Tutors	Students
Mr Eric Alata Mrs Daniela Dragomirescu	Mr Henri Cazottes Mr Audran Deyts Mr Pierre Noël Mr Thomas Delmas Mr Gaël Loubet Mrs Hélène Ravily

Summary

Introduction

The Internet of Things (IoT) is currently expanding exponentially: nowadays the number of smart devices used around the world is over 6 billion. In four years, this number is expected to reach 20 billion.

Thus, we think that understanding and gaining more insight into the Internet of Things is now essential and crucial for IT students.

And, as students ourselves, we would like to promote innovation and creativity in INSA projects.

The goal of our project is to offer a free, easy and well-documented access to an open LoRa network deployed over the INSA campus. LoRa technology has been created for smart devices, taking into account energy, mobility and other specific issues that characterize these kinds of technologies. With this LoRa network, users will be able to carry out educational, professional or personal projects related to IoT.

I. Objectives - Specifications

1. Context

The goal of our project is to deploy a LoRa network over the INSA campus, accessible in a free, open, well-documented and easy way via a web platform, connected to The Things Network, an international IoT community.

We also offer a demonstrator and concrete application using our network: a connected weather station with several sensors.

Future applications can be varied, only limited by the user's imagination: how to easily find free parking slots? How to quickly know if the coffee machine has a problem or needs refilling? How to make the campus a more pleasant place with interactive games? etc.

These targeted users are INSA students, teachers, staff or anyone wanting to create or use connected applications through smart devices. This will promote the image of INSA as a connected and innovative campus and it will be the first school in France to offer this kind of service. Furthermore, teachers and researchers will have the perfect tool to create demonstrations of smart devices and on which to base classes and laboratory work. Finally, the students will have the opportunity to broaden their area of expertise, gaining more knowledge and experience, thus becoming more competitive in the professional and IoT world. Hopefully, students will also be able to develop ideas on their own, designing more personalized projects, fully integrated to the INSA curriculum.

Our team is initiating this project with the long-term perspectives that futures students will maintain this network, provide documentation and easy access to the LoRa and IoT community. In the end, the ideal scenario would imply a complete integration of our services into the INSA course program to offer personalized and varied projects to students. Moreover, this long-term project will promote an image of INSA as an innovative, creative, and attractive school. It will also provide students skills and knowledge that will make them competitive and qualified for careers in the domain of the Internet of Things.

2. Specifications

The specifications we followed to complete this project are to deploy a LoRa network over the INSA campus, provide documentation to easily access this network, test it, foresee both security and energy constraints and provide a demonstration of an application using our network.

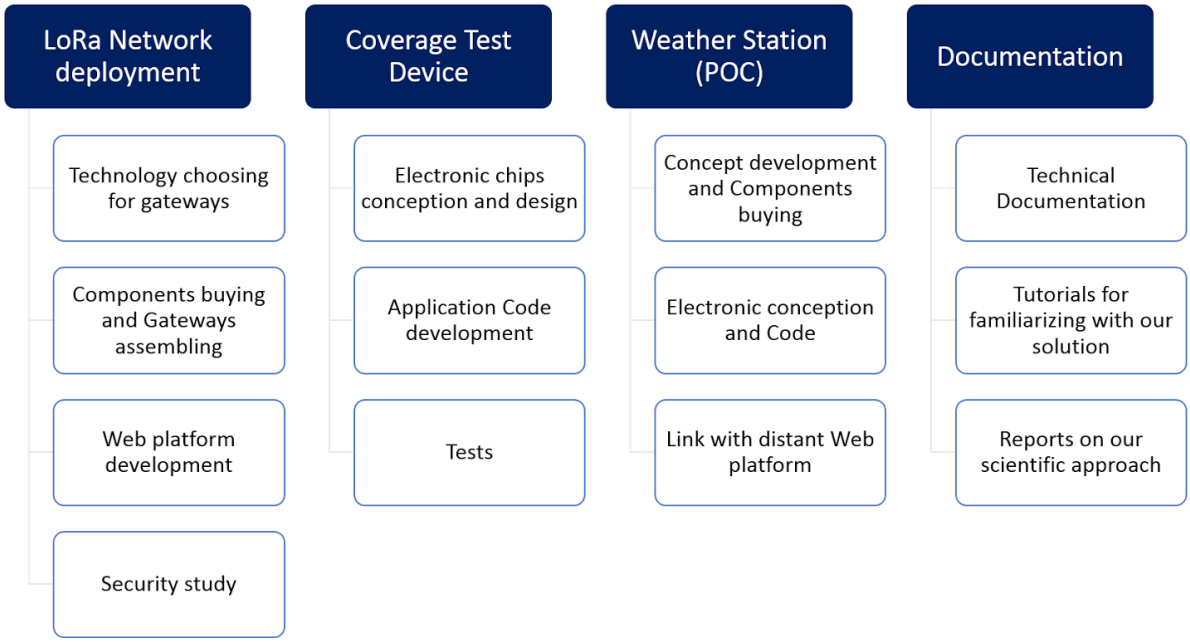


Figure 1.1: General goals of LoRa Project

As shown on the figure above, we separated our project into four main parts. The LoRa network deployment includes the study of the coverage and technological choices, as well as the acquisition and assembly of the devices, and the development of the Web platform. A short state-of-the-art on security problems was also written, to study which features were offered by LoRa, what kind of attacks our network could be subjected to, with hypotheses and proposed solutions. The second part concerns tests and it includes the design of electronic chips with the code development to perform tests and validate the effective communication between the nodes and the gateways. The third part is our proof of concept, a weather station with several sensors, which allows us to demonstrate the correct functioning of our service. The last part concerns documentation; it includes technical documentation, tutorials and all the reports we did on our scientific approach.

3. Temporal organization

We scheduled the different parts of the project from a temporal perspective by making a Gantt Diagram showing our main activities (see *Figure* below).

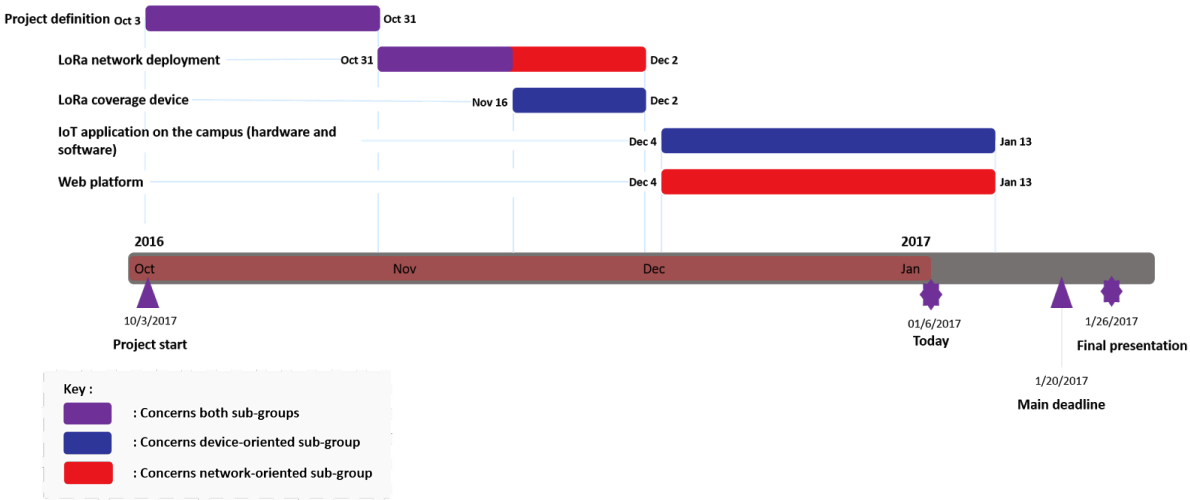


Figure 1.2: Initial Gantt diagram (October 2016)

As shown in the previous figure, we started by working together on deploying the LoRa network, because we were all interested in the subject and it was a good opportunity to learn how to work together and bond as a group. Then we split into two sub-teams: an Embedded Systems team, which created hardware devices and ensured they communicated with the LoRa Network and TTN, and a Web team which developed the web platform and connected it with TTN.

Of course, this Gantt diagram was an estimation, during the project itself we had to adapt to several practical issues, in particular regarding equipment delivery (we will detail this point in the following section), which made us fall behind schedule on several of our activities.

We had a more precise follow-up on the different tasks of our project by using SCRUM methodology and the dedicated IceScrum software. By using these tools we were able to split the work between all members of the group, organize our time into sprints in which we scheduled several “stories”: the main activities we had to complete during the sprint.

The main activities we accomplished during the four sprints are listed below:

Sprint 1	Sprint 2	Sprint 3	Sprint 4
[Lora network Deployment] - First reflexion on the network	[Documentation] - SOA	[Network Coverage Test device] - Second PCB Conception, Printing and Testing	Documentation
[Lora network Deployment] - Buying HW devices	[Network Coverage Test device] - Deploying the software environment	[Lora network Deployment] - Work on Gateways	[Web] - Platform improvement
[Project management] - Choosing the final application of our project using LoRa	[Network Coverage Test device] - PCB Conception, Printing and Testing	[Web] - Platform design	[Network Coverage Test device] - Final Application Development

4. Financial aspects

To reach our goals, we had to buy electronic equipment, typically specific chips for Gateways and several Arduino boards on which we based our nodes. The total cost was about €1380, which is quite high for basic electronic devices. However, to put that cost into perspective, only half of it was actually used for the LoRa Network itself. The other half was dedicated to application nodes; therefore this material can be reused for different projects in the future and it should be considered as a long-term investment.

The real cost of the LoRa network was about €700, but this cost can be divided into four parts, as we just bought the same equipment dedicated to one Gateway conception four times. Although a single gateway would have been enough, we chose to add three more not only in order to maximize the network's performances, but also to use the triangularization functionalities provided by LoRa, and if possible to create a malicious gateway that tries to pose as a legitimate one.

A single gateway costs approximately €180, which is quite high. However, this price is justified by the quality of the equipment provided (in particular a chip to plug into a Raspberry Pi computer which constitutes a gateway on its own), the services provided by it and also by the opportunity to join a proactive IoT community: The Things Network, which could help if needed so.

The biggest problem we encountered, related to our material purchases, was not linked with cost considerations. However, it was very difficult for us to convince INSA’s administration to buy the electronic chips needed to create Gateways. In fact, these components are only offered by a supplier linked to TTN, so it was not possible to find them among INSA’s regular suppliers. With the help of our tutors, we finally convinced the administration and were able to order the components, which were delivered to us only at the end of December.

II. LoRa network deployment

1. Propagation study

During the design study of our service, we decided to perform a study of the coverage of our antennas in order to have a more precise idea of their performances and to know where to install them on the INSA campus.

We decided to use ICS software, as we had already used it last year during Networking and Telecommunications labs supervised by Prof Alexandre Boyer. We encountered some issues during the parameter choice phase, so we asked Mr Boyer, and with his recommendations, we finally obtained the following results.

