# POLITECNICO DI TORINO

**Master's Degree in Electronic Engineering**


Politecnico
di Torino

**Master's Degree Thesis**

# Long-range low-power electronic system for precision agriculture

**Supervisors**

**Prof. Danilo DEMARCHI**

**Ph.D. Umberto GARLANDO**

**Ph.D. Alessandro SANGINARIO**

**Candidate**

**Mattia BAREZZI**

**July 2021**

# Summary

This thesis is involved in the regional project WAPPFRUIT - Smart technologies applied to water management in fruit growing for rural development in the Piedmont region.
As part of the work, it is required to design an embedded system to collect environmental data (temperature, humidity, pressure, etc.) and soil data to have punctual knowledge of water needs at a certain point of the land. Each of these systems is part (known as "nodes") of a LoRa network to characterize fields of orchards.

The first chapter is related to an **introduction** to the project, what has involved players and some theoretical concepts useful to know the reasons to have chosen the LoRa protocol and the working principles of used soil sensors.

The following chapter is a substantial **investigation** about electronic and non-electronic components involved in the system and it would demonstrate the requirements of the project, what are the edges that all nodes should satisfy starting from the choice of the suitable microcontroller as the core of all actions of the embedded system and ending to the range of options of some off-board elements as a solar radiation shield and a proper case to protect all nodes from water, dust, direct sunlight and cold. In conclusion, an estimation of the theoretical consumption in the main modes of the system will be investigated.

As a consequence of the investigation process, an implementation step, in Chapter 3, is performed starting from the **firmware** contained in the Murata module where the microcontroller is placed. Initially, the developed proof of concept on a development board will be presented, successively transferred on a custom board. All main pieces of codes developed in the LoRa middleware are discussed both in the main procedure and involved library.

After a code development, Chapter 4 is dedicated to the **hardware implementation**, where it will be presented the realized schematics and the PCB layout of the electronic board. In addition of it, it will be clarified some concepts about RF

design, testability and usability. Finally, a Bill Of Material (BOM) is presented to find the product's value from an economic point of view, together with some considerations about manufacturing and some fancy 3D views of the final system.

Chapter 5 will present the **experimental results** from practical and performance perspectives both in standby and runtime phases.

Finally, Chapter 6 is related to the **conclusion**, and it will consider some thoughts about the developed LoRa node and clarify some reflections about the future development of the triennial project.

On a final note, keywords will be highlighted in **bold** to keep the focus on the concept. Instead, *italic* is used when a pin of a determined integrated circuit is discussed. A list of used acronyms will be presented to help the reader to avoid confusion about the terminology.

# Acknowledgements

*"Ad Anna e Alide"*

Questa tesi magistrale conclude il mio lungo percorso accademico in parte svolto a Parma e in parte a Torino.
Tanti sono i ringraziamenti che vorrei fare e sono sicuro che nel battere questi tasti dimenticherò tante persone che mi hanno aiutato a maturare, crescere e a diventare un uomo migliore.
Ai miei genitori Anna e Massimo per aver creduto e avermi stimolato a puntare sempre al massimo delle proprie possibilità aiutandomi in ogni situazione, anche quelle meno favorevoli. Il supporto che ho avuto in questi anni da fuorisede rimarrà impagabile.
A mio fratello Paolo per tutti i momenti insieme, passati e futuri, per tutti i consigli, per tutti i videogiochi. Il legame che ci unisce durerà per sempre.
Ai miei nonni Vilma e Alide per il calore e la semplicità con cui mi ricordate in qualsiasi momento quali sono le mie origini.
Alla zia Mara e allo zio Giuliano, la mia seconda famiglia. Non sarei la persona che sono se non ci foste voi. La presenza costante nella mia vita ha forgiato il mio carattere.
Alla zia Cristina e lo zio Stefano per essere stati un grandissimo supporto in tutti questi anni "lontano da casa". Senza il vostro aiuto, non avrei completato questo percorso qui a Torino.
Ai miei "pochi ma buoni" amici di Busseto e di Parma per le innumerevoli serate passate in compagnia. La mia scelta di andare a studiare in una città lontana non ha scalfito minimamente l'amicizia che ci lega. Le cene, le pizze, le feste, le serate a Mario Kart, i Lucca Comics fanno parte della mia persona.
Ai miei coinquilini Michele e Angelo per aver condiviso tantissimi momenti in compagnia in quella che è stata la mia prima vera esperienza da solo. La sola vostra presenza ha allargato la mia conoscenza sugli argomenti più disparati, dalla politica, al lavoro alla filosofia. Non avrei potuto incontrare persone migliori nel mio percorso.
Ai miei amici "torinesi" Stefania, Silvia, Lorenzo, Marco, Michele, Gianmarco e

Andrea. Anche qui le serate si sono sprecate! mi avete fatto conoscere la bellezza di questa città non come un turista ma come un vero cittadino sabaudo.
Infine volevo ringraziare il gruppo del MiNES con cui ho lavorato in questi mesi: penso di non aver mai visto un gruppo tanto coeso e produttivo in vita mia. Ogni singola persona al suo interno mette una passione, un impegno e una disponibilità fuori dal comune con il solo fine di diffondere la propria conoscenza agli altri in un clima che dovrebbe essere lo standard accademico.
Desidero ringraziare il prof. Demarchi per avermi accolto all'interno del team di lavoro e ai miei due correlatori "Umbi" e "Sangi" per la dedizione e il prezioso aiuto con cui abbiamo costruito questa tesi. Volevo anche ringraziare Stefano per tutti i consigli su come massimizzare le prestazioni dello stadio in radiofrequenza. E a tutti gli altri colleghi tra cui Stefano, Andrea B., Fabiana, Andrea P. per avermi supportato e, soprattutto, sopportato durante questi mesi di pandemia.

# Table of Contents

**List of Tables** IX

**List of Figures** X

**Acronyms** XIV

**1 Introduction** 1
- 1.1 The Concept of LPWAN . . . 3
  - 1.1.1 Long Range (LoRa) . . . 6
- 1.2 Soil Moisture Sensors . . . 10
  - 1.2.1 Soil Volumetric Water Content (VWC) . . . 14
  - 1.2.2 Soil Water Potential . . . 17
  - 1.2.3 Soil-Water Retention Curve . . . 20
- 1.3 State of Art of Agriculture Data Loggers . . . 21

**2 Investigation** 24
- 2.1 Requirements . . . 24
- 2.2 Outline . . . 26
- 2.3 Components . . . 26
  - 2.3.1 Ultra Low-Power Microcontroller and Transducer LoRa . . . 26
  - 2.3.2 Capacitors, Resistors and Inductors . . . 31
  - 2.3.3 Battery . . . 32
  - 2.3.4 Inversion Polarity Protection . . . 35
  - 2.3.5 Battery Monitoring Stage . . . 36
  - 2.3.6 3.3 LDO Regulator . . . 39
  - 2.3.7 5V Boost Regulator . . . 42
  - 2.3.8 Load Switches . . . 43
  - 2.3.9 LoRa Antenna . . . 44
  - 2.3.10 Light Emitting Diodes (LEDs) . . . 45
  - 2.3.11 Switch for TEROSxx Channels . . . 46
  - 2.3.12 TEROS Sensors . . . 49

2.3.13 Case and Glands . . . 52
2.3.14 Headers e SWD Interface . . . 53
2.4 Designed Consumption . . . 55

**3 Firmware Implementation** 62
3.1 Introduction . . . 62
3.2 Discovery Board . . . 65
3.3 Initialization . . . 65
3.4 Loop . . . 68
3.5 LoRa Packaging (Tx) and Parsing (Rx) . . . 70
3.6 MAX4562 Library . . . 75
3.7 TEROS21 Library . . . 76
3.8 Python Code . . . 79
3.9 Decode Payload on TTN Platform . . . 80

**4 Hardware Implementation** 82
4.1 Schematics . . . 82
4.2 PCB Layout . . . 88
4.2.1 Murata Wireless Module . . . 89
4.2.2 LoRa Section . . . 90
4.2.3 Test Points and Jumpers (Debug Features) . . . 92
4.2.4 User Experience . . . 94
4.2.5 I2C Interface . . . 95
4.3 Bill of Materials (BOM) . . . 97
4.4 3D models and Variations . . . 98
4.5 Manufacturing . . . 99

**5 Experimental Results** 105
5.1 PCB Assembly . . . 105
5.2 Backend Data . . . 107
5.3 Power Consumption . . . 108

**6 Conclusion and Future Perspective** 112

**Bibliography** 114

# List of Tables


1.1 Overview of LPWAN technologies in Europe. . . . . 5
1.2 Data rate classification in LoRa technology considering TTN (The Things Network) rules. . . . . 8
1.3 Mineral particles classification. . . . . 12
1.4 Infiltration rate classification. . . . . 12
1.5 Typical values of relative dielectric permittivity. . . . . 16
1.6 Soil water potential values example (Source [7]). . . . . 18
1.7 Comparison among commercial data loggers. . . . . 23

2.1 Comparison among approaches considering only available products compatible with the European frequency (868 MHz). . . . . 30
2.2 Comparison between non-rechargeable cylindrical batteries as referred in Book[9]. . . . . 34
2.3 Estimated standby current consumption of LoRa end node. . . . . 56
2.4 Worst-case consumptions. . . . . 59
2.5 Computation of realistic lifetime of the LoRa end node at various temperatures. . . . . 61

4.1 Description of layers in the PCB layout. . . . . 88
4.2 Bill of materials on board (PCB cost not included). . . . . 97
4.3 Total bill of materials. . . . . 98

5.1 Comparison between nominal consumptions and experimental consumptions. . . . . 110

# List of Figures


1.1 The sixth sustainable development goal: clean water and sanitation. 1
1.2 LoRa chirp modulation (Source [4]). . . . . 6
1.3 Timing of an end device LoRa class-A (Source [4]). . . . . 9
1.4 Full LoRa protocol stack (Source [4]). . . . . 9
1.5 The composition of soil (Source [5]). . . . . 11
1.6 Some remarkable conditions of a generic soil (Source [5]). . . . . 14
1.8 An example of soil-water retention curve (Source [7]). . . . . 21

2.1 Overview of the designed system architecture. . . . . 25
2.2 LoRa plant monitoring end node overview. . . . . 27
2.3 Three illustrated approaches for a development of a LoRa system. . 29
2.4 Block diagram of the Murata module (Source [8]). . . . . 29
2.5 Different sizes of SMD ceramic capacitors. . . . . 31
2.6 The non-rechargeable LiSOCl2 battery in cylindrical AA size. . . . 33
2.7 Typical discharge profiles at 20 °C varying the resistance load (Source [10]). . . . . 35
2.8 LM66100 protection circuit as illustrated in the TI datasheet[11]. . 36
2.9 Screenshot from schematic related to the power management where it was presented a simple battery voltage detector stage (Source [12]). 37
2.10 Typical circuit configuration of BQ35100 to monitor the state of charge of a battery (Source [12]). . . . . 38
2.11 Example of power activity and when it is needed to activate *GE* pin (Source [12]). . . . . 39
2.12 Concept of dropout voltage (Source [13]). . . . . 40
2.13 Quiescent current of LDO regulator (Source [13]). . . . . 41
2.14 Graph presents all ST LDO regulators comparing their maximum current and their quiescent current (Source here). . . . . 42
2.15 Block diagram of SiP32431[16] (courtesy of Vishay). . . . . 43
2.16 Comparison between SMA male and U.FL female connectors. . . . 45
2.17 Preliminary version of the schematic section related to the switch among TEROS sensors. . . . . 46

2.18 MAX4562EEE pin configuration. . . . . 47
2.19 MAX4562EEE possible implementation. . . . . 48
2.20 MAX4562EEE I2C data format. . . . . 49
2.21 Example of values obtained in a soil from capacitive soil matric sensor (Source [6]). . . . . 49
2.22 TEROS sensors showcase. . . . . 50
2.23 Voltage characteristics of microcontroller STM32L0 series. . . . . 51
2.24 Data line DDI serial timing of TEROSxx (Sources [19], [20]). . . . . 51
2.25 DDI response of TEROS21 (Source [20]). . . . . 52
2.26 DDI response of TEROS11-12 (Source [19]). . . . . 52
2.27 A quick sketch of the idea of PCB inside RP1210BF IP65 case. . . . 54
2.28 SWD interface connections as suggested in Murata datasheet[8]. . . 55
2.29 LoRa end node runtime consumption graph (not in scale). . . . . 58
2.30 Current capacity versus discharged current in function of the temperature (Source [10]). . . . . 60

3.1 LoRa end node packed payload. . . . . 64
3.2 B-L072Z-LRWAN1 LoRa/Sigfox Discovery kit. . . . . 65
3.3 LoRa end node packed payload. . . . . 71

4.2 LoRa module schematic sheet. . . . . 84
4.3 Power stage schematic sheet. . . . . 85
4.4 Antenna and peripherals schematic sheet. . . . . 86
4.5 TEROS sensors stage schematic sheet. . . . . 87
4.6 Screenshot of VDD layer that shows three power islands to lower as possible the power resistive paths. . . . . 89
4.7 Screenshot of the first layer (Top layer) showing the Murata module footprint (on the left) and RF path (on the right). . . . . 90
4.8 Screenshot of the performed simulation considering a width equal to 0.438 mm. . . . . 91
4.9 Screenshot highlights where are placed Test Points (TPx), Jumpers (JPx) and Solder Bridges (SBx). . . . . 93
4.10 3D representation of the bottom layer of PCB, useful to show the following user experience on it. . . . . 95
4.11 Screenshot from oscilloscope without and with external pull-up resistances. . . . . 96
4.12 Screenshot of bottom view 3D model for both variations. . . . . 98
4.13 The average and standard deviation of critical parameters . . . . . 99
4.14 MD srl 4-layers PCB stack-up (Courtesy of `https://www.mdsrl.it/`).100
4.15 Top layer graphical gerber file. . . . . 101
4.16 GND layer graphical gerber file. . . . . 102

4.17 VCC layer graphical gerber file. . . . . 103
4.18 Bottom layer graphical gerber file. . . . . 104

5.1 Photo of bottom (on the left) and top side (on the right) of the fabricated printed circuit board. . . . . 105
5.2 Photo before (on the left) and after (on the right) Murata assembly. 106
5.3 Photo of demo assembled PCB. . . . . 106
5.4 Screenshot from The Things Network live data console. . . . . 107
5.5 Photo of the Agilent E3631A power supply. . . . . 108
5.6 Photo of the Keithley DMM7510 digital multimeter when end node is in standby mode. . . . . 109
5.7 Graph of current consumption profile in all modes using Keithley DMM7510 digital multimeter. . . . . 111
5.8 Graph of current consumption profile in standby using Keithley DMM7510 digital multimeter. . . . . 111

# Acronyms

**IoT**
: Internet of Things

**LoRa**
: Long Range protocol - physical layer

**LoRaWAN**
: Long Range Wide Area Network - networking layers

**SDI-12**
: Serial Digital Interface at 1200 baud

**I2C**
: Inter-Integrated Circuit

**UART**
: Universal Asynchronous Receiver-Transmitter

**TTN**
: The Things Network - a LoRa cloud platform

# Chapter 1

# Introduction

Careful management of earth resources is the main key of the future of humankind. For this reason, in 2015, the United nation General Assembly, one of the six principal organs of the United Nations (UN), established the Sustainable Development Goals (SDGs) that are a collection of 17 interlinked global goals designed to assure a more sustainable future for all over the world and are intended to be achieved by the year 2030.


**Figure 1.1:** The sixth sustainable development goal: clean water and sanitation.

One of these SDGs is about "clean water and sanitation for all" and, by United Nations General Assembly, the official wording states: "Ensure availability and sustainable management of water and sanitation for all."
To assure improvement by this side, whatever progress toward this target will be measured by using eleven indicators that evaluate six targets.

- Safe and affordable drinking water (Target 6.1);
- End open defecation and provide access to sanitation and hygiene (Target 6.2);
- Improve water quality, wastewater treatment and safe reuse (Target 6.3);
- Increase water-use efficiency and ensure freshwater supplies (Target 6.4);

- Implement IWRM (Integrated Water Resources Management) at all levels (Target 6.5);
- Protect and restore water-related ecosystems (Target 6.6).

As it could be seen, these targets are all put in contact among them to reach as close as possible the SDG 6, also called Global Rule 6, related to water matter.
But for all country these targets have a different meaning: for example, in Italy, as stated in [1], the total water footprint of consumption is 25% higher than the European average, approximately 132 million $\frac{m^3}{year}$ of water, so 6,309 liters of water per capita per day, including the quantities of water resulting from imported products.
Agriculture is the most thirsty economic sector with the major percentage of the water footprint of consumption (85%).
In Belpaese, the water availability is abundant by nature, so the first two Targets are rather well satisfied, but there is a lot of work to preserve and to use these resources frugally with farsightedness. Proper use of water in agricultural lands takes on great importance.
Moreover, in the last years, the concept of precise agriculture has become dominant and in this concept falls back the regional project WAPPFRUIT - Smart technologies applied to water management in fruit growing for rural development in the Piedmont region.
As stated in [2], the main goal of WAPPFRUIT is the innovation of farms by means of cutting-edge technologies that will let the correct definition of the water requirement and the complete automation of the micro-irrigation system. Through sensors that measure matric water potential, the plants' water requirements will be indirectly identified. Sensor data will be read automatically by a control unit, and an algorithm will activate a localized irrigation system when needed. All data will be remotely available by means of a web interface and a smartphone application.
To perform this task in a practical way, the activities of this project will lead to the realization of a sensor system to be buried in the soil. Such wireless and low-power sensors will transmit information about soil water content and potential. Following an algorithm developed in the project, the control unit will activate or not the irrigation in those sectors where the water is lacking.
Moreover, it is possible to say that the land is water-on-demand because a typical automatic irrigation system is clocked by a timer that, at a fixed span of time, switches on irrigators and, after a while, switches off themselves.
In this way, it is possible to preserve as much water as possible respect traditional techniques as manual irrigation or using a timed automatic irrigation system.
This thesis work will present the first stage of the three-year project, where a measurement campaign will be performed in lands involved in the project to get information about the type of soil of orchards and where, successively, will be

developed an algorithm to predict water needs of these plants. Moreover, an electronic embedded system will be designed to perform this task.
A distributed sensor system should be engineered in such a way as to sample data changes in plant needs at every point of a spatial grid with a reasonable resolution, and possibly, the system should provide information about the actual situation at whatever moment remotely.
In this context, the system is Internet of Things (IoT) compliant: it describes the network of physical objects (also called "things") that are embedded systems with sensors, software, firmware and their communication ways to connect and exchange data with other devices and systems over the internet.
In this work, the LoRa gateway, the stakeholder that gathers data from all nodes, is the point where the local LoRa network is connected to the internet. Furthermore, it will be considered a full-fledged IoT system.
In the following sections, it will be shown some theoretical concepts briefly about the selected communication protocol (LoRa), the working principles of involved soil sensors and comparison with respect to the state of the art of data loggers to acquire agricultural data.

## 1.1 The Concept of LPWAN

The Low Power Wide Area Network (LPWAN) concept is, basically, a wide number of devices connected among them considering the new paradigm to consume a very little energy, in the order of $\mu$watts in standby and milliwatts for a few time when something has to transmit data.
Here, it is where IoT (Internet of Things) arises as strictly connected with LPWAN. The paper [3] explains a comparative study among LPWAN technologies in this context.
Agricultural IoT applications have specific requirements such as long-range, low data rate, low energy consumption and cost-effectiveness. And well-developed short-range technologies as Bluetooth or Zigbee are not compliant with these scenarios that require long-range transmission. On the other hand, technologies based on cellular communication as 3G or LTE provide good coverage on well-established base stations at the expense of large device energy (in the order of hundreds of mA to handle a complex protocol).
The leading emergent technologies under this concept are Sigfox, LoRa and NB-IoT.
In few words, **Sigfox** is a protocol where end devices are connected with proprietary base stations using binary phase-shift keying (BPSK) modulation in an ultra-narrow band (100 Hz) in unlicensed sub-GHz ISM band carrier as 868 MHz in Europe and 915 MHz in North America. This modulation allows a very low power consumption, high receiver sensitivity and low-cost antenna at the expense of a very low maximum

throughput (100 bps).
**LoRa** is the physical layer of a technology that modulates signals in an unlicensed sub-GHz ISM band in the same carrier (868 MHz in Europe) of Sigfox using a proprietary spread spectrum technique. This modulation spreads the narrow-band signal over a wider channel bandwidth (125 kHz or 250 kHz), resulting in low noise levels and high interference resilience.
Sharing the same carrier frequency, typically, devices manufactured for one protocol are compliant with the other one. An example is the B-L072Z-LRWAN1 Discovery kit by ST Microelectronics, where one single board is provided to develop a Sigfox node or a LoRa node. The only thing to do is program its firmware with the proper stack (starting from X-CUBE-SFOX package for Sigfox protocol or I-CUBE-LRWAN package for LoRa protocol).
**NB-IoT** is a Narrow Band IoT technology founded by the 3GPP consortium and that coexists with GSM (Global System for Mobile communication) and LTE (Long-Term Evolution) under licensed frequency bands (700 MHz, 800 MHz, 900 MHz). To perform this coexistence, NB-IoT occupies a frequency bandwidth of 200 kHz, which corresponds to one resource block in GSM and LTE transmission. To be IoT compliant, NB-IoT is based on LTE protocol reducing its functionalities to the minimum and enhance only those are required for IoT applications as avoiding features as broadcasting backend system to those end device, measurements to monitor the channel quality, carrier aggregation and dual connectivity. In this way, the energy consumption can be reduced with respect to standard radio modules used, such as smartphones.
B-L462E-CELL1 by ST Microelectronics is an example of a development board where a NB-IoT device could be developed.
A proper package (X-CUBE-CELLULAR) has been designed to embed the NB-IoT stack.
All significant parameters are inserted in Table 1.1 as an overview of LPWAN technologies.

There are only some of the features from each of these IoT technologies, but it is helpful to analyze at-first-sight which protocol should fit in the best way in a specific application. For example, if the application should not suffer from interference or fading, NB-IoT guarantees the quality of service. If the application should be insensitive to the latency and do not have a large amount of data to send, Sigfox or class-A LoRa are better choices. In addition, the deployment of NB-IoT is limited to LTE base stations, so it is not suitable for rural regions that do not benefit from LTE coverage when Sigfox and LoRa, once that a provider has installed few base stations, provide good coverage.
Finally, one of the most important features, when it is needed to handle a quite complex data payload containing information from various sensors, is the maximum payload length. Sigfox is the best in terms of energy-effective, but it has a very

| Feature | LoRa (in LoRaWAN Protocol) | Sigfox | NB-IoT |
|---|---|---|---|
| Modulation | CSS | BPSK | QPSK |
| Frequency | Unlicensed ISM bands (868 MHz) | Unlicensed ISM bands (868 MHz) | Licensed LTE bands (700 MHz, 800 MHz, 900 MHz) |
| Bandwidth | 125 kHz, 250 kHz, 500 kHz | 100 Hz | 200 kHz |
| Maximum Data Rate | 50 kbps | 100 bps | 200 kbps |
| Maximum Payload Length | 222 bytes (The Things Network, a LoRa provider, policy access) | 12 bytes (in uplink), 8 bytes (in downlink) | 1600 bytes |
| Range | 5 km (urban), 20 km (rural) | 10 km (urban), 40 km (rural) | 1 km (urban), 10 km (rural) |
| Low Latency | Only class-C | No | Yes |
| Quality of Service | No | No | Yes |

**Table 1.1:** Overview of LPWAN technologies in Europe.

little useful payload. Moreover, if you would send a packet of 40-50 bytes, you have to send sub-packets of it, also considering the maximum messages per day (140 messages/day in Sigfox). This is the main drawback of this technology. On the other hand, LoRa has a more reasonable payload length that guarantees more flexibility for whatever usage. If it is not sufficient, the last choice is to employ NB-IoT at the expense of more consumption, more costs and less range.
In agriculture, a long battery lifetime of the end device is required. Many physical quantities as temperature, humidity, plant variables (soil temperature, water content, soil potential), or leaf variables are usually investigated by one node. Typically, devices update sensed data a few times per hour because of appreciable variations are in that order of magnitude. Thus, LoRa is the superior technology in the near future for this specific application.

### 1.1.1 Long Range (LoRa)

LoRa technology is one of the Low Power Wide Area Network (LPWAN) technologies available on the market.
LoRa uses frequency bands radio in the sub-gigahertz interval without royalty. For each earth zone, there are some carrier waves available: 169 MHz, 433 MHz, 868 MHz in Europe, 915 MHz in North America.
The latter two ones there are the most common in the proper region due to the fact they are less crowded (for example, 433 MHz is used in other radio applications). Moreover, in Italy, it is used 868 MHz.
The modulation technique is Chirp Spread Spectrum (CSS), where each pulse is modulated in frequency linearly in wideband to code the information. So, in the time domain, it is a sinusoidal signal where the frequency goes up or down in time. They are called, respectively, upchirp, where the frequency increases linearly in time, and downchirp, where the frequency decreases linearly in time. They are shown in Figure 1.2.

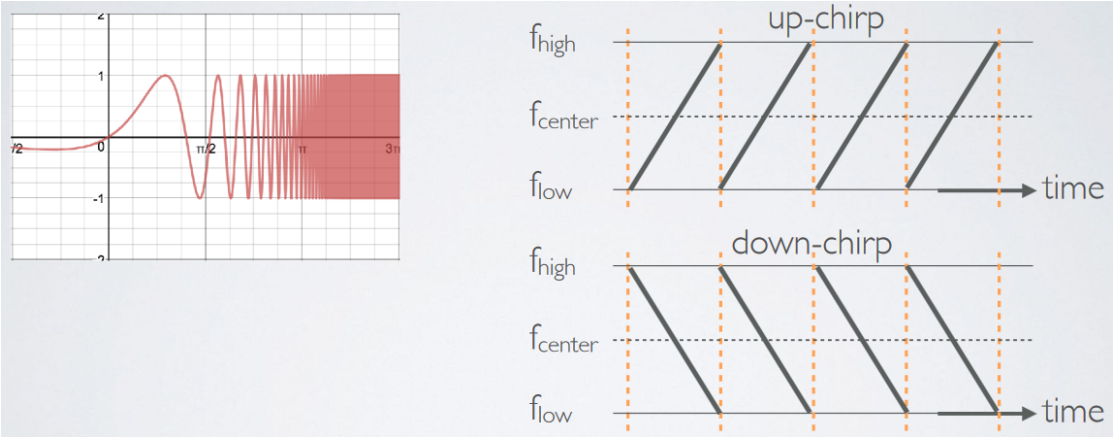



**Figure 1.2:** LoRa chirp modulation (Source [4]).

This kind of modulation allows reaching long-range transmission in the order of 2-5 km in urban regions where some obstacles are present and 10 km in the rural regions where the conditions are almost ideal (as line-of-sight), consuming a very little quantity of energy.
This technology can be employed for end-to-end communication where two devices could exchange data without any supporting appliances or exchange data from some devices, called **nodes**, to internet thanks to a "collector device" called **gateway** that, in practice, gather data from all LoRa devices in its radius of action and send these data to a LoRa provider via LAN or WLAN. All of this is possible because LoRa owns a proper communication protocol called **LoRaWAN** in such a way to

be compliant with a simple ISO/OSI model and allows reliable communication in both directions (end-device -> gateway and gateway -> end-device).
Let's see the fundamental concepts about LoRa: a radio protocol can be easily classified for bit rate. In this case, a first valuable relation is the desired bit rate with respect to the modulation variables called $R_b$.

$$R_b = SF \cdot \frac{1}{\frac{2^{SF}}{BW}} bits/sec$$

where $BW$ is the modulation bandwidth (Hz) and $SF$ is the spreading factor (from 7 to 12).
The **modulation bandwidth** is the interval of frequencies where the signal is upchirped or downchirped centered in the carrier wave (as 868 MHz).
We have three possibilities in LoRa technology: 125 kHz, 250 kHz and 500 kHz (uncommon).
The other parameter is the **spreading factor**: as said before, LoRa uses CSS to code digital signals using chirp pulses modulated in frequency linearly in wide band to transmit. Moreover, it uses a single chirp, also called symbol, to encode a certain number of raw bits. For the sake of completeness, the symbol (or chirp) has $2^{SF}$ values, so, if the spreading factor is equal to 7, we have a range from 0 to 127 of decimal data that they can be encoded.
This number is defined by this spreading factor. And LoRa employs spreading factor from 7 to 12. This means that a spreading factor equal to 7 can be expressed as "7 bits encoded in a single chirp". Moreover, lower spreading factors mean faster communications, so having fixed time slots, more chirps are employed to code a certain amount of data (bits). Fewer values mean also fewer airtime on the transmission medium.
On the contrary, higher values mean a high spreading factor (e.g., 12) so, fewer chirps are needed to code the payload. So, fixing a certain slot of time, higher values means also higher airtime on the transmission medium.
Each step, increasing the spreading factor by one, doubles the airtime and halves the available bit rate to send the same quantity of data. In the end, these parameters can be collapsed in the concept of Data Rate that represents all combinations of these variables. In Table 1.2 are reported all typical values of data rate for EU868.

As shown, LoRa uses six spreading factors (SF7 to SF12) to adapt the data rate and range tradeoff. Higher spreading factor allows longer range at the expense of lower data rate, and vice versa.
In addition to these rules, each LoRa provider provides some kind of fair access policy in such a way that all LoRa nodes can send their data: for this reason, each end device can send data with a 30 seconds uplink airtime, per 24 hours, per device and at most 10 downlink messages per 24 hours, including the ACKs for confirmed uplinks as reported on The Things Network website.

| Data Rate | Configuration | Bandwidth | Bit/s | Maximum Data Payload |
|---|---|---|---|---|
| DR0 | SF12 | 125 kHz | 250 | 51 |
| DR1 | SF11 | 125 kHz | 440 | 51 |
| DR2 | SF10 | 125 kHz | 980 | 51 |
| DR3 | SF9 | 125 kHz | 1760 | 115 |
| DR4 | SF8 | 125 kHz | 3125 | 222 |
| DR5 | SF7 | 125 kHz | 5470 | 222 |
| DR6 | SF7 | 250 kHz | 11000 | 222 |
| DR7 | FSK:50 kpbs | 50 kbps | 50000 | 222 |

**Table 1.2:** Data rate classification in LoRa technology considering TTN (The Things Network) rules.

LoRaWAN provides various classes of end devices to address different requirements in IoT appliances.

- **Bidirectional end devices (Class-A)** allow bidirectional communication whereby each uplink transmission (from an end device) is followed by two short downlink receive windows, as shown in Figure 1.3. These transmission slots are scheduled by the end device based on its own communication needs with a small variation based on a random time basis to avoid possible jams if more than one end devices send data at the same time. This class is the lowest power end-device system for applications that only require short downlink communication (for example, a single byte of configuration of that node) after the end device has sent an uplink message.

- **Bidirectional end devices with scheduled receive slots (Class-B)** that they have the same timing as class-A where, in addition to the random receive windows, these devices open extra receive windows at the scheduled time. To open these receive windows at the scheduled time, end devices receive a time-synchronized beacon from the base station. In this way, the network server knows when the end device is listening.

- **Bidirectional end devices with maximal receive slots (Class-C)** have

almost continuously open RX windows and only close when transmitting. This guarantees low latency at the expense of greater energy consumption.

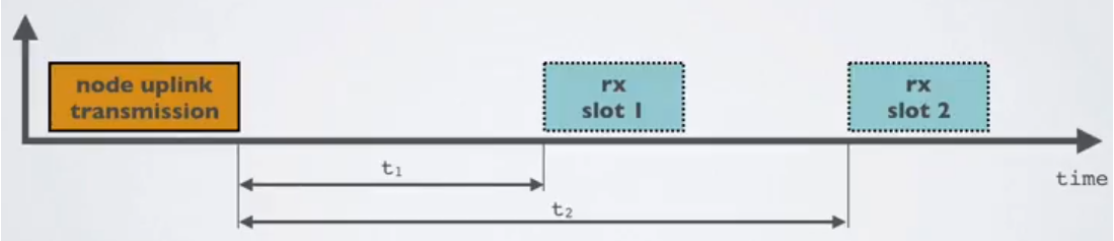



**Figure 1.3:** Timing of an end device LoRa class-A (Source [4]).

It is important to clarify that these classes are defined by the protocol where the first class (class-A) is the most common, class-B is on the market but less common and class-C is under development. The result is a so-called LoRa protocol stack that is shown in Figure 1.4.

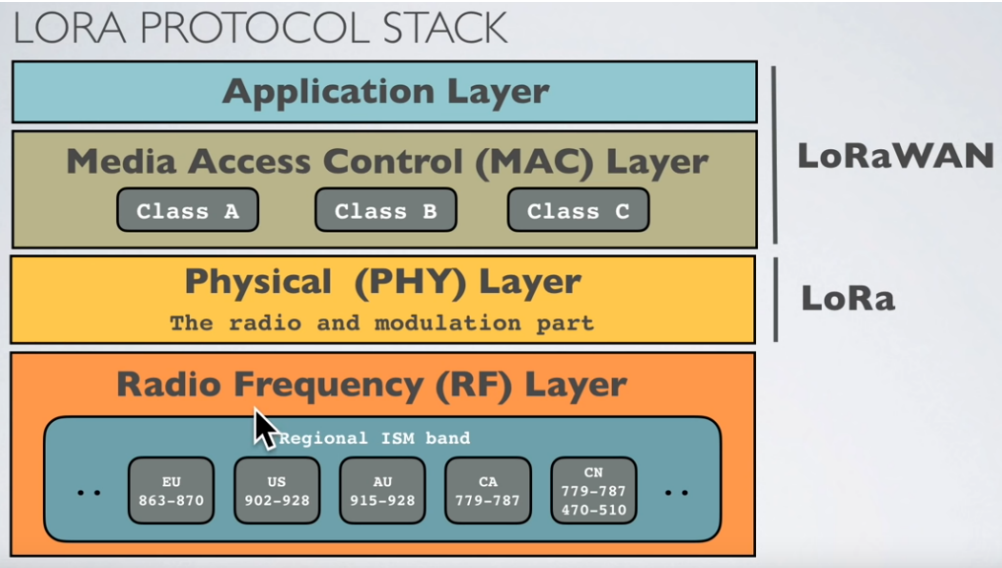



**Figure 1.4:** Full LoRa protocol stack (Source [4]).

Finally, it is better to explain two concepts: Time on Air (ToA) and Duty Cycle. When a signal is sent from a sender (both end device or gateway) it takes a certain amount of time before a receiver receives this signal. This is the **Time on Air (ToA)** or also commonly called **Airtime**.
Instead, **Duty Cycle** is the proportion of time during which a LoRa device is

operating. The duty cycle is typically expressed as a ratio or percentage with respect to the Time on Air. In Europe, ETSI (European Telecommunications Standards Institute) has defined 1% duty cycle per day (for some frequency channels, this number is 0.1%) depending on the channel with respect to the airtime.
For the sake of clarity, in this work, it has been measured the airtime to send a useful data payload composed of 33 bytes (plus at least 13 bytes defined by LoRaWAN protocol as handshake) equal to 2.3 sec in DR0 and equal to 0.3 sec in DR3.
This means practically that, in DR0, it is needed to wait for $99 \cdot 2.3\,\text{sec} = 227.7\,\text{sec}$ and, in DR3, $99 \cdot 0.3\,\text{sec} = 29.7\,\text{sec}$ to send a new packet with updated data.
This is coherent with the definition of data rate: if you want to send data on a longer range (smaller data rate level or higher spreading factor), you will wait more time to send another packet with respect to sending data closer to you (higher data rate level or lower spreading factor). In this way, the transmission medium is shared in an optimal way for all LoRa users in a certain zone.
For this reason, it is defined, by LoRa, a smart mode called **Adaptive Data Rate**, included in all LoRa stacks provided by various manufacturers. This mode handles a proper initialization of LoRa communication to a LoRaWAN network: a first packet is sent at maximum power (so in DR0). If it is received by a gateway, an acknowledgment is sent to the end device. At this point, packets with greater data rates (DR1, DR2, and so on) are sent, and always an acknowledgment from gateway is expected. When the range of a certain data rate is not sufficient to reach an available gateway, no one acknowledgment is received in some RX windows. The LoRa embedded system set itself up to the latest working data rate.
In this way, the system could send at the maximum frequency of that data rate, consuming the lowest energy power possible.

