

University of Padova

DEPARTMENT OF INFORMATION ENGINEERING

Master's degree in Automation Engineering

**Automation system of a powder-based 3D printer
in the Industry 4.0 environment**

Author

Luca Trevisan

Supervisor

Stefano Vitturi

Company's supervisor

Andrea Beretta

02 DECEMBER 2019
ACADEMIC YEAR 2018/2019

Contents

1	Introduction	9
2	3D printer system	13
2.1	The demonstrator system	13
2.2	Automation system	14
3	LoRa and LoRaWAN	17
3.1	The protocol	17
3.1.1	Chirp Spread Spectrum modulation	18
3.1.2	LoRa frame	20
3.2	The architecture	21
3.3	The communication	22
4	Testing LoRa	25
4.1	Connection range test	25
4.1.1	Hardware setup	25
4.1.2	The experiments	25
4.2	Server connection tests	32
4.2.1	Setting the hardware	32
4.2.2	Retrieving data	36
4.3	Data Storage	36
4.3.1	The algorithm	37
4.4	Data handling	38
4.4.1	The PLC	39
4.4.2	Developed software	40
5	Related Works	55
5.1	The security vulnerabilities of LoRa	55
5.2	Other tests on the communication range	56
5.3	Enhanced LoRaWAN node	58

6 Conclusions and future directions

59

List of Figures

2.1	Main components of the 3D printer	14
2.2	Scheme of the demonstrator system automation	15
3.1	Chirps examples	18
3.2	Example of a LoRa message preamble	19
3.3	Easy example of LoRa modulation	19
3.4	Modulated LoRa message example	20
3.5	LoRa frame structure	20
3.6	Graph showing the architecture of a LoRa connection	22
3.7	Graph showing the connection deployment of LoRa	22
4.1	LoRa Technology Evaluation Kit - 800	26
4.2	Radio tab of the LoRa mote in LoRa dev utility by Microchip	26
4.3	Experimental setup	27
4.4	Graph showing the loss of packets with respect to the antennas	28
4.5	Graphs showing the correctly received packets with different spreading factors in the morning	30
4.6	Graphs showing the correctly received packets with different spreading factors in the afternoon	32
4.7	Example of the gateway tab of LoRa Development Utilities	33
4.8	Example of "Device Overview" on the TTN dashboard	35
4.9	Example of "Data" tab of a LoRa mote in the TTN dashboard	35
4.10	Example of data stored in the "statwolf" database	37
4.11	Example of behaviour of the algorithm	38
4.12	Omron NX102-9020	39
4.13	Electric scheme of the connection between PLC and power supply	40
4.14	Racks configured on the Sysmac Studio Environment	41
4.15	Ethernet ports settings on the Sysmac Studio Environment	41
4.16	Connection test on the Sysmac Studio Environment	42
4.17	Scheme of the connection between TTN and PLC	42

4.18 SktTCPAccept Sysmac studio function	43
4.19 SktTCPRecv Sysmac studio function	44
4.20 SktClose Sysmac studio function	44
4.21 SubDelimiter Sysmac studio function	45
4.22 FileOpen Sysmac studio function	46
4.23 FileWrite Sysmac studio function	46
4.24 Database configuration tab	48
4.25 DB_Connect Sysmac studio function	48
4.26 DB_CreateMapping Sysmac studio function	49
4.27 DB_Select Sysmac studio function	50
4.28 DB_Close Sysmac studio function	50
4.29 DB_Connect Sysmac studio function	51
4.30 DB_CreateMapping Sysmac studio function	51
4.31 DB_Insert Sysmac studio function	52
4.32 Example of a packet sent by the PLC to the Statwolf database	52
4.33 DB_Close Sysmac studio function	53
5.1 Graph showing the signal power with respect to the distance	57

List of Tables

4.1	Transmission statistics for the two antennas	28
4.2	Spreading factor comparison in the morning	29
4.3	Spreading factor comparison in the afternoon	31
4.4	StatwolfColumns structure	45
4.5	StatwolfColumns structure	49

Chapter 1

Introduction

The technological progress led to an environment in which the concept of connectivity is crucial. It is possible to send information to every part of the world by simply pushing a button on the phone, the only requirement is an internet connection. Many protocols and infrastructures have been developed to make the transmission as fast and as robust as possible and, nowadays, data can be exchanged at a rate of Gigabit per second. The last research activities are focused on the massive interconnection between all the devices that can transmit information; this is called Internet of Things (IoT). IoT was born roughly at the end of the last century as a definition of an interconnected system of uniquely identifiable objects equipped with radio frequency identification (RFID) technology. Since the ways in which the devices can be connected to each others are growing continuously, in these days the "IoT" indicates a wide range of technologies and architectures. Typical examples are Smart Cities, in which each vehicle can send information about its position or its energy so to avoid traffic congestion or energy overloads, or smart homes, in which the people habits are studied to ensure the maximum level of comfort, and many many others. This process has not yet had the same success in the field of the industries. In the majority of the cases, the industrial plants built decades ago are still working and innovate them would mean to spend money for something that seems to be good enough. Only in the last few years the IoT has been introduced in this scenario bringing with it a new term, "IIoT", that stands for "Industrial Internet of Things", or Industry 4.0. The aim of the IIoT is to build production processes in which all the involved devices are able to collaborate to maximize the efficiency. As an example, IIoT systems can be used to predict plant failures or to actuate security systems that prevent workplace accidents, facts that would affect both the production and the finances. The easiest way to implement such kind of systems is a massive use of sensors, to analyze all the variables involved in the processes and

transforming them in information that can be further analysed.

Industry 4.0 systems that rely on wireless communication, are characterized by reduced overheads and small amount of data per node. Moreover, in many cases, the distances that the transmissions have to cover are not too high. In any case, however, the reliability/availability of the data and the efficiency on the energy consumption are aspects of paramount importance.

This work is centered on the use of a Low Power Wide Area Network (LPWAN) called LoRa, discussed in the next chapter. LPWANs combine a very long communication range with extremely long life battery, at the cost of a limited throughput, satisfying most of the requirements introduced before. Popular LPWANs are NB-IoT, SigFox, Ingenu Weightless. LoRaWAN, actually, is one of the most widespread LPWANs and has being studied in many IoT scenarios in the last years. Its efficiency and robustness make it an effective candidate for the IIoT applications that this framework will investigate.

This work is part of the ADMIN4D (ADditive Manufacturing & INdustry 4.0 as innovation Driver) regional project that involves, among the others, the National Research Council of Italy (CNR), the Universities of Padua and Venice and *Desamanera*. *Desamanera* develops 3D printers of huge dimensions in order to create objects using innovative materials and forefront technologies. The aim of the project is the development of an innovative system that allows to collect and elaborate data from new materials and binders used in the 3D printing process. To assure the robustness of the result, sensors are located inside the artefacts. Monitoring, for example, the light and the temperature, the printing process can be adjusted to reach the desired specifications. The data have to be transmitted from the printer to a server passing through an industrial environment in which the packets loss has to be negligible.

The aim of this thesis is to investigate the communication systems used by ADMIN4D. So, first the LoRa and LoRaWAN protocol will be addressed to understand their potentialities and implementations. LoRa devices play different roles with respect to their data rate: with a low data rate they can be used to perform slow tasks, as temperature or humidity monitoring, or sensors data collection; with an higher one, for example, they can be used to perform more demanding tasks to avoid failures.

The experiment phase will start adopting the hardware and software apparatus that will be used on the *Desamanera* 3D printers in the future. Since this is the first

approach to these kind of technologies, the aim will be understand limits and difficulties that can be encountered so to prevent them.

One of the main characteristic of the IoT, is the easy accessibility of the stored data. For this reason, databases and interfaces that can link LoRa to the storage apparatus will be examined. In particular, a MySQL database, named “Statwolf”, is used as storage unit. Ad hoc algorithms will be developed to exchange data between the LoRa devices and the database. In this way, not only the data transfer between LoRa devices will be tested, but also that with an online environment.

The NX102-9020 CPU, developed by Omron, will be assembled and prepared for the first prototypes of the 3D printer. This PLC has the ability to interface itself with cloud databases and to retrieve or send data to them. Thanks to that, it can store the machine statistics on Statwolf and access the oldest data, if needed. If all the experiments will give successful results, the ADMIN4D project will be in accordance with the IIOT fundamentals and all the surrounding environment will be developed respecting the ideas that seem outlining the future industries.

At the end of this dissertation, the plans and ideas for the future development of the work done during the thesis period will be clarified. While writing the thesis, it has been possible to exchange ideas and opinions with the other members of the project and to agree about the priorities of the activities. By doing so, all the tests described in this framework are planned to be implemented on Desamanera machines in the future.

Chapter 2

3D printer system

The realization of the 3D printer represents the main task for ADMIN4D and drives all the project members to an innovation on their corresponding sectors. As said before, regarding the IIoT, some LoRa sensors will be located into the artefacts in order to monitor the printing process. In this way, it is possible to study such sensors both from a connectivity point of view and from an electronic one. The printer, which is part of a comprehensive environment called demonstrator system (DS), will be developed with the idea of modularity, to build a customizable machine that can satisfy the costumer needs. For this purpose, a new distributed automation system has been developed.

2.1 The demonstrator system

By implementing a layered printing technology, the 3D printer allows to create objects starting from a bed constituted by dusts and binders. Studies have been performed in order to individuate which materials are more appropriate for this project and the marble dust seems to be a good choice. This process allows to use scrap materials to build new artefacts, resulting fascinating both from an environmental point of view and from a business one. Scrap materials have surely a low cost and can be re-utilized to create new products. An example is a printed piece of coral reef that, using bio-binders, is capable of constituting a valid resource for the surrounding fauna. From a practical point of view, using some specific printing heads, a controlled quantity of water is injected on the dusts to start the reaction. A system called *Recoater*, that lays the powders on the printing plane, allows to build the desired object by repeating the binding process layer by layer. Figure 2.1 shows the main elements that compose the DS and it helps to understand the printing process: the Recoater, together with the printing heads, move along the X

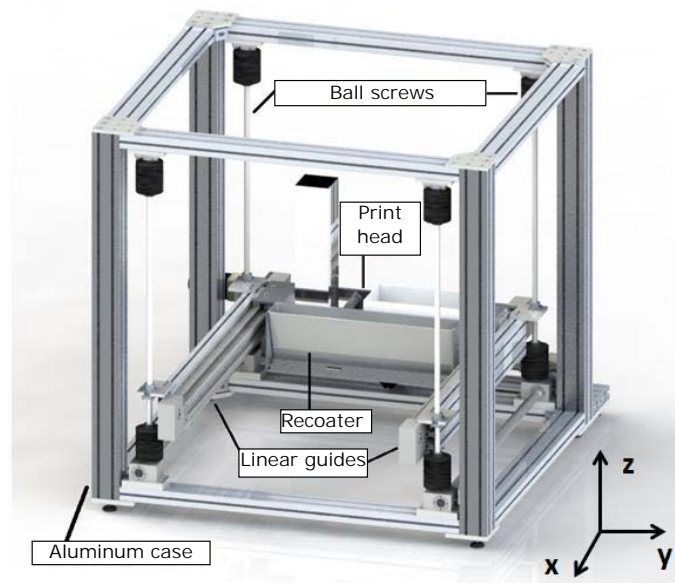


Figure 2.1: Main components of the 3D printer

axis to put down the material and, once ended the layer, they move along the Z axis to reach the position of the next one. While the movement on the X axis is helped by two linear guides connected by a transmission shaft, the Z axis operations are ensured by four mechanical jacks equipped with ball screws. The movements along each directions are handled by an electrical motor.

The dimension of the artefacts that can be printed are very impressive: the printing plane reaches the size of 4000 x 4000 x 3000 mm, on which the printing elements can move at a speed of 1.2 m/s, on the longitudinal direction, and 15 mm/s on the vertical one. At the sides of the printing plane there are plexiglass panels that allow to contain the dusts caused by the processes, and their height can be adjusted in relation of the object to print.

2.2 Automation system

Figure 2.2 illustrates the structure of the automation system of the 3D printer. As can be seen, it is composed by two communication networks related to the two main tasks of the project. The first one connects the Programmable Logic Controller (PLC), developed by Omron, to the devices involved in the printing process. The PLC has a time cycle of few milliseconds, in the worst cases, and has to be supported by a very fast connection. For this reason, a real-time Ethernet, named “EtherCAT”, is implemented. In the last years, EtherCAT had a great success thanks to its speed

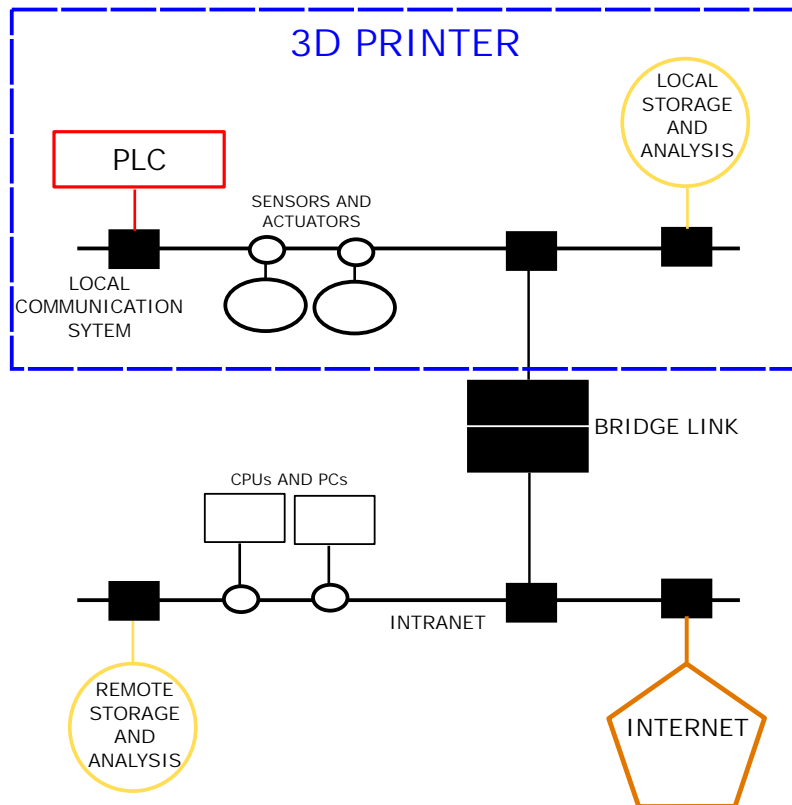


Figure 2.2: Scheme of the demonstrator system automation

and reliability, qualities that led Omron to implement a EtherCAT port on their PLC. This part of the network has to manage the data coming from the printing process. These data are divided in two categories: DS data and sensors data. DS data are the ones that come from the printer and represent the statistics of motors, frictions, guides, etc. The sensors data are the ones coming from the printed objects and from the machineries used in this context.