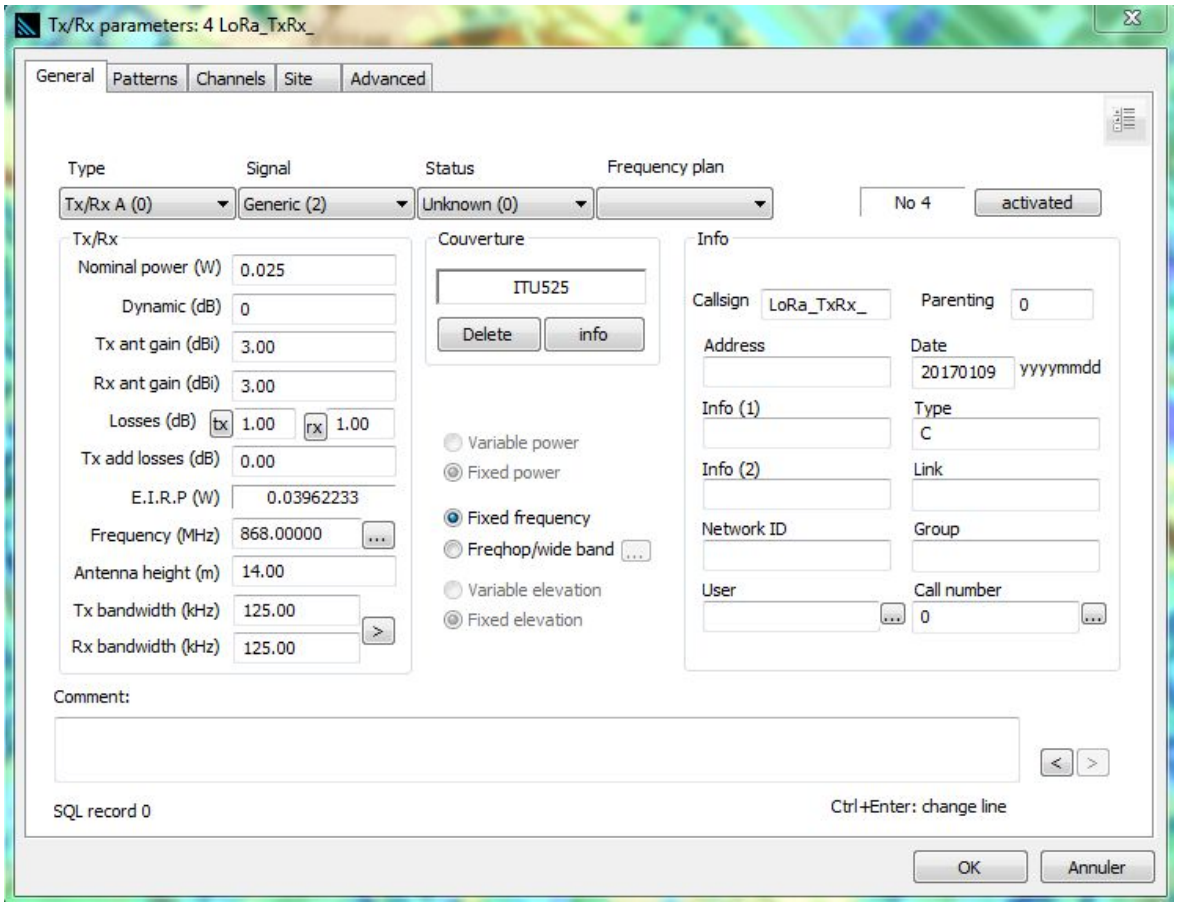


Figure 2.1: Example of chosen parameters for an antenna located on the GP building

The minimum threshold is 11 dBµV/m: if the electric field on a point is less than 11 dBµV/m, then it is outside coverage and the field value is not displayed. Thus, as can be seen on the scale below, the colored zones represent areas where the signal is received (from blue to brown, dark blue being just above the threshold).

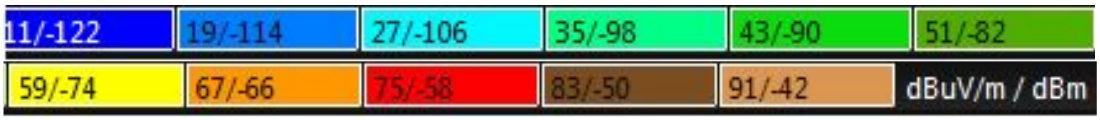


Figure 2.2: Electric field scale

The first picture represents our results with only one antenna, located on top of the GEI building. As we can see, most of the zone is covered, but the strength of the signal is not very high in the blue zones. The next pictures represent our results with four antennas, located on the roofs of the GP, GC, R7 and RU buildings, providing a better coverage over the INSA campus. However, this dense configuration is not really necessary since the coverage is already managed quite adequately with one antenna.