## 1.2 Soil Moisture Sensors

Another concept to be discussed is the way to sense soil parameters. In particular, the correlation between soil and water. In the agriculture context, this is performed by the so-called soil-water retention curve, proper of each soil. This is the reason to perform a campaign employing on-field sensors.
As it will be shown later, the topic is more complex: some kind of sensors has good performance in dry soil range, other ones in the wet soil range, some have compact dimensions, other ones require complex support appliances to allow to work its sensor properly.
For this reason, it is needed a little digression on these soil parameters and their state-of-art sensors.
A good starting point is provided by the Book [5], freely available by FAO (Food

and Agriculture Organization).
Dry soil is composed of particles of different sizes: most of these particles originate from the degradation of rocks, and they are called mineral particles. Residues of plants or animals are called organic particles.
These are the so-called soil particles that seem to touch each other, but in reality, there is space among them. These spaces are called pores.
It is reasonable that when the soil is dry, pores are mainly filled with air. When, on the contrary, irrigation or rainfall occurred, the pores are mainly filled with water. In Figure 1.5 is presented graphically.

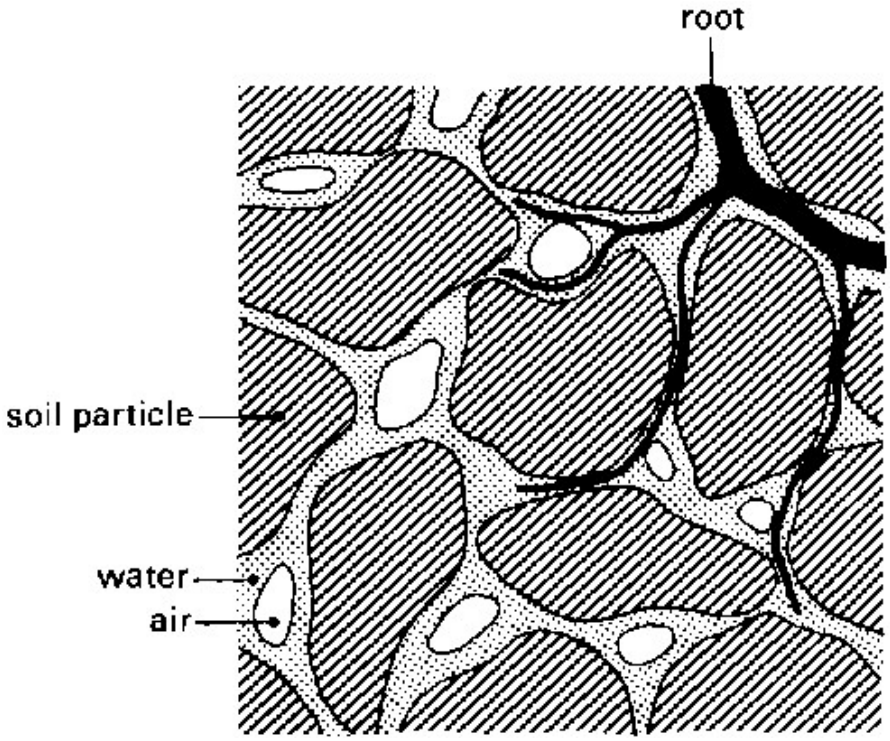



**Figure 1.5:** The composition of soil (Source [5]).

The mineral particles of the soil differ widely in size and it can be classified as in Table 1.3.

The amount of sand, silt and clay present in a considered soil determines the **soil texture**.
For this reason, it is defined that coarse-textured soils have a sandy component predominant (sandy soils), medium-textured soils have a loamy component predominant (loamy soils), and finally, fine-textured soils have a clayey component predominant (clayey soils). Before introducing the concept of soil moisture content,

| Particle Type | Size Limits |
|---|---|
| Gravel | Larger than 1 mm |
| Sand | Range from 0.5 mm to 1 mm |
| Silt | Range from 0.002 mm to 0.5 mm |
| Clay | Less than 0.002 mm |

**Table 1.3:** Mineral particles classification.

it should be clarified the idea of **infiltration**: when rain or whatever irrigation water task is supplied to a field, it seeps into the soil and this process is called irrigation.
This phenomenon could be evaluated quantitatively, considering the so-called **infiltration rate** of that soil. This is the velocity at which water can seep into it. Practically, it is measured by the depth (in mm) of the water layer that the soil can absorb in an hour.
Reasonable values are shown in Table 1.4.

| Classification | Range |
|---|---|
| Low infiltration rate | Less than 15 mm/h |
| Medium infiltration rate | From 15 mm/h to 50 mm/h |
| High infiltration rate | More than 50 mm/h |

**Table 1.4:** Infiltration rate classification.

Factors influencing the infiltration rate depend on soil texture, soil moisture content, and soil structure.
The **soil texture** is a constant factor of soil and it is a variable that influences the infiltration rate.
As said before, coarse-textured soils have mainly large particles in between which there are large pores. On the contrary, fine-textured soils have mostly small particles in between which there are small pores.
In coarse soils, water enters and moves more easily into larger pores or, in other

words, it takes less time to infiltrate into the soil.
In better words, the infiltration rate is higher for coarse-textured soils (sandy soils) than for fine-textured soils (clayey soils).
The **soil structure** can be influenced by farmer practices and this changes the infiltration rate of the soil itself. Generally speaking, water infiltrates (higher infiltration rate) faster into granular soils (for example, in a plowed field) than massive and compact soils (unworked field) considering two fields with the same soil texture.
The last parameter is the actual **soil moisture content** that indicates the amount of water present in the soil. As said before, it is expressed as the amount of water (in mm of water depth) present in a depth of one meter of soil. But this is, in reality, an average: the water infiltrates faster (higher infiltration rate) when the soil is dry than when it is wet so, as a consequence, when irrigation water occurred to a field, the water at first infiltrates easily, but as the soil becomes wet, the infiltration rate decreases.
The soil moisture content can also be expressed in percent of volume, as explained in Chapter 1.2.1.
The amount of water stored in the soil is not constant with time but decreases if there is no irrigation water event. Each soil has three noteworthy conditions with different soil volumetric water content and soil water potential values: saturation, field capacity and permanent wilting point conditions, as shown in Figure 1.6.

The **saturation** condition occurs when all soil pores, during a rain shower or an irrigation application are filled with water. As a consequence, there is no air left in the soil.
This is a condition where a plant will suffer. Many crops cannot withstand saturated soil conditions for a period of more than 2-5 days. When water present in larger pores drains on deeper tiers of soil and rain stops falling, the water drained from the pores is replaced by air. This process has different velocities for each type of soil: in sandy soils, drainage is completed within a period of few hours, where, on the contrary, in clayey soils, drainage may take some days.
The second remarkable condition is the so-called **field capacity**. After the drainage has stopped, the large soil pores are filled with both air and water while smaller pores are still full of water. At this stage, the soil is at field capacity. At this condition, the water and air contents of the soil are considered to be ideal for crop growth.
The last condition is the **permanent wilting point**, and it occurs when the roots are no more be able to suck water from the soil.
This is verified when the water stored in the soil is taken up by the plant roots or evaporated from the topsoil into the atmosphere. The dryer the soil becomes, the more tightly the remaining water is retained and the more difficult it is for the plant roots to extract it. At a certain stage, the uptake of water is not sufficient to

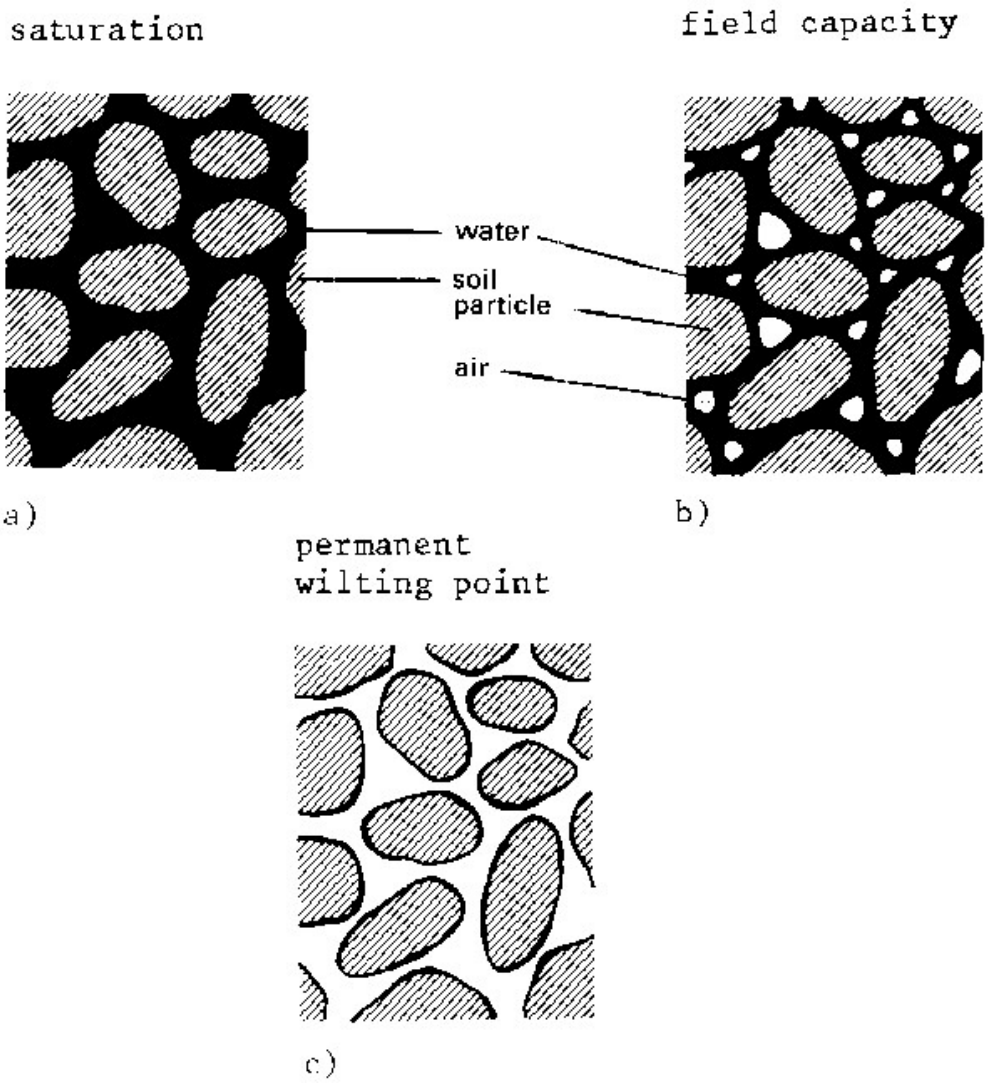



**Figure 1.6:** Some remarkable conditions of a generic soil (Source [5]).

meet the plant's needs. The plant loses freshness and wilts and their leaves change color from green to yellow until the plant itself dies.

### 1.2.1 Soil Volumetric Water Content (VWC)

The soil moisture content indicates the amount of water present in the soil and it can be expressed on a gravimetric or volumetric basis.
As explained in [6], gravimetric water content ($\theta_g$) indicates the mass of water per mass of dry soil. It is measured by weighing a soil sample ($m_{wet}$), drying the

sample to remove the water, then weighing the dried soil ($m_{dry}$). So, it is possible to express

$$\theta_g = \frac{m_{water}}{m_{soil}} = \frac{m_{wet} - m_{dry}}{m_{dry}};$$

This is an uncommon output variable when, instead, water content sensors send their data expressed as volumetric water content ($\theta_v$).
The volumetric water content is the volume of liquid water per volume of soil (expressed in $m^3/m^3$) where it is possible to obtain the correlation with respect to the previous definition

$$\theta_v = \frac{V_{water}}{V_{soil}} = \frac{\frac{m_{water}}{\rho_{water}}}{\frac{m_{soil}}{\rho_{soil}}} = \frac{\theta_g \cdot \rho_{soil}}{\rho_{water}};$$

where $\rho_{soil}$ is the ratio of dry soil mass to sample volume. The density of water is close to 1, so it is often ignored. This value is usually represented in percentage terms because it is less than one in an agricultural field.
The output value can be used to estimate the amount of stored water in a profile or how much irrigation is required to reach a desired amount of water.
There are two common classes of measuring techniques:

- **Neutron moisture measurement** is a technique where it is employed a neutron moisture meter, also called neutron probe. The probe itself is lowered into the soil down an access tube where neutrons are emitted and interact with soil water. The density of the neutron flux is dependent upon the amount of water in the surrounding soil. The probe contains a source of fast neutrons, and the gauge monitors the flux of slow neutrons scattered by the soil.
  This is one of the most accurate devices available for measuring the amount of water content in the soil (it is not hampered by environmental factors such as temperature or pressure and it is only slightly affected by salinity, the chemical composition of normal soils, or the degree of binding of water to the soil particles). But, as a drawback, it requires proper safe handling due to the fact that there is a radioactive substance that can be a potential health hazard. In addition, each operator in possession of a neutron moisture meter must have a special license and it must be periodically monitored using a film badge similar to those worn by X-ray technicians. For this reason, the neutron moisture meter is not commonly used by individual farmers.

- **Dielectric measurements** are performed evaluating the dielectric permittivity ($\epsilon$) to estimate the corresponding volumetric water content. In general, to compute the dielectric permittivity, it is injected a pulse (in DC or AC) between two (or more) electrodes where the soil is the medium in the middle.

In this way, it is computed the time to hold the charge injected and it is possible to estimate the dielectric permittivity after a computation. The result, expressed in dielectric permittivity, is shown in Table 1.5. Keep in mind that it is not possible to directly measuring water content, but it is only possible via indirect ways. This approach is more flexible respect the previous one due to the fact that it is not required a specialized technician and it is possible to bury these sensors for whatever time you want.

| **Material** | **Relative Dielectric Permittivity** |
|---|---|
| Air | 1 |
| Dry Soil | ≈4 |
| Ice | 5 |
| Saturated Soil | ≈40 to 50 |
| Water | 80 |

**Table 1.5:** Typical values of relative dielectric permittivity.

For this reason, the dielectric measurement is the best approach when you want to investigate water content in low-cost smart agriculture research. The most common output from this type of sensor is the percentage of volumetric water content (%VWC), but typically this value has to be computed from a raw output provided by the sensor. So, typically, it is provided an analog value in millivolt ($mV$) or raw counts (for example, as output from the internal ADC contained in the dielectric sensor).
Another critical aspect to highlight is the importance of calibration: the %VWC is estimated starting from a mV output and this value is correlated, in the factory laboratory under measurement, to well-known soil condition in such a way as to realize a polynomial equation. In Chapter 2.3.12 is shown the fancy equation used for TEROS 11, calibrated in the factory, as an example.
It is important to perform this task to eliminate systematic errors in the estimation of dielectric permittivity such as bulk density variation, dependence from temperature (dielectric permittivity of water changes with temperature, it is $\epsilon_{water} = 80$ at 20 °C), kind of soil (high clay content affects bound water) and salinity.
When this task is performed on the factory, the calibration is done on a "generic" soil type. Moreover, an on-field calibration should be performed before using the sensor.

### 1.2.2 Soil Water Potential

It is not sufficient to know the quantity of water stored in a volume of soil, but it is needed to evaluate another variable called the soil water potential. As in all natural systems, the movement of a material is dependent on energy gradients. Soil water potential is an expression of the energy state of water in the soil and must be known or estimated to describe water flux.
Formally, the total soil water potential is defined as the amount of work per unit quantity of pure water that must be done by external forces to transfer reversibly and isothermally an infinitesimal amount of water from the standard state of the soil at the point under consideration.
It is helpful to recall that potential is equal to a force per distance, so $mgd = \rho V gd$ (unit of measure is $N \cdot m$), where $\rho$ is the mass density, $V$ the volume considered, $g$ the gravitational force and $d$ the distance respect to the reference point. This potential can be expressed per unit mass, per unit weight, and per unit volume. The latter one is common in the agricultural field.
Remember that a potential per unit volume $\psi$ is equal to a potential over a volume so $\psi = \rho V gd/V = \rho gd$ with unit of measure $\frac{N}{m^2}$, so a water pressure unit.
Considering this relation, we do not need to compute the soil-water potential directly by computing the amount of work needed but measure the soil-water potential indirectly from a pressure measurement.
This potential is evaluated considering the fundamental forces acting on soil water

$$\psi_T = \psi_p + \psi_z + \psi_s + \psi_a;$$

where $\psi_p$, $\psi_z$, $\psi_s$ and $\psi_a$ are, respectively, pressure, gravitational, osmotic and air pressure potentials. The **pressure potential** $\psi_p$ is the energy per unit of volume of water required to transfer an infinitesimal quantity of water from a reference pool of water at the elevation of the soil to the point of interest in the soil at reference air pressure and temperature. If soil is saturated, $\psi_p$ is positive and also denoted a hydrostatic pressure potential. If the soil is unsaturated (case of interest in the agricultural case), $\psi_p$ is negative, and it is also called matric potential.
Practically, the matrix arrangement of solid soil particles results in capillary and electrostatic forces and determines the soil water matric potential. The magnitude of this value depends on the texture and the physical-chemical properties of the soil solid matter. Most methods for measuring soil water potential are sensitive only to the matric potential.
The **gravitational potential** $\psi_z$ is the energy per unit of volume of water required to move an infinitesimal amount of pure, free water from the reference elevation ($Z_o$) to the soil water elevation ($Z_{soil}$).

Water molecules have energy by virtue of position in the gravitational force field, just as all matter has potential energy. This influence is easily seen when attractive forces between water and soil are less than attractive forces acting on the water molecule, and water moves downward.
The **osmotic potential ($\psi_s$)**, also called solute potential, is present due to the fact that soil water is a solution where a certain amount of salt is present in the soil solution. The polar nature of the water molecule results in interaction with other electrostatic poles present in the solution as free ions. It is always negative since it is defined relative to pure $H_2O$. This contribution is important only if have semi-permeable membrane (as plant roots) or at the air-water interface.
The last contribution is the **air pressure potential ($\psi_a$)** that takes into account changes in air pressure, different than the reference pressure (atmospheric pressure).
The first one (matric potential) is the main contribution in the evaluation of the soil water potential in agriculture by a sensor. This is typically expressed in kPa. To give an idea of the order of magnitude in Table 1.6 are reported typical values. These values are different for each soil, and they should be measured by a campaign with the correspondent volumetric water content (VWC).

| **Condition** | **Matric potential $\psi_p$ (in kPa)** |
|---|---|
| Almost Saturation | $-1$ |
| | $-10$ |
| Field Capacity | $-33$ |
| | $-100$ |
| | $-1{,}000$ |
| Wilting Point | $-1{,}500$ |
| | $-10{,}000$ |
| Air Dry | $-100{,}000$ |
| Oven Dry | $-1{,}000{,}000$ |

**Table 1.6:** Soil water potential values example (Source [7]).

Higher potential means negative kPa values close to zero and, on the contrary,

**(a)** A Watermark gypsium block by Irrometer.

**(b)** A capacitive sensor TEROS21 by Meter Group Inc.

**(c)** A thermal matric sensor by Campbell Scientific.

**(d)** A tensiometer by Irrometer.

**Figure 1.7:** Illustration of four kind of soil matric potential sensors.

lower potential values are the most negative values. In this range, it should be identified the field capacity and the wilting point as explained in the introduction of this chapter.
There are four kinds of sensors (Figure 1.7) to sense this physical parameter indirectly:

- **Gypsum block** is a sensor where a standard matrix equilibrates with soil. Electrical resistance is proportional to the water content of the matrix. It is the most common way to acquire matric potential being inexpensive, but it has poor stability (requires good soil contact), accuracy (the resistance depends on salts in soil) and response. Accurate only in the dry range and it has a decent measuring range (0 kPa to −980 kPa).

- **Capacitive sensors** have a standard matrix equilibrates with soil where the water content of the matrix is evaluated by capacitance instead of resistance. It is stable (not subject to salts and dissolution) and it has a good accuracy from −10 kPa to −500 kPa. This is the best compromise in terms of performance, compactness, calibration difficult to characterize in a decent way the soil type of a field.

- **Thermal matric sensors** use the heat dissipation principle: it measures the rate at which heat is conducted away from a heat source, using temperature response close a heat source using a thermocouple or a thermistor. Employing this concept, they are robust (a ceramic element with embedded heater and temperature sensor), have a large measurement range (−9.8 kPa to −100,000 kPa, so wet and dry range), stable (not subject to salts and dissolution) but, as drawback, they require a complex temperature correction procedure and calibration.

- **Tensiometers** are devices where it equilibrates water under tension with soil water through a porous cup. It measures water pressure relative to atmospheric pressure evaluating the height of water filled in the tensiometer tube feed by the porous cup. This is the sensor with the highest accuracy in the wet range (0 kPa to −80 kPa) due to the limited air entry value of the porous cup. As a drawback, it is quite bulky and requires significant maintenance.

### 1.2.3 Soil-Water Retention Curve

The two physical variables, volumetric water potential and soil matric potential, determined by simultaneous measurements and provided by two different sensors, describe a relationship.
An example of this characteristic is shown in Figure 1.8 taken by [7].

As it can be seen, fixing the soil water potential, in clay soil, the soil volumetric water content is bigger than sandy soil. This is due to the fact that when soil drains, the largest soil pores empty first since the capillary forces are smallest in these pores. As the soil drains further, the maximum diameter of the water-filled pores further decreases, corresponding with pores that have decreasing values for the water potential (water is held by larger capillary forces). The largest pores it

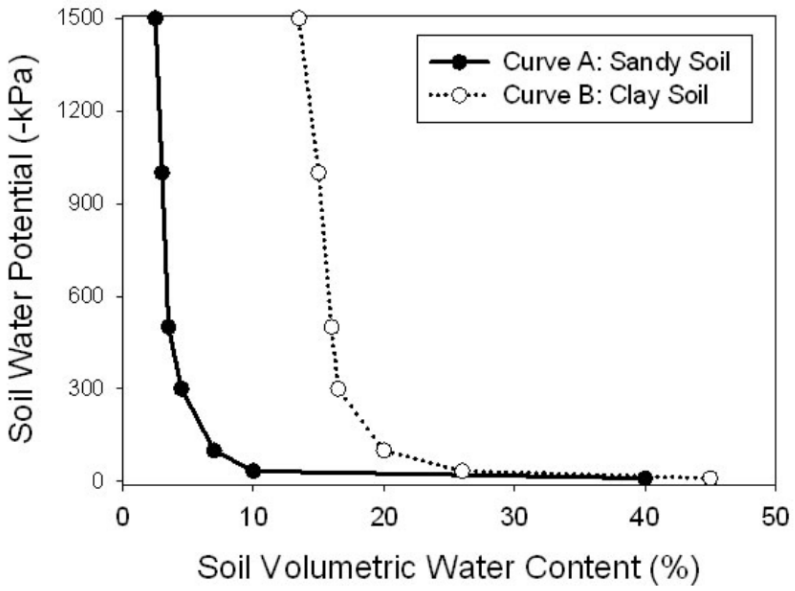



**Figure 1.8:** An example of soil-water retention curve (Source [7]).

is present in sandy soil with respect to clay soil.
In unknown soil, this soil-water retention is unique and is a function of pore size distribution. This is the reason because a measurement campaign should be performed, and the experimental soil-water retention curve of that soil should be computed.

## 1.3 State of Art of Agriculture Data Loggers

To design a good product is essential to know the commercial state of art regarding agricultural environment data loggers.
There are plenty of choices differing in kind of supported protocols (MODBUS, I2C, SDI-12), type of supply (mains-connected, rechargeable or non-rechargeable battery, solar-powered, other kinds of energy-harvesting devices), number of physical digital channels to be able to read data from sensors, presence of physical analog channels, rugged-grade (evaluated in IP-xx grade), supported protocol (hardwired as RS-485, USB or via radiofrequency as Bluetooth, LoRa, Sigfox, GSM/GPRS/3G/4G cellular protocol).
To give an idea about what the market offers, Table 1.7 shows some models that have one or more of the features stated above.
For the sake of completeness, indicated number of channels in the table is referred to as the physical ones. Each channel can typically handle more than one sensor

sharing the transmission media via I2C or SDI-12 protocol, for example. The price is not reported because it generally is to be requested by the manufacturer but, to give an idea, they are in a range of 500€ to 2000€ for single data logger.
These devices have benefit and drawbacks each of them:

- **Hardwired devices** requiring supply cable and this is a very difficult problem because it should be buried on the field: this requires a lot of work to have a dig where placing connection cables. But, on the other hand, there is no problem in terms of supply. **Non-hardwired devices** can be stand-alone but require some care in the design process. It will be shown some considerations in Section 2.3.3.

- The **number of physical channels** is very important because it should be greater than the number of sensors that should be connected to a single node. Otherwise, sensors should be joined with them, or an adapter should be purchased.

- **Lifetime** is crucial: each of the considered data loggers guarantees from 3 months to 2 years of sampling based on sampling time programmed, used protocol, and so on.

- Significant is the choice of the **protocol**: a hardwired data connection means the presence of a fixed PC in the field (very annoying) or spending some time to download data from every single node before that they have full memory. On the other hand, a radiofrequency protocol uses air as a transmission medium but growing the complexity of the single node and its consumption. In addition, there is to be considered some "maintenance" costs as, for example, the fee to use that frequencies or service cost (SIM and mobile plan) for cellular protocol.

- Each shown product is supported by a **program or a mobile application** in such a way to download data in whatever format that is needed easily. It is important to evaluate also that suite, its support guide, compatibility with other employed data loggers in the choice of the best datalogger for a specific application.

- Almost all data loggers guarantee some **Ingress Protection code** (IP code) and, in the case of agricultural data loggers, at least an IP-65 grade of the entire structure, to avoid that water or dust could enter in the electronic logic, is considered. For other environments, it could be needed a higher IP grade to afford, for example, immersions, fire-retardant, or radiation hardening.

| Part | Company | Comments |
|---|---|---|
| EM50 (Link) | ICT international | SDI-12 compatible, 5 channels, supply from alkaline or lithium battery (stated typically 8-12 months). Integrated memory so it is possible to download data via COM port using a laptop. |
| EM50G (Link) | ICT international | SDI-12 compatible, 5 channels, supply from alkaline or lithium battery (stated 6+ months). Integrated memory as backup memory and GSM/GPRS cellular protocol to send data. |
| SensorData (Link) | Digital Matter | SDI-12 compatible, 2 channels, supply from alkaline C-type. Integrated memory LoRa protocol to send data. Integrated GPS module. |
| Solar DataSnap (Link) | Acclima | SDI-12 compatible, accomodates up to 10 channels, solar panel in such a way to have an supply independent system. Integrated memory so data can be acquired via USB. |
| GP-BTDL (Link) | GroPoint | SDI-12 compatible, up to 10 channels, supply from alkaline or lithium battery (stated several months). Integrated memory and data can be downloaded via app using Bluetooth protocol. |
| ZL6 (Link) | Meter Group | SDI-12 compatible, 6 channels, supply from 6 alkaline or NiMH battery (stated 3-12 months depending on configuration). Integrated memory and communication via bluetooth (Basic model) and 3G/4G cellular (Standard and Pro models). |
| ADS-260 (Link) | Infinite | SDI-12 compatible, 8 channels, supply from lithium thionyl D-size battery. Integrated memory as backup memory and Sigfox protocol to send data. |

**Table 1.7:** Comparison among commercial data loggers.

# Chapter 2

# Investigation

## 2.1 Requirements

The first step of this thesis is to collect some requirements and constraints that have to be considered in all design steps of this project (for example, the choice of the right component, the best firmware methodology and which PCB design constraints have).
For this reason, it has been reported some general rules used as "design philosophy" in all development steps:

- Ultra low power system that in our context means a standby consumption of few $\mu$A to guarantee years lifetime;
- Be able to get air temperature and humidity;
- Compliant with a maximum of six TEROS sensors by METER Group, Inc. employing one of the two available serial protocols to gather data from all of those. In particular, it has been required the usage of three TEROS 21 and three TEROS 11 for soil moisture, volumetric water content and soil temperature measurements;
- Small system in such a way that is compatible with a solar radiation shield for outdoor applications to get good measurements also in the presence of wind and to protect the printed circuit board from the direct rays of the sunlight;
- Compatible with a pre-existing LoRa network sharing airtime resource with the same gateway if present;
- It has been required a system practical and straightforward for on-field operations as replacing a discharged battery, reset the system for whatever problem, easy to secure on a pole;

- The system must be reliable for everyone. For this reason, it should be designed protection against the inversion of the polarity of the battery, the connector of the antenna must be robust, the reset button must be clear to see, connectors for TEROS must be in an easy-to-access position;

- The system has to have a way to point out when the battery goes down to a fixed percentage of its capacity;

- The system should be a cost as low as possible in such a way to manufacture several nodes to place in the land;

- All raw data must be collected via LoRa protocol and stored for subsequent analysis.

Moreover, a possible LoRa system architecture to perform an entire measurement campaign could be described as in Figure 2.1, where a variable number $N$ of nodes are placed at some point of the fields, and their sensors are buried on the soil. A gateway, in a range of some kilometers, collects their data in LoRaWAN protocol and sends them to a LoRa provider as The Things Network in such a way to be processed and analyzed.

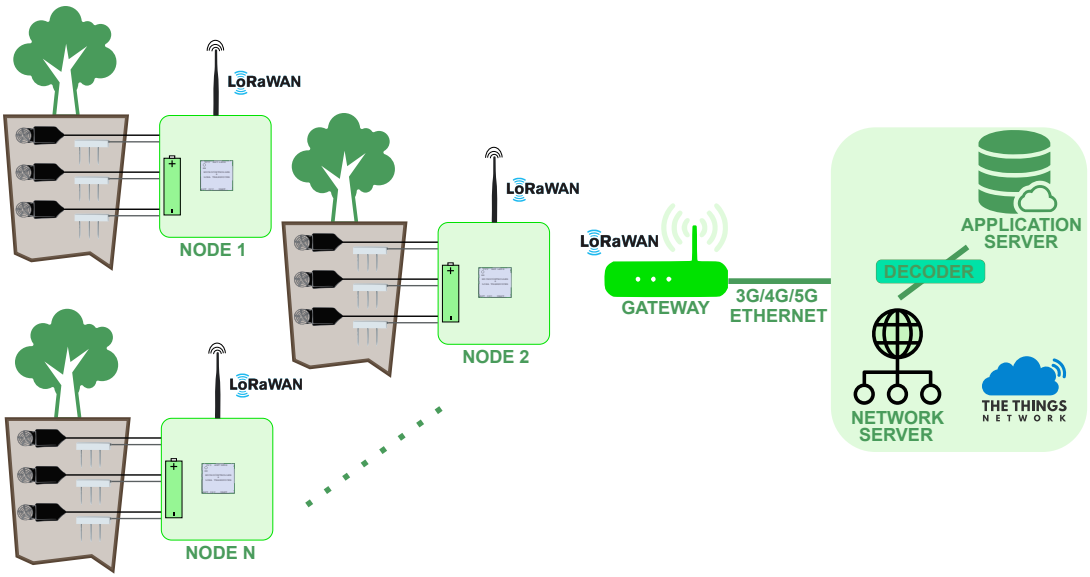



**Figure 2.1:** Overview of the designed system architecture.

## 2.2 Outline

To start a complex project, the best choice is to outline the main components and peripherals involved in the entire system.
So, to better clarify in which direction the master thesis goes, in Figure 2.2 is shown a simple block diagram to represent the printed circuit board to be realized and what should be placed.

At first sight, it is possible to subdivide the system into:

- A **microcontroller** in the middle of Figure 2.2 to handle every communication protocol and to timer for each operation properly. In addition, it is its work to handle the entire LoRa protocol;
- A **battery** to supply every section of the system. The node should be energy-independent for its entire lifetime;
- Six **soil sensors** provided by the WAPPFRUIT project to monitor soil parameters. Each of them should be disconnected when inactive and should communicate their data to the microcontroller;
- An **antenna** connected to the microcontroller via an RF path;
- A **reset button** to re-initialize the system in whatever moment;
- An **analogic multiplexer** to share the unique UART channel of the microcontroller (typically a microcontroller owns 2-3 UART channels at maximum);
- Some **LEDs** used to debug the system at the test development stage;

This is the result of a specific design process where each component has been studied to satisfy in the best way the requirements.

## 2.3 Components

### 2.3.1 Ultra Low-Power Microcontroller and Transducer LoRa

The first thing to choose is the platform where LoRa firmware should be run.
On the market, there are many choices in the base of what you want and how small the final Printed Circuit Board (PCB) should be.
We can highlight three main approaches to get a LoRa end node:

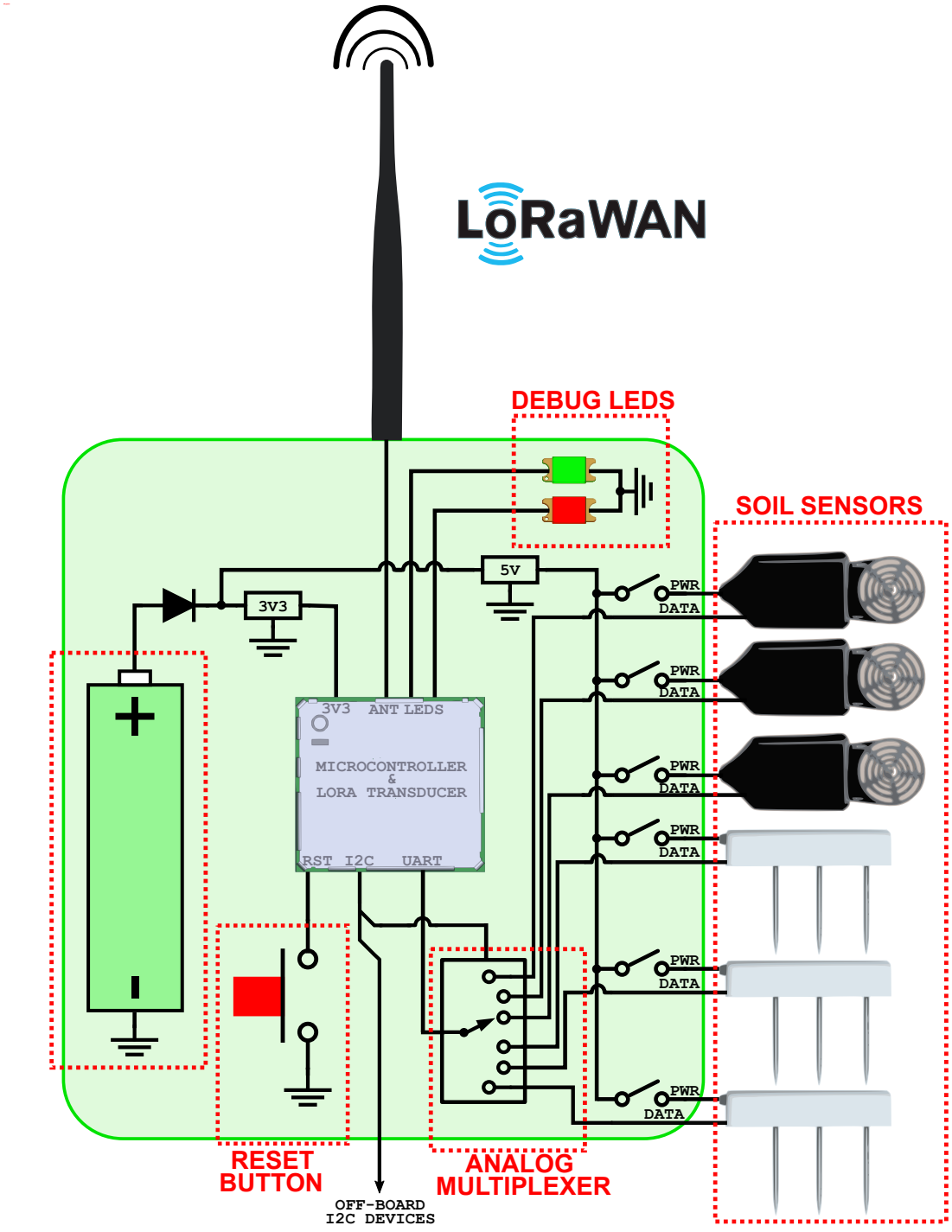


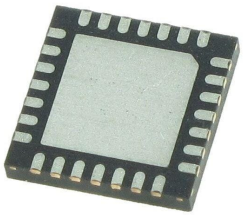

**Figure 2.2:** LoRa plant monitoring end node overview.

- **Design a fully custom embedded system**, buying your own favorite microcontroller (example by ST, Texas Instruments, Microchip, Atmel, and so on) and your own LoRa transducer (actually, the reality is that only Semtech, the LoRa founder member, provides a LoRa transducer IC). You have to design everything from placing components, draw matched tracks, RF filters and supply the power amplifier in the right way, following the data described in Semtech's datasheet. This guarantees the maximum of flexibility (it is possible to choose every microcontroller flavor that owns an available SPI peripheral) and offers the possibility of the most integrated approach to realize an embedded system but requires long design time and equipment to handle tiny components; For the sake of completeness, in the last year, companies, as ST Microelectronics or Microchip, presented **integrated circuits that include MCU and LoRa RF in one single package** as for example, ATSAMR34J18 by Microchip and STM32WL55 by STM. This can be, in the next future, the best solution for embedded solutions really integrated. But for now, it is not suitable due to they are new products that can not have a well-established developer community to support them.