Inside the artefacts are placed sensors that transmit useful information that have to be gathered and analysed. Since no replacements are scheduled for those sensors, they have to guarantee a very low power consumption. For this reason, LPWAN technologies have been adopted and their implementations are discussed in the next chapters.

The second network is an Intranet that has to manage the data storage and analysis. To achieve this task, it has to be connected both to EtherCAT and to internet. To retrieve the elaborated data from the PLC and from the local storage, a gateway that acts as a bridge between the two kind of connections, has been developed. Then, a cloud database service, managed by the members of the ADMIN4D project, has been configured and connected to the network to store all the data.

Chapter 3

LoRa and LoRaWAN

3.1 The protocol

LoRa, developed by Semtech, defines the physical layer on which is based the LoRaWAN media access control (MAC) protocol. Since it is an open network standard, it operates in a non-licensed band below 1000 MHz. It makes use of a Chirp Spread Spectrum (CSS) modulation that allows to set the values of the spreading factor (SF) from 7 to 12. CSS is defined in the standard IEEE 802.15.4a and uses wideband linear frequency modulated chirp pulses to encode information. As chirp is intended a sinusoidal signal of frequency that increases or decreases over time. This kind of modulation works below the noise level resulting rather robust to interference and jamming. The bandwidth is fixed to $B = 125$ kHz in Europe and the duration of a symbol varies as function of SF from $2^7/B$ to $2^{12}/B$. As a consequence, the higher the spreading factor, the lower the data rate, but the better is the noise immunity. Thanks to the availability of six quasi orthogonal virtual channels, each LoRaWAN physical channel can be modeled as a superimposition of channels enhancing the robustness of the data transmission. Indeed, using different spreading factors, a station can receive more than one data packets simultaneously. LoRa properties can be summarized in:

- Scalable: LoRa modulation is both bandwidth and frequency scalable so it can be used for both narrowband and wideband applications;
- Low power consumption: the data rate and the duty cycle are very low, so the devices can be in idles state for long times, limiting power consumption;
- High robustness: LoRa communication is very resistant to both in-band and out-of-band interference mechanisms;

- Multipath / fading resistant: chirps are relatively broadband and so LoRa is immune to multipath and fading, resulting ideal for the urban environments;
- Doppler resistant: Doppler shift causes a relatively negligible shift in the time axis of the baseband signal. This offset tolerance mitigate the stringent requirement of synchronism between the devices' clocks;
- Long range capability;
- Enhanced network capacity: thanks to the orthogonal spreading factors, multiple signals can be transmitted at the same time and on the same channel.

The medium access policy establishes an ALOHA behaviour adopting both a duty-cycled transmission and a listen-before-talk-adaptive-frequency-agility. In this way collisions are avoided even if two devices are transmitting with the same SF on the same channel at the same time.

3.1.1 Chirp Spread Spectrum modulation

Chirp Spread Spectrum was created for radar applications during the Second World War on the security field. Abandoned after the war, it has been adopted in the latest years for many data communication systems due to its low transmission power requirements and its robustness.

To achieve the spreading on the entire spectrum, LoRa generates a chirp signal that varies in frequency continuously. A chirp is a tone in which the frequency either increases (up-chirp) or decreases (down-chirp) in time (Fig 3.1).

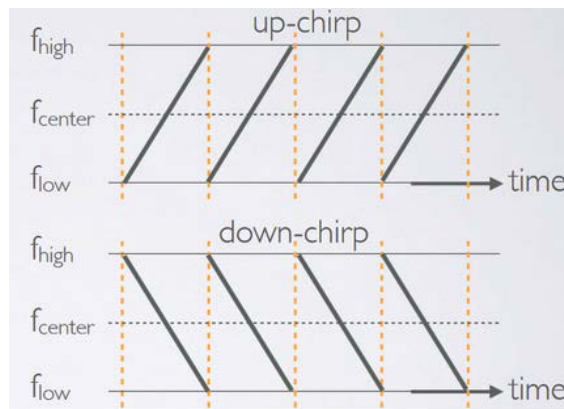


Figure 3.1: Chirps examples

By encoding the data in chirps or shifted chirps, LoRa is able to achieve all the properties highlighted before. For example, it uses a preamble of 8 up-chirps

followed by 2 down-chirps to determine the start of transmission and so to achieve synchronization (Fig 3.2).

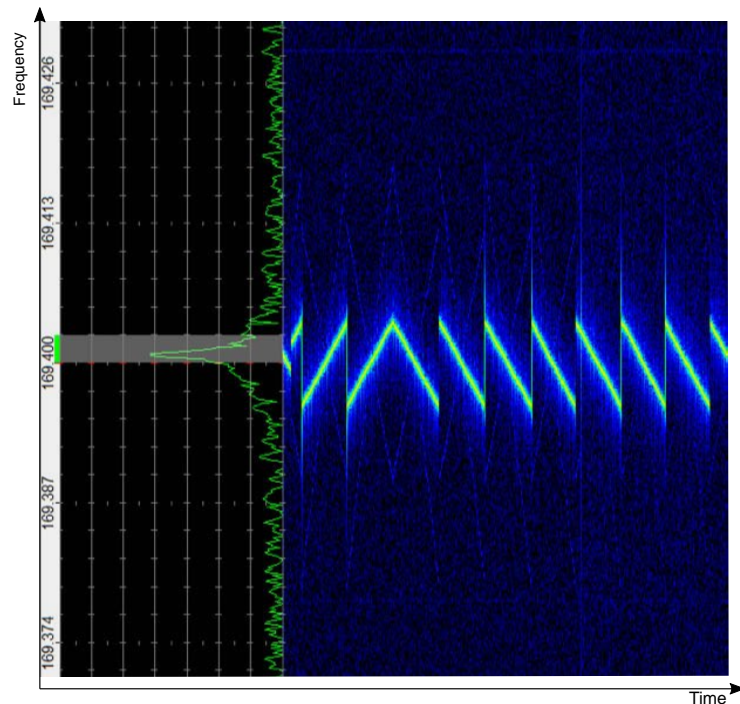


Figure 3.2: Example of a LoRa message preamble

After the preamble, the LoRa signal is modulated in frequency jumps, that establish how the data are encoded. In this context, the spreading factor plays a very important role: since each symbol is represented by a chirp, the latter can be divided into 2^{SF} steps (or chips) that can be re-arranged to represent the symbol. In the Figure 3.3 the spreading factor is 7 and so 7 bits are needed to represent the number 95, that in this example is 1011111.

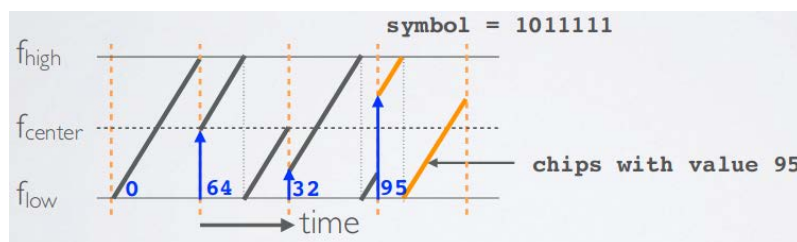


Figure 3.3: Easy example of LoRa modulation

In this way a SF12 can encode a chirp in 2^{12} chips increasing the accuracy with which each message is modulated, but, since it will use 12 bit for each symbol, it increases also the duration of the transmission. On Figure 3.4 it can be seen an example of a LoRa message that follows these rules:

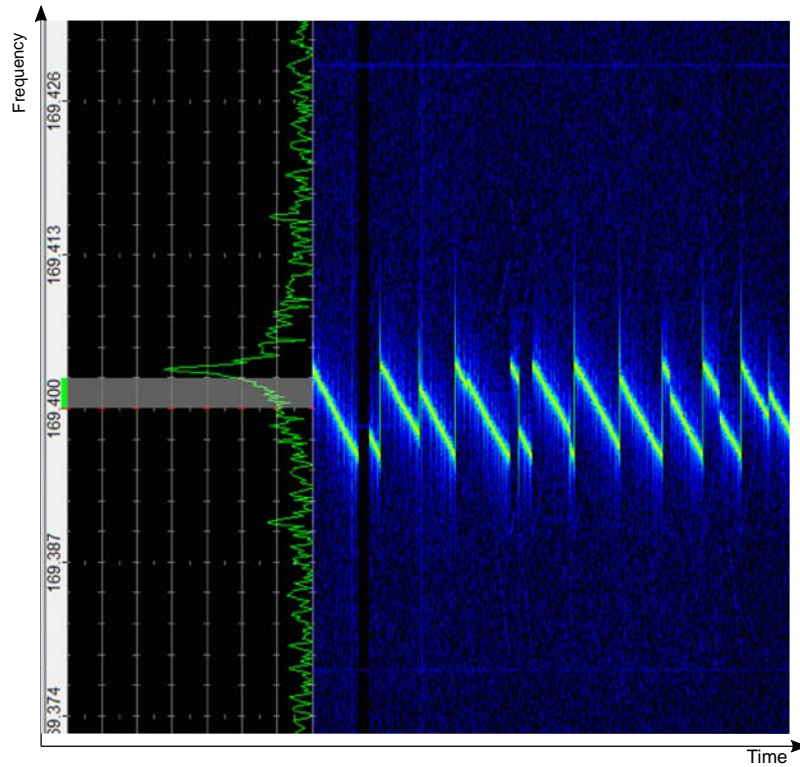


Figure 3.4: Modulated LoRa message example

3.1.2 LoRa frame

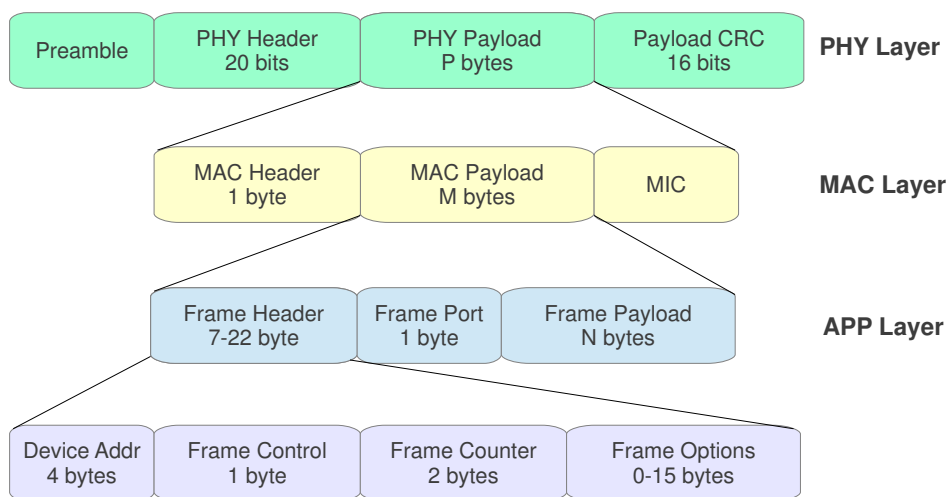


Figure 3.5: LoRa frame structure

Figure 3.5 shows the structure of a LoRa frame in all its layers. The first one is the physical layer that starts with a preamble. It handles the synchronization by encoding a sync word (0x34 for public purposes, 0x12 for private ones) that must be the same in all the devices that are communicating. After the preamble, there is the physical header, if it is present. It is transmitted with a code rate of 4/8 and indicates the size of the payload, the code rate used for the transmission and the indication whether a 16-bit CRC is added at the end of the frame or not. Also the header includes a CRC that allows the receiver to discard invalid packets. The payload is sent after the header and, at the end of the frame, there is the optional CRC.

The packet processed in the MAC layer consists in a MAC header, MAC payload and Message Integrity Code (MIC). The MAC header defines the kind of frame (data or management), transmission direction (uplink or downlink) and whether it shall be acknowledged. The payload contains either the actual data that have to be sent, or other information such as join requests or acks. The MIC portion is based on the header and the payload, and allows to prevent the forgery of messages as well as to authenticate the end node.

Moving to the APP layer, the MAC payload is divided in header, frame port and frame payload. The frame header defines the device address (DevAddr) that is the short address of the transmitting device, the frame control in which are defined the ack requirements, the frame counter that saves the number of frames sent and the frame options for commands used to change data rate, transmission power and connection validation, etc. The frame port indicates the port on which the message is sent and the frame payload represents the data encrypted using an AES key of 28 bytes.

3.2 The architecture

A LoRaWAN network is deployed in a cellular like star-of-stars topology (Fig. 3.6) and it is composed by three types of devices: End Device (ED), Gateway (GW) and Network Server (NS). In this application the EDs are represented by the sensors inside the printed artefacts and there is only one GW that plays the role of packets tunnel. In fact it simply takes as input the data from the sensors and transfers them to the NS without applying any change. In general EDs can belong to three different classes, namely A, B and C, but the used ones are of type A: after the access to the channel, they open two reception windows for a small amount of time in which they can receive data from the server. In this way the transmission and reception

are sparse in time but with very limited energy consumption. This property results appealing in an industrial environment in which the self-sufficiency is more and more requested.

