOCs), the system’s capability, equipped with the BME688 sensor, was harnessed to directly detect VOCs. This capability was leveraged to conduct a second test utilizing the same microcosm configuration. In this subsequent test, the BME688 sensor was specifically configured to provide readings for both CO<sub>2</sub> levels and the volatile organic compounds (VOCs) generated within the microcosm.

The initial b-VOC value of 0.49 ppm corresponds to the measurement taken just 5 minutes after the closure of the microcosm. This value is derived from the contrast between the b-VOCs present in the environment outside the microcosm and those within. The subsequent values pertain to both CO<sub>2</sub> and VOCs, representing the conditions at the time of the final sensor measurement before the microcosm was opened.

This approach not only expands the scope of the study by incorporating direct VOC detection but also enables a nuanced exploration of the dynamic interplay between CO<sub>2</sub> and VOCs within the controlled microcosm environment. The recorded values offer insights into the intricate relationships and fluctuations in both CO<sub>2</sub> and VOC concentrations, providing a comprehensive understanding of the system’s performance under varied conditions.

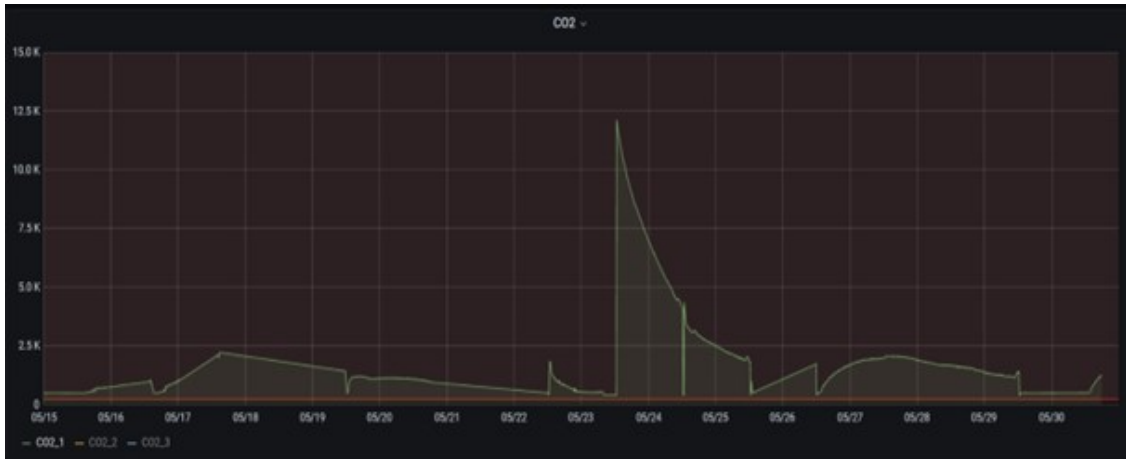


Figure 5.21. CO2 Concentration Display (15/05/23 - 30/05/23)

In the system architecture implementation, Node-RED assumes a central role as the backend infrastructure, orchestrating the seamless flow of data from LoRaWAN-enabled sensors transmitted via The Things Network (TTN) or ChirpStack to a Grafana dashboard for real-time and historical data visualization.

The Node-RED setup involves its installation on a dedicated server within the domain of the Reply company in Torino. Notably, to optimize computing resource consumption, Node-RED is encapsulated within a Docker container. This containerization strategy not only enhances resource efficiency but also streamlines deployment and maintenance processes.

Configuration of Node-RED follows, wherein it is tailored to function as an intermediary layer connecting the LoRaWAN network and the Grafana dashboard. This encompasses the establishment of nodes and flows to efficiently manage the data flow.

For connectivity with The Things Network, Node-RED is configured to establish a secure connection to the TTN broker, incorporating necessary credentials and security measures to ensure the integrity of transmitted data. Similarly, for ChirpStack integration, Node-RED is configured to establish a connection to the ChirpStack broker, implementing appropriate security measures for data confidentiality.