- **Use a module**. Some companies (for example ST, Microchip) have realized in suitable RF shielded packages of usually cm-dimensions, module where you have a microcontroller and an IC LoRa transducer. The same companies, to speed up the design-to-market time, offer development boards in Arduino-size dimensions. This is the best solution to have an almost-fully-customized embedded system (compact, but you are constrained by the chosen microcontroller in the module) to debug easily the firmware and to realize a working PCB in few months of the development process. Usually, they are provided in LGA packages, so quite simple to handle.

- **Use a commercial board and develop a shield board** for your own sensors. Here, the actual market provides many solutions: for example, evergreen Arduino (MKR WAN1310) provides a board where you can use its Arduino IDE, its C simplified language and a configure-and-launch LoRa supporting library. When you don't have time to develop an embedded system and the requirements are not so strictly in terms of size and consumption, this is the faster solution: you should realize only a printed circuit board for your sensors and you should only connect pins of your shield with headers of the board.

It has been investigated each of the proposed categories where it should be considered, sizing, standby consumption, manufacturing support, community support, ease to purchase, and obviously price. In Table 2.1, there is an outline of the commercial products considered. The project must be compact and monolithic,



**(a)** Bottom view of Semtech SX1276 IC LoRa transducer.


**(b)** Murata module CMWX1ZZABZ.

**(c)** The last LoRa board by Arduino MKR1310.

**Figure 2.3:** Three illustrated approaches for a development of a LoRa system.

having a standby consumption magnitude in order of µA. Moreover, the module approach is the best one. As the last thing, it has been decided to use the Murata module CMWX1ZZABZ[8] that includes a ST microcontroller ultra low power STM32L072, a RF transducer Semtech SX1276 compatible with European LoRa frequency (868 MHz), a matching stage for the RF path, a TCXO as clock source of Semtech (it will be possible to use this clock source also as clock timing for the microcontroller) and a 32.768 kHz crystal for the real-time clock peripheral.

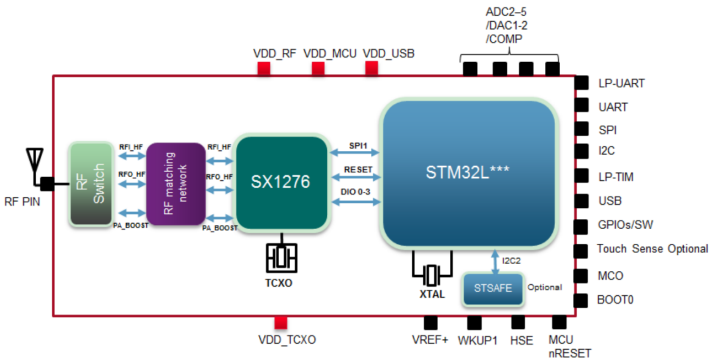



**Figure 2.4:** Block diagram of the Murata module (Source [8]).

This module guarantees a standby consumption of 1.65 µA that includes Semtech contribution and ST $\mu C$ usage in stop mode. When the system has to acquire data from various sensors, the microcontroller is woken up periodically by real-time peripheral (RTC), check sensors, "pack" all data acquired and send a LoRa packet of some bytes as payload. When the connection is established, it will be checked if the gateway has a packet for it. In that packet, some bytes are used as a simple command, for example, reset the system, change sampling time, etc.

| Category | Part | Company | Price ( € ) | Comments |
|---|---|---|---|---|
| IC | SX1276 | Semtech | 6.34 (Digikey) | Integrated circuit with transducer section communication in SPI. |
| Module | CMWX1ZZABZ-091 | Murata | 13.78 (Digikey) | Based on $\mu C$ STM32L072 by ST and Semtech transducer IC. Available a development board. Big community support. |
| Module | RFM95W | RF solutions | 11.09 (Digikey) | Many resources on the official website. |
| Module | RAK811 | RAK wireless | 17.23 (Digikey) | Presented on The Things Network (a popular LoRa provider). |
| Module | RN2483 | Microchip | 11.09 (Digikey) | Based on $\mu C$ PIC18LF46K22 by Microchip and Semtech transducer IC. Available a development board. |
| Board | MKR WAN 1310 | Arduino | 31.31 (Digikey) | Arduino IDE compatible. Big support community. |
| Board | Maduino | Maduino | 31 (Futurashop) | Based on $\mu C$ ATmega328 (with bootloader Arduino pro mini 3.3 V – 8 MHz) and integrated with module RFM95W. |
| Board | Wisnode LoRa | RAK wireless | 32.09 (AliExpress) | Presented on The Things Network (a popular LoRa provider). |

**Table 2.1:** Comparison among approaches considering only available products compatible with the European frequency (868 MHz).

### 2.3.2 Capacitors, Resistors and Inductors

Almost all electrical components require some passive components close to them to filter noise, to guarantee fixed performances, and so on.
The choice of the class of these resistors (through-hole or SMD) has a rather high impact on the total footprint of the final printed circuit board.
Excluding the through-hole category, because they own an enormous area footprint impact, the surface mounting device (SMD) category itself owns a very wide choice of dimensions: in imperial code, there are available from 2512 to 0201. The first two figures express the value in inches of length, and the two latter one expresses the value in inches of width.
For example, considering a 0805 device means that its length is 0.08 inch or 80 mils and its width is 0.05 inch or 50 mils.
Clarified this concept, a big passive component is easy to handle and soldering/desoldering by hand but has a non-negligible impact on the footprint area of the final PCB. Not only, a big component also typically has a bigger voltage rating and breakdown voltage, bigger stray elements.
In the case of capacitance, there are several reasons for choosing a larger package over a smaller one. In addition to the stated above reasons, one larger package typically allows for higher capacitance because there is more physical space. You could not, for example, get a decent 100 µF capacitor in an 0402 package.
Larger sizes usually mean a bigger dissipated power on that component.

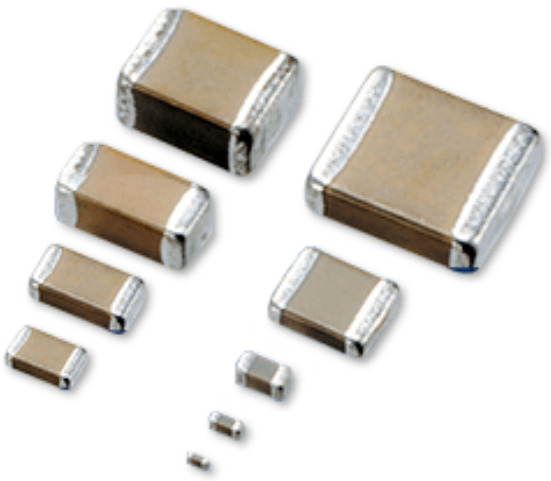


**Figure 2.5:** Different sizes of SMD ceramic capacitors.

Another reason is that there are different dielectrics. X7R is typically the best performer in terms of stability and has better DC performance. X5R is less good in that regard. Generally, X7R capacitors are physically larger, for the same voltage/capacitance ratings, than X5R capacitors.
This application does not have to handle higher voltage than 5 V, so it does not require big capacitances and it does not require a very small tolerance because there is no precision analog stage or special purposes to handle. A good choice could be 0603, a sort of trade-off among these variables. A 0603 component affects a little the area footprint but allows to handle hand operations as soldering/desoldering if needed.
Eventually, with a future and tested revision, these passive components could be shrunk to 0402 or 0201 sizes to have a more compact board.

### 2.3.3 Battery

The device to realize is definitely in the IoT (Internet of Things) category: it must be supplied autonomously without mains, it must collect data, and it must guarantee as long as a possible lifetime. For this reason, two approaches can be followed:

- A **Rechargeable System** correctly designed can survive many years employing a battery and some kind of energy harvesting (the most common one is a solar cell to recharge the battery when the sunlight shines in the sky). Virtually, if consumption and the spans of time when the battery is recharged are sufficiently large are predicted in advance, you can have a reliable IoT device. The drawbacks of this approach are a greater complexity (you have definitively more components, more volume occupied by node, you have to predict this stated consumption and recharging battery time in such a way that the system has to work in the worst case) and surely a greater cost.
  Moreover, this can be affordable if the system is in a well-developed state (for example, you have many revisions of the project), or the system has to be commercial and guarantee many years of working state without maintenance. In addition, it can be useful when the system is particularly energy-intensive (the rechargeable battery capacity cannot guarantee the required lifetime except if it is supplied in some way).

- A **Non-Rechargeable System** is designed to work until the non-rechargeable battery is dead. As opposite the first case, here we can have a simpler system that only should be alert when the battery should be replaced. In addition, typically, we have a cheaper system (no energy harvesting module is needed, no charging integrated circuit is needed).
  This approach is good when you want a simpler system (the design time is

reduced with respect to the other approach), when you want a cheaper as possible node and when you have the possibility to change a discharged battery easily when is needed.

Starting from this precondition and looking at the requirements, the system to realize falling down in the latter category because the node must only collect data, consuming a very little quantity of energy, it has to cost as little as possible to cover the entire lands in a proper way.
So, the topic moves to how non-rechargeable battery best fits the application: it is required compactness and popular sizes, a high nominal voltage to work as a single cell, a high discharge capacity (expressed in mAh) and last but not least, a large working temperature (in outdoor it is possible to reach −20 °C in the winter and 40 °C in the summer).
Table 2.2 can summarize the discussed considerations, excluding 9 V "1604" category because they are big and do not provide good discharge capacity (below 1,500 mAh) and excluding the button-shape category because they have ridiculous discharge capacity and low maximum output currents. The best category for this application can be the cylindrical category: here, there are many choices as alkaline, LiMnO2, LiFeS2, LiSOCl2, and Li-poly in different sizes.

Finally, the choice has been the LiSOCl2 (lithium thionyl chloride) batteries that are composed of a lithium metal anode and a thionyl chloride (SOCl2) as an active cathode. In particular, it has been chosen a model of SAFT called LS14500 (Figure 2.6).

This kind of battery, as cited in the datasheet[10], guarantees an extraordinary shelf life (less than 1% per year of storage, at 20 °C), high capacity, and high energy (as a consequence, it has a nominal capacity of 2,600 mA h), a typical operating voltage of 3.6 V, AA-size, resistance to corrosion, a very wide operating temperature range (as stated in the datasheet, from −60 °C to 85 °C).


**Figure 2.6:** The non-rechargeable LiSOCl2 battery in cylindrical AA size.

In particular, the nominal voltage is very "convenient" because it requires only a low-dropout regulator to distribute the supply voltage in the entire board at 3.3 V (the most common nominal voltage for low-voltage electronic devices), and in the entire life-cycle, the discharge profile voltage is very flat as illustrated in Figure 2.7.

| Chemistry | Part | $V_{nom}$(V) | Discharge capacity(mAh) | Comments |
|---|---|---|---|---|
| Alkaline | MN1300 | 1.5 | 11500 (D size) @250 mA | Popular, various sizes (D,C,AA,AAA). |
| | MN1400 | 1.5 | 5100 (C size) @250 mA | |
| | MN1500 | 1.5 | 2800 (AA size) @10 mA | |
| | MN2400 | 1.5 | 1200 (AAA size) @10 mA | |
| LiMnO2 | U10013 | 3 | 11100 (D size) @250 mA | High energy-density, flat discharge, excellent shelf life, good low temperature performance. |
| | CR2 | 3 | 850 (CR2 size) @20 mA | |
| | CR123A | 3 | 1550 (2/3A size) @20 mA | |
| LiFeS2 | L91 | 1.5 | 3200 (AA size) @25 mA | 90% after 15 years at 20 °C. |
| LiSOCl2 | LS14500 | 3.6 | 2600 (AA size) @2 mA | High energy-density, very flat discharge, very good temperature range, excellent shelf life. |
| Li-poly | BR-2/3A | 3 | 1200 (2/3A size) @2.5 mA | |

**Table 2.2:** Comparison between non-rechargeable cylindrical batteries as referred in Book[9].

When the resistance load is high (so, the output current is in the order of a few mA), as shown by red, black, orange lines, it is always in the proximity of a cell voltage of 3.5 V-3.6 V, whatever is the discharged capacity (so, the activity time of that battery).
This is a good benefit when the system is operative but declares some issues in the estimation of when the battery is at end-of-life and should be replaced, as it will be shown in Section 2.3.5.

As partial drawbacks of this type of battery, it is useful to highlight the fact that it has a very high output impedance. The chemical reaction that enables extremely low self-discharge and long shelf life (passivation formation) have the unwanted effect of limiting the available output current. The considered datasheet states a maximum continuous output current of 50 mA and a maximum peak output

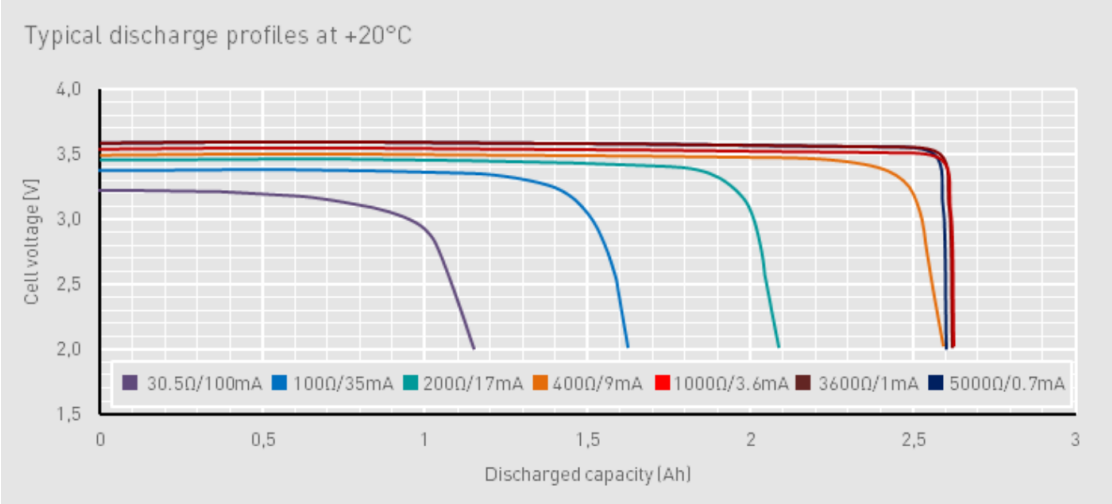



**Figure 2.7:** Typical discharge profiles at 20 °C varying the resistance load (Source [10]).

current up to 250 mA. These are definitively low values with respect to other non-rechargeable batteries.
But this is not a big problem in this application because the energy-intensive activity as transmission/reception in LoRa RF lasts no more than few seconds.

### 2.3.4 Inversion Polarity Protection

When a board is taken out from a laboratory, there are some reliability conditions in such a way as not to damage the board if a procedure is not performed correctly. In the case of a battery, the most common maintenance step is to replace it when it is dead. But if the designated person to substitute the battery is distracted, a common risk is to invert the polarity in the battery case.
If it is not considered in the design phase, the risk is to damage some components being the entire board useless!
For this reason, a very simple technique could be inserting a diode in series to the battery.
A simple pn diode (voltage drop around 0.7 V) but also a Schottky diode (voltage drop approximately 0.2-0.3 V) could solve the problem: when a battery is connected in the reverse position, a negative voltage is on the terminals of the diode avoiding to damage components downstream of that.
But it stands a problem: a non-negligible voltage drop, when it is in forward biasing (so when the battery is correctly connected), takes the regulator to work in the dropout region (so, the regulator does not "regulate" taking on the out pin the same unregulated voltage of the input pin).

An alternative solution can be the usage of an integrated chip configured as an ideal diode assuring the input polarity protection. An example arrives from Texas Instruments with its LM66100[11] that also guarantees a very low quiescent current when the system in operating condition and impacts a very little the total standby consumption of the PCB.
This die is composed of an operational amplifier connected with a logic stage that drives a nMOS in pass-transistor configuration, as illustrated in Figure 2.8.
When a negative input voltage is applied, the ideal diode will stay off and prevent current flow to protect the system load, as stated in its datasheet[11].

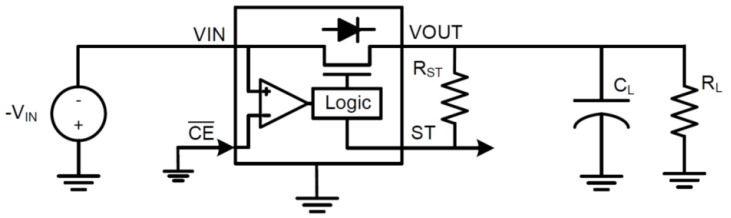



**Figure 2.8:** LM66100 protection circuit as illustrated in the TI datasheet[11].

In this way, an accidental battery replacement can not damage the board guaranteeing its functionality.

### 2.3.5 Battery Monitoring Stage

Despite efforts to minimize load currents and maximize battery life, applications are ultimately area constrained, and batteries will need replacement at some point. In a low-cost portable device, monitoring the discharge status of the battery and estimating the remaining capacity may be a high priority. For this application, the estimation of the remaining battery capacity is not critical (if the node switches off, there are no security issues), but anyway, it is a task to perform in such a way as to avoid "holes" in the periodic collection of data from sensors.
Moreover, having consciousness of the health state of the battery is an analysis to perform.
Initially, thinking about a traditional battery, it has been designed a simple battery voltage detector stage. In Figure 2.9, a quasi self-explaining schematic shows IC3, a load switch that connects *IN* and *OUT* pins, driven by a digital signal from $\mu C$ called *DIO_BATT_MEAS*. When a signal active-high arrives, the voltage of the battery is on *OUT* pin. At this point, a voltage divider halves the voltage that it will be read by an ADC channel of the microcontroller.

This rough approach can be useful to have an approximate estimation of the

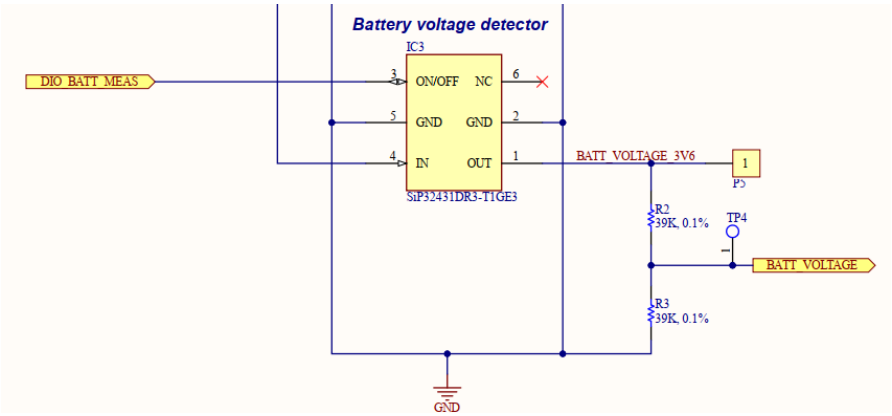



**Figure 2.9:** Screenshot from schematic related to the power management where it was presented a simple battery voltage detector stage (Source [12]).

charge of a battery for which of them own a sloping discharge. Moreover, by fixing a threshold when the voltage of battery goes down on it, the system can send a LoRa packet to the gateway containing an alert to replace it as fast as possible.
The problem, referring to the characteristic of the chosen battery illustrated in the Chapter 2.3.3, is its flat discharge curve. When the LiSOCl2 is active, the voltage is practically the nominal one. When the battery is dead, the system will be already switched off.
Consulting [12] has been demonstrated that 95% of the entire state of charge of LiSOCl2 is inside an interval of 100 mV. This led to a big problem: using an embedded ADC of a microcontroller (typically 8-9 effective bits on 12 stated bits) is very difficult to sample that resolution on a full scale of 3.3 V (estimated 36 mV). Another choice could be choosing an external ADC having an I2C protocol and the right number of effective bits to sample with a resolution of, for example, 1 mV. Performing some calculations, the right number of bits should be 16 bits and the actual market provides some integrated circuits as LTC2451 by Linear Technology or ADS111x series by Texas Instruments.
But, the main problem remains: how to choose the right value voltage threshold to consider the battery as dead? This approach requires many experiments to find the right point and, anyway, provide a superficial estimation due to do not consider temperature variations and the output current at the moment of sampling the voltage battery (and in this kind of battery, this latter variable has a not negligible impact).
Therefore, it remains only one feasible solution: employing an integrated circuit specialized in the monitoring of battery variables (voltage, current, and temperature of the battery correlating all of them together) employing I2C protocol as for example, BQ35100 from Texas Instruments.

The working principle can be explained seeing the Figure 2.10 recorded from its datasheet[12].

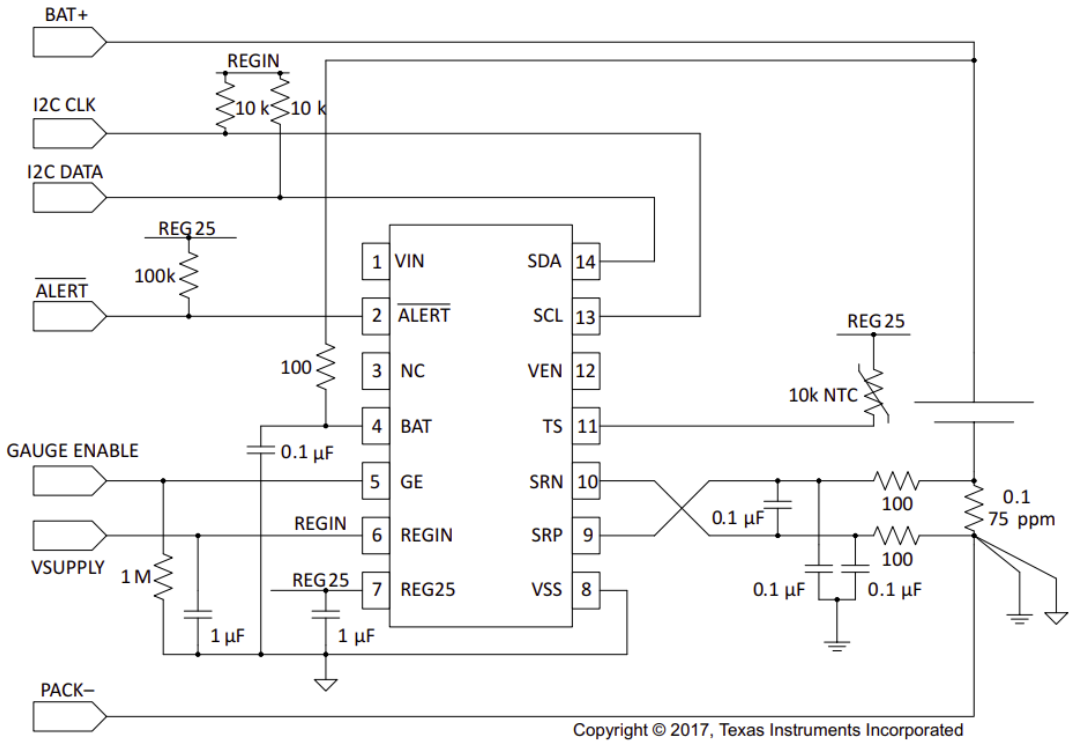



**Figure 2.10:** Typical circuit configuration of BQ35100 to monitor the state of charge of a battery (Source [12]).

The integrated circuit communicates with a microcontroller via I2C protocol (*SDA* and *SCL* pins) to configure, set and get raw data by BQ35100. This IC needs the voltage of the battery (by *BAT* pin), sink current by battery (*SRN* and *SRP* connected to a very small sensing resistance and the temperature of the shell of the battery.
For the latter one variable, IC provides three modes to feed it:

- **Using a NTC**, so a Negative Temperature Coefficient thermistor connected to the *TS* pin. This is the proposed solution as default because it is simple and flexible;
- **Using the internal temperature sensor of the die**. It requires that the integrated circuit should be in contact with the battery shell. Usually complicated.
- **Provide the temperature data from I2C protocol** using whatever

temperature sensor in contact with the battery. Useful if there are some sensors close to the battery for additional purposes.

Clarified this concept, to avoid consumptions all time, there is a specific pin called *GAUGE ENABLE (GE)* that informs BQ35100 to acquire data about consumption. The datasheet clarifies that it should be activated when an activity starts soon. In this way, the chip is able to sample all variables to obtain the better estimation possible. Typically, this is a task of the microcontroller.
In Figure 2.11, there is an example of what is intended.

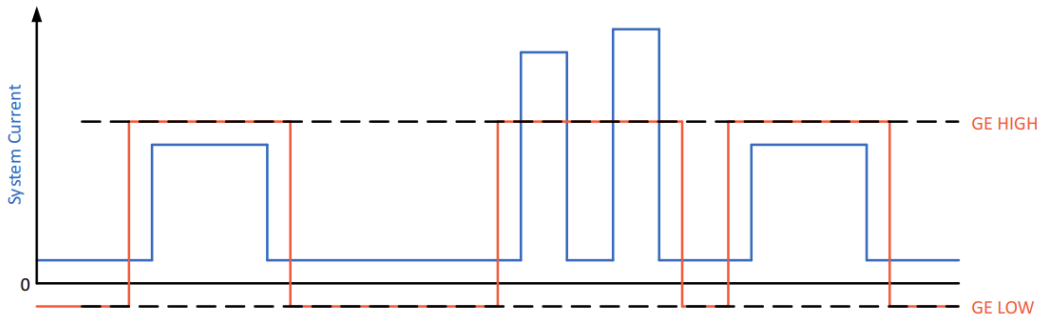


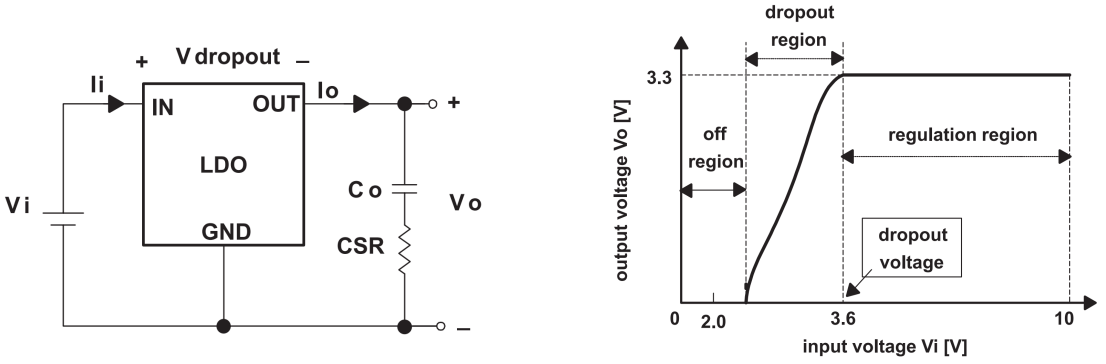

**Figure 2.11:** Example of power activity and when it is needed to activate *GE* pin (Source [12]).

For the LiSOCl2 battery, there is a specific mode called **End-Of-Service (EOS)** that considers the resistance correlation to estimate the lifetime of the battery.
Furthermore, at the battery installation step, it is required to alert the BQ35100 that it will estimate the battery output resistance as new and, during its entire lifetime, it will evaluate its variations.
At some point, when the battery should be replaced, a pin (*ALERTn*) is activated low and a microcontroller, sampling it, can alert someone.
This is the working principle in a nutshell. But the reality is quite complex: there are some tasks to do to have a serviceable battery monitor: write and configure a library for this specific battery, take care of some tracks on PCB board and perform at least one campaign to test that all is working.
Moreover, for timing reasons, this stage was postponed to a following revision of the board.

### 2.3.6 3.3 LDO Regulator

One of the required main aspects is the low standby consumption. When the device does not have to sample, the microcontroller must stay in stop mode (one of the low-power modes defined by ST architecture).
Moreover, there is evidence that one stage of a nominal voltage of 3.3 V should

always be on in such a way to supply the microcontroller, its RTC, and to guarantee the data retention of its internal registers.
It is possible to supply the $\mu C$ directly using the battery as a voltage generator, but this means exposing the core of the system to several variables (dead battery, high loads, spike of voltage) that depend on the nature of an electrochemical cell. For this reason, it is always advised to place a regulator in such a way to guarantee a very stable voltage as a supply of the microcontroller avoiding its strange behaviors. A typical regulated voltage for this kind of electronic component is 3.3 V, which is also compatible with a wide category of integrated sensors and MEMS.
Established the fact that the battery (starting point) has a nominal voltage of 3.6 V, to guarantee a regulated 3.3 V, it is better to choose a linear step-down for these electronic elements, with respect to a switching (greater conversion efficiency but more noise on the output) type.
It is important to focus on two fundamental characteristics of a linear dropout (LDO) voltage regulator: the **dropout voltage** and the **quiescent current**. These two characteristics are well highlighted in a Texas Instruments application note [13].
The first element to clarify is dropout voltage: in a typical voltage regulator configuration, as in Figure 2.12a, is the input-to-output differential voltage drop between IN pin and OUT pin at which the circuit ceases to regulate against further reductions in input voltage. In that region, the pass element (typically a transistor) is simply a resistor, and dropout is expressed in terms of its on-resistance ($R_{on}$). As suggested by the same figure, the dropout voltage can be computed as $V_{dropout} = I_o \cdot R_{on}$.




**(a)** Typical configuration of a linear voltage regulator.

**(b)** Graph shows all regions where a voltage regulator can work.

**Figure 2.12:** Concept of dropout voltage (Source [13]).

This region must be avoided to work properly because, otherwise, there is no precise voltage regulation (the output voltage assumes the input voltage minus a

drop voltage of regulator). In our case, we have to choose a device with a very low dropout voltage because the working voltage is 3.3 V respect a 3.6 V as input voltage. In Figure 2.12b, there is an example of a LDO regulator with a dropout voltage of 300 mV at a certain drained current. If you work with this device, you are always approaching the dropout region, so the regulator does not regulate!
For this reason, it is very important choosing a step-down converter that has a dropout voltage of 100 mV and no more than 200 mV for this application, assuming that for the entire working lifetime of the battery, it will not go down on 3.5 V (reasonable for this kind of battery that has a very flat discharge curve as seen in Chapter 2.3.3).
The latter concept, more important in the computation of the lifetime of the node, is the quiescent current (or ground current), which is the difference between input and output currents when the output current approaches a null value and the regulator is on.
Mathematically it can be defined as $I_q = I_i - I_o$ referred to the Figure 2.13.

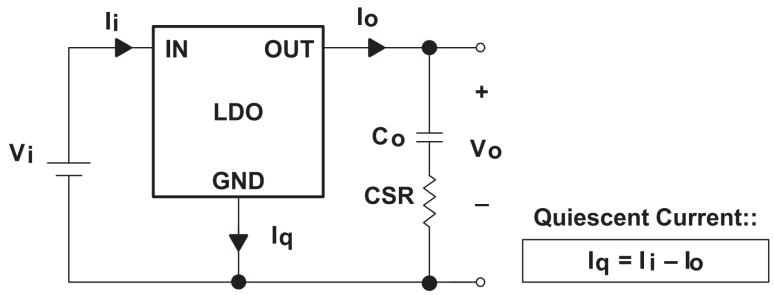



**Figure 2.13:** Quiescent current of LDO regulator (Source [13]).

It is useful to highlight the fact that the regulator is on because it is common to find integrated circuits that own an enable pin to switch on/off the regulator itself. But in this application, there is no control machine to perform the task of switching on or off the regulator because the entire state machine is inside the microcontroller that must always be supplied to temporize the sequence of measurements and standby moments. When an enable pin is switched off is common to say that this is the standby current and not the quiescent current.
In this work, the quiescent current is a fundamental parameter to lowering as possible the total standby consumption of the board.
In the choice of the best LDO regulator, it can be useful a table by ST microcontroller wherein the x-axis there is the maximum output current, and in the y-axis, there is the dropout voltage when there is a maximum of the absorption from the regulator. As shown in Figure 2.14, it has been investigated two models by ST Microelectronics STLQ020 and LD39130S starting from the model used to regulate on the LoRa

discovery board (LD39050). This last model can be useful to have greater maximum current to supply all elements on the board.
In the LoRa end node, it has been computed a worst-case current consumption in all operations below 200 mA (it can be seen in Chapter 2.4).
If there is needed allowance, it can be chosen LD39130S model which provides a maximum current of 300 mA paying in a quiescent current of 1 µA at no load.
But for this application, STLQ020[14] provides a very low dropout voltage of 160 mV at the maximum current of 200 mA and a very low quiescent current of 300 nA at no load should be sufficient. There is also a convenient SOT323-5L available package for this latter model.
An alternative from another supplier can be TPS7A05 by Texas Instruments.

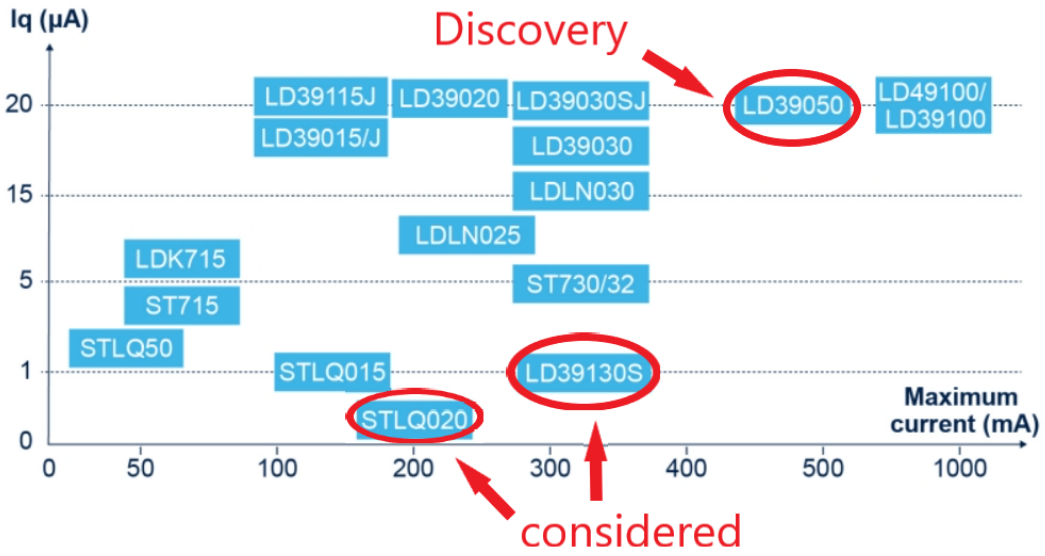



**Figure 2.14:** Graph presents all ST LDO regulators comparing their maximum current and their quiescent current (Source here).

### 2.3.7 5V Boost Regulator

It is required to provide a 5 V continuous voltage to supply TEROS sensors. Considering the fact that the regulator is off when the load switch is open, there are no constraints related to the quiescent current, so the choice is very wide on the market.
For this reason, initially has been chosen MCP16251/2 by Microchip that allows a low power mode (quiescent current of 600 nA) when the enable pin is fixed to ground but, successively, it has been chosen a model by Texas Instrument signed

TPS61222[15] that requires only three external passive components reducing as much as possible the area footprint of that stage.
The datasheet shows a requirement of one inductor, one input capacitance and one output capacitance to place close to the integrated circuit. The input capacitance is recommended to improve the transient behavior of the regulator and EMI behavior of the total power supply stage. The output capacitance is recommended to minimize output voltage ripple.
It should be a convenient component to supply TEROS sensors and the analog multiplexer.

### 2.3.8 Load Switches

Each node in the land should consume as little as possible, and the most straightforward solution to reduce the total consumption of the system is to switch off the entire sections when they do not work.
In this application, each TEROS sensor consumes 30 µA when it is asleep, but also the 5 V boost regulator is useless when no one measurement is required. It draws a non-negligible standby current for this ultra-low-power system.
For this reason, each stage that consumes current should be isolated by a load switch in a pass transistor configuration.
As shown in Figure 2.15 from the datasheet of SiP32431DR3[16] by Vishay, a controlling device as a microcontroller can drive the *ON/OFF* pin, taking the power MOS from off state to saturation condition. This is practically a gate for all components downstream of it.