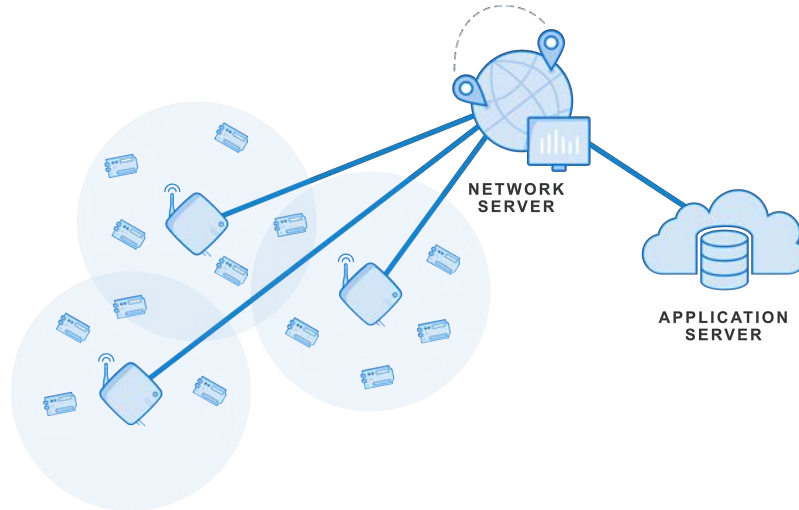


Figure 3.6: Graph showing the architecture of a LoRa connection

3.3 The communication

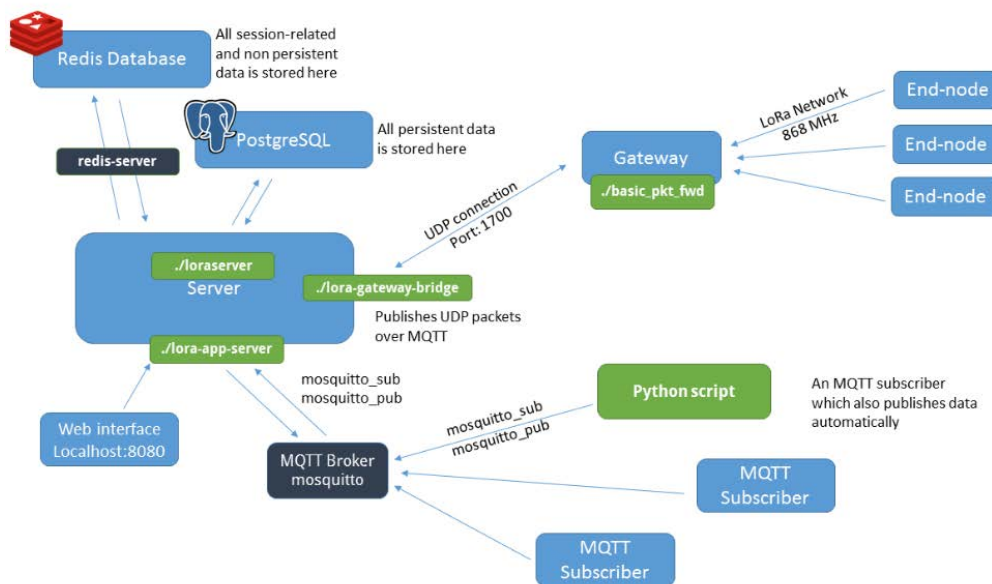


Figure 3.7: Graph showing the connection deployment of LoRa

For security reasons, all the communications on the LoRa architecture have multiple levels of message ciphering and signing. Each transmission needs two 128-b symmetric keys, called application and network keys (AppSKey and NwkSKey). Since the

management of all the resources in a LoRa connection is assigned to the Network Server, those keys deserve to identify the device that sent the messages. The connection to the servers is based on two joining mechanisms, the over-the-air (OTAA) and the activation by personalization (ABP). OTAA works with *JoinRequest* and *JoinAccept* messages and each node presents a 64-bit DevEUI, a 64-bit AppEUI, and a 128-bit AppKey. The DeviceEUI is a globally unique identifier comparable with the MAC address for a TCP/IP device. The Application EUI is used to identify the Application server and the Application Key is the key to access the server. So, first, the ED transmits an uplink message that contains its credentials and the payload with the data, then the server performs a downlink to the ED in which it allows the connection or refuses it, if the credentials do not match.

The ABP method differs from the OTAA one because it does not require the *Join* request from the nodes. The messages are sent with the DevAddr and both session keys (NwkSKey and AppSKey) making the node unique and distinguishable from the others. Moreover, while with the OTAA method the server can perform accurate checks on the incoming messages, the use of the ABP one is more risky.

Chapter 4

Testing LoRa

4.1 Connection range test

The first aim of this activity is to emulate a real application scenario and investigate the behaviour of the LoRa connection between EDs and GW. In this direction, a LoRa sensor has been located inside a cube printed directly by *Desamanea* in order to test the network performance in this specific configuration.

4.1.1 Hardware setup

To perform the tests it has been used a *Microchip LoRa Technology Evaluation Kit – 800* (Fig 4.1) which contains:

- LoRaWAN™ Gateway Core board;
- LoRaWAN Gateway Radio daughter card, 868MHz;
- Two RN2483 LoRa™ Motes with built-in light and temperature sensors.

The motes represent the EDs, while the Gateway Core board is the aforementioned LoRa gateway. The reference manual related to the kit suggests also to adopt a Network Server for the testing part that runs only on a Windows operating system. However, since all the experiments have been effectuated on a Linux based system, other solutions have been tested.

4.1.2 The experiments

Since the firmware present on the LoRa motes allows to perform an uplink (transmission to the GW) only every 5, 10, 20, 30, 60 minutes, the ED has been connected

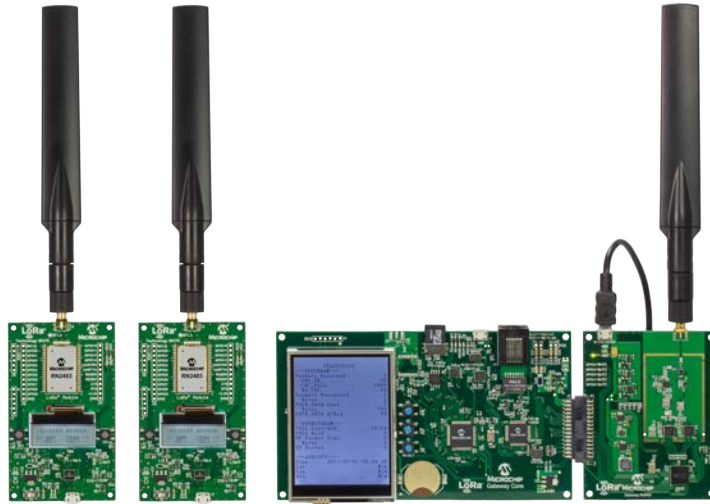


Figure 4.1: LoRa Technology Evaluation Kit - 800

to a computer via USB and controlled by the *LoRa dev Utility* provided by Microchip. This software, available for Windows, MAC OS and Linux, implements some very useful tools to manage LoRa devices (Figure 4.2). For example, it allows to send packets with different schedules. The time that elapses from a transmission to the next one depends on the spreading factor (0.05s for SF7, 0.138s for SF9, 1.04s for SF12). The payload consists in 8 bytes.

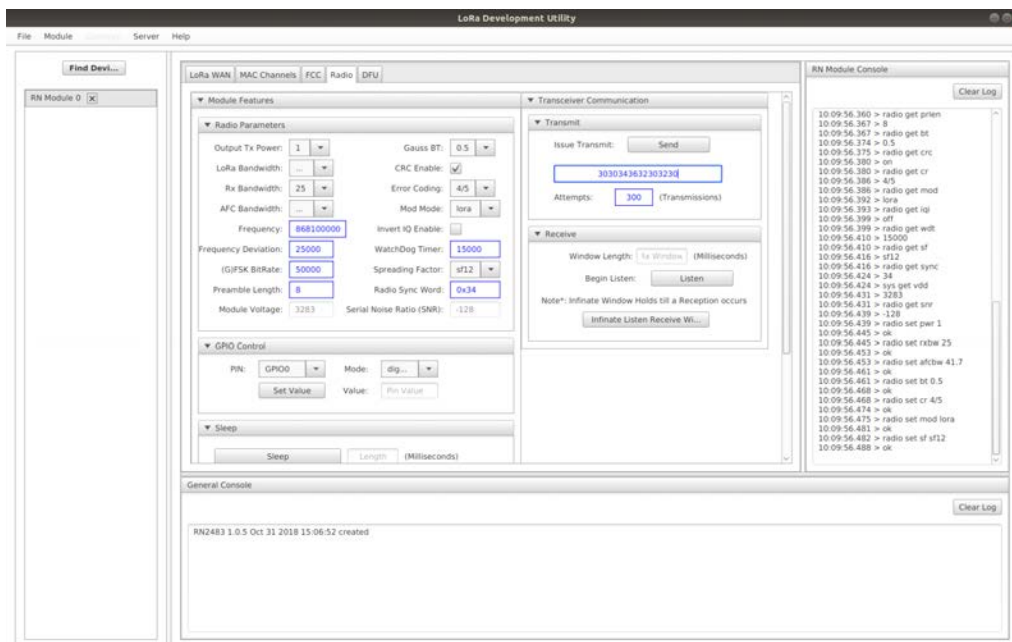


Figure 4.2: Radio tab of the LoRa mote in LoRa dev utility by Microchip

After the mote has been reconfigured, it has been placed into a cube printed by *Desamanea* using sandy materials collected on the banks of the Po river (Fig 4.3). The main variables that influence a LoRa connection are the distance, the spreading factor and the antennas. As a consequence, the experiments aimed to investigate the behavior of the transmissions for different operational configurations.

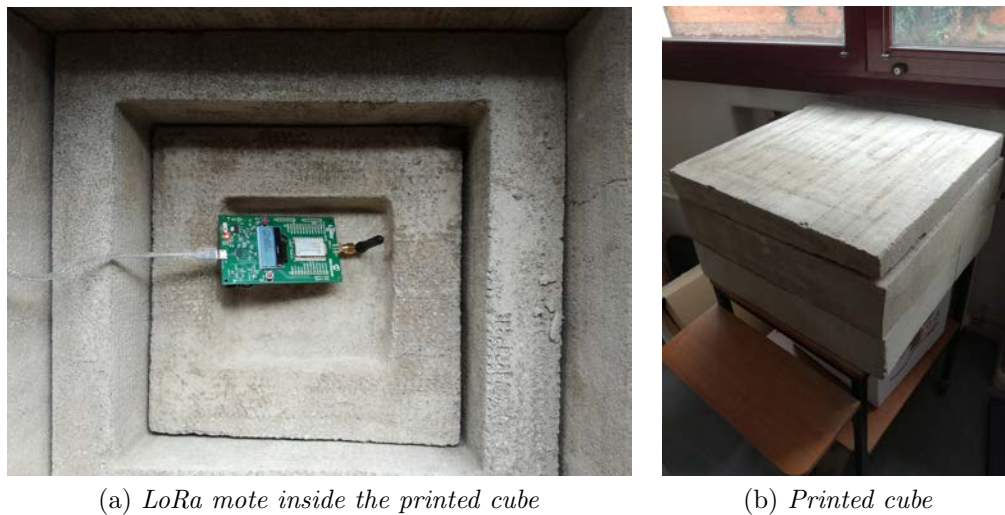


Figure 4.3: Experimental setup

Antennas performance

The first tests regarded the differences between a right angle antenna and a vertical one. While on the gateway the vertical one has been used for all the experiments, on the ED the two antennas have been switched. The small dimensions of the right angle antenna make it very suitable for the considered applications, but the connection could be worse with respect to the other antenna case. To check such an aspect, the gateway has been positioned in different locations of the Department of Information Engineering here in Padua where the ED was located on the printed cube. In this way the connection has been tested also in an environment that presents obstacles such buildings or electronic devices. The data collected consist in the analysis of 300 packets sent by the ED to the GW with a spreading factor of 12 and with a band of 868 MHz. The results can be seen in the Table 4.1, where the second column represents the percentage of packets received without error and the third denotes the number of packets that have not been received from the mote.

From the collected data it can be seen that the hypothesis on the effectiveness of the bigger antenna is not satisfied. In fact the graph in Figure 4.4 highlights an higher packet dispersion for the vertical antenna that makes it unsuitable for the project

Antenna	Correct %	Lost (packets)	Distance (m)
Vertical	95	4	2
Right angle	99	2	2
Vertical	100	0	10
Right angle	98	2	10
Vertical	99	2	45
Right angle	98	2	45
Vertical	97	15	66
Right angle	100	0	66
Vertical	77	213	110
Right angle	78	175	110
Vertical	78	189	130
Right angle	99	1	130
Vertical	91	73	150
Right angle	90	103	150

Table 4.1: Transmission statistics for the two antennas

purpose. This fact can be explained thinking about the nature of the two antennas: the vertical polarization of the vertical one causes an orientation sensitivity that affects the signal transmission, while the circular polarization of the right angle antenna extends the radius of the transmission.

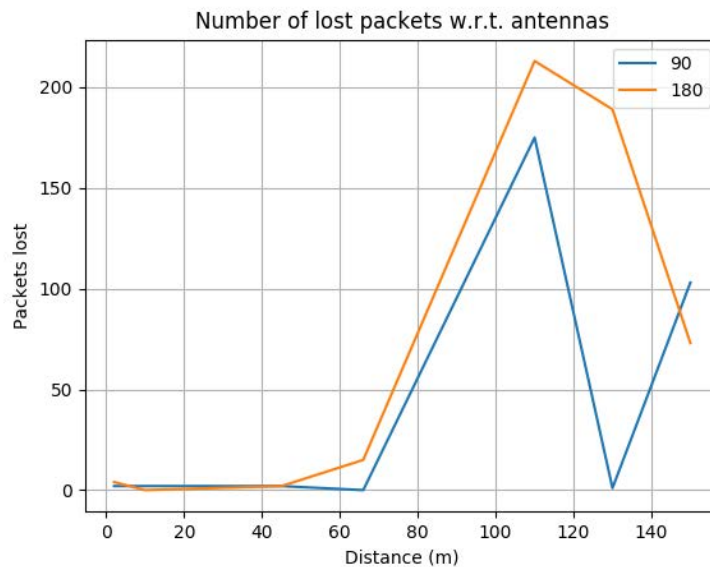


Figure 4.4: Graph showing the loss of packets with respect to the antennas

Due to these results, the vertical antenna has been discarded and it has been used only the right angle one for the subsequent experiments.