While the application (Node-RED) resides within a Docker container,

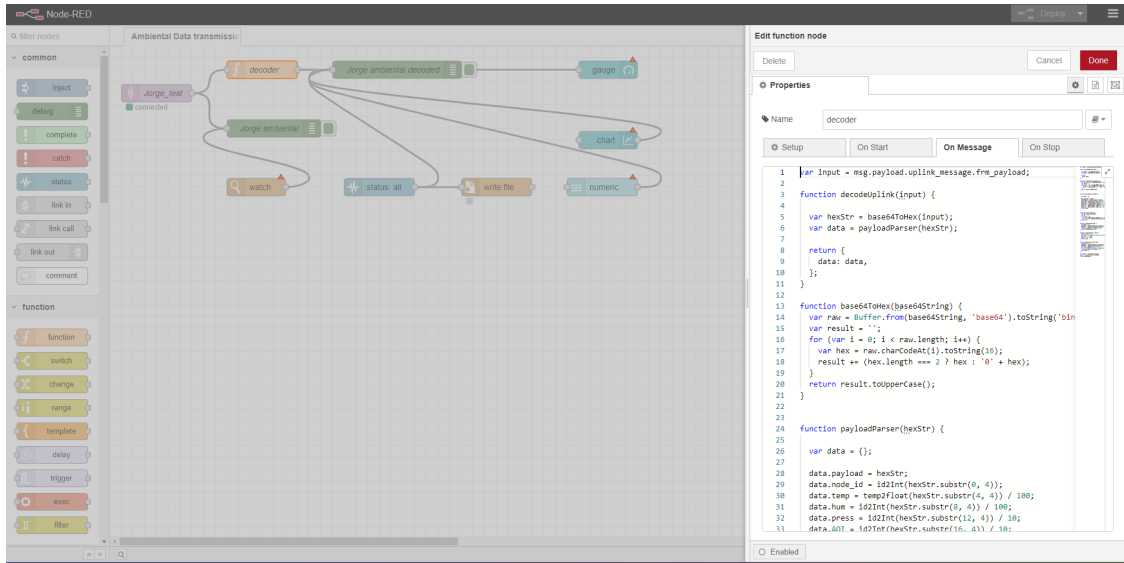


Figure 5.22. Node-RED Backend Architecture

it encountered challenges during the data collection process. The server, at times, became unreachable, leading to data loss. This issue primarily stemmed from complications with firewall policies that intermittently closed the port crucial for data transmission.

Upon successful connection, Node-RED is set up to receive data packets from LoRaWAN-enabled sensors. To extract meaningful information from the raw payload, a custom decoder, specifically developed for the sensor data format, is applied.

The processed data is then directed to the Grafana dashboard for visualization. In the real-time domain, Node-RED facilitates the dynamic presentation of sensor data on Grafana using its versatile visualization options, creating informative graphs and charts. Simultaneously, Node-RED is programmed to store historical data persistently. This historical data is visualized on the Grafana dashboard and concurrently stored in a file on the disk, ensuring a comprehensive record for future analysis and reference.

In the controlled laboratory environment of the microcosmos, data collection from sensors was meticulously orchestrated to capture a diverse range of parameters. These sensors, strategically placed to monitor specific variables, generated a continuous stream of data representative of the microcosmic conditions.