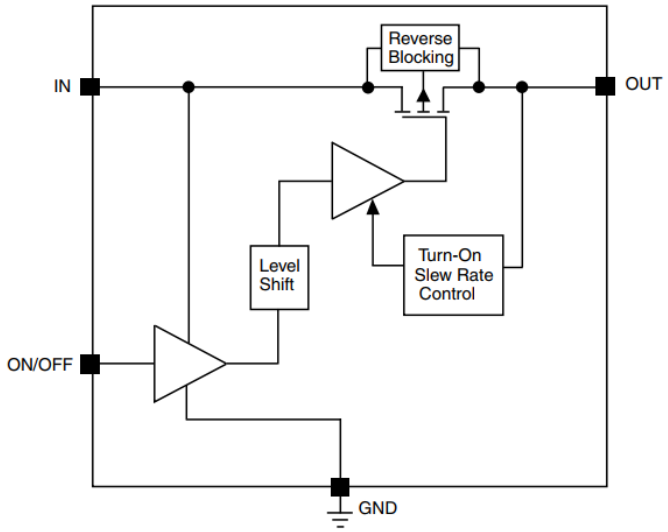



**Figure 2.15:** Block diagram of SiP32431[16] (courtesy of Vishay).

Moreover, considering a null consumption of those components, the only consumption in standby condition is the quiescent current of the load switch: this value is typically 10 pA at 3.3 V and 50 pA at 5 V, so absolutely negligible respect to other always active components in standby that consume hundreds of nA or few µA. This approach, composed of a gate to supply or not a specific section, allows reducing the standby consumption of a mA factor (considering all components included TEROS sensors).
The chosen model is SiP32431DR3-T1GE3 by Vishay with high enable and in SC-70-6 package that is compact but easy to handle and to solder also by hand.

### 2.3.9 LoRa Antenna

An antenna is an interface between radio waves propagating through space and electric currents moving in metal conductors. Being LoRa a radio protocol, the choice of the right antenna for the application is crucial.
The required LoRa omnidirectional antenna is typical of two formats (showed in Figure 2.16):

- **SMA antenna** uses an RF connector of 7.9 mm (referred to as a male connector). The female connectors can be soldered over an edge of a PCB as surface-mounted. They are designed to have a characteristic impedance of 50 ohms. The SMA connectors are most commonly used in microwave systems, hand-held radio, mobile telephone antennas and are commonly used in radio astronomy, particularly at higher frequencies (5 GHz+). They are rated for up to 500 mating cycles.

- **U.FL antenna** uses a miniaturized RF connector of 2.0 mm. The male connectors are surface mounted and soldered directly to the printed circuit board. They are designed to have a characteristic impedance of 50 ohms. U.FL connectors are commonly used inside laptops and embedded systems to connect the Wi-Fi/Bluetooth antenna, GPS antenna. It guarantees a very limited footprint on a PCB paying in brittleness (few reconnections are designed for this type of antennas).

The outdoor application requires a reliable antenna to avoid whatever problem in the mounting step and in the working step over a wide range of weather conditions. So, the choice is the SMA connector, not having particular constraints regarding the final dimensions.
Finally, it should be considered the body size of the antenna. Here, there is a world of choices: smaller or bigger, consumer or rugged rated, straight or 90°-angled.
Definitely, it will be considered an antenna that can perform a km-distance communication (so not too small to have the maximum sensitivity), rugged for whatever

**Figure 2.16:** Comparison between SMA male and U.FL female connectors.

climate condition (possibly IP65 rated to avoid dust and splash water could enter inside the system).

### 2.3.10 Light Emitting Diodes (LEDs)

An embedded system has not always a direct interface to observe in which state the machine is present (is it on? is it in transmission or reception LoRa state? Does it measure from some sensors?). But, in the development step, when there is to test the custom board, it is helpful to have a clear and eye-catching signal when the debug port can not be used. For this reason, to know roughly the state of the microcontroller (writing a good debug code) has been designed to insert a couple of small LEDs, one of red color and the other green.
Typically, 0805 or 0603 SMD packages (considering the same format of capacitors, resistors and inductors of Chapter 2.3.2) are standard for embedded applications. As the opposite of it, when a led is on, it draws a current of some mA quantity, which is not compatible with a standby current of µA order.
Moreover, these LEDs have been deactivated in the **Release** firmware of the custom board to avoid drawing useless current.

### 2.3.11 Switch for TEROSxx Channels

As shown in Section 2.3.12, TEROS sensors need to be supplied and they communicate on the data line in DDI protocol that is a serial-type. But having six sensors and only one UART peripheral on the microcontroller, there is the needed to have one component to regulate the access to the serial peripheral.
At the first stage, it has been designed to use a simple digital multiplexer as TI SN74HC251[17], shown in Figure 2.17.

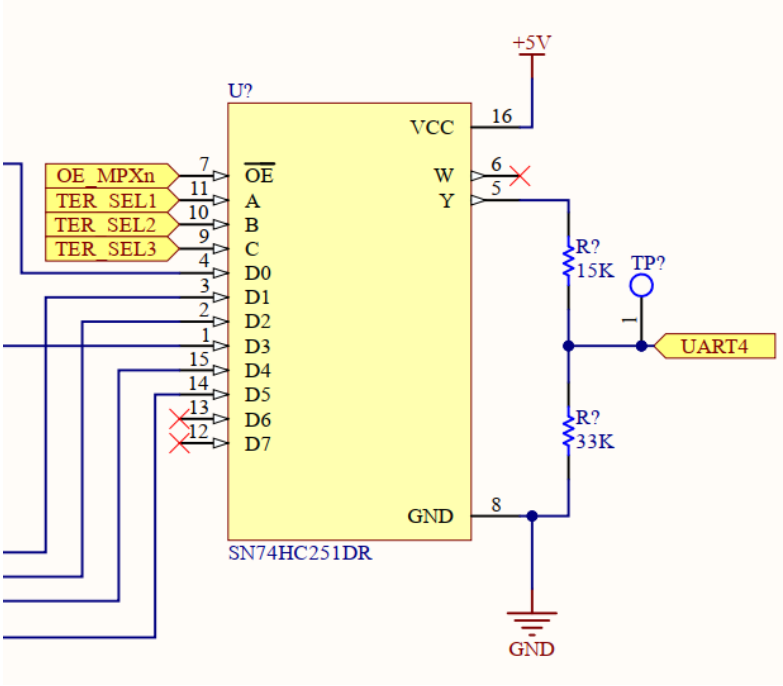



**Figure 2.17:** Preliminary version of the schematic section related to the switch among TEROS sensors.

In a few words, there are eight input channels (*D0..7*), three selection pins to select data from one of the input channels (*A*, *B*, *C*) and one output-enable (*OEn*) to enable the device to work properly. On the other side, there are supply pins (*VCC*, *GND*) and the output channel and its opposite digital level (respectively *Y* and *W*).
Being the data signal on input channels of a nominal voltage of 3.6 V (as high-logic level), if the supply voltage is the same of the microcontroller (3.3 V), it is not respected what the datasheet states (supply voltage must be higher than input and output voltages on the other pins of the IC).
Given the fact that it is not possible to use direct voltage of the battery (always better to supply all components downstream of a protection some kind of voltage

regulations), the last available supply voltage is 5 V used to supply TEROS sensors correctly. This is acceptable if the input high and low-level windows are respected (as in the case of this component, but it has to be verified for all multiplexers) but it restores the signal at the output pin to 5 V (when on the activated input channel there is a high-level signal) that, connected to the microcontroller UART pin, breaks itself.
A dirty solution could be a voltage divider on the output stage before being connected to the UART port.
The presence of several voltages simultaneously can take issues. For this reason, it should be better to consider an analogical counterpart of the same component: an analogic multiplexer.
They are less common with respect to the digital and, typically, they are more expensive because they have a niche group of applications as the audio/video world. Anyway, they can be useful because they can be supplied with a nominal voltage and all input and output data can "travel" without any distortion (excluding the very small drop of the voltage of its $R_{on}$).
After research on the electronic markets about available analog multiplexer IC, it has been chosen a Maxim MAX4562[18] and, in particular, the MAX4562EEE model that guarantees a temperature range from −40 °C to 85 °C that is compliant to outdoor usage. The basic model (MAX4562) provides a temperature range of only 0 °C to 70 °C, not sufficient for agricultural use.
In addition to these features, it has been designed to use the pre-existing I2C channel (SCL and SDA) to reduce the number of selection and output enable pins needed to work with a typical multiplexer.
In Figure 2.18 is shown its internal connections.

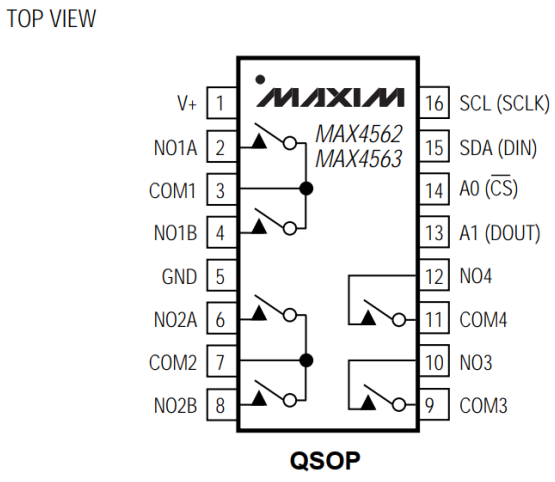



**Figure 2.18:** MAX4562EEE pin configuration.

As it can be seen, it is not a classic multiplexer, not having only one output pin (there are four *COM* pins) and having a weird nomenclature (*NO1A*, *NO1B*, etc.) linked to the internal relationship.
But this is not an actual problem: it is possible to shortcircuit COM ports each other and to use every each *NOxx* pin as input channel of one of the TEROS sensors. An implementation is shown in Figure 2.19.

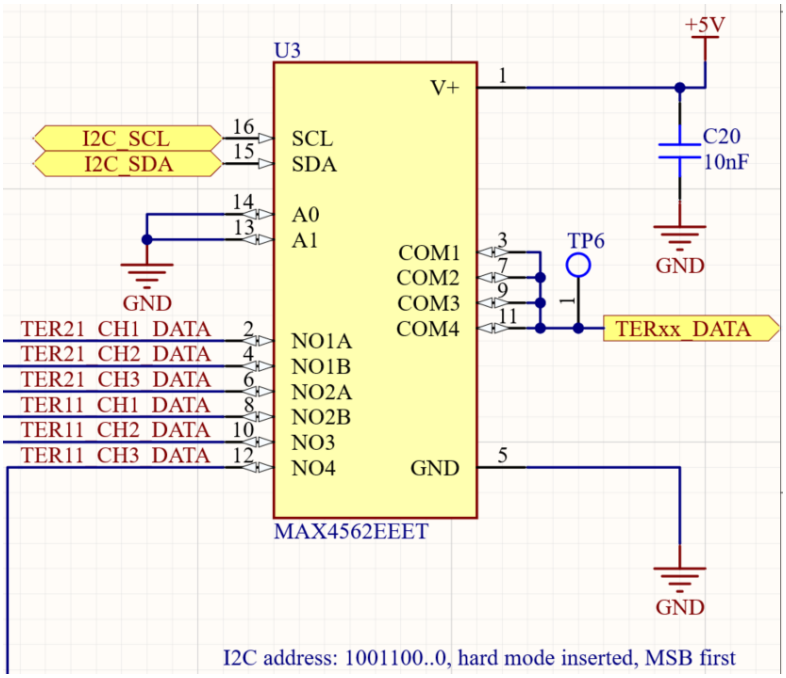



**Figure 2.19:** MAX4562EEE possible implementation.

Finally, there is to take a look at the I2C configuration.
As shown in Figure 2.18, it is possible to identify six switches called SW1A, SW1B, SW2A, SW2B, SW3, SW4, respectively. Moreover, Maxim integrated circuit requires to enable one single bit for each switch to perform one task. Tasks in this IC are two: set a specified switch open or closed, set a specified switch to soft or hard mode. These tasks must be activated, triggering two bits called C0 and C1 (Command-bits).
The result is a basic finite state machine where a microcontroller should only send an address byte (where the last two bits are determined by the logical value of *A0* and *A1*).

| ADDRESS BYTE | | | | | | | | | COMMAND BYTE | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB | | MSB | | | | | | | LSB |
| 1 | 0 | 0 | 1 | 1 | A1 | A0 | 0 | ACK | C1 | C0 | SW2A | SW1B | SW2A | SW2B | SW3 | SW4 |

**Figure 2.20:** MAX4562EEE I2C data format.

### 2.3.12 TEROS Sensors

A part of the investigation stage is to analyze how to handle the soil moisture TEROS sensors indicated among requirements in Chapter 2.1.
Moreover, it can be useful an overall description about how to work starting from TEROS 21 by METER Group, Inc.
This is a **soil water matric potential sensor** that, as explained in the Chapter 1.2, is the measurement of the amount of energy required for a plant to perform work in order to extract moisture from the soil. Soil water potential is expressed by a number of different units, but the most common unit is kilopascals (kPa).

| | Kilopascals (kPa) | Megapascals (MPa) | Bars | Centibars |
|---|---|---|---|---|
| | -1 | -0.001 | -0.01 | -1 |
| | -10 | -0.01 | -0.1 | -10 |
| Field Capacity | -33 | -0.033 | -0.33 | -33 |
| | -100 | -0.1 | -1 | -100 |
| | -1000 | -1.0 | -10 | -1000 |
| Wilting Point | -1500 | -1.5 | -15 | -1500 |

**Figure 2.21:** Example of values obtained in a soil from capacitive soil matric sensor (Source [6]).

The more negative the number then, the greater amount of work a plant needs to do in order to extract moisture from the soil.
Each type of soil owns proper field capacity and wilting point and each plant owns its soil water moisture variations when some meteorological phenomena, irrigation, or feeding nutrients events occur.
But this only physical variable is not sufficient to have a good knowledge of the actual state of the field: this data should be correlated with the **temperature soil** and its **volumetric water content (VWC)**. This last parameter is expressed in $m^3/m^3$, so the volume of water on a soil volume considered.
For this reason, the board should be able to handle data sensors from both TEROS 11 and TEROS 21.

Both kinds of sensors are digital so, they can be sampled by a digital protocol:

**(a)** Soil water content TEROS 11 sensor by Meter Group Inc.

**(b)** Soil moisture potential TEROS 21 sensor by Meter Group Inc.

**Figure 2.22:** TEROS sensors showcase.

their datasheets specify SDI-12 and DDI protocol.
The former one is the most complete and configurable but requires additional hardware to handle correct communication so, it will not be discussed in the future in this thesis.
The latter one is the most simple and requires only one UART channel of the microcontroller in a half-duplex communication. As a drawback, this protocol cannot configure the internal registers of the sensor or update its firmware. For these reasons, these sensors are intended as plug-and-play for the board requiring the purchase of an SDI-12 compliant professional datalogger if it is needed to configure/update its firmware.
The DDI protocol is very simple: one pin for the ground (*GND*), one for the data communication as transmitter stakeholder (half-duplex), one for the supply (*VDD*). The supply voltage can be in a range of 3.6 V and 15 V. It is undesired to have exotic high voltage (no 10-15 V), and it is a bad choice to use as reference voltage the nominal battery voltage (it is true that it has a flat discharge curve but it is anyway unregulated, so it suffers from variations when the load changes and wearing in its lifetime). A legit voltage can be 5 V supplied by a boost voltage regulator in such a way to have margin in every moment of the battery and avoiding to handle high voltages aboard.
Instead, regarding voltage of active-high digital communication line is specified a typical voltage of 3.6 V that is compliant with the voltage range that microcontroller's pins can handle as specified in Figure 2.23.

| Symbol | Definition | Min | Max | Unit |
|---|---|---|---|---|
| $V_{DD}$–$V_{SS}$ | External main supply voltage (including $V_{DDA}$, $V_{DD\_USB}$, $V_{DD}$)(1) | –0.3 | 4.0 | V |
| $V_{IN}$(2) | Input voltage on FT and FTf pins | $V_{SS}$ –0.3 | $V_{DD}$+4.0 | |
| | Input voltage on TC pins | $V_{SS}$ –0.3 | 4.0 | |
| | Input voltage on BOOT0 | $V_{SS}$ | $V_{DD}$ +4.0 | |
| | Input voltage on any other pin | $V_{SS}$ –0.3 | 4.0 | |

**Figure 2.23:** Voltage characteristics of microcontroller STM32L0 series.

At this point, when it is assured electrical compatibility among microcontroller and sensors, it is possible to take a look at meaning DDI protocol.
For both categories of TEROS sensors, there is the same start-up procedure: starting from the unsupplied sensor, supply it with 5 V between *VDD* and *GND* pins. The data line grows to a high level (so 3.6 V) within 100 ms. At this time, the sensor will measure within 200 ms as specified by the datasheets.
Now, it will start to communicate a *start* bit (low-level bit), 8 bits as a byte ASCII meaning, and a *stop* bit (high-level bit). This is repeated for all bytes of the DDI serial response.

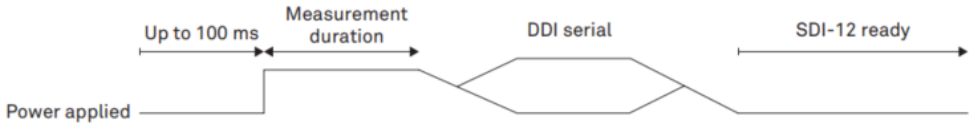



**Figure 2.24:** Data line DDI serial timing of TEROSxx (Sources [19], [20]).

The received string has a different meaning for TEROS 21 and TEROS 11/12. For the first one, after <TAB> character is indicated, the soil water potential in kPa with a resolution of 0.1 kPa and the soil temperature in the point where the sensor is placed with a resolution of 0.1 °C. Finally, a character to express the sensor type, a character for the checksum check, and a character for CRC check. All of this is summarized in Figure 2.25.

| COMMAND | RESPONSE |
|---|---|
| - | <TAB><matricPotential> <temperature><CR><sensorType><Checksum><CRC> |

NOTE: There is no actual command. The response is returned automatically upon power up.

The values in this command are space delimited. As such, a + sign is not assigned between values and a – sign is only present if the value is negative.

**Figure 2.25:** DDI response of TEROS21 (Source [20]).

For the latter category, it is needed to make a distinction: TEROS 12 contains, in addition respect to TEROS 11, the estimation of the soil electrical conduction. For this reason, two models of TEROS (11 and 12) has a different string size.
After <TAB> command, there is **calibratedCountsVWC** a number needed to calculate the volumetric water content using a stated conversion formula (expressed in $m^3/m^3$ with a resolution of $0.001\,m^3/m^3$), the soil temperature as TEROS 21 with a resolution of 0.1 °C and, only in TEROS 12, the electrical conductivity expressed in mS/cm. Finally, as TEROS 21, sensor type character, checksum and CRC check codes.
All of this is summarized in Figure 2.26.

| COMMAND | RESPONSE |
|---|---|
| NA | <TAB><calibratedCountsVWC> <temperature> <electricalConductivity><CR><sensorType><Checksum><CRC> |

NOTES: <electricalConductivity> is only output on the TEROS 12.

There is no actual command. The response is returned automatically upon power up.

**Figure 2.26:** DDI response of TEROS11-12 (Source [19]).

A proper UART library to handle the DDI response of these sensors in the firmware development step will be developed.

### 2.3.13 Case and Glands

Every electronic system must be protected by atmospheric agents like rain, snow, wind, sunlight but, in the particular case of agriculture applications, also from water splashes, vibrations and impacts from typical operations in an agriculture field as watering, manuring on plants, sowing, and other tractor operations.
For this reason, it must be designed a protective case that will be small, reliable and easily connectable to a pole. On the market, there are no ready-to-use protective cases that are compatible with the system. In fact, in addition to IPxx protection of the case, every point of contact between the inside shell and the outer world must be sealed to avoid seepages. Every drop that seepages into the shell can take shortcircuits.

The first point to clarify is the concept of IP grade.
The first figure indicates the effective protection against solid particles starting from 0 (no protection), passing through 3 (effective against solid particles greater than 2.5 mm as tools or thick wires), reaching level 6 (no ingress of dust so complete protection) called dust-tight.
The latter figure indicates effective protection against harmful ingress of water starting from 0 (no protection), passing through 5 (effective protection against water projected by a nozzle from any direction) to 8 (protection against continuous immersion more in the depth of 1 meter). It exists other categories as 9X, effective protection against powerful high-temperature water jets, which they are no interesting in the agriculture applications.
It has been established a case of IP65 grade as the minimum able to guarantee effective protection from atmospheric agents and typical agriculture operations for an outdoor case.
Moreover, a choice can be the protection enclosure RF1210BF compliant with IP65 grade by Hammond Manufacturing.
It is composed of polycarbonate in light gray color with an opaque lid and allowing a fastening using central holes (very convenient in this application taking into account some clearance for both left and right side when it is sealed) but also more lateral holes for a more common attachment.
Finally, it has been considered a presence of a bottom flange to attach the box to a pole.
Stated that a drawing is worth thousands of words, a sketch is proposed in Figure 2.27. All lateral sides will be drilled to insert glands in such a way as to guarantee a complete sealing for the antenna on one side and a bundle of sensor cables on the other sides.
After this choice, it has been chosen a cable gland 50.616 PA/RSW by Jacob GmbH for a cable diameter in a range between 3.0 mm-10.0 mm for each TEROS sensor cable and a cable gland PNC1/2W BK080 by Alpha Wire for a cable diameter in a range between 10.0 mm-14.0 mm for the sealing of the LoRa antenna.

### 2.3.14 Headers e SWD Interface

A well-established communication is fundamental to allow an interface from and to the end device.
In particular, to load the firmware on the microcontroller, an interface is needed. As cited in the Murata datasheet[8] of CMWX1ZZABZ, an SWD connector is necessary. In Figure 2.28, it is shown a screenshot of that particular stage.
It is a very simple serial interface where two specific pins (*SWCLK* and *SWDIO* on the module) allow, respectively, a clock source to establish a precise timing and a half-duplex data stream. The remaining pins on the SWD connector are related

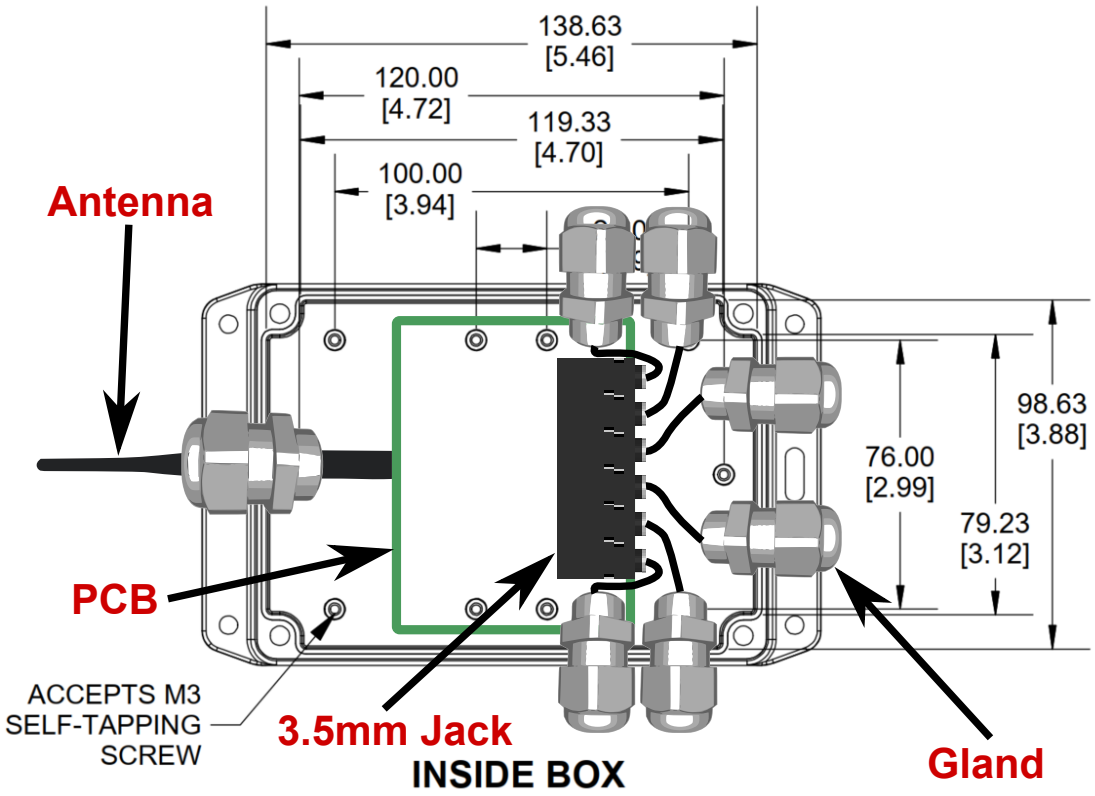



**Figure 2.27:** A quick sketch of the idea of PCB inside RP1210BF IP65 case.

to the supply, common ground, and, on pin 5, a direct connection to the hardware reset pin of the module. A possibility to keep things as simple as possible is a 6x1 2.54 mm header. In this way, it is possible to use common jumper wires used in the Nucleo board, Arduino board, etc.
There are also some fancy connectors with more compactness, but it is not important in this specific embedded system: it will require specific cables and a programmer with the same connector.
These 2.54 mm headers are useful in others contexts: a first revision requires some expedients as 1x1 headers to supply the board or to connect a common ground using laboratory supply appliances. They are the common size in electronics.
The unique exception is related to the I2C off-board connector: this is used to collect data from an external PCB board containing I2C sensors for environmental acquisitions as temperature, humidity, pressure, VOC (Volatile Organic Compounds), etc. In other projects in the laboratory, it has been used a B5B-XH-AM(LF)(SN) connector by JST Sales America Inc. It is a vertical header connector 2.54 mm composed of 5 positions. It has the feature to be asymmetrical: in this way, it is

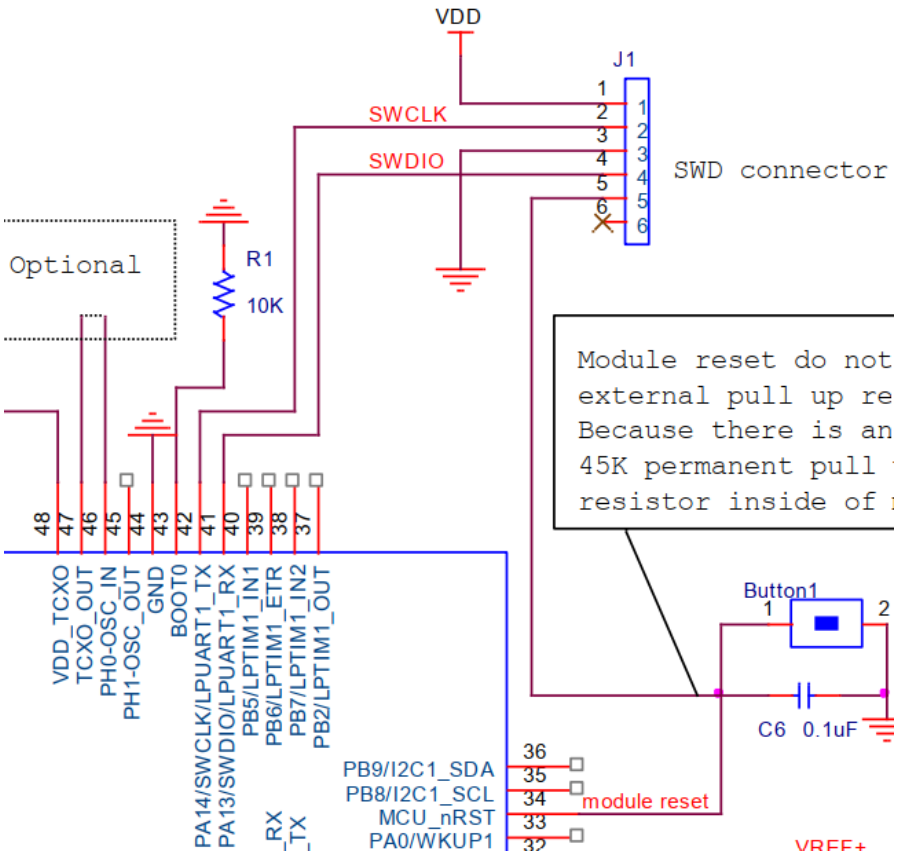



**Figure 2.28:** SWD interface connections as suggested in Murata datasheet[8].

impossible to connect the male connector in the wrong way, destroying the module. For compatibility reasons, it has been employed this female connector in the board allowing the usage of also previously realized I2C sensors modules.

## 2.4 Designed Consumption

Almost always, the microcontroller is in an ultra low power mode and almost all parts of the circuit are switched off. To compute a theoretical lifetime of a single battery with the PCB is necessary to know their sleep/standby modes. In this way, It is possible to get an estimated **standby consumption** of the entire system. Here, it has been reported Table 2.3 including all components that are switched on when the system is in low-power mode.
Note that **Q. curr** means quiescent current (also called ground current or standby current). TQC means **Typical Quiescent Current** and MQC means **Maximum Quiescent Current**.

| Part Name | Function | Note | N.parts | TQC (µA) | MQC (µA) |
|---|---|---|---|---|---|
| CMWX1ZZABZ[8] | $\mu$C + LoRa stage | In stop mode | 1 | 1.65 | 1.65 |
| STLQ020C33R[14] | 3.3 V LDO regulator | Q. curr | 1 | 0.3 | 1 |
| SiP32431[16] | Load switch | Q. curr | 1 | 0.00001 | 0.1 |
| LM66100[11] | Inv. polarity protection | Q. curr | 1 | 0.15 | 0.3 |
| HDC2080[21] | Temp. and hum. sensor | In sleep mode | 1 | 0.05 | 0.1 |
| Ceramic cap. | Decoupling | Charged | 13 | 0.5 | 0.5 |
| Total | | | | 2.65 | 3.85 |

**Table 2.3:** Estimated standby current consumption of LoRa end node.

Some considerations:

- For all parts are reported typical (*Typ.*) and maximum (*Max.*) quiescent currents, where available, as cited in own datasheets. Usually, the difference between the two values is due to the environmental temperature considered. The first one states 25 °C of the latter one considering the entire working temperature range of that integrated circuit. For an outdoor system is always better to consider the second one for a realistic battery lifetime.
- For all considered components, it has been searched their ultra low power model in such a way as to reduce as possible the total standby consumption of the system. Every microampere saved means days or months of additional work before replacing the battery in the land.
- Quiescent currents are outlined in the somewhere section of datasheets except for the ceramic capacitors. To compute their leakage current, you have to know the insulation resistance and rated voltage of these capacitors. Below, it has been reported calculus estimate around 0.5 µA of total capacitor current leakages.

Vishay ceramic capacitors datasheet[22] shows, for big values (as 0.1 µF, 1 µF, and 10 µF), values provided in $\Omega \cdot F$ form. This value has to be divided for the nominal

capacity of capacitors.
Moreover, it has been performed to know the insulation resistance (IR):

$$IR(C = 0.1\,\mu\text{F}) = \frac{\text{Value in datasheet}(\Omega \cdot F)}{\text{Nominal Value}(F)} = \frac{500\,\Omega\,\text{F}}{0.1\,\mu\text{F}} = 5{,}000 \times 10^6\,\Omega;$$
$$IR(C = 1\,\mu\text{F}) = \frac{\text{Value in datasheet}(\Omega \cdot F)}{\text{Nominal Value}(F)} = \frac{500\,\Omega\,\text{F}}{1\,\mu\text{F}} = 500 \times 10^6\,\Omega;$$
$$IR(C = 10\,\mu\text{F}) = \frac{\text{Value in datasheet}(\Omega \cdot F)}{\text{Nominal Value}(F)} = \frac{500\,\Omega\,\text{F}}{10\,\mu\text{F}} = 50 \times 10^6\,\Omega$$

Knowing the insulation resistance, it has been computed simply as first Ohm's law having current as a variable using the rated voltage of that capacitor (it has been considered 6.3 V for capacitors):

$$I_{leakage}(C = 0.1\,\mu\text{F}) = \frac{\text{Rated voltage}(V)}{\text{Insulation resistance}(\Omega)} = \frac{6.3\,\text{V}}{5{,}000 \times 10^6\,\Omega} = 1.3\,\text{nA};$$
$$I_{leakage}(C = 1\,\mu\text{F}) = \frac{\text{Rated voltage}(V)}{\text{Insulation resistance}(\Omega)} = \frac{6.3\,\text{V}}{500 \times 10^6\,\Omega} = 12.6\,\text{nA};$$
$$I_{leakage}(C = 10\,\mu\text{F}) = \frac{\text{Rated voltage}(V)}{\text{Insulation resistance}(\Omega)} = \frac{6.3\,\text{V}}{50 \times 10^6\,\Omega} = 126\,\text{nA}$$

Finally, it can be added as contribute row in Table 2.3 as a sum of leakage currents of the involved always charged capacitors.

$$I_{leakage} = 4 \cdot I_{leakage}(C = 0.1\,\mu\text{F}) + 6 \cdot I_{leakage}(C = 1\,\mu\text{F}) + 3 \cdot I_{leakage}(C = 10\,\mu\text{F}) = \\ = 4 \cdot 1.3\,\text{nA} + 6 \cdot 12.6\,\text{nA} + 3 \cdot 126\,\text{nA} = 458.8\,\text{nA} \approx 0.5\,\mu\text{A}$$

More complex is to compute a theoretical **runtime consumption** and, subsequently, an estimation of the battery lifetime.
This is due to the fact that there are many components with their own consumption in each stage. Moreover, the work is to isolate the main contributions in such a way as to compute a reasonable value.
The first step is to find the main procedure of the node. It is useless finding the consumption in the startup stage because it is an operation that is present only once. It is more important analysing what are the main steps during its work.
This is easy due to it is practically a data logger, so there are operations as a measurement from sensors, a data processing step, a LoRa transmission step, a LoRa reception step, and, after these, a sleep time until the microcontroller is woken up by RTC. This process in a loop.
Furthermore, it can be summarised in a graph as in Figure 2.29.