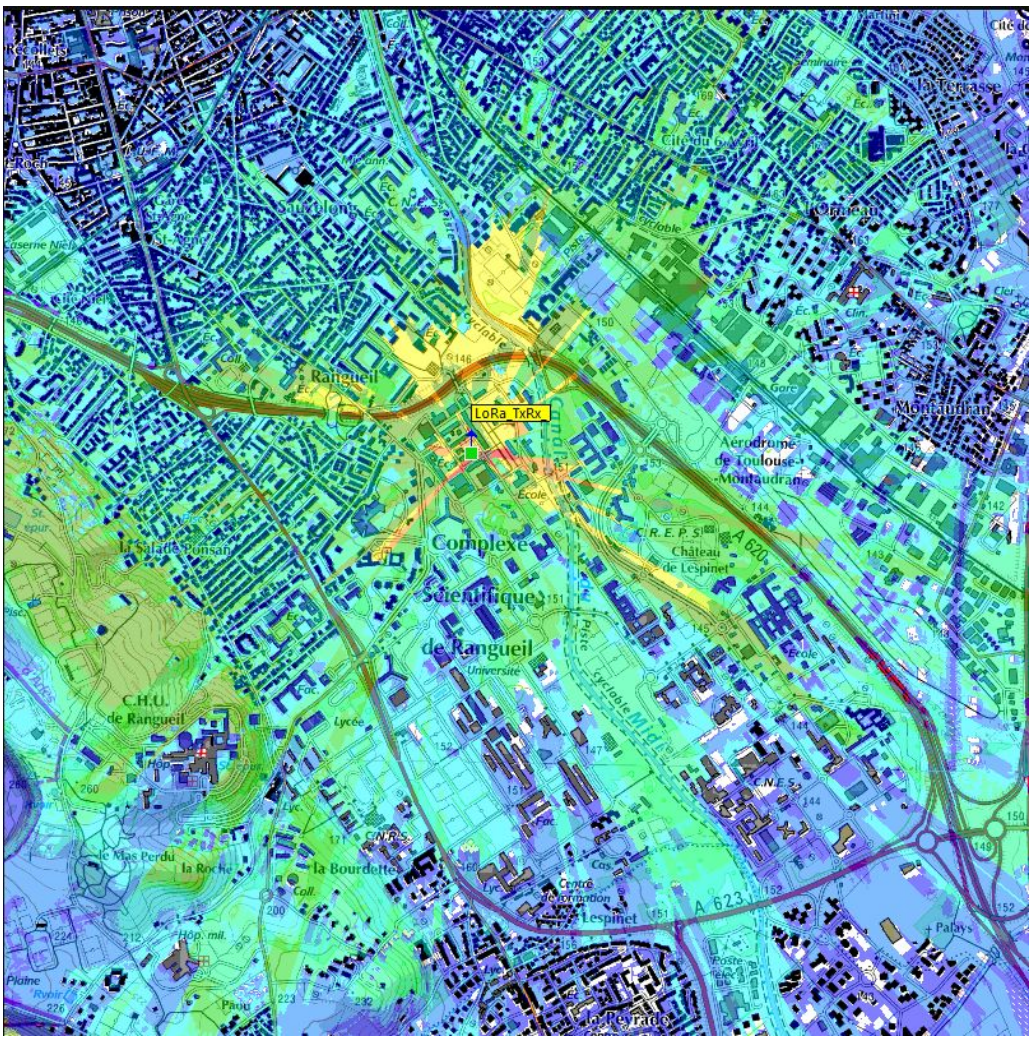


Figure 2.3: Propagation study with 1 antenna

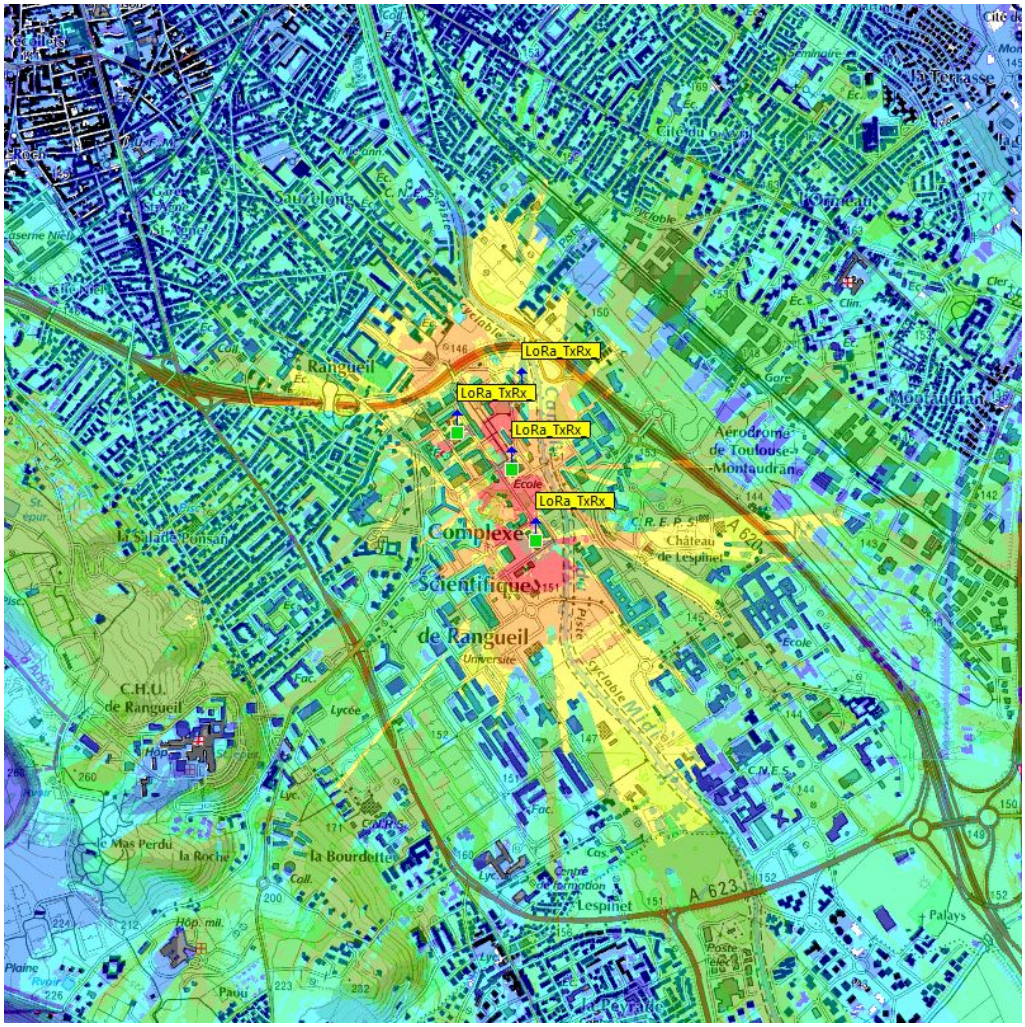


Figure 2.4: Propagation study with 4 antennas

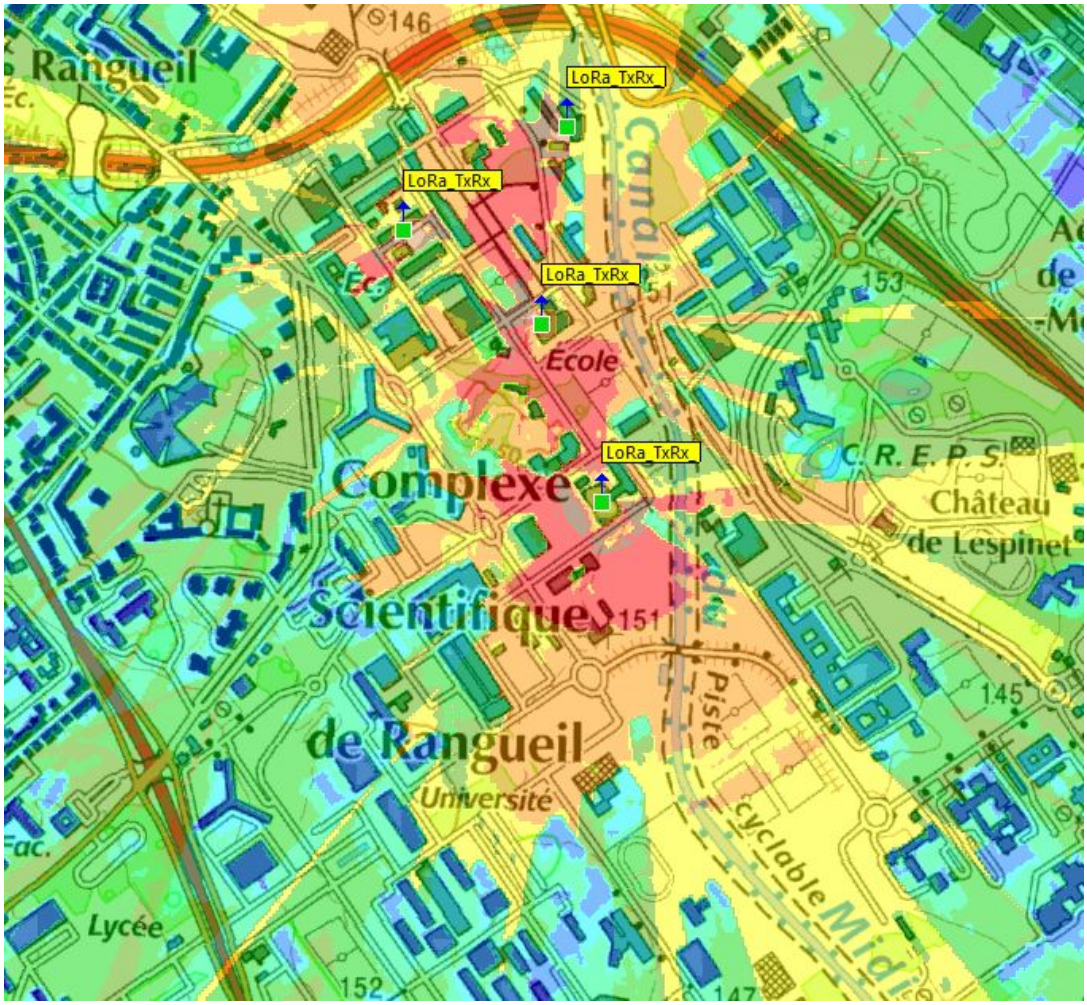


Figure 2.5: Zoom on the INSA campus area for the study with 4 antennas

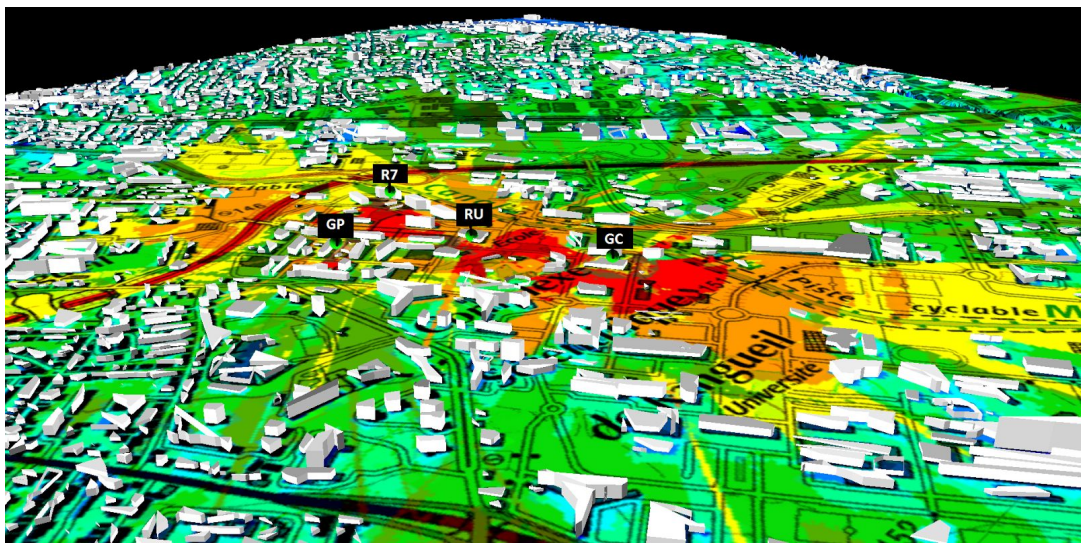


Figure 2.6: 3D view of the propagation study with 4 antennas

2. Choice of gateways

a. TTN

The Things Network (TTN) is an open project which aims to make IoT available to all. The idea is that everyone can create a gateway and share it with people around them. TTN is centralizing these gateways to create a huge shared network. It provides both infrastructure (servers) and a great community with a lot of documentation. It is for these reasons that we decided to base our work on this project. Thanks to the different guides available on their Wiki, we were able to build our gateway and connect it to the existing network quite easily . Once a node is configured to work with TTN, it can use any of the registered gateways, anywhere in the world.

b. Hardware choice

On the Wiki page there are several methods to create a LoRa gateway. Some are cheaper but are not fully LoRAWAN compliant. We decided to use the IMST iC880a board which supports multi-channel communications and works easily with a Raspberry Pi and was the easiest to build. Only 7 wires are needed: it is almost plug and play.

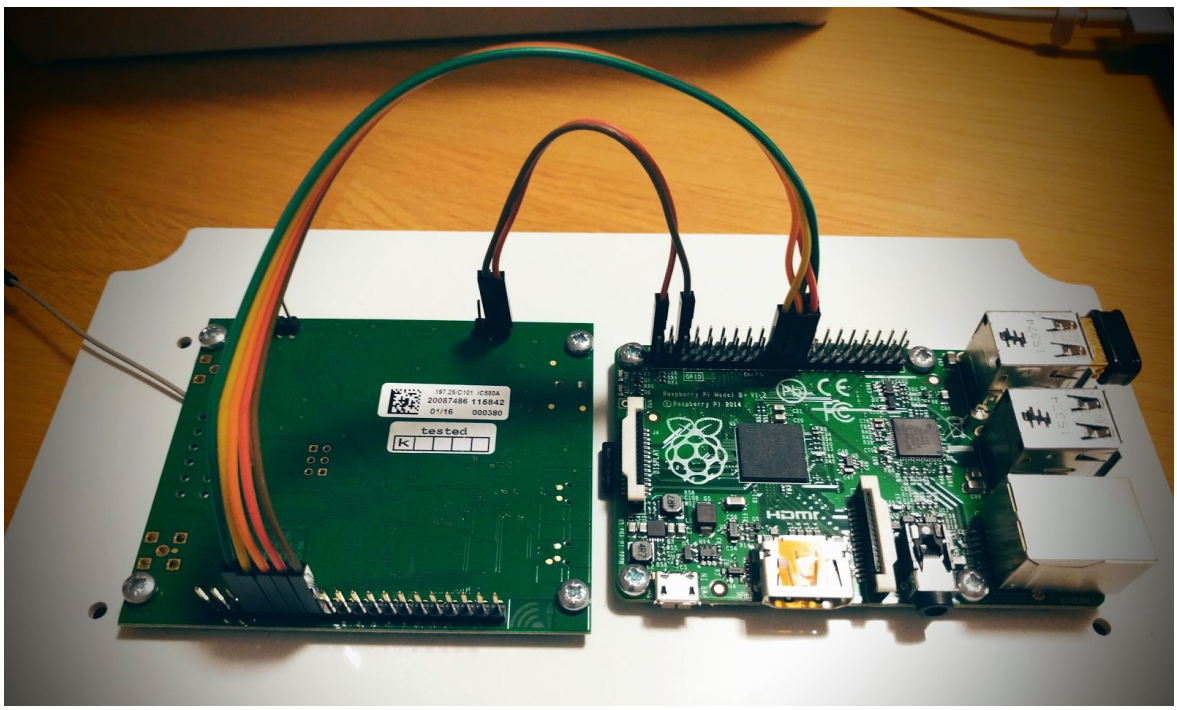


Figure 2.7: Picture of IMST iC880a on the left, Raspberry Pi on the right. Picture taken from TTN wiki.

In addition to being easy to install, the iC880a LoRa board is also open source (which means it can be used to design a fully integrated gateway) and the Raspberry Pi is a very well-known device with a huge community and documentation.

3. Network deployment

a. Logistics

Three problems needed to be resolved:

- gateways needed to be placed in high locations
- gateways needed an Internet connexion, at least to reach TTN servers
- gateways needed to be connected to the electricity distribution network

To solve the first problem, we decided to place the gateways on the buildings' roofs. To do so we asked Mr. Gaffier, executive secretary, who gave us an access to GEI's roof.

For the second problem we contacted the CSN (Centre des Services Numériques) to create special accesses for the Raspberry Pi boards. These accesses allow the boards to contact TTN servers (only) on port 1700 (input and output).

For the second and last problems, we have an Internet as well as an electric outlet directly on the roof, so we just had to pass wires through a wall, in order to place the gateway outside.