Spreading factor differences

According to the theory, to a higher spreading factor should correspond a better propagation. So, since the buildings in the university area introduce noise, the SF7 transmission should be weaker with respect to the SF12. To maintain a good level of homogeneity between different experiments, tests have been carried out using the same parameters and the same locations, varying exclusively the SF. The results are reported in both Table 4.2 and Figure 4.5:

	Correct %	Lost (packets)	Distance (m)
SF7	97	13	2
	97	10	10
	96	22	45
	39	269	66
	0	300	110
	0	300	130
	0	300	150
	98	12	150
	0	300	175
	98	11	250
SF9	80	3	2
	100	0	10
	99	4	45
	93	82	66
	0	300	110
	0	300	130
	0	300	150
	100	0	150
	0	300	175
	98	5	250
SF12	60	1	2
	99	1	10
	99	2	45
	94	58	66
	70	192	110
	0	300	130
	51	198	150
	100	0	150
	0	300	175
	100	1	250

Table 4.2: Spreading factor comparison in the morning

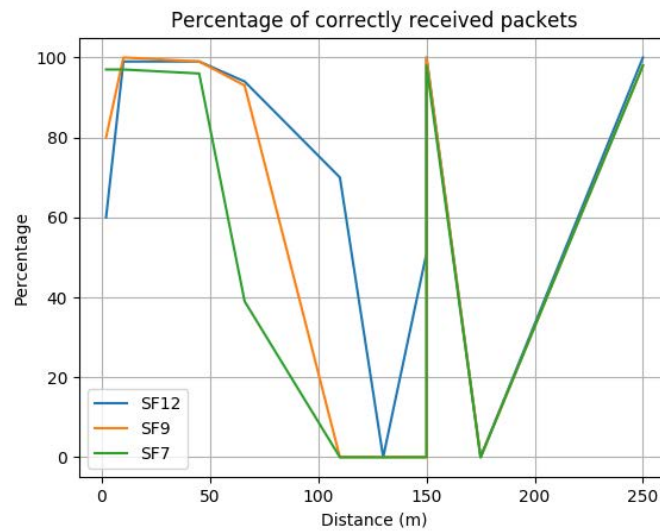


Figure 4.5: Graphs showing the correctly received packets with different spreading factors in the morning

The graphs clearly confirm the assumptions: the transmissions with a spreading factor of 12 are more robust than those with SF7 and 9. In most of the positions the number of lost packets for SF12 is less or equal with respect to the other spreading factors, and the percentage of correctly received packets is clearly higher. In some locations, for example at a distance of 110m or 150m, the gateway received packets only with a spreading factor of 12. At a distance of 175m, however, the shielding elements were such that even with the higher SF the gateway could not register any transmission. This fact could be explained by the presence of a huge number of people carrying electronic devices during the transmissions, that could affect them. To better assess such an hypothesis, other measurements have been performed in an interval of time in which the buildings were less crowded. The results can be seen on Tab 4.3 and Figure 4.6.

	Correct %	Lost (packets)	Distance (m)
SF7	97	7	2
	99	10	10
	97	17	45
	98	12	66
	0	300	110
	0	300	130
	0	300	150
	97	9	150
	0	300	175
	98	8	250
SF9	96	2	2
	98	15	10
	99	3	45
	99	3	66
	0	300	110
	0	300	130
	34	243	150
	93	39	150
	0	300	175
	98	7	250
SF12	98	2	2
	99	2	10
	99	2	45
	98	10	66
	89	102	110
	95	82	130
	96	24	150
	99	2	150
	20	296	175
	100	1	250

Table 4.3: Spreading factor comparison in the afternoon

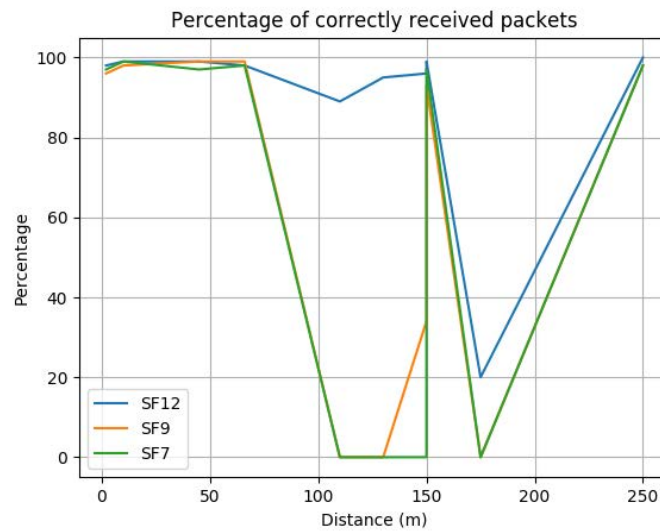


Figure 4.6: Graphs showing the correctly received packets with different spreading factors in the afternoon

From the data it can be seen that, while for the SF7 and SF9 cases the behaviour is almost the same, the SF12 reveals some important differences. The less number of devices reduced the action of the noise and allowed the gateway to receive packets even on the locations in which it did not collect any message before. The graphs, then, show a greater percentage of correctly received packets and hence a better performance of the communication.

4.2 Server connection tests

Once ensured that the transmission between end devices and gateway has been established, the connection from the GW to the network server has been tested. To do that it has been used a cloud service, named “The Things Network” (TTN) [28], that allows to register one or more gateways and devices on the platform, providing a LoRaWAN Network Server to manage them. TTN is widely utilized by IoT LoRa applications all over the world because of its easy implementation and high level of security.

4.2.1 Setting the hardware

After the registration of a new user on the TTN site and the installation of the LoRa Development Utility mentioned before, the gateway has to be configured. Once connected the GW to the computer via USB, LoRa Dev Utility detects it and opens the corresponding configuration window (Fig. 4.7). The steps to create a

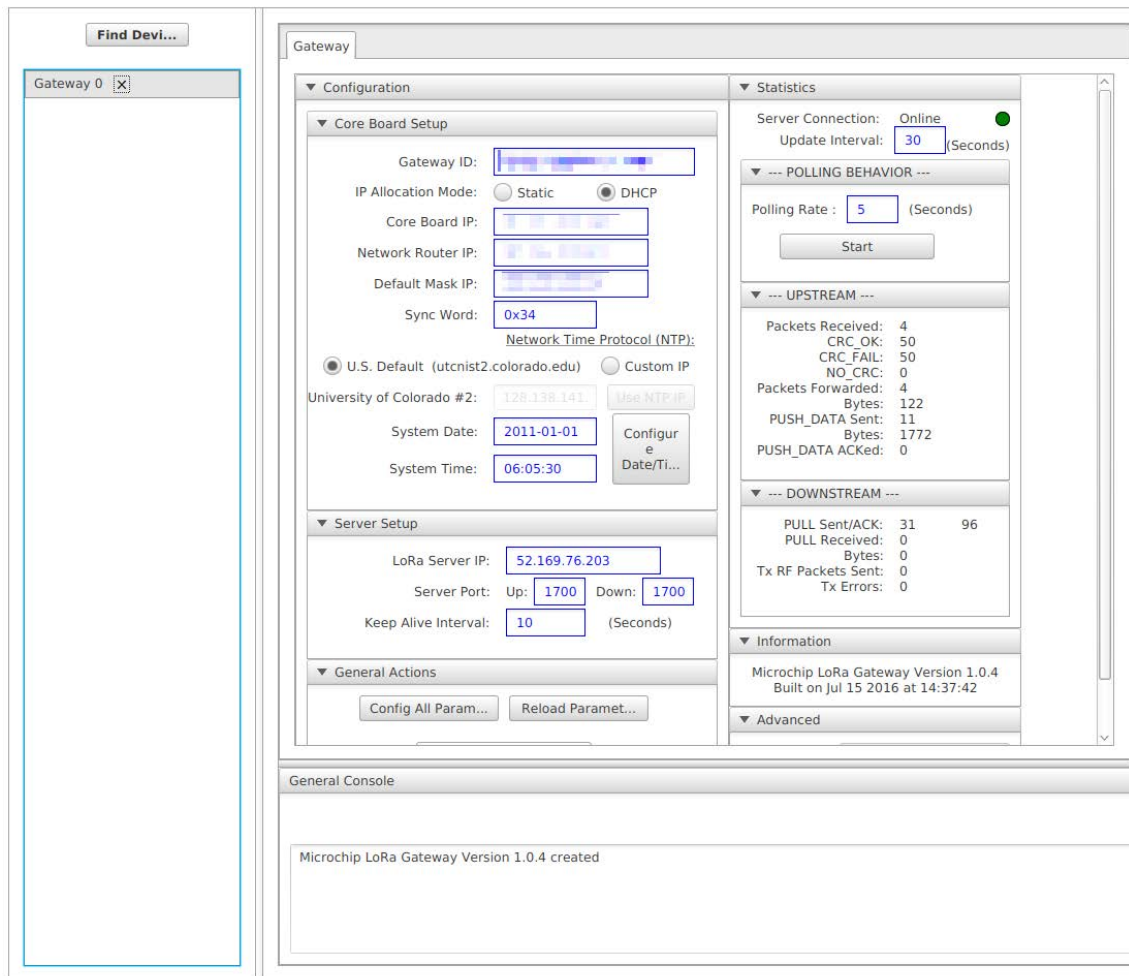


Figure 4.7: Example of the gateway tab of LoRa Development Utilities

connection with the server are:

- Set the gateway id (GWID), an unique identifier of 14 bytes that allows the server to monitor only the data passing through that specific GW;
- Select “DHCP” for the IP allocation mode. In this way the GW can retrieve the information about its own IP and the the router to which it is connected;
- Set the time accurately. It is suggested to be precise at seconds although the transmission would be out of sync;
- Set the TTN IP on the LoRa Server IP flag. The server IP can be found on the site and differs in base of the region in which the gateway is operating. For Europe it is 52.169.76.203;
- Save the settings on the SD card present on the gateway, so to not loose the configuration once the GW is switched off;

- Check if the "Server Connection" displays "Online". If not, try to enable the port forwarding on port 1700 of the router.

Once established the connection between server and gateway, the latter has to be registered on the TTN platform. To do that, it is necessary to connect to the "Console" page on its own TTN profile, select the "Gateway" tab and register a new GW filling all the parameters with the same ones used on the LoRa Development Utility.

Now that the gateway has been registered, to see the data coming from the end devices, first it is needed to create an application on the TTN dashboard, and, subsequently, to configure the LoRa motes with the chosen parameters. In detail, the following actions are required:

- On the "Console" page, select "Application" and then create a new one. Once completed this step, it will be assigned an unique identifier called "AppEUI" that will be needed for the connection with this specific application;
- Register a new device on the application just created. By choosing "auto-generate" on "DeviceEUI" and "AppKey" fields, TTN will assign the keys to the device;
- On the "Device Overview" page it is possible to read all the data corresponding to the registered device (Fig 4.8).

At the end of the configuration, all the parameters that are needed to send data from EDs to the NS will be available. The last step is to connect the LoRa mote to the LoRa Development Utility and to fill the fields with the corresponding values. To test the connection, click the join button on the "LoRa WAN" tab of the mote on the utility; a "Join OTAA, accepted" should appear on the log. Make sure that the gateway is connected to the server before starting this procedure. Once accepted, data received from the motes are available on the "Data" tab of the corresponding device in the TTN dashboard (Fig. 4.9)

The screenshot shows the 'DEVICE OVERVIEW' page for a device with Application ID 'loramotes'. The device ID is 'loramote0' and the description is 'Mote0'. The activation method is 'OTAA'. The device EUI, Application EUI, App Key, Device Address, Network Session Key, and App Session Key are all displayed with copy icons. The status is '26 seconds ago' and the frames up/down counts are 2511 and 1 respectively.

Figure 4.8: Example of "Device Overview" on the TTN dashboard

The screenshot shows the 'APPLICATION DATA' page with a table of data for a LoRa mote. The table has columns for time, counter, port, and payload. The data is filtered by 'uplink'.

time	counter	port	payload
15:50:29	2517	155	33 31 36 20 30 32 32 00
15:40:08	2515	70	34 30 38 20 30 32 32 00
15:34:58	2514	84	34 32 39 20 30 32 32 00
15:24:38	2512	125	33 37 33 20 30 32 32 00

Figure 4.9: Example of "Data" tab of a LoRa mote in the TTN dashboard

Problems encountered on the hardware set up

Configuring the hardware, arose some problems mainly due to the operating system of the machine in which the LoRa Development Utility software was installed. Indeed, such software tool has been originally created for Windows and its functions are not optimized for other systems. Using Linux, to set the parameters on the connected devices in the right way, it is necessary to make sure that the commands are correctly transferred. In many cases, looking at the terminal associated to the software, which acts as a log process, the sent commands did not show up. The only

way to solve the problem has been to continue to set the same parameter until it reaches the device.

Another important problem, that affected the initial part of the work, was the lost of the access keys from the memory of the LoRa mote. In fact, when rebooted, the end device could not send data to the server anymore because the AppKey was not retained in the memory. To solve the problem, the mote has been updated with the version 1.0.5 of its firmware, that fixed the issue.