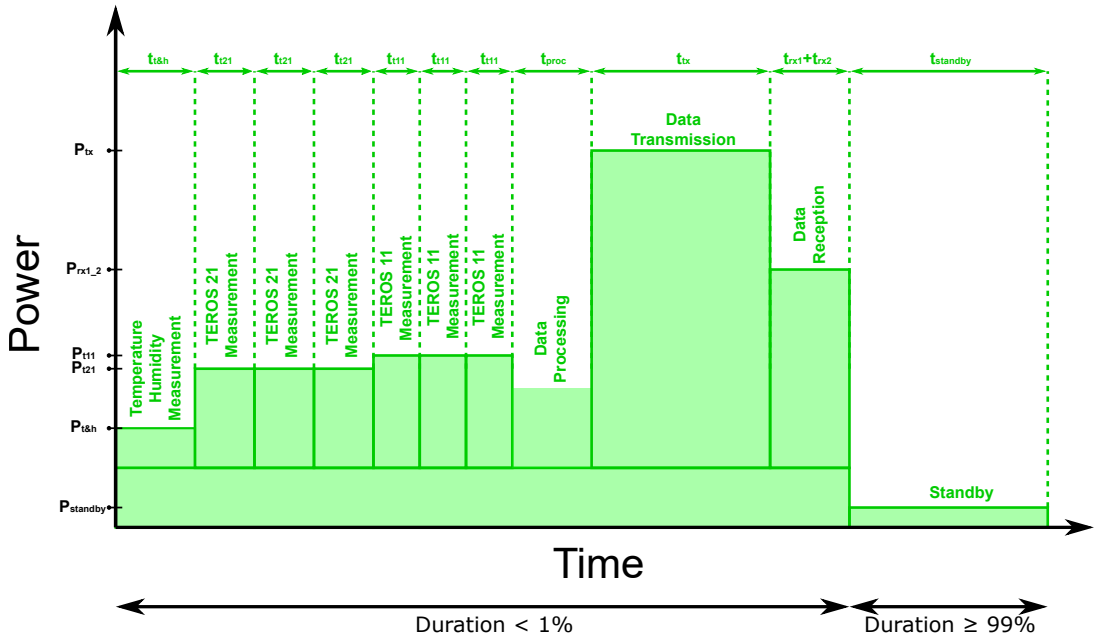



**Figure 2.29:** LoRa end node runtime consumption graph (not in scale).

It is clear to see each step, where a temperature/humidity sensor is sampled, three TEROS 21 and three TEROS 11 are sampled, a very small data processing stage (neglected), data transmission, data reception. All of these are on a "floor" consumption composed of ground currents and each biased electronic component (neglected).
Each of these processes should be analyzed considering its worst-case energy consumption and how much time employing.
All found data are reported in Table 2.4.
Let's consider a sampling rate of 1 hour, so 3,600 s. Being clocked by RTC, to find time in low-power mode, at this time should be subtracted the runtime where all tasks are performed before coming back to sleep.
Let's try to consider the case of Data Rate (DR0):

$$\begin{aligned} t_{runtime} &= t_{t\&h} + 3 \cdot t_{t21} + 3 \cdot t_{t11} + t_{tx} + t_{rx1} + t_{rx2} = \\ &= 1.27\,\text{ms} + \approx 3 \cdot 0.5\,\text{s} + \approx 3 \cdot 0.5\,\text{s} + 2.58\,\text{s} = 5.6\,\text{s}; \end{aligned}$$

After this, it can be computed the time in standby:

$$time\ \ in\ \ standby = t_{standby} = 3{,}600\,\text{s} - 5.6\,\text{s} = 3{,}594.4\,\text{s};$$

| Operation | Time | Voltage | Current | Power | Energy | Note |
|---|---|---|---|---|---|---|
| **Murata LoRa module** | | | | | | |
| TX | 2.3 s | 3.3 V | 128 mA | 422.4 mW | 0.269,9 mW h | DR0 (1) |
| RX1 | 0.2 s | 3.3 V | 22.2 mA | 73.3 mW | 0.004,1 mW h | |
| RX2 | 0.08 s | 3.3 V | 22.2 mA | 73.3 mW | 0.001,6 mW h | |
| TX | 0.308 s | 3.3 V | 128 mA | 422.4 mW | 0.036,1 mW h | DR3 (2) |
| RX1 | 0.08 s | 3.3 V | 22.2 mA | 73.3 mW | 0.001,6 mW h | |
| RX2 | 0.08 s | 3.3 V | 22.2 mA | 73.3 mW | 0.001,6 mW h | |
| **TEROS 21** | | | | | | |
| Power-up | 0.1 s | 5 V | 10 mA | 50 mW | 0.001,4 mW h | DDI serial (3) |
| Meas. Durat. | 0.2 s | 5 V | 10 mA | 50 mW | 0.002,8 mW h | |
| **TEROS 11** | | | | | | |
| Power-up | 0.1 s | 5 V | 16 mA | 80 mW | 0.002,2 mW h | DDI serial (3) |
| Meas. Durat. | 0.15 s | 5 V | 16 mA | 80 mW | 0.003,4 mW h | |
| **HDC2080 (Temp.&Hum. sensor)** | | | | | | |
| RH meas. | 660 ns | 3.3 V | 891 nA | 2.94 mW | 0.000,5 nW h | (4) |
| Temp. meas. | 610 ns | 3.3 V | 730 nA | 2.41 mW | 0.000,4 nW h | |

**Table 2.4:** Worst-case consumptions.

(1) Max. Power, so Data Rate 0 (DR0) in the experimental case of full packet payload composed by 33 bytes. Worst case values.
(2) Max. Power, so Data Rate 3 (DR3) in the experimental case of full packet payload composed by 33 bytes. Worst case values.
(3) Considered maximum consumption. Worst case values.
(4) Worst case, one-shot measurement and considering both 14 bit of resolution. Neglected in following computations.

It can be computed the energy consumed in standby time:

$$\begin{aligned} E_{standby} &= I_{standby} \cdot V_{standby} \cdot t_{standby} = \\ &= 2.65\,\mu\text{A} \cdot 3.3\,\text{V} \cdot 3{,}594.4\,\text{s} = 8.7\,\mu\text{W h}; \end{aligned}$$

It should be considered the worst-case, considering a LiSOCl2 battery. Consider Figure 2.30 taken by Saft datasheet[10].

At a typical temperature as 20 °C, when a current of 40 mA is drained (an average value in the energy-intensive task of LoRa transmission/reception), it can be assumed real energy equal to 1.5 A h, respect nominal energy density of 2.6 A h. This means

$$\begin{aligned} E_{battery} &= C_{battery} \cdot V_{battery} = \\ &= 1.5\,\text{A h} \cdot 3.6\,\text{V} = 5{,}400\,\text{mW h}; \end{aligned}$$

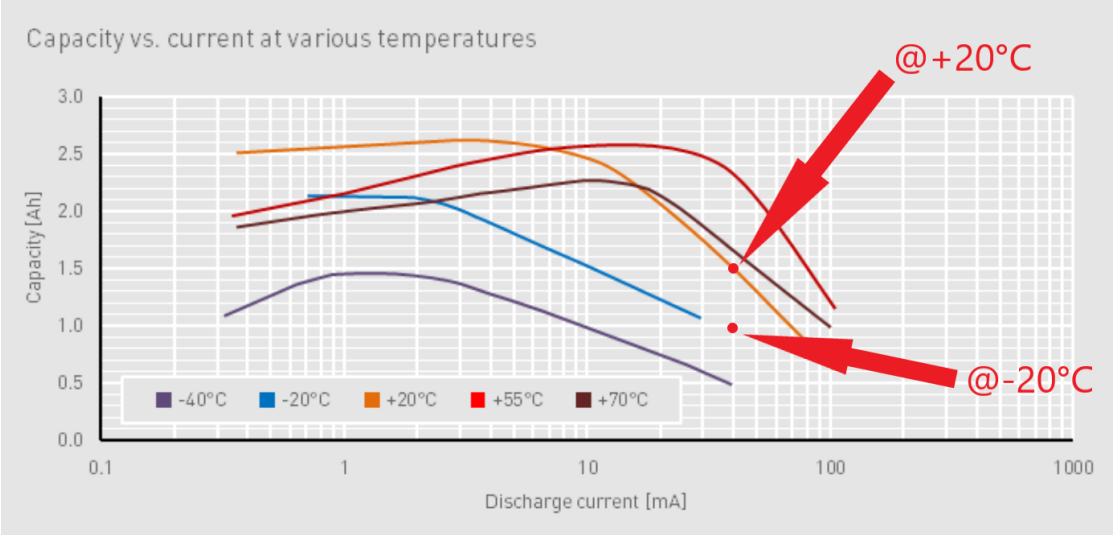



**Figure 2.30:** Current capacity versus discharged current in function of the temperature (Source [10]).

It can be possible to compute the total energy spent in one hour time:

$$E_{1-hour} = E_{runtime} + E_{standby} = \\ = 305\,\mu\text{W h} + 8.7\,\mu\text{W h} = 314\,\mu\text{W h} = 0.314\,\text{mW h};$$

So, how many times can be stay in $E_{battery}$?

$$t_{lifetime} = \frac{E_{battery}}{E_{1hour}} = \\ = \frac{5{,}400\,\text{mW h}}{0.314\,\text{mW h}} = 17{,}197\,\text{h} \approxeq 1.96 \;\; years;$$

The same computation can be performed in other remarkable practical situations as at $-20\,°\text{C}$ (imagine that it should be computed the lifetime in winter) or at laboratory conditions where a higher data rate occurs (DR3).
Table 2.5 is presented to consider these conditions.

The laboratory conditions are irrealistic due to the fact that gateway and node are placed in the same room, without obstacles so, a line-of-sight path occurs.
On the contrary, also using Data Rate (DR0) all the time is irrealistic: the maximum power is employed in the initialization step of the node to compute the best power quantity to reach the gateway, avoiding wasting additional energy. These are consumption in the worst-case, and they are practical only if gateway and node are far several kilometeres.
A realistic lifetime, considering a good design in the placement of gateway and

| Case | Energy drained | Capacity battery | Energy battery | Lifetime |
|---|---|---|---|---|
| **Battery at 20 °C @$I_{max-drained}$ = 40 mA with sampling time of 1 hour** | | | | |
| DR0 | 0.314 mW h | 1.5 A h | 5,400 mA h | 17,197 h = 1.96 year |
| DR3 | 0.081,4 mW h | 1.5 A h | 5,400 mA h | 66,339 h = 7.5 years |
| **Battery at −20 °C @$I_{max-drained}$ = 40 mA with sampling time of 1 hour** | | | | |
| DR0 | 0.314 mW h | 1.0 A h | 3,600 mA h | 11,465 h = 1.3 year |
| DR3 | 0.081,4 mW h | 1.0 A h | 3,600 mA h | 44,226 h = 5 years |

**Table 2.5:** Computation of realistic lifetime of the LoRa end node at various temperatures.

nodes should be taken to 3-4 years at a sampling time of 1 h.
Another important consideration is related to sampling time: as it can be showed by previous calculus, the contribution of few seconds in runtime are greater than the contribution of the system in standby (in DR0 of several orders of magnitude too). Moreover, it is easy to demonstrate that a sampling time of 30 min practically halves the lifetime of the system, where instead, a doubled sampling time (2 h) doubles the lifetime. For this reason, it is important to choose a sampling time that allows appreciating whatever changes on the field but avoiding useless measurements that led to a reduced lifetime.

# Chapter 3

# Firmware Implementation

## 3.1 Introduction

The first part of the experimental project is to implement every functionality on software in such a way as to verify the feasibility of the working procedure.
Some points should be clarified before talking about the embedded system.
Every peripheral has two or three ways to exchange data between peers: **Polling** mode, **Interrupt** mode, **DMA** mode. The first two ones are common for all peripherals: polling mode provides a continuous check of the sampled peripheral based on timebase generation of the microcontroller (typically, as default, it is 1 ms). Instead, in interrupt mode, a routine inside the peripheral alarms the core when a previously configured task occurred.
It is clear that the polling mode is a waste of work and energy if the peripheral is used only in some conditions. In the other case, interrupt is useful when the peripheral offers critical application: when a task on the peripherals occurred, the main application is frozen, waiting for the interrupt routine to terminate. When it is completed, an interrupt is sent to the core to continue the execution of the main application.
The last operation mode is called DMA, and it is present in some high-speed peripherals (as UART, USART, Timers), offering the best data transmission throughput, thanks to the direct access of the peripheral to MCU internal RAM. In this way, it is avoided a lot of overhead work by MCU in the context switching, present in the interrupt case. The MCU is only alarmed when a data transfer is completed.
In this application, no DMA is required due to the low throughput is needed. Practically, all used peripherals are configured in polling mode, leaving interrupt tasks only to the LoRa stack-connected tasks (RTC and some DIO lines).
In addition to this hardware implementation, there is the choice of a real-time Operative System (RTOS) or not: the traditional approach is a basic listing with

some supporting libraries to perform a task. But when tasks of a certain firmware are various and numerous, it is needed an underlying layer to allow concurrent activities. Moreover, it is introduced the concept of **Thread**: you can subdivide specific operations within a single application into individual threads sharing the same memory address space.
They offer advanced synchronization primitives, which allow both to coordinate the simultaneous access to physical resources from different threads and to avoid wasting CPU cycles while waiting for slower and asynchronous events.
When the complexity of firmware becomes important, an RTOS becomes vital. In the case of the STM32 microcontroller, many RTOS systems could be chosen, and the official tool for the CubeHAL framework is FreeRTOS.
At the first stage of the project, an RTOS should be pretty useless because only simple tasks are performed in the entire working cycle of the embedded system. A future version of the embedded system, for example, integrating some kind of Machine Learning (ML) technique to interpret sensed data or to perform some actuation task, should require consideration in the usage of an RTOS system.
Let's introduce the programmed firmware for the embedded system: when you use a specific protocol, typically, it is provided, by the manufacturer, a hierarchical supporting library to handle some ISO/OSI layers of that protocol. And this is the case of the LoRa protocol: ST Microelectronics provides a library (usually called **Stack** because it is a hierarchical "heap" of libraries to handle commands and timing for all designed protocol layers) called I-CUBE-LRWAN[23].
In that library is explained that all LoRa tasks are clocked by the configured RTC calendar. The user should only specify the time between a packet and the following one. So, every underlying layer handles this condition in such a way as to respect all ETSI rules and to control the initialization with the Adaptive Data Rate.
For this reason, RTC cannot be stopped for whatever reason. It is only possible to configure an additional alarm to wake up the machine from a standby mode.
In Figure 3.1 is reported the block diagram of the working cycle.
It can be highlighted three main steps in LoRa plant monitoring firmware: **Initialization**, **Loop and LoRa packaging**, and **LoRa parsing** of the received packet from a gateway. They are explained in Chapters 3.3, 3.4, and 3.5.

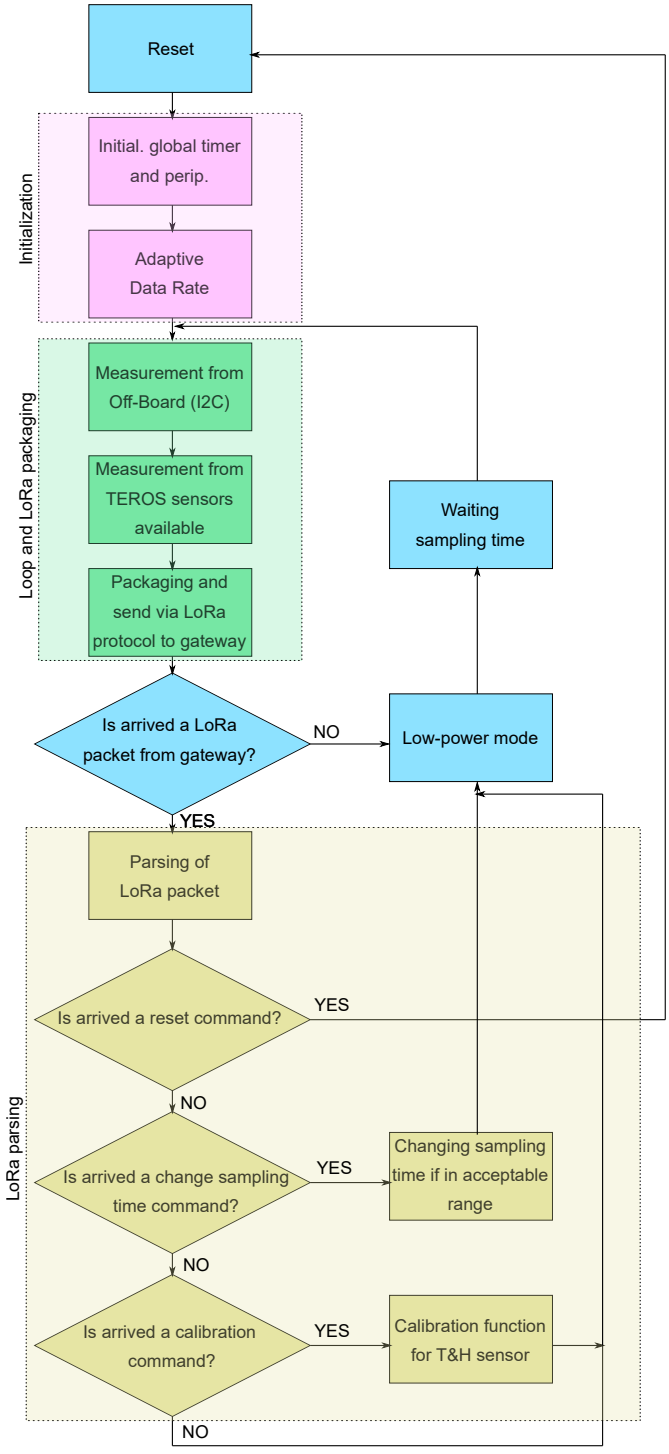



**Figure 3.1:** LoRa end node packed payload.

## 3.2 Discovery Board

Typically, a microcontroller's manufacturer provides a development board to test a working firmware before deploying it into a custom board. For the Murata module CMWX1ZZABZ, ST Microelectronics provides the B-L072Z-LRWAN1 Discovery kit. This board includes an RF interface, LEDs, push buttons, antenna, Arduino Uno V3 connectors, and a USB connector to easily supply the board. In addition, there is also an integrated ST-LINK/V2, an embedded in-circuit debugger and programmer for the STM32L0 MCUs.
The LoRa stack is class A certified, and it is included in I-CUBE-LRWAN firmware package.
The same board could also be used for a Sigfox embedded system, easily loading a different firmware package (X-CUBESFOX expansion package).
For this agriculture application, it has been used, in the first stage, to test all sensors together as a breadboard to allow connection between the discovery and sensors. After this, it has been used as a programmer for the custom board. This is easily performed working on a specific jumper on it, to drive data on the integrated SWD connector (to a custom board), or on the embedded Murata module.

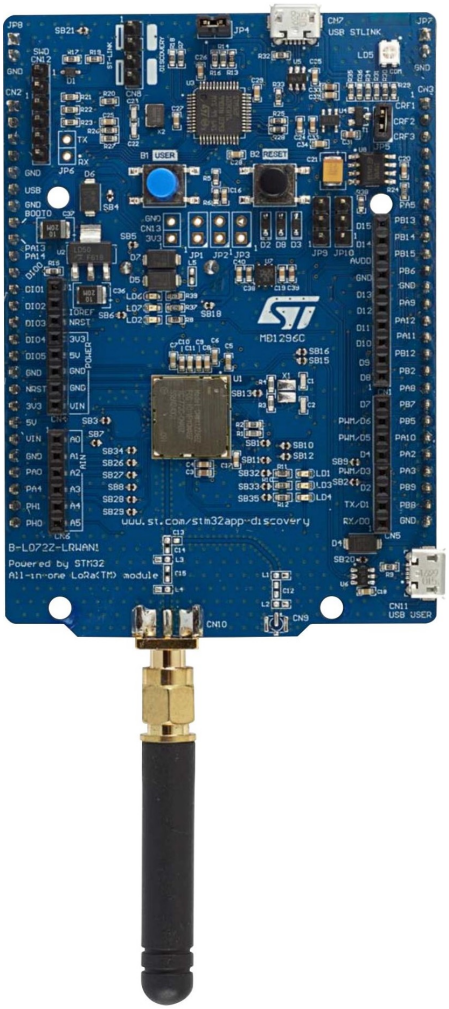



**Figure 3.2:** B-L072Z-LRWAN1 LoRa/Sigfox Discovery kit.

## 3.3 Initialization

Listing 3.1 shows the first lines of the main function.
The block diagram of Figure 3.1 is related to the purple section.

**Listing 3.1:** Initialization code

```
int main(void)
{
    HAL_Init();      /* STM32 HAL library initialization*/
    SystemClock_Config(); /* Configure the system clock*/
    DBG_Init(); /* Configure the debug mode*/
    HW_Init();  /* Configure the hardware*/

    //Configure all pins related to load switches
    __HAL_RCC_GPIOB_CLK_ENABLE();      /* GPIO Ports Clock Enable */
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
    ...

    HAL_Delay(3000);
    LPM_SetOffMode(LPM_APPLI_Id, LPM_Disable);//Disable Stand-by mode

    MX_I2C1_Init(); //I2C1 INITIALIZATION
    //Variables to obtain temperature and humidity from HDC2080
    sensor library
    float temperature=0, humidity=0;
    MX_USART1_UART_Init();//    TEROS UART init
    //Initialization of struct data_payload variables
    data_payload_runtime.sensors_active_on = 0x00;
    data_payload_runtime.config_done = 0x00;
    data_payload_runtime.soil_moisture1 = 0xFFFFFFFF;
    data_payload_runtime.temperature_soil1 = 0xFFFF;
    ...

    /* Configure the Lora Stack*/
    LORA_Init(&LoRaMainCallbacks, &LoRaParamInit);
    LORA_Join();
    InitAnalogMUX(hi2c1);    //init mux

    //check if the value of startTxTimeout is stored in the EEPROM
    uint32_t timeout = 0;
    timeout = readTimeout();
    if((timeout>=10000)&&(timeout<=3600000)){    //3.600.000 = 1h
        PRINTF("Used custom timeout value: %d ms \n\r",timeout);
        startTxTimeout = timeout;
    }
    else{
        PRINTF("Used default timeout value\n\r");
        startTxTimeout = APP_TX_DUTYCYCLE;
    }

    LoraStartTx(TX_ON_TIMER);    //start RTC

    //Init timer for periodic reset (30 days)
    TimerInit(&TxTimeoutPeriodicReset, OnTimerPeriodicResetEvent);
```

```
48      //reset each 30 days (2.592.000.000 ms)
49      TimerSetValue(&TxTimeoutPeriodicReset, 2592000000);
50      TimerStart(&TxTimeoutPeriodicReset);
51      ...
52  }
```

The code is rather self-explaining; there are many functions to initialize HAL functions, SPI related to the embedded LoRa transceiver, user-activated GPIOs (lines 9-10), I2C peripheral (line 19), LoRa stack (lines 28-29), and the initial configuration of the analog MUX via I2C protocol (line 30).
HAL (Hardware Abstraction Layer) function is a fundamental pack of libraries provided by the manufacturer to configure and use every peripheral of the microcontroller without using the low-level bit configuration. Every function of the firmware uses HAL functions (an example is line 10, wherein a single line of code is possible to configure all aspects of a GPIO).
Before starting the timer (line 44), it has been implemented a check (lines 33-42) if a sampling time is memorized in the embedded EEPROM. If not stored, it has been used as a default value. In addition, a simple check verifies that values have not some strange values (below 10 sec or above 1 h).
Finally, an alarm is programmed every 30 days: this is performed because the internal register of the RTC will overflow over 32 bits. And this leads to unpredictable behavior. When this alarm is triggered, a simple routine (OnTimerPeriodicResetEvent) performs a software reset (lines 47-50).
The following part of the initialization code (Listing 3.2) is related to the practical implementation of the Adaptive Data Rate that is performed on a dummy packet. This is done because the runtime LoRa packet is quite big (also 33 bytes). Moreover, it is better to send before small packets (few bytes) in such a way as not to have enormous waiting time (waiting time depends on airtime as shown in Chapter 1.1.1 and the airtime itself depends on the dimension of the final uplink LoRa packet).

**Listing 3.2:** Initialization code (Dummy packet section)

```
1  int main(void)
2  {
3      while(condition){    //condition where the while condition does
       some cycle
4                  //with a dummy packet (AppData_init)
5          if (LORA_JoinStatus() != LORA_SET)
6          {
7              /*Not joined, try again later*/
8              LORA_Join();
9          }
10
11         if ((AppProcessRequest == LORA_SET)
12             &&
13             (LORA_JoinStatus() == LORA_SET))
```

```
14          {
15              /*reset notification flag*/
16              AppProcessRequest = LORA_RESET;
17              /*Send*/
18              LORA_send(&AppData_init,
19                                  LORAWAN_DEFAULT_CONFIRM_MSG_STATE);
20          }
21
22          if (LoraMacProcessRequest == LORA_SET)
23          {
24              /*reset notification flag*/
25              LoraMacProcessRequest = LORA_RESET;
26              LoRaMacProcess();
27          }
28
29          /*If a flag is set at this point, mcu must not enter low
30          power and   must loop*/
31          DISABLE_IRQ();
32
33          /* if an interrupt has occurred after DISABLE_IRQ, it is kept
34          pending and cortex will not enter low power anyway  */
35          if ((LoraMacProcessRequest != LORA_SET)
36          &&
37          (AppProcessRequest != LORA_SET))
38          {
39              #ifndef LOW_POWER_DISABLE
40              LPM_EnterLowPower();
41              #endif
42          }
43          ENABLE_IRQ();
44      }
45      ...
46  }
```

## 3.4 Loop

Listing 3.3 shows the main code of the loop function.
The block diagram of Figure 3.1 is related to the first two blocks of the green section.

**Listing 3.3:** Loop code

```
1  while (1)
2  {
3      if (AppProcessRequest == LORA_SET){
4          AppProcessRequest = LORA_RESET; /*reset notification flag*/
5          temperature = GetTemperature(hi2c1);     //temperature
```

```
6           if(temperature==0xFFFFFFFF){
7               data_payload_runtime.temperature_env = 0xFFFF;
8           }
9           else{
10              data_payload_runtime.temperature_env = temperature*10;
11          }
12          humidity = GetHumidity(hi2c1);  //humidity
13          if(humidity==0xFFFFFFFF){
14              data_payload_runtime.humidity_env = 0xFFFF;
15          }
16          else{
17              data_payload_runtime.humidity_env = humidity*10;
18              data_payload_runtime.sensors_active_on =
19                  data_payload_runtime.sensors_active_on | 0b01000000;
20          }
21          //TEROS21
22          SelectInput(hi2c1,0b00100000);  //select CH1 at the output of
                                              analog MUX
23          HAL_Delay(100);
24          d = GetMeasurement_TEROS21(&huart1,&data_payload_runtime,
25                                                          0b00100000);

26          HAL_Delay(100);
27          RestPosition(hi2c1);
28          HAL_Delay(100);
29          if(d!=1){
30              data_payload_runtime.soil_moisture1 = 0xFFFFFFFF;
31              data_payload_runtime.temperature_soil1 = 0xFFFF;
32          }
33          else{
34              data_payload_runtime.sensors_active_on =
35                  data_payload_runtime.sensors_active_on | 0b00100000;
36          }
37          //Same procedure for all channels
38          ...
39          //When all channels are sampled
40          /*Send*/
41          Send(NULL,&data_payload_runtime);
42      }
43      if (LoraMacProcessRequest == LORA_SET){
44          LoraMacProcessRequest = LORA_RESET;/*reset notification flag
        */
45          LoRaMacProcess();
46      }
47      /*If a flag is set at this point, mcu must not enter low power
        and must loop*/
48      DISABLE_IRQ();
49      /* if an interrupt has occurred after DISABLE_IRQ, it is kept
50      *pending and cortex will not enter low power anyway  */
```

```
51      if ((LoraMacProcessRequest != LORA_SET)
52      &&
53      (AppProcessRequest != LORA_SET)){
54          #ifndef LOW_POWER_DISABLE
55          LPM_EnterLowPower();
56          #endif
57      }
58      ENABLE_IRQ();
59  }
```

This is the content of the loop cycle. When a request from RTC LoRa timing is triggered, a series of measurements are performed.
In the beginning, HDC2080 (placed on an off-board connector) is tested via I2C peripheral to perform a measurement and get data (first temperature and after humidity data). If the sensor is not inserted, so I2C peripheral responds error, the relative temporary variable (temperature, humidity) are fixed to a specific value (0xFFFFFFFF) in such a way that the following if-condition could flag the corresponding bit among activated sensors (Lines 5-20).
The same thing is done for each channel connected to the UART peripheral of the microcontroller: a bit is flagged if the communication is established and data are collected (Lines 22-36).
After these tasks, the function Send is invoked: all collected data are proper organized in the LoRa payload. When the LoRa packet is sent, the machine comes back to sleep.

## 3.5 LoRa Packaging (Tx) and Parsing (Rx)

When data are collected, a proper handler is invoked because each type of variable must be subdivided in byte-size packets starting from MSB byte.
Moreover, a 4 bytes variable, for example, Int or Float, should be divided in the LoRa buffer in 4 pieces. But, it is possible to send only pieces that contain useful information. For example, the soil matric potential has a range value in the three LSB bytes, so there only are sent them.
In Figure 3.3 is shown the designed order of sent bytes if all sensors are connected. Data are divided into four categories: configuration data, off-board data, TEROS21 data, and TEROS11 data.
The LoRa packet always starts with configuration data composed by the activated sensors: in this way, it is possible to perform an if-condition on the cloud platform to show or not the corresponding variables referred to that sensor.
The second byte is anyway important: it is in charge of signaling if a previously byte command has been performed correctly.
After this, all remaining bytes are sensor data so, if the corresponding bit on

the sensor_active byte is "1", corresponding variables are shown on the console interface.

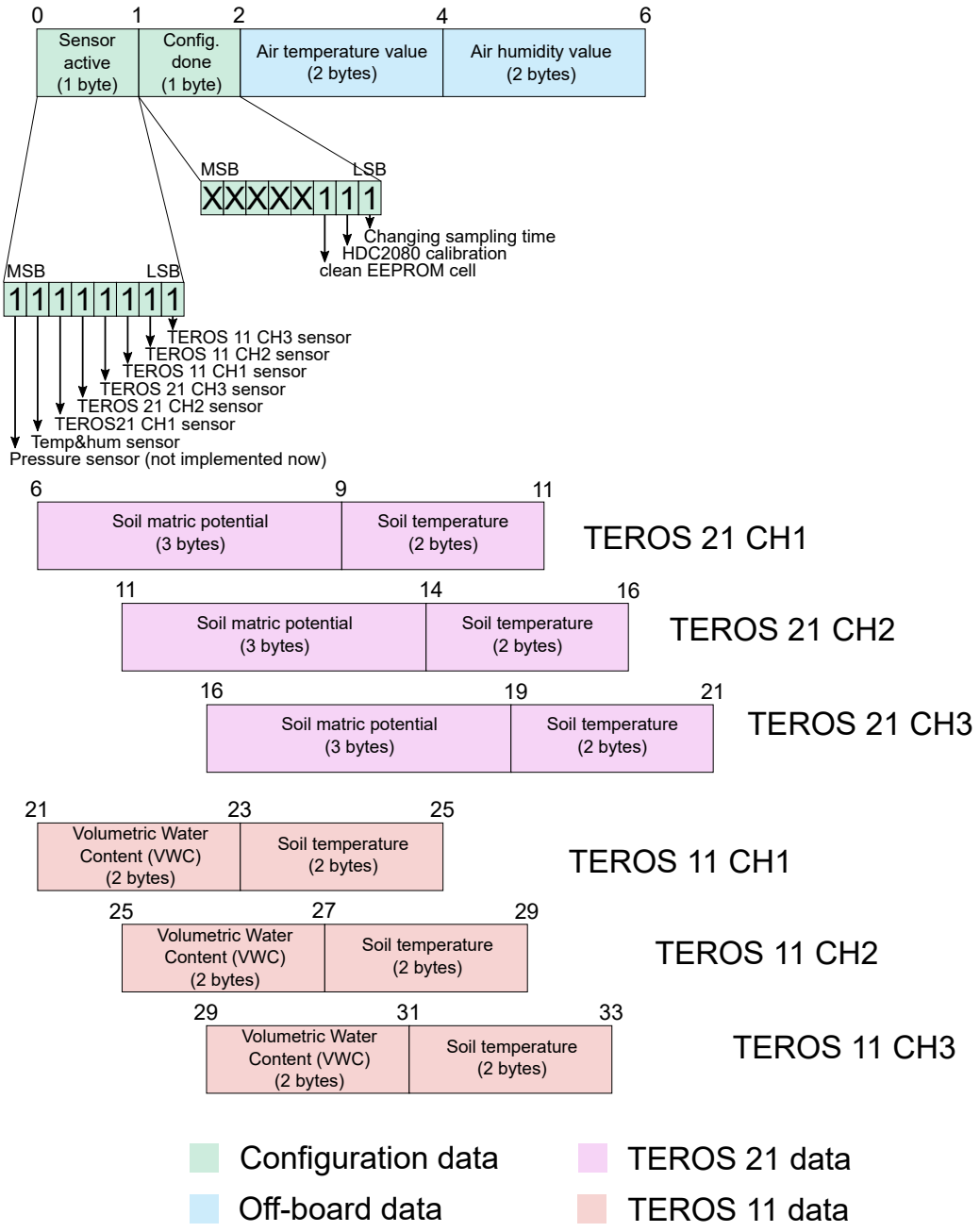



**Figure 3.3:** LoRa end node packed payload.

Listing 3.4 shows how it works the function related to the creation of the working LoRa data payload.
Current data are contained in the struct data_payload. If LoRa connection is established, AppData struct (struct required from LoRa stack) must be filled indicating port (line 8), buffer size (line 47) computed it for each byte inserted in the buffer, and, obviously, the buffer itself that contains all bytes to send.
As it is possible to see, it is inserted two configuration bytes with current data. Using that byte mask, a sequence of if-condition check all bits of the mask, append that data on the current buffer.
Note that all data bigger than one byte are parsed shifting as multiple of 8 to insert MSB useful byte on the buffer. This is repeated until all bytes are appended in the buffer (example from line 31 to line 39).
In the end, when the buffer is filled, the LORA_send function is invoked passing data_payload struct as illustrated before and kind of message type (confirmed or not confirmed uplink message).

**Listing 3.4:** LoRa uplink packaging

```
1  static void Send(void *context, data_payload *data_payload_runtime){
2      if (LORA_JoinStatus() != LORA_SET){
3          /*Not joined, try again later*/
4          LORA_Join();
5          return;
6      }
7      uint32_t i = 0;
8      AppData.Port = LORAWAN_APP_PORT;
9      //Sensor active on byte
10     AppData.Buff[i++] =
11                 data_payload_runtime->sensors_active_on & 0xFF;
12     //configuration done byte
13     AppData.Buff[i++] = data_payload_runtime->config_done & 0xFF;
14     if(data_payload_runtime->config_done != 0){
15         data_payload_runtime->config_done = 0x00;
16     }
17     //HDC2080 data (I2C)
18     if((data_payload_runtime->sensors_active_on&0b01000000)!=0){
19         AppData.Buff[i++] =
20             (data_payload_runtime->temperature_env >> 8) & 0xFF;
21         AppData.Buff[i++] =
22             data_payload_runtime->temperature_env & 0xFF;
23         AppData.Buff[i++] =
24             (data_payload_runtime->humidity_env >> 8) & 0xFF;
25         AppData.Buff[i++] =
26             data_payload_runtime->humidity_env & 0xFF;
27     }
28     //TEROS21 sensors (CH1)
29     if((data_payload_runtime->sensors_active_on&0b00100000)!=0){
```

```
30          AppData.Buff[i++] =
31              (data_payload_runtime->soil_moisture1 >> 16) & 0xFF;
32          AppData.Buff[i++] =
33              (data_payload_runtime->soil_moisture1 >> 8) & 0xFF;
34          AppData.Buff[i++] =
35              data_payload_runtime->soil_moisture1 & 0xFF;
36          AppData.Buff[i++] =
37              (data_payload_runtime->temperature_soil1 >> 8) & 0xFF;
38          AppData.Buff[i++] =
39              data_payload_runtime->temperature_soil1 & 0xFF;
40      }
41      //if condition for all remaining channels
42      //(both TEROS21 and TEROS11)
43      ...
44      if(data_payload_runtime->sensors_active_on != 0){
45          data_payload_runtime->sensors_active_on = 0x00;
46      }
47      AppData.BuffSize = i;
48      LORA_send(&AppData, LORAWAN_DEFAULT_CONFIRM_MSG_STATE);
49  }
```

On the contrary, a function called LORA_RxData is invoked when whatever payload is sent to the end node in one of the two RX windows as specified in the LoRa class-A specification.
In this application, two are the possibilities: on port 1 (fixed value), one byte or three bytes can be received.
If one byte is received, it means that a simple command is required except changing of sampling time. On the contrary, when three bytes are received, the changing of sampling time is invoked.
This control is performed by if(AppData->BuffSize == x).
At this point, it is controlled which bits are high in the buffer (Buff): each bit is related to a simple command as clean used EEPROM cells, perform a system reset, perform a calibration of HDC2080 or change the sampling time.
When we are in this latter case (3 bytes received), two append bytes are parsed to set a new alarm in the RTC. In addition, the same value is written in the EEPROM to save this sampling time for whatever additional reset. After, when it is needed to perform a clean of this value from EEPROM, you can activate the cleanUsedCells() function.