Practically, we bought and made an ergonomic package for the gateway, which we directly attached to a pole. A picture of the infrastructure is presented below.

Figure 2.8: Picture of our LoRa gateway installed on the roof

b. Tests

The first tests were directly done in the Fablab, using the Wi-fi module of the Raspberry to connect the gateway to the Internet. In this configuration, the LoRa range was very short, and it barely reached INSA residences. We first thought that it was due to the fact that the gateway was inside that caused this problem, but as soon as we had access to an Ethernet port dedicated to the gateway, the range considerably increased. We concluded that it was the Wi-fi module which was interfering with the LoRa transceiver.

Once we first placed the gateway on the roof, we made a qualitative coverage test of our network. Using our network coverage test device (which conception and functioning are presented in a following part), we periodically sent predefined LoRa frames, and checked on TTN the SNR of the received frames. We then moved our test device (without checking the GPS location using the module) over INSA campus. We checked that the whole campus was covered, from GC to Rangueil, even if some places were more hardly reachable (the SNR was lower). Typically, the opposite side of the GC presented a very low SNR.

We then made a more concrete study by driving a car around the campus to test the limits of our gateway coverage. We validated that the whole campus was covered, then tested the true limits of the network. Going to the north, we get data until Saouzelong (signal going through a lot of building walls before reaching the gateway), which represents 1.4 kms. On the worst case (between Saouzelong and St Michel), the SNR was of -8.2. We then took several observations by going to the Pech-David hill, which overlooks the South of Toulouse. There, the reception was better (the SNR was 6), and the distance to INSA was around 1.6 kms. Those results are relevant with what the theoretical study made before said (with an increasement of performances on Pech-David, and a quick).

An information about speed: a node can emit and be detected by the gateway as long as it is slower than 30 kms per hour (at this speed it is still possible to emit).

It is important to notice that the gateway tends to heat up after a few days connected and communicating. As we made our test in january, this was not a problem for us. However, it is important to check the gateway in summer, as it is exposed in direct sunlight, and eventually to find a way to cool it down.

Another important point is that, in the current state, the gateway’s configuration related to DNS is erased when it is reset.

4. Security

As is often the case in fast-growing domains, security issues are often set aside or neglected. However, security problems are crucial in the IoT domain which is quite new and raises new security issues. There is a huge amount of data concerning localization, health, and any critical or personal data in the IoT and smart systems are far from being flawless.

Thus, we spent a great deal of time studying the security concerns of our project. We have written two documents on the subject and have made some hypotheses and examined different attacks and solutions. For instance, we made the assumption that the gateways were not physically reachable by people with malicious intentions since they were located on roofs. We also thought about the attacks which aim to compromise the nodes by adding, deleting or modifying them. We wanted to add functionalities to our system to detect both these kinds of attacks as well as suspicious data. During the SOA (Services Oriented Architecture) labs, we tried to implement a SOAP solution to validate the accuracy of the received data: if the values are out of range compared to the limits we have chosen (for example, a temperature of 80°C or a location outside the INSA campus), users are notified that there may be an integrity/functioning issue. We also had the idea to create boxes that will raise an alarm if opened. Unfortunately, we did not have time to concretely deal with other security attacks or propose other solutions. A whole project could be developed on this subject. On the other hand, we used our knowledge for the choice of technology: we decided to use the LoRaWan protocol since it provides two-level encryption and two methods for the nodes to join the network: OTAA (Over The Air Activation) and ABP (Activation By Personalization). We chose OTAA since it is more secure than ABP. The subject is studied in more detail in our documentation. Moreover, we have taken into account the best practices recommended for our use of LoRa. Overall, our study on security enabled us to gain more knowledge and to heighten our awareness of security issues in the connected object networks.

5. Improvements

It could be useful to optimize the gateway hardware, especially by removing wires and using an adequate impermeable package. For instance the open-hardware shield provided by ch2i could be ordered (available on https://github.com/ch2i/iC880A-Raspberry-PI).

Again, the design of a more efficient antenna could be an axis of improvement.

Finally, it is important to check the heat resistance of the gateways during summer days in Toulouse. Even in winter the boards tend to slightly heat up after a few days being connected.

III. Device creation

1. Components choice

In order to develop our applicative nodes, we had to implement some functionalities:

- LoRa communication, following the LoRaWAN standard
- computing
- location
- measurement of physical quantities

Regarding LoRa communication:

The choice of the LoRaWAN MAC protocol was justified by its integrated security processes. To implement it, two LoRa transceivers were used.

- **sx1272**

The first was the eval-board sx1272RF1-ABS from Semtech. This choice was strongly recommended by our project tutors. This transceiver only implements the LoRa communication layer. Thus we had to import a LoRaWAN stack in microcontroller. This step was laborious because of a lack of place and the non-existence of a dedicated library for our microcontroller. To communicate with it, the SPI protocol was used. This module cost €74.78. The energy supply had to be 3.3V and is provided by the microcontroller.

Figure 3.1: sx1272 Transceiver

Supply voltage: 3.3V

sx1272	Condition	Datasheet			Measurement	
		Typical	Maximal	Unit	Typical	Unit
Sleep mode		0.1	1	µA		
Idle mode	RC oscillator enable	1.5		µA		
Standby mode	Crystal oscillator enable	1.4	1.6	mA		
Synthesizer mode	FRRx	4.5		mA		
Receive mode	LnaBoost Off LnaBoost On	10.5 11.2		mA		
Transmit mode with impedance matching	RFOP = +20 dBm on PA_BOOST RFOP = +17 dBmon PA_BOOST RFOP = +13 dBm on RFO pin RFOP = +7 dBm on RFO pin	125 90 28 18		mA		

- **RN2483**

The second LoRA transceiver we used was a Microchip RN2483 based on a sx1276 Semtech chip. This has an integrated microcontroller (PIC) which contains a LoRaWan stack. We can also communicate with it thanks to a serial port (UART). This module cost €13.24. Thus is was cheaper and more user-friendly than the one suggested. As a surface-mounted component it can be used directly on the board which allowed us to optimize space.
The energy supply had to be 3.3V and is provided by the microcontroller.

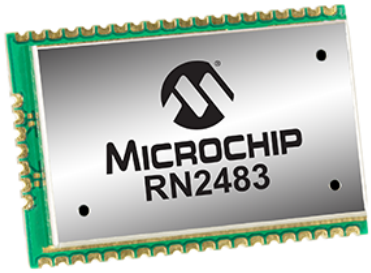


Figure 3.2: RN2483 Transceiver

Supply voltage: 3.3V

RN2783	Condition			Datasheet		Measurement	
	Band (MHz)	TX Power Setting	Output Power (dBm)	Typical	Unit	Typical	Unit
Sleep mode				9.9	µA	Not tested	Not tested
Idle mode				2.8	mA	3.35	mA
Receive mode				14.2	mA	Not tested	Not tested
Transmit mode with impedance matching	969	-3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	-4.0 -2.9 -1.9 -1.7 -0.6 0.4 1.4 2.5 3.6 4.7 5.8 6.9 8.1 9.3 10.4 11.6 12.5 13.5 14.1	17.3 18.0 18.7 20.2 21.1 22.3 23.5 24.7 26.1 27.5 28.8 30.0 31.2 32.4 33.7 35.1 36.5 38.0 38.9	mA	Tests: 969 MHz (band): TW Power Setting: 5 Measure: 23.2 mA	mA
	433	-3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	-3.5 -2.3 -1.3 -2.3 -1.2 -0.1 1.0 2.1 3.2 4.3 4.3 5.4 6.5 7.6 8.8 9.9 10.9 11.9 12.9 13.6	14.7 15.1 15.6 15.8 16.4 17.0 17.7 18.5 19.4 20.3 21.4 22.3 23.3 24.5 25.8 27.3 28.8 30.7 32.9			

We studied the power consumption of the RN module. We did not have time to test the Semtech due to the numerous issues we met with this module. Regarding the Microchip, we observed power consumption close to what was indicated on the datasheet. As we used a high-level library to manage this device, the TX power Setting was already defined as 5, so we chose not to change it at first. We noticed that the idle mode current was a bit higher than what was indicated on the datasheet, which led to an increase of energy consumption by the device: This might be caused by our daughter board, or by the fact that the RN module was once submitted to high temperatures, which might have damaged it slightly .

We did not implement the downlink in our LoRa Network, so we could not test the RX consumption of the RN. Regarding the low-power consumption, we did not have time to test it, but we noticed that when the RN wakes up, its consumption jumps to 4.8 mA for several seconds. Thus, we should use this low-power mode with precaution to be sure that it actually saves power (for instance, by not waking up the module at too high a frequency which would make it consume more energy).

Regarding the computing process:

We decided to use easy-to-use Arduino boards and especially the Arduino Micro. This has a compact design (48mmx18mm for 13g), relative high computing and storage capacities (based on a ATmega32U4 microprocessor, computing at 16MHz, with 32kB of Flash memory, 2.5kB of SRAM and 1kB of EEPROM) and is very interactive (20 digital pins with 7 for PWM, 12 analog input pins, programmable thanks to a micro-usb connection). To compare, the most famous Arduino Uno was similar in terms of computing capacities (based on a ATmega3228 microprocessor, computing at 16MHz, with 32kB of Flash memory, 2kB of SRAM and 1kB of EEPROM), but was designed on a bigger board and with less connectivity (14 digital pins with 6 for PWM, 6 analogue input pins, programmable to thanks to a mini-usb connection). This choice was not crucial because a lot of libraries have been developed regarding the ATmega3228 architecture and thus have to be adapted to our architecture. Moreover, a sleep mode is integrated in this microcontroller which allows efficient energy management .

The energy supply had to be between 7V and 12V, but the effective voltage is 5V. This board can be use to provide supply to other modules.

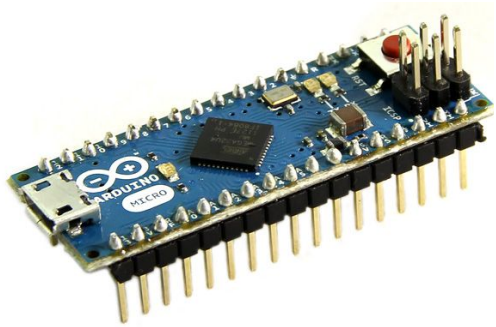


Figure 3.3: Arduino Micro board

Supply voltage: 5V

Arduino Micro (ATmega32u4)	Datasheet		Measurement	
	Typical	Unit	Typical	Unit
Sleep mode	6	µA	11.4	mA
Standby mode	14	mA	20.2	mA

We conducted several tests on the Arduino Micro's consumption. Though the consumption varied depending on the amount of code put on the board, we tested this consumption in different situations:

- Arduino being in standby mode: 20.2 mA
  We observed an important discrepancy, which we will try to analyse in the next paragraph.
- Arduino being in low-power mode (according to the low-power library we used): 11.4mA

The difference between theory and practice here is very important (actual consumption being almost 2000 times higher than the theoretical one). The main reason for this behaviour is that many of the Arduino peripherals are not turned off by the library (for instance, timers, UARTS, SPI are not turned off). What is more, GPIOs levels remain at their initial value (if they were at '1' before entering low-power mode, this tension level will be maintained throughout sleep period, which will increase the consumption.