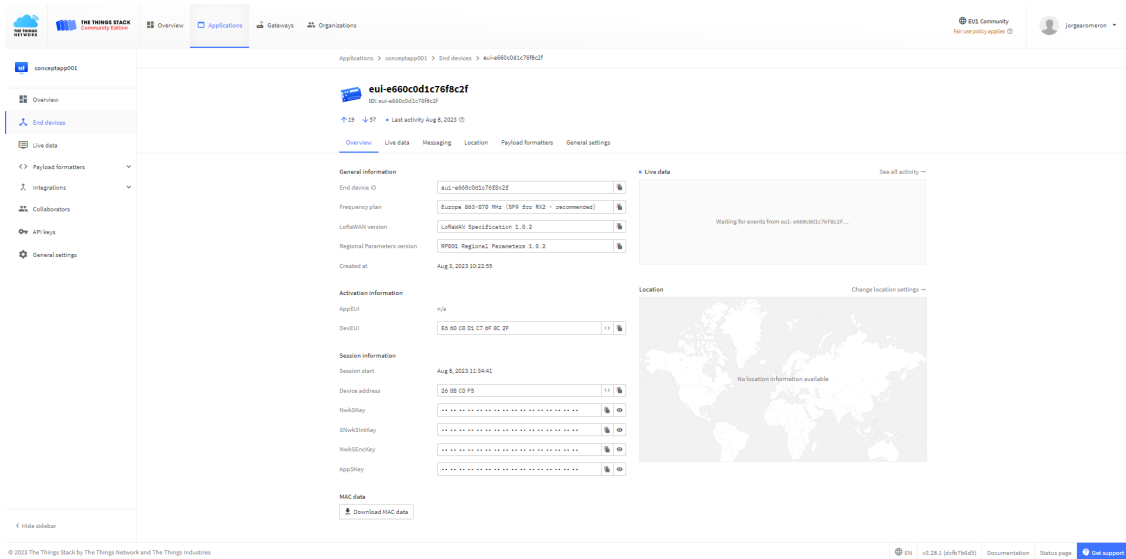


Figure 5.23. The Things Network (TTN) Broker Connection

The seamless transition of this data from the microcosmos to the LoRaWAN gateway marked a critical phase in the process. Leveraging LoRaWAN technology, the collected data was transmitted wirelessly to the gateway, providing a bridge between the localized sensor network and the broader network infrastructure.

Upon reaching the gateway, the data embarked on its journey through the LoRaWAN network to The Things Network (TTN) server broker. This intermediary step ensured the secure and reliable transfer of sensor data to a centralized platform for subsequent processing and visualization.

The Node-RED backend, strategically deployed for its versatility, became the nexus for data handling. Within the Node-RED environment, the received data underwent a decoding process tailored to the specific sensor data format. This crucial step transformed raw payloads into meaningful information, laying the groundwork for insightful analysis.

Subsequently, the processed data seamlessly interfaced with the Grafana dashboard, offering real-time visualizations of the microcosmic conditions. The dynamic and customizable nature of Grafana's visualizations provided a user-friendly interface for monitoring key parameters.

Notably, Node-RED’s capability to store historical data in parallel with real-time visualization ensured a comprehensive record of the microcosmic conditions. The historical data, written to disk, became a valuable resource for retrospective analysis and reference.

The data collection journey from the microcosmos to the Grafana dashboard was characterized by a meticulous process: from sensor-rich laboratory facilities to LoRaWAN transmission, TTN server brokerage, Node-RED backend processing, and culminating in dynamic real-time visualizations and persistent historical data storage. This integrated system underscored the importance of each phase in facilitating a holistic understanding of the microcosmic environment.



## Chapter 6

# Conclusions and Future Developments

In conclusion, this thesis underscores the vital role of food in sustaining life and addresses the extensive challenges associated with its safety throughout the globalized supply chain. Recognizing the profound impact of food-related health problems, the project advocates for a proactive and comprehensive approach to food safety, especially in the context of increasingly complex supply chains. The technologies developed in this thesis offer not only immediate solutions but also a path towards significant environmental benefits and reduced health risks. Improved food safety control and monitoring contribute to mitigating production losses, reducing waste, lowering energy consumption, and ensuring high-quality consumer products. A pivotal aspect of this project challenges the conventional notion of traceability, proposing a shift towards dynamic, real-time interventions to fortify the food supply chain against potential risks. This departure from traditional methods aims to bring about more timely and targeted responses. At the core of this endeavor is the characterization of innovative, low-cost data acquisition tools. Implemented along supply chains with block-based distributed control systems, these tools advance risk-based decision-making support systems. Integrating sensors with wireless communication systems enhances the agility of data collection, analysis, and sharing, ensuring continuous availability across vast physical extensions.