**Listing 3.5:** LoRa downlink parsing

```
1  static void LORA_RxData(lora_AppData_t *AppData)
2  {
3      switch (AppData->Port)
4      {
5          case 1:
6          {
```

```
7                 if(AppData->BuffSize == 1){
8                     if(AppData->Buff[0] & 0b00001000){
9                         if(cleanUsedCells() == 0){
10                            Error_Handler();
11                        }
12                        //Clean timeout value in EEPROM region
13                        data_payload_runtime.config_done =
14                        data_payload_runtime.config_done | 0b00000100;
15                    }
16                    if(AppData->Buff[0] & 0b00000100){
17                        PRINTF("RESET\r\n");
18                        HAL_NVIC_SystemReset();
19                    }
20                    if(AppData->Buff[0] & 0b00000010){
21                        //add set bit for a configuration of HDC2080
22                        flag_i2c = flag_i2c | 0b00000010;
23                    }
24                }
25
26                if(AppData->BuffSize == 3){
27                    if(AppData->Buff[0] & 0b00001000){
28                        if(cleanUsedCells() == 0){
29                            Error_Handler();
30                        }
31                        //Clean timeout value in EEPROM region
32                        data_payload_runtime.config_done =
33                        data_payload_runtime.config_done | 0b00000100;
34                    }
35                    if(AppData->Buff[0] & 0b00000100){
36                        PRINTF("RESET\r\n");
37                        HAL_NVIC_SystemReset();
38                    }
39                    if(AppData->Buff[0] & 0b00000010){
40                        //add set bit for a configuration of HDC2080
41                        flag_i2c = flag_i2c | 0b00000010;
42                    }
43                    if(AppData->Buff[0] & 0b00000001){
44                        uint32_t time_sendData = (AppData->Buff[1] << 8)
45                                                 | (AppData->Buff[2]);
46                        if(writeTimeout(time_sendData*1000) == 1){
47                            //after write on EEPROM,
48                            //update the value in the RTC timer
49                            TimerSetValue(&TxTimer,  time_sendData*1000);
50                            OnTxTimerEvent(NULL);
51                            data_payload_runtime.config_done =
52                            data_payload_runtime.config_done | 0b00000001
    ;
53                        }
54                    }
```

```
55                 }
56                 break;
57             }
58             default:
59                 break;
60         }
61 }
```

## 3.6 MAX4562 Library

For the analog MUX MAX4562, a proper library has been developed. Being in practice, a basic finite state machine, the only thing to do it is to configure it in the proper way (hard mode). In Chapter 2.3.11, it has been indicated all fundamental information to comprehend how it works this integrated circuit.
Moreover, three C functions have been developed to command this chip and close the right switch at the right time: InitAnalogMUX to configure the analog MUX as all open switches and hard switching mode, SelectInput to close a specific switch shortcircuiting it to the UART line, RestPosition to open all switches.
Listing 3.6 shows this code.
It is an almost self-explaining code: each HAL_I2C_Master_Transmit sends a byte command to the MAX4562 at the I2C address 0b10011000 (0x98).
In SelectInput, a parameter byte channel_bit is required to select a switch to close.

**Listing 3.6:** Analog MUX (MAX4562) library

```
1  void InitAnalogMUX(I2C_HandleTypeDef hi2c){
2      //It can be used also as Identifier function
3      HAL_StatusTypeDef ret;
4      uint8_t switchset_all_opens = 0x80; //0b10000000 command
5                                          //SWITCHSET to open all switches
6      uint8_t modeset_all_hard = 0xC0;     //0b11000000 command
7              //MODESET to apply hard switching mode to all switches
8      ret = HAL_I2C_Master_Transmit(&hi2c, MAX4562_ADDR,
9                            &switchset_all_opens, 1, HAL_MAX_DELAY);
10     if ( ret != HAL_OK ) {
11         return ret;
12     }
13     ret = HAL_I2C_Master_Transmit(&hi2c, MAX4562_ADDR,
14                                   &modeset_all_hard, 1, HAL_MAX_DELAY);
15     if ( ret != HAL_OK ) {
16         return ret;
17     }
18     return;
19 }
20
21 void SelectInput(I2C_HandleTypeDef hi2c, uint8_t channel_bit){
```

```
22      HAL_StatusTypeDef ret;
23      if(channel_bit & 0b00100000){
24          RestPosition(hi2c);
25          PRINTF("SELECTED SW1A OF ANALOG MUX\r\n");
26          uint8_t select_sw1a = 0xA0; //0b10100000
27          ret = HAL_I2C_Master_Transmit(&hi2c, MAX4562_ADDR,
28                                        &select_sw1a, 1, HAL_MAX_DELAY);
29          if ( ret != HAL_OK ) {
30              return ret;
31          }
32          return;
33      }
34      if(channel_bit & 0b00010000){
35          RestPosition(hi2c);
36          PRINTF("SELECTED SW1B OF ANALOG MUX\r\n");
37          uint8_t select_sw1b = 0x90; //0b10010000
38          ret = HAL_I2C_Master_Transmit(&hi2c, MAX4562_ADDR,
39                                        &select_sw1b, 1, HAL_MAX_DELAY);
40          if ( ret != HAL_OK ) {
41              return ret;
42          }
43          return;
44      }
45      //if condition for all remaining channels (SW2A,SW2B,SW3,SW4)
46      ...
47  }
48
49  void RestPosition(I2C_HandleTypeDef hi2c){
50      HAL_StatusTypeDef ret;
51      uint8_t switchset_all_opens = 0x80; //0b10000000 command
52                                          SWITCHSET to open all switches
53      ret = HAL_I2C_Master_Transmit(&hi2c, MAX4562_ADDR,
54                                    &switchset_all_opens, 1, HAL_MAX_DELAY);
55      if ( ret != HAL_OK ) {
56          return ret;
57      }
58      return;
59  }
```

## 3.7 TEROS21 Library

Also, TEROS sensors require a proper library for parsing data obtained by UART peripheral.
Once configured the UART peripheral using provided data, a proper sequence to switch on the power supply to one single channel is performed (line 14 shows PB15 set to 1 to switch on the load switch connected to the CH1).

At this point, the HAL_UART_Receive function gets every single byte from the stream, memorizing it in a temporary variable data_RX. After a long sequence of if-condition, if the communication occurs fine, the string variable stream is filled with received data from the sensor (lines 19-52).
When communication has an error or is terminated, the power supply is switched off.
When the stream is completed, two dedicated functions parse the soil moisture and the soil temperature converting them into int variables (lines 68, 69).
Finally, these variables are inserted in the general struct used in the main procedure (lines 71, 72).

**Listing 3.7:** TEROS21 library

```
1  int GetMeasurement_TEROS21(UART_HandleTypeDef *huart1, data_payload *
       data, uint8_t channel_bit){
2      //activate power supply to selected TEROS 21
3      uint8_t data_RX;
4      //buffer length is declared considering the worst-case UART
       string
5      char stream[20];
6      char *p;
7      //switch off all channels
8      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);
9      ...
10     HAL_Delay(500);
11
12     //switch on the selected channel using mask variable channel_bit
13     if(channel_bit == 0b00100000){
14         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_SET);
15     }
16     ...
17
18     HAL_StatusTypeDef status;
19     uint8_t temp = USART1->RDR; //flush
20     HAL_Delay(10);
21     status = HAL_UART_Receive(huart1, &data_RX,1,100);
22     if(status == HAL_OK){
23         //Handle two kind of errors: timeout error in while cycle
24         //and communication error in if-else condition
25         if(data_RX == 9){
26             stream[0] = data_RX;
27             HAL_UART_Receive(huart1, &data_RX,1,100);
28             if(data_RX == 45){
29                 ...
30                 HAL_UART_Receive(huart1, &data_RX,1,100);
31                 //check if <CR> character is present
32                 if(data_RX == 13){
33                     //almost payload complete
```

```
34                      i++;
35                      stream[i] = data_RX;
36                  }
37                  else{
38                      //Error handler
39                  }
40                  //sensor type info character
41                  HAL_UART_Receive(huart1, &data_RX,1,100);
42                  i++;
43                  stream[i] = data_RX;
44                  //checksum info character
45                  HAL_UART_Receive(huart1, &data_RX,1,100);
46                  i++;
47                  stream[i] = data_RX;
48                  //CRC info character
49                  HAL_UART_Receive(huart1, &data_RX,1,100);
50                  i++;
51                  stream[i] = data_RX;
52                  //data stream terminated
53                  if(status == HAL_TIMEOUT){
54                      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15,
55                                                      GPIO_PIN_RESET);
56                      ...
57                      return -1;
58                  }
59                  HAL_Delay(100);
60
61                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15,
62                                                      GPIO_PIN_RESET);
63                  ...
64                  HAL_Delay(100);
65
66                  HAL_UART_Abort(huart1);
67                  temp = USART1->RDR;
68                  int soilMoisture = GetMatricPotential(stream);
69                  int soilTemperature = GetSoilTemperature(stream);
70                  if(channel_bit == 0b00100000){
71                      data->soil_moisture1 = soilMoisture;
72                      data->temperature_soil1 = soilTemperature;
73                      return 1;
74                  }
75                  ...
76              }
77              return -2;
78          }
```

## 3.8 Python Code

A simple python code has been realized to simplify the computation of the right basic command to send in downlink using a LoRa gateway.
This is integrally reported in Listing 3.8.
From line 4 to line 13, a user could insert desired tasks to send at the end node included changing sampling time.
In lines 24-43, these data are computed to generate hexadecimal that should be inserted in The Things Network downlink page.
For example, executing the code in Listing 3.8, a result of 0x04 is printed out. Instead, when software_reset_MCU is set to 0 and change_sampling is set to 1, a result of 0x010014 is printed out.

**Listing 3.8:** Python code to generate the hexadecimal number to create the basic command in downlink

```
1  #Python code to generate a config. payload (conf_payload) for
       LoRa_plant_monitoring (v0.1)
2  ##################################################################
3  #Compile this part to generate the right payload to send
4  time_samples = 20     # expressed in sec
5  #Config bits (1=yes, 0=no)
6  # change time sampling using time_samples data
7  change_sampling = 0
8  # perform a calibration procedure to eliminate humidity from HDC2080
       temp/hum sensor
9  calibration_HDC2080 = 0
10 #software reset on MCU
11 software_reset_MCU = 1
12 #clean cells of EEPROM from timeout value and related flag
13 clean_cells_EEPROM = 0
14 ##################################################################
15 def padhexa_3bytes(s):
16     #add zeros to have a hexadecimal number composed by 6 digits (3
       bytes)
17     return '0x' + s[2:].zfill(6)
18
19 def padhexa_1byte(s):
20     #add zeros to have a hexadecimal number composed by 2 digits (1
       bytes)
21     return '0x' + s[2:].zfill(2)
22 ##################################################################
23
24 config_byte = 0
25 if change_sampling != 0:
26     config_byte = config_byte|0b00000001
27 if calibration_HDC2080 != 0:
28     config_byte = config_byte|0b00000010
```

```
29 if software_reset_MCU != 0:
30     config_byte = config_byte|0b00000100
31 if clean_cells_EEPROM != 0:
32     config_byte = config_byte|0b00001000
33
34 if change_sampling != 0:
35     tmp = config_byte << 16
36     result = tmp | time_samples
37     #result printed at the output
38     print(padhexa_3bytes(hex(result)))
39
40 if change_sampling == 0:
41     result = config_byte
42     #result printed at the output
43     print(padhexa_1byte(hex(result)))
```

## 3.9 Decode Payload on TTN Platform

Live data from The Things Network console can show the received raw payload but, if you want to know the meaning of it is needed to write a decoding code. An easy way to do it is to write it in Javascript. In Listing 3.9 is shown the employed code based on what is programmed in the firmware code of the microcontroller. After an if-condition to select the right port (fixed port in this application), the first byte is sensor_active_on that indicates which channels have useful data. Starting from line 14 until the end, if-conditions parse data, checking if the proper bit in the sensor_active_on mask is set to 1.
In Figure 5.4 is shown an example of what it is possible to see using this code.

**Listing 3.9:** Decoder payload on TTN platform (written in Javascript)

```
1  function Decoder(bytes, port) {
2      // Decode an uplink message from a buffer
3      // (array) of bytes to an object of fields.
4      var decoded = {};
5      if (port == 2) //runtime port
6      {
7          // Decode bytes to int
8          var sensor_active_on = bytes[0];
9          decoded.sa_on = sensor_active_on;     //byte about data
10                                               //received from sensors
11         var conf_done = bytes[1];
12         decoded.c_done = conf_done;           //config_done byte
13         i = 2;
14         if((sensor_active_on & 64)!==0){     //if bit of HDC2080 is 1
15             var temperatureInt = (bytes[i] << 8) | bytes[i+1];
16             decoded.temperature = temperatureInt/10;     //Temperature
```

```
17              i=i+2;
18              var humidityInt = (bytes[i] << 8) | bytes[i+1];
19              decoded.humidity = humidityInt/10;          //Humidity
20              i=i+2;
21          }
22          if((sensor_active_on & 32)!==0){//if bit of CH1 TEROS21 is 1
23              var soil_moistureIntCH1 = (bytes[i] << 16) |
24                                        (bytes[i+1] << 8) | bytes[i+2];
25              decoded.soil_moiCH1 = -soil_moistureIntCH1/10;
26              i=i+3;
27              var temp_soilIntCH1 = (bytes[i] << 8) | bytes[i+1];
28              decoded.temp_soilCH1 = temp_soilIntCH1/10;
29              i=i+2;
30          }
31          if((sensor_active_on & 16)!==0){ //if bit of CH2 TEROS21 is 1
32              ...
33          }
34          if((sensor_active_on & 8)!==0){  //if bit of CH3 TEROS21 is 1
35              ...
36          }
37          if((sensor_active_on & 4)!==0){  //if bit of CH1 TEROS11 is 1
38              var vol_water_contentIntCH1 = (bytes[i] << 8) |
39                                                    bytes[i+1];
40              decoded.vol_water_contentCH1 =
41                                        -vol_water_contentIntCH1/1000;
42              i=i+2;
43              var temp_soil_t11IntCH1 = (bytes[i] << 8) | bytes[i+1];
44              decoded.temp_soil_t11CH1 = temp_soil_t11IntCH1/10;
45              i=i+2;
46          }
47          if((sensor_active_on & 2)!==0){  //if bit of CH2 TEROS11 is 1
48              ...
49          }
50          if((sensor_active_on & 1)!==0){  //if bit of CH3 TEROS11 is 1
51              ...
52          }
53          return decoded;
54      }
55  }
```

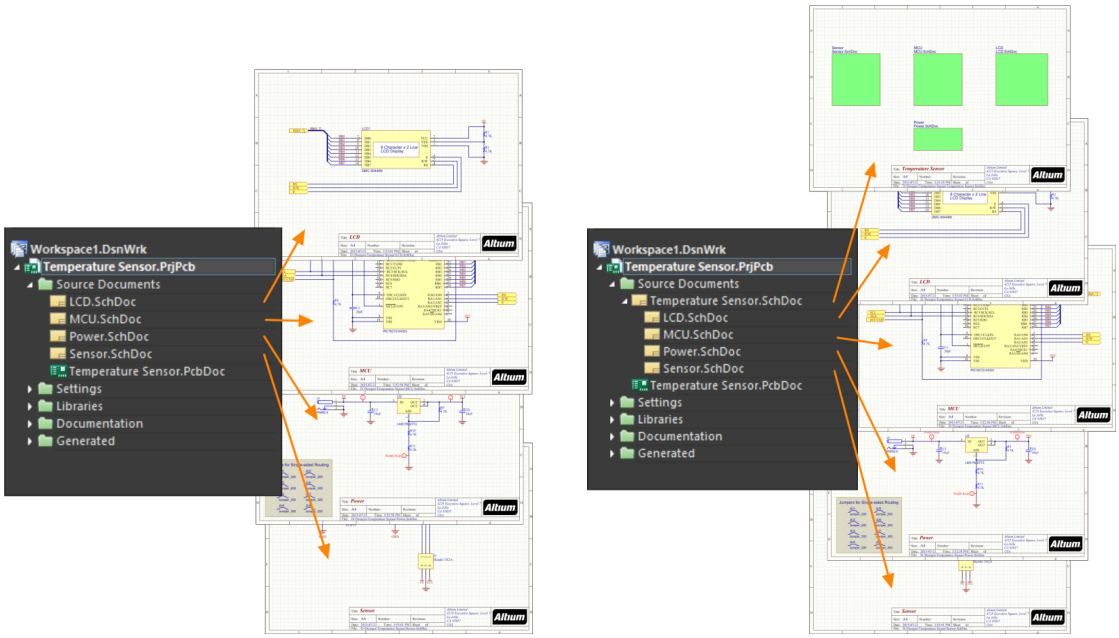
# Chapter 4

# Hardware Implementation

## 4.1 Schematics

In a hardware implementation, the first task is to realize a schematic that describes all electrical connections.
For not trivial projects, the number of sheets composing the electronic embedded system is typically more than one. For this reason, a multiple sheet project should be designed where the system should be divided into subsections where, through some logical nets, every section could talk with the other ones.
Two main approaches are possible: hierarchical or not-hierarchical (also called flat).
A **hierarchical** project (Figure 4.1b) implements one top-level sheet that has the only aim to connect each schematic sheet connecting them through some logical nets. This approach is particularly interesting when the electronic project is very complex (at least 5-10 schematic sheets), and several schematic sheets should be connected to each other.
The **flat** approach (Figure 4.1a), instead, does not have a top-level sheet, and every schematic sheet "talks" with other ones directly. A proper naming should be performed to do this: each **node** (example SWDIO in red in Figure 4.2) should be named in such a way as to identify in whatever moment each track in the following step of PCB design. When a connection should be connected from a sheet to another one, a **Port** (example DIO_LED_GRN in Figure 4.2) should be defined: a port identifies clearly an input, output, or bidirectional line among sheets.
The remaining text (in blue) are related to annotation of each component (example U1 in Figure 4.2), text to identify the value of passive components (example 10K in R1 in Figure 4.2), the model of integrated circuit (example CMWX1ZZABZ-091 in Figure 4.2) and any comment that could be useful to identify functions easily (example on PH0 net has been added OSC_IN in Figure 4.2 due to indicate that the pin can be connected to an external oscillator to the module).

Another recommendation is to sign each pin of whatever component that should not be connected (so-called floating pin) using the proper primitive, graphically expressed as red crosses (for example, in Figure 4.2, PH0 is not connected or floating). In this way, it is possible to communicate to the CAD program not to consider that pin (otherwise, it should be got an error in the Design Rule Check as an unconnected pin).
In particular cases, a certain net should guarantee some kind of rule (for example, a particular track width or a specific clearance): this is the case of an RF path shown in Figure 4.4, where some red "pinned on net" symbols appear. This is useful to force any constrain to that net. In this case, the clearance (of that net with respect to other near nets or polygon pour) is fixed to a specific value in such a way to have only one variable (the track width) that allows having a matched transmission line. Dedicated to this problem, Chapter 4.2.2 presents it.
This project has four sheets to describe the entire printed circuit board, so a flat design approach has been chosen. Figures 4.2, 4.3, 4.4 and 4.5 show the final schematics of the LoRa plant monitoring project.




**(a)** Example of flat design project. **(b)** Example of hierchical design project.

**Figure 4.1:** Comparison of two main approaches of an electronic schematic design in Altium Designer (Source here).

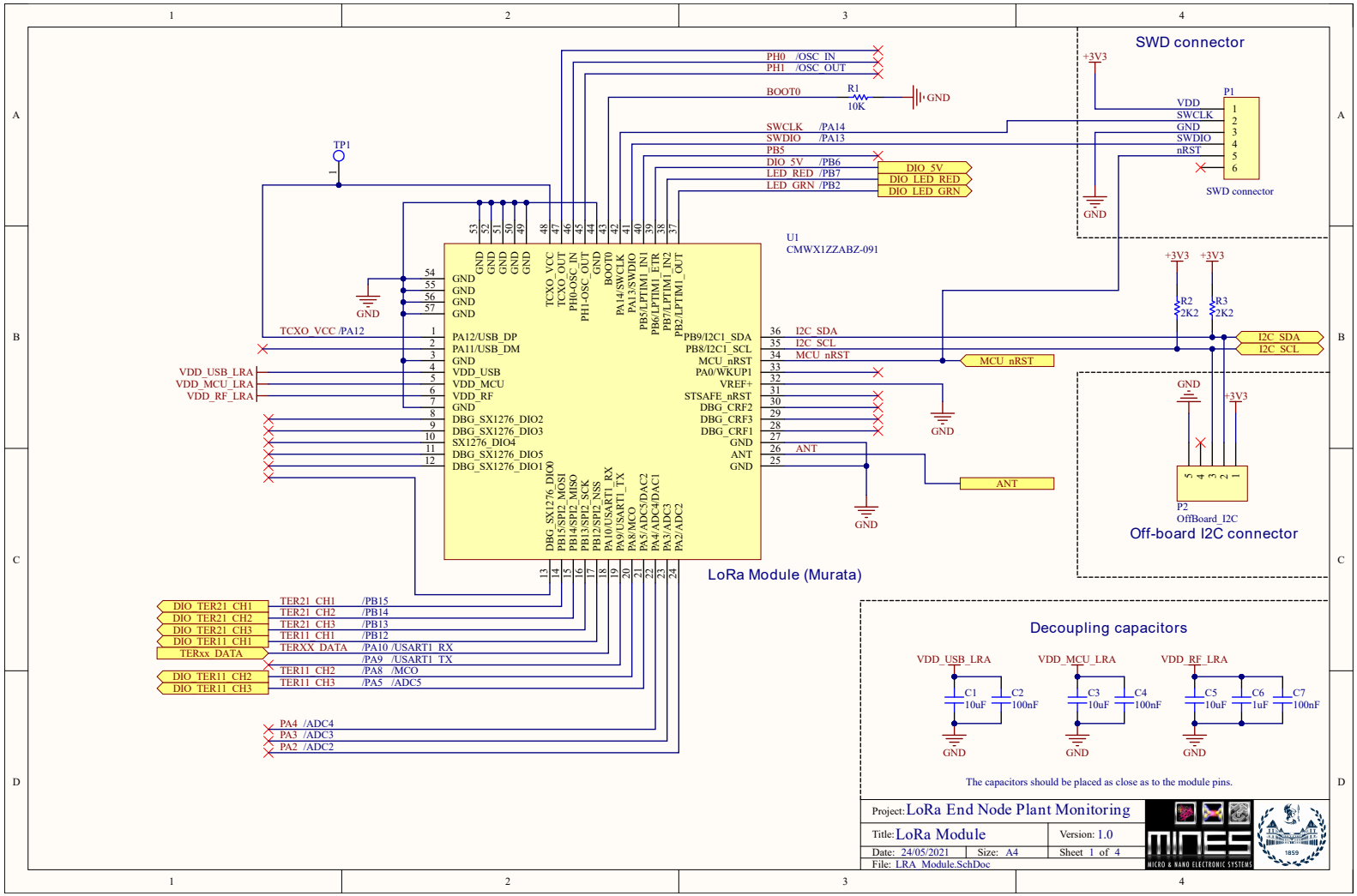



**Figure 4.2:** LoRa module schematic sheet.

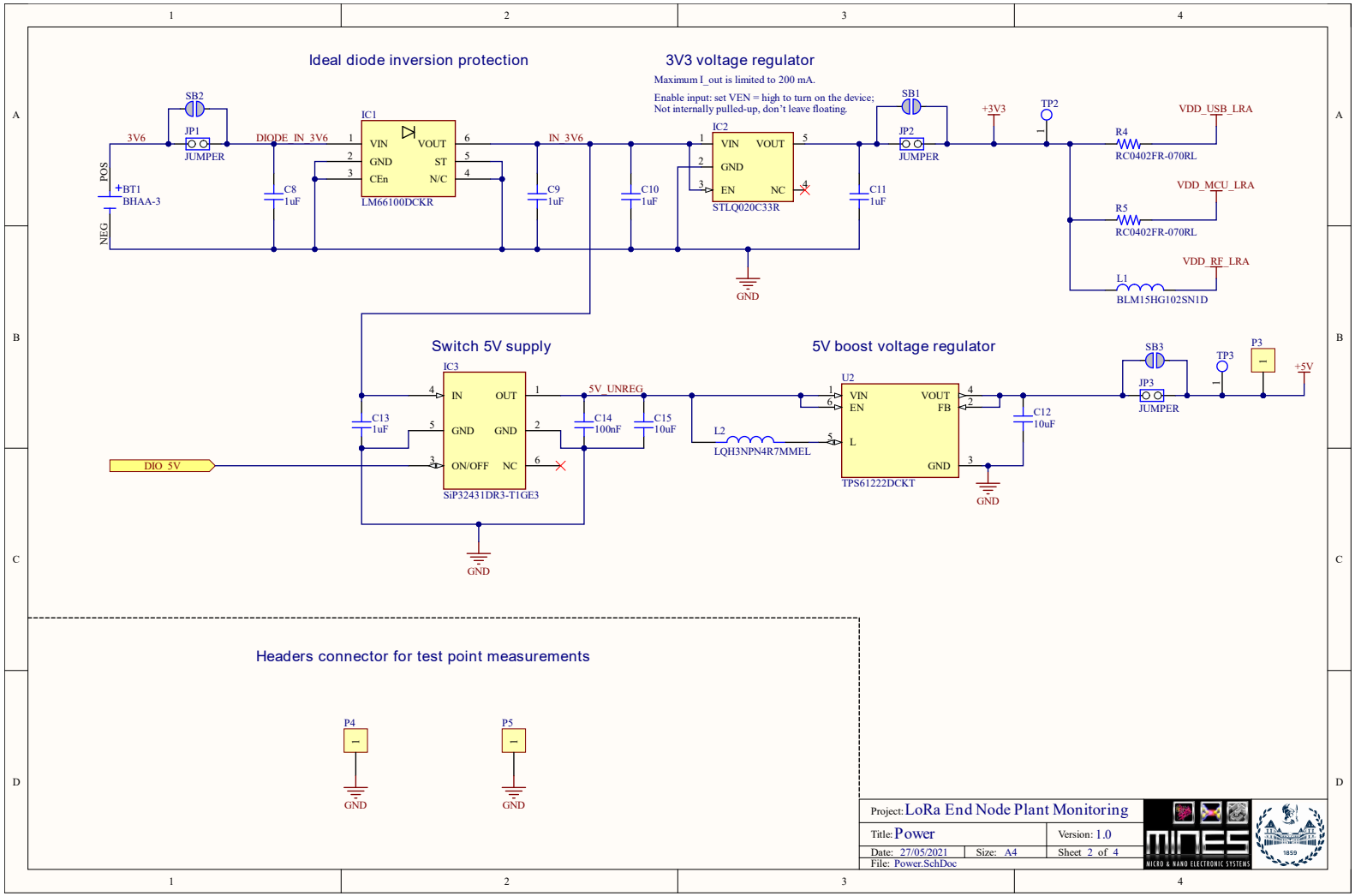



**Figure 4.3:** Power stage schematic sheet.

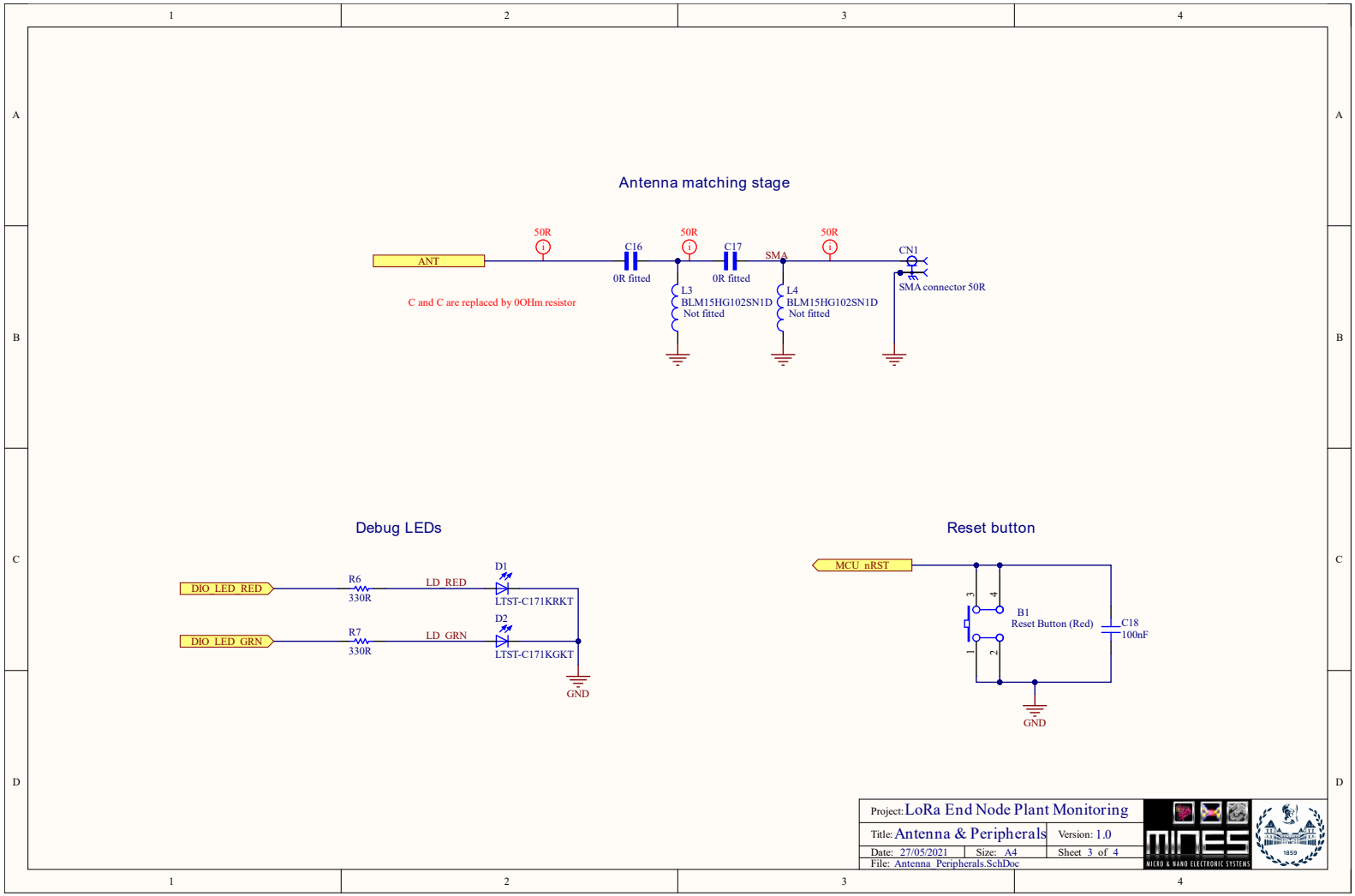



**Figure 4.4:** Antenna and peripherals schematic sheet.

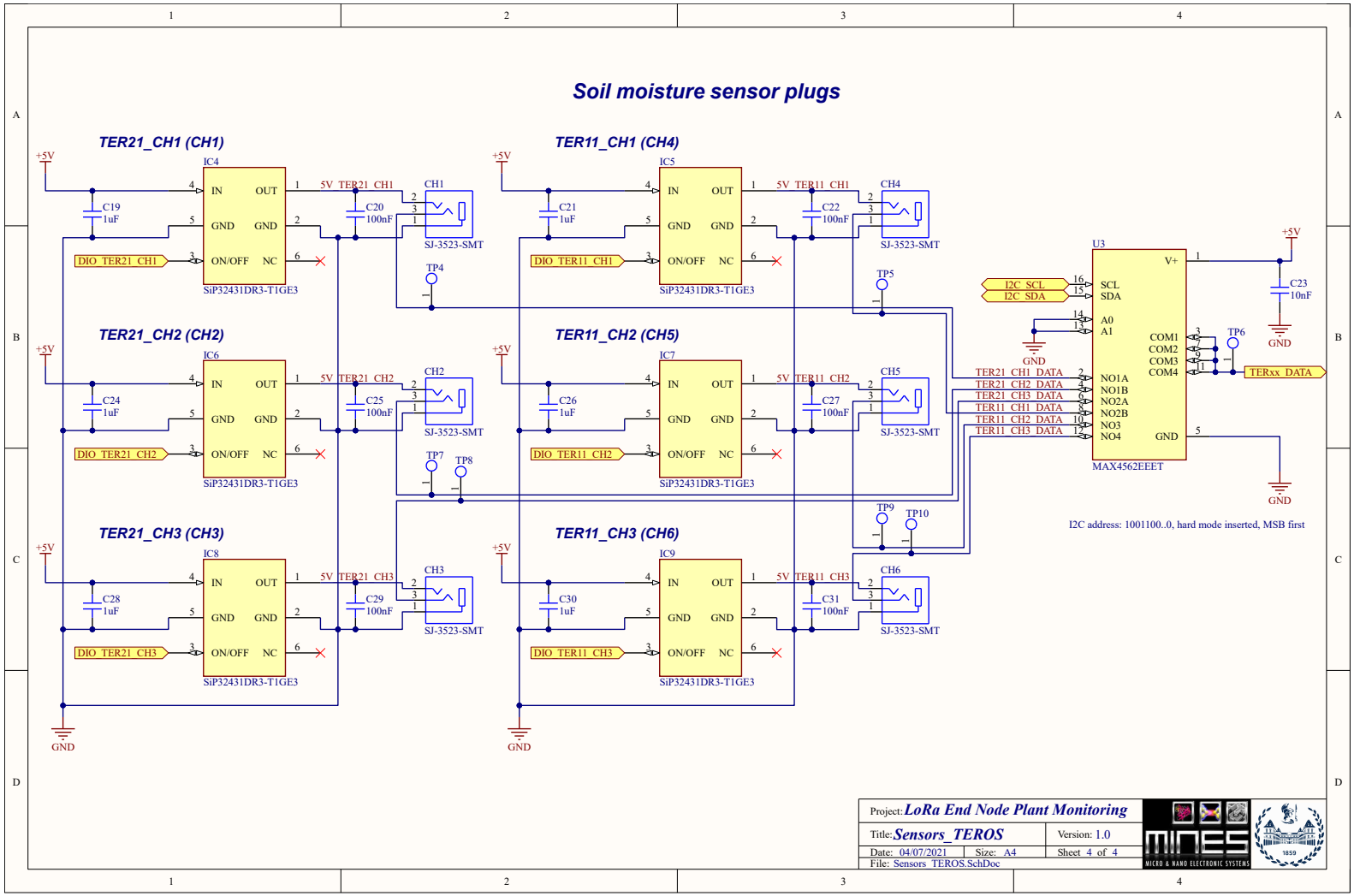



**Figure 4.5:** TEROS sensors stage schematic sheet.

## 4.2 PCB Layout

When all logical connections are issued in one or more schematics, the last step is the PCB layout design.
This stage is more complex than the previous one because it should satisfy more constraints as feasibility, geometrical constraints, physical constraints, and reliability.
For these reasons, some aspects have a dedicated following subchapter to examine more in depth what motivation is related to that PCB section.
As general aspect, it has been chosen **4 layers** PCB. A PCB, typically, starts with 2 layers and these layers are increased in number when the complexity grows up, or some specific reasons occur.
A 4 layers PCB is typically employed in RadioFrequency (RF) boards. For example, the LoRa layout guideline application note[24] shows that a 4 layers provides an additional advantage of sandwiching, between ground layers, the distributed RF decoupling of DC power trace/plane and signal bus, reducing the noise level and unwanted electromagnetic signals.
Table 4.1 reports the choices of layers arrangement.

| | | |
|---|---|---|
| Layer 1 | Top layer | General-purpose signals with a polygon pour connected to GND. |
| Layer 2 | GND layer | It is the reference plane connected to GND. It is used to reduce as possible all electromagnetic paths to GND, in particular for RF output. |
| Layer 3 | VCC layer | Plane subdivided into power islands (5 V for TEROS sensors, 3.3 V for the microcontroller stage and GND as additional reference plane for RF section to prevent radiated emissions at the board edge). |
| Layer 4 | Bottom layer | General-purpose signals with a polygon pour connected to GND. |

**Table 4.1:** Description of layers in the PCB layout.

As indicated in Table 4.1, it is introduced the concept of **power islands**: when a layer is available, it is useful to dedicate it to a single function, for example, supplying a section of the board with the minimum resistive trace as possible. In this board, it is easy to isolate three main regions, called "islands", 5 V for TEROS sensors, 3.3 V for the microcontroller, regulator, connectors, and GND under the LoRa RF area. A GND polygon pour, as an additional reference plane with respect to the entire second layer (GND layer), prevents radiated emissions at the board edge. Figure 4.6 shows them.