After talking with other project groups, we found that some of them had the same problem using the same library, but different Arduino boards. Therefore it seems that the problem is not a hardware one but a software one linked to the library we used. Although this library is recommended by many users, it presents many issues. First, it is not well-designed (for instance function descriptions are not very clear and the code is very difficult to read), and the function sets are very limited, so it is impossible to turn off properly timers and other functions. We tried to use other libraries, but what we found on the Internet was often based on the first library we used, and the other versions we found did not have a strong user base, so they were not considered as reliable. What is more, it was impossible to modify the current library, as it used peripheral level information which is contained in Arduino.h (which is the basic Arduino library we could not change).

To conclude, the Arduino boards using ATmega32U4 as microcontroler are more a pedagogical tool than an industrial tool; they might not be reliable, so the performances can not be that good. As we had very little time to work on it, we could not find another solution to improve energy consumption.

Regarding location:

Although our network provides enough coverage, we did not have the time to develop a location program through triangulation. Thus we chose to use a GPS module specifically an Adafruit Breakout V3 because it was easy to use and reliable. This communicated with the microcontroller through a serial port. As all this module is not low energy, thus its use is only dedicated to local and debug application. A simple GPS module (without a routed board) could be used as a surface mounted component and thus allow space optimization.

The energy supply had to be between 3.3V and 5V and is provided by the microcontroller.

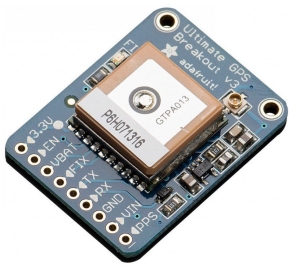


Figure 3.4: GPS Ultimate Breakout

Supply voltage: 3.3V

Breakout v3	Datasheet		Measurement	
	Typical	Unit	Typical	Unit
Backup mode	7	µA		
Acquisition mode	25	mA		
Tracking mode	20	mA		

We chose not to make consumption tests on the GPS to check the datasheet for two reasons. First, this module would not be used in an autonomous device for long periods. Then, we had no time left to make this study.

Regarding measurement of physical quantities:

We chose several different sensors and actuators for the development of our application devices. Again, the choice of an easy-to-use apparatus was made, but we also took into account their respective costs.

Temperature and humidity sensor (AM2303):

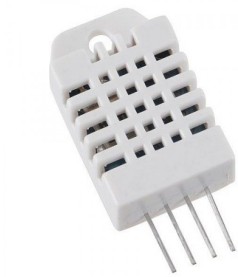


Figure 3.5: AM2303 temperature and humidity sensor

Supply voltage: 3.3V

AM2303	Datasheet		Measurement	
	Typical	Unit	Typical	Unit
Sleep mode	15	µA	25	µA
Measuring mode	500	mA	700	µA

Regarding low-power consumption, the value was given by a digital multimeter with accuracy of 1 or 2 µA, so the 25 µA value we obtained might not be extremely precise.

2. Coverage test device

a. Context

The first application we developed using our LoRa network is a coverage test device which, at the push of a button, sends a LoRa frame containing the current GPS location of the device and the quality of LoRa coverage at that location to the distant Web platform.

Developing such an application allowed us to test the performances of our network, and then to compare them to the theoretical coverage study presented in a previous part of this document. It was also a good way to test our gateway locations, and check if there were any conception errors .

From a technical point of view, this was an opportunity for us to practise many different concepts linked with IoT communications and Embedded Systems design. We also used different technological devices: Arduino, LoRa Transceivers, applicative sensors (GPS module, etc.).

b. Development phase

We worked on two kinds of devices, based on two different LoRa transceivers (one which includes the LoRaWAN functionalities, the other which does not). To be functional, both of our devices embed several components:

- An Arduino Micro, which computes all the data and requiring communication with the gateways
- A LoRa transceiver, which transmits the data
- An Adafruit Breakout V3 GPS
- Several resistors for adapting voltage

i. First version for the board: Semtech

For this first device version, we based our board on a Semtech transceiver (*ref*. sx1272). This kind of transceiver does not include a LoRa Stack to implement LoRaWAN functionalities. As The Things Network only works with the LoRaWAN protocol, we had to add the LoRa Stack to our main program stored in the Arduino Micro board. The final version of the created board is presented in the following *figure*.

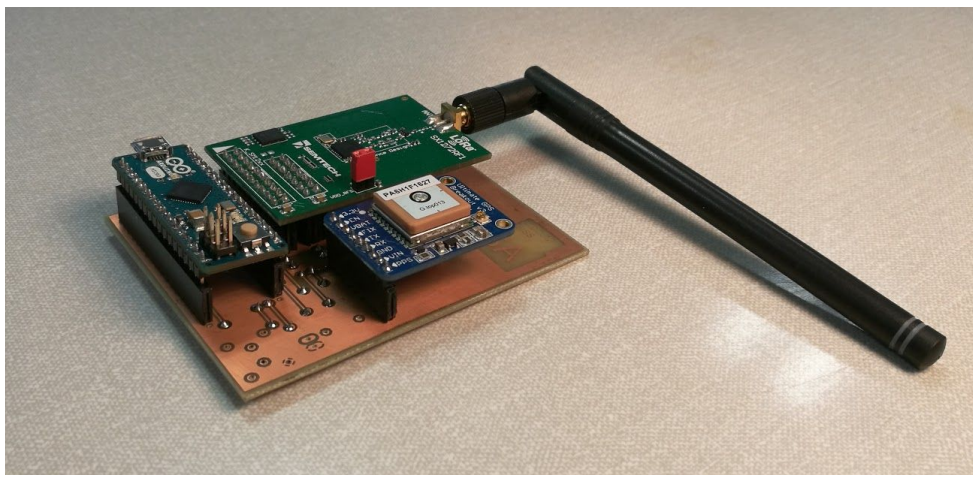
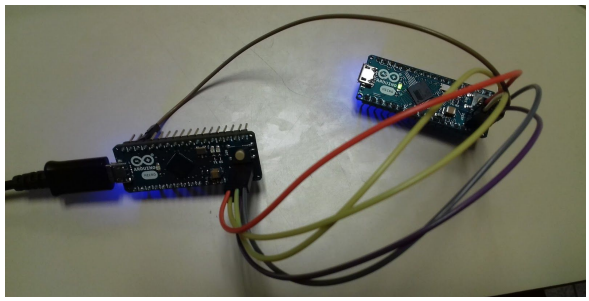


Figure 3.6: Coverage Test Device, Semtech version

The main problem we met with this board is related to the size of our application program. We had to store a library, called LMIC (LoRa MAC in C), containing the LoRaWAN stack directly in our Arduino Micro, so it took up most of the available memory. This caused an overloading of Arduino's memory, which caused bootloader deletion. This phenomenon is commonly referred to as an Arduino being"bricked".

To put it simply, the bootloader is the code that allows us to program the Arduino board just by plugging it into a computer through a USB port. When the bootloader is deleted, we cannot program the board through the USB anymore. Therefore, we have to use a dedicated programmer, a hardware device that will be connected both to the PC and the Arduino, and which will do the same job as the bootloader by transforming the user's code into commands the board can understand.

To solve this problem, we first had to learn how to rewrite ("burn") the bootloader on a bricked Arduino. To do so, we had to use a second Arduino as a programmer: we loaded a specific program on the "Debug" Arduino, then connected it to the bricked Arduino in a very specific way (cf. *Figure* below).



Debug Arduino Pin	Bricked Arduino Pin
ICSP1	ICSP1
ICSP2	ICSP2
ICSP3	ICSP3
ICSP4	ICSP4
Pin 10	ICSP5
ICSP6	ICSP6

Figure 3.7: Wiring for bootloader burning (Arduino Micro)

Then, by executing the program contained in the Debug Arduino through several commands, it was possible to fix the bricked Arduino by adding the bootloader.

To avoid this problem, we tried to use the Semtech with another kind of Arduino Board (Arduino Pro Mini), which possesses slightly different technical specifications. If the board is not bricked anymore, it is still impossible to make the Semtech emit. In the end we concluded that there was a compatibility issue between Arduino Micro boards and the Semtech sx1272, which was never mentioned or referenced in the available online documentation.

ii. Second version for the board: Microchip

The second version of this board was based on a Microchip RN2483 transceiver. This transceiver includes a LoRaWan stack, which enables it to communicate easily with the gateways and TTN. Our Arduino program simply sends commands to this device thanks to a dedicated library.

Most of the job was to develop a daughter-board to be embedded on the mother-board including the GPS and Arduino Micro. It is composed of the Microchip RN2483, a SMA connector allowing the board to host an antenna for improved long range communication and a suited pinout for a good interconnection with this other board. On this device, we can actually put two SMA connectors (and thus two antennas), each one corresponding to two separate frequencies (434 and 868 MHz), allowing us to simultaneously communicate on both of them. Nevertheless, we only use the higher frequency on which gateways operate. Our first attempt to build the board was a failure. We started to solder the components on it, and it turned out that one layer was printed backwards: the circuits were thus not aligned with our components.We then printed the board the other way around; we wired it to our Arduino and GPS. On TTN, we created an application on our gateway, and registered our newly created device on it. On the Arduino, we registered the application and launched the test. We managed to connect to the gateway and send data to TTN, visible on the online application, proving that our board was completely functional.

Below is a photograph of the final board and its pinout, as well as a screenshot of the data received on the TTN test application.