In conjunction with these advancements, significant strides were made through collaborative efforts with microbiologists in laboratory facilities. The

synergy of microbiological expertise and technological innovation was exemplified in the thesis's objective to measure parameters values within microcosms, utilizing hazelnuts as a medium. This interdisciplinary approach not only enriches the understanding of food safety but also opens new avenues for research and application, fostering a safer and more sustainable global food supply chain. The thesis's culmination not only marks a milestone in technology-driven food safety but also exemplifies the potential when scientific disciplines converge for a common goal.

Furthermore, this thesis emphasizes the importance of looking for strong partnerships between industry and academia, where the fusion of cutting-edge technological innovation and the depth of scientific knowledge propels advancements with far-reaching implications. The ongoing and even heightened need for active monitoring of food safety, especially in the aftermath of the COVID-19 pandemic, underscores the critical role that collaborative research plays in ensuring the well-being of populations worldwide. The dynamic nature of foodborne risks, coupled with the evolving landscape of global trade, demands continuous vigilance and innovation in monitoring mechanisms. Through this collaborative effort, this Thesis not only bridges the gap between theoretical insights from academia and practical applications in industry but also creates a synergistic force capable of tackling the multifaceted challenges that characterize the modern food supply chain.

Moreover, this collaborative endeavor serves as a model to the adaptability and resilience of research ecosystems. The partnership between industry and academia not only accelerates the translation of theoretical knowledge into actionable solutions but also fosters a continuous feedback loop, enabling rapid adjustments to emerging challenges. As we navigate the complexities of the post-pandemic landscape, the importance of such collaborative initiatives becomes even more pronounced, offering a blueprint for addressing not only existing issues but also anticipating and preemptively addressing future threats to food safety and public health.



# Bibliography

- [1] M.R. Abdmeziem and D. Tandjaoui. “Tailoring mikey-ticket to e-health applications in the context of internet of things”. In: *International Conference on Advanced Networking, Distributed Systems and Applications (Short Papers)*. June 2014, pp. 72–77.
- [2] Sarder Fakhrul Abedin et al. “A system model for energy efficient green-IoT network”. In: (2015), pp. 177–182. DOI: [10.1109/ICOIN.2015.7057878](https://doi.org/10.1109/ICOIN.2015.7057878).
- [3] Ian F Akyildiz et al. “A survey on sensor networks”. In: *IEEE Communications magazine* 40.8 (2002), pp. 102–114. DOI: [10.1109/MCOM.2002.1024422](https://doi.org/10.1109/MCOM.2002.1024422). URL: [\[1\]](#).
- [4] Arun Aryal et al. “The emerging big data analytics and IoT in supply chain management: a systematic review”. In: *Supply chain management*. 25.2 (2018-12-24). ISSN: 1359-8546.
- [5] Valentina Bianchi et al. “IoT and Biosensors: A Smart Portable Potentiostat With Advanced Cloud-Enabled Features”. In: *IEEE Access* (2021). Received September 6, 2021; accepted October 11, 2021; date of publication October 14, 2021; date of current version October 22, 2021. Corresponding author: Andrea Boni (andrea.boni@unipr.it). This work was supported by the Project “Biosensoristica innovativa per i test sierologici e molecolari e nuovi dispositivi PoCT per la diagnosi di infezione da SARS-CoV-2” funded in 2020, by “Bando Straordinario di Ateneo per Progetti di Ricerca Biomedica in Ambito SARS-COV-2 e COVID-19,” University of Parma. DOI: [10.1109/ACCESS.2021.3120022](https://doi.org/10.1109/ACCESS.2021.3120022).
- [6] Mainak Chakraborty and Ajit Pratap Kundan. “Grafana”. In: *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*. Berkeley, CA: Apress, 2021, pp. 187–240. ISBN: 978-1-4842-6888-9. DOI: [10.1007/978-1-4842-6888-9\\_6](https://doi.org/10.1007/978-1-4842-6888-9_6). URL: [https://doi.org/10.1007/978-1-4842-6888-9\\_6](https://doi.org/10.1007/978-1-4842-6888-9_6).