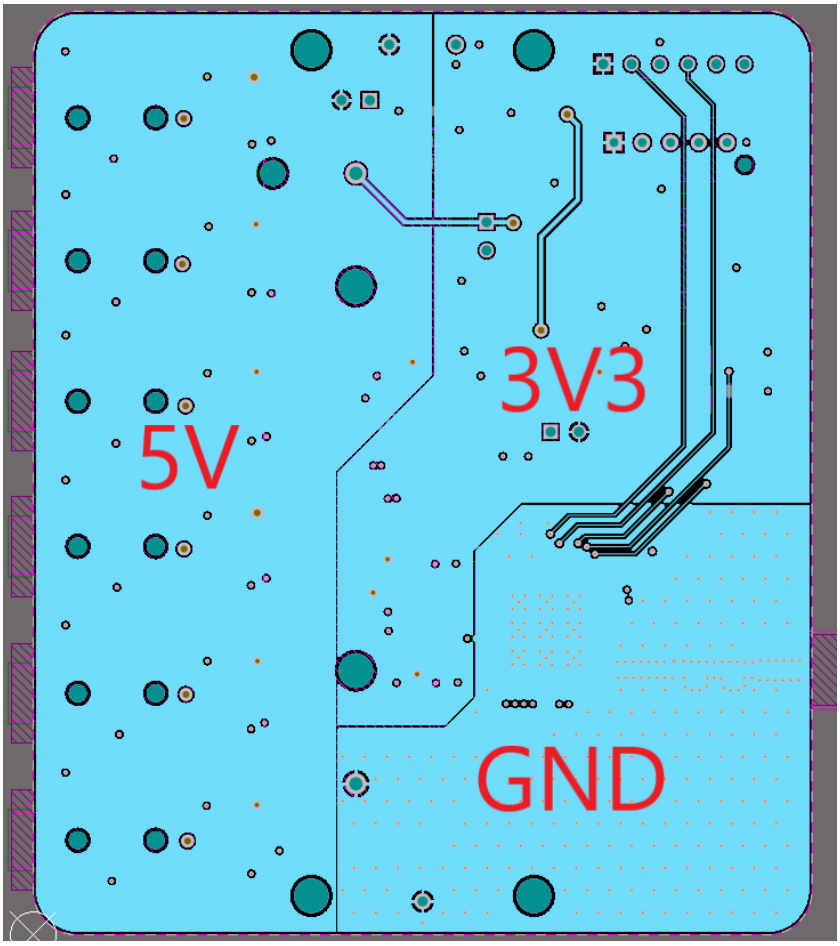



**Figure 4.6:** Screenshot of VDD layer that shows three power islands to lower as possible the power resistive paths.

### 4.2.1 Murata Wireless Module

Murata CMWX1ZZABZ[8] LoRa module needs vias under its nine GND pads. In Figure 4.7 is shown on the left part of the image (it has U1 designator, and it has four signs on the silkscreen to circumscribe it).
On the left of the Murata module, there are bypass capacitors, as indicated in the datasheet. These should be placed as close to the proper pin as possible in such a way to obtain the best performance by them.
Almost all digital pins are chosen far as possible from the RF area, and they travel using vias over the entire board.

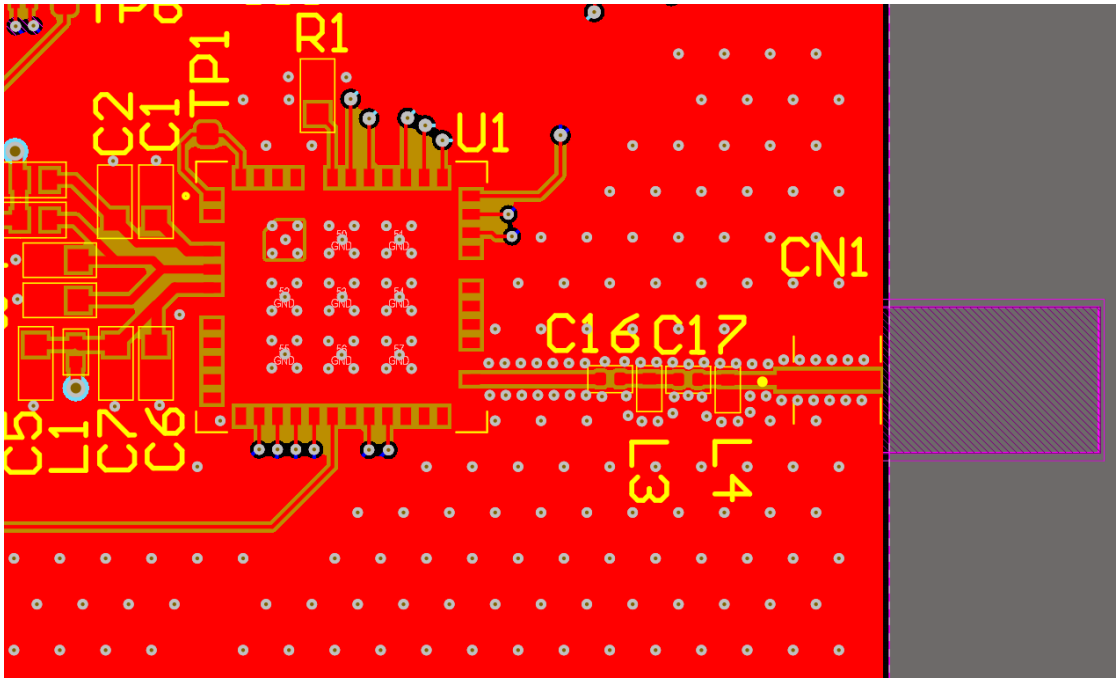



**Figure 4.7:** Screenshot of the first layer (Top layer) showing the Murata module footprint (on the left) and RF path (on the right).

### 4.2.2 LoRa Section

LoRa is a radiofrequency protocol. Moreover, as with all radiofrequency protocols, it must satisfy a proper matching to work properly.
In practice, this means that if a 50 Ω generator (Semtech IC) and a 50 Ω load (Antenna) are present, also the PCB track (transmission line) must be equal to 50 Ω to avoid reflections. These reflections lead to wasted energy and a reduction of the range and, if the matching is wrong, to a completely unworking board.
Moreover, each RF path is drawn after a simulation: in this case, a coplanar waveguide is needed.
A simulator, as AppCAD, can be used to determine the width of the track. Why is the width the only variable? Because, typically, the other parameters (height H from the actual plane and the reference plane, clearance G respect to adjacent tracks, thickness T of tracks, and relative dielectric constant $\epsilon_r$) are fixed by the manufacturer (H, T, $\epsilon_r$) or fixed a priori (G). The result is the right width of the track if you fix the target nominal characteristic impedance (50 Ω). An example is shown in Figure 4.8, where an actual characteristic impedance of 52.1 Ω is found (tolerance of 10% is considered good, so from 45 Ω to 55 Ω).
Note that also the dimensions of pads should be compatible with the width of the

track that connects Murata and the antenna.

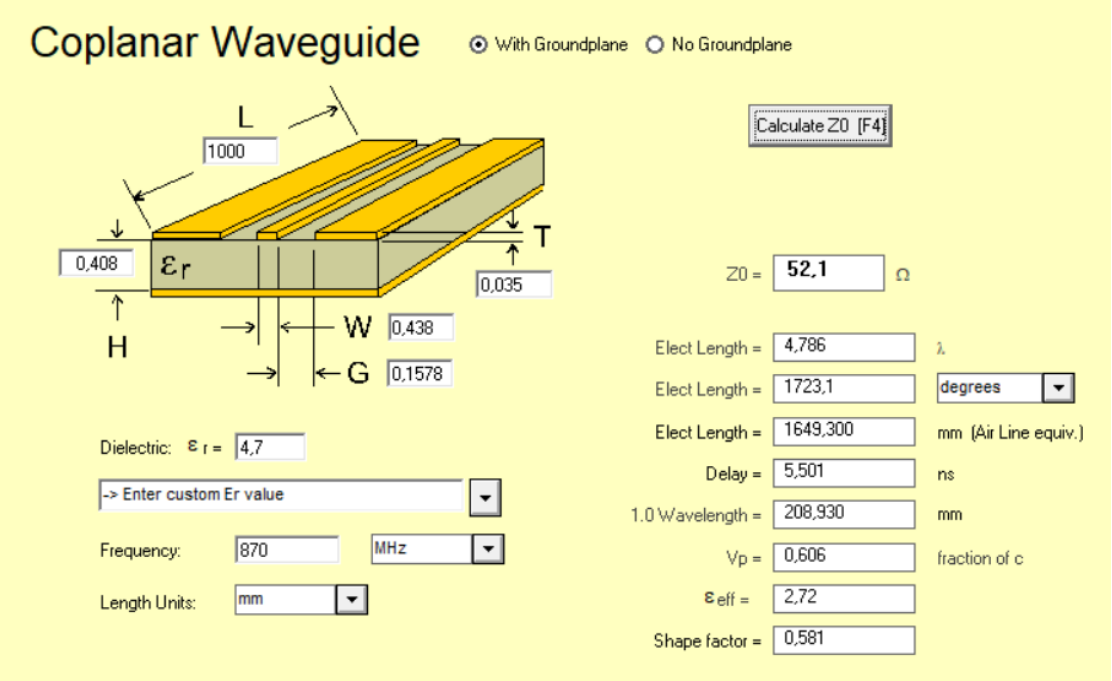



**Figure 4.8:** Screenshot of the performed simulation considering a width equal to 0.438 mm.

But, when a PCB board is manufactured, no possibility to perform some adjustment in the characteristic impedance is possible. For this reason, it is common practice to incorporate an antenna tuning network.
In Figure 4.7, on the RF path, it is possible to see four footprints referred to as two capacitances and two inductances. They have composed a mixed Pi-network and T network.
This tuning network is important because it allows adjusting the center frequency of the antenna, which will have shifted from what is given in the reference if the ground plane's size/shape/configuration differs from that design at all.
Tuning is usually done with a Vector Network Analyser (VNA), but it could be done at a very basic level by simply observing the RSSI changing as you use a trial-and-error approach to select tuning components.
So, it is common practice to have these footprints to place these passive components, starting from a short circuit on the feed path when the board is switched on for the first time.
In addition, to confine the electromagnetic field of the RF path, it is common usage using some **vias shielding** and **via stitching**.
These vias must be connected to GND from the top layer to the bottom layer,

and they should be as tiny as possible (compatible with the PCB manufacturer's limitations). In the case of vias shielding, they should be placed as close as possible to the RF track, and they should be as dense as possible. In this way, the electromagnetic field lines are referred to as GND as close as possible, avoiding some electromagnetic interference (EMI).
A similar target is performed by vias stitching, where spacing is referred to as the guided wavelength.
The guided wavelength is computed as follow:

$$\lambda_g = \frac{\lambda_0}{\sqrt{\epsilon_{eff}}} = \frac{c}{f}\frac{1}{\sqrt{\epsilon_{eff}}} = \frac{300{,}000{,}000\,\mathrm{m/s}}{870{,}000{,}000\,1/\mathrm{s}}\frac{1}{\sqrt{2{,}72}} \approxeq 0.209\,\mathrm{m} = 209\,\mathrm{mm};$$

As a rule of thumb, the correct spacing should be in a range of $\frac{\lambda_g}{20}$ and $\frac{\lambda_g}{10}$. Moreover, considering the above-computed value:

$$\frac{\lambda_g}{20} < spacing < \frac{\lambda_g}{10} \Rightarrow 10.5\,\mathrm{mm} < spacing < 20.9\,\mathrm{mm};$$

### 4.2.3 Test Points and Jumpers (Debug Features)

In a correct development process of a complex printed circuit board is important to simplify your life as possible when there is to do some functionality and performance measurements, supplying the board with laboratory equipment, or checking the voltage in the power supply stage or on a data line. Moreover, it has been designed a predisposition to perform these tasks as shown in Figure 4.9.
First, JPx are headers through-hole with 2 pins that allow separating a part of a circuit physically from another one downstream. In this way, it is possible to supply the circuit with an electrical appliance as a voltage generator.
For example, JP1 is configured to provide a nominal voltage of 3.6 V (something to emulate a battery), JP2 is configured to isolate the LDO 3.3 V regulator from the circuit to be able to program the microcontroller. Finally, JP3 can provide supply to the entire 5 V section.
In addition to this function, each of them can be used to measure the current drained (allowing to evaluate the most important factor for a low-power circuit, the current consumption) where JP1 being just after the battery track has the most important role, so measuring the entire consumption of the board in all working modes.
When they are not needed because the development step is terminated, they can be easily removed, desoldering them and soldered the corresponding solder bridge (respectively, SB2 for JP1, SB1 for JP2, SB3 for JP3). In this way, there is less as possible vertical occupation in such a way to have a thin vertical height referred

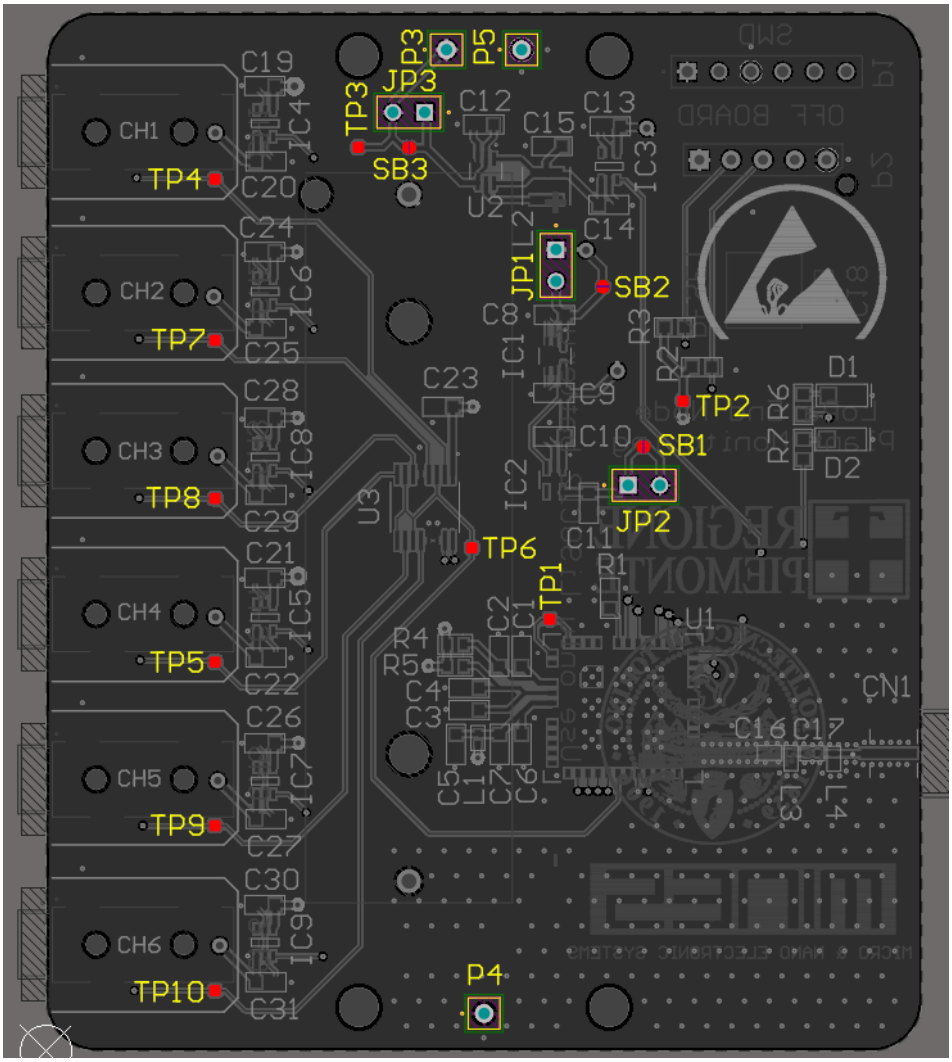



**Figure 4.9:** Screenshot highlights where are placed Test Points (TPx), Jumpers (JPx) and Solder Bridges (SBx).

from the top layer (useful when the board is installed in an enclosure box).
On the other hand, there are also some test points (TPx) strewed on all boards. In this way, it is possible to perform a measurement (with a voltmeter or an oscilloscope in whatever case) to evaluate the correct presence of voltage at that point.
TP4,TP5,TP7,TP8,TP9,TP10 are test points placed on data line for each TEROS sensor. TP6 is placed on the output of the analog switch.
TP2 and TP3 are power test points, and they verify respectively 3.3 V on output from LDO regulator (or from whatever power supply from the JP2) and 5 V is

output from boost regulator (or from whatever power supply from the JP3).
Finally, TP1 verifies the signal that provide supplying of the internal TCXO of the Murata module.

### 4.2.4 User Experience

Being a board that can be managed by non-technical workers, it should be clear and simple as possible for a few maintenance operations as the replacing battery, reset the device, or connecting/disconnecting removable elements.
For these reasons, it should be a quasi-professional look in such a way as not to confuse the person in charge to perform one of these tasks.
As highlighted in Figure 4.10, this is the final aspect of the bottom layer of the board that should be placed as top view in the protection case.
All small and confusing components as a microcontroller, passive elements, and other integrated circuits are in the bottom view (top layer of the board), so they are hidden on the view when it is installed in the shell case.

In this way, each operation on the field should be more accessible. Let's state some comments about them:

- The **battery shell** is placed in the middle of the board to guarantee the maximum ease as possible, also considering the small room on the case (placing it on edge could make it more difficult to hook/unhook it). In addition to it, there is a writing to indicate the way to attach the battery and a self-explaining sign to remember which type of battery must be placed (other types of battery are not compliant with this printed circuit board).

- There are six 3.5 mm stereo **jacks for TEROS sensors**, as stated in the requirements, organized on the right of the board with their own sign on the silkscreen to make a proper connection easier as possible.

- An **SWD connector** allows programming the board with a proper firmware that can be updated over time.

- An **off-board connector**, as specified in the requirements, allows connecting an external board with I2C sensors (for example, temperature, humidity, pressure sensors I2C compatible).

- An eye-catching **reset button** with a high stroke that should be clear to find for whatever operation requires a hard reset of the board.

- Finally, **graphical writings** on the remaining room on the board about stakeholders involved in the project.


**Figure 4.10:** 3D representation of the bottom layer of PCB, useful to show the following user experience on it.

### 4.2.5 I2C Interface

I2C interface requires some pull-up resistances on SDA and SCL lines to work properly.
Typically, these resistances are internally provided by the microcontroller so, no external resistances are required to communicate via I2C protocol.
As illustrated in [25] on page 426, the resistance value is typically around 20 kΩ due to keeping the dynamic consumption under control. But, this value is too high when the connection between the master and one slave overcomes few centimetres: the RC constant grows on that line, and the communication degrades until a value where the connection is no more established.
This has been observed in the development stage, and in Figure 4.11 are shown two

screenshots of the situation with and without dedicated external pull-up resistances. Two pull-up 2.2 kΩ resistances are placed on SCL and SDA lines as a trade-off between low power leaks and low RC constant.

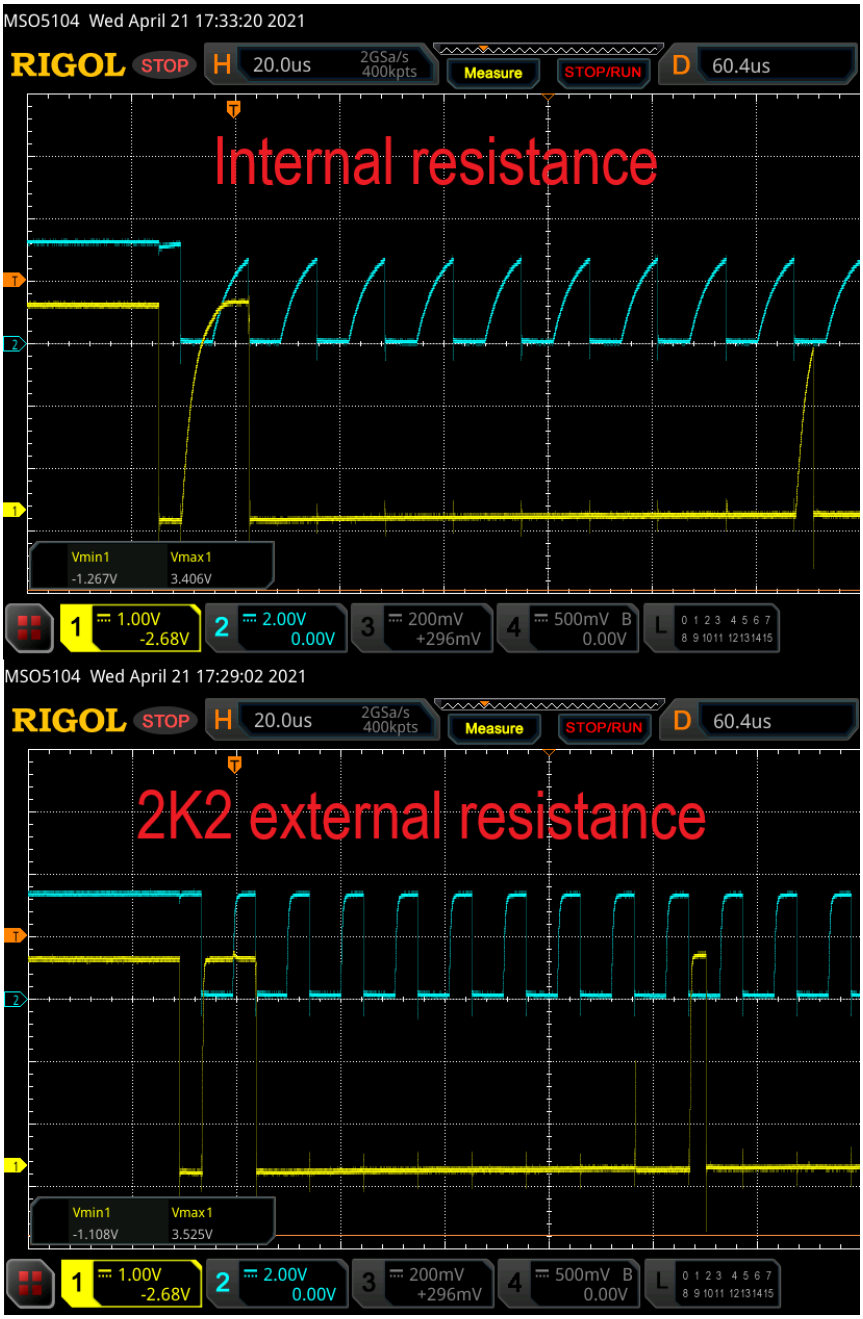



**Figure 4.11:** Screenshot from oscilloscope without and with external pull-up resistances.

## 4.3 Bill of Materials (BOM)

| Component | Description | Qty. | Designator | Price (€) |
|---|---|---|---|---|
| FSM8JSMA | Reset Button (Red) | 1 | B1 | 0.39 |
| BHAA-3 | Battery Holder AA 1 Cell | 1 | BT1 | 2.02 |
| CL10B104KO8 NNNC | Capacitor 100nF 0603 | 11 | C2,C4,C7,C14, C18,C20,C22,C25 C27,C29,C31, | 0.08 |
| C1608X5R1A106K 080AC | Capacitor 10uF 0603 | 5 | C1,C3,C5,C12, C15 | 0.38 |
| CL10B105KP8 NNNC | Capacitor 1uF 0603 | 12 | C6,C8,C9,C10, C11,C13,C15,C19 C21,C24,C26,C28, C30 | 0.08 |
| C0603X103J4HAC7 867 | Capacitor 10nF 0603 | 1 | C23 | 0.16 |
| SJ-3523-SMT | 3.50mm jack female | 6 | CH1,CH2,CH3, CH4,CH5,CH6 | 0.78 |
| EMPCB.SMAFSTJ .B.HT | SMA connector 50R, Female Socket | 1 | CN1 | 2.51 |
| LTST-C171KRKT | Red LED 0805 | 1 | D1 | 0.23 |
| LTST-C171KGKT | Green LED 0805 | 1 | D2 | 0.23 |
| STLQ020C33R | 3V3 LDO | 1 | IC2 | 0.49 |
| SiP32431DR3-T1GE3 | Load Switch | 7 | IC3,IC4,IC5,IC6, IC7,IC8,IC9 | 0.49 |
| LM66100DCKR | Ideal diode, rev.pol.prot | 1 | IC1 | 0.35 |
| TSW-150-07-T-S | x50 2.54mm headers | 1 | P1,P3,P4,P5 | 2.65 |
| SNT-100-BK-T | Shunt connector 2.54mm | 3 |  | 0.21 |
| BLM15HG102SN1 D | Ferrite 0402 | 1 | L1 | 0.13 |
| LQH3NPN4R7MM EL | Inductor 4.7uH 1212 | 1 | L2 | 0.28 |
| B5B-XH-AM(LF)(SN) | Header1x5, Off-board conn. | 1 | P2 | 0.21 |
| RNCP0603FTD10 K0 | Resistor 10K 0603 | 1 | R1 | 0.08 |
| CR0603-JW-331ELF | Resistor 330R 0603 | 2 | R6,R7 | 0.08 |
| CMWX1ZZABZ-078 | Murata LoRa module | 1 | U1 | 12.86 |
| TPS61222DCKT | 5V switching regulator | 1 | U2 | 1.21 |
| MAX4562EEE+ | Analog switch | 1 | U3 | 4.02 |
| CRGCQ0603J2K2 | Resistor 2K2 0603 | 2 | R2,R3 | 0.08 |
| RC0402FR-070RL | Resistor 0R 0402 | 4 | R4,R5,C16, C17(0R fitted) | 0.08 |
|  |  |  | **Total components** | 40.94 |

**Table 4.2:** Bill of materials on board (PCB cost not included).

Price in euro is intended per element in small batches (typically less than 10 units). They are indicated by Digikey (`https://www.digikey.it/`) otherwise intended is specified.

| Component | Description | Qty. | Price (€) |
|---|---|---|---|
| Electronic components | See Table 4.2 | 1 | 40.94 |
| RP1210BF | Case IP65 | 1 | 17.48 |
| 50.616 PA/RSW | Gland TEROS sensors | 6 | 4.98 |
| PNC1/2W BK080 | Gland antenna | 1 | 2.25 |
| ANT-868-CW-HW-SMA | Long SMA antenna 50R rugged | 1 | 8.01 |
| Printed Circuit Board | 1 piece manufactured by MD srl | 1 | 62.26 |
| | | **Price per PCB** | 160.82 |

**Table 4.3:** Total bill of materials.

## 4.4 3D models and Variations

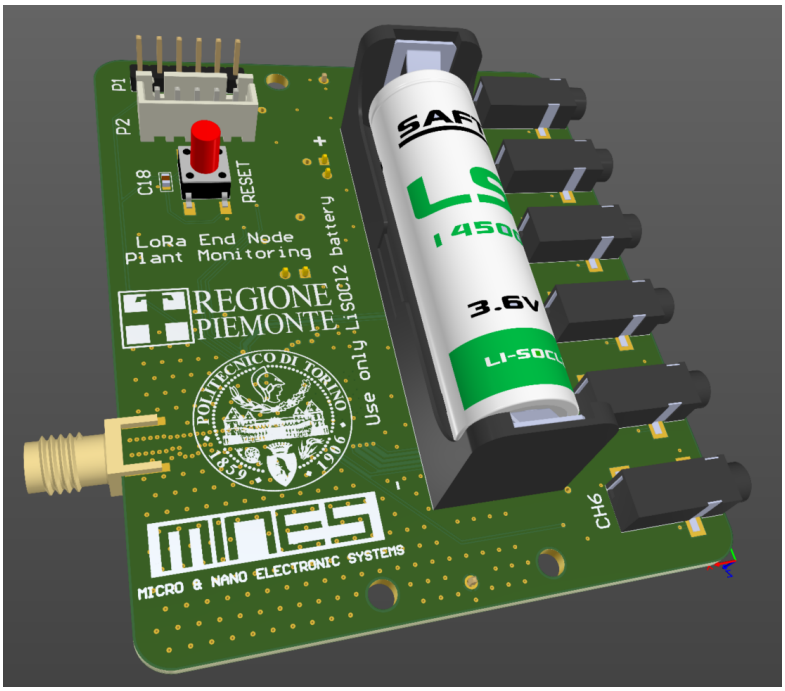
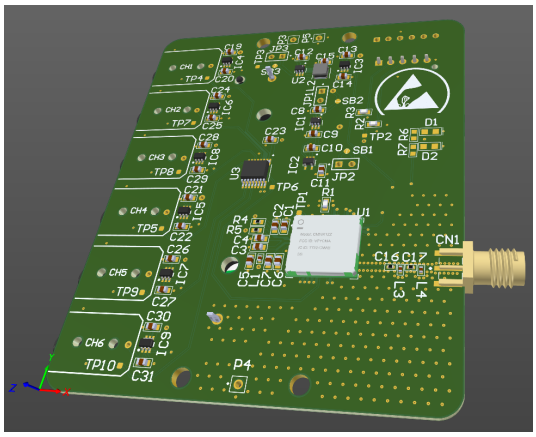


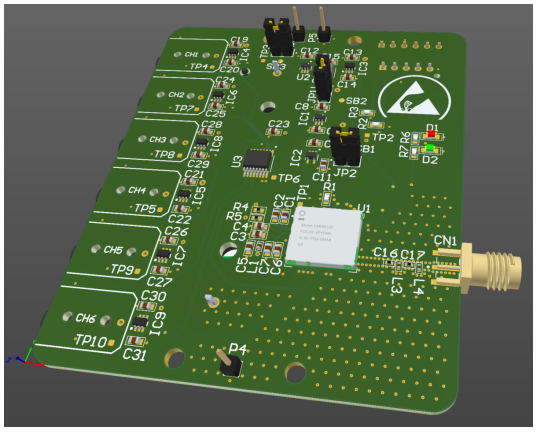

**Figure 4.12:** Screenshot of bottom view 3D model for both variations.

A really appreciated feature of modern electronic CAD is that it can show the final result of the printed circuit board before manufacturing it.
In this way, it is possible to evaluate some mechanical constraints or aesthetic features of the board if needed.
In Figure 4.12 is shown the 3D bottom view, previously analyzed.
Another important feature of Altium is the concept of **Variation**: when a schematic and the corresponding board are designed, it is possible to indicate which components consider. This can be very useful when you have to generate a custom Bill

Of Material with the components that should be placed in reality.
In this thesis project, two variations are defined: **Release** and **Debug**. The first one considers only the component that should be placed in the PCB that will be installed on the agriculture field. The other one also considers all components that are required to evaluate the correctness and the performance of the board in the development stage (for example, many headers, LEDs).
In Figure 4.13a and 4.13b are shown, respectively, the release board and the debug board. As it is possible to see, some components are missing in the release board because they are useless and they only occupy precious space.



**(a)** Top view 3D model in release variation.



**(b)** Top view 3D model in debug variation.

**Figure 4.13:** Comparison between top view 3D model variations.

## 4.5 Manufacturing

There are companies that allow realizing small batches or prototype PCB to test in real condition every functionality of the electronic circuit.
It is possible to find in China popular manufacturer as PCBWAY or JLCPCB, in Europe it is popular Eurocircuits for fast prototyping.
In this project, a small RF section is present that needs very small vias (0.20 mm) with respect to standard low-cost via dimensions (0.30 mm-0.35 mm) to maximize its performance (smaller vias means small stray capacitances).
For this reason, it has been chosen an Italian manufacturer called MD srl that satisfies this requirement at a lower cost.
It has been chosen the standard thickness stack-up (1.55 mm) because it is sufficient for the involved radiofrequency of the board (868 MHz).
In Figure 4.14, it is reported the manufacturer stack-up where it is shown the thickness of all layers used to fabricate the PCB, and this has to be used to configure

the internal stack-up of Altium Designer electronic CAD. This is a mandatory step to compute the right width trace of the RF path properly. Without these values, Altium is not able to simulate the right values of the RF stage.

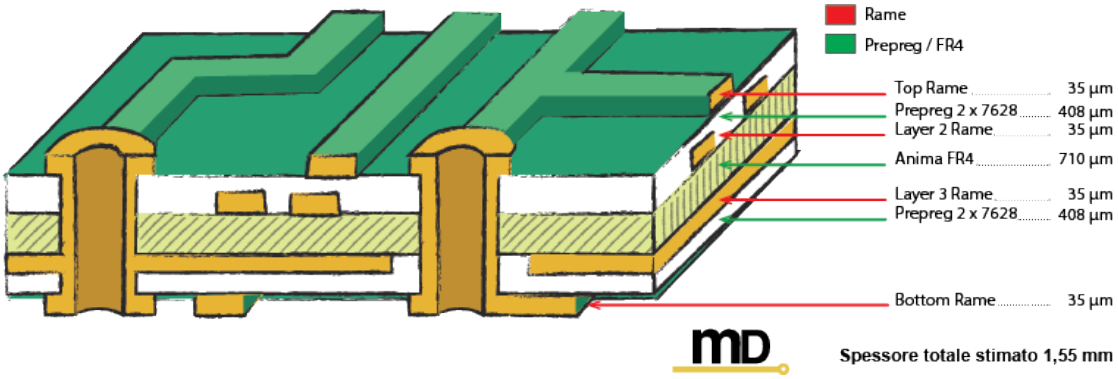



**Figure 4.14:** MD srl 4-layers PCB stack-up (Courtesy of `https://www.mdsrl.it/`).

Other important dimensions to respect are minimum clearance (0.15 mm) among all elements on board, minimum width of tracks (0.15 mm), minimum dimensions of vias, and annular ring.
Another aspect is related to silkscreen: the minimum width line silkscreen is 0.16 mm. This is easily respected by designators and boundaries of components, but it is less easy to respect when graphical logos are needed.
In that case, more complex is the logo, and a wider logo should be placed on the board to have a good representation on the PCB.
For this reason, Piedmont's logo, Politecnico's logo, and Mines's logo are placed on an almost-void area where only stitching and shielding vias are placed.
When all requirements are satisfied, gerbers can be generated by electronic CAD, and it can be sent to PCB manufacturer.
Gerbers are mechanical files where it is indicated how each layer should be, how to place every mechanical drill, where placing silkscreen, and so on.
An additional file called assembly file should be provided if assembly service using automatic pick and place machine is used. One assembly file for each side (top and bottom). This assembly file contains the exact position where all components must be placed.
This service will be used to realize a small batch of PCB in such a way as to accelerate the design-to-field process when a complete and tested revision is designed.
In Figures 4.15, 4.16, 4.17, and 4.18 are shown four gerber files, respectively top layer, GND layer (inner layer), VCC layer (inner layer), and bottom layer.