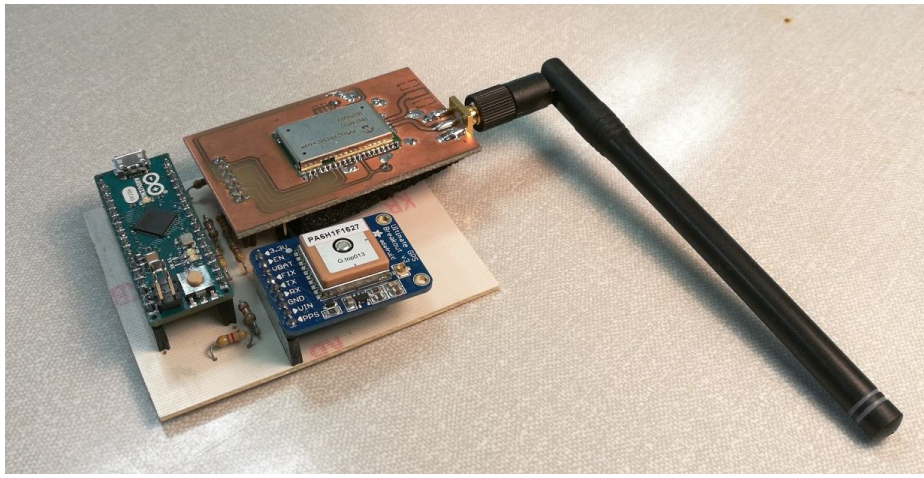


Figure 3.8: Coverage Test Device, Microchip version

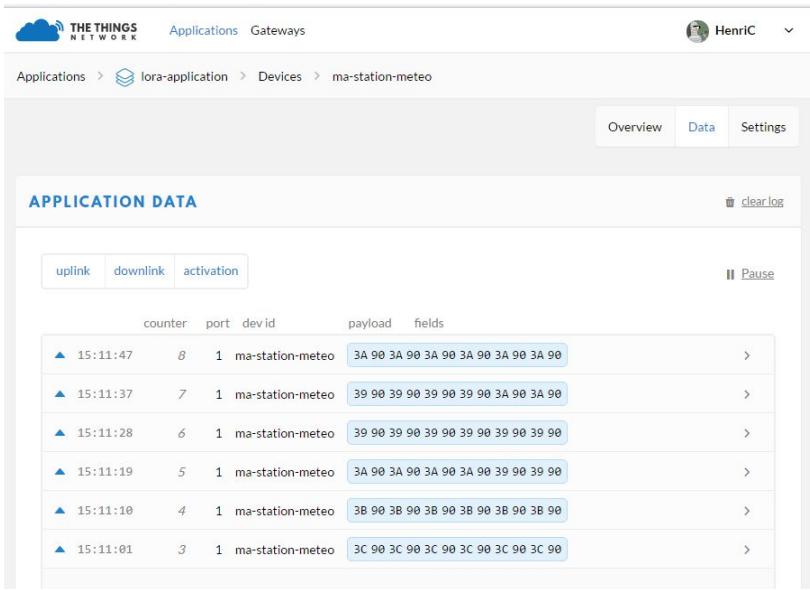


Figure 3.9: TTN Test application: Data received

Regarding the application code, the main goal was to periodically send a GPS location to the server. There is no need for the node to provide information about the quality of the LoRa signal, this function will be handled by the gateway.

In practice, while the GPS is not fixed (when it has not found a satellite and is not able to give latitude/longitude/altitude information), the code will wait for a valid signal. Then, it will periodically collect GPS data, which means saving the latitude, longitude and altitude information given in local variables (2 floats and 1 integer). Latitude and longitude will be transformed: the incoming format being an angle with the equator or the Greenwich meridian and a character indicating the direction (‘N’ for North, etc.), we process this data to generate an angle between 0 and 360° for the longitude (0 being the International Date Line), and between 0 and 180° for the latitude (0 being the south pole). Floats being stocked on 4 bytes and integers on 2 bytes, we will then send a LoRa frame whose payload will be 10 bytes long. To create such a frame, we create a byte array, in which we will copy byte by byte the GPS data in the following format:

Lon(1)	Lon(2)	Lon(3)	Lon(4)	Lat(1)	Lat(2)	Lat(3)	Lat(4)	Alt(1)	Alt(2)

Figure 3.10: GPS frame architecture

Regarding energy consumption, we do not take it into consideration for this application, because the coverage test device is not expected to function for a very long time. The main use of this device will be to accompany the user over short distances, and short amounts of time. In the end, this device will be connected through USB to a battery or directly to a computer, so we will not consider the low-power options for this device.

It is possible to optimize our Program by coding each piece of GPS information on a certain amount of bits. By only keeping 4 decimals for latitude and longitude, it is possible to code this information on only one byte. The accuracy of the GPS location is still within 7 meters for latitude, and 15 meters for longitude. The altitude will still be coded on 2 bytes. Using such a technique, it is possible to lower power consumption, and thus the emission time.

Another possibility would be to enter a low-power mode between each GPS measure. However this functionality does not seem particularly useful, as the GPS module is the one which consumes the most power, and turning it off would imply an important starting time (and consumption) to make it contact satellites again .

Finally, we built an adapted package in wood, with a plexiglass top allowing a user to check if the GPS “fix” is done, and to be visually relevant.

c. Test phase and results analysis

The tests of the devices were first dedicated to validate the GPS information returned. When it was done and the coverage test device was completely functional, we tested it at the same time we made our coverage tests, which are presented in a previous part of this document.

d. Improvements

We could choose a more adapted antenna and make it autonomous in terms of energy (add a battery). A final optimization could be to design an integrated system with a microprocessor ATmega32U4, a RN2483 transceiver and a PA6H GPS module, based on a two-layer board and SMD component (as 0402 package), and thus improve compactness.

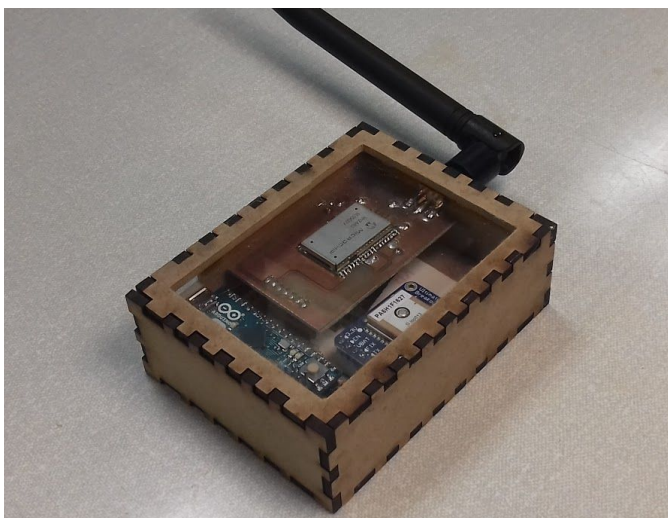


Figure 3.11: Coverage Test Device, Final packaging

3. Weather station

a. Context

We decided to create a connected weather station to provide an example of an application using our network. The data collected by the sensors integrated in the weather station are processed by the gateway and the server, then displayed on the web platform. In addition to providing our proof of concept, this weather station can also be used as a showcase for the school. The first version of the weather station is shown below.

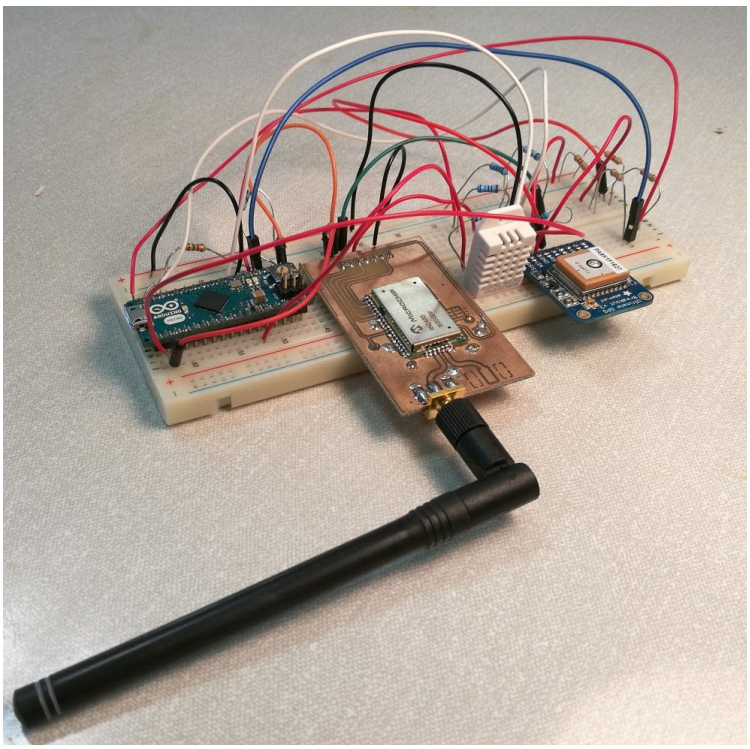


Figure 3.12: Weather station (breadboard)

b. Development phase

We chose to make the weather station as simple as possible. By the end of the semester we were under time constraints, so we only had a couple of weeks to develop a first version of the weather station. The hardware part was pretty simple to implement: An Arduino Micro board collects data from a humidity and temperature sensor, and sends them through our LoRa Network by using the RN2483 transceiver. We made the wiring on a breadboard and used this electrical assembly to test our code. We then added a light sensor which returns a value between 0 and 1023 regarding sunshine.

We then developed a PCB for the Weather station (using Eagle), printed it using Fablab's equipment. The final version of this PCB, completely functional, is shown on the picture above.

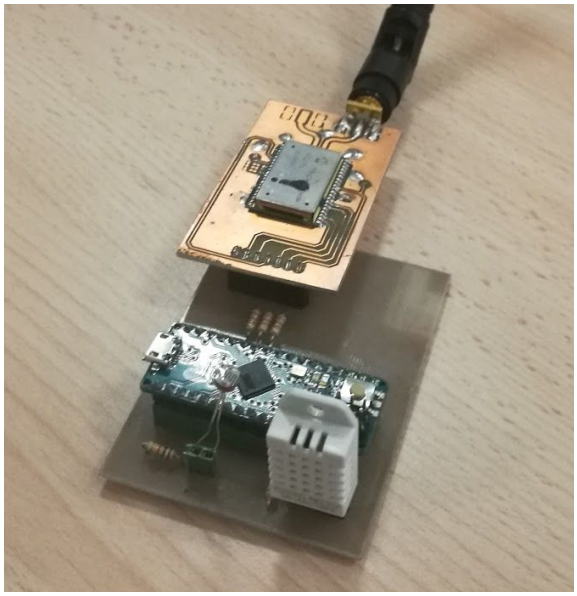


Figure 3.13: Weather station (PCB)

As a weather station is expected to stay in position for months or even years, we paid a lot of attention to the energy consumption of our device. To do that, we implemented a low-power library in our code, which turns most of the components (except for the Real-Time Clock) of the Arduino off (UARTS, CPU, etc.). What is more, as our temperature and humidity sensor is a passive one, we do not turn it off during low-power phases. Finally, we tried to it wake up only when genuinely justified, so we only take readings once an hour, and send a LoRa frame four times a day.

To detail the application code itself, we collect temperature and humidity data once an hour (the system is in low-power mode until then). Recorded temperature and humidity are in the float format. Then we convert the obtained data to have it fit into a one byte format (short int). To do so:

- **Temperature measure:**

We chose the available range of the sensor : [-50°C,77°C]. In this way, if we shift this range by adding 50 degrees, we get a range of [0, 127]. What is more, if we multiply the panned measure by 2, we obtain a value between 0 and 254, so we can use all of the 256 available values of a byte. Thus we are able to store a temperature measure on eight bits, with an accuracy of ½ degree. Finally, we just have to cast the result in a short integer to get a value that we can send.