4.2.2 Retrieving data

TTN implements some application integrations that help to manage the data. One of them is called “Data Storage” and stores the received data up to a week. Once enabled, it is possible to use the APIs present on its platform. The one used to download the stored messages is called “query” and generates the following code:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
KEY' 'https://loramotes.data.thethingsnetwork.org/api/v2/query?last=PERIOD'
```

“KEY” and “PERIOD” are, respectively, the key used to authorize the API and the period of time to check (See [30] for more information). The output of the command consists in a JSON file containing the different information about the device id of the mote, the raw payload and the time at which it has been received, for example:

```
"device_id":"loramote0","raw":"MzIyIDAxOQA=","time":"2019-10-16T14:31:25.188808639Z"
```

4.3 Data Storage

Since the integration available from TTN can store data only for a limited period of time, this section aims to find a way to not loose the oldest data. This is made possible by the collaboration with *DataVeneta S.R.L* and *Statwolf* that are partners of the ADMIN4D project.

In this specific scenario, a MySQL database is available to store the data coming from the sensors involved in the processes. Its name is “statwolf” and presents a “plc” tab composed by 4 columns: “sensore”, “dataora”, “tipo”, “valore” (Fig 4.10). The first column indicates the ID of the sensor that uploaded the data. For example, 104 stands for the light and 105 for the temperature sensor of the Padua LoRa devices. The second column represents day and time at which the data has been sent, the third one indicates the kind of sensor and the fourth the corresponding value.

sensori	$\begin{matrix} Z \\ \downarrow \\ A \end{matrix}$ dataora	tipo	valore
104	2019-10-24 15:06:33	LUCE	452
105	2019-10-24 15:06:33	TEMP	22
104	2019-10-24 15:01:23	LUCE	277
105	2019-10-24 15:01:23	TEMP	22
104	2019-10-24 14:56:13	LUCE	335
105	2019-10-24 14:56:13	TEMP	22
104	2019-10-24 14:51:03	LUCE	301
105	2019-10-24 14:51:03	TEMP	22
104	2019-10-24 14:45:53	LUCE	327

Figure 4.10: Example of data stored in the “statwolf” database

The last step to perform is the upload of the data received by the TTN integration into the MySQL database.

4.3.1 The algorithm

To send data to the database, a Python code called “send_data.py” has been developed. By importing the “mysql.connector” library, Python is able to connect to a MySQL server and to perform the main instructions to manage it. One of the main problems of this task is represented by the difference between the date format of the database and that of the JSON file downloaded from TTN. The problem has been solved as described by algorithm 1:

```

Begin;
Database credentials definition;
Open connection with database;
while connected do
    retrieve last date time corresponding to the temperature sensor (last);
    retrieve data of the last week from TTN integration;
    split data in a structure in accordance with the database columns;
    convert date time in the database format and with the current zone time;
    if last is present on the TTN data then
        | send all TTN data after last;
    else
        | send all TTN data;
    end
    Close connection;
    Sleep 5 mins;
end
End;

```

Algorithm 1: send_data.py pseudocode

As can be seen, the *if* statement, that checks if the last date time on the database is present on the downloaded data, is necessary to avoid the same data. If the check fails, all the TTN data are uploaded. The latter case can be verified in two scenarios: or there are no data on the database, or the last upload dates back to more than one week (as mentioned before, the storage API can retain the data for a week). In this case, the last upload does not appear on the downloaded TTN data, and so all the scores are uploaded. The *sleep* command can be tuned in base of the period that elapses between a LoRa transmission and the next one. Figure 4.11 helps to understand the behaviour of the algorithm just explained.

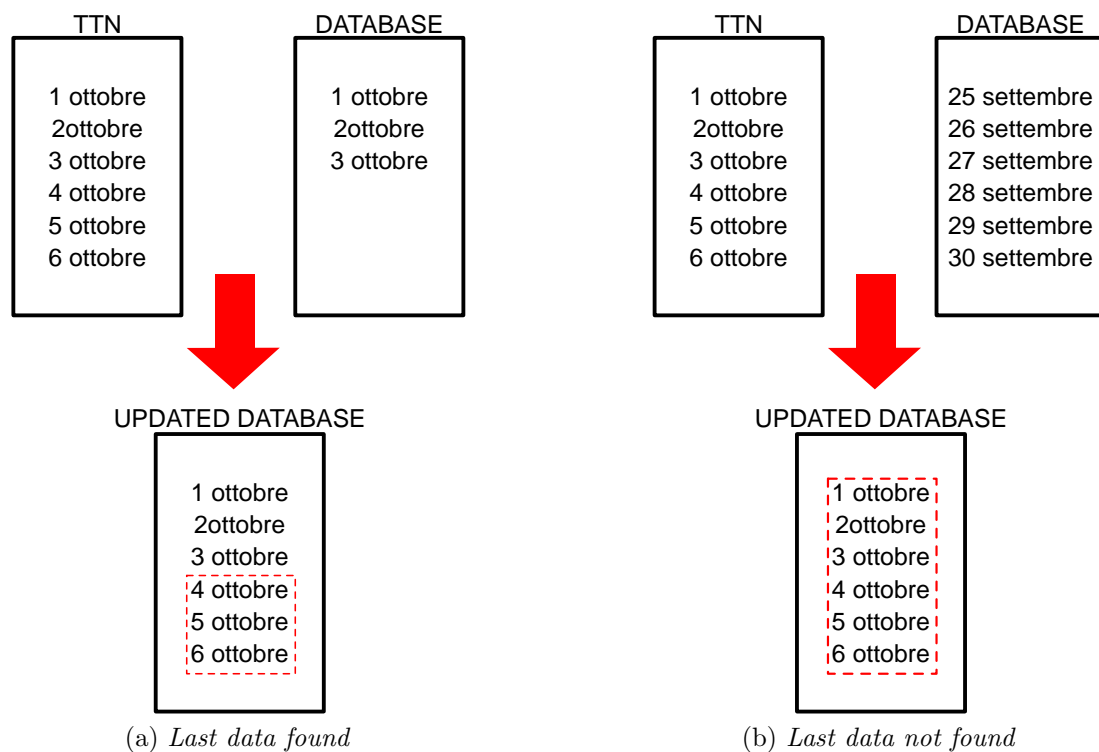


Figure 4.11: Example of behaviour of the algorithm

4.4 Data handling

The automation system of the printers makes use of an Omron Programmable Logic Controller (PLC) (Figure 4.12). Such a PLC is able to receive data from the sensors during the printing process and to store their values. Also, such data may be analysed to implement a real-time feed-back during the printing process.



Figure 4.12: Omron NX102-9020

4.4.1 The PLC

The Omron NX102-9020 is equipped with two industrial Ethernet ports and an EtherCAT one through which it can communicate with all the connected devices. Up to 32 racks can be attached directly to its bus, but it can transfer a maximum voltage of 10V, giving the possibility to adopt a huge amount of inputs and outputs. Since the CPU can supply only the buses of its attached racks, the NX-PF0730 power supply module has been added to allow also the transmission of data between all the PLC parts. Figure 4.12 shows also the presence of three others modules included to test the CPU features: NX-ID4442, as a DC input unit; NX-ILM400, as an IO-Link master unit; NX-OD4256, as a transistor output unit. To transfer power to the CPU, the S8VK-G12024 power supply has been connected following the scheme in Figure 4.13. The blue connections stand for the positive pins, while the brown ones represent the negative pins. The yellow cable links the ground of the PLC with the ground of the power supply, so to avoid any kind of risk. Once switched on, the lights on the PLC and on the power supply should light up as in Figure 4.12. One of the main features of the PLC that was crucial for the project, is the ability to implement databases connections such as the MySQL used in the ADMIN4D project. The connection with the database allows to store all the statistics coming from the components used in the printing processes such as motors, guidances, frictions, etc. The analysis of such data may allow to prevent failures. Moreover, maintenance can be scheduled when necessary.

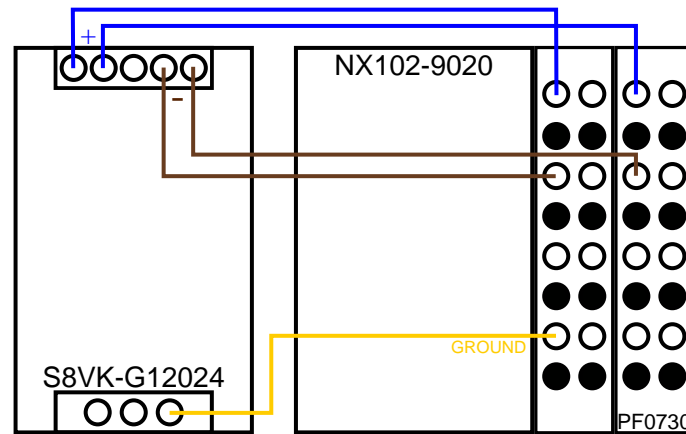


Figure 4.13: Electric scheme of the connection between PLC and power supply

4.4.2 Developed software

To program the NX102-9020 CPU, the Sysmac Studio Development Environment has to be used. This software contains all the libraries necessary to realize any kind of task that the PLC can perform. Two programs have been created, one that handles the link between PLC and a possible local LoRa network server, and one that tests the connection between PLC and MySQL database. To do that, it has been necessary to understand the structure of the software and to become familiar with the ladder language. In the next sections the functions used to create the algorithms will be explained in detail.

PLC setup

To program the PLC, it is first necessary to configure it, in order to establish a communication with the PC in which Sysmac Studio is installed. Hence, the first step to perform is to create a new project on Sysmac Studio with the right model and firmware version, written also on the side of the PLC. Once created, the hardware configuration has to be reported also on the software. To do so, it is necessary to arrange the racks on the “Configurations and Setup/CPU Rack” tab, paying attention to add them in the right order (Figure 4.14). Assembled the PLC configuration, the Ethernet ports have to be set on the “Configurations and Setup/Controller Setup/Built-in EtherNet/IP Port Settings”. The “IP address” and the “Subnet mask” have to match the ones given by the network and the “Default gateway” must to be equal the IP of the router at which the PLC is connected (Figure 4.15). Once connected PLC and PC via Ethernet to the same network, the “Controller/Communication Setup” tab on the status bar allows to test the connection between the devices. Setting the IP address of the port through which the

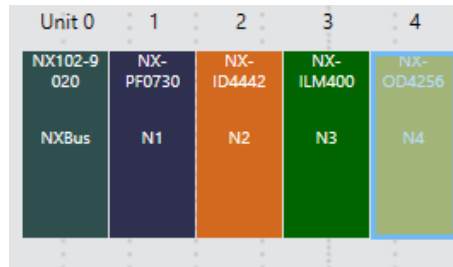


Figure 4.14: Racks configured on the Sysmac Studio Environment

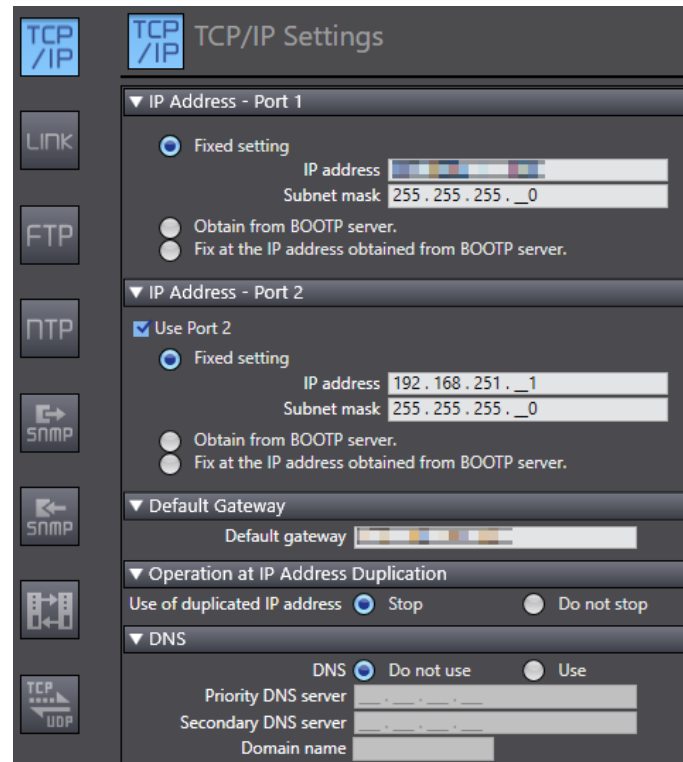


Figure 4.15: Ethernet ports settings on the Sysmac Studio Environment

PLC is connected and pressing the “Ethernet communication test” button, should give a “Test ok” message that testifies the successful of the link (Figure 4.16). Once verified the connection, it is possible to activate the “Online” mode to perform the tests and debugging on the PLC in real time.

Interactions with LoRa sensors

The LoRa communication is controlled entirely by the network server, that has also the responsibility to store the incoming messages. In particular, the capability to download and store the messages on a file designed by the administrator, has been considered an hypothesis. Hence, an algorithm capable to establish a connection with the PLC via TCP and to transfer the desired data has been developed. Since

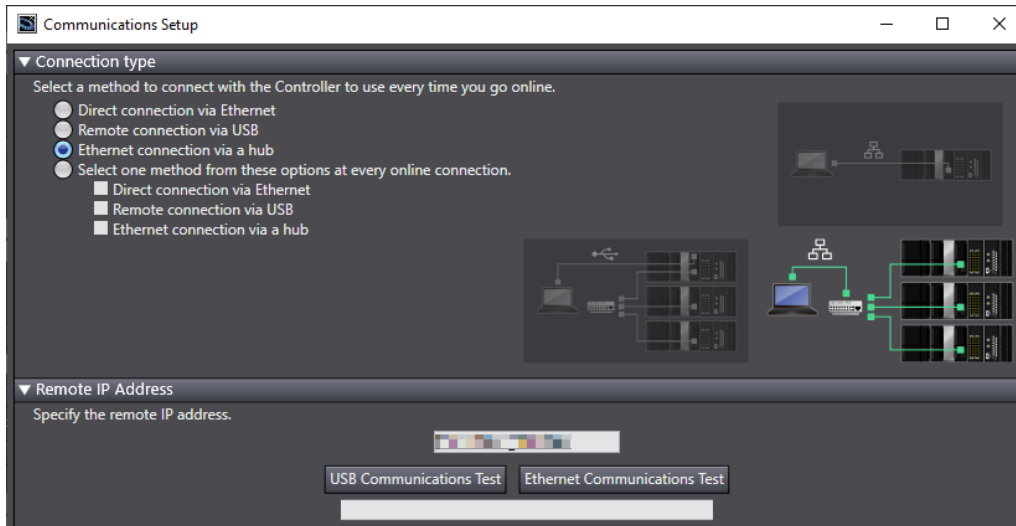


Figure 4.16: Connection test on the Sysmac Studio Environment

the purpose of this connection is monitoring the data coming from the LoRa sensor in order to check the status of the printed object, only the last available data is sent to the PLC. The oldest data are not useful to adjust the printing process that requires the most updated data as fast as possible.

The NX102-9020 CPU supports the TCP Socket protocol that helps the exchanging of information via TCP/IP. Sockets are a kind of data structure provided by the operating systems that allows to create a client/server relationship between the communicating devices. In this specific case, the PLC is treated as server and the device that sends the data as client. (Figure 4.17 helps the comprehension)

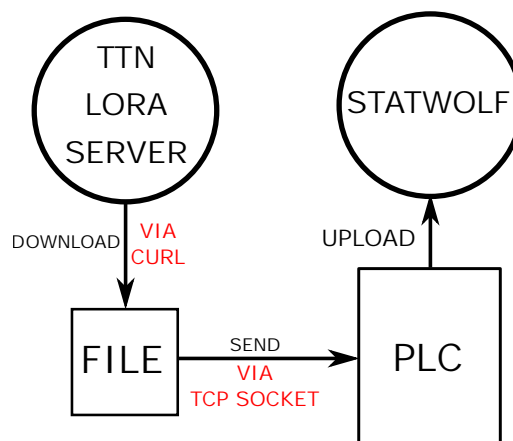


Figure 4.17: Scheme of the connection between TTN and PLC

Server side

The algorithm developed for the server uses sockets to receive the sensor statistics, and its structure is explained by the Algorithm 2: The PLC waits for a connection, then, when established, opens a receive instance to retrieve the data from the connected client. Since the socket protocol provides for a byte stream connection, the CPU has to convert the data in to a string and split it. Hence, the string members are saved into the SD card.