- [7] Bharat S Chaudhari and Marco Zennaro. *LPWAN Technologies for IoT and M2M Applications*. Academic Press, 2020.
- [8] A.P. Das, P.S. Kumar, and S. Swain. “Recent advances in biosensor based endotoxin detection”. In: *Biosensors and Bioelectronics* 51 (2014), pp. 62–75. ISSN: 0956-5663. DOI: <https://doi.org/10.1016/j.bios.2013.07.020>. URL: <https://www.sciencedirect.com/science/article/pii/S0956566313004922>.
- [9] Shilpa Devalal and A. Karthikeyan. “LoRa Technology - An Overview”. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2018, pp. 284–290. DOI: [10.1109/ICECA.2018.8474715](https://doi.org/10.1109/ICECA.2018.8474715).
- [10] Gartner, Inc. *Gartner Says 8.4 Billion Connected “Things” Will Be in Use in 2017, Up 31 Percent From 2016*. <https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016>. [Accessed on August 21, 2023]. 2017.
- [11] Guojun Ji, Limei Hu, and Kim Hua Tan. “A study on decision-making of food supply chain based on big data”. In: *Journal of Systems Science and Systems Engineering* 26.2 (Apr. 2017), pp. 183–198. DOI: [10.1007/s11518-016-5320-6](https://doi.org/10.1007/s11518-016-5320-6). URL: <https://doi.org/10.1007/s11518-016-5320-6>.
- [12] Zaid Jameel Radhi Kamoona and Muhammad Ilyas. “Investigating the Performance of LoRa Communication for Nominal LoRa and Interleaved Chirp Spreading LoRa”. In: *2022 International Conference on Artificial Intelligence of Things (ICAIoT)*. 2022, pp. 1–7. DOI: [10.1109/ICAIoT57170.2022.10121818](https://doi.org/10.1109/ICAIoT57170.2022.10121818).
- [13] “LoRaWAN™ Regional Parameters v1.1rA”. In: *LoRaWAN™ 1.1 Specification*. (cit. on pp. 4, 13). 2017.
- [14] Davide Magrin, Marco Centenaro, and Lorenzo Vangelista. “Performance evaluation of LoRa networks in a smart city scenario”. In: *2017 IEEE International Conference on Communications (ICC)*. 2017, pp. 1–7. DOI: [10.1109/ICC.2017.7996384](https://doi.org/10.1109/ICC.2017.7996384).
- [15] Marin B. Marinov, Borislav T. Ganev, and Dimitar N. Nikolov. “Indoor Air Quality Assessment Using Low-cost Commercial Off-the-Shelf Sensors”. In: *2021 6th International Symposium on Environment-Friendly Energies and Applications (EFEA)*. 2021, pp. 1–4. DOI: [10.1109/EFEA49713.2021.9406260](https://doi.org/10.1109/EFEA49713.2021.9406260).