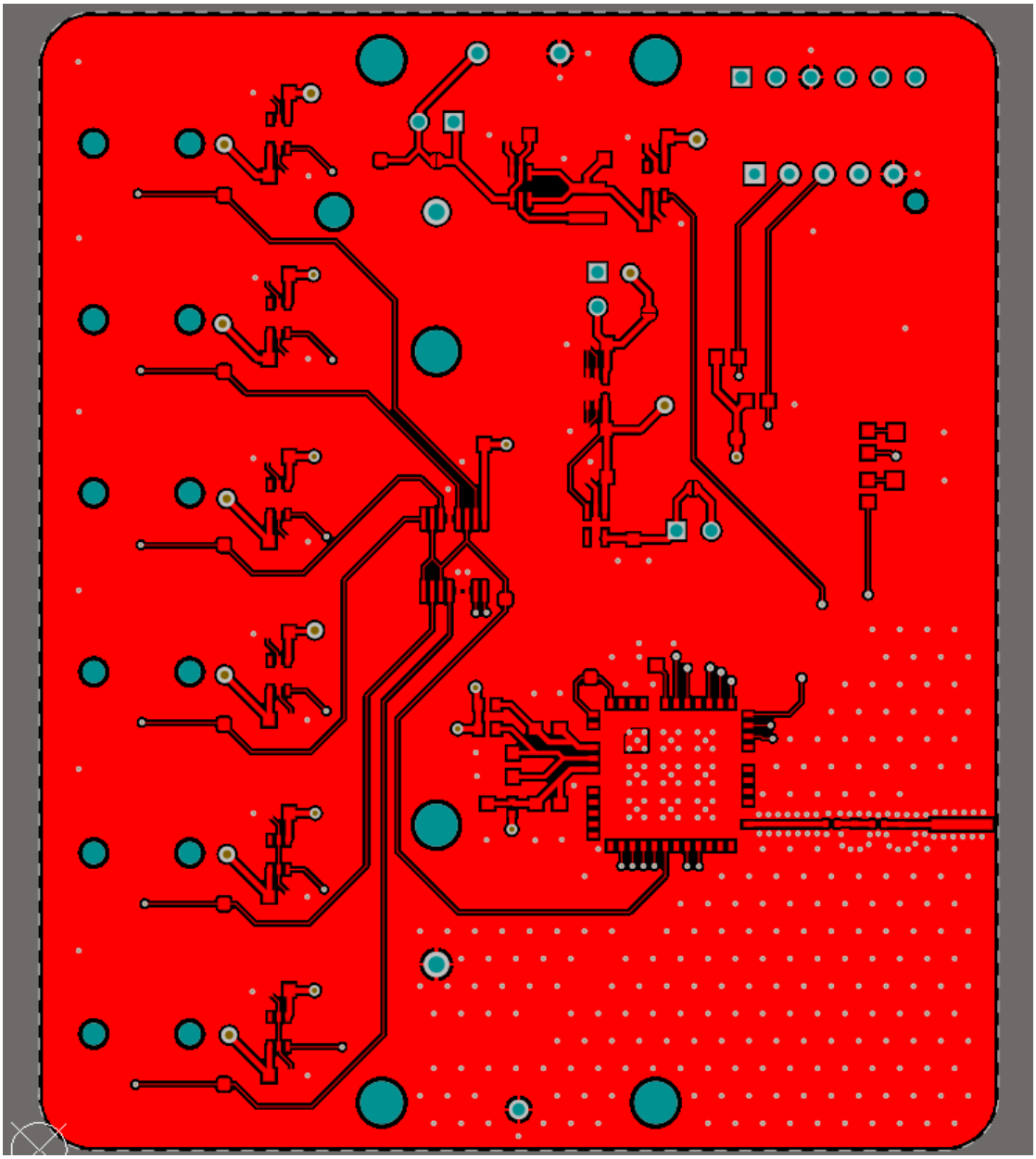


**Figure 4.15:** Top layer graphical gerber file.

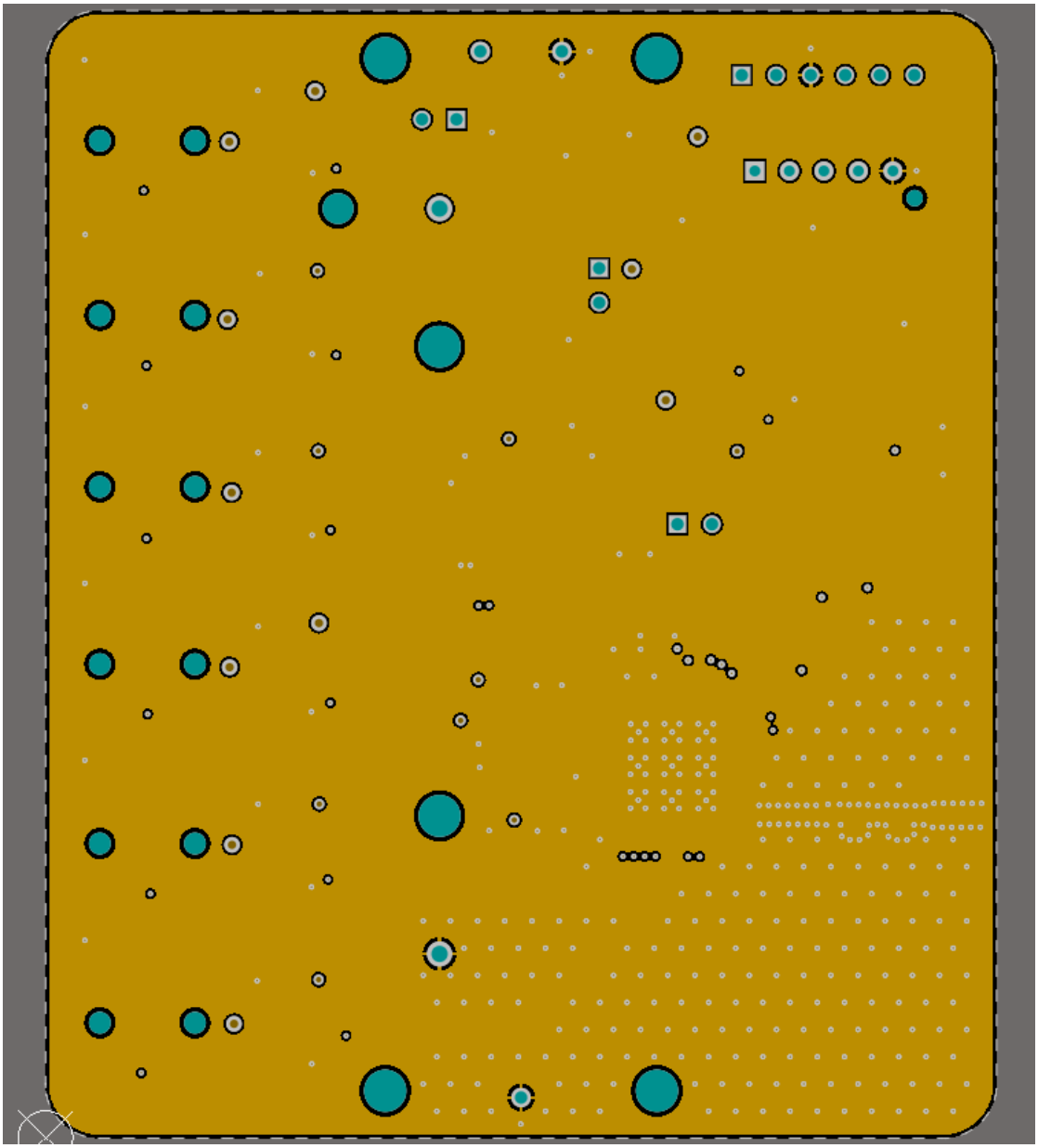


**Figure 4.16:** GND layer graphical gerber file.

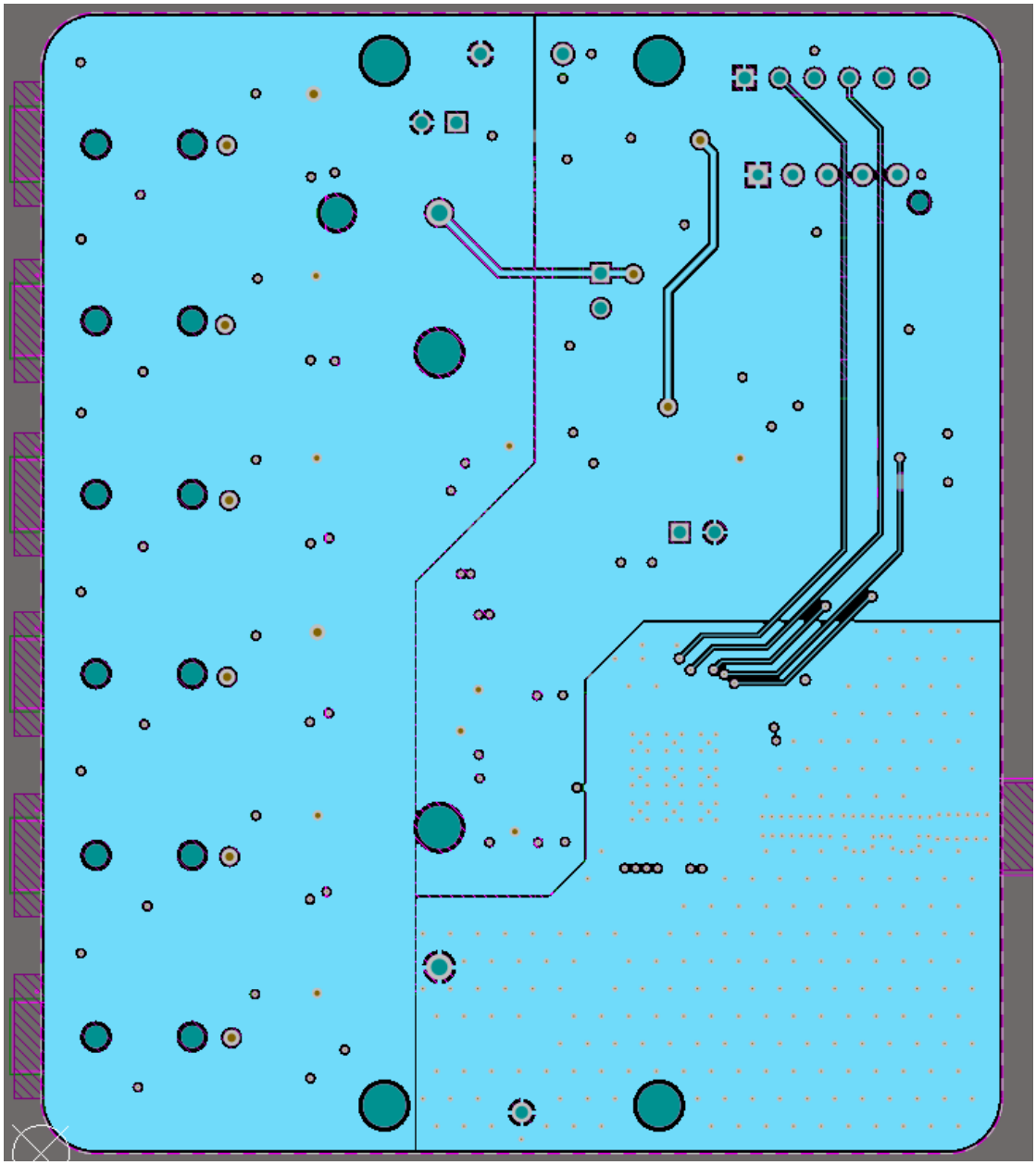


**Figure 4.17:** VCC layer graphical gerber file.

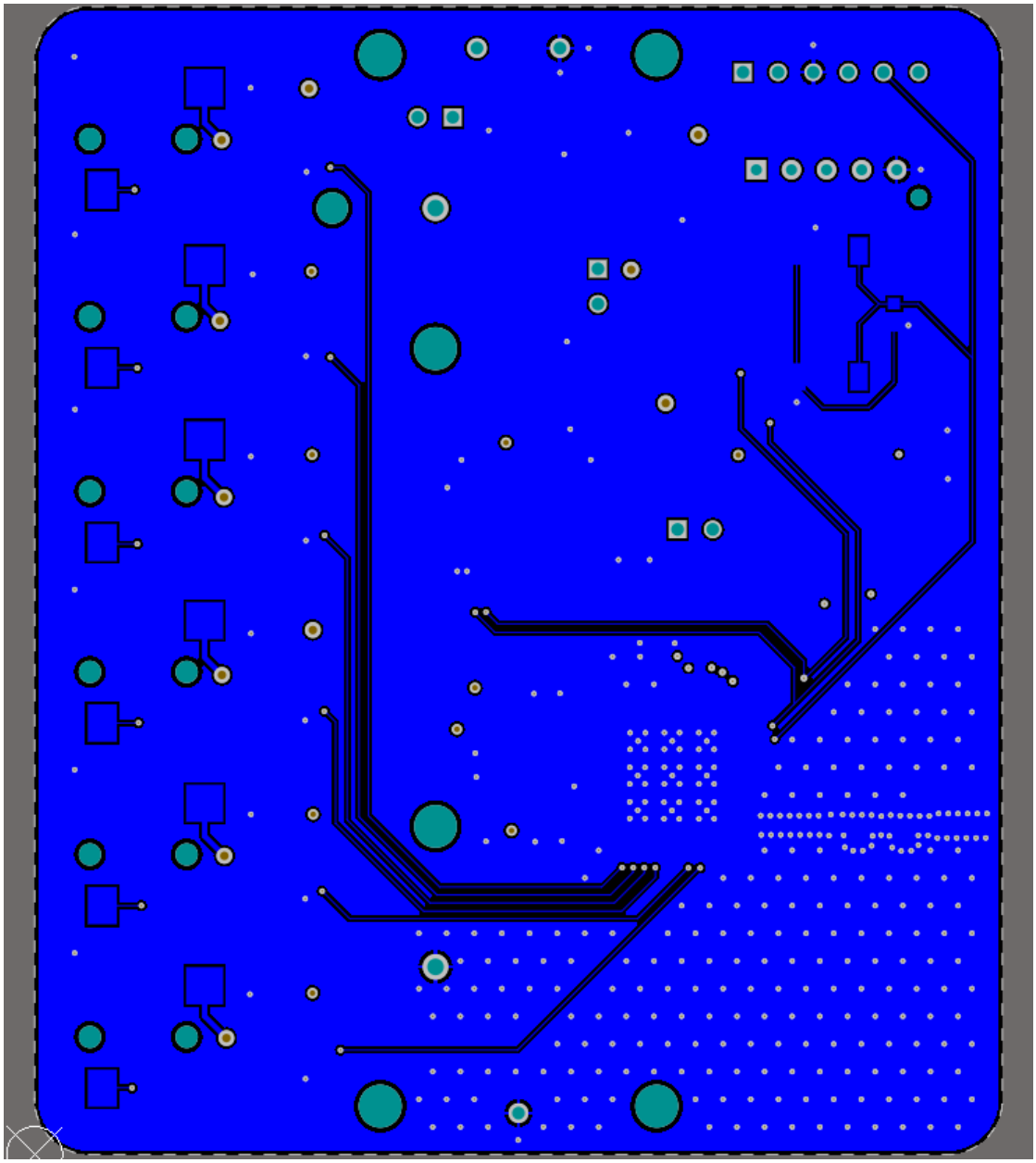


**Figure 4.18:** Bottom layer graphical gerber file.

# Chapter 5

# Experimental Results

## 5.1 PCB Assembly

Manufactured printed circuit board by MD srl is shown in Figure 5.1.

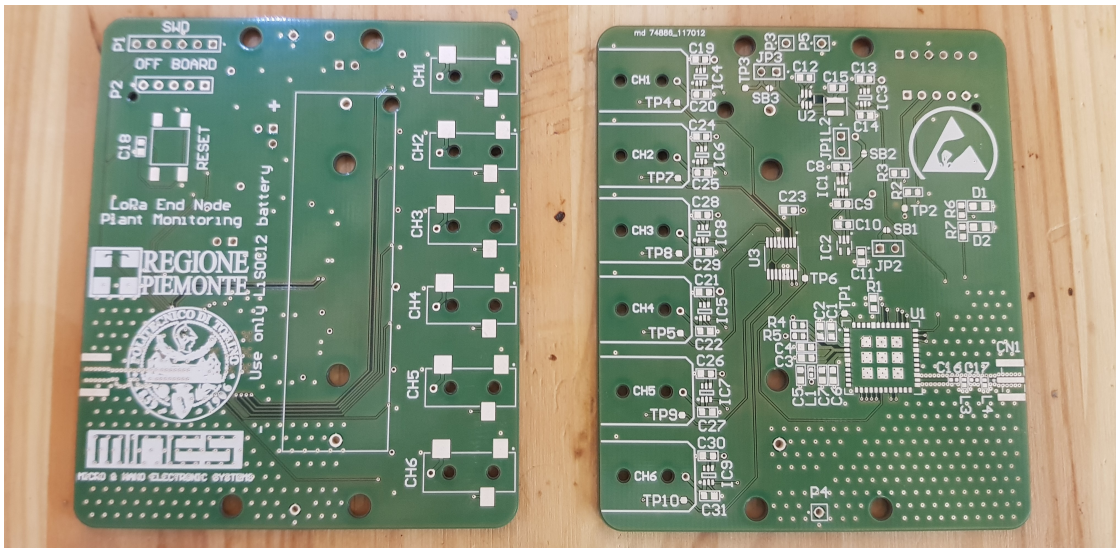



**Figure 5.1:** Photo of bottom (on the left) and top side (on the right) of the fabricated printed circuit board.

In the first step, PCB has been assembled in the Mines's laboratory using available appliances to solder all components on the board.
The main problem has been the Murata module that owns a pitch of 0.8 mm, having pads width equal to 0.5 mm.
For this reason, solder paste has been placed in a very accurate way only on Murata pads, and a reflow step has been performed.

In this way, it is possible to verify the correct connection of these pads, and it is possible to rework the component if it is not placed correctly.
In Figure 5.2, it is shown the difference between bare PCB and Murata module mounted on it.

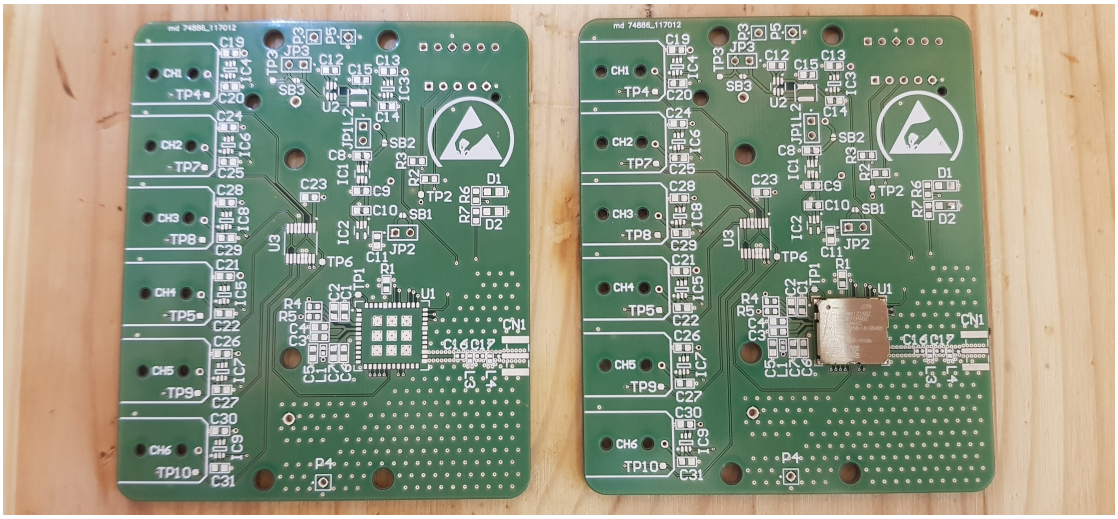


**Figure 5.2:** Photo before (on the left) and after (on the right) Murata assembly.

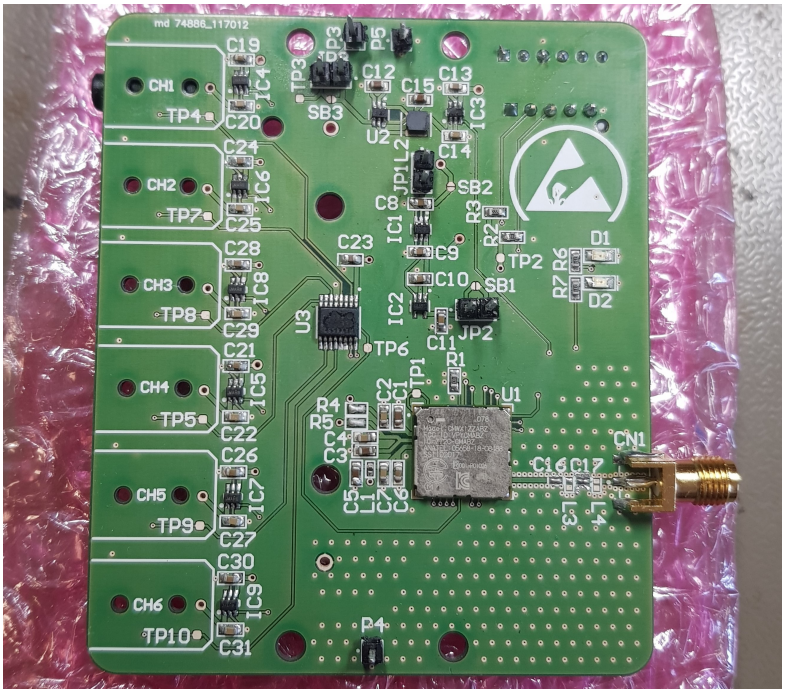


**Figure 5.3:** Photo of demo assembled PCB.

At this point, solder paste is placed on all other SMD pads of the top side, and the proper component is placed on it. Only through-hole components are left out in this stage. Afterward, a second reflow process has been performed.
The final step is to solder all through-hole components and the SMA antenna.
The final result is shown in Figure 5.3.

## 5.2 Backend Data

A part of this thesis is related to the handler of data until they reach the cloud platform.
This is possible due to the presence of a LoRaWAN network composed of many gateways. When an end device is able to connect to at least one of these connected gateways, its data will be sent to a cloud platform as TTN.
In Figure 5.4 is shown a typical **Live Data** section screen where a raw payload (not shown in the figure) is decoded by code illustrated in Chapter 3.9.
The result is variables that indicate, in this case, physical quantities (as humidity, soil moisture, soil temperature) and some kind of communication variables (c_done indicates when a previously sent downlink command is correctly performed, sa_on indicates the presence of a sensor on a specific channel of the board). These variables have been illustrated in Figure 3.3.
In particular, in the figure is shown a running demonstration where an HDC2080 temperature and humidity sensor is connected to the I2C (off-board) connector, and two TEROS21 sensors are plugged on channel 1 and channel 3. All other channels are not active because no sensors are plugged in.

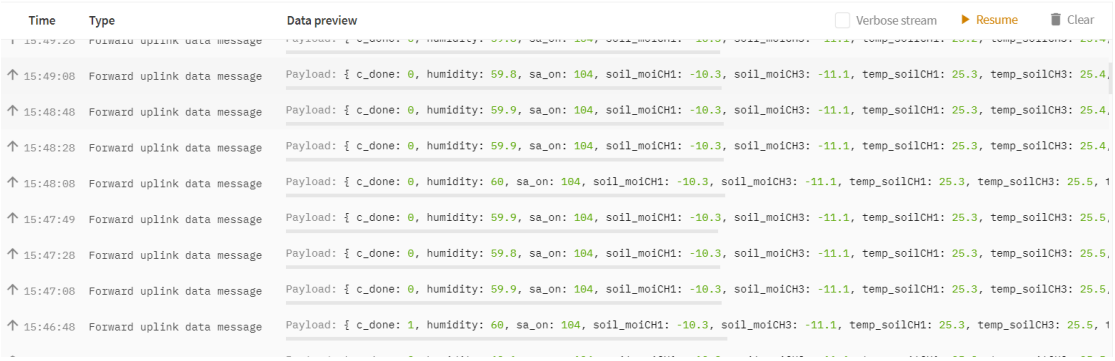



**Figure 5.4:** Screenshot from The Things Network live data console.

## 5.3 Power Consumption

In the investigation chapter, in particular, in Chapter 2.4, an estimation of the current consumption of all components has been investigated.
For this reason, an evaluation using an amperemeter has been established.
The main issue is that general-purpose amperemeters are not able to sense such small current values. In Figure 5.5 is shown the result when the microcontroller is in standby mode using a general-purpose power supply.

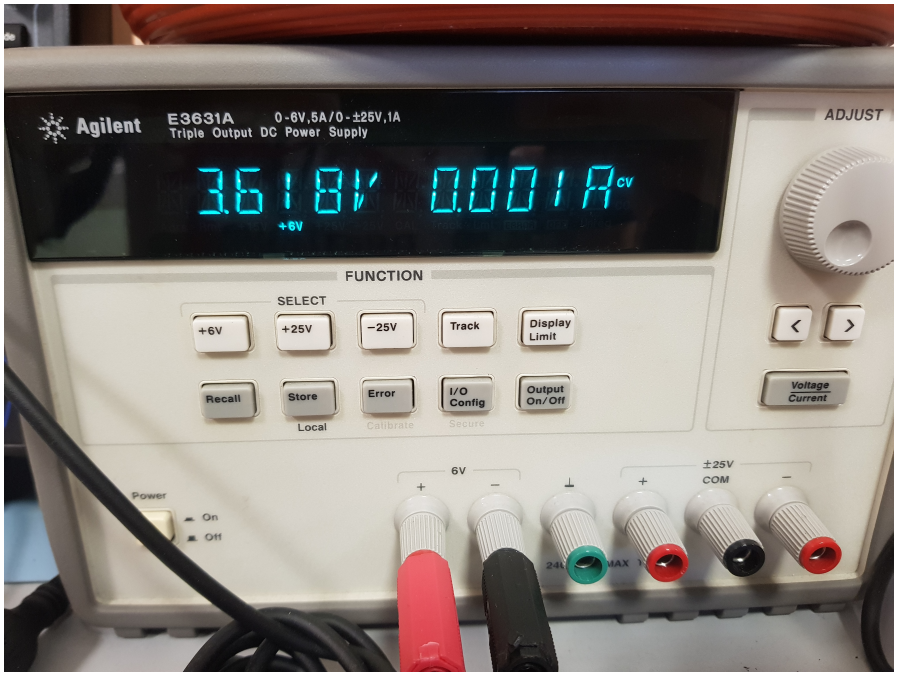



**Figure 5.5:** Photo of the Agilent E3631A power supply.

The last figure shows 1, but it is a fake value due to the fact that this supply generator is not able to sense such a small current. It can only supply it.
Moreover, a specific amperemeter should be employed to estimate the real standby current of the PCB.
Politecnico di Torino provides Keithley DMM7510, a very accurate digital multimeter that has been employed to compute the real current profile of the board.
In standby, DMM7510, previously set up on the correct range, so 100 µA, shows a noisy behavior as shown in Figure 5.6.

A feature of this fancy digital multimeter is to store thousands of samples to see clearly the profile of current consumption.
In Figure 5.7 it has been shown a profile in the configuration of demonstration to see the functionality of the board: two TEROS 21 are connected to the proper jacks,


**Figure 5.6:** Photo of the Keithley DMM7510 digital multimeter when end node is in standby mode.

and an I2C off-board containing a temperature and humidity sensor HDC2080 is connected. After this, a LoRa TX is performed before coming back to standby mode.
After a fixed time (5 sec), the system is woken up to open two RX windows to receive a packet from the gateway eventually.
The maximum measured current absorption occurs during the LoRa transmission, and it is equal to 32.51 mA. Consulting Murata datasheet[8], the supply current in transmitter mode when RFOP setting = 10 dBm is equal to 36 mA considering the total current consumption including MCU on the active state.
Instead, considering the microcontroller in runtime without executing particular tasks, the average current consumption of 11.0 mA is measured. Consulting the microcontroller's datasheet[26], on page 69, the current consumption in run mode, code with data processing running from flash memory, in range 1, HSI clock source, a typical current of 7.15 mA. The difference is probably due to biasing current consumption of other active electronic components as 3.3 V regulator.
In Table 5.1 is shown a comparison between computed current consumptions and measured current consumptions.
Regarding TEROS21 consumptions, datasheet[20] shows only a generic current drain during measurement where a maximum current of 16.0 mA and a typical

current of 5.0 mA is issued. Considering the power-up as the maximum current drained by the sensor and a more constant current during the effective measurement, our experimental results, respectively 14.5 mA and 6 mA, confirm the datasheet's values.
Keep in mind that these timing results are measured having a time resolution of 50 ms configured in the Keithley DMM7510 digital multimeter.

| Operation | Estimated Time | Measured Time | Estimated peak current | Measured peak current |
|---|---|---|---|---|
| **Murata LoRa module in ADR (used DR3) with payload of 16 bytes** | | | | |
| Run time | / | / | 7.15 mA | 11 mA |
| Standby time | / | / | 2.65 µA | 1.89 µA |
| TX | 0.308 s | 0.15 s | 36 mA | 32.51 mA |
| RX1 | 0.08 s | 0.10 s | 22.2 mA | 13 mA |
| RX2 | 0.08 s | 0.10 s | 22.2 mA | 12.7 mA |
| **TEROS 21** | | | | |
| Power-up | 0.1 s | 0.05 s | 10 mA | 14.5 mA |
| Meas. Durat. | 0.2 s | 0.25 s | 10 mA | 6 mA |

**Table 5.1:** Comparison between nominal consumptions and experimental consumptions.

But the graph in Figure 5.7 does not show the standby current being orders of magnitude lower than runtime sections.
Performing a new measurement campaign during the standby phase, a plot as in Figure 5.8 occurs. In that stage, a very low current is present in the circuit that has been affected by whatever type of noise as thermal noise of electronic components. The most important figure is the average standby current consumption that is equal to 1.886 µA in our tests. Comparing it to the estimated standby current consumption, considering as reference the typical quiescent current because measurements have been performed in a laboratory at 25 °C, in these conditions, the battery lifetime is guaranteed, and probably a greater battery lifetime will occur.

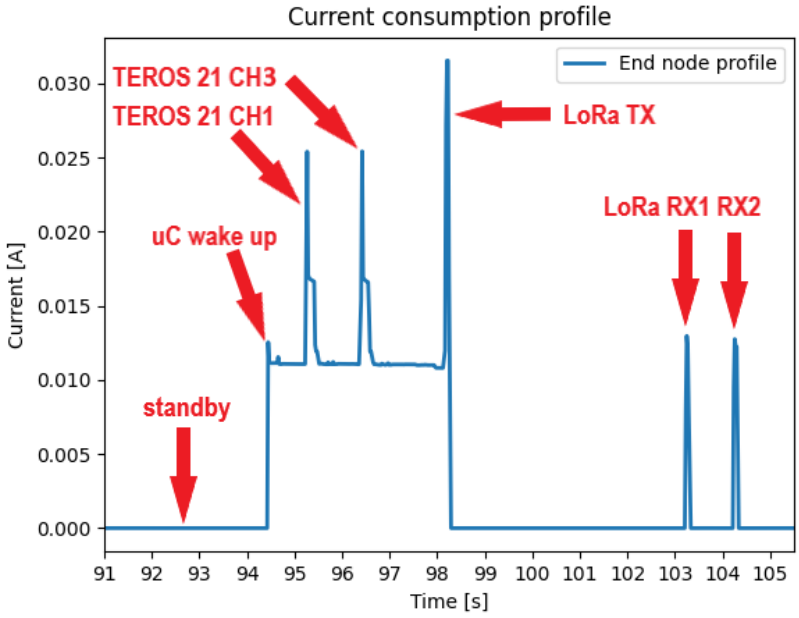



**Figure 5.7:** Graph of current consumption profile in all modes using Keithley DMM7510 digital multimeter.

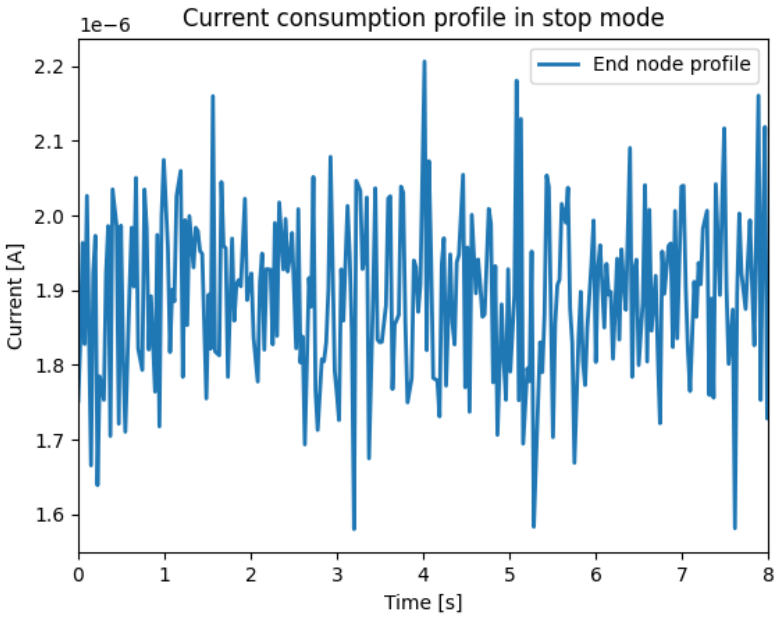



**Figure 5.8:** Graph of current consumption profile in standby using Keithley DMM7510 digital multimeter.

# Chapter 6

# Conclusion and Future Perspective

A complete embedded electronic system has been developed, starting from requirements to a final physical object.
The development includes all design choices, both from firmware implementation and hardware implementation, covering all those aspects that should be considered in the fabrication of a printed circuit board.
In particular, beyond the design of the functional tasks, experimental characterizations of the runtime and standby consumption have been performed to foresee the system lifetime when it is placed in a harsh environment as an agricultural field.
The triennial project WAPPFRUIT is only in the first year so, there is time to realize an enhancement in a second and more complete board and, above all, there is the need to realize a novel PCB employing the same platform and the same LoRa module to realize the actuation stage composing the first working micro-irrigation system.
In this way, one type of node punctually senses data from soil and air sensors, sending all data on a cloud platform when, instead, a different kind of node receives commands via LoRa radiofrequency technology to activate water pumps in a more thoughtful way as possible in such a way to preserve this natural resource together maximize the production of the orchard.
As highlighted in Chapter 2.3.5, some electronic components allow to enhance the board's features, and they have been omitted for timing reasons. In the case of TI BQ35100, this is an integrated circuit able to sense various parameters of the power supply battery in such a way as the state of charge could be estimated and, consequently, computing when the battery itself should be replaced.
Some protection devices can be added in all those stages where internal protection is not present (as on the input of pins of the Murata module).

Another important aspect where it is possible to work is related to the choice of the nominal voltage: this board has been realized considering a nominal voltage of 3.3 V (employing as reference the design schematics of the ST development board). Still, the Murata module should work at lower voltages, also transmitting/receiving LoRa packets. In this context, it is essential to study this aspect because it can improve the lifetime performance of a standalone board and allow to work directly with plenty of integrated circuits that work at 1.8 V.
Being LoRa a well-established technology, some manufacturers are starting to provide its microcontroller with an internal sub-GHz peripheral. This can be led to a reduction of the current consumption both in standby and runtime due to all communication paths are internal to the die and, in addition, saving some PCB layout footprint. Examples are the STM32WLE and STM32WL5 series by ST Microelectronics, where a sub-GHz peripheral is incorporated. In particular, the STM32WL5 series contains two cores (instead of one), one Cortex M4 and one Cortex M0+ (as Murata module that includes a Cortex M0+ single-core microcontroller). This could be used to subdivide tasks in these cores: M0+ dedicated as Always-On machine to enter/exit from standby modes and M4 to perform all CPU-intensive tasks. This can be a study that should be performed.

# Bibliography


[1] *Italy's water footprint*. Available here. EXPO 2015 (cit. on p. 2).

[2] *WAPPFRUIT - Tecnologie intelligenti applicate alla gestione dell'acqua in frutticoltura*. Available here. eip-agri (cit. on p. 2).

[3] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. «A comparative study of LPWAN technologies for large-scale IoT deployment». In: *ICT Express* 5.1 (2019), pp. 1–7. ISSN: 2405-9595 (cit. on p. 3).

[4] *LoRa documentation*. Available here. Mobilefish.com (cit. on pp. 6, 9).

[5] A. Goffeau C. Brouwer and M. Heibloem. *Irrigation Water Management: Training Manual No. 1 - Introduction to Irrigation*. FAO - FOOD and AGRICULTURE ORGANIZATION OF THE UNITED NATIONS, Available here, 1985 (cit. on pp. 10, 11, 14).

[6] *Soil water status: content and potential (2S-I)*. Available here. Logan, Utah, USA: Campbell Scientific, Inc. (cit. on pp. 14, 49).

[7] *Slides - Soil Water Content and Soil Water Potential*. Available here. ICT International (cit. on pp. 18, 20, 21).

[8] *CMWX1ZZABZ data sheet*. Available here. Nagaokakyo, Kyoto, Japan: Murata (cit. on pp. 29, 53, 55, 56, 89, 109).

[9] P. Horowitz and W. Hill. *The Art of Electronics*. Cambridge University Press, 2015 (cit. on p. 34).

[10] *LS14500 data sheet*. Available here. Levallois-Perret, France: Saft (cit. on pp. 33, 35, 59, 60).

[11] *LM66100 data sheet*. Available here. Dallas, USA: Texas Instruments (cit. on pp. 36, 56).

[12] *BQ35100 data sheet*. Available here. Dallas, USA: Texas Instruments (cit. on pp. 37–39).

[13] *Understanding the Terms and Definitions of LDO Volt. Regulators (SLVA079)*. Available here. Dallas, USA: Texas Instruments (cit. on pp. 40, 41).

[14] *STLQ020 data sheet*. Available here. Geneva, Switzerland: STMicroelectronics (cit. on pp. 42, 56).

[15] *TPS61222 data sheet*. Available here. Dallas, USA: Texas Instruments (cit. on p. 43).

[16] *SiP32431 data sheet*. Available here. Malvern, Pennsylvania, United States: Vishay Siliconix (cit. on pp. 43, 56).

[17] *SN74HC251 data sheet*. Available here. Dallas, USA: Texas Instruments (cit. on p. 46).

[18] *MAX4562 data sheet*. Available here. San Jose, California, United States: Maxim Integrated (cit. on p. 47).

[19] *TEROS11-12 integration guide data sheet*. Available here. Pullman, USA: METER Group, Inc. (cit. on pp. 51, 52).

[20] *TEROS21 integration guide data sheet*. Available here. Pullman, USA: METER Group, Inc. (cit. on pp. 51, 52, 109).

[21] *HDC2080 data sheet*. Available here. Dallas, USA: Texas Instruments (cit. on p. 56).

[22] *VJ....W1BC Basic SMD Ceramic Capacitors data sheet*. Available here. Malvern, Pennsylvania, United States: Vishay Siliconix (cit. on p. 56).

[23] *I-CUBE-LRWAN LoRa package (UM2073) User manual*. Available here. Geneva, Switzerland: STMicroelectronics (cit. on p. 63).

[24] *SGW2828 LoRa Module RF Design and PCB Layout Guideline*. Available here. Hong Kong: SG Wireless (cit. on p. 88).

[25] C. Noviello. *Mastering STM32*. leanpub, Available here, 2018 (cit. on p. 95).

[26] *STM32L072CZ data sheet*. Available here. Geneva, Switzerland: STMicroelectronics (cit. on p. 109).