```

Begin;
Wait for a socket connection request;
if connected then
    | Receive the socket;
    | Close the connection;
end
Manage the obtained bytes;
Save the data on the inserted SD card;
End;

```

Algorithm 2: Receive socket pseudocode

Now will be investigated in the specific the Sysmac Studio functions used to implement it:

SkdTCPAccept:

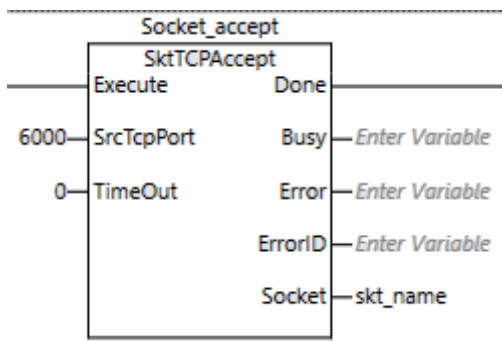


Figure 4.18: SkdTCPAccept Sysmac studio function

The SkdTCPAccept function waits for the time set with *TimeOut* for a connection on the port *SrcTCPPort* to be established with the remote node. If *TimeOut* is set to 0, it stays idle until a connection request is sent. The instruction reference manual suggests to use the port 6000 to establish the connection because of its rare occupation by other programs. As output it returns an *_sSocket* object that will be take as input from the other function as identifier of the specific socket connection. Other

outputs, that are common for the majority of the Sysmac Studio functions, are the identifiers of the function block status: *Busy* is a boolean variable and it is *True* if the function block is being executed but it is not finished yet; *Error* is a

boolean variable and it is *True* if the execution of the function terminates with an error; *ErrorID* is a word variable and stores the identifier of the error, if there is one.

SkTCPRcv:

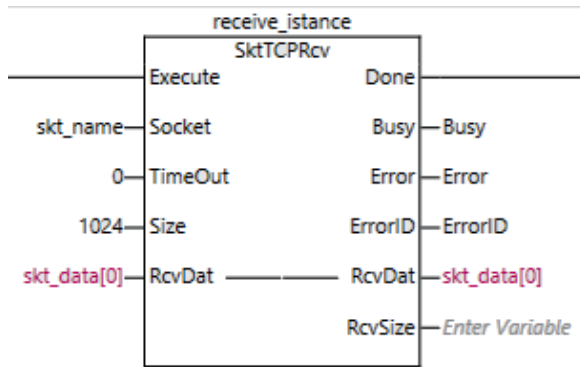


Figure 4.19: SkTCPRcv Sysmac studio function

The SkTCPRcv function stores in the “RcvDat” array the bytes located in the receive buffer for the socket indicated by “Socket”. The number of bytes to store is specified with “Size” and has an upper limit of 2000 bytes. The number of bytes that is actually stored is assigned to “RcvSize”. If there is no data in the receive buffer, the instruction waits for the time that is set with “TimeOut”. Setting it to 0, erases the Timeout exception.

In the considered case (Figure 4.19) “skt_name” is the variable that identifies the socket connection established with the previous function. The received data coming from the client are stored in “skt_data”, an array of bytes with a size of 1024. The information about the size of the received data is not useful in this specific application.

“skt_name” is the variable that identifies the socket connection established with the previous function.

SkClose:

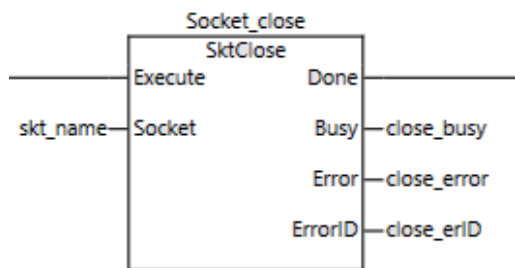


Figure 4.20: SkClose Sysmac studio function

The SkClose instruction closes the socket that is specified with “Socket”, whether TCP or UDP. If the socket uses TCP, it is disconnected before it is closed. All TCP and UDP ports that currently use the socket service are closed. In Figure 4.20 the “Socket” variable is in accordance with the functions previously used and the output variable are saved in order to monitor the status of the socket. The program can not conclude the current cycle if the socket connection has not been closed, otherwise the PLC can not accept other connec-

connection has not been closed, otherwise the PLC can not accept other connec-

tions. For this reason the “SktClose” function has been positioned right after the receive function, so to give to the CPU the possibility to accept other connections as soon as possible.

SubDelimiter:

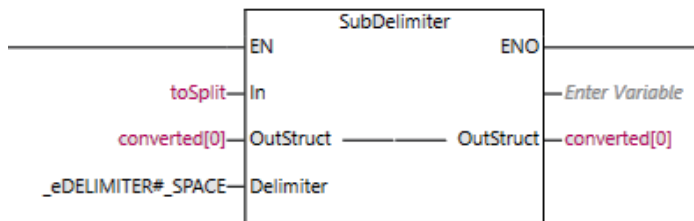


Figure 4.21: SubDelimiter Sysmac studio function

The SubDelimiter function converts the input string and stores the results on the members of storage structure “OutStruct”. The text strings are delimited with the “Delimiter”. The results are stored on the structure members from the first to the last. Considering Figure 4.21, “toSplit” represents the string obtained after the conversion of the byte stream coming from the client. “Converted” is a structure composed by “StatwolfColumns”. The latter is a defined variable configured in the following way (Table 4.4):

Name	Base Type
sensore	STRING[10]
giorno	STRING[11]
ora	STRING[9]
luce	STRING[6]
temp	STRING[6]
delimiter	STRING[1]

Table 4.4: StatwolfColumns structure

The structure is very similar to the one of the database, but it was necessary to add another field called “delimiter” to allow the program to write the data line by line on the SD card. This fact will be clear once explained the client side of this connection.

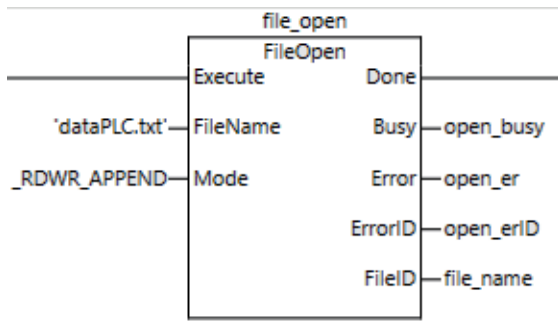
FileOpen:

Figure 4.22: FileOpen Sysmac studio function

The FileOpen function opens the file specified by “FileName” in the SD Memory Card in the mode specified by “Mode”. The result is outputted to file ID “FileID”. The latter is used to specify the file in other instructions, such as FileRead and FileWrite. The “_RDWR_APPEND” mode indicates the possibility to whether read or write the file without modifying its contents (Figure 4.22). In this way the program does no overwrite the data stored on the SD card. Moreover allows the CPU to create

the file if it does no exist on the specific path. All the output variables that monitor the status of the function has been saved. If the file is not accessible or there are errors, as the missing of the SD card, the program has to handle the situation and respond in an appropriate way.

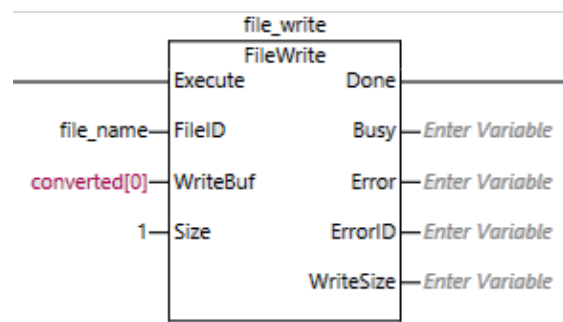
FileWrite:

Figure 4.23: FileWrite Sysmac studio function

The FileWrite function writes data to the position indicated by “FileID” in the SD Memory Card. The contents of the write buffer, “WriteBuf”, are written into the file. The data size, that is actually written, is outputted to “WriteSize”. “converted[0]”, in Figure 4.23, is the same as before. The function writes the values stored in the structure one after the other on the file in the SD card. This is the reason for which the “delimiter” field has been added at the end of the structure. It contains the `\n` character that makes easier the reading of the file.

End:

At the end, the program resets all the flags used to check the status of the functions involved and sets the flag that call the SktTcpAccept function in order to wait for

the next connection.

Client side

The server structure has been created, now the client has to implement an interface to communicate with it. So, first let's give an example of the developed code:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((IP, port))
converted = str.encode('loramote0 2019-10-30 17:25:27 41 20 \n')
print(f"Sending msg...")
s.sendall(converted)
print("Sent!")
s.close()
```

The language used for this small algorithm is Python. It uses a library, named `socket`, that contains functions that help the realization of both the client and the server side of a socket TCP connection. Examining the code, it can be seen that the first line creates a socket object, “s”, that is connected to the specific IP of the server in the second line. Then, using the string method `encode()`, it converts the string containing the sensor data in the UTF8 format, in agreement with the data type requested by the socket protocol. After the conversion, it sends the bytes to the specific address and then closes the connection. The string is composed by `sensor-day-time-light-temp-\n`, the same structure of the “StatwolfColumns” seen at the server side. In this way, the PLC can store the data on its structures with the same order in the right way.

This algorithm is implemented on the main program that handles the data coming from the TTN server. Every minute it checks if there are new data and, if so, sends them to the PLC via TCP socket.

Connection with the MySQL database

The purpose of this section is to test the connection between the NX102-9020 CPU and the MySQL database created for this project. The first step to achieve this goal is the configuration of such database on Sysmac Studio. On the “Configurations and Setup/Host Connection Settings/DB Connection” create a new DB and set all the connection access keys (Figure 4.24). Once filled the tab, the success of the connection can be evaluated by pressing the “Communication Test” button.

Once arranged the communication, the test program has been developed. It retrieves n sensors data from the database and saves the values on specific structures. The

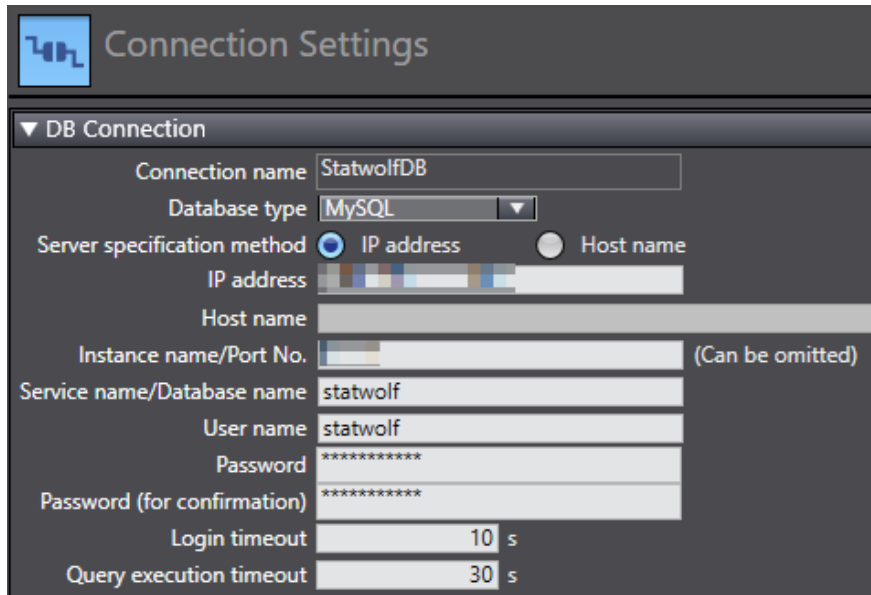


Figure 4.24: Database configuration tab

code is explained in the Algorithm 3:

Begin;

Establish a connection with the database;

Create functions that map data sensor on the defined PLC structures;

Save the last data coming from the light and temperature sensors;

Close the connection;

End;

Algorithm 3: Receive DBdata pseudocode

The Sysmac studio functions used to implement such algorithm are the following:

DB_Connect:

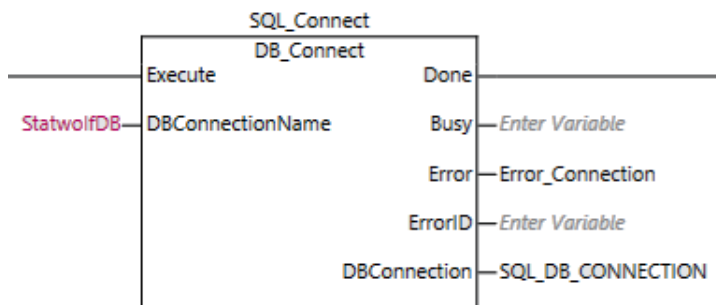


Figure 4.25: DB_Connect Sysmac studio function

if the database is not reachable.

This instruction is used to connect the PLC to the database, specified in the “DBConnectionName” string input variable. The “DBConnectionName” is the one set during the configuration of the database. On Figure the 4.25 “Error_Connection” variable stops the execution of the program

“SQL_DB.CONNECTION” is the identifier of the established connection that will be used to the data transfer.