- [16] MicroPython Project. *MicroPython*. Accessed on 18102023. 2023. URL: <https://micropython.org/>.
- [17] Christos Milarokostas et al. “A Comprehensive Study on LPWANs With a Focus on the Potential of LoRa/LoRaWAN Systems”. In: *IEEE Communications Surveys Tutorials* 25.1 (2023), pp. 825–867. DOI: [10.1109/COMST.2022.3229846](https://doi.org/10.1109/COMST.2022.3229846).
- [18] Juha Petajarvi et al. “On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology”. In: *2015 14th International Conference on ITS Telecommunications (ITST)*. IEEE, 2015, pp. 55–59. DOI: [10.1109/ITST.2015.7377400](https://doi.org/10.1109/ITST.2015.7377400). URL: [1].
- [19] Albert Pötsch and Florian Haslhofer. “Practical limitations for deployment of LoRa gateways”. In: *2017 IEEE International Workshop on Measurement and Networking (MN)*. 2017, pp. 1–6. DOI: [10.1109/IWMN.2017.8078360](https://doi.org/10.1109/IWMN.2017.8078360).
- [20] Mohsen Pourkiaei and Anne-Claude Romain. “Scoping review of indoor air quality indexes: Characterization and applications”. In: *Journal of Building Engineering* 75 (2023), p. 106703. ISSN: 2352-7102. DOI: <https://doi.org/10.1016/j.jobbe.2023.106703>. URL: <https://www.sciencedirect.com/science/article/pii/S2352710223008823>.
- [21] Pycom. *Pycom Products Documentation*. Accessed on 18102023. 2023. URL: <https://docs.pycom.io/products/>.
- [22] Abderahman Rejeb, John G. Keogh, and Karim Rejeb. “Big data in the food supply chain: a literature review”. In: *Journal of Data, Information and Management* 4 (2022). Received: 17 November 2020 / Accepted: 29 December 2021, pp. 33–47. DOI: [10.1007/s42488-021-00064-0](https://doi.org/10.1007/s42488-021-00064-0). URL: <https://doi.org/10.1007/s42488-021-00064-0>.
- [23] Jeffrey D. Sachs et al. *Implementing the SDG Stimulus. Sustainable Development Report 2023*. Paris, Dublin: SDSN, Dublin University Press, 2023. DOI: [10.25546/102924](https://doi.org/10.25546/102924).
- [24] Curt Schurgers and M.B. Srivastava. “Energy efficient routing in wireless sensor networks”. In: *IEEE Military Communications Conference MILCOM. Communications for Network-Centric Operations: Creating the Information Force*. Vol. 1. 2001, pp. 357–361.
- [25] Deepti Sehrawat and Nasib Singh Gill. “Smart Sensors: Analysis of Different Types of IoT Sensors”. In: (2019), pp. 523–528. DOI: [10.1109/ICOEI.2019.8862778](https://doi.org/10.1109/ICOEI.2019.8862778).

- [26] Mohamed A. Shenashen et al. “Progress in sensory devices of pesticides, pathogens, coronavirus, and chemical additives and hazards in food assessment: Food safety concerns”. In: *Progress in Materials Science* 124 (2022), p. 100866. ISSN: 0079-6425. DOI: <https://doi.org/10.1016/j.pmatsci.2021.100866>. URL: <https://www.sciencedirect.com/science/article/pii/S0079642521000906>.
- [27] Giovanni Stanco et al. “On the performance of IoT LPWAN technologies: the case of Sigfox, LoRaWAN and NB-IoT”. In: *ICC 2022 - IEEE International Conference on Communications*. 2022, pp. 2096–2101. DOI: [10.1109/ICC45855.2022.9839078](https://doi.org/10.1109/ICC45855.2022.9839078).
- [28] The Things Network. *LoRaWAN PHY Data Rate and Frame Format*. Accessed on 10082023. 2023. URL: <https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/>.
- [29] *Transforming Our World: The 2030 Agenda for Sustainable Development*. 2015.
- [30] Vijayalakshmi Velusamy et al. “An overview of foodborne pathogen detection: In the perspective of biosensors”. In: *Biotechnology Advances* 28.2 (2010), pp. 232–254. ISSN: 0734-9750. DOI: <https://doi.org/10.1016/j.biotechadv.2009.12.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0734975009002134>.