In the end, we used this formula to convert the float into a short int.

```
float temperature = XX,XXXX;
short converted_temperature = (short)((temperature +50)*2);
```

- **Humidity measure:**

For this measure, we used the same technique as that for the temperature, but we did not have to add an offset, as the degree of humidity is contained between 0% and 100%. Even if it is not optimized, having half percent accuracy for the humidity is enough for our application and allows us to cast the humidity on a single byte (we are using 200 values to code this result, so we still have 56 free values, which is not well optimized).

In the end, we used this formula to convert the float into a short int.

```
float humidity = XX,XXXX;
short converted_humidity  = (short)(humidity*2);
```

Our device wakes up each hour, takes and converts a measurement as explained above and fills a byte array with it. The byte array has the following form:

Hum1	Tmp1	Hum2	Tmp2	Hum3	Tmp3	Hum4	Tmp4	Hum5	Tmp5	Hum6	Tmp6

Figure 3.14: Temperature and Humidity frame architecture

When the array is filled after six hours, it is sent in a 12 bytes LoRa frame to the web platform. This operation is thus done four times a day. From the web platform perspective, when a frame is received, it is necessary to split each byte composing the frame, and select which bytes are linked with temperature and which are linked with humidity. Then we convert each of the bytes into a float again (by dividing the result by 2 and subtracting 50 for the temperature). Thus, we can get the exact transmitted value on the web platform.

- **Light measure:**

This measure is made every hour. The value between 0 and 1024 is divided by four to allow a storage on one byte (a value being between 0 and 255). Each six hours, a six-values frame (6 bytes) is sent through LoRa.

Lum1	Lum2	Lum3	Lum4	Lum5	Lum6

Figure 3.15: Light frame architecture

c. Tests phase and results analysis

We tested this weather station by collecting six temperature and humidity values in a row, which were then converted and sent to the network through the transceiver.Then, we checked on the TTN website to verify that the received values were correct. We managed to retrieve the original data (six humidity values and six temperature values) from the raw hexadecimal code, as predicted. We could not test the final software under real conditions, because the test would be too long (six hours), but our test code simply reduces the waiting time between two measurements, which is sufficient to prove the functionality of the device.

We also checked the potential validity of the temperature and humidity measured . When indoors, we retrieved a temperature of about 22°C and humidity levels of 40%. These values seem to be correct with regard to test conditions.

Eventually, we tested the device's energy consumption with the final code implemented. During the setup phase, the board consumes 40 mA. Then, in nominal functioning phase (when data is processed and sent to the LoRa network), the average consumption is about 38.78 mA.

d. Improvements

We could build an adapted package (wood and Poly(methyl 2-methylpropenoate)) and choose the most adapted antenna. Moreover we could make it autonomous in terms of energy (add a battery and design an energy recovery system such as solar cells)). A final optimization could be to design an integrated system with a microprocessor ATmega32U4, a RN2483 transceiver and sensors, based on a two-layer board and SMD components (as 0402 package), and thus improve compactness.We also need to compare the acquired temperature and humidity values with the ones retrieved by a commercial

weather station, in order to check the reliability of our data. We could also add other sensors to the weather station including:

- rain gauge: to measure rainfall amounts
- noise sensor: to monitor noise pollution

http://fr.rs-online.com/web/p/microphones-a-condenseur/7800721/

- gas and dust sensor: to analyse air and determine pollutants and toxic gases
- atmospheric pressure sensor: to predict weather

http://fr.rs-online.com/web/p/capteurs-de-pression-absolue/8212422/

- anemometer: to monitor wind power

http://fr.rs-online.com/web/p/capteurs-a-effet-hall/1811457/ and magnet
http://www.bricomusique.com/tuto-comment-fabriquer-un-anemometre/

- weathervane : to monitor wind direction

http://fr.rs-online.com/web/p/codeurs-rotatifs/7967806/

4. Ideas

Some software improvements should be made.

Regarding libraries, we had to optimize and complete them:
We had to take a closer look at the GPS library in order to take precise control of all intimed stuff, and thus optimize energy management and reduce the amount of storage space needed.
The Arduino library is not really energy efficient. Again a closer look at the associated library could increase results, but a more relevant way would be to change the hardware platform and not use an Arduino development board.

For instance, EFM32 or STM32 only consumes a few microamps (typically 7µA) in sleep mode. The use of STM32 will allow an easier way to implement sx1272 thanks to an optimal lmic library (LoRawan stack) provided by IBM.

Regarding the RN2483 module, we used a library dedicated to it, but it is not optimal and does not implement a downlink or provided non-desired functions. Thus we had to complete and arrange it in order to increase communication and decrease the size.

The use of a downlink -a real need in industry- will allow us to check emissions and thus implement a more robust way of communication. Again a downlink allows us to use actuator or to reprogram device without having to get them.

Another way to improve this project is to develop other applications; we have several suggestions:

- car park monitor to improve traffic campus (some ideas include: monitoring of places, monitoring of car flows, use of global system)
- light and temperature readings in classrooms in order to improve energy savings by informing the concierge/security post where light bulbs are not switched off and windows are not closed
- open door detection system to prevent security officers from making unneeded trips
- delocated game to test downlink and reactivity. We suggest a two-player game (such as Tic-Tac-Toe) where play structures are not proximate. This application is not energy efficient and is proposed as a visual proof-of-concept.
- gas detector to detect leaks.
- pollution detection station (noise, gas, dust, and other pollutants) to get information about campus environment

As a pedagogical tool, these network and platform are designed to enable the creation of all imaginable applications.

IV. Web platform

1. Functions

a. Overview

There is an existing platform on TTN which provides data nodes and basic functions to manipulate them. However it is not very intuitive and needs some work to get it working.

We wanted a very easy-to-use tool which is intuitive for users. Our platform is an abstraction of TTN and adds a lot of functionalities:

- data format and value check through a SOAP service
- data storage
- data charts
- public / private nodes management
- map visualisation
- REST API to retrieve data

In order to facilitate the public/private node management we decided to use a single account on TTN. When a user adds a node on our platform, it will automatically be added to our TTN account. Then we manage the restriction by adding a “restriction” attribute to the node in our database. If it is set to public, the node data will be accessible to everyone. If it is set to private, only the user who added it will be allowed to access the data on our website.

As this verification is done by our server (and not the TTN server), it means that administrators of the platform can access all the nodes’ data, whether it is public or private, simply by browsing the database. This could be considered as a confidentiality problem, however:

- The initial goal of this entire project is to provide public data so people can create rich applications with it. Private nodes are just a bonus for development projects.
- If a user really wants privacy he/she can encrypt data. However by doing that the chart display functions will be lost.

b. Account creation

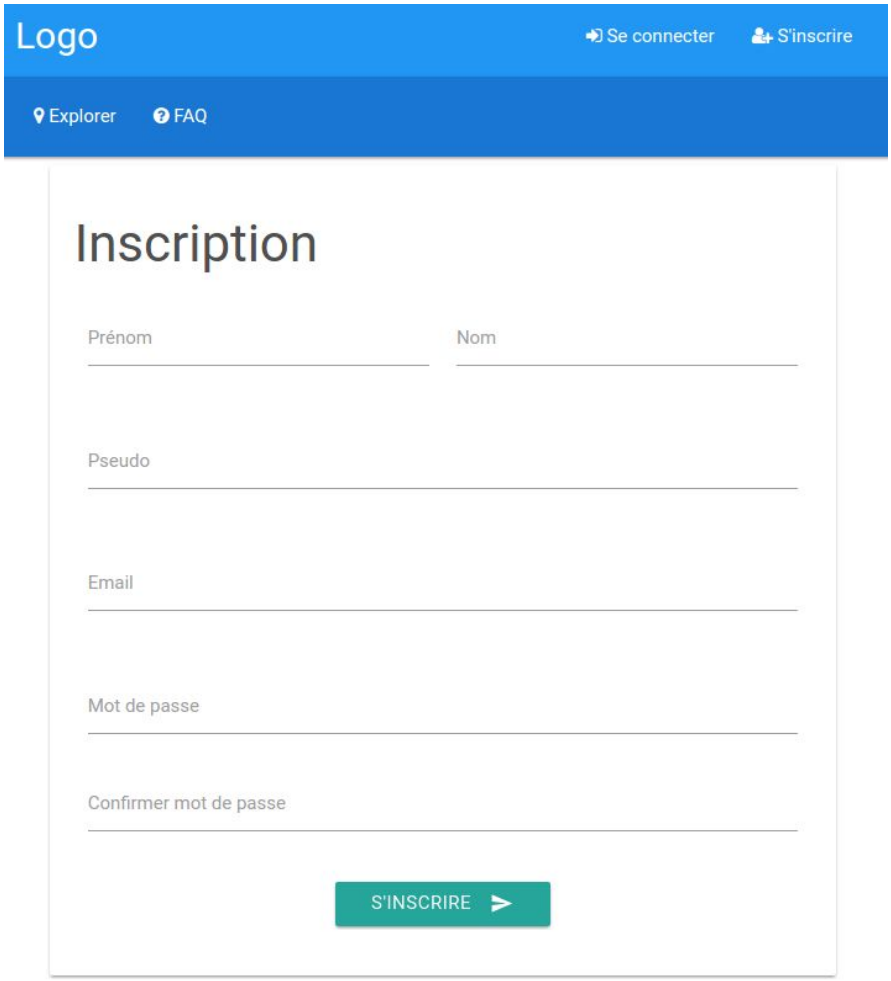


Figure 4.1: Web Platform - Sign in screen

As you can see on the above screenshot, you do not need any specific information to register on our platform. It is accessible to everyone.