DB_CreateMapping:

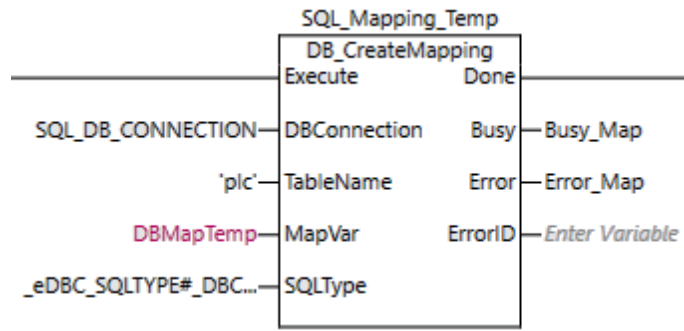


Figure 4.26: DB_CreateMapping Sysmac studio function

able that has to coincide with the name of the reference table on the database; “SQLType” that defines the kind of map. For example, in Figure 4.26 has been used “_eDBC_SQLTYPE#_DBC_SQLTYPE_SELECT” that specifies that the map will be used to retrieve data from the database. The last input is “MapVar” that points to the structure that will be used to store data. In this case “DbMapTemp” has the following structure (Table 4.5):

Name	Base Type
sensore	INT
tipo	STRING[256]
dataora	STRING[256]
valore	INT

Table 4.5: StatwolfColumns structure

The elements of the structure correspond to the columns of the “plc” tab on the MySQL database. In this way, when retrieved, the data will be saved in the corresponding fields. Output variables has been used to monitor the status of the function block.

This instruction is used to map the table, specified in the “TableName” input variable, with a DB Map Variable, specified in the “MapVar” input variable. It is required to send commands to the database with other Sysmac functions. It takes as input: “DBConnection” that identifies the connection established before; “TableName” that is a string variable

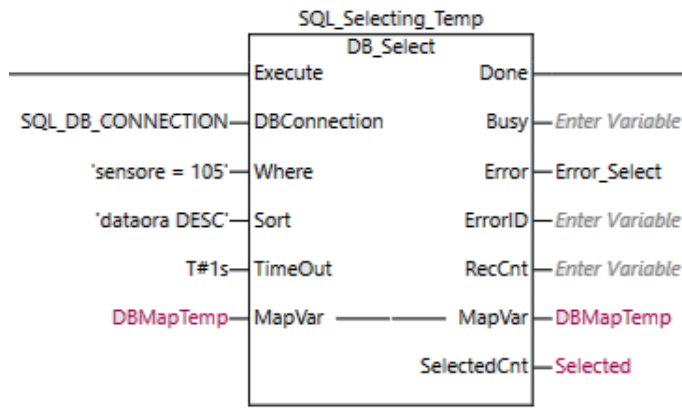
DB_Select:

Figure 4.27: DB_Select Sysmac studio function

example “dataora DESC” starts the retrieve process from the last row, in temporal order, in the “dataora” column); “TimeOut” sets the maximum time interval to detect the instruction execution. On “SelectedCnt” output variable is stored the number of items that have been retrieved.

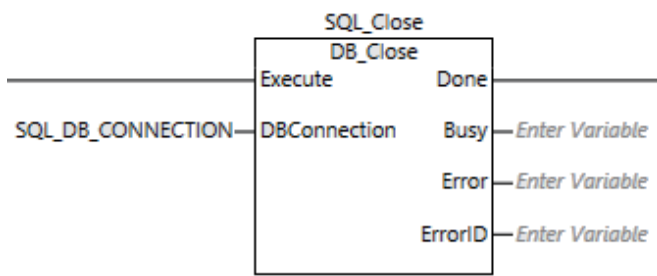
DB_Close:

Figure 4.28: DB_Close Sysmac studio function

This instruction is used to close the database connection specified in the “DBConnection” input variable. Output variables can be used to make sure that the program does not end until the connection with the database is closed. It is not recommended to open a connection with the same database without closing the previous one.

End:

After ensured the closure of the communication, the program resets all the parameters used to check the status of the processes and then waits for a signal that enables

This function is used to retrieve records from a table mapped by a “DB.Create Mapping” instruction. Data are saved into the DB Map Variable, specified in the “MapVar”. To understand the function let’s refer to Figure 4.27: as stated before, “DBConnection” identifies the opened connection; “Where” defines the retrieval condition; “Sort” specifies the retrieve order (in this ex-

the connection again.

Sending data from PLC to MySQL database

The last step to perform, is the transfer of the elaborated data to the Statwolf database. For this purpose, it has been implemented a Sysmac Studio algorithm that sends an example of a packet to the connected MySQL database. The following instructions have been used:

DB_Connect:

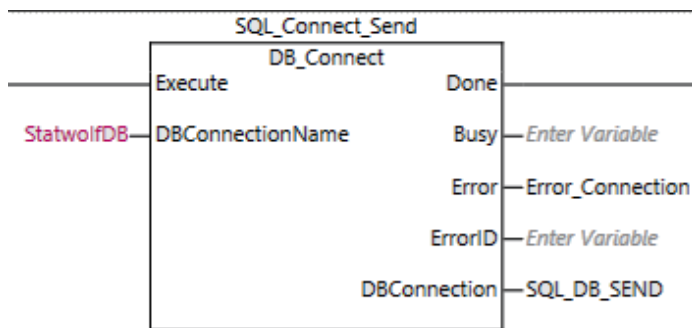


Figure 4.29: DB_Connect Sysmac studio function

The function in Figure 4.29 is the same of Figure 4.25. “StatwolfDB” is the name of the connection established with the Statwolf database and has to be equal for all the functions that want to connect with it. The name of the instance, “SQL_Connect_send” and the name of the new connection, “SQL_DB_Send”, must be unique so to avoid in-

tersections with different parts of the programs installed on the PLC.

DB_CreateMapping:

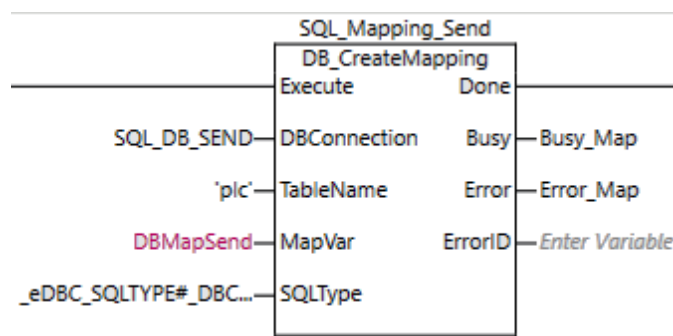


Figure 4.30: DB_CreateMapping Sysmac studio function

The function in Figure 4.30 is the same of Figure 4.26. Input and output parameters are the same, with the exception of the map variable, “DBMapSend”, that stores the parameters that have to be transferred to the database, and the SQLType variable. Since the function has to send data to the database, it uses the “INSERT” MySQL command that, in Sysmac Studio,

corresponds to the “_eDBC_SQLTYPE#_DBC_SQLTYPE_INSERT” internal variable.

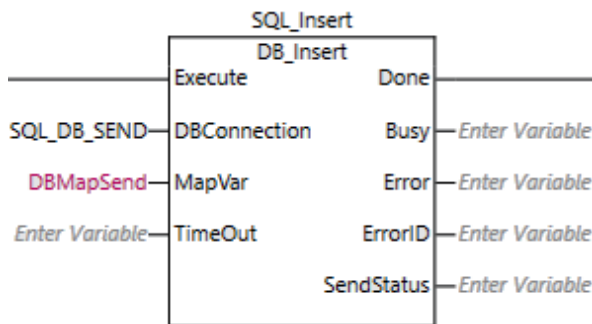
Populating the map:

After the creation of the map, its members have to be populated with the values to send. To do so, the assignment of each variable has been performed through the following structured language code:

```
DBMapSend.sensore:=999;
DBMapSend.tipo:='TEST_PLC_CNR';
DBMapSend.dataora:='2019-11-18 23:59:59';
DBMapSend.valore:=999;
```

As can be seen, the chosen values are an example of a standard packet that will be sent from the PLC.

DB_Insert:



This instruction is used to insert the values of the “DB Map” variable, specified in the “MapVar” input variable, to the table mapped by a “DB.CreateMapping” instruction. If the instruction ends without errors, on the database should appear the data stored in the “MapVar” variable. In this specific case, an example of this behaviour is

Figure 4.31: DB_Insert Sysmac studio function

represented by Figure 4.32:


sensore	 dataora	tipo	valore
999	2019-11-18 23:59:59	TEST_PLC_CNR	999
101	2019-11-18 11:02:14	PROX	21.009
101	2019-11-18 11:02:05	PROX	21.265
104	2019-11-18 10:35:08	LUCE	445
105	2019-11-18 10:35:08	TEMP	18

Figure 4.32: Example of a packet sent by the PLC to the Statwolf database

DB_Close:

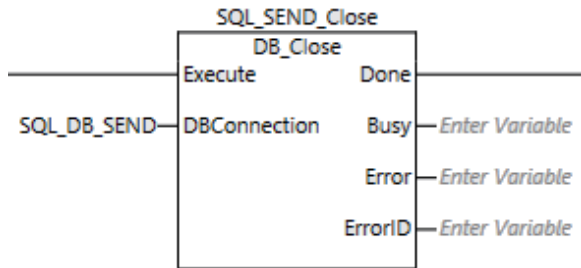


Figure 4.33: DB_Close Sysmac studio function

The instruction in Figure 4.33 is the same of Figure 4.28. The only things that change are the instance name, “SQL_SEND_Close”, and the name of the DBConnection, “SQL_DB_SEND”. After the end of this instruction is not possible to insert other data on the database until another connection is opened.

End:

After ensured the closure of the communication, the program resets all the parameters used to check the status of the processes and then waits for a signal that enables the connection again.

Chapter 5

Related Works

In this section some selected literature contributions concerned with LoRa are presented. That could reveal of interest for the ADMIN4D project.

5.1 The security vulnerabilities of LoRa

Since LoRa networks are even more deployed in the IoT environment (for example, 1.5 million of LoRa devices are currently connected throughout the Netherlands), in [8] some vulnerability tests have been carried out. For their experiments the authors used low-cost hardware, as an Arduino board, that is similar to that adopted for the thesis and that probably will be inserted on the Desamanera 3D printers. This is the reason why it is important to understand the limits of the implemented LoRa devices and their reliability.

LoRaWAN utilizes a symmetric-key cryptography to ensure the security of the transmitted data. However, the messages sent via the LoRa modulation require between 900 milliseconds and 1.2 seconds to reach the receiving device. This time window could be used from the attackers to steal data, or even break the transmission by jamming it. Radio jamming affects all the IoT transmissions, and it is not easy to avoid completely the problem. A malicious entity can send a powerful radio signal close to the LoRa device and compromise the whole data transmission. Normally it is not so simple to find the hardware to perform such attacks, but in the LoRa case the situation is different. LoRa devices suffer from coexistence issues, in fact concurrent LoRa transmissions at same frequency and spreading factor can interfere with each other. In this way, by using a simple LoRa device, an attacker can send a huge quantity of LoRa messages at a certain frequency to wipe out all the other transmissions at that frequency. In the paper the authors proved that roughly 99% of the tested transmissions were affected by this jamming approach. It is hard to

avoid this kind of attacks at all, but some countermeasures can be taken in consideration to prevent them. For example, the devices can be programmed to change transmission frequency after a certain period of time. In this way the data would be lost only on the time window in which the device has transmitted on the jammed frequency.

Another type of attack is called *replay*: replay attacks re-send or repeat a valid data transmission to the receiver. Usually they aim to fool the device by using old data from the network so to create confusion on retrieving these data. On LoRa it is not possible to do so thanks to the data encryption based on the keys of the transmitter. However, it is possible to de-synchronize the frame counters of the linked devices. This issue affects mainly the end devices that join the LoRa servers with the ABP method. Depending on the server implementation, the de-synchronization of a device, with respect to the server, could represent either a small or big problem. In many cases, when a message is received with a wrong frame counter, it is simply ignored and the server waits until the device frame counter matches again. In other cases all the transmissions coming from a wrong frame counter are rejected and both the counters have to be reset to zero. In any case, adopting a OTAA joining method prevents all this kind of problems since it is not based on the frame counter check. One more common violation that affects the majority of the devices is a physical intrusion. If a malicious entity arrives directly to the hardware of the transmitter, it can retrieve all the network keys necessary to connect to the server. In this way it could send false data, compromising all the statistics related to the device. However, the plans of the ADMIN4D project are to encapsulate the LoRa sensors on a secure place in to the printed artefacts, making difficult to access them.

5.2 Other tests on the communication range

Since the communication range tests described on this thesis were limited by the surrounding noisy environment, this section wants to give some other data regarding the distance that a LoRa connection could reach. In [11], the authors repost the results of some tests on the LoRa network in a suburban environment in Belgium. They built wearable LoRa devices that have been carried by two people positioned at different distances on a Watersports Track. This experiment is particularly useful for the ADMIN4D project, because the LoRa module RN2483, installed on the devices, is the same used by the LoRa nodes in this framework. The two people walked in opposite directions reaching a distance of 1.44 Km, with the antennas

of the devices not directed towards each other. Then, they started to move closer pointing to each other. The results can be seen in Figure 5.1: the configuration

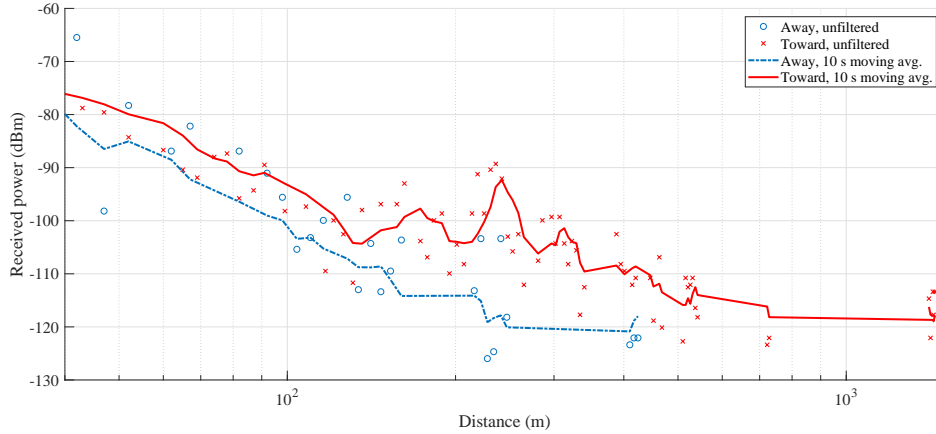


Figure 5.1: Graph showing the signal power with respect to the distance

of the devices consists in a receiver and a sender that transmit the data with a spreading factor of 12 and a coding rate of 4/5. As highlighted on the LoRa section, SF 12 gives the highest robustness to noise and enhances the quality of the communication, paying the price of a lower data rate. From the graph it can be seen that, when the two antennas are not paired, the signal is lower than the paired case and the distance between transmitter and receiver can be 700 meters at maximum. Conversely, the data transmitted while pointing each other, cover all the travelled distance demonstrating the effectiveness of LoRa. The authors wanted to put the attention on the difference between the transmission while the devices were moving and the static one. The packet loss was detected only in the first case, while, at a distance of 1.44 Km, the packets started to be detected again from the receiver that, in the meanwhile, was stopped. This phenomenon is due to the Doppler effect that affects the data transmission when the devices are moving. However, thinking about the implementation of the gateway and sensors that will be adopted on the ADMIN4D project, the Doppler effect will result negligible since the devices will be placed on a steady position. In this way, the expectation is the possibility to position the project gateways on a greater distance than the one tested on experiments already performed.