Today you do not need an account to see public data but it can evolve with new requirements. Having an account allows you to add nodes to the platform.

c. Adding a node

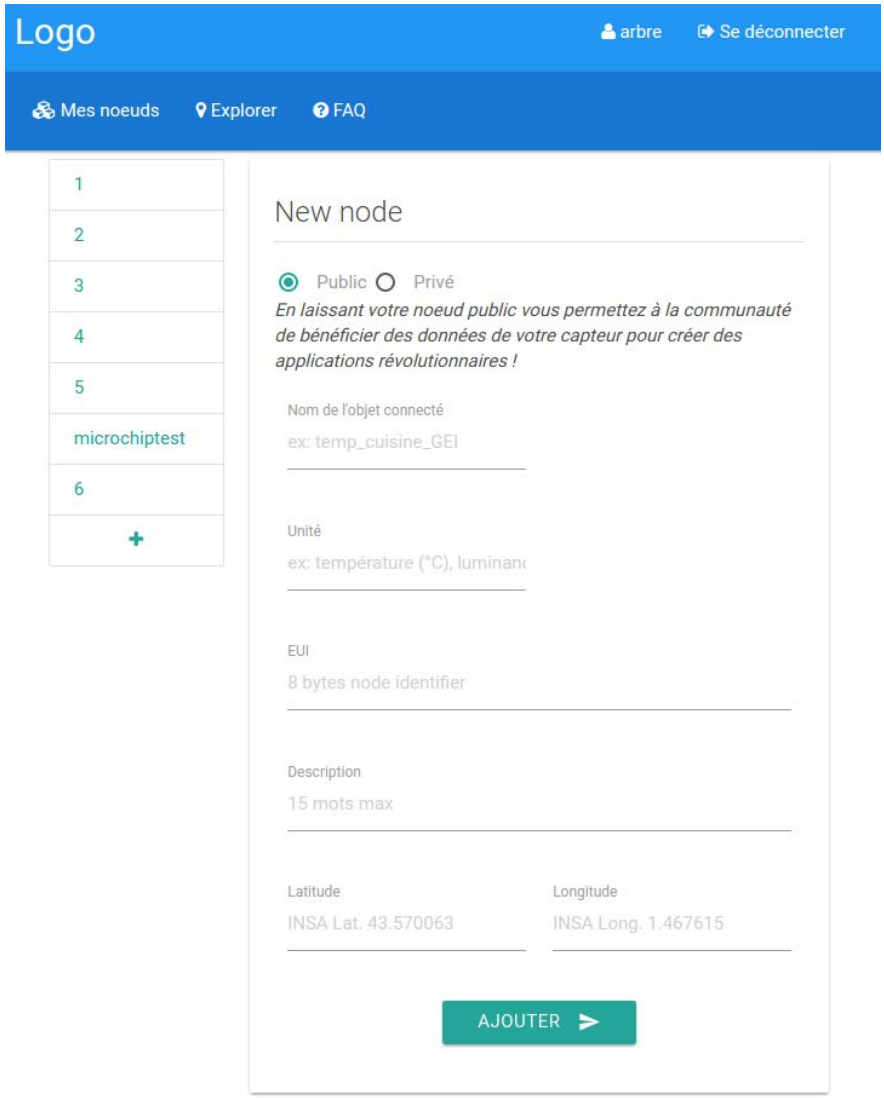


Figure 4.2: Web Platform - Node adding

Let's imagine that the GEI Department wants to monitor the temperature in each classroom of the building to adapt the heating strategy. The GEI buys simple LoRa connected sensors and places them in the department rooms.

Then they add it to the platform with the correct information. The only specific thing to know is the EUI number (Extended Unique Identifier) of each device. It is provided by the constructor. And here we are: you can see the information of your nodes on the platform or even retrieve it via our REST API.

The nodes can be public or private, which means that you need to be authenticated when you try to retrieve data of a private node.

d. Managing nodes

Once you added a node, you can modify its description and its position but you cannot change its EUI, its name and its unit.

e. Displaying data

For the moment displaying data in charts is still under development. But here is what it will look like:

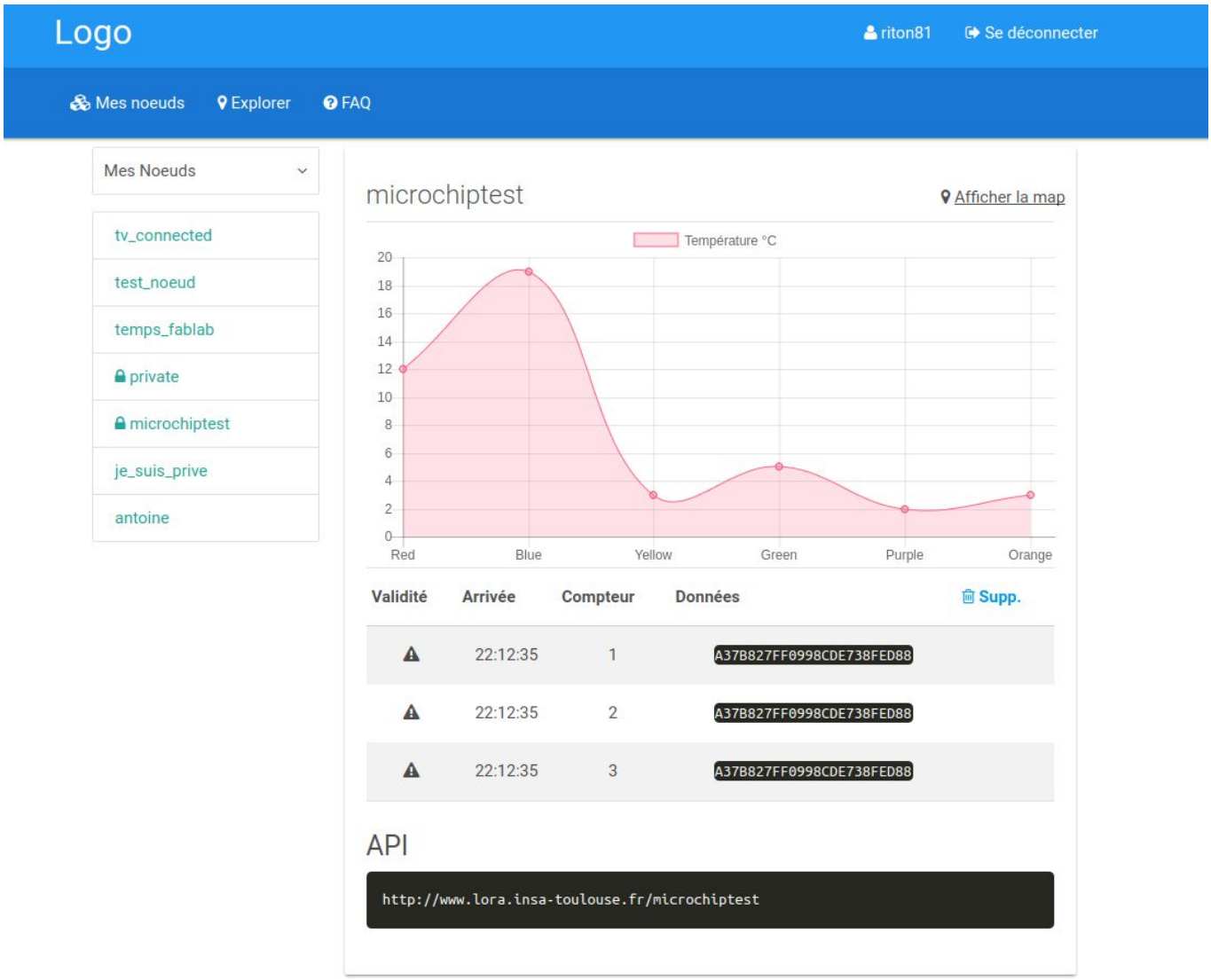


Figure 4.3: Data displaying (platform)

2. Architecture

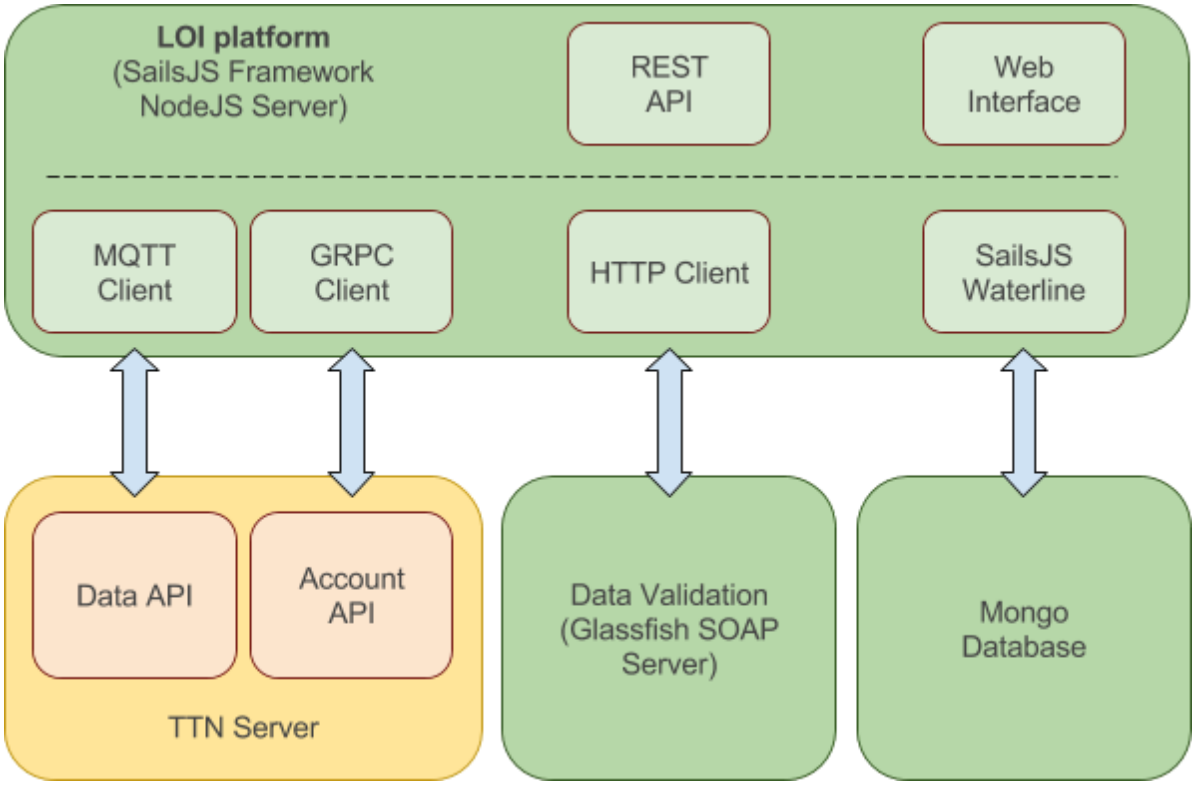


Figure 4.4: Platform architecture

On this chart we can see all the different stakeholders of our web platform.

3. Documentation

All of our web platform is open source. Sources are available in the our public repository: https://bitbucket.org/lorawebapp/lorawebapp/

For the installation users just follow the README.md. It will guide them through the process of starting their platform.

Conclusion

This project has enabled us to undertake a real, challenging project, which allowed us to collaborate using all of our knowledge and skills. We have been able to concretely put into practice what we had learned on smart systems, and take into account their specific issues, such as security and energy considerations. It has also been a good experience to work in a multidisciplinary group and to share our knowledge with one another. We have also realized the time it takes to clearly define such a project and to handle the logistics related to it. We have implemented the first steps of this project, with the long-term perspectives to maintain, reinforce and extend it. We strongly hope that other students will continue our work. Another objective would be to integrate our services into the INSA courses in a way that could provide the opportunity to students, as future engineers,to develop personalized and varied projects, in order to encourage innovation, creativity and responsibility.

We would like to thank all those who helped us during this project, especially:
Mr. Eric Alata and Mrs. Daniela Dragomirescu, our tutors,
Mr. Joseph Shea, who corrected our official papers,
Mr. Etienne Sicard, our photographer,
MM. Alexandre Boyer, Sebastien Di Mercurio, José Martin, Fabien Nougarolles and Lucien Senaneuch,
Mr. Yoni Pomares,
And all of those who attended our presentations and those we have forgotten.