This section, in accordance with its purpose, does not want to establish a limit of the maximum range of the network, but it wants to encourage the idea of new tests, once the LoRa devices will be placed on the 3D printer. There are so many factors that can not be known a priori and that change zone by zone. The ones collected during this thesis are related to a specific environment, but they are not valid in

general.

5.3 Enhanced LoRaWAN node

The architecture of a LoRa connection is formed by end devices (EDs), gateways (GWs) and network server (NS). In ([5]), however, the authors tried to introduce a new concept: the *enhanced LoRaWAN node (e-Node)*. The idea behind the e-Node is to create a device capable to act both as ED and GW that plays the role of extender for a LoRa connection. So, when a message is received, the device has to forward it to the nearest gateway or e-Node. This new device opens a new way to implement a LoRa network. Since now, the star of stars topology considered the LoRa gateways as its centres. In this way, if a end node could not reach a gateway, one of the two devices had to be moved. Moreover, if the Network Server is not implemented on the same gateway device, a LoRa gateway has the necessity of an internet connection and, in some cases, this limitation can constitute a problem. Playing the role of a gateway, an end node can act as a message repeater allowing to reach very distant gateways. The power consumption would not be a problem since the LoRa nodes can last for months, but there are some adjustments that have to be considered. To guarantee such power dispersion, a LoRa ED has a very low data rate and can receive messages only after the sending of a message. This peculiarity forces the e-Nodes to be synchronized with the senders, although they can not receive any message.

On their paper, the authors utilized a Microchip RN2485 to implement successfully such a behaviour and are continuing the studies to make the e-Node a Proxy for other IIoT applications. On the ADMIN4D project, the e-Nodes could be a solution in the situations on which the buyers want to place an end node too far from their gateways. It is an idea, but with some tests it could reveal a valid feature on which rely the future projects.

Chapter 6

Conclusions and future directions

The aim of this work was test some of the communication networks adopted in the ADMIN4D project and to implement the ideas developed during the research period. The Microchip Lora development kit, adopted to perform the trials, resulted easy to configure and functional with respect to almost all the activities. During the connection range tests, the LCD display mounted on the gateway was very useful to gather all the statistics of the communication, facilitating their elaboration. As expected, these tests have shown a correlation between the range of the connection and the surrounding environment. Actually, the presence of different buildings on the network area introduced disturbing factors that affect the data transmission. The maximum distance at which the sensors have performed a connection with the LoRa gateway was around 250m, with the presence of only few constructions on the way. However, for some positions, even at a distance of only 130m, all the packets sent from the sensors have been lost due to the presence of obstacles as well as to the interference. Generally speaking, the obtained outcomes are encouraging, and an effective deployment of the network is expected to lead to effective results. The connection to the MySQL database, even if implemented with a remote LoRa network server, has been satisfactory and represents the basis for some future activities. The NX102-9020 CPU, thanks to its built-in interfaces, allowed to perform other connection tests with the online services provided by the project facilities. Particularly with the implementation of ad hoc algorithms, it has been possible to retrieve data both from the LoRa network server and the MySQL database. Moreover, Desamanera has just performed successful tests on the data transmissions from the PLC to the database. In this way, it has been demonstrated that it is possible to manage the Statwolf database with the Omron CPU on both sides of the connection. This fact paves the way to implement a real time behaviour in which new data can be elaborated relying on the old ones and the results can be directly uploaded on

the database.

Drawing the conclusions of the activities carried out, the protocol defined by LoRa and LoRaWAN seems to be a satisfactory choice to take for the ADMIN4D project. The amount of data already available on the Statwolf database, testifies the success and the reliability of the developed apparatus, both from the software and the hardware point of view.

The devices match perfectly the idea of Industry 4.0: they are able to communicate to each others and to cooperate to reach the same objectives. During the experimental period the batteries of the end devices have not been replaced demonstrating a sustainability that complies with the project future directions. The NX102-9020 CPU showed its potential as well as the capability to integrate itself in a connected world that used to be very distant from the PLCs industrial environment since now.

When the experiments described on this thesis have been carried out, the ADMIN4D project had just entered the experimental phase on the connections field. For this reason, the complete links between the different devices involved had not been tested yet. One of the goals to reach in the next future, is the implementation of a local LoRa Network Server. This functionality has already been tested on a CPU running a Linux based operating system, but it has not gave the expected results. By relying on an open source LoRa Server project named “ChirpStack” (“LoRaServer” until a month before this thesis), the experiment ended with a success. The problems raised up trying to establish a connection between the Microchip LoRa Gateway and the LoRa network server. Indeed, when the gateway receives a packet from a sensor, it appends to the packet the time at which it has been received but, unfortunately, the gateway time format is not compatible with the server one. Due to this problem, the server could not handle the packets received from the gateway: indeed, the decryption process ended with a “time parse” error, making impossible to store the incoming data. To solve this problem, it has been decided to change gateway and to implement it together with the ChirpStack server on a device suggested by the server project site. The idea is to use a Raspberry pi 3 mounting a iC880A LoRaWAN Concentrator, that, together, should act both as LoRa gateway and server. In this way, the incompatibility problems between GW and NS should disappear. Moreover, the advantage to use a local LoRa server is the possibility to have full access to the logs of the server and to its configuration. Thus, the connected devices statistics can be studied to perform other experiments in the future. Another important aspect is the improvement of the security of the entire network since less components are connected to the Internet.

Since all the tests were thought in relation to a future real implementation, the next steps aim to mount the devices on a real 3D printer and to repeat the procedures investigated in this thesis in a working environment. Moreover, new varieties of sensors will be considered, in order to better understand the parameters that influence the printing process more evidently. In this direction, the adoption of new humidity and light sensors based on the LoRa technology has already been planned. New tests will be carried out to better check the communication range.

On the PLC side, all the developed programs will be transferred on the printer already available in Desamanera and the first artefact prototypes will be produced. It will be also necessary to gather information about motors and motion axis in order to better analyze the processes. These data, together with those from the LoRa sensors, will give the possibility to create a model of the ideal functioning of the printer that could be used to detect anomalies or dysfunctions. Unfortunately, the constraints on the transfer data rate, given by the LoRa motes firmware, represent a bottle neck for the model creation. One solution is to operate directly on the firmware of the end devices to understand if it is possible to change the rate, while respecting the duty cycle limits given by LoRa.

In a further, upcoming phase of the project, program optimization will start. The intention is to make the algorithms self-sufficient, so to reduce the user's intervention at the minimum. For example, there exists the possibility to send commands to the LoRa sensors via serial communication using programs as "PuTTY" [34]. In this way, scripts that retrieve the keys from the gateway and set them on the sensors via USB link can be implemented. This feature brings with it the opportunity to connect the sensors directly to the gateway and to download automatically the communication keys on them. In this way, the portability of this project would certainly be enhanced, and Desamanera could spare the time needed for the first set up on the buyer's company. Another important feature that should be added to the transmitting/receiving devices, is concerned with outputs representing the device status. The idea is to use RGB LEDs that turn green when the transmission is active, red when a problem arises on the connection with other devices. In this way the detection of a network failure would result much easier.

Then, the algorithms implemented will be further studied to enhance their robustness. At the moment, without the possibility to try them on the 3D printer environment, some problems could not have been considered yet. After the realization of the first prototypes, the programs will be updated to handle all the possible failure situations and to respond in the right way. On the automation field, since the running CPUs are connected directly to mechanical parts, it is crucial to develop programs

that take in consideration as many scenarios as possible. Another suggestion that could be taken in consideration is the implementation of a mobile application that monitors the data as it is typical in the IIoT context. In this way the status of the machines will be always accessible and the app could send warning notifications in case of necessity.

Bibliography

- [1] Michele Luvisotto, Federico Tramarin, Lorenzo Vangelista, and Stefano Vitturi, *On the Use of LoRaWAN for Indoor Industrial IoT Applications*. Wiley, Padova, 2018.
- [2] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund, *Using LoRa for industrial wireless networks*. 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), Trondheim, 2017, pp. 1-4.
- [3] L. Tessaro, C. Raffaldi, M. Rossi, and D. Brunelli, *Lightweight Synchronization Algorithm with Self-Calibration for Industrial LORA Sensor Networks*. 2018 Workshop on Metrology for Industry 4.0 and IoT, Brescia, 2018, pp. 259-263.
- [4] D. F. Carvalho, P. Ferrari, A. Flammini, and E. Sisinni, *A Test Bench for Evaluating Communication Delays in LoRaWAN Applications*. 2018 Workshop on Metrology for Industry 4.0 and IoT, Brescia, 2018, pp. 248-253.
- [5] E. Sisinni, D. F. Carvalho, P. Ferrari, A. Flammini, D. R. C. Silva, and I. M. D. Da Silva, *Enhanced flexible LoRaWAN node for industrial IoT*. 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, 2018, pp. 1-4.
- [6] Alireza Zourmand, Andrew Lai Kun Hing, Chan Wai Hung, and Mohammad AbdulRehman, *Internet of Things (IoT) using LoRa technology*. 2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS 2019), 29 June 2019, Selangor, Malaysia
- [7] Hyeonwoo Cho and Sang Woo Kim, *Mobile Robot Localization Using Biased Chirp-Spread-Spectrum Ranging*. IEEE Transactions on Industrial Electronics, VOL. 57, NO. 8, August 2010
- [8] Emekcan Aras, Gowri Sankar Ramachandran, Piers Lawrence, and Danny Hughes, *Exploring the Security Vulnerabilities of LoRa*. 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), June 2017

- [9] Aloys Augustin, Thomas Heide Clausen, Jiazi Yi, and William Mark Townsley, *A Study of LoRa: Long Range & Low Power Networks for the Internet of Things*. *Sensors* 16(9):1466, October 2016
- [10] Ming Xue, and Changjun Zhu *The Socket Programming and Software Design for Communication Based on Client/Server*. 2009 Pacific-Asia Conference on Circuits, Communications and Systems, 16-17 May 2009
- [11] Patrick Van Torre, Thomas Ameloot and Hendrik Rogier, *Long-range body-to-body LoRa link at 868 MHz*. 13th European Conference on Antennas and Propagation (EuCAP 2019)
- [12] Microchip Technology Inc., *LoRa Technology Evaluation Suite User's Guide*. 2016
- [13] Microchip Technology Inc., *LoRa Technology Gateway User's Guide*. 2016
- [14] Microchip Technology Inc., *LoRaWAN Library Plug-in for MPLAB Code Configurator User's Guide*. 2017
- [15] Microchip Technology Inc., *RN2483 LoRa™ Technology Module Command Reference User's Guide*. 2015
- [16] Omron, *NJ_NX-series CPU Unit Built-in EtherNet/IP Port User's Manual*.
- [17] Omron, *NJ_NX-series Instructions Reference Manual*.
- [18] Omron, *NJ_NX-series Troubleshooting Manual*.
- [19] Omron, *NJ_NX-series Database Manual*.
- [20] Security in LoRaWAN Applications,
<https://smartmakers.io/en/security-in-lorawan-applications/>,
last seen on 21 September 2019
- [21] LoRa™ Modulation Basics,
<https://www.semtech.com/uploads/documents/an1200.22.pdf>,
last seen on 2 October 2019
- [22] LoRa,
<https://lora.readthedocs.io/en/latest/>,
last seen on 10 October 2019

- [23] SYMBOL, SPREADING FACTOR & CHIP ,
https://www.mobilefish.com/download/lora/lora_part13.pdf,
last seen on 10 October 2019
- [24] LoRa- (Long Range) Network and Protocol Architecture with Its Frame Structure ,
<http://www.techplayon.com/lora-long-range-network-architecture-protocol-architecture-and-frame-formats/>,
last seen on 10 October 2019
- [25] Desamanera,
<https://www.desamanera.com/>,
last seen on 19 October 2019
- [26] Statwolf,
<https://www.statwolf.com/>,
last seen on 23 October 2019
- [27] DataVeneta,
<http://www.dataveneta.it/>,
last seen on 23 October 2019
- [28] The Things Network,
<https://www.thethingsnetwork.org/>,
last seen on 15 November 2019
- [29] Setting up your own gateway and endpoint with Microchip's LoRa Technology Evaluation Kit,
<https://www.thethingsnetwork.org/labs/story/setting-up-your-own-gateway-and-endpoint-with-microchips-lora-technology-evaluation-kit>,
last seen on 20 October 2019
- [30] TTN Storage Integration,
<https://www.thethingsnetwork.org/docs/applications/storage/>,
last seen on 23 October 2019
- [31] NX102-□□□□□,
<http://www.ia.omron.com/products/family/3705/>,
last seen on 2 November 2019

- [32] Sysmac Studio,
<https://industrial.omron.it/it/products/sysmac-studio>,
last seen on 3 November 2019
- [33] ChirpStack LoRa server,
<https://www.chirpstack.io/>,
last seen on 8 November 2019
- [34] Download PuTTY,
<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>,
last seen on 8 November